Data Structures, Estimation and Optimization with R

Fan Wang

2024-01-01

Contents

Preface

This is a work-in-progress website consisting of R panel data and optimization examples for Statistics/Econometrics/Economic Analysis. Materials gathered from various projects in which R code is used. Files are from the R4Econ repository. This is not a R package, but a list of examples in PDF/HTML/Rmd formats. REconTools is a package that can be installed with tools used in projects involving R.

Bullet points show which base R, tidyverse or other functions/commands are used to achieve various objectives. An effort is made to use only base R (R Core Team, 2019) and tidyverse (Wickham, 2019) packages whenever possible to reduce dependencies. The goal of this repository is to make it easier to find/re-use codes produced for various projects. Some functions also rely on or correspond to functions from REconTools (Wang, 2020).

From other repositories: for research support toolboxes, see matlab toolbox, r toolbox, and python toolbox; for code examples, see matlab examples, stata examples, r examples, python examples, and latex examples; for packaging example, see pkgtestr for developing r packages; for teaching, see intro mathematics for economists, and intro statistics for undergraduates. see here for all of fan's public repositories.

The site is built using Bookdown (Xie, 2020).

Please contact FanWangEcon for issues or problems.

Chapter 1

Array, Matrix, Dataframe

1.1 List

1.1.1 Lists

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

- r list tutorial
- r vector vs list
- r initialize empty multiple element list
- r name rows and columns of 2 dimensional list
- r row and colum names of list
- list dimnames
- r named list to string

1.1.1.1 Iteratively Build Up a List of Strings

Build up a list of strings, where the strings share common components. Iteratre over lists to generate variations in elements of the string list.

```
# common string components
st_base_name <- 'snwx_v_planner_docdense'
st_base_middle <- 'b1_xi0_manna_88'
# numeric values to loop over
ar_st_beta_val <- c('bt60', 'bt70', 'bt80', 'bt90')
ar_st_edu_type <- c('e1lm2', 'e2hm2')

# initialize string list
ls_snm <- vector(mode = "list", length = length(ar_st_beta_val)*length(ar_st_edu_type))

# generate list
it_ctr = 0
for (st_beta_val in ar_st_beta_val) {
  for (st_edu_type in ar_st_edu_type) {
    it_ctr = it_ctr + 1
    # snm_file_name <- 'snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt90'
    snm_file_name <- paste(st_base_name, st_edu_type, st_base_middle, st_beta_val, sep ='_')
    ls_snm[it_ctr] <- snm_file_name
  }
}

# print
for (snm in ls_snm) {
```

```
  print(snm)
}
```

```
## [1] "snwx_v_planner_docdense_e1lm2_b1_xi0_manna_88_bt60"
## [1] "snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt60"
## [1] "snwx_v_planner_docdense_e1lm2_b1_xi0_manna_88_bt70"
## [1] "snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt70"
## [1] "snwx_v_planner_docdense_e1lm2_b1_xi0_manna_88_bt80"
## [1] "snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt80"
## [1] "snwx_v_planner_docdense_e1lm2_b1_xi0_manna_88_bt90"
## [1] "snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt90"
```

```
# if string in string
grepl('snwx_v_planner', snm)
```

```
## [1] TRUE
```

1.1.1.2 Named List of Matrixes

Save a list of matrixes. Retrieve Element of that list via loop.

```
# Define an array to loop over
ar_fl_mean <- c(10, 20, 30)

# store restuls in named list
ls_mt_res = vector(mode = "list", length = length(ar_fl_mean))
ar_st_names <- paste0('mean', ar_fl_mean)
names(ls_mt_res) <- ar_st_names

# Loop and generat a list of dataframes
for (it_fl_mean in seq(1, length(ar_fl_mean))) {
  fl_mean = ar_fl_mean[it_fl_mean]

  # dataframe
  set.seed(it_fl_mean)
  tb_combine <- as_tibble(
    matrix(rnorm(4,mean=fl_mean,sd=1), nrow=2, ncol=3)
    ) %>%
    rowid_to_column(var = "id") %>%
    rename_all(~c(c('id','var1','varb','vartheta')))

  ls_mt_res[[it_fl_mean]] = tb_combine
}

# Retrieve elements
print(ls_mt_res[[1]])
print(ls_mt_res$mean10)
print(ls_mt_res[['mean10']])

# Print via Loop
for (it_fl_mean in seq(1, length(ar_fl_mean))) {
  tb_combine = ls_mt_res[[it_fl_mean]]
  print(tb_combine)
}
```

1.1.1.3 One Dimensional Named List

1. define list
2. slice list
3. print r named list as a single line string

- R Unlist named list into one string with preserving list names

```
# Define Lists
ls_num <- list(1,2,3)
ls_str <- list('1','2','3')
ls_num_str <- list(1,2,'3')

# Named Lists
ar_st_names <- c('e1','e2','e3')
ls_num_str_named <- ls_num_str
names(ls_num_str_named) <- ar_st_names

# Add Element to Named List
ls_num_str_named$e4 <- 'this is added'
```

Initiate an empty list and add to it

```
# Initiate List
ls_abc <- vector(mode = "list", length = 0)
# Add Named Elements to List Sequentially
ls_abc$a = 1
ls_abc$b = 2
ls_abc$c = 'abc\'s third element'
# Get all Names Added to List
ar_st_list_names <- names(ls_abc)
# Print list in a loop
print(ls_abc)
```

```
## $a
## [1] 1
##
## $b
## [1] 2
##
## $c
## [1] "abc's third element"
```

```
for (it_list_ele_ctr in seq(1,length(ar_st_list_names))) {
  st_list_ele_name <- ar_st_list_names[it_list_ele_ctr]
  st_list_ele_val <- ls_abc[it_list_ele_ctr]
  print(paste0(st_list_ele_name,'=',st_list_ele_val))
}
```

```
## [1] "a=1"
## [1] "b=2"
## [1] "c=abc's third element"
```

1.1.1.4 Named List Print Function

- r print input as string
- r print parameter code as string
- How to convert variable (object) name into String

The function below ffi_lst2str is also a function in REconTools: ff_sup_lst2str.

```
# list to String printing function
ffi_lst2str <- function(ls_list, st_desc, bl_print=TRUE) {

  # string desc
  if(missing(st_desc)){
    st_desc <- deparse(substitute(ls_list))
  }
```

```
  # create string
  st_string_from_list = paste0(paste0(st_desc, ':'),
                               paste(names(ls_list), ls_list, sep="=", collapse=";" ))

  if (bl_print){
    print(st_string_from_list)
  }
}

# print full
ffi_lst2str(ls_num)
```

```
## [1] "ls_num:=1;=2;=3"
```

```
ffi_lst2str(ls_str)
```

```
## [1] "ls_str:=1;=2;=3"
```

```
ffi_lst2str(ls_num_str)
```

```
## [1] "ls_num_str:=1;=2;=3"
```

```
ffi_lst2str(ls_num_str_named)
```

```
## [1] "ls_num_str_named:e1=1;e2=2;e3=3;e4=this is added"
```

```
# print subset
ffi_lst2str(ls_num[2:3])
```

```
## [1] "ls_num[2:3]:=2;=3"
```

```
ffi_lst2str(ls_str[2:3])
```

```
## [1] "ls_str[2:3]:=2;=3"
```

```
ffi_lst2str(ls_num_str[2:4])
```

```
## [1] "ls_num_str[2:4]:=2;=3;=NULL"
```

```
ffi_lst2str(ls_num_str_named[c('e2','e3','e4')])
```

```
## [1] "ls_num_str_named[c(\"e2\", \"e3\", \"e4\")]:e2=2;e3=3;e4=this is added"
```

1.1.1.5 Two Dimensional Unnamed List

Generate a multiple dimensional list:

1. Initiate with an N element empty list
2. Reshape list to M by Q
3. Fill list elements
4. Get list element by row and column number

List allows for different data types to be stored together.

Note that element specific names in named list are not preserved when the list is reshaped to be two dimensional. Two dimensional list, however, could have row and column names.

```
# Dimensions
it_M <- 2
it_Q <- 3
it_N <- it_M*it_Q

# Initiate an Empty MxQ=N element list
ls_2d_flat <- vector(mode = "list", length = it_N)
ls_2d <- ls_2d_flat
```

```
# Named flat
ls_2d_flat_named <- ls_2d_flat
names(ls_2d_flat_named) <- paste0('e',seq(1,it_N))
ls_2d_named <- ls_2d_flat_named

# Reshape
dim(ls_2d) <- c(it_M, it_Q)
# named 2d list can not carry 1d name after reshape
dim(ls_2d_named) <- c(it_M, it_Q)
```

Print Various objects generated above, print list flattened.

```
# display
ffi_lst2str(ls_2d_flat_named)
```

```
## [1] "ls_2d_flat_named:e1=NULL;e2=NULL;e3=NULL;e4=NULL;e5=NULL;e6=NULL"
```

```
# print(ls_2d_flat_named)
ffi_lst2str(ls_2d_named)
```

```
## [1] "ls_2d_named:=NULL;=NULL;=NULL;=NULL;=NULL;=NULL"
```

```
print(ls_2d_named)
```

```
##      [,1] [,2] [,3]
## [1,] NULL NULL NULL
## [2,] NULL NULL NULL
```

Select element from list:

```
# Select Values, double bracket to select from 2dim list
print('ls_2d[[1,2]]')
```

```
## [1] "ls_2d[[1,2]]"
```

```
print(ls_2d[[1,2]])
```

```
## NULL
```

1.1.1.6 Define Two Dimensional Named LIst

For naming two dimensional lists, *rowname* and *colname* does not work. Rather, we need to use *dimnames*. Note that in addition to dimnames, we can continue to have element specific names. Both can co-exist. But note that the element specific names are not preserved after dimension transform, so need to be redefined afterwards.

How to select an element of a two dimensional list:

1. row and column names: *dimnames*, *ls_2d_flat_named[['row2','col2']]*
2. named elements: *names*, *ls_2d_flat_named[['e5']]*
3. select by index: *index*, *ls_2d_flat_named[[5]]*
4. converted two dimensional named list to tibble/matrix

Neither *dimnames* nor *names* are required, but both can be used to select elements.

```
# Dimensions
it_M <- 3
it_Q <- 4
it_N <- it_M*it_Q

# Initiate an Empty MxQ=N element list
ls_2d_flat_named <- vector(mode = "list", length = it_N)
dim(ls_2d_flat_named) <- c(it_M, it_Q)
```

```r
# Fill with values
for (it_Q_ctr in seq(1,it_Q)) {
  for (it_M_ctr in seq(1,it_M)) {
    # linear index
    ls_2d_flat_named[[it_M_ctr, it_Q_ctr]] <- (it_Q_ctr-1)*it_M+it_M_ctr
  }
}

# Replace row names, note rownames does not work
dimnames(ls_2d_flat_named)[[1]] <- paste0('row',seq(1,it_M))
dimnames(ls_2d_flat_named)[[2]] <- paste0('col',seq(1,it_Q))

# Element Specific Names
names(ls_2d_flat_named) <- paste0('e',seq(1,it_N))

# Convert to Matrix
tb_2d_flat_named <- as_tibble(ls_2d_flat_named) %>% unnest()
mt_2d_flat_named <- as.matrix(tb_2d_flat_named)
```

Print various objects generated above:

```r
# These are not element names, can still name each element
# display
print('ls_2d_flat_named')
```

```
## [1] "ls_2d_flat_named"
```

```r
print(ls_2d_flat_named)
```

```
##      col1 col2 col3 col4
## row1 1    4    7    10
## row2 2    5    8    11
## row3 3    6    9    12
## attr(,"names")
##  [1] "e1"  "e2"  "e3"  "e4"  "e5"  "e6"  "e7"  "e8"  "e9"  "e10" "e11" "e12"
```

```r
print('tb_2d_flat_named')
```

```
## [1] "tb_2d_flat_named"
```

```r
print(tb_2d_flat_named)
print('mt_2d_flat_named')
```

```
## [1] "mt_2d_flat_named"
```

```r
print(mt_2d_flat_named)
```

```
##      col1 col2 col3 col4
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

Select elements from list:

```r
# Select elements with with dimnames
ffi_lst2str(ls_2d_flat_named[['row2','col2']])
```

```
## [1] "ls_2d_flat_named[[\"row2\", \"col2\"]]:=5"
```

```r
# Select elements with element names
ffi_lst2str(ls_2d_flat_named[['e5']])
```

```
## [1] "ls_2d_flat_named[[\"e5\"]]:=5"
```

```
# Select elements with index
ffi_lst2str(ls_2d_flat_named[[5]])
```

```
## [1] "ls_2d_flat_named[[5]]:=5"
```

1.1.1.7 Two-Dimensional Named List for Joint Probability Mass

There are two discrete random variables, generate some random discrete probability mass, name the columns and rows, and then convert to matrix.

```
set.seed(123)

# Generate prob list
it_Q <- 2
it_M <- 2
ls_2d <- vector(mode = "list", length = it_Q*it_M)
dim(ls_2d) <- c(it_Q, it_M)
# Random joint mass
ar_rand <- runif(it_Q*it_M)
ar_rand <- ar_rand/sum(ar_rand)
# Fill with values
it_ctr <- 0
for (it_Q_ctr in seq(1,it_Q)) {
  for (it_M_ctr in seq(1,it_M)) {
    # linear index
    ls_2d[[it_M_ctr, it_Q_ctr]] <- ar_rand[(it_Q_ctr-1)*it_M+it_M_ctr]
  }
}
# Replace row names, note rownames does not work
dimnames(ls_2d)[[1]] <- paste0('E',seq(1,it_M))
dimnames(ls_2d)[[2]] <- paste0('A',seq(1,it_Q))
# rename
ls_prob_joint_E_A <- ls_2d
mt_prob_joint_E_A <- matrix(unlist(ls_prob_joint_E_A), ncol=it_M, byrow=F)
print('ls_prob_joint_E_A')
```

```
## [1] "ls_prob_joint_E_A"
```

```
print(ls_prob_joint_E_A)
```

```
##    A1        A2
## E1 0.1214495 0.1727188
## E2 0.3329164 0.3729152
```

```
print(mt_prob_joint_E_A)
```

```
##           [,1]      [,2]
## [1,] 0.1214495 0.1727188
## [2,] 0.3329164 0.3729152
```

Create conditional probabilities: $F = P(A_1|E_1)$, $B = P(A_1|E_2)$, $C = P(E_1|A_1)$, $D = P(E_1|A_2)$

```
fl_F <- mt_prob_joint_E_A[1,1]/sum(mt_prob_joint_E_A[1,])
fl_B <- mt_prob_joint_E_A[2,1]/sum(mt_prob_joint_E_A[2,])
fl_C <- mt_prob_joint_E_A[1,1]/sum(mt_prob_joint_E_A[,1])
fl_D <- mt_prob_joint_E_A[1,2]/sum(mt_prob_joint_E_A[,2])
print(paste0('fl_F=', fl_F, ',fl_B=',fl_B,',fl_C=',fl_C,',fl_D=',fl_D))
```

```
## [1] "fl_F=0.412857205138471,fl_B=0.471665472604598,fl_C=0.267294503388642,fl_D=0.316546995323062"
```

1.2 Array

1.2.1 Array Basics

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.2.1.1 Sum and Product of Elements in Array

Product of Elements in Array.

```
ar_a <- c(1,2,3)
ar_b <- c(1,2,3,4)
prod(ar_a)
```

```
## [1] 6
```

```
prod(ar_b)
```

```
## [1] 24
```

1.2.1.2 Multidimesional Arrays

```

ar_a <- c(1,2,3)
ar_b <- c(1,2,3/1,2,3)
rep(0, length(ar_a))
```

1.2.1.2.1 Repeat one Number by the Size of an Array

```
## [1] 0 0 0
```

1.2.1.2.2 Generate 2 Dimensional Array First, we will generate an NaN matrix with 3 rows and 3 columnes.

```
mt_x <- array(NA, dim=c(3, 3))
dim(mt_x)
```

```
## [1] 3 3
```

```
print(mt_x)
```

```
##      [,1] [,2] [,3]
## [1,]   NA   NA   NA
## [2,]   NA   NA   NA
## [3,]   NA   NA   NA
```

Second, we will generate a matrix with 2 rows and four columns.

```
mt_x <- array(c(1, 1.5, 0, 2, 0, 4, 0, 3), dim=c(2, 4))
dim(mt_x)
```

```
## [1] 2 4
```

```
print(mt_x)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  1.0    0    0    0
## [2,]  1.5    2    4    3
```

1.2.1.2.3 Generate 3 Dimensional Array First, we will create a three dimensional array with the same data as what was used to create the 2-dimensional array on top.

```
# Multidimensional Array
# 1 is r1c1t1, 1.5 in r2c1t1, 0 in r1c2t1, etc.
```

```
# Three dimensions, row first, column second, and tensor third
x <- array(c(1, 1.5, 0, 2, 0, 4, 0, 3), dim=c(2, 2, 2))
dim(x)
```

```
## [1] 2 2 2
```

```
print(x)
```

```
## , , 1
##
##      [,1] [,2]
## [1,]  1.0    0
## [2,]  1.5    2
##
## , , 2
##
##      [,1] [,2]
## [1,]    0    0
## [2,]    4    3
```

Second, in the example below, we will generate a 3-dimensional array. The first dimension corresponds to different income levels, the second marital status, and the third the number of kids.We compute in the example below taxable income in 2008 given income levels given IRS rules.

```
# A, Income Array
ar_income <- seq(0,200000,length.out=3)

# B. Exemptions and Deductions
fl_exemption <- 3500# exemption amount per household member
mt_deduction <- matrix(data=NA, nrow=2, ncol=5)# Marital-status and number of children-specific dedu
mt_deduction[1,1] <- 5450# Single filers
mt_deduction[1,2:5] <- 8000# Single filer with children
mt_deduction[2,] <- 10900# Married couples filing jointly

# C. Taxable Income
mn_taxable_income <- array(NA, dim=c(length(ar_income), 2, 5))
for (y in 1:length(ar_income)){
    for (m in 1:2){
        for (k in 0:4){
            mn_taxable_income[y,m,k+1] <- ar_income[y]-fl_exemption*m-fl_exemption*k-mt_deduction[m,
        }
    }
}

# D. Name dimensions
dimnames(mn_taxable_income)[[1]] = paste0('income=', round(ar_income, 0))
dimnames(mn_taxable_income)[[2]] = paste0('married=', 0:1)
dimnames(mn_taxable_income)[[3]] = paste0('kids=', 0:4)

# E. Print
dim(mn_taxable_income)
```

```
## [1] 3 2 5
```

```
print(mn_taxable_income)
```

```
## , , kids=0
##
##               married=0 married=1
## income=0          -8950    -17900
## income=1e+05      91050     82100
```

```
## income=2e+05    191050    182100
##
## , , kids=1
##
##                married=0 married=1
## income=0         -15000    -21400
## income=1e+05      85000     78600
## income=2e+05     185000    178600
##
## , , kids=2
##
##                married=0 married=1
## income=0         -18500    -24900
## income=1e+05      81500     75100
## income=2e+05     181500    175100
##
## , , kids=3
##
##                married=0 married=1
## income=0         -22000    -28400
## income=1e+05      78000     71600
## income=2e+05     178000    171600
##
## , , kids=4
##
##                married=0 married=1
## income=0         -25500    -31900
## income=1e+05      74500     68100
## income=2e+05     174500    168100
```

1.2.1.3 Array Slicing

1.2.1.3.1 Get a Subset of Array Elements, N Cuts from M Points There is an array with M elements, get N elements from the M elements.

First cut including the starting and ending points.

```
it_M <- 5
it_N <- 4
ar_all_elements = seq(1,10,10)
```

1.2.1.3.2 Remove Elements of Array Select elements with direct indexing, or with head and tail functions. Get the first two elements of three elements array.

```
# Remove last element of array
vars.group.bydf <- c('23','dfa', 'wer')
vars.group.bydf[-length(vars.group.bydf)]
```

```
## [1] "23"  "dfa"
```

```
# Use the head function to remove last element
head(vars.group.bydf, -1)
```

```
## [1] "23"  "dfa"
```

```
head(vars.group.bydf, 2)
```

```
## [1] "23"  "dfa"
```

Get last two elements of array.

```r
# Remove first element of array
vars.group.bydf <- c('23','dfa', 'wer')
vars.group.bydf[2:length(vars.group.bydf)]
```

```
## [1] "dfa" "wer"
```

```r
# Use Tail function
tail(vars.group.bydf, -1)
```

```
## [1] "dfa" "wer"
```

```r
tail(vars.group.bydf, 2)
```

```
## [1] "dfa" "wer"
```

Select all except for the first and the last element of an array.

```r
# define array
ar_amin <- c(0, 0.25, 0.50, 0.75, 1)
# select without head and tail
tail(head(ar_amin, -1), -1)
```

```
## [1] 0.25 0.50 0.75
```

Select the first and the last element of an array. The extreme values.

```r
# define array
ar_amin <- c(0, 0.25, 0.50, 0.75, 1)
# select head and tail
c(head(ar_amin, 1), tail(ar_amin, 1))
```

```
## [1] 0 1
```

1.2.1.4 NA in Array

```r

# Convert Inf and -Inf to NA
x <- c(1, -1, Inf, 10, -Inf)
na_if(na_if(x, -Inf), Inf)
```

1.2.1.4.1 Check if NA is in Array

```
## [1]  1 -1 NA 10 NA
```

1.2.1.5 Complex Number

Handling numbers with real and imaginary components. Two separate issues, given an array of numbers that includes real as well as imaginary numbers, keep subset that only has real components. Additionally, for the same array, generate an equal length version of the array that includes the real components of all numbers.

Define complex numbers.

```r
# Define a complex number
cx_number_a <- 0+0.0460246857561777i
# Define another complex number
cx_number_b <- complex(real = 0.02560982, imaginary = 0.0460246857561777)
# An array of numbers some of which are complex
ar_cx_number <- c(0.02560982+0.000000000i, 0.00000000+0.044895305i,
                  0.00000000+0.009153429i, 0.05462045+0.000000000i,
                  0.00000000+0.001198538i, 0.00000000+0.019267050i)
```

Extract real components from a complex array.

```r
# equi-length real component
ar_fl_number_re <- Re(ar_cx_number)
print(ar_fl_number_re)
```

```
## [1] 0.02560982 0.00000000 0.00000000 0.05462045 0.00000000 0.00000000
```

```r
# equi-length img component
ar_fl_number_im <- Im(ar_cx_number)
print(ar_fl_number_im)
```

```
## [1] 0.000000000 0.044895305 0.009153429 0.000000000 0.001198538 0.019267050
```

Keep only real elements of array.

```r
# subset of array that is real
ar_fl_number_re_subset <- Re(ar_cx_number[Re(ar_cx_number)!=0])
print(ar_fl_number_re_subset)
```

```
## [1] 0.02560982 0.05462045
```

1.2.1.6 Number Formatting

1.2.1.6.1 e notation

1. Case one: *1.149946e+00*
   - this is approximately: 1.14995
2. Case two: *9.048038e-01*
   - this is approximately: 0.90480
3. Case three: *9.048038e-01*
   - this is approximately: 0.90480

1.2.1.7 String Conversions

1.2.1.7.1 Add Positive and Negative Sign in Front of Values We have a sequence of integers, some positive and some negative. We convert this into a string array, and append positive sign in front of positive values.

```r
# An array of integers
ar_it_vals <- seq(-5, 5, by = 1)
# Add positive sign in front of positive and zero elements
st_it_vals <- paste0(ar_it_vals)
st_it_vals[ar_it_vals>0] <- paste0("+", st_it_vals[ar_it_vals>0])
st_it_vals[ar_it_vals==0] <- paste0("±", st_it_vals[ar_it_vals==0])
# Display
print(st_it_vals)
```

```
##  [1] "-5" "-4" "-3" "-2" "-1" "±0" "+1" "+2" "+3" "+4" "+5"
```

1.2.1.8 Basic array calculations

First, we demonstrate how purrr::reduce() works with a simple summation example. We use the addition operator.

```r
# Using R pipe operator
# 1 + 2 + 3 = 6
fl_sum <- 1:3 |> purrr::reduce(`+`)
print(fl_sum)
```

```
## [1] 6
```

Second, what if there is an NA value? NA will be ignored, we will write a custom function. The custom function, to work with reduce, should be such that it is “a binary function that takes two values and returns a single value”.

```
# define sum function that ignores NA
sum_ignore_na <- function(x,y) {
  if (!is.na(x) && !is.na(y)) {
    x + y
  } else if (is.na(x)) {
    y
  } else if (is.na(y)) {
    x
  } else {
    NA
  }
}

# Using R pipe operator
# 1 + 10 + 1 = 12
fl_sum <- c(1, 10, NA, 1) |> purrr::reduce(sum_ignore_na)
print(fl_sum)
```

```
## [1] 12
```

1.2.2 Generate Arrays

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.2.2.1 Generate Often Used Arrays

1.2.2.1.1 Equi-distance Array with Bound Consider multiple income groups in income bins that are equal-width, for the final income group, consider all individuals above some final bin minimum bound. Below the code generates this array of numbers: $0, 20000, 40000, 60000, 80000, 100000, 100000000$.

```
# generate income cut-offs
fl_bin_start <- 0
# width equal to 20,000
fl_bin_width <- 2e4
# final point is 100 million
fl_bin_final_end <- 1e8
# final segment starting point is 100,000 dollars
fl_bin_final_start <- 1e5
# generate tincome bins
ar_income_bins <- c(
  seq(fl_bin_start, fl_bin_final_start, by = fl_bin_width),
  fl_bin_final_end
)
# Display
print(ar_income_bins)
```

```
## [1] 0e+00 2e+04 4e+04 6e+04 8e+04 1e+05 1e+08
```

Generate finer bins, at 5000 USD intervals, and stopping at 200 thousand dollars.

```
fl_bin_start <- 0
fl_bin_width <- 5e3
fl_bin_final_end <- 1e8
fl_bin_final_start <- 2e5
ar_income_bins <- c(
  seq(fl_bin_start, fl_bin_final_start, by = fl_bin_width),
  fl_bin_final_end
)
print(ar_income_bins)
```

```
##  [1] 0.00e+00 5.00e+03 1.00e+04 1.50e+04 2.00e+04 2.50e+04 3.00e+04 3.50e+04 4.00e+04
## [10] 4.50e+04 5.00e+04 5.50e+04 6.00e+04 6.50e+04 7.00e+04 7.50e+04 8.00e+04 8.50e+04
## [19] 9.00e+04 9.50e+04 1.00e+05 1.05e+05 1.10e+05 1.15e+05 1.20e+05 1.25e+05 1.30e+05
## [28] 1.35e+05 1.40e+05 1.45e+05 1.50e+05 1.55e+05 1.60e+05 1.65e+05 1.70e+05 1.75e+05
## [37] 1.80e+05 1.85e+05 1.90e+05 1.95e+05 2.00e+05 1.00e+08
```

1.2.2.1.2 Log Space Arrays Often need to generate arrays on log rather than linear scale, below is log 10 scaled grid.

```
# Parameters
it.lower.bd.inc.cnt <- 3
fl.log.lower <- -10
fl.log.higher <- -9
fl.min.rescale <- 0.01
it.log.count <- 4
# Generate
ar.fl.log.rescaled <- exp(log(10) * seq(log10(fl.min.rescale),
  log10(fl.min.rescale +
    (fl.log.higher - fl.log.lower)),
  length.out = it.log.count
))
ar.fl.log <- ar.fl.log.rescaled + fl.log.lower - fl.min.rescale
# Print
ar.fl.log
```

```
## [1] -10.000000  -9.963430  -9.793123  -9.000000
```

1.2.2.2 Generate Arrays Based on Existing Arrays

1.2.2.2.1 Probability Mass Array and Discrete Value Array There are two arrays, an array of values, and an array of probabilities. The probability array sums to 1. The array of values, however, might not be unique.

First, generate some array of numbers not sorted and some proability mass for each non-sorted, non-unique element of the array.

```
set.seed(123)
it_len <- 10
ar_x <- ceiling(runif(it_len) * 5 + 10)
ar_prob <- dbinom(seq(0, it_len - 1, length.out = it_len), it_len - 1, prob = 0.5)
print(cbind(ar_x, ar_prob))
```

```
##       ar_x     ar_prob
##  [1,]   12 0.001953125
##  [2,]   14 0.017578125
##  [3,]   13 0.070312500
##  [4,]   15 0.164062500
##  [5,]   15 0.246093750
##  [6,]   11 0.246093750
##  [7,]   13 0.164062500
##  [8,]   15 0.070312500
##  [9,]   13 0.017578125
## [10,]   13 0.001953125
```

```
print(paste0("sum(ar_prob)=", sum(ar_prob)))
```

```
## [1] "sum(ar_prob)=1"
```

Second, sorting index for ar_x, and resort ar_prob with the same index:

```
ls_sorted_res <- sort(ar_x, decreasing = FALSE, index.return = TRUE)
ar_idx_increasing_x <- ls_sorted_res$ix
```

```
ar_x_sorted <- ls_sorted_res$x
ar_prob_sorted <- ar_prob[ar_idx_increasing_x]
print(cbind(ar_x_sorted, ar_prob_sorted))
```

```
##       ar_x_sorted ar_prob_sorted
##  [1,]          11    0.246093750
##  [2,]          12    0.001953125
##  [3,]          13    0.070312500
##  [4,]          13    0.164062500
##  [5,]          13    0.017578125
##  [6,]          13    0.001953125
##  [7,]          14    0.017578125
##  [8,]          15    0.164062500
##  [9,]          15    0.246093750
## [10,]          15    0.070312500
```

Third, sum within group and generate unique, using the aggregate function. Then we have a column of unique values and associated probabilities.

```
ar_x_unique <- unique(ar_x_sorted)
mt_prob_unique <- aggregate(ar_prob_sorted, by = list(ar_x_sorted), FUN = sum)
ar_x_unique_prob <- mt_prob_unique$x
print(cbind(ar_x_unique, ar_x_unique_prob))
```

```
##      ar_x_unique ar_x_unique_prob
## [1,]          11      0.246093750
## [2,]          12      0.001953125
## [3,]          13      0.253906250
## [4,]          14      0.017578125
## [5,]          15      0.480468750
```

Finally, the several steps together.

```
# data
set.seed(123)
it_len <- 30
ar_x <- ceiling(runif(it_len) * 20 + 10)
ar_prob <- runif(it_len)
ar_prob <- ar_prob / sum(ar_prob)
# step 1, sort
ls_sorted_res <- sort(ar_x, decreasing = FALSE, index.return = TRUE)
# step 2, unique sorted
ar_x_unique <- unique(ls_sorted_res$x)
# step 3, mass for each unique
mt_prob_unique <- aggregate(ar_prob[ls_sorted_res$ix], by = list(ls_sorted_res$x), FUN = sum)
ar_x_unique_prob <- mt_prob_unique$x
# results
print(cbind(ar_x_unique, ar_x_unique_prob))
```

```
##       ar_x_unique ar_x_unique_prob
##  [1,]          11      0.071718383
##  [2,]          13      0.040040920
##  [3,]          15      0.017708800
##  [4,]          16      0.141199002
##  [5,]          17      0.020211876
##  [6,]          19      0.052488290
##  [7,]          20      0.049104113
##  [8,]          21      0.067328518
##  [9,]          22      0.109454333
## [10,]          23      0.060712145
```

```
## [11,]          24      0.107671406
## [12,]          25      0.015694798
## [13,]          26      0.068567789
## [14,]          28      0.090925756
## [15,]          29      0.001870451
## [16,]          30      0.085303420
```

1.2.2.3 Generate Integer Sequences

1.2.2.3.1 Gapped Possibly Overlapping Consecutive Sequences Now, we generate a set of integer sequences, with gaps in between, but possibly overlapping, for example: $(1, 2, 3, 4, 5), (5, 6), (10, 11)$.

First, we select a small random subset of integers between min and max, and we generate randomly a sequence of `length.out` of the same length. Each `length.out` up to a max. (we adjust in apply in the next block to make sure max given duration does not exceed bound.)

```
# Number of random starting index
it_start_idx <- 11
it_end_idx <- 100
it_startdraws <- 6
# Maximum duration
it_duramax <- 3

# Random seed
set.seed(987)
# Draw random index between min and max
ar_it_start_idx <- sample(
  x = seq(from = it_start_idx, to = it_end_idx, by = 1),
  size = it_startdraws, replace = FALSE
)
ar_it_start_idx <- sort(ar_it_start_idx)
# Draw random durations, replace = TRUE because can repeat
ar_it_duration <- sample(
  x = it_duramax, size = it_startdraws, replace = TRUE
)

# Print
print(glue::glue(
  "random starts + duration: ",
  "{ar_it_start_idx} + {ar_it_duration}"
))
```

```
## random starts + duration: 35 + 3
## random starts + duration: 39 + 3
## random starts + duration: 42 + 1
## random starts + duration: 56 + 2
## random starts + duration: 57 + 1
## random starts + duration: 73 + 1
```

Second, we expand the indexes with neighboring values, and create a list of consecutive integer sequences.

```
# start and end sequences
# note the min operator inside, the makes sure we do not exceed max
ls_ar_it_recession <- apply(
  cbind(ar_it_start_idx, ar_it_start_idx + ar_it_duration),
  1, function(row) {
    return(seq(row[1], min(row[2], it_end_idx)))
  }
)
# Draw it_m from indexed list of it_N
print("ls_ar_it_recession")
```

```
## [1] "ls_ar_it_recession"
```

```r
print(ls_ar_it_recession)
```

```
## [[1]]
## [1] 35 36 37 38
##
## [[2]]
## [1] 39 40 41 42
##
## [[3]]
## [1] 42 43
##
## [[4]]
## [1] 56 57 58
##
## [[5]]
## [1] 57 58
##
## [[6]]
## [1] 73 74
```

Third, we can bring the sequences generated together if we want to

```r
# Combine arrays
ar_it_recession_year <- (
  sort(do.call(base::c, ls_ar_it_recession))
)
# Print
print(glue::glue(
  "print full as array:",
  "{ar_it_recession_year}"
))
```

```
## print full as array:35
## print full as array:36
## print full as array:37
## print full as array:38
## print full as array:39
## print full as array:40
## print full as array:41
## print full as array:42
## print full as array:42
## print full as array:43
## print full as array:56
## print full as array:57
## print full as array:57
## print full as array:58
## print full as array:58
## print full as array:73
## print full as array:74
```

1.2.2.3.2 Gapped non-Overlapping Consecutive Sequences Now, we generate a set of integer sequences, with gaps in between, but not overlapping, for example: $(1, 2, 3), (5, 6), (10, 11)$. We follow a very similar structure as above, but now adjust starting draws by prior accumulated durations.

Note that in the code below, we could end up with less that `it_startdraws` if there are consecutive start draws. We can only have non-consecutive start draws to avoid overlaps.

```r
# Number of random starting index
it_start_idx <- 11
```

```r
it_end_idx <- 100
it_startdraws_max <- 6
it_duramax <- 3

# Random seed
set.seed(987)
# Draw random index between min and max
ar_it_start_idx <- sort(sample(
  seq(it_start_idx, it_end_idx),
  it_startdraws_max,
  replace = FALSE
))
# Draw random durations, replace = TRUE because can repeat
ar_it_duration <- sample(it_duramax, it_startdraws_max, replace = TRUE)

# Check space between starts
ar_it_startgap <- diff(ar_it_start_idx)
ar_it_dura_lenm1 <- ar_it_duration[1:(length(ar_it_duration) - 1)]
# Adjust durations
ar_it_dura_bd <- pmin(ar_it_startgap - 2, ar_it_dura_lenm1)
ar_it_duration[1:(length(ar_it_duration) - 1)] <- ar_it_dura_bd

# Drop consecutive starts
ar_bl_dura_nonneg <- which(ar_it_duration >= 0)
ar_it_start_idx <- ar_it_start_idx[ar_bl_dura_nonneg]
ar_it_duration <- ar_it_duration[ar_bl_dura_nonneg]

# list of recession periods
ls_ar_it_recession_non_overlap <- apply(
  cbind(ar_it_start_idx, ar_it_start_idx + ar_it_duration),
  1, function(row) {
    return(seq(row[1], min(row[2], it_end_idx)))
  }
)

# print
print("ls_ar_it_recession_non_overlap")
```

```
## [1] "ls_ar_it_recession_non_overlap"
```

```r
print(ls_ar_it_recession_non_overlap)
```

```
## [[1]]
## [1] 35 36 37
##
## [[2]]
## [1] 39 40
##
## [[3]]
## [1] 42 43
##
## [[4]]
## [1] 57 58
##
## [[5]]
## [1] 73 74
```

1.2.3 String Arrays

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.2.3.1 Positive or Negative Floating Number to String

There is a number, that contains decimal and possibly negative sign and has some decimals, convert this to a string that is more easily used as a file or folder name.

```
ls_fl_rho <- c(1, -1, -1.5 -100, 0.5, 0.11111111, -199.22123)
for (fl_rho in ls_fl_rho) {
  st_rho <- paste0(round(fl_rho, 4))
  st_rho <- gsub(x = st_rho,  pattern = "-", replacement = "n")
  st_rho <- gsub(x = st_rho,  pattern = "\\.", replacement = "p")
  print(paste0('st_rho=', st_rho))
}
```

```
## [1] "st_rho=1"
## [1] "st_rho=n1"
## [1] "st_rho=n101p5"
## [1] "st_rho=0p5"
## [1] "st_rho=0p1111"
## [1] "st_rho=n199p2212"
```

1.2.3.2 String Replace

- r string wildcard replace between regex
- R - replace part of a string using wildcards

```
# String replacement
gsub(x = paste0(unique(df.slds.stats.perc$it.inner.counter), ':',
                unique(df.slds.stats.perc$z_n_a_n), collapse = ';'),
     pattern = "\n",
     replacement = "")
gsub(x = var,  pattern = "\n", replacement = "")
gsub(x = var.input,  pattern = "\\.", replacement = "_")
```

String replaces a segment, search by wildcard. Given the string below, delete all text between carriage return and pound sign:

```
st_tex_text <- "\n% Lat2ex Comments\n\\newcommand{\\exa}{\\text{from external file: } \\alpha + \\be
st_clean_a1 <- gsub("\\%.*?\\\n", "", st_tex_text)
st_clean_a2 <- gsub("L.*?x", "[LATEX]", st_tex_text)
print(paste0('st_tex_text:', st_tex_text))
```

```
## [1] "st_tex_text:\n% Lat2ex Comments\n\\newcommand{\\exa}{\\text{from external file: } \\alpha +
```

```
print(paste0('st_clean_a1:', st_clean_a1))
```

```
## [1] "st_clean_a1:\n\\newcommand{\\exa}{\\text{from external file: } \\alpha + \\beta}\n"
```

```
print(paste0('st_clean_a2:', st_clean_a2))
```

```
## [1] "st_clean_a2:\n% [LATEX] Comments\n\\newcommand{\\exa}{\\text{from external file: } \\alpha +
```

String delete after a particular string:

```
st_tex_text <- "\\end{equation}\n}\n% Even more comments from Latex preamble"
st_clean_a1 <- gsub("\\\n%.*","", st_tex_text)
print(paste0('st_tex_text:', st_tex_text))
```

```
## [1] "st_tex_text:\\end{equation}\n}\n% Even more comments from Latex preamble"
```

```
print(paste0('st_clean_a1:', st_clean_a1))
```

```
## [1] "st_clean_a1:\\end{equation}\n}"
```

1.2.3.3 Search If and Which String Contains

- r if string contains
- r if string contains either or grepl
- Use grepl to search either of multiple substrings in a text

Search for a single substring in a single string:

```
st_example_a <- 'C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd'
st_example_b <- 'C:/Users/fan/R4Econ/amto/tibble/_main.html'
grepl('_main', st_example_a)
```

```
## [1] FALSE
```

```
grepl('_main', st_example_b)
```

```
## [1] TRUE
```

Search for if one of a set of substring exists in a set of strings. In particular which one of the elements of *ls_spn* contains at least one of the elements of *ls_str_if_contains*. In the example below, only the first path does not contain either the word *aggregate* or *index* in the path. This can be used after all paths have been found recursively in some folder to select only desired paths from the full set of possibilities:

```
ls_spn <- c("C:/Users/fan/R4Econ//panel/basic/fs_genpanel.Rmd",
            "C:/Users/fan/R4Econ//summarize/aggregate/main.Rmd",
            "C:/Users/fan/R4Econ//summarize/index/fs_index_populate.Rmd")
ls_str_if_contains <- c("aggregate", "index")
str_if_contains <- paste(ls_str_if_contains, collapse = "|")
grepl(str_if_contains, ls_spn)
```

```
## [1] FALSE  TRUE  TRUE
```

1.2.3.4 String Split

Given some string, generated for example by cut, get the lower cut starting points, and also the higher end point

```
# Extract 0.216 and 0.500 as lower and upper bounds
st_cut_cate <- '(0.216,0.500]'
# Extract Lower Part
substring(strsplit(st_cut_cate, ",")[[1]][1], 2)
```

```
## [1] "0.216"
```

```
# Extract second part except final bracket Option 1
intToUtf8(rev(utf8ToInt(substring(intToUtf8(rev(utf8ToInt(strsplit(st_cut_cate, ",")[[1]][2]))), 2))
```

```
## [1] "0.500"
```

```
# Extract second part except final bracket Option 2
gsub(strsplit(st_cut_cate, ",")[[1]][2],  pattern = "]", replacement = "")
```

```
## [1] "0.500"
```

Make a part of a string bold. Go from "ABC EFG, OPQ, RST" to "**ABC EFG**, OPQ, RST". This could be for making the name of an author bold, and preserve affiliation information.

```
st_paper_author_ori <- "ABC EFG, OPQ, RST"
ar_st_ori <- strsplit(st_paper_author_ori, ", ")[[1]]
st_after_1stcomma <- paste0(ar_st_ori[2:length(ar_st_ori)], collapse= ", ")
```

```
st_paper_author <- paste0('<b>', ar_st_ori[1], "</b>, ", st_after_1stcomma )
print(st_paper_author)
```

```
## [1] "<b>ABC EFG</b>, OPQ, RST"
```

1.2.3.5 String Concatenate

Concatenate string array into a single string.

```
# Simple Collapse
vars.group.by <- c('abc', 'efg')
paste0(vars.group.by, collapse='|')
```

```
## [1] "abc|efg"
```

Concatenate a numeric array into a single string.

```
# Simple Collapse
set.seed(123)
ar_fl_numbers <- runif(5)
paste0('ar_fl_numbers = ',
       paste(round(ar_fl_numbers,3), collapse=', ')
)
```

```
## [1] "ar_fl_numbers = 0.288, 0.788, 0.409, 0.883, 0.94"
```

1.2.3.6 String Add Leading Zero

```
# Add Leading zero for integer values to allow for sorting when
# integers are combined into strings
it_z_n <- 1
it_a_n <- 192
print(sprintf("%02d", it_z_n))
```

```
## [1] "01"
```

```
print(sprintf("%04d", it_a_n))
```

```
## [1] "0192"
```

1.2.3.7 Substring Components

Given a string, with certain structure, get components.

- r time string get month and year and day

```
snm_full <- "20100701"
snm_year <-substr(snm_full,0,4)
snm_month <-substr(snm_full,5,6)
snm_day <-substr(snm_full,7,8)
print(paste0('full:', snm_full,
             ', year:', snm_year,
             ', month:', snm_month,
             ', day:', snm_day))
```

```
## [1] "full:20100701, year:2010, month:07, day:01"
```

1.2.4 Mesh Matrices, Arrays and Scalars

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

- r expand.grid meshed array to matrix

- r meshgrid
- r array to matrix
- r reshape array to matrix
- dplyr permuations rows of matrix and element of array
- tidyr expand_grid mesh matrix and vector

1.2.4.1 Mesh Two or More Vectors with expand_grid

In the example below, we have a matrix that is 2 by 2 (endogenous states), a vector that is 3 by 1 (choices), and another matrix that is 4 by 3 (exogenous states shocks).

We want to generate a tibble dataset that meshes the matrix and the vector, so that all combinations show up. Additionally, we want to add some additional values that are common across all rows to the meshed dataframe.

Note *expand_grid* is a from tidyr 1.0.0.

```
# A. Generate the 5 by 2 Matrix (ENDO STATES)
# it_child_count = N, the number of children
it_N_child_cnt = 2
# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
fl_rho = 0.1
fl_lambda = 1.1
mt_nP_A_alpha = cbind(ar_nN_A, ar_nN_alpha, fl_rho, fl_lambda)
ar_st_varnames <- c('s_A', 's_alpha', 'p_rho', 'p_lambda')
tb_states_endo <- as_tibble(mt_nP_A_alpha) %>%
  rename_all(~c(ar_st_varnames)) %>%
  rowid_to_column(var = "state_id")

# B. Choice Grid
it_N_choice_cnt = 3
fl_max = 10
fl_min = 0
ar_nN_d = seq(fl_min, fl_max, length.out = it_N_choice_cnt)
ar_st_varnames <- c('c_food')
tb_choices <- as_tibble(ar_nN_d) %>%
  rename_all(~c(ar_st_varnames)) %>%
  rowid_to_column(var = "choice_id")

# C. Shock Grid
set.seed(123)
it_N_shock_cnt = 4
ar_nQ_shocks = exp(rnorm(it_N_shock_cnt, mean=0, sd=1))
ar_st_varnames <- c('s_eps')
tb_states_exo <- as_tibble(ar_nQ_shocks) %>%
  rename_all(~c(ar_st_varnames)) %>%
  rowid_to_column(var = "shock_id")

# dataframe expand with other non expanded variables
ar_st_varnames <-
tb_states_shk_choices <- tb_states_endo %>%
  expand_grid(tb_choices) %>%
  expand_grid(tb_states_exo) %>%
  select(state_id, choice_id, shock_id,
         s_A, s_alpha, s_eps, c_food,
         p_rho, p_lambda)

# display
```

state_id	choice_id	shock_id	s_A	s_alpha	s_eps	c_food	p_rho	p_lambda
1	1	1	-2	0.1	0.5709374	0	0.1	1.1
1	1	2	-2	0.1	0.7943926	0	0.1	1.1
1	1	3	-2	0.1	4.7526783	0	0.1	1.1
1	1	4	-2	0.1	1.0730536	0	0.1	1.1
1	2	1	-2	0.1	0.5709374	5	0.1	1.1
1	2	2	-2	0.1	0.7943926	5	0.1	1.1
1	2	3	-2	0.1	4.7526783	5	0.1	1.1
1	2	4	-2	0.1	1.0730536	5	0.1	1.1
1	3	1	-2	0.1	0.5709374	10	0.1	1.1
1	3	2	-2	0.1	0.7943926	10	0.1	1.1
1	3	3	-2	0.1	4.7526783	10	0.1	1.1
1	3	4	-2	0.1	1.0730536	10	0.1	1.1
2	1	1	2	0.9	0.5709374	0	0.1	1.1
2	1	2	2	0.9	0.7943926	0	0.1	1.1
2	1	3	2	0.9	4.7526783	0	0.1	1.1
2	1	4	2	0.9	1.0730536	0	0.1	1.1
2	2	1	2	0.9	0.5709374	5	0.1	1.1
2	2	2	2	0.9	0.7943926	5	0.1	1.1
2	2	3	2	0.9	4.7526783	5	0.1	1.1
2	2	4	2	0.9	1.0730536	5	0.1	1.1
2	3	1	2	0.9	0.5709374	10	0.1	1.1
2	3	2	2	0.9	0.7943926	10	0.1	1.1
2	3	3	2	0.9	4.7526783	10	0.1	1.1
2	3	4	2	0.9	1.0730536	10	0.1	1.1

```r
kable(tb_states_shk_choices) %>% kable_styling_fc()
```

Using expand_grid directly over arrays

```r
# expand grid with dplyr
expand_grid(x = 1:3, y = 1:2, z = -3:-1)
```

1.2.4.2 Mesh Arrays with expand.grid

Given two arrays, mesh the two arrays together.

```r
# use expand.grid to generate all combinations of two arrays

it_ar_A = 5
it_ar_alpha = 10

ar_A = seq(-2, 2, length.out=it_ar_A)
ar_alpha = seq(0.1, 0.9, length.out=it_ar_alpha)

mt_A_alpha = expand.grid(A = ar_A, alpha = ar_alpha)

mt_A_meshed = mt_A_alpha[,1]
dim(mt_A_meshed) = c(it_ar_A, it_ar_alpha)

mt_alpha_meshed = mt_A_alpha[,2]
dim(mt_alpha_meshed) = c(it_ar_A, it_ar_alpha)

# display
kable(mt_A_meshed) %>%
  kable_styling_fc()
```

-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2

```
kable(mt_alpha_meshed) %>%
  kable_styling_fc_wide()
```

0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9
0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9
0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9
0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9
0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9

Two Identical Arrays, individual attributes, each column is an individual for a matrix, and each row is also an individual.

```
# use expand.grid to generate all combinations of two arrays

it_ar_A = 5

ar_A = seq(-2, 2, length.out=it_ar_A)
mt_A_A = expand.grid(Arow = ar_A, Arow = ar_A)
mt_Arow = mt_A_A[,1]
dim(mt_Arow) = c(it_ar_A, it_ar_A)
mt_Acol = mt_A_A[,2]
dim(mt_Acol) = c(it_ar_A, it_ar_A)

# display
kable(mt_Arow) %>%
  kable_styling_fc()
```

```
kable(mt_Acol) %>%
  kable_styling_fc()
```

1.3 Matrix

1.3.1 Generate Matrixes

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.3.1.1 Create a N by 2 Matrix from 3 arrays

Names of each array become row names automatically.

```
ar_row_one <- c(-1,+1)
ar_row_two <- c(-3,-2)
ar_row_three <- c(0.35,0.75)
```

-2	-2	-2	-2	-2
-1	-1	-1	-1	-1
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2

-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2

ar_row_one	-1.00	1.00
ar_row_two	-3.00	-2.00
ar_row_three	0.35	0.75

```
mt_n_by_2 <- rbind(ar_row_one, ar_row_two, ar_row_three)
kable(mt_n_by_2) %>%
  kable_styling_fc()
```

1.3.1.2 Name Matrix Columns and Rows

```
# An empty matrix with Logical NA
mt_named <- matrix(data=NA, nrow=2, ncol=2)
colnames(mt_named) <- paste0('c', seq(1,2))
rownames(mt_named) <- paste0('r', seq(1,2))
mt_named
```

```
##    c1 c2
## r1 NA NA
## r2 NA NA
```

1.3.1.3 Generate NA Matrix

- Best way to allocate matrix in R, NULL vs NA?

Allocate with NA or NA_real_ or NA_int_. Clarity in type definition is preferred.

```
# An empty matrix with Logical NA
mt_na <- matrix(data=NA, nrow=2, ncol=2)
str(mt_na)
```

```
##  logi [1:2, 1:2] NA NA NA NA
```

```
# An empty matrix with numerica NA
mt_fl_na <- matrix(data=NA_real_, nrow=2, ncol=2)
mt_it_na <- matrix(data=NA_integer_, nrow=2, ncol=2)

str(mt_fl_na)
```

```
##  num [1:2, 1:2] NA NA NA NA
```

```
str(mt_fl_na)
```

```
##  num [1:2, 1:2] NA NA NA NA
```

1.3.1.4 Generate Matrixes with values

Random draw from the normal distribution, random draw from the uniform distribution, and combine resulting matrixes.

```
# Generate 15 random normal, put in 5 rows, and 3 columns
mt_rnorm <- matrix(rnorm(15,mean=0,sd=1), nrow=5, ncol=3)

# Generate 15 random normal, put in 5 rows, and 3 columns
mt_runif <- matrix(runif(15,min=0,max=1), nrow=5, ncol=5)
```

0.129	-0.446	-0.556	0.318	0.369	0.266	0.318	0.369
1.715	1.224	1.787	0.232	0.152	0.858	0.232	0.152
0.461	0.360	0.498	0.143	0.139	0.046	0.143	0.139
-1.265	0.401	-1.967	0.415	0.233	0.442	0.415	0.233
-0.687	0.111	0.701	0.414	0.466	0.799	0.414	0.466

with byrow=FALSE, the default, will fill col by col

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

```
# Combine
mt_rnorm_runif <- cbind(mt_rnorm, mt_runif)

# Display
kable(round(mt_rnorm_runif, 3)) %>% kable_styling_fc()
```

Now we generate a matrix with sequential integers, and either fill matrix by columns or fill matrix by rows.

```
# with byrow set to FALSE, will fill first col, then second col, etc..
mt_index_colbycol <- matrix(seq(0, 15), nrow=4, ncol=4, byrow=FALSE)
# Display
kable(mt_index_colbycol,
  caption= "with byrow=FALSE, the default, will fill col by col") %>%
  kable_styling_fc()
```

```
# with byrow set to TRUE, will fill row by row
mt_index_rowbyrow <- matrix(seq(0, 15), nrow=4, ncol=4, byrow=TRUE)
# Display
kable(mt_index_rowbyrow,
  caption= " with byrow=TRUE, will fill row by row") %>%
  kable_styling_fc()
```

1.3.1.5 Replace a Subset of Matrix Values by NA_real_

For values in matrix that fall below or above some thresholds, we will replace these values by NA_real_.

```
fl_max_val <- 0.8
fl_min_val <- 0.2
mt_rnorm_runif_bd <- mt_rnorm_runif
mt_rnorm_runif_bd[which(mt_rnorm_runif < fl_min_val)] <- NA_real_
mt_rnorm_runif_bd[which(mt_rnorm_runif > fl_max_val)] <- NA_real_
# Print
print(mt_rnorm_runif_bd)
```

```
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
## [1,]       NA        NA        NA 0.3181810 0.3688455 0.2659726 0.3181810 0.3688455
## [2,]       NA        NA        NA 0.2316258        NA        NA 0.2316258        NA
```

with byrow=TRUE, will fill row by row

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Each row sort low to high

-0.556	-0.446	0.129	0.266	0.318	0.318	0.369	0.369
0.152	0.152	0.232	0.232	0.858	1.224	1.715	1.787
0.046	0.139	0.139	0.143	0.143	0.360	0.461	0.498
-1.967	-1.265	0.233	0.233	0.401	0.415	0.415	0.442
-0.687	0.111	0.414	0.414	0.466	0.466	0.701	0.799

Each column sort low to high

-1.265	-0.446	-1.967	0.143	0.139	0.046	0.143	0.139
-0.687	0.111	-0.556	0.232	0.152	0.266	0.232	0.152
0.129	0.360	0.498	0.318	0.233	0.442	0.318	0.233
0.461	0.401	0.701	0.414	0.369	0.799	0.414	0.369
1.715	1.224	1.787	0.415	0.466	0.858	0.415	0.466

```
## [3,] 0.4609162 0.3598138 0.4978505        NA        NA        NA        NA        NA
## [4,]        NA 0.4007715        NA 0.4145463 0.2330341 0.4422001 0.4145463 0.2330341
## [5,]        NA        NA 0.7013559 0.4137243 0.4659625 0.7989248 0.4137243 0.4659625
```

1.3.1.6 Sort Each Matrix Row or Column

Now we sort within each row or within each column of the random matrix.

```
# Within row sort
mt_rnorm_runif_row_sort <- t(apply(
  mt_rnorm_runif, 1, sort
))
# Within column sort, note no transpose
mt_rnorm_runif_col_sort <- apply(
  mt_rnorm_runif, 2, sort
)
# Display
kable(round(mt_rnorm_runif_row_sort, 3),
      caption="Each row sort low to high") %>%
  kable_styling_fc()
```

```
kable(round(mt_rnorm_runif_col_sort, 3),
      caption="Each column sort low to high") %>%
  kable_styling_fc()
```

1.3.1.7 Compute Column and Row Statistics

Compute column and row means, and also column and row sums

```
print(paste0('colSums=',
             paste(round(
               colSums(mt_rnorm_runif),3), collapse=',')
             ))
```

```
## [1] "colSums=0.353,1.65,0.464,1.521,1.359,2.411,1.521,1.359"
```

```
print(paste0('colMeans=',
             paste(round(
               colMeans(mt_rnorm_runif),3), collapse=',')
             ))
```

```
## [1] "colMeans=0.071,0.33,0.093,0.304,0.272,0.482,0.304,0.272"
```

```
print(paste0('rowSums=',
             paste(round(
```

```
              rowSums(mt_rnorm_runif),3), collapse=',')
            ))
```

```
## [1] "rowSums=0.768,6.352,1.928,-1.094,2.683"
```

```
print(paste0('rowMeans=',
             paste(round(
               rowMeans(mt_rnorm_runif),3), collapse=',')
             ))
```

```
## [1] "rowMeans=0.096,0.794,0.241,-0.137,0.335"
```

1.3.1.8 Add Column to Matrix with Common Scalar Value

Given some matrix of information, add a column, where all rows of the column have the same numerical value. Use the matrix created prior. - R add column to matrix - r append column to matrix constant value

```
fl_new_first_col_val <- 111
fl_new_last_col_val <- 999
mt_with_more_columns <- cbind(rep(fl_new_first_col_val, dim(mt_rnorm_runif)[1]),
                              mt_rnorm_runif,
                              rep(fl_new_last_col_val, dim(mt_rnorm_runif)[1]))
# Display
kable(mt_with_more_columns) %>% kable_styling_fc_wide()
```

111	0.1292877	-0.4456620	-0.5558411	0.3181810	0.3688455	0.2659726	0.3181810	0.3688455	999
111	1.7150650	1.2240818	1.7869131	0.2316258	0.1524447	0.8578277	0.2316258	0.1524447	999
111	0.4609162	0.3598138	0.4978505	0.1428000	0.1388061	0.0458312	0.1428000	0.1388061	999
111	-1.2650612	0.4007715	-1.9666172	0.4145463	0.2330341	0.4422001	0.4145463	0.2330341	999
111	-0.6868529	0.1106827	0.7013559	0.4137243	0.4659625	0.7989248	0.4137243	0.4659625	999

1.3.2 Linear Algebra

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.3.2.1 Matrix Multiplication

Multiply Together a 3 by 2 matrix and a 2 by 1 vector

```
ar_row_one <- c(-1,+1)
ar_row_two <- c(-3,-2)
ar_row_three <- c(0.35,0.75)
mt_n_by_2 <- rbind(ar_row_one, ar_row_two, ar_row_three)

ar_row_four <- c(3,4)

# Matrix Multiplication
mt_out <- mt_n_by_2 %*% ar_row_four
print(mt_n_by_2)
```

```
##               [,1]  [,2]
## ar_row_one   -1.00  1.00
## ar_row_two   -3.00 -2.00
## ar_row_three  0.35  0.75
```

```
print(ar_row_four)
```

```
## [1] 3 4
```

```
print(mt_out)
```

```
##                 [,1]
## ar_row_one      1.00
## ar_row_two    -17.00
## ar_row_three    4.05
```

1.4 Regular Expression, Date, etc.

1.4.1 String Regular Expression

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.4.1.1 Character Class

The regex documentation states that: "A character class is a list of characters enclosed between '[' and ']' which matches any single character in that list"

First, in the example below, we look for strings that contain at a single letter, symbol, or number in the string list enclosed in square brackets.

```
# Fou words with metacharacters
ls_st_regex_charclass <- c(
  '00d',
  'z\\12323_4',
  'pa(_2+\\3',
  'p99.9_sdfasdpf0',
  'k9p.e_d+fd')
# Matches any characters with the letter p
print(grepl("[p]", ls_st_regex_charclass))
# Matches any characters with backslash
print(grepl("[\\]", ls_st_regex_charclass))
# Matches any characters with the number 3
print(grepl("[3]", ls_st_regex_charclass))

# > print(grepl("[p]", ls_st_regex_charclass))
# [1] FALSE FALSE  TRUE  TRUE  TRUE
# > print(grepl("[\\]", ls_st_regex_charclass))
# [1] FALSE  TRUE  TRUE FALSE FALSE
# > print(grepl("[3]", ls_st_regex_charclass))
# [1] FALSE  TRUE  TRUE FALSE FALSEZ
```

Second, using the same set of words as examples, we now test if the strings contain at least a letter, symbol, or number in the string lis enclosed in square brakets.

```
# Matches any characters eithr with letter p or d
print(grepl('[pd]', ls_st_regex_charclass))
# Matches any characters eithr with letter p or _
print(grepl('[p_]', ls_st_regex_charclass))
# Matches any characters eithr with letter p or _ or 0
print(grepl('[p_0]', ls_st_regex_charclass))

# > print(grepl('[pd]', ls_st_regex_charclass))
# [1]  TRUE FALSE  TRUE  TRUE  TRUE
# > print(grepl('[p_]', ls_st_regex_charclass))
# [1] FALSE  TRUE  TRUE  TRUE  TRUE
# > print(grepl('[p_0]', ls_st_regex_charclass))
# [1] TRUE TRUE TRUE TRUE TRUE
```

Third, using '^', carat, we exclude strings that include characters, letters, and symols. The documentation states that: "unless the first character of the list is the caret '^', when it matches any character not in the list".

```
# Finds strings that has anything other than d and 0
print(grepl('[^d0]', ls_st_regex_charclass))

# > print(grepl('[^d0]', ls_st_regex_charclass))
# [1] FALSE  TRUE  TRUE  TRUE  TRUE
```

1.4.1.2 Repetition Quantifiers

We have the following quantifiers:

- '?': The preceding item is optional and will be matched at most once.
- '*': The preceding item will be matched zero or more times.
- '+':The preceding item will be matched one or more times.
- '{n}': The preceding item is matched exactly n times.
- '{n,}': The preceding item is matched n or more times.
- '{n,m}': The preceding item is matched at least n times, but not more than m times.

Now, we identifier strings where certain characters appear a certain number of times.

```
# Fou words with metacharacters
ls_st_regex_rep_quantifer <- c(
  '00d',
  'z\\12323_40',
  'ppa(_2+\\3',
  'p99.9_sdfasdpf0',
  'k9p.e_d+fd')
# Matches any characters pp
print(grepl("[p]{2}", ls_st_regex_rep_quantifer))
# Matches any characters with the number 3
print(grepl("[9]{2}", ls_st_regex_rep_quantifer))

# > print(grepl("[p]{2}", ls_st_regex_rep_quantifer))
# [1] FALSE FALSE  TRUE FALSE FALSE
# > print(grepl("[9]{2}", ls_st_regex_rep_quantifer))
# [1] FALSE FALSE FALSE  TRUE FALSE
```

1.4.1.3 Matches Strings With Multiple Conditions with Repetition Quantifiers

Now we match string that satisfy multiple conditions jointly. We have the following quantifiers:

- '?': The preceding item is optional and will be matched at most once.
- '*': The preceding item will be matched zero or more times.
- '+':The preceding item will be matched one or more times.
- '{n}': The preceding item is matched exactly n times.
- '{n,}': The preceding item is matched n or more times.
- '{n,m}': The preceding item is matched at least n times, but not more than m times.

First, we define our string array.

```
ls_st_regex_joint <- c(
  '_asdf123p',
  'pz12p323_40_',
  'ppa(_2+\\3',
  'p9_sdfasdpf0',
```

```
  'p_k9p.e_d+fd',
  'p123k_dfk')
```

Second, we identify three cases below:

1. Matching words containing just "p_"
2. Matching words containing "p9_" (replace 9 by another other alpha-numeric)
3. Matching words containing either "p_" or "p9_"

```
# Start with p, followed by _
print(grepl("p_", ls_st_regex_joint))
# Start with p, followed by a single alpha-numeric, then _
print(grepl("p[[:alnum:]]_", ls_st_regex_joint))
# Start with p, followed by either:
# 1 single alpha-numeric, then _
# no alpha-numeric, then _
print(grepl("p[[:alnum:]]?_", ls_st_regex_joint))

# > print(grepl("p_", ls_st_regex_joint))
# [1] FALSE FALSE FALSE FALSE  TRUE FALSE
# > print(grepl("p[[:alnum:]]_", ls_st_regex_joint))
# [1] FALSE FALSE FALSE  TRUE FALSE FALSE
# > print(grepl("p[[:alnum:]]?_", ls_st_regex_joint))
# [1] FALSE FALSE FALSE  TRUE  TRUE FALSE
```

Third, we identify cases, where there the word contains substring starting with "p" and ending with "_", with any number (including 0) of alpha-numeric characters in between. Note:

1. In the first string, both "_" and "p" appear, but "p" appears after, so does not match
2. Note in the second word, "p" and "_" appear multiple times
3. Note in the third word, "p" and "_" both appear, but are separated by a non-alpha-numeric character

```
print(grepl("p[[:alnum:]]*_", ls_st_regex_joint))

# > print(grepl("p[[:alnum:]]*_", ls_st_regex_joint))
# [1] FALSE  TRUE FALSE  TRUE  TRUE  TRUE
```

Fourth, we use alternative repetition quantifiers, plus, rather than asterisks, which means we must have at least one alpha-numeric character in between "p" and the "_", in which case, the fifth word no longer satisfies the search condition.

```
# p and _ separated by at least 1 alpha numerics
print(grepl("p[[:alnum:]]+_", ls_st_regex_joint))

# > print(grepl("p[[:alnum:]]+_", ls_st_regex_joint))
# [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE
```

Chapter 2

Manipulate and Summarize Dataframes

2.1 Variables in Dataframes

2.1.1 Generate Dataframe

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

2.1.1.1 Simple Dataframe, Name Columns

```
# 5 by 3 matrix
mt_rnorm_a <- matrix(rnorm(4,mean=0,sd=1), nrow=5, ncol=3)

# Column Names
ar_st_varnames <- c('id','var1','varb','vartheta')

# Combine to tibble, add name col1, col2, etc.
tb_combine <- as_tibble(mt_rnorm_a) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~c(ar_st_varnames))

# Display
kable(tb_combine) %>% kable_styling_fc()
```

2.1.1.2 Dataframe with Row and Column Names and Export

First, we generate an empty matrix. Second, we compute values to fill in matrix cells.

```
# an NA matrix
it_nrow <- 5
it_ncol <- 3
mt_na <- matrix(NA, nrow=it_nrow, ncol=it_ncol)
```

id	var1	varb	vartheta
1	-1.1655448	-0.8185157	0.6849361
2	-0.8185157	0.6849361	-0.3200564
3	0.6849361	-0.3200564	-1.1655448
4	-0.3200564	-1.1655448	-0.8185157
5	-1.1655448	-0.8185157	0.6849361

3	5	7
5	8	11
7	11	15
9	14	19
11	17	23

```
# array of nrow values
ar_it_nrow <- seq(1, it_nrow)
ar_it_ncol <- seq(1, it_ncol)

# Generate values in matrix
for (it_row in ar_it_nrow) {
  for (it_col in ar_it_ncol) {
    print(glue::glue("row={it_row} and col={it_col}"))
    mt_na[it_row, it_col] = it_row*it_col + it_row + it_col
  }
}
```

```
## row=1 and col=1
## row=1 and col=2
## row=1 and col=3
## row=2 and col=1
## row=2 and col=2
## row=2 and col=3
## row=3 and col=1
## row=3 and col=2
## row=3 and col=3
## row=4 and col=1
## row=4 and col=2
## row=4 and col=3
## row=5 and col=1
## row=5 and col=2
## row=5 and col=3
```

```
# Display
kable(mt_na) %>% kable_styling_fc()
```

Third, we label the rows and the columns. Note that we will include the column names as column names, but the row names will be included as a variable.

```
# Column Names
ar_st_col_names <- paste0('colval=', ar_it_ncol)
ar_st_row_names <- paste0('rowval=', ar_it_nrow)

# Create tibble, and add in column and row names
tb_row_col_named <- as_tibble(mt_na) %>%
  rename_all(~c(ar_st_col_names)) %>%
  mutate(row_name = ar_st_row_names) %>%
  select(row_name, everything())

# Display
kable(tb_row_col_named) %>% kable_styling_fc()
```

Finally, we generate a file name for exporting this tibble to a CSV file. We create a file name with a time stamp.

```
# Create a file name with date stamp
st_datetime <- base::format(Sys.time(),  "%Y%m%d-%H%M%S")
# Copying a fixed date to avoid generating multiple testing files
```

row_name	colval=1	colval=2	colval=3
rowval=1	3	5	7
rowval=2	5	8	11
rowval=3	7	11	15
rowval=4	9	14	19
rowval=5	11	17	23

```
# The date string below is generated by Sys.time()
st_snm_filename <- paste0("tibble_out_test_", st_datetime, '.csv')

# Create a file name with the time stamp.
spn_file_path = file.path(
  "C:", "Users", "fan",
  "R4Econ", "amto", "tibble", "_file",
  st_snm_filename,
  fsep = .Platform$file.sep)

# Save to file
write_csv(tb_row_col_named, spn_file_path)
```

2.1.1.3 Generate Tibble given Matrixes and Arrays

Given Arrays and Matrixes, Generate Tibble and Name Variables/Columns

- naming tibble columns
- tibble variable names
- dplyr rename tibble
- dplyr rename tibble all variables
- dplyr rename all columns by index
- dplyr tibble add index column
- see also: SO-51205520

```
# Base Inputs
ar_col <- c(-1,+1)
mt_rnorm_a <- matrix(rnorm(4,mean=0,sd=1), nrow=2, ncol=2)
mt_rnorm_b <- matrix(rnorm(4,mean=0,sd=1), nrow=2, ncol=4)

# Combine Matrix
mt_combine <- cbind(ar_col, mt_rnorm_a, mt_rnorm_b)
colnames(mt_combine) <- c('ar_col',
                          paste0('matcolvar_grpa_', seq(1,dim(mt_rnorm_a)[2])),
                          paste0('matcolvar_grpb_', seq(1,dim(mt_rnorm_b)[2])))

# Variable Names
ar_st_varnames <- c('var_one',
                    paste0('tibcolvar_ga_', c(1,2)),
                    paste0('tibcolvar_gb_', c(1,2,3,4)))

# Combine to tibble, add name col1, col2, etc.
tb_combine <- as_tibble(mt_combine) %>% rename_all(~c(ar_st_varnames))

# Add an index column to the dataframe, ID column
tb_combine <- tb_combine %>% rowid_to_column(var = "ID")

# Change all gb variable names
tb_combine <- tb_combine %>%
                  rename_at(vars(starts_with("tibcolvar_gb_")),
                            funs(str_replace(., "_gb_", "_gbrenamed_")))
```

```
# Tibble back to matrix
mt_tb_combine_back <- data.matrix(tb_combine)

# Display
kable(mt_combine) %>% kable_styling_fc_wide()
```

ar_col	matcolvar_grpa_1	matcolvar_grpa_2	matcolvar_grpb_1	matcolvar_grpb_2	matcolvar_grpb_3	matcolvar_grpb_4
-1	-1.3115224	-0.1294107	-0.1513960	-3.2273228	-0.1513960	-3.2273228
1	-0.5996083	0.8867361	0.3297912	-0.7717918	0.3297912	-0.7717918

```
kable(tb_combine) %>% kable_styling_fc_wide()
```

ID	var_one	tibcolvar_ga_1	tibcolvar_ga_2	tibcolvar_gbrenamed_1	tibcolvar_gbrenamed_2	tibcolvar_gbrenamed_3	tibcolvar_gbrenamed_4
1	-1	-1.3115224	-0.1294107	-0.1513960	-3.2273228	-0.1513960	-3.2273228
2	1	-0.5996083	0.8867361	0.3297912	-0.7717918	0.3297912	-0.7717918

```
kable(mt_tb_combine_back) %>% kable_styling_fc_wide()
```

ID	var_one	tibcolvar_ga_1	tibcolvar_ga_2	tibcolvar_gbrenamed_1	tibcolvar_gbrenamed_2	tibcolvar_gbrenamed_3	tibcolvar_gbrenamed_4
1	-1	-1.3115224	-0.1294107	-0.1513960	-3.2273228	-0.1513960	-3.2273228
2	1	-0.5996083	0.8867361	0.3297912	-0.7717918	0.3297912	-0.7717918

2.1.1.4 Generate a Table from Lists

We run some function, whose outputs are named list, we store the values of the named list as additional rows into a dataframe whose column names are the names from named list.

First, we generate the function that returns named lists.

```
# Define a function
ffi_list_generator <- function(it_seed=123) {
  set.seed(it_seed)
  fl_abc <- rnorm(1)
  ar_efg <- rnorm(3)
  st_word <- sample(LETTERS, 5, replace = TRUE)
  ls_return <- list("abc" = fl_abc, "efg" = ar_efg, "opq" = st_word)
  return(ls_return)
}
# Run the function
it_seed=123
ls_return <- ffi_list_generator(it_seed)
print(ls_return)
```

```
## $abc
## [1] -0.5604756
##
## $efg
## [1] -0.23017749  1.55870831  0.07050839
##
## $opq
## [1] "K" "E" "T" "N" "V"
```

Second, we list of lists by running the function above with different starting seeds. We store results in a two-dimensional list.

```
# Run function once to get length
ls_return_test <- ffi_list_generator(it_seed=123)
it_list_len <- length(ls_return_test)

# list of list frame
it_list_of_list_len <- 5
```

abc	efg	opq
-0.5604756	-0.23017749, 1.55870831, 0.07050839	K, E, T, N, V
-1.385071	0.03832318, -0.76303016, 0.21230614	J, A, O, T, N
0.933327	-0.52503178, 1.81443979, 0.08304562	C, T, M, S, K
0.366734	0.3964520, -0.7318437, 0.9462364	Z, L, J, Y, P
-0.5677337	-0.814760579, -0.493939596, 0.001818846	Y, C, O, F, U

```
ls_ls_return <- vector(mode = "list", length = it_list_of_list_len*it_list_len)
dim(ls_ls_return) <- c(it_list_of_list_len, it_list_len)

# Fill list of list
ar_seeds <- seq(123, 123 + it_list_of_list_len - 1)
it_ctr <- 0
for (it_seed in ar_seeds) {
  print(it_seed)
  it_ctr <- it_ctr + 1
  ls_return <- ffi_list_generator(it_seed)
  ls_ls_return[it_ctr,] <- ls_return
}
```

```
## [1] 123
## [1] 124
## [1] 125
## [1] 126
## [1] 127
```

```
# print 2d list
print(ls_ls_return)
```

```
##      [,1]       [,2]      [,3]
## [1,] -0.5604756 numeric,3 character,5
## [2,] -1.385071  numeric,3 character,5
## [3,] 0.933327   numeric,3 character,5
## [4,] 0.366734   numeric,3 character,5
## [5,] -0.5677337 numeric,3 character,5
```

Third, we convert the list to a tibble dataframe. Prior to conversion we add names to the 1st and 2nd dimensions of the list. Then we print the results.

```
# get names from named list
ar_st_names <- names(ls_return_test)
dimnames(ls_ls_return)[[2]] <- ar_st_names
dimnames(ls_ls_return)[[1]] <- paste0('seed_', ar_seeds)

# Convert to dataframe
tb_ls_ls_return <- as_tibble(ls_ls_return)

# print
kable(tb_ls_ls_return) %>% kable_styling_fc()
```

Fourth, to export list to csv file, we need to unlist list contents. See also Create a tibble containing list columns

```
# Unlist
tb_unlisted <- tb_ls_ls_return %>%
  rowwise() %>%
  mutate_if(is.list,
    funs(paste(unlist(.), sep='', collapse=', ')))
# print on screen, can see values
print(tb_unlisted)
```

2.1.1.5 Rename Tibble with Numeric Column Names

After reshaping, often could end up with variable names that are all numeric, intgers for example, how to rename these variables to add a common prefix for example.

```
# Base Inputs
ar_col <- c(-1,+1)
mt_rnorm_c <- matrix(rnorm(4,mean=0,sd=1), nrow=5, ncol=10)
mt_combine <- cbind(ar_col, mt_rnorm_c)

# Variable Names
ar_it_cols_ctr <- seq(1, dim(mt_rnorm_c)[2])
ar_st_varnames <- c('var_one', ar_it_cols_ctr)

# Combine to tibble, add name col1, col2, etc.
tb_combine <- as_tibble(mt_combine) %>% rename_all(~c(ar_st_varnames))

# Add an index column to the dataframe, ID column
tb_combine_ori <- tb_combine %>% rowid_to_column(var = "ID")

# Change all gb variable names
tb_combine <- tb_combine_ori %>%
                  rename_at(
                    vars(num_range('',ar_it_cols_ctr)),
                    funs(paste0("rho", . , 'var'))
                    )

# Display
kable(tb_combine_ori) %>% kable_styling_fc_wide()
```

ID	var_one	1	2	3	4	5	6	7	8	9	10
1	-1	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199
2	1	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086
3	-1	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632
4	1	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472
5	-1	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199

```
kable(tb_combine) %>% kable_styling_fc_wide()
```

ID	var_one	rho1var	rho2var	rho3var	rho4var	rho5var	rho6var	rho7var	rho8var	rho9var	rho10var
1	-1	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199
2	1	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086
3	-1	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632
4	1	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472
5	-1	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199

2.1.1.6 Tibble Row and Column and Summarize

Show what is in the table: 1, column and row names; 2, contents inside table.

```
tb_iris <- as_tibble(iris)
print(rownames(tb_iris))
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12"  "13"
##  [14] "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"  "24"  "25"  "26"
##  [27] "27"  "28"  "29"  "30"  "31"  "32"  "33"  "34"  "35"  "36"  "37"  "38"  "39"
##  [40] "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48"  "49"  "50"  "51"  "52"
##  [53] "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"
##  [66] "66"  "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"  "78"
##  [79] "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"  "89"  "90"  "91"
##  [92] "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"  "100" "101" "102" "103" "104"
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	4.3	3.0	1.1	0.1
setosa	4.4	2.9	1.4	0.2
setosa	4.4	3.0	1.3	0.2
setosa	4.4	3.2	1.3	0.2
setosa	4.5	2.3	1.3	0.3
setosa	4.6	3.1	1.5	0.2
setosa	4.6	3.4	1.4	0.3
setosa	4.6	3.6	1.0	0.2
setosa	4.6	3.2	1.4	0.2
setosa	4.7	3.2	1.3	0.2

```
## [105] "105" "106" "107" "108" "109" "110" "111" "112" "113" "114" "115" "116" "117"
## [118] "118" "119" "120" "121" "122" "123" "124" "125" "126" "127" "128" "129" "130"
## [131] "131" "132" "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143"
## [144] "144" "145" "146" "147" "148" "149" "150"
```

```
colnames(tb_iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
colnames(tb_iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
summary(tb_iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

2.1.1.7 Sorting and Rank

2.1.1.7.1 Sorting

- dplyr arrange desc reverse
- dplyr sort

```
# Sort in Ascending Order
tb_iris %>% select(Species, Sepal.Length, everything()) %>%
  arrange(Species, Sepal.Length) %>% head(10) %>%
  kable() %>% kable_styling_fc()
```

```
# Sort in Descending Order
tb_iris %>% select(Species, Sepal.Length, everything()) %>%
  arrange(desc(Species), desc(Sepal.Length)) %>% head(10) %>%
  kable() %>% kable_styling_fc()
```

2.1.1.7.2 Create a Ranking Variable We use dplyr's ranking functions to generate different types of ranking variables.

The example below demonstrates the differences between the functions row_number(), min_rank(), and dense_rank().

- *row_number*: Given 10 observations, generates index from 1 to 10, ties are given different ranks.
- *min_rank*: Given 10 observations, generates rank where second-rank ties are both given "silver", and the 4th highest ranked variable not given medal.

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
virginica	7.9	3.8	6.4	2.0
virginica	7.7	3.8	6.7	2.2
virginica	7.7	2.6	6.9	2.3
virginica	7.7	2.8	6.7	2.0
virginica	7.7	3.0	6.1	2.3
virginica	7.6	3.0	6.6	2.1
virginica	7.4	2.8	6.1	1.9
virginica	7.3	2.9	6.3	1.8
virginica	7.2	3.6	6.1	2.5
virginica	7.2	3.2	6.0	1.8

Ranking variable

Species	Sepal.Length	row_number	min_rank	dense_rank
setosa	5.1	2	2	2
setosa	4.9	5	5	4
setosa	4.7	7	7	5
setosa	4.6	8	8	6
setosa	5.0	3	3	3
setosa	5.4	1	1	1
setosa	4.6	9	8	6
setosa	5.0	4	3	3
setosa	4.4	10	10	7
setosa	4.9	6	5	4

- *dense_rank*: Given 10 observations, generates rank where second-rank ties are both given "silver" (2nd rank), and the 4th highest ranked variable is given "bronze" (3rd rank), there are no gaps between ranks.

Note that we have "desc(var_name)" in order to generate the variable based on descending sort of the the "var_name" variable.

```
tb_iris %>%
  select(Species, Sepal.Length) %>% head(10) %>%
  mutate(row_number = row_number(desc(Sepal.Length)),
         min_rank = min_rank(desc(Sepal.Length)),
         dense_rank = dense_rank(desc(Sepal.Length))) %>%
  kable(caption = "Ranking variable") %>% kable_styling_fc()
```

2.1.1.8 REconTools Summarize over Tible

Use R4Econ's summary tool.

```
df_summ_stats <- REconTools::ff_summ_percentiles(tb_iris)
kable(t(df_summ_stats)) %>% kable_styling_fc_wide()
```

stats	n	unique	NAobs	ZEROobs	mean	sd	cv	min	p01	p05	p10	p25	p50	p75	p90	p95	p99	max
Petal.Length	150	43	0	0	3.758000	1.7652982	0.4697441	1.0	1.149	1.300	1.4	1.6	4.35	5.1	5.80	6.100	6.700	6.9
Petal.Width	150	22	0	0	1.199333	0.7622377	0.6355511	0.1	0.100	0.200	0.2	0.3	1.30	1.8	2.20	2.300	2.500	2.5
Sepal.Length	150	35	0	0	5.843333	0.8280661	0.1417113	4.3	4.400	4.600	4.8	5.1	5.80	6.4	6.90	7.255	7.700	7.9
Sepal.Width	150	23	0	0	3.057333	0.4358663	0.1425642	2.0	2.200	2.345	2.5	2.8	3.00	3.3	3.61	3.800	4.151	4.4

2.1.2 Generate Categorical Variables

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Cuts a continuous var to a categorical var with labels

cars	mpg_grp	mpg
Cadillac Fleetwood	10<=mpg<20	10.4
Lincoln Continental	10<=mpg<20	10.4
Camaro Z28	10<=mpg<20	13.3
Mazda RX4 Wag	20<=mpg<30	21.0
Hornet 4 Drive	20<=mpg<30	21.4
Volvo 142E	20<=mpg<30	21.4
Honda Civic	30<=mpg<40	30.4
Lotus Europa	30<=mpg<40	30.4
Fiat 128	30<=mpg<40	32.4
Mazda RX4	NA	NA

2.1.2.1 Cut Continuous Variable to Categorical Variable

We have a continuous variable, we cut it with explicitly specified cuts to generate a categorical variable, and label it. We will use base::cut().

```
# break points to specific
fl_min_mpg <- min(mtcars$mpg)
fl_max_mpg <- max(mtcars$mpg)
ar_fl_cuts <- c(10, 20, 30, 40)
# generate labels
ar_st_cuts_lab <- c("10<=mpg<20", "20<=mpg<30", "30<=mpg<40")
# generate new variable
mtcars_cate <- mtcars %>%
  tibble::rownames_to_column(var = "cars") %>%
  mutate(mpg_grp = base::cut(mpg,
      breaks = ar_fl_cuts,
      labels = ar_st_cuts_lab,
      # if right is FALSE, interval is closed on the left
      right = FALSE
    )
  ) %>% select(cars, mpg_grp, mpg) %>%
  arrange(mpg) %>% group_by(mpg_grp) %>%
  slice_head(n=3)
# Display
st_caption <- "Cuts a continuous var to a categorical var with labels"
kable(mtcars_cate,
    caption = st_caption
) %>% kable_styling_fc()
```

2.1.2.2 Factor, Label, Cross and Graph

Generate a Scatter plot with different colors representing different categories. There are multiple underlying factor/categorical variables, for example two binary variables. Generate scatter plot with colors for the combinations of these two binary variables.

We combine here the *vs* and *am* variables from the *mtcars* dataset. *vs* is engine shape, *am* is auto or manual shift. We will generate a scatter plot of *mpg* and *qsec* over four categories with different colors.

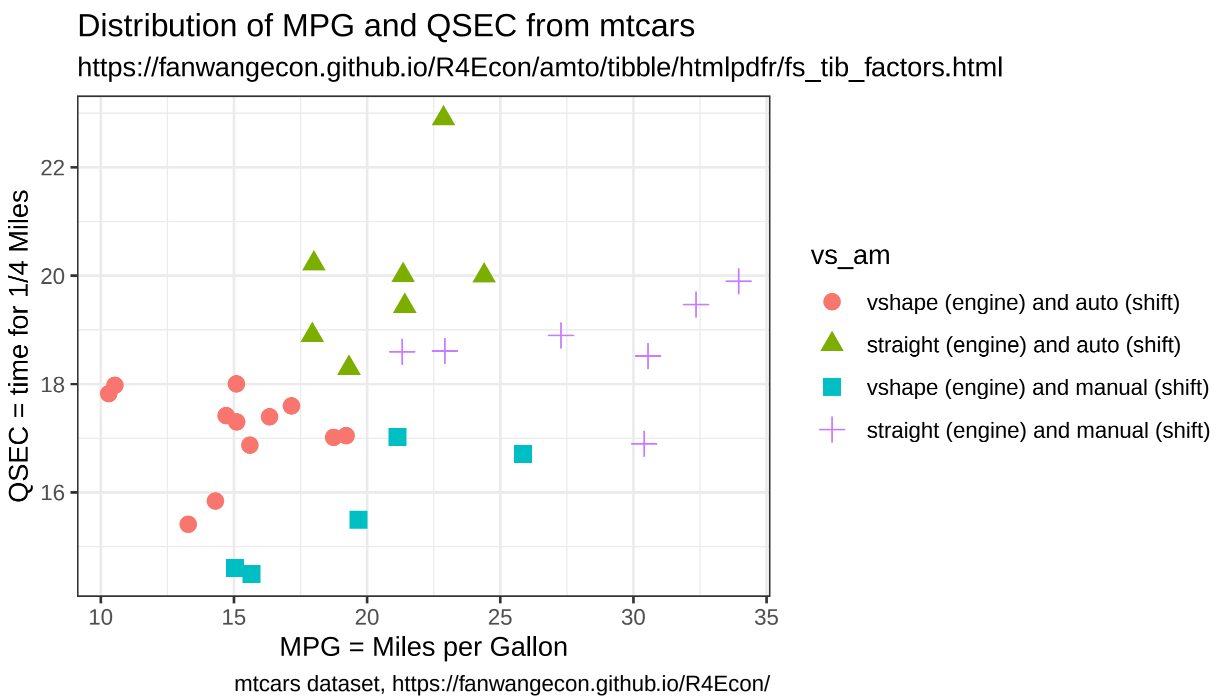
- *am*: Transmission (0 = automatic, 1 = manual)
- *vs*: Engine (0 = V-shaped, 1 = straight)
- *mpg*: miles per galon
- *qsec*: 1/4 mile time

```
# First make sure these are factors
tb_mtcars <- as_tibble(mtcars) %>%
  mutate(vs = as_factor(vs), am = as_factor(am))
```

```
# Second Label the Factors
am_levels <- c(auto_shift = "0", manual_shift = "1")
vs_levels <- c(vshaped_engine = "0", straight_engine = "1")
tb_mtcars <- tb_mtcars %>%
  mutate(vs = fct_recode(vs, !!!vs_levels),
         am = fct_recode(am, !!!am_levels))

# Third Combine Factors
tb_mtcars_selected <- tb_mtcars %>%
  mutate(vs_am = fct_cross(vs, am, sep='_', keep_empty = FALSE)) %>%
  select(mpg, qsec, vs_am)

# relabel interaction variables
am_vs_levels <- c("vshape (engine) and auto (shift)" = "vshaped_engine_auto_shift",
                  "vshape (engine) and manual (shift)" = "vshaped_engine_manual_shift",
                  "straight (engine) and auto (shift)" = "straight_engine_auto_shift",
                  "straight (engine) and manual (shift)" = "straight_engine_manual_shift")
tb_mtcars_selected <- tb_mtcars_selected %>%
  mutate(vs_am = fct_recode(vs_am, !!!am_vs_levels))

# Show
print(tb_mtcars_selected[1:10,])
```

Now we generate scatter plot based on the combined factors

```
# Labeling
st_title <- paste0('Distribution of MPG and QSEC from mtcars')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/amto/tibble/htmlpdfr/fs_tib_factors.html')
st_caption <- paste0('mtcars dataset, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'MPG = Miles per Gallon'
st_y_label <- 'QSEC = time for 1/4 Miles'

# Graphing
plt_mtcars_scatter <-
  ggplot(tb_mtcars_selected,
         aes(x=mpg, y=qsec, colour=vs_am, shape=vs_am)) +
  geom_jitter(size=3, width = 0.15) +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption) +
  theme_bw()

# show
print(plt_mtcars_scatter)
```



2.1.3 Drawly Random Rows

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

2.1.3.1 Draw Random Subset of Sample

- r random discrete

We have a sample of N individuals in some dataframe. Draw without replacement a subset $M < N$ of rows.

```
# parameters, it_M < it_N
it_N <- 10
it_M <- 5

# Draw it_m from indexed list of it_N
set.seed(123)
ar_it_rand_idx <- sample(it_N, it_M, replace=FALSE)

# dataframe
df_full <- as_tibble(matrix(rnorm(4,mean=0,sd=1), nrow=it_N, ncol=4)) %>% rowid_to_column(var = "ID"

# random Subset
df_rand_sub_a <- df_full[ar_it_rand_idx,]

# Random subset also
df_rand_sub_b <- df_full[sample(dim(df_full)[1], it_M, replace=FALSE),]

# Print
# Display
kable(df_full) %>% kable_styling_fc()
```

```
kable(df_rand_sub_a) %>% kable_styling_fc()
```

```
kable(df_rand_sub_b) %>% kable_styling_fc()
```

ID	V1	V2	V3	V4
1	0.1292877	0.4609162	0.1292877	0.4609162
2	1.7150650	-1.2650612	1.7150650	-1.2650612
3	0.4609162	0.1292877	0.4609162	0.1292877
4	-1.2650612	1.7150650	-1.2650612	1.7150650
5	0.1292877	0.4609162	0.1292877	0.4609162
6	1.7150650	-1.2650612	1.7150650	-1.2650612
7	0.4609162	0.1292877	0.4609162	0.1292877
8	-1.2650612	1.7150650	-1.2650612	1.7150650
9	0.1292877	0.4609162	0.1292877	0.4609162
10	1.7150650	-1.2650612	1.7150650	-1.2650612

ID	V1	V2	V3	V4
3	0.4609162	0.1292877	0.4609162	0.1292877
10	1.7150650	-1.2650612	1.7150650	-1.2650612
2	1.7150650	-1.2650612	1.7150650	-1.2650612
8	-1.2650612	1.7150650	-1.2650612	1.7150650
6	1.7150650	-1.2650612	1.7150650	-1.2650612

2.1.3.2 Random Subset of Panel

There are N individuals, each could be observed M times, but then select a subset of rows only, so each person is randomly observed only a subset of times. Specifically, there there are 3 unique students with student ids, and the second variable shows the random dates in which the student showed up in class, out of the 10 classes available.

```
# Define
it_N <- 3
it_M <- 10
svr_id <- 'student_id'

# dataframe
set.seed(123)
df_panel_rand <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(date = row_number()) %>%
  ungroup() %>% mutate(in_class = case_when(rnorm(n(),mean=0,sd=1) < 0 ~ 1, TRUE ~ 0)) %>%
  dplyr::filter(in_class == 1) %>% select(!!sym(svr_id), date) %>%
  rename(date_in_class = date)

# Print
kable(df_panel_rand) %>% kable_styling_fc()
```

2.1.4 Generate Variables Conditional On Others

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

ID	V1	V2	V3	V4
5	0.1292877	0.4609162	0.1292877	0.4609162
3	0.4609162	0.1292877	0.4609162	0.1292877
9	0.1292877	0.4609162	0.1292877	0.4609162
1	0.1292877	0.4609162	0.1292877	0.4609162
4	-1.2650612	1.7150650	-1.2650612	1.7150650

student_id	date_in_class
1	1
1	2
1	8
1	9
1	10
2	5
2	8
2	10
3	1
3	2
3	3
3	4
3	5
3	6
3	9

2.1.4.1 Categorical Variable based on Several Variables

Given several other variables, and generate a new variable when these varaibles satisfy conditions. Note that case_when are ifelse type statements. So below

1. group one is below 16 MPG
2. when do *qsec >= 20* second line that is elseif, only those that are *>=16* are considered here
3. then think about two dimensional mpg and qsec grid, the lower-right area, give another category to manual cars in that group

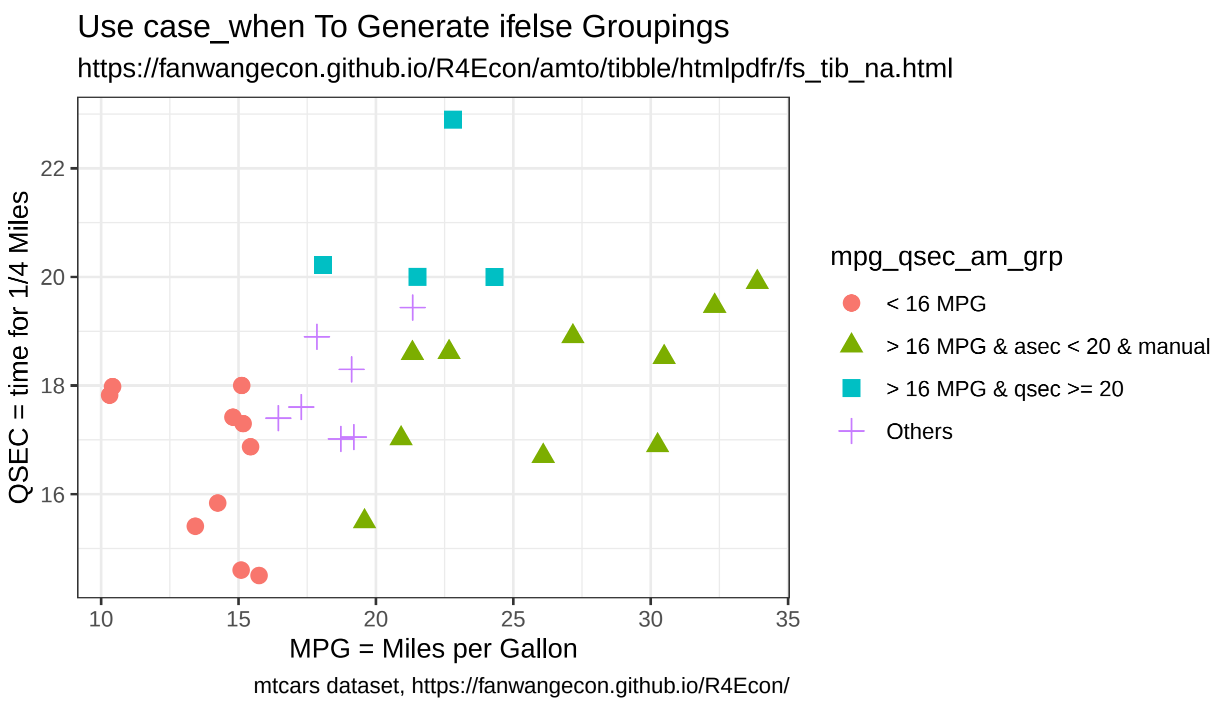
First, we generate categorical variables based on the characteristics of several variables.

```
# Get mtcars
df_mtcars <- mtcars

# case_when with mtcars
df_mtcars <- df_mtcars %>%
    mutate(
        mpg_qsec_am_grp =
            case_when(
                mpg < 16 ~ "< 16 MPG",
                qsec >= 20 ~ "> 16 MPG & qsec >= 20",
                am == 1 ~ "> 16 MPG & asec < 20 & manual",
                TRUE ~ "Others"
            )
    )
```

Now we generate scatter plot based on the combined factors

```
# Labeling
st_title <- paste0("Use case_when To Generate ifelse Groupings")
st_subtitle <- paste0(
    "https://fanwangecon.github.io/",
    "R4Econ/amto/tibble/htmlpdfr/fs_tib_na.html"
)
st_caption <- paste0(
    "mtcars dataset, ",
    "https://fanwangecon.github.io/R4Econ/"
)
st_x_label <- "MPG = Miles per Gallon"
st_y_label <- "QSEC = time for 1/4 Miles"

# Graphing
```

```
plt_mtcars_casewhen_scatter <-
    ggplot(
        df_mtcars,
        aes(
            x = mpg, y = qsec,
            colour = mpg_qsec_am_grp,
            shape = mpg_qsec_am_grp
        )
    ) +
    geom_jitter(size = 3, width = 0.15) +
    labs(
        title = st_title, subtitle = st_subtitle,
        x = st_x_label, y = st_y_label, caption = st_caption
    ) +
    theme_bw()

# show
print(plt_mtcars_casewhen_scatter)
```



2.1.4.2 Categorical Variables based on one Continuous Variable

We generate one categorical variable for gear, based on "continuous" gear values. Note that the same categorical label appears for gear is 3 as well as gear is 5.

```
# Generate a categorical variable
df_mtcars <- df_mtcars %>%
    mutate(gear_cate = case_when(
        gear == 3 ~ "gear is 3",
        gear == 4 ~ "gear is 4",
        gear == 5 & hp <= 110 ~ "gear 5 hp les sequal 110",
        gear == 5 & hp > 110 & hp <= 200 ~ "gear 5 hp 110 to 130",
        TRUE ~ "otherwise"
    ))
# Tabulate
df_mtcars_gear_tb <- df_mtcars %>%
```

Categorical from continuous with non-continuous values matching to same key

gear	gear 5 hp 110 to 130	gear 5 hp les sequal 110	gear is 3	gear is 4	otherwise
3	NA	NA	15	NA	NA
4	NA	NA	NA	12	NA
5	2	1	NA	NA	2

```
  group_by(gear_cate, gear) %>%
  tally() %>%
  spread(gear_cate, n)
# Display
st_title <- "Categorical from continuous with non-continuous values matching to same key"
df_mtcars_gear_tb %>% kable(caption = st_title) %>%
  kable_styling_fc()
```

2.1.4.3 Generate NA values if Variables have Certain Value

In the example below, in one line:

1. generate a random standard normal vector
2. two set na methods:
    - if the value of the standard normal is negative, set value to -999, otherwise MPG, replace the value *-999* with *NA*
    - case_when only with type specific NA values
    - Assigning NA yields error in case_when
    - note we need to conform NA to type
3. generate new categorical variable based on NA condition using is.na with both string and numeric NAs jointly considered.
    - fake NA string to be printed on chart

```
# Get mtcars
df_mtcars <- mtcars

# Make some values of mpg randomly NA
# the NA has to conform to the type of the remaining values for the new variable
# NA_real_, NA_character_, NA_integer_, NA_complex_
set.seed(2341)
df_mtcars <- df_mtcars %>%
    mutate(mpg_wth_NA1 = na_if(
        case_when(
            rnorm(n(), mean = 0, sd = 1) < 0 ~ -999,
            TRUE ~ mpg
        ),
        -999
    )) %>%
    mutate(mpg_wth_NA2 = case_when(
        rnorm(n(), mean = 0, sd = 1) < 0 ~ NA_real_,
        TRUE ~ mpg
    )) %>%
    mutate(mpg_wth_NA3 = case_when(
        rnorm(n(), mean = 0, sd = 1) < 0 ~ NA_character_,
        TRUE ~ "shock > 0 string"
    ))

# Generate New Variables based on if mpg_wth_NA is NA or not
# same variable as above, but now first a category based on if NA
# And we generate a fake string "NA" variable, this is not NA
# the String NA allows for it to be printed on figure
```

	mpg	mpg_wth_NA1	mpg_wth_NA2	mpg_wth_NA3
Mazda RX4	NA	NA	NA	shock > 0 string
Mazda RX4 Wag	21.0	21.0	21.0	NA
Datsun 710	22.8	NA	NA	NA
Hornet 4 Drive	21.4	NA	21.4	NA
Hornet Sportabout	18.7	NA	18.7	NA
Valiant	18.1	18.1	NA	shock > 0 string
Duster 360	14.3	14.3	NA	shock > 0 string
Merc 240D	24.4	NA	24.4	NA
Merc 230	22.8	22.8	22.8	NA
Merc 280	19.2	19.2	NA	NA
Merc 280C	17.8	NA	NA	NA
Merc 450SE	16.4	16.4	16.4	NA
Merc 450SL	17.3	NA	NA	shock > 0 string

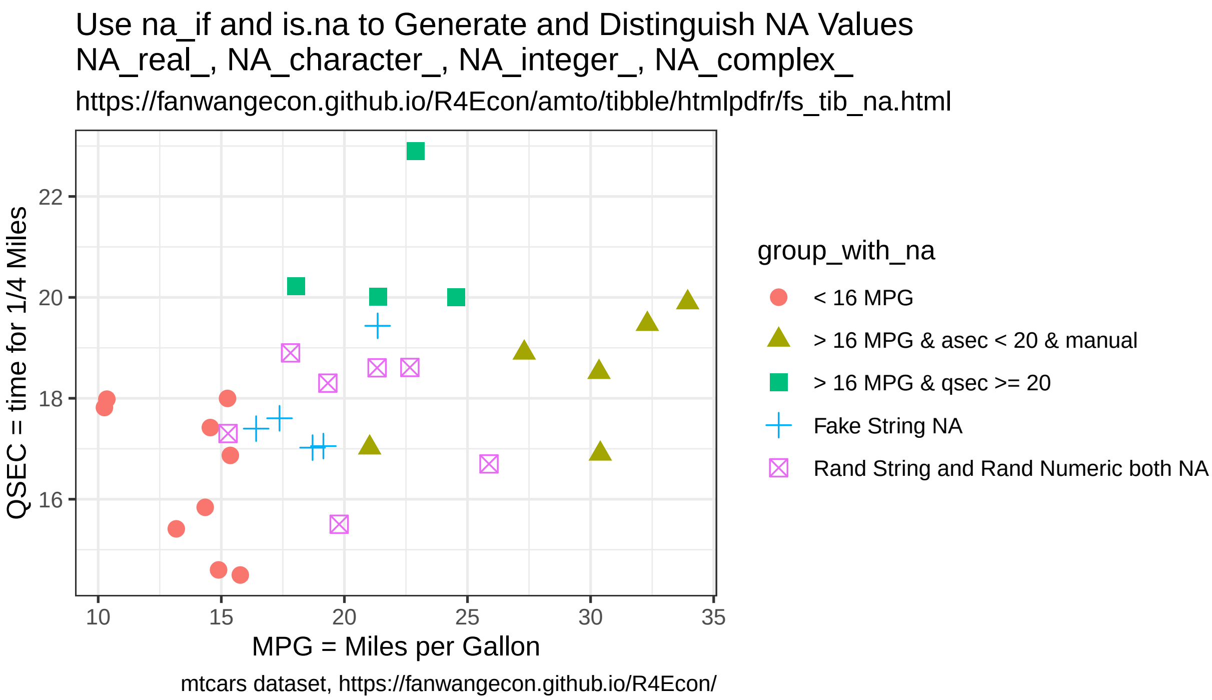
```
df_mtcars <- df_mtcars %>%
    mutate(
        group_with_na =
            case_when(
                is.na(mpg_wth_NA2) & is.na(mpg_wth_NA3) ~
                    "Rand String and Rand Numeric both NA",
                mpg < 16 ~ "< 16 MPG",
                qsec >= 20 ~ "> 16 MPG & qsec >= 20",
                am == 1 ~ "> 16 MPG & asec < 20 & manual",
                TRUE ~ "Fake String NA"
            )
    )

# show
kable(head(df_mtcars %>% select(starts_with("mpg")), 13)) %>%
    kable_styling_fc()
```

```
# # Setting to NA
# df.reg.use <- df.reg.guat %>% filter(!!sym(var.mth) != 0)
# df.reg.use.log <- df.reg.use
# df.reg.use.log[which(is.nan(df.reg.use$prot.imputed.log)),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==Inf),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==-Inf),] = NA
# df.reg.use.log <- df.reg.use.log %>% drop_na(prot.imputed.log)
# # df.reg.use.log$prot.imputed.log
```

Now we generate scatter plot based on the combined factors, but now with the NA category

```
# Labeling
st_title <- paste0(
    "Use na_if and is.na to Generate and Distinguish NA Values\n",
    "NA_real_, NA_character_, NA_integer_, NA_complex_"
)
st_subtitle <- paste0(
    "https://fanwangecon.github.io/",
    "R4Econ/amto/tibble/htmlpdfr/fs_tib_na.html"
)
st_caption <- paste0(
    "mtcars dataset, ",
    "https://fanwangecon.github.io/R4Econ/"
)
st_x_label <- "MPG = Miles per Gallon"
```

```
st_y_label <- "QSEC = time for 1/4 Miles"

# Graphing
plt_mtcars_ifisna_scatter <-
    ggplot(
        df_mtcars,
        aes(
            x = mpg, y = qsec,
            colour = group_with_na,
            shape = group_with_na
        )
    ) +
    geom_jitter(size = 3, width = 0.15) +
    labs(
        title = st_title, subtitle = st_subtitle,
        x = st_x_label, y = st_y_label, caption = st_caption
    ) +
    theme_bw()

# show
print(plt_mtcars_ifisna_scatter)
```



2.1.4.4 Approximate Values Comparison

- r values almost the same
- all.equal

From numeric approximation, often values are very close, and should be set to equal. Use *isTRUE(all.equal)*. In the example below, we randomly generates four arrays. Two of the arrays have slightly higher variance, two arrays have slightly lower variance. They sd are to be 10 times below or 10 times above the tolerance comparison level. The values are not the same in any of the columns, but by allowing for almost true given some tolerance level, in the low standard deviation case, the values differences are within tolerance, so they are equal.

This is an essential issue when dealing with optimization results.

```
# Set tolerance
tol_lvl <- 1.5e-3
sd_lower_than_tol <- tol_lvl / 10
sd_higher_than_tol <- tol_lvl * 10

# larger SD
set.seed(123)
mt_runif_standard <- matrix(rnorm(10, mean = 0, sd = sd_higher_than_tol), nrow = 5, ncol = 2)

# small SD
set.seed(123)
mt_rnorm_small_sd <- matrix(rnorm(10, mean = 0, sd = sd_lower_than_tol), nrow = 5, ncol = 2)

# Generates Random Matirx
tb_rnorm_runif <- as_tibble(cbind(mt_rnorm_small_sd, mt_runif_standard))

# Are Variables the same, not for strict comparison
tb_rnorm_runif_approxi_same <- tb_rnorm_runif %>%
    mutate(
        V1_V2_ALMOST_SAME =
            case_when(
                isTRUE(all.equal(V1, V2, tolerance = tol_lvl)) ~
                    paste0("TOL=", sd_lower_than_tol, ", SAME ALMOST"),
                TRUE ~
                    paste0("TOL=", sd_lower_than_tol, ", NOT SAME ALMOST")
            )
    ) %>%
    mutate(
        V3_V4_ALMOST_SAME =
            case_when(
                isTRUE(all.equal(V3, V4, tolerance = tol_lvl)) ~
                    paste0("TOL=", sd_higher_than_tol, ", SAME ALMOST"),
                TRUE ~
                    paste0("TOL=", sd_higher_than_tol, ", NOT SAME ALMOST")
            )
    )

# Pring
kable(tb_rnorm_runif_approxi_same) %>% kable_styling_fc_wide()
```

V1	V2	V3	V4	V1_V2_ALMOST_SAME	V3_V4_ALMOST_SAME
-0.0000841	0.0002573	-0.0084071	0.0257260	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
-0.0000345	0.0000691	-0.0034527	0.0069137	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
0.0002338	-0.0001898	0.0233806	-0.0189759	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
0.0000106	-0.0001030	0.0010576	-0.0103028	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
0.0000194	-0.0000668	0.0019393	-0.0066849	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST

2.1.5 String Dataframes

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

2.1.5.1 List of Strings to Tibble Datfare

There are several lists of strings, store them as variables in a dataframe.

```
# Sting data inputs
ls_st_abc <- c('a', 'b', 'c')
ls_st_efg <- c('e', 'f', 'g')
```

id	var1	var2	var3
1	a	e	o
2	b	f	p
3	c	g	q

```
ls_st_opq <- c('o', 'p', 'q')
mt_str = cbind(ls_st_abc, ls_st_efg, ls_st_opq)

# Column Names
ar_st_varnames <- c('id','var1','var2','var3')

# Combine to tibble, add name col1, col2, etc.
tb_st_combine <- as_tibble(mt_str) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~c(ar_st_varnames))

# Display
kable(tb_st_combine) %>% kable_styling_fc()
```

2.1.5.2 Find and Replace

Find and Replace in Dataframe.

```
# if string value is contained in variable
("bridex.B" %in% (df.reg.out.all$vars_var.y))
# if string value is not contained in variable:
# 1. type is variable name
# 2. Toyota|Mazda are strings to be excluded
filter(mtcars, !grepl('Toyota|Mazda', type))

# filter does not contain string
rs_hgt_prot_log_tidy %>% filter(!str_detect(term, 'prot'))
```

2.2 Counting Observation

2.2.1 Counting and Tabulations

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

2.2.1.1 Tabulate Two Categorial Variables

First, we tabulate a dataset, and show categories as rows, and display frequencies.

```
# We use the mtcars dataset
tb_tab_joint <- mtcars %>%
    group_by(gear, am) %>%
    tally()
# Display
tb_tab_joint %>%
    kable(caption = "cross tabulation, stacked") %>%
    kable_styling_fc()
```

We can present this as cross tabs.

```
# We use the mtcars dataset
tb_cross_tab <- mtcars %>%
    group_by(gear, am) %>%
```

cross tabulation, stacked

gear	am	n
3	0	15
4	0	4
4	1	8
5	1	5

cross tabulation

gear	0	1
3	15	NA
4	4	8
5	NA	5

```
    tally() %>%
    spread(am, n)
# Display
tb_cross_tab %>%
    kable(caption = "cross tabulation") %>%
    kable_styling_fc()
```

2.2.1.2 Tabulate Once Each Distinct Subgroup

We have two variables variables, am and mpg, the mpg values are not unique. We want to know how many unique mpg levels are there for each am group. We use the dplyr::distinct function to achieve this.

```
tb_dist_tab <- mtcars %>%
    # .keep_all to keep all variables
    distinct(am, mpg, .keep_all = TRUE) %>%
    group_by(am) %>%
    tally()
# Display
tb_dist_tab %>%
    kable(caption = "Tabulate distinct groups") %>%
    kable_styling_fc()
```

2.2.1.3 Expanding to Panel

There are N individuals, each observed for Y_i years. We start with a dataframe where individuals are the unit of observation, we expand this to a panel with a row for each of the years that the individual is in the survey for.

Algorithm:

1. generate testing frame, the individual attribute dataset with invariant information over panel
2. uncount, duplicate rows by years in survey
3. group and generate sorted index
4. add indiviual specific stat year to index

First, we construct the dataframe where each row is an individual.

Tabulate distinct groups

am	n
0	16
1	12

ID	ar_years_in_survey	ar_start_yaer	ar_end_year
1	2	1	2
2	3	2	4
3	1	3	3
4	10	1	10
5	2	1	2
6	5	1	5

ID	ar_start_yaer	ar_end_year	yr_in_survey	calendar_year
1	1	2	1	1
1	1	2	2	2
2	2	4	1	2
2	2	4	2	3
2	2	4	3	4
3	3	3	1	3
4	1	10	1	1
4	1	10	2	2
4	1	10	3	3
4	1	10	4	4

```
# 1. Array of Years in the Survey
ar_years_in_survey <- c(2, 3, 1, 10, 2, 5)
ar_start_yaer <- c(1, 2, 3, 1, 1, 1)
ar_end_year <- c(2, 4, 3, 10, 2, 5)
mt_combine <- cbind(ar_years_in_survey, ar_start_yaer, ar_end_year)

# This is the individual attribute dataset, attributes that are invariant acrosss years
tb_indi_attributes <- as_tibble(mt_combine) %>% rowid_to_column(var = "ID")

# Display
tb_indi_attributes %>%
    head(10) %>%
    kable() %>%
    kable_styling_fc()
```

Second, we change the dataframe so that each unit of observation is an individual in an year. This means we will duplicate the information in the prior table, so if an individual appears for 4 years in the survey, we will now have four rows for this individual. We generate a new variable that is the calendar year. This is now a panel dataset.

```
# 2. Sort and generate variable equal to sorted index
tb_indi_panel <- tb_indi_attributes %>% uncount(ar_years_in_survey)

# 3. Panel now construct exactly which year in survey, note that all needed is sort index
# Note sorting not needed, all rows identical now
tb_indi_panel <- tb_indi_panel %>%
    group_by(ID) %>%
    mutate(yr_in_survey = row_number())

tb_indi_panel <- tb_indi_panel %>%
    mutate(calendar_year = yr_in_survey + ar_start_yaer - 1)

# Show results Head 10
tb_indi_panel %>%
    head(10) %>%
    kable() %>%
    kable_styling_fc()
```

2.3 Sorting, Indexing, Slicing

2.3.1 Sorting

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

2.3.1.1 Generate Sorted Index within Group with Repeating Values

There is a variable, sort by this variable, then generate index from 1 to N representing sorted values of this index. If there are repeating values, still assign index, different index each value.

- r generate index sort
- dplyr mutate equals index

```
# Sort and generate variable equal to sorted index
df_iris <- iris %>% arrange(Sepal.Length) %>%
              mutate(Sepal.Len.Index = row_number()) %>%
              select(Sepal.Length, Sepal.Len.Index, everything())

# Show results Head 10
df_iris %>% head(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

Sepal.Length	Sepal.Len.Index	Sepal.Width	Petal.Length	Petal.Width	Species
4.3	1	3.0	1.1	0.1	setosa
4.4	2	2.9	1.4	0.2	setosa
4.4	3	3.0	1.3	0.2	setosa
4.4	4	3.2	1.3	0.2	setosa
4.5	5	2.3	1.3	0.3	setosa
4.6	6	3.1	1.5	0.2	setosa
4.6	7	3.4	1.4	0.3	setosa
4.6	8	3.6	1.0	0.2	setosa
4.6	9	3.2	1.4	0.2	setosa
4.7	10	3.2	1.3	0.2	setosa

2.3.1.2 Populate Value from Lowest Index to All other Rows

We would like to calculate for example the ratio of each individual's highest to the the person with the lowest height in a dataset. We first need to generated sorted index from lowest to highest, and then populate the lowest height to all rows, and then divide.

Search Terms:

- r spread value to all rows from one row
- r other rows equal to the value of one row
- Conditional assignment of one variable to the value of one of two other variables
- dplyr mutate conditional
- dplyr value from one row to all rows
- dplyr mutate equal to value in another cell

Links:

- see: dplyr rank
- see: dplyr case_when

2.3.1.2.1 Short Method: mutate and min We just want the lowest value to be in its own column, so that we can compute various statistics using the lowest value variable and the original variable.

```
# 1. Sort
df_iris_m1 <- iris %>% mutate(Sepal.Len.Lowest.all = min(Sepal.Length)) %>%
                select(Sepal.Length, Sepal.Len.Lowest.all, everything())


# Show results Head 10
df_iris_m1 %>% head(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

Sepal.Length	Sepal.Len.Lowest.all	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	4.3	3.5	1.4	0.2	setosa
4.9	4.3	3.0	1.4	0.2	setosa
4.7	4.3	3.2	1.3	0.2	setosa
4.6	4.3	3.1	1.5	0.2	setosa
5.0	4.3	3.6	1.4	0.2	setosa
5.4	4.3	3.9	1.7	0.4	setosa
4.6	4.3	3.4	1.4	0.3	setosa
5.0	4.3	3.4	1.5	0.2	setosa
4.4	4.3	2.9	1.4	0.2	setosa
4.9	4.3	3.1	1.5	0.1	setosa

2.3.1.2.2 Long Method: row_number and case_when This is the long method, using row_number, and case_when. The benefit of this method is that it generates several intermediate variables that might be useful. And the key final step is to set a new variable (A=*Sepal.Len.Lowest.all*) equal to another variable's (B=*Sepal.Length*'s) value at the index that satisfies condition based a third variable (C=*Sepal.Len.Index*).

```
# 1. Sort
# 2. generate index
# 3. value at lowest index (case_when)
# 4. spread value from lowest index to other rows
# Note step 4 does not require step 3
df_iris_m2 <- iris %>% arrange(Sepal.Length) %>%
              mutate(Sepal.Len.Index = row_number()) %>%
              mutate(Sepal.Len.Lowest.one =
                       case_when(row_number()==1 ~ Sepal.Length)) %>%
              mutate(Sepal.Len.Lowest.all =
                       Sepal.Length[Sepal.Len.Index==1]) %>%
              select(Sepal.Length, Sepal.Len.Index,
                     Sepal.Len.Lowest.one, Sepal.Len.Lowest.all)


# Show results Head 10
df_iris_m2 %>% head(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

2.3.1.3 Generate Sorted Index based on Deviations

Generate Positive and Negative Index based on Ordered Deviation from some Number.

There is a variable that is continuous, substract a number from this variable, and generate index based on deviations. Think of the index as generating intervals indicating where the value lies. 0th index indicates the largest value in sequence that is smaller than or equal to number x, 1st index indicates the smallest value in sequence that is larger than number x.

Sepal.Length	Sepal.Len.Index	Sepal.Len.Lowest.one	Sepal.Len.Lowest.all
4.3	1	4.3	4.3
4.4	2	NA	4.3
4.4	3	NA	4.3
4.4	4	NA	4.3
4.5	5	NA	4.3
4.6	6	NA	4.3
4.6	7	NA	4.3
4.6	8	NA	4.3
4.6	9	NA	4.3
4.7	10	NA	4.3

The solution below is a little bit convoluated and long, there is likely a much quicker way. The process below shows various intermediary outputs that help arrive at deviation index *Sepal.Len.Devi.Index* from initial sorted index *Sepal.Len.Index*.

search:

- dplyr arrange ignore na
- dplyr index deviation from order number sequence
- dplyr index below above
- dplyr index order below above value

```
# 1. Sort and generate variable equal to sorted index
# 2. Plus or minus deviations from some value
# 3. Find the zero, which means, the number closests to zero including zero from the negative side
# 4. Find the index at the highest zero and below deviation point
# 5. Difference of zero index and original sorted index
sc_val_x <- 4.65
df_iris_deviate <- iris %>% arrange(Sepal.Length) %>%
              mutate(Sepal.Len.Index = row_number()) %>%
              mutate(Sepal.Len.Devi = (Sepal.Length - sc_val_x)) %>%
              mutate(Sepal.Len.Devi.Neg =
                       case_when(Sepal.Len.Devi <= 0 ~ (-1)*(Sepal.Len.Devi))) %>%
              arrange((Sepal.Len.Devi.Neg), desc(Sepal.Len.Index)) %>%
              mutate(Sepal.Len.Index.Zero =
                       case_when(row_number() == 1 ~ Sepal.Len.Index)) %>%
              mutate(Sepal.Len.Devi.Index =
                       Sepal.Len.Index - Sepal.Len.Index.Zero[row_number() == 1]) %>%
              arrange(Sepal.Len.Index) %>%
              select(Sepal.Length, Sepal.Len.Index, Sepal.Len.Devi,
                     Sepal.Len.Devi.Neg, Sepal.Len.Index.Zero, Sepal.Len.Devi.Index)


# Show results Head 10
df_iris_deviate %>% head(20) %>%
  kable() %>%
  kable_styling_fc_wide()
```

2.3.2 Group, Sort and Slice

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Sepal.Length	Sepal.Len.Index	Sepal.Len.Devi	Sepal.Len.Devi.Neg	Sepal.Len.Index.Zero	Sepal.Len.Devi.Index
4.3	1	-0.35	0.35	NA	-8
4.4	2	-0.25	0.25	NA	-7
4.4	3	-0.25	0.25	NA	-6
4.4	4	-0.25	0.25	NA	-5
4.5	5	-0.15	0.15	NA	-4
4.6	6	-0.05	0.05	NA	-3
4.6	7	-0.05	0.05	NA	-2
4.6	8	-0.05	0.05	NA	-1
4.6	9	-0.05	0.05	9	0
4.7	10	0.05	NA	NA	1
4.7	11	0.05	NA	NA	2
4.8	12	0.15	NA	NA	3
4.8	13	0.15	NA	NA	4
4.8	14	0.15	NA	NA	5
4.8	15	0.15	NA	NA	6
4.8	16	0.15	NA	NA	7
4.9	17	0.25	NA	NA	8
4.9	18	0.25	NA	NA	9
4.9	19	0.25	NA	NA	10
4.9	20	0.25	NA	NA	11

2.3.2.1 Sort in Ascending and Descending Orders

We sort the mtcars dataset, sorting in *ascending* order by cyl, and in *descending* order by mpg. Using *arrange*, *desc(disp)* means sorting the disp variable in descending order. In the table shown below, *cyc* is increasing, and *disp* id decreasing within each *cyc* group.

```
kable(mtcars %>%
  arrange(cyl, desc(disp)) %>%
    # Select and filter to reduce display clutter
    select(cyl, disp, mpg)) %>%
  kable_styling_fc()
```

2.3.2.2 Get Highest Values from Groups

There is a dataframe with a grouping variable with N unique values, for example N classes. Find the top three highest scoring students from each class. In the example below, group by *cyl* and get the cars with the highest and second highest *mpg* cars in each *cyl* group.

```
# use mtcars: slice_head gets the lowest sorted value
df_groupby_top_mpg <- mtcars %>%
  rownames_to_column(var = "car") %>%
  arrange(cyl, desc(mpg)) %>%
  group_by(cyl) %>%
  slice_head(n=3) %>%
  select(car, cyl, mpg, disp, hp)

# display
kable(df_groupby_top_mpg) %>% kable_styling_fc()
```

2.3.2.3 Differences in Within-group Sorted Value

We first take the largest N values in M groups, then we difference between the ranked top values in each group.

We have N classes, and M students in each class. We first select the 3 students with the highest scores from each class, then we take the difference between 1st and 2nd, and the difference between the 2nd and the 3rd students.

Note that when are using descending sort, so *lead* means the next value in descending sequencing, and *lag* means the last value which was higher in descending order.

	cyl	disp	mpg
Merc 240D	4	146.7	24.4
Merc 230	4	140.8	22.8
Volvo 142E	4	121.0	21.4
Porsche 914-2	4	120.3	26.0
Toyota Corona	4	120.1	21.5
Datsun 710	4	108.0	22.8
Lotus Europa	4	95.1	30.4
Fiat X1-9	4	79.0	27.3
Fiat 128	4	78.7	32.4
Honda Civic	4	75.7	30.4
Toyota Corolla	4	71.1	33.9
Hornet 4 Drive	6	258.0	21.4
Valiant	6	225.0	18.1
Merc 280	6	167.6	19.2
Merc 280C	6	167.6	17.8
Mazda RX4	6	160.0	NA
Mazda RX4 Wag	6	160.0	21.0
Ferrari Dino	6	145.0	19.7
Cadillac Fleetwood	8	472.0	10.4
Lincoln Continental	8	460.0	10.4
Chrysler Imperial	8	440.0	14.7
Pontiac Firebird	8	400.0	19.2
Hornet Sportabout	8	360.0	18.7
Duster 360	8	360.0	14.3
Ford Pantera L	8	351.0	15.8
Camaro Z28	8	350.0	13.3
Dodge Challenger	8	318.0	15.5
AMC Javelin	8	304.0	15.2
Maserati Bora	8	301.0	15.0
Merc 450SE	8	275.8	16.4
Merc 450SL	8	275.8	17.3
Merc 450SLC	8	275.8	15.2

car	cyl	mpg	disp	hp
Toyota Corolla	4	33.9	71.1	65
Fiat 128	4	32.4	78.7	66
Honda Civic	4	30.4	75.7	52
Hornet 4 Drive	6	21.4	258.0	110
Mazda RX4 Wag	6	21.0	160.0	110
Ferrari Dino	6	19.7	145.0	175
Pontiac Firebird	8	19.2	400.0	175
Hornet Sportabout	8	18.7	360.0	175
Merc 450SL	8	17.3	275.8	180

car	cyl	mpg	disp	hp	mpg_diff_higher_minus_lower	mpg_diff_lower_minus_higher
Toyota Corolla	4	33.9	71.1	65	1.5	NA
Fiat 128	4	32.4	78.7	66	2.0	-1.5
Honda Civic	4	30.4	75.7	52	NA	-2.0
Hornet 4 Drive	6	21.4	258.0	110	0.4	NA
Mazda RX4 Wag	6	21.0	160.0	110	1.3	-0.4
Ferrari Dino	6	19.7	145.0	175	NA	-1.3
Pontiac Firebird	8	19.2	400.0	175	0.5	NA
Hornet Sportabout	8	18.7	360.0	175	1.4	-0.5
Merc 450SL	8	17.3	275.8	180	NA	-1.4

```
# We use what we just created in the last block.
df_groupby_top_mpg_diff <- df_groupby_top_mpg %>%
  group_by(cyl) %>%
  mutate(mpg_diff_higher_minus_lower = mpg - lead(mpg)) %>%
  mutate(mpg_diff_lower_minus_higher = mpg - lag(mpg))

# display
kable(df_groupby_top_mpg_diff)  %>% kable_styling_fc()
```

2.4 Advanced Group Aggregation

2.4.1 Cumulative Statistics within Group

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

2.4.1.1 Cumulative Mean

There is a dataset where there are different types of individuals, perhaps household size, that is the grouping variable. Within each group, we compute the incremental marginal propensity to consume for each additional check. We now also want to know the average propensity to consume up to each check considering all allocated checks. We needed to calculatet this for Nygaard, Sørensen and Wang (2021). This can be dealt with by using the cumall function.

Use the df_hgt_wgt as the testing dataset. In the example below, group by individual id, sort by survey month, and cumulative mean over the protein variable.

In the protein example

First select the testing dataset and variables.

```
# Load the REconTools Dataset df_hgt_wgt
data("df_hgt_wgt")
# str(df_hgt_wgt)

# Select several rows
df_hgt_wgt_sel <- df_hgt_wgt %>%
  filter(S.country == "Cebu") %>%
  select(indi.id, svymthRound, prot)
```

Second, arrange, groupby, and cumulative mean. The protein variable is protein for each survey month, from month 2 to higher as babies grow. The protein intake observed is increasing quickly, hence, the cumulative mean is lower than the observed value for the survey month of the baby.

```
# Group by indi.id and sort by protein
df_hgt_wgt_sel_cummean <- df_hgt_wgt_sel %>%
  arrange(indi.id, svymthRound) %>%
  group_by(indi.id) %>%
```

indi.id	svymthRound	prot	prot_cummean
17	0	0.5	0.5000000
17	2	0.7	0.6000000
17	4	0.5	0.5666667
17	6	0.5	0.5500000
17	8	6.1	1.6600000
17	10	5.0	2.2166667
17	12	6.4	2.8142857
17	14	20.1	4.9750000
17	16	20.1	6.6555556
17	18	23.0	8.2900000
17	20	24.9	9.8000000
17	22	20.1	10.6583333
17	24	10.1	10.6153846
17	102	NA	NA
17	138	NA	NA
17	187	NA	NA
17	224	NA	NA
17	258	NA	NA
18	0	1.2	1.2000000
18	2	4.7	2.9500000
18	4	17.2	7.7000000
18	6	18.6	10.4250000
18	8	NA	NA
18	10	16.8	NA
18	12	NA	NA
18	14	NA	NA
18	16	NA	NA
18	18	NA	NA
18	20	NA	NA
18	22	15.7	NA
18	24	22.5	NA
18	102	NA	NA
18	138	NA	NA
18	187	NA	NA
18	224	NA	NA
18	258	NA	NA

```
  mutate(prot_cummean = cummean(prot))

# display results
REconTools::ff_summ_percentiles(df_hgt_wgt_sel_cummean)
# display results
df_hgt_wgt_sel_cummean %>% filter(indi.id %in% c(17, 18)) %>%
  kable() %>% kable_styling_fc()
```

Third, in the basic implementation above, if an incremental month has NA, no values computed at that point or after. This is the case for individual 18 above. To ignore NA, we have, from this. Note how results for individual 18 changes.

```
# https://stackoverflow.com/a/49906718/8280804
# Group by indi.id and sort by protein
df_hgt_wgt_sel_cummean_noNA <- df_hgt_wgt_sel %>%
  arrange(indi.id, svymthRound) %>%
  group_by(indi.id, isna = is.na(prot)) %>%
  mutate(prot_cummean = ifelse(isna, NA, cummean(prot)))
```

indi.id	svymthRound	prot	isna	prot_cummean
17	0	0.5	FALSE	0.5000000
17	2	0.7	FALSE	0.6000000
17	4	0.5	FALSE	0.5666667
17	6	0.5	FALSE	0.5500000
17	8	6.1	FALSE	1.6600000
17	10	5.0	FALSE	2.2166667
17	12	6.4	FALSE	2.8142857
17	14	20.1	FALSE	4.9750000
17	16	20.1	FALSE	6.6555556
17	18	23.0	FALSE	8.2900000
17	20	24.9	FALSE	9.8000000
17	22	20.1	FALSE	10.6583333
17	24	10.1	FALSE	10.6153846
17	102	NA	TRUE	NA
17	138	NA	TRUE	NA
17	187	NA	TRUE	NA
17	224	NA	TRUE	NA
17	258	NA	TRUE	NA
18	0	1.2	FALSE	1.2000000
18	2	4.7	FALSE	2.9500000
18	4	17.2	FALSE	7.7000000
18	6	18.6	FALSE	10.4250000
18	8	NA	TRUE	NA
18	10	16.8	FALSE	11.7000000
18	12	NA	TRUE	NA
18	14	NA	TRUE	NA
18	16	NA	TRUE	NA
18	18	NA	TRUE	NA
18	20	NA	TRUE	NA
18	22	15.7	FALSE	12.3666667
18	24	22.5	FALSE	13.8142857
18	102	NA	TRUE	NA
18	138	NA	TRUE	NA
18	187	NA	TRUE	NA
18	224	NA	TRUE	NA
18	258	NA	TRUE	NA

```
# display results
df_hgt_wgt_sel_cummean_noNA %>% filter(indi.id %in% c(17, 18)) %>%
  kable() %>% kable_styling_fc()
```

2.4.2 Groups Statistics

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

2.4.2.1 Aggrgate Groups only Unique Group and Count

There are two variables that are numeric, we want to find all the unique groups of these two variables in a dataset and count how many times each unique group occurs

- r unique occurrence of numeric groups
- How to add count of unique values by group to R data.frame

hgt0	wgt0	n_obs_group
40	2000	122
45	2000	4586
45	4000	470
50	2000	9691
50	4000	13106
55	2000	126
55	4000	1900
60	6000	18

```
# Numeric value combinations unique Groups
vars.group <- c('hgt0', 'wgt0')

# dataset subsetting
df_use <- df_hgt_wgt %>% select(!!!syms(c(vars.group))) %>%
            mutate(hgt0 = round(hgt0/5)*5, wgt0 = round(wgt0/2000)*2000) %>%
            drop_na()

# Group, count and generate means for each numeric variables
# mutate_at(vars.group, funs(as.factor(.))) %>%
df.group.count <- df_use %>% group_by(!!!syms(vars.group)) %>%
                    arrange(!!!syms(vars.group)) %>%
                    summarise(n_obs_group=n())

# Show results Head 10
df.group.count %>% kable() %>% kable_styling_fc()
```

2.4.2.2 Aggrgate Groups only Unique Group Show up With Means

Several variables that are grouping identifiers. Several variables that are values which mean be unique for each group members. For example, a Panel of income for N households over T years with also household education information that is invariant over time. Want to generate a dataset where the unit of observation are households, rather than household years. Take average of all numeric variables that are household and year specific.

A complicating factor potentially is that the number of observations differ within group, for example, income might be observed for all years for some households but not for other households.

- r dplyr aggregate group average
- Aggregating and analyzing data with dplyr
- column can't be modified because it is a grouping variable
- see also: Aggregating and analyzing data with dplyr

```
# In the df_hgt_wgt from R4Econ, there is a country id, village id,
# and individual id, and various other statistics
vars.group <- c('S.country', 'vil.id', 'indi.id')
vars.values <- c('hgt', 'momEdu')

# dataset subsetting
df_use <- df_hgt_wgt %>% select(!!!syms(c(vars.group, vars.values)))

# Group, count and generate means for each numeric variables
df.group <- df_use %>% group_by(!!!syms(vars.group)) %>%
            arrange(!!!syms(vars.group)) %>%
            summarise_if(is.numeric,
                         funs(mean = mean(., na.rm = TRUE),
                              sd = sd(., na.rm = TRUE),
                              n = sum(is.na(.)==0)))
```

```
# Show results Head 10
df.group %>% head(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

S.country	vil.id	indi.id	hgt_mean	momEdu_mean	hgt_sd	momEdu_sd	hgt_n	momEdu_n
Cebu	1	1	61.80000	5.3	9.520504	0	7	18
Cebu	1	2	68.86154	7.1	9.058931	0	13	18
Cebu	1	3	80.45882	9.4	29.894231	0	17	18
Cebu	1	4	88.10000	13.9	35.533166	0	18	18
Cebu	1	5	97.70556	11.3	41.090366	0	18	18
Cebu	1	6	87.49444	7.3	35.586439	0	18	18
Cebu	1	7	90.79412	10.4	38.722385	0	17	18
Cebu	1	8	68.45385	13.5	10.011961	0	13	18
Cebu	1	9	86.21111	10.4	35.126057	0	18	18
Cebu	1	10	87.67222	10.5	36.508127	0	18	18

```
# Show results Head 10
df.group %>% tail(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

S.country	vil.id	indi.id	hgt_mean	momEdu_mean	hgt_sd	momEdu_sd	hgt_n	momEdu_n
Guatemala	14	2014	66.97000	NaN	8.967974	NA	10	0
Guatemala	14	2015	71.71818	NaN	11.399984	NA	11	0
Guatemala	14	2016	66.33000	NaN	9.490352	NA	10	0
Guatemala	14	2017	76.40769	NaN	14.827871	NA	13	0
Guatemala	14	2018	74.55385	NaN	12.707846	NA	13	0
Guatemala	14	2019	70.47500	NaN	11.797390	NA	12	0
Guatemala	14	2020	60.28750	NaN	7.060036	NA	8	0
Guatemala	14	2021	84.96000	NaN	15.446193	NA	10	0
Guatemala	14	2022	79.38667	NaN	15.824749	NA	15	0
Guatemala	14	2023	66.50000	NaN	8.613113	NA	8	0

2.4.3 One Variable Group Summary

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

There is a categorical variable (based on one or the interaction of multiple variables), there is a continuous variable, obtain statistics for the continuous variable conditional on the categorical variable, but also unconditionally.

Store results in a matrix, but also flatten results wide to row with appropriate keys/variable-names for all group statistics.

Pick which statistics to be included in final wide row

2.4.3.1 Build Program

```
# Single Variable Group Statistics (also generate overall statistics)
ff_summ_by_group_summ_one <- function(
  df, vars.group, var.numeric, str.stats.group = 'main',
  str.stats.specify = NULL, boo.overall.stats = TRUE){

  # List of statistics
  # https://rdrr.io/cran/dplyr/man/summarise.html
  strs.center <- c('mean', 'median')
  strs.spread <- c('sd', 'IQR', 'mad')
```

```
strs.range <- c('min', 'max')
strs.pos <- c('first', 'last')
strs.count <- c('n_distinct')

# Grouping of Statistics
if (missing(str.stats.specify)) {
  if (str.stats.group == 'main') {
    strs.all <- c('mean', 'min', 'max', 'sd')
  }
  if (str.stats.group == 'all') {
    strs.all <- c(strs.center, strs.spread, strs.range, strs.pos, strs.count)
  }
} else {
  strs.all <- str.stats.specify
}

# Start Transform
df <- df %>% drop_na() %>%
  mutate(!!(var.numeric) := as.numeric(!!sym(var.numeric)))

# Overall Statistics
if (boo.overall.stats) {
  df.overall.stats <- df %>%
    summarize_at(vars(var.numeric), funs(!!!strs.all))
  if (length(strs.all) == 1) {
    # give it a name, otherwise if only one stat, name of stat not saved
    df.overall.stats <- df.overall.stats %>%
      rename(!!strs.all := !!sym(var.numeric))
  }
  names(df.overall.stats) <-
    paste0(var.numeric, '.', names(df.overall.stats))
}

# Group Sort
df.select <- df %>%
  group_by(!!!syms(vars.group)) %>%
  arrange(!!!syms(c(vars.group, var.numeric)))

# Table of Statistics
df.table.grp.stats <- df.select %>%
  summarize_at(vars(var.numeric), funs(!!!strs.all))

# Add Stat Name
if (length(strs.all) == 1) {
  # give it a name, otherwise if only one stat, name of stat not saved
  df.table.grp.stats <- df.table.grp.stats %>%
    rename(!!strs.all := !!sym(var.numeric))
}


# Row of Statistics
str.vars.group.combine <- paste0(vars.group, collapse='_')
if (length(vars.group) == 1) {
  df.row.grp.stats <- df.table.grp.stats %>%
    mutate(!!(str.vars.group.combine) :=
             paste0(var.numeric, '.',
                    vars.group, '.g',
```

```
                    (!!!syms(vars.group)))) %>%
      gather(variable, value, -one_of(vars.group)) %>%
      unite(str.vars.group.combine, c(str.vars.group.combine, 'variable')) %>%
      spread(str.vars.group.combine, value)
  } else {
    df.row.grp.stats <- df.table.grp.stats %>%
      mutate(vars.groups.combine :=
               paste0(paste0(vars.group, collapse='.')),
             !!(str.vars.group.combine) :=
               paste0(interaction(!!!(syms(vars.group))))) %>%
      mutate(!!(str.vars.group.combine) :=
               paste0(var.numeric, '.', vars.groups.combine, '.',
                      (!!sym(str.vars.group.combine)))) %>%
      ungroup() %>%
      select(-vars.groups.combine, -one_of(vars.group)) %>%
      gather(variable, value, -one_of(str.vars.group.combine)) %>%
      unite(str.vars.group.combine, c(str.vars.group.combine, 'variable')) %>%
      spread(str.vars.group.combine, value)
  }

  # Clean up name strings
  names(df.table.grp.stats) <-
    gsub(x = names(df.table.grp.stats),pattern = "_", replacement = "\\.")
  names(df.row.grp.stats) <-
    gsub(x = names(df.row.grp.stats),pattern = "_", replacement = "\\.")

  # Return
  list.return <-
    list(df_table_grp_stats = df.table.grp.stats,
         df_row_grp_stats = df.row.grp.stats)

  # Overall Statistics, without grouping
  if (boo.overall.stats) {
    df.row.stats.all <- c(df.row.grp.stats, df.overall.stats)
    list.return <- append(list.return,
                          list(df_overall_stats = df.overall.stats,
                               df_row_stats_all = df.row.stats.all))
  }

  # Return
  return(list.return)

}
```

2.4.3.2 Test

Load data and test

```
# Library
library(tidyverse)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')
```

2.4.3.2.1 Function Testing By Gender Groups Need two variables, a group variable that is a factor, and a numeric

```
vars.group <- 'sex'
var.numeric <- 'hgt'
```

```
df.select <- df %>% select(one_of(vars.group, var.numeric)) %>% drop_na()
```

Main Statistics:

```
# Single Variable Group Statistics
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.group = 'main')$df_table_grp_stats
```

Specify Two Specific Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.specify = c('mean', 'sd'))$df_table_grp_stats
```

Specify One Specific Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.specify = c('mean'))$df_table_grp_stats
```

2.4.3.2.2 Function Testing By Country and Gender Groups Need two variables, a group variable that is a factor, and a numeric. Now joint grouping variables.

```
vars.group <- c('S.country', 'sex')
var.numeric <- 'hgt'
```

```
df.select <- df %>% select(one_of(vars.group, var.numeric)) %>% drop_na()
```

Main Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.group = 'main')$df_table_grp_stats
```

Specify Two Specific Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.specify = c('mean', 'sd'))$df_table_grp_stats
```

Specify One Specific Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.specify = c('mean'))$df_table_grp_stats
```

2.4.4 Nested within Group Stats

> Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

By Multiple within Individual Groups Variables, Averages for All Numeric Variables within All Groups of All Group Variables (Long to very Wide). Suppose you have an individual level final outcome. The individual is observed for N periods, where each period the inputs differ. What inputs impacted the final outcome?

Suppose we can divide N periods in which the individual is in the data into a number of years, a number of semi-years, a number of quarters, or uneven-staggered lengths. We might want to generate averages across individuals and within each of these different possible groups averages of inputs.

Then we want to version of the data where each row is an individual, one of the variables is the final outcome, and the other variables are these different averages: averages for the 1st, 2nd, 3rd year in which indivdiual is in data, averages for 1st, …, final quarter in which indivdiual is in data.

2.4.4.1 Build Function

This function takes as inputs:

1. **vars.not.groups2avg**: a list of variables that are not the within-indivdiual or across-individual grouping variables, but the variables we want to average over. Withnin indivdiual grouping averages will be calculated for these variables using the not-listed variables as within indivdiual groups (excluding vars.indi.grp groups).
2. **vars.indi.grp**: a list or individual variables, and also perhaps villages, province, etc id variables that are higher than individual ID. Note the groups are are ACROSS individual higher level group variables.
3. the remaining variables are all within individual grouping variables.

the function output is a dataframe:

1. each row is an individual
2. initial variables individual ID and across individual groups from *vars.indi.grp*.
3. other variables are all averages for the variables in *vars.not.groups2avg*
    - if there are 2 within individual group variables, and the first has 3 groups (years), the second has 6 groups (semi-years), then there would be 9 average variables.
    - each average variables has the original variable name from vars.not.groups2avg plus the name of the within individual grouping variable, and at the end 'c_x', where x is a integer representing the category within the group (if 3 years, x=1, 2, 3)

```
# Data Function
# https://fanwangecon.github.io/R4Econ/summarize/summ/ByGroupsSummWide.html
f.by.groups.summ.wide <- function(df.groups.to.average,
                                  vars.not.groups2avg,
                                  vars.indi.grp = c('S.country','ID'),
                                  display=TRUE) {

# 1. generate categoricals for full year (m.12), half year (m.6), quarter year (m.4)
# 2. generate categoricals also for uneven years (m12t14) using
#  stagger (+2 rather than -1)
# 3. reshape wide to long, so that all categorical date groups appear in var=value,
    # and categories in var=variable
# 4. calculate mean for all numeric variables for all date groups
# 5. combine date categorical variable and value, single var:
    # m.12.c1= first year average from m.12 averaging

######## ######## ######## ######## #######
# Step 1
######## ######## ######## ######## #######
# 1. generate categoricals for full year (m.12), half year (m.6), quarter year (m.4)
# 2. generate categoricals also for uneven years (m12t14) using stagger
#  (+2 rather than -1)

######## ######## ######## ######## #######
# S2: reshape wide to long, so that all categorical date groups appear in var=value,
# and categories in var=variable; calculate mean for all
# numeric variables for all date groups
######## ######## ######## ######## #######
df.avg.long <- df.groups.to.average %>%
       gather(variable, value, -one_of(c(vars.indi.grp,
                                         vars.not.groups2avg))) %>%
       group_by(!!!syms(vars.indi.grp), variable, value) %>%
```

```
      summarise_if(is.numeric, funs(mean(., na.rm = TRUE)))

if (display){
  dim(df.avg.long)
  options(repr.matrix.max.rows=10, repr.matrix.max.cols=20)
  print(df.avg.long)
}

######## ######## ######## ######## #######
# S3 combine date categorical variable and value, single var:
# m.12.c1= first year average from m.12 averaging; to do this make
# data even longer first
######## ######## ######## ######## #######

# We already have the averages, but we want them to show up as variables,
    # mean for each group of each variable.
df.avg.allvars.wide <- df.avg.long %>%
   ungroup() %>%
   mutate(all_m_cate = paste0(variable, '_c', value)) %>%
   select(all_m_cate, everything(), -variable, -value) %>%
   gather(variable, value, -one_of(vars.indi.grp), -all_m_cate) %>%
   unite('var_mcate', variable, all_m_cate) %>%
   spread(var_mcate, value)

if (display){
  dim(df.avg.allvars.wide)
  options(repr.matrix.max.rows=10, repr.matrix.max.cols=10)
  print(df.avg.allvars.wide)
}

return(df.avg.allvars.wide)
}
```

2.4.4.2 Test Program

In our sample dataset, the number of nutrition/height/income etc information observed within each country and month of age group are different. We have a panel dataset for children observed over different months of age.

We have two key grouping variables: 1. country: data are observed for guatemala and cebu 2. month-age (survey month round=svymthRound): different months of age at which each individual child is observed

A child could be observed for many months, or just a few months. A child's height information could be observed for more months-of-age than nutritional intake information. We eventually want to run regressions where the outcome is height/weight and the input is nutrition. The regressions will be at the month-of-age level. We need to know how many times different variables are observed at the month-of-age level.

```
# Library
library(tidyverse)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')
```

2.4.4.2.1 Generate Within Individual Groups In the data, children are observed for different number of months since birth. We want to calculate quarterly, semi-year, annual, etc average nutritional intakes. First generate these within-individual grouping variables. We can also generate uneven-staggered calendar groups as shown below.

```r
mth.var <- 'svymthRound'
df.groups.to.average<- df %>%
        filter(!!sym(mth.var) >= 0 & !!sym(mth.var) <= 24)  %>%
        mutate(m12t24=(floor((!!sym(mth.var) - 12) %/% 14) + 1),
               m8t24=(floor((!!sym(mth.var) - 8) %/% 18) + 1),
               m12 = pmax((floor((!!sym(mth.var)-1) %/% 12) + 1), 1),
               m6 = pmax((floor((!!sym(mth.var)-1) %/% 6) + 1), 1),
               m3 = pmax((floor((!!sym(mth.var)-1) %/% 3) + 1), 1))
```

```r
# Show Results
options(repr.matrix.max.rows=30, repr.matrix.max.cols=20)
vars.arrange <- c('S.country','indi.id','svymthRound')
vars.groups.within.indi <- c('m12t24', 'm8t24', 'm12', 'm6', 'm3')
as.tibble(df.groups.to.average %>%
          group_by(!!!syms(vars.arrange)) %>%
          arrange(!!!syms(vars.arrange)) %>%
          select(!!!syms(vars.arrange), !!!syms(vars.groups.within.indi)))
```

2.4.4.2.2 Within Group Averages With the within-group averages created, we can generate averages for all variables within these groups.

```r
vars.not.groups2avg <- c('prot', 'cal')
vars.indi.grp <- c('S.country', 'indi.id')
vars.groups.within.indi <- c('m12t24', 'm8t24', 'm12', 'm6', 'm3')

df.groups.to.average.select <- df.groups.to.average %>%
                        select(one_of(c(vars.indi.grp,
                                        vars.not.groups2avg,
                                        vars.groups.within.indi)))
df.avg.allvars.wide <- f.by.groups.summ.wide(df.groups.to.average.select,
                                          vars.not.groups2avg,
                                          vars.indi.grp, display=FALSE)
```

This is the tabular version of results

```r
dim(df.avg.allvars.wide)
```

```
## [1] 2023   38
```

```r
names(df.avg.allvars.wide)
```

```
##  [1] "S.country"      "indi.id"        "cal_m12_c1"     "cal_m12_c2"
##  [5] "cal_m12t24_c0"  "cal_m12t24_c1"  "cal_m3_c1"      "cal_m3_c2"
##  [9] "cal_m3_c3"      "cal_m3_c4"      "cal_m3_c5"      "cal_m3_c6"
## [13] "cal_m3_c7"      "cal_m3_c8"      "cal_m6_c1"      "cal_m6_c2"
## [17] "cal_m6_c3"      "cal_m6_c4"      "cal_m8t24_c0"   "cal_m8t24_c1"
## [21] "prot_m12_c1"    "prot_m12_c2"    "prot_m12t24_c0" "prot_m12t24_c1"
## [25] "prot_m3_c1"     "prot_m3_c2"     "prot_m3_c3"     "prot_m3_c4"
## [29] "prot_m3_c5"     "prot_m3_c6"     "prot_m3_c7"     "prot_m3_c8"
## [33] "prot_m6_c1"     "prot_m6_c2"     "prot_m6_c3"     "prot_m6_c4"
## [37] "prot_m8t24_c0"  "prot_m8t24_c1"
```

```r
df.avg.allvars.wide[1:20,] %>% kable() %>% kable_styling_fc_wide()
```

S.country	indi.id	cal_m12_c1	cal_m12_c2	cal_m12t24_c0	cal_m12t24_c1	cal_m3_c1	cal_m3_c2	cal_m3_c3	cal_m3_c4	cal_m3_c5
Cebu	1	132.15714	NaN	97.08333	342.6000	9.10	95.50	85.3	315.30	NaN
Cebu	2	90.72857	255.6500	81.46667	240.0286	83.35	12.30	155.1	144.35	228.0
Cebu	3	96.80000	658.8167	31.56667	634.4429	0.50	28.85	57.0	280.95	459.3
Cebu	4	27.45714	371.7000	24.55000	325.0143	4.50	25.95	39.4	45.95	221.2
Cebu	5	101.34286	1080.8500	79.15000	959.9429	14.10	143.80	71.3	161.15	452.6
Cebu	6	185.35714	521.5333	162.23333	493.3286	23.85	184.70	169.1	355.65	653.4
Cebu	7	157.25714	570.9800	145.50000	513.7833	8.30	137.80	407.8	200.40	390.6
Cebu	8	471.92857	844.8333	379.20000	871.0429	158.95	423.00	417.5	861.05	691.3
Cebu	9	32.27143	415.2167	16.58333	373.9571	5.05	10.40	15.1	89.95	142.4
Cebu	10	67.18571	395.2500	68.58333	347.1857	9.55	26.40	164.6	116.90	296.6
Cebu	11	14.90000	245.3833	11.80000	215.1143	0.50	5.20	30.0	31.45	126.2
Cebu	12	453.61429	745.6833	419.51667	733.1857	325.60	483.65	463.0	546.90	766.8
Cebu	13	47.51429	210.2500	36.81667	196.1714	17.45	40.00	28.5	94.60	216.9
Cebu	14	608.85714	924.5833	527.30000	949.3857	259.05	554.30	688.1	973.60	525.5
Cebu	15	74.67143	440.0833	64.21667	396.8429	62.40	39.60	80.1	119.30	292.5
Cebu	16	128.45714	519.9333	90.50000	496.5429	4.80	11.00	205.3	331.15	290.8
Cebu	17	130.78571	718.8667	97.48333	663.4000	5.50	7.80	249.9	319.50	774.6
Cebu	18	172.64000	497.9500	172.64000	497.9500	29.60	234.40	NaN	335.20	NaN
Cebu	19	74.45714	314.7333	80.10000	275.5714	3.65	95.70	171.3	75.60	131.3
Cebu	20	110.90000	583.2000	80.51667	541.7714	7.30	120.65	77.8	221.30	391.2

indi.id	cal_m3_c6	cal_m3_c7	cal_m3_c8	cal_m6_c1	cal_m6_c2	cal_m6_c3	cal_m6_c4	cal_m8t24_c0	cal_m8t24_c1
1	NaN	NaN	NaN	52.300	238.63333	NaN	NaN	52.300	238.6333
2	152.85	305.0	347.60	47.825	147.93333	177.9000	333.4000	47.825	219.7444
3	549.95	612.0	890.85	14.675	206.30000	519.7333	797.9000	14.675	507.9778
4	271.00	581.3	442.85	15.225	43.76667	254.4000	489.0000	15.225	262.3889
5	1345.20	1178.1	1082.00	78.950	131.20000	1047.6667	1114.0333	78.950	764.3000
6	506.50	416.8	523.00	104.275	293.46667	555.4667	487.6000	104.275	445.5111
7	637.10	688.1	569.55	73.050	269.53333	513.8500	609.0667	73.050	457.9375
8	897.95	637.1	972.35	290.975	713.20000	829.0667	860.6000	290.975	800.9556
9	203.60	753.2	594.25	7.725	65.00000	183.2000	647.2333	7.725	298.4778
10	303.00	385.1	541.90	17.975	132.80000	300.8667	489.6333	17.975	307.7667
11	223.05	239.6	330.20	2.850	30.96667	190.7667	300.0000	2.850	173.9111
12	676.85	998.5	677.55	404.625	518.93333	706.8333	784.5333	404.625	670.1000
13	195.15	281.3	186.50	28.725	72.56667	202.4000	218.1000	28.725	164.3556
14	1039.60	800.0	1071.40	406.675	878.43333	868.2333	980.9333	406.675	909.2000
15	237.45	607.8	632.65	51.000	106.23333	255.8000	624.3667	51.000	328.8000
16	354.50	563.3	778.25	7.900	289.20000	333.2667	706.6000	7.900	443.0222
17	892.90	551.9	600.45	6.650	296.30000	853.4667	584.2667	6.650	578.0111
18	NaN	NaN	497.95	132.000	335.20000	NaN	497.9500	132.000	443.7000
19	350.50	304.6	375.75	49.675	107.50000	277.4333	352.0333	49.675	245.6556
20	582.10	466.1	738.85	63.975	173.46667	518.4667	647.9333	63.975	446.6222

indi.id	prot_m12_c1	prot_m12_c2	prot_m12t24_c0	prot_m12t24_c1	prot_m3_c1	prot_m3_c2	prot_m3_c3	prot_m3_c4
1	5.3571429	NaN	4.3666667	11.300000	0.65	3.65	2.6	13.15
2	3.1857143	8.550000	2.7333333	8.171429	3.20	1.25	5.2	4.10
3	4.5000000	21.116667	1.6833333	21.157143	1.05	2.15	2.3	11.40
4	0.8714286	6.850000	0.9000000	5.971429	0.75	1.10	1.2	0.60
5	2.4000000	19.483333	2.3166667	17.114286	1.35	3.00	3.4	2.35
6	8.4000000	15.116667	7.3833333	15.028571	0.85	7.40	9.8	16.25
7	3.3000000	20.440000	2.7833333	18.100000	0.95	1.70	8.6	4.60
8	13.6857143	32.716667	11.0166667	32.285714	3.90	11.35	10.8	27.25
9	0.9571429	18.283333	0.9166667	15.842857	0.50	0.50	0.5	2.10
10	2.0428571	8.466667	1.9333333	7.642857	0.85	0.50	4.9	3.35
11	1.0285714	6.833333	1.1166667	5.928571	0.80	1.70	1.2	0.50
12	14.7714286	22.133333	14.1666667	21.600000	7.40	13.60	26.6	17.40
13	1.9571429	6.800000	1.6666667	6.357143	0.70	1.50	2.1	3.60
14	24.6714286	28.050000	23.0833333	28.928571	15.05	28.85	32.6	26.15
15	2.2571429	10.633333	1.6333333	9.971429	1.65	0.90	1.5	4.60
16	4.6857143	18.083333	4.1000000	16.671429	0.65	2.70	8.7	8.70
17	2.8142857	19.716667	2.2166667	17.814286	0.60	0.50	6.1	5.70
18	11.7000000	19.100000	11.7000000	19.100000	2.95	17.90	NaN	16.80
19	2.5428571	10.466667	2.8833333	9.042857	0.50	2.95	6.7	2.10
20	3.2000000	16.966667	2.1833333	15.871429	0.50	3.85	2.7	5.50

indi.id	prot_m3_c5	prot_m3_c6	prot_m3_c7	prot_m3_c8	prot_m6_c1	prot_m6_c2	prot_m6_c3	prot_m6_c4	prot_m8t24_c0	prot_m8t24_c1
1	NaN	NaN	NaN	NaN	2.150	9.6333333	NaN	NaN	2.150	9.633333
2	5.4	5.15	7.7	13.95	2.225	4.4666667	5.233333	11.866667	2.225	7.188889
3	18.5	18.05	18.0	27.05	1.600	8.3666667	18.200000	24.033333	1.600	16.866667
4	1.8	4.85	10.1	9.75	0.925	0.8000000	3.833333	9.866667	0.925	4.833333
5	7.1	23.15	24.5	19.50	2.175	2.7000000	17.800000	21.166667	2.175	13.888889
6	26.8	14.10	11.4	12.15	4.125	14.1000000	18.333333	11.900000	4.125	14.777778
7	16.4	23.00	21.5	20.65	1.325	5.9333333	19.700000	20.933333	1.325	15.000000
8	42.7	26.45	25.8	37.45	7.625	21.7666667	31.866667	33.566667	7.625	29.066667
9	4.2	10.85	39.5	22.15	0.500	1.5666667	8.633333	27.933333	0.500	12.711111
10	7.5	6.05	9.2	11.00	0.675	3.8666667	6.533333	10.400000	0.675	6.933333
11	3.6	6.35	7.3	8.70	1.250	0.7333333	5.433333	8.233333	1.250	4.800000
12	18.6	20.35	25.6	23.95	10.500	20.4666667	19.766667	24.500000	10.500	21.577778
13	8.3	4.75	6.2	8.40	1.100	3.1000000	5.933333	7.666667	1.100	5.566667
14	12.7	29.75	29.5	33.30	21.950	28.3000000	24.066667	32.033333	21.950	28.133333
15	9.4	5.80	13.6	14.60	1.275	3.5666667	7.000000	14.266667	1.275	8.277778
16	7.8	12.95	25.9	24.45	1.675	8.7000000	11.233333	24.933333	1.675	14.955556
17	20.1	21.55	24.9	15.10	0.550	5.8333333	21.066667	18.366667	0.550	15.088889
18	NaN	NaN	NaN	19.10	10.425	16.8000000	NaN	19.100000	10.425	18.333333
19	3.4	11.40	12.3	12.15	1.725	3.6333333	8.733333	12.200000	1.725	8.188889
20	7.9	20.15	11.8	20.90	2.175	4.5666667	16.066667	17.866667	2.175	12.833333

2.5 Distributional Statistics

2.5.1 Histogram

2.5.1.1 Generate Test Score Dataset

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

- r generate text string as csv
- r tibble matrix hand input

First, we will generate a test score dataset, directly from string. Below we type line by line a dataset with four variables in comma separated (csv) format, where the first row includes the variables names. These texts could be stored in a separate file, or they could be directly included in code and read in as csv

```
ar_test_scores_ec3 <- c(107.72,101.28,105.92,109.31,104.27,110.27,91.92846154,81.8,109.0071429,103.0
ar_test_scores_ec1 <- c(101.72,101.28,99.92,103.31,100.27,104.27,90.23615385,77.8,103.4357143,97.07,
mt_test_scores <- cbind(ar_test_scores_ec1, ar_test_scores_ec3)
ar_st_varnames <- c('course_total_ec1p','course_total_ec3p')
tb_final_twovar <- as_tibble(mt_test_scores) %>% rename_all(~c(ar_st_varnames))
summary(tb_final_twovar)
```

2.5.1.1.1 A Dataset with only Two Continuous Variable

```
##  course_total_ec1p course_total_ec3p
##  Min.   : 40.48    Min.   : 44.23
##  1st Qu.: 76.46    1st Qu.: 79.91
##  Median : 86.35    Median : 89.28
##  Mean   : 83.88    Mean   : 87.90
##  3rd Qu.: 95.89    3rd Qu.:100.75
##  Max.   :104.27    Max.   :112.22
```

```
ff_summ_percentiles(df = tb_final_twovar, bl_statsasrows = TRUE, col2varname = FALSE)
```

```
ar_final_scores <- c(94.28442509,95.68817475,97.25219512,77.89268293,95.08795497,93.27380863,92.3,84
mt_test_scores <- cbind(seq(1,length(ar_final_scores)), ar_final_scores)
ar_st_varnames <- c('index', 'course_final')
tb_onevar <- as_tibble(mt_test_scores) %>% rename_all(~c(ar_st_varnames))
summary(tb_onevar)
```

2.5.1.1.2 A Dataset with one Continuous Variable and Histogram

```
##      index        course_final
##  Min.   : 1.0   Min.   :  2.293
##  1st Qu.:12.5   1st Qu.: 76.372
##  Median :24.0   Median : 86.959
##  Mean   :24.0   Mean   : 82.415
##  3rd Qu.:35.5   3rd Qu.: 94.686
##  Max.   :47.0   Max.   :100.898
```

```
ff_summ_percentiles(df = tb_onevar, bl_statsasrows = TRUE, col2varname = FALSE)
```

```
#load in data empirically by hand
txt_test_data <- "init_prof, later_prof, class_id, exam_score
 'SW', 'SW', 1, 102
 'SW', 'SW', 1, 102
 'SW', 'SW', 1, 101
```

```
'SW', 'SW', 1, 100
'SW', 'SW', 1, 100
'SW', 'SW', 1, 99
'SW', 'SW', 1, 98.5
'SW', 'SW', 1, 98.5
'SW', 'SW', 1, 97
'SW', 'SW', 1, 95
'SW', 'SW', 1, 94
'SW', 'SW', 1, 91
'SW', 'SW', 1, 91
'SW', 'SW', 1, 90
'SW', 'SW', 1, 89
'SW', 'SW', 1, 88.5
'SW', 'SW', 1, 88
'SW', 'SW', 1, 87
'SW', 'SW', 1, 87
'SW', 'SW', 1, 87
'SW', 'SW', 1, 86
'SW', 'SW', 1, 86
'SW', 'SW', 1, 84
'SW', 'SW', 1, 82
'SW', 'SW', 1, 78.5
'SW', 'SW', 1, 76
'SW', 'SW', 1, 72
'SW', 'SW', 1, 70.5
'SW', 'SW', 1, 67.5
'SW', 'SW', 1, 67.5
'SW', 'SW', 1, 67
'SW', 'SW', 1, 63.5
'SW', 'SW', 1, 60
'SW', 'SW', 1, 59
'SW', 'SW', 1, 44.5
'SW', 'SW', 1, 44
'SW', 'SW', 1, 42.5
'SW', 'SW', 1, 40.5
'SW', 'SW', 1, 40.5
'SW', 'SW', 1, 36.5
'SW', 'SW', 1, 35.5
'SW', 'SW', 1, 21.5
'SW', 'SW', 1, 4
'MP', 'MP', 2, 105
'MP', 'MP', 2, 103
'MP', 'MP', 2, 102
'MP', 'MP', 2, 101
'MP', 'MP', 2, 101
'MP', 'MP', 2, 100.5
'MP', 'MP', 2, 100
'MP', 'MP', 2, 99
'MP', 'MP', 2, 97
'MP', 'MP', 2, 97
'MP', 'MP', 2, 97
'MP', 'MP', 2, 97
'MP', 'MP', 2, 96
'MP', 'MP', 2, 95
'MP', 'MP', 2, 91
'MP', 'MP', 2, 89
'MP', 'MP', 2, 85
'MP', 'MP', 2, 84
```

```
'MP', 'MP', 2, 84
'MP', 'MP', 2, 84
'MP', 'MP', 2, 83.5
'MP', 'MP', 2, 82.5
'MP', 'MP', 2, 81.5
'MP', 'MP', 2, 80.5
'MP', 'MP', 2, 80
'MP', 'MP', 2, 77
'MP', 'MP', 2, 77
'MP', 'MP', 2, 75
'MP', 'MP', 2, 75
'MP', 'MP', 2, 71
'MP', 'MP', 2, 70
'MP', 'MP', 2, 68
'MP', 'MP', 2, 63
'MP', 'MP', 2, 56
'MP', 'MP', 2, 56
'MP', 'MP', 2, 55.5
'MP', 'MP', 2, 49.5
'MP', 'MP', 2, 48.5
'MP', 'MP', 2, 47.5
'MP', 'MP', 2, 44.5
'MP', 'MP', 2, 34.5
'MP', 'MP', 2, 29.5
'CA', 'MP', 3, 103
'CA', 'MP', 3, 103
'CA', 'MP', 3, 101
'CA', 'MP', 3, 96.5
'CA', 'MP', 3, 93.5
'CA', 'MP', 3, 93
'CA', 'MP', 3, 93
'CA', 'MP', 3, 92
'CA', 'MP', 3, 90
'CA', 'MP', 3, 90
'CA', 'MP', 3, 89
'CA', 'MP', 3, 86.5
'CA', 'MP', 3, 84.5
'CA', 'MP', 3, 83
'CA', 'MP', 3, 83
'CA', 'MP', 3, 82
'CA', 'MP', 3, 78
'CA', 'MP', 3, 75
'CA', 'MP', 3, 74.5
'CA', 'MP', 3, 70
'CA', 'MP', 3, 54.5
'CA', 'MP', 3, 52
'CA', 'MP', 3, 50
'CA', 'MP', 3, 42
'CA', 'MP', 3, 36.5
'CA', 'MP', 3, 28
'CA', 'MP', 3, 26
'CA', 'MP', 3, 11
'CA', 'SN', 4, 103
'CA', 'SN', 4, 103
'CA', 'SN', 4, 102
'CA', 'SN', 4, 102
'CA', 'SN', 4, 101
'CA', 'SN', 4, 100
```

```
 'CA', 'SN', 4, 98
 'CA', 'SN', 4, 98
 'CA', 'SN', 4, 98
 'CA', 'SN', 4, 95
 'CA', 'SN', 4, 95
 'CA', 'SN', 4, 92.5
 'CA', 'SN', 4, 92
 'CA', 'SN', 4, 91
 'CA', 'SN', 4, 90
 'CA', 'SN', 4, 85.5
 'CA', 'SN', 4, 84
 'CA', 'SN', 4, 82.5
 'CA', 'SN', 4, 81
 'CA', 'SN', 4, 77.5
 'CA', 'SN', 4, 77
 'CA', 'SN', 4, 72
 'CA', 'SN', 4, 71.5
 'CA', 'SN', 4, 69
 'CA', 'SN', 4, 68.5
 'CA', 'SN', 4, 68
 'CA', 'SN', 4, 67
 'CA', 'SN', 4, 65.5
 'CA', 'SN', 4, 62.5
 'CA', 'SN', 4, 62
 'CA', 'SN', 4, 61.5
 'CA', 'SN', 4, 61
 'CA', 'SN', 4, 57.5
 'CA', 'SN', 4, 54
 'CA', 'SN', 4, 52.5
 'CA', 'SN', 4, 51
 'CA', 'SN', 4, 50.5
 'CA', 'SN', 4, 50
 'CA', 'SN', 4, 49
 'CA', 'SN', 4, 43
 'CA', 'SN', 4, 39.5
 'CA', 'SN', 4, 32.5
 'CA', 'SN', 4, 25.5
 'CA', 'SN', 4, 18"

csv_test_data = read.csv(text=txt_test_data, header=TRUE)
ar_st_varnames <- c('first_half_professor',
                    'second_half_professor',
                    'course_id', 'exam_score')
tb_test_data <- as_tibble(csv_test_data) %>%
  rename_all(~c(ar_st_varnames))
summary(tb_test_data)
```

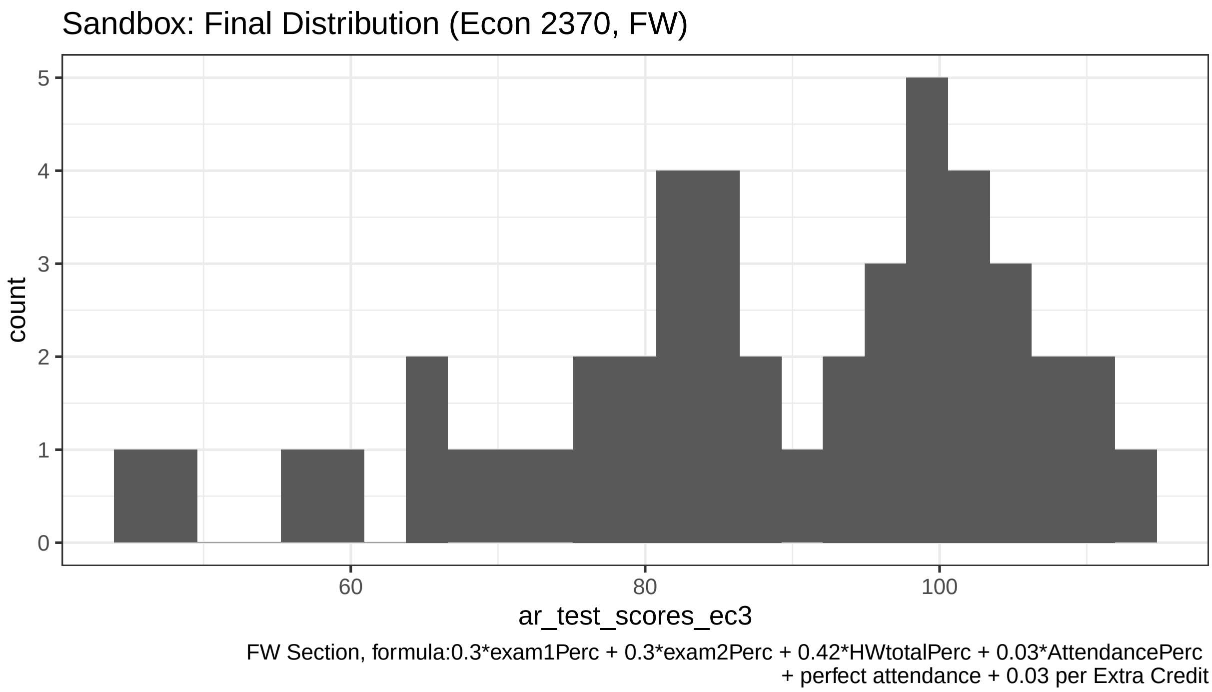
2.5.1.1.3 A Dataset with Multiple Variables

```
##  first_half_professor second_half_professor   course_id       exam_score
##  Length:157           Length:157            Min.   :1.000   Min.   :  4.00
##  Class :character     Class :character      1st Qu.:1.000   1st Qu.: 60.00
##  Mode  :character     Mode  :character      Median :2.000   Median : 82.00
##                                             Mean   :2.465   Mean   : 75.08
##                                             3rd Qu.:4.000   3rd Qu.: 94.00
##                                             Max.   :4.000   Max.   :105.00
```

2.5.1.2 Test Score Distributions

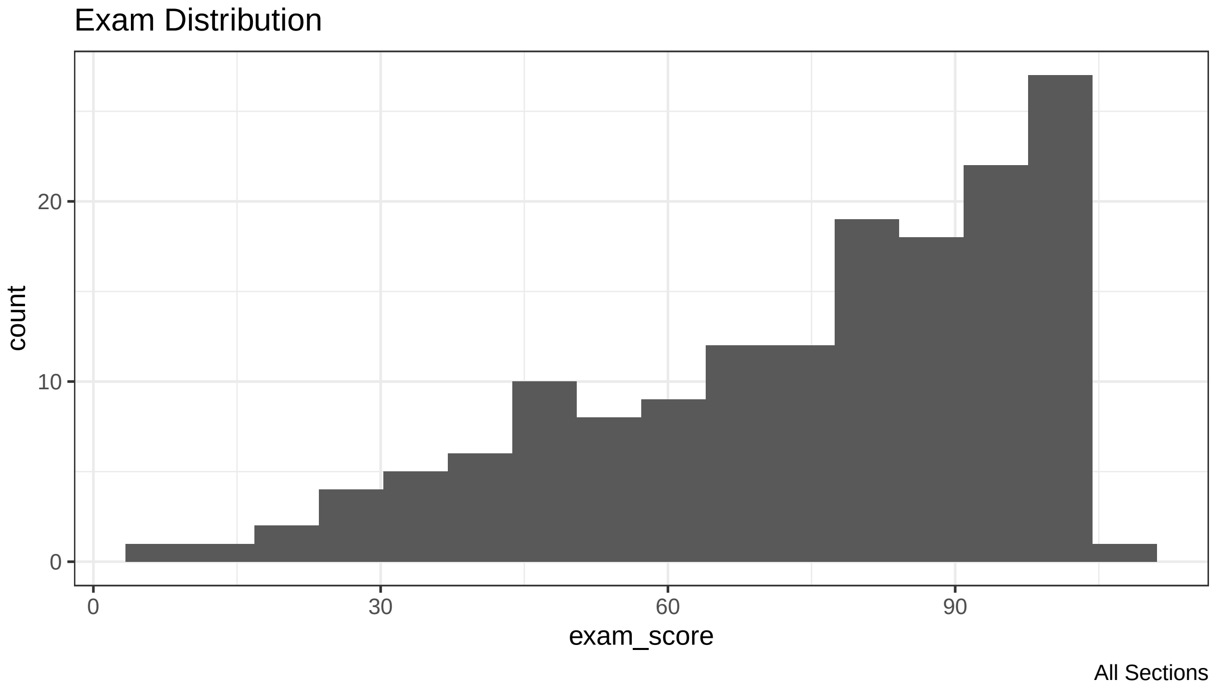
```
ggplot(tb_final_twovar, aes(x=ar_test_scores_ec3)) +
  geom_histogram(bins=25) +
  labs(title = paste0('Sandbox: Final Distribution (Econ 2370, FW)'),
       caption = paste0('FW Section, formula:',
                        '0.3*exam1Perc + 0.3*exam2Perc + ',
                        '0.42*HWtotalPerc + 0.03*AttendancePerc \n',
                        '+ perfect attendance + 0.03 per Extra Credit')) +
  theme_bw()
```

2.5.1.2.1 Histogram




```
ggplot(tb_test_data, aes(x=exam_score)) +
  geom_histogram(bins=16) +
  labs(title = paste0('Exam Distribution'),
       caption = 'All Sections') +
  theme_bw()
```

gear	carb	mpg	cyl	disp
3	1	214	60	2580
3	2	187	80	3600
4	1	228	40	1080
4	4	NA	60	1600
4	4	210	60	1600



2.6 Summarize Multiple Variables

2.6.1 Apply Function Over Multiple Columns and Rows

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

2.6.1.1 Convert Subset of Variables to Numeric

Multiply a subset of variables all by 10. We use dplyr's across function to achieve this.

Note that in the example below, we also use across with group by to include a string array of grouping by variables.

Note: "across() makes it easy to apply the same transformation to multiple columns, allowing you to use select() semantics inside in"data-masking" functions like summarise() and mutate()."

```
# grouping by variables
ar_st_groups <- c("gear", "carb")
# use across to conduct operation over multiple variables
mtcars_3var_times10 <- mtcars %>%
  group_by(across(one_of(ar_st_groups))) %>%
  mutate(across(matches("mpg|cyl|disp"), ~ .x * 10)) %>%
  select(gear, carb, mpg, cyl, disp) %>% head(n=5)
# pring
# Multiply several variables by 10
kable(mtcars_3var_times10 %>% slice_head(n = 5)) %>%
    kable_styling_fc()
```

2.6.1.2 Compute Row-specific Quantiles using Data Across Columns

We want to compute quantiles for each location, based on monthly variations in columns.

First, we generate a table with 12 columns (for months) and 3 rows (for locations).

```
# Generate data, 12 months as columns, and
mt_data_rand <- matrix(rnorm(36, mean=0, sd=1), nrow=3, ncol=12)
it_rows <- seq(1, dim(mt_data_rand)[1])
it_cols <- seq(1, dim(mt_data_rand)[2])
# convert to table, column as month with leading 0
colnames(mt_data_rand) <- paste0('m', sprintf("%02d", it_cols))
tb_data_full <- as_tibble(mt_data_rand, rownames = NA) %>%
    mutate(loc = paste0("loc", sprintf("%02d", row_number()))) %>%
    select(loc, everything())
# Display
kable(tb_data_full) %>% kable_styling_fc_wide()
```

loc	m01	m02	m03	m04	m05	m06	m07	m08	m09	m10	m11	m12
loc01	1.2240818	0.1106827	0.4978505	-0.4727914	-1.0260044	-1.6866933	-1.1381369	-0.2950715	0.8215811	-0.0619117	-0.6947070	2.168956
loc02	0.3598138	-0.5558411	-1.9666172	-1.0678237	-0.7288912	0.8377870	1.2538149	0.8951257	0.6886403	-0.3059627	-0.2079173	1.207962
loc03	0.4007715	1.7869131	0.7013559	-0.2179749	-0.6250393	0.1533731	0.4264642	0.8781335	0.5539177	-0.3804710	-1.2653964	-1.123109

Second, using apply to compute quantiles, row by row

```
# Extract the data components from the tibble, tibble has row and column names
tb_data_only <- tb_data_full %>%
    column_to_rownames(var = "loc") %>%
    select(contains("m"))
# Compute row specific quantiles
ar_quantiles_by_row <- apply(tb_data_only, 1, quantile, probs=0.75)
# Display
print(ar_quantiles_by_row)
```

```
##     loc01     loc02     loc03
## 0.5787831 0.8521217 0.5907772
```

Third, generate matrix of two columns, ID and quantile.

```
# One particular quantil from location
tb_loc_quantile <- as_tibble(ar_quantiles_by_row) %>%
    mutate(loc = names(ar_quantiles_by_row)) %>%
    rename(quantile = value) %>%
    select(loc, everything())
# Display
kable(tb_loc_quantile) %>% kable_styling_fc()
```

2.6.1.3 Compute Row-specific Sums using Data Across Columns

We compute sum over several variables in the mtcars dataset. We will sum over several variables with shared prefix, after adding these prefix first. We introduce an NA value to make sure that we can sum ignoring NA

We sum using three different methods below: (1) purrr:reduce(), (2) base::rowSums(), (3) Manual sum. Note that the rowSums option is able to sum ignoring NA.

```
# we introduce NA value to first row
mtcars[1,1] <- NA
```

loc	quantile
loc01	0.5787831
loc02	0.8521217
loc03	0.5907772

```
# Rename variables, and sum across
mtcars_rowsum <- mtcars %>%
    rename(stats_mpg = mpg, stats_cyl = cyl, stats_hp = hp) %>%
    mutate(
        cs_reduce = purrr::reduce(
            dplyr::pick(contains("stats")),
            `+`
        ),
        cs_rowsum = base::rowSums(
            dplyr::pick(contains("stats")),
            na.rm = TRUE
        ),
        cs_manual = stats_mpg + stats_cyl + stats_hp
    ) %>%
    select(matches("stats|cs"), gear)
# Display
# caption: "sum across columns"
kable(mtcars_rowsum %>% slice_head(n = 5)) %>% kable_styling_fc_wide()
```

	stats_mpg	stats_cyl	stats_hp	cs_reduce	cs_rowsum	cs_manual	gear
Mazda RX4	NA	6	110	NA	116.0	NA	4
Mazda RX4 Wag	21.0	6	110	137.0	137.0	137.0	4
Datsun 710	22.8	4	93	119.8	119.8	119.8	4
Hornet 4 Drive	21.4	6	110	137.4	137.4	137.4	3
Hornet Sportabout	18.7	8	175	201.7	201.7	201.7	3

See this discussion for column sum peed comparisons.

2.6.1.4 Sum Across Rows within Group

Following from the prior section, we now sum across rows within group.

```
# we introduce NA value to first row
# mtcars[1,1] <- NA
# Rename variables, and sum across
mtcars_grpsum <- mtcars_rowsum %>%
    arrange(gear) %>% group_by(gear) %>%
    # srs = sum row sum
    mutate_at(vars(matches("stats|cs")),
        .funs = list(gs = ~sum(., na.rm=TRUE))
    ) %>%
    select(gear, matches("gs")) %>%
    slice_head(n=1)
# Display
# caption: "gs = group sum, cs = col sum over the columns
# with stats as prefix, sum across rows after col sum; gear = 4
# difference for cs-rowsum-gs because it allowed for summing
# ignoring NA for values across columns"
kable(mtcars_grpsum) %>% kable_styling_fc_wide()
```

gear	stats_mpg_gs	stats_cyl_gs	stats_hp_gs	cs_reduce_gs	cs_rowsum_gs	cs_manual_gs
3	241.6	112	2642	2995.6	2995.6	2995.6
4	273.4	56	1074	1287.4	1403.4	1287.4
5	106.9	30	978	1114.9	1114.9	1114.9

2.6.1.5 Replace NA for Multiple Variables

Replace some variables NA by some values, and other variables' NAs by other values.

date	var1	var2	var3	var4	var5
1	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	NA
3	NA	NA	NA	NA	NA

date	var1	var2	var3	var4	var5
1	0	0	99	NA	99
2	0	0	99	NA	99
3	0	0	99	NA	99

```
# Define
it_N <- 3
it_M <- 5
svr_id <- "date"

# NA dataframe, note need to define as NA_real_
# if define as NA, will not be able to replace with 99 column
# would be logical rather than double.
df_NA <- as_tibble(matrix(NA_real_, nrow = it_N, ncol = it_M)) %>%
    rowid_to_column(var = svr_id) %>%
    rename_at(
        vars(starts_with("V")),
        funs(str_replace(., "V", "var"))
    )
kable(df_NA) %>%
    kable_styling_fc()
```

```
# Replace NA
df_NA_replace <- df_NA %>%
    mutate_at(vars(one_of(c("var1", "var2"))), list(~ replace_na(., 0))) %>%
    mutate_at(vars(one_of(c("var3", "var5"))), list(~ replace_na(., 99)))

kable(df_NA_replace) %>%
    kable_styling_fc()
```

2.6.1.6 Cumulative Sum Multiple Variables

Each row is a different date, each column is the profit a firms earns on a date, we want to compute cumulatively how much a person is earning. Also renames variable names below jointly.

```
# Define
it_N <- 3
it_M <- 5
svr_id <- "date"

# random dataframe, daily profit of firms
# dp_fx: daily profit firm ID something
set.seed(123)
df_daily_profit <- as_tibble(matrix(rnorm(it_N * it_M), nrow = it_N, ncol = it_M)) %>%
    rowid_to_column(var = svr_id) %>%
    rename_at(
        vars(starts_with("V")),
        funs(str_replace(., "V", "dp_f"))
    )
kable(df_daily_profit) %>%
    kable_styling_fc()
```

date	dp_f1	dp_f2	dp_f3	dp_f4	dp_f5
1	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715
2	-0.2301775	0.1292877	-1.2650612	1.2240818	0.1106827
3	1.5587083	1.7150650	-0.6868529	0.3598138	-0.5558411

```
# cumulative sum with suffix
df_cumu_profit_suffix <- df_daily_profit %>%
    mutate_at(vars(contains("dp_f")), .funs = list(cumu = ~ cumsum(.)))
kable(df_cumu_profit_suffix) %>%
    kable_styling_fc_wide()
```

date	dp_f1	dp_f2	dp_f3	dp_f4	dp_f5	dp_f1_cumu	dp_f2_cumu	dp_f3_cumu	dp_f4_cumu	dp_f5_cumu
1	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715
2	-0.2301775	0.1292877	-1.2650612	1.2240818	0.1106827	-0.7906531	0.1997961	-0.8041450	0.7784198	0.5114542
3	1.5587083	1.7150650	-0.6868529	0.3598138	-0.5558411	0.7680552	1.9148611	-1.4909979	1.1382337	-0.0443870

```
# cumulative sum variables naming to prefix
df_cumu_profit <- df_cumu_profit_suffix %>%
    rename_at(vars(contains("_cumu")), list(~ paste("cp_f", gsub("_cumu", "", .), sep = ""))) %>%
    rename_at(vars(contains("cp_f")), list(~ gsub("dp_f", "", .)))
kable(df_cumu_profit) %>%
    kable_styling_fc_wide()
```

date	dp_f1	dp_f2	dp_f3	dp_f4	dp_f5	cp_f1	cp_f2	cp_f3	cp_f4	cp_f5
1	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715
2	-0.2301775	0.1292877	-1.2650612	1.2240818	0.1106827	-0.7906531	0.1997961	-0.8041450	0.7784198	0.5114542
3	1.5587083	1.7150650	-0.6868529	0.3598138	-0.5558411	0.7680552	1.9148611	-1.4909979	1.1382337	-0.0443870

Chapter 3

Functions

3.1 Dataframe Mutate

3.1.1 Row Input Functions

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

We want evaluate nonlinear function f(Q__i, y__i, ar__x, ar__y, c, d), where c and d are constants, and ar__x and ar__y are arrays, both fixed. x__i and y__i vary over each row of matrix. We would like to evaluate this nonlinear function concurrently across N individuals. The eventual goal is to find the i specific Q that solves the nonlinear equations.

This is a continuation of R use Apply, Sapply and dplyr Mutate to Evaluate one Function Across Rows of a Matrix

3.1.1.1 Set up Input Arrays

There is a function that takes $M = Q + P$ inputs, we want to evaluate this function N times. Each time, there are M inputs, where all but Q of the M inputs, meaning P of the M inputs, are the same. In particular, $P = Q * N$.

$$M = Q + P = Q + Q * N$$

```
# it_child_count = N, the number of children
it_N_child_cnt = 5
# it_heter_param = Q, number of parameters that are heterogeneous across children
it_Q_hetpa_cnt = 2

# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
ar_nP_A_alpha = c(ar_nN_A, ar_nN_alpha)
ar_nN_N_choice = seq(1,it_N_child_cnt)/sum(seq(1,it_N_child_cnt))

# N by Q varying parameters
mt_nN_by_nQ_A_alpha = cbind(ar_nN_A, ar_nN_alpha, ar_nN_N_choice)

# Convert Matrix to Tibble
ar_st_col_names = c('fl_A', 'fl_alpha', 'fl_N')
tb_nN_by_nQ_A_alpha <- as_tibble(mt_nN_by_nQ_A_alpha) %>% rename_all(~c(ar_st_col_names))

# Show
```

fl_A	fl_alpha	fl_N
-2	0.1	0.0666667
-1	0.3	0.1333333
0	0.5	0.2000000
1	0.7	0.2666667
2	0.9	0.3333333

fl_A	fl_alpha	fl_N	fl_out_m1	fl_out_m2	fl_out_m3	fl_out_m4	fl_out_m5
-2	0.1	0.0666667	-0.55	-0.55	-0.55	-0.55	-0.55
-1	0.3	0.1333333	-1.30	-1.30	-1.30	-1.30	-1.30
0	0.5	0.2000000	NaN	NaN	NaN	NaN	NaN
1	0.7	0.2666667	1.70	1.70	1.70	1.70	1.70
2	0.9	0.3333333	0.95	0.95	0.95	0.95	0.95

```
kable(tb_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

3.1.1.2 Mutate over Simple Function

For this example, use a very simple function with only one type of input, all inputs are scalars.

```
# Define Implicit Function
ffi_nonlinear <- function(fl_A, fl_alpha){

  fl_out <- (fl_A + fl_alpha*fl_A)/(fl_A)^2

  return(fl_out)
}
```

Apply the function over the dataframe, note five different ways below, the third way allows for parameters to be strings.

```
# variable names
svr_fl_A <- 'fl_A'
svr_fl_alpha <- 'fl_alpha'

# Evaluate
tb_nN_by_nQ_A_alpha_mutate_rows <- tb_nN_by_nQ_A_alpha %>%
  mutate(fl_out_m1 = ffi_nonlinear(fl_A=.$fl_A, fl_alpha=.$fl_alpha),
         fl_out_m2 = ffi_nonlinear(fl_A=`$`(., 'fl_A'), fl_alpha=`$`(., 'fl_alpha')),
         fl_out_m3 = ffi_nonlinear(fl_A=.[[svr_fl_A]], fl_alpha=.[[svr_fl_alpha]]),
         fl_out_m4 = ffi_nonlinear(fl_A=fl_A, fl_alpha=fl_alpha),
         fl_out_m5 = ffi_nonlinear(fl_A, fl_alpha))

# print
kable(tb_nN_by_nQ_A_alpha_mutate_rows) %>% kable_styling_fc()
```

3.1.1.3 Testing Function with Scalar and Arrays

Test non-linear Equation.

```
# Test Parameters
fl_N_agg = 100
fl_rho = -1
fl_N_q = ar_nN_N_choice[4]*fl_N_agg
ar_A_alpha = mt_nN_by_nQ_A_alpha[4,]
# Apply Function
ar_p1_s1 = exp((ar_A_alpha[1] - ar_nN_A)*fl_rho)
ar_p1_s2 = (ar_A_alpha[2]/ar_nN_alpha)
```

```
ar_p1_s3 = (1/(ar_nN_alpha*fl_rho - 1))
ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
ar_p2 = fl_N_q^((ar_A_alpha[2]*fl_rho-1)/(ar_nN_alpha*fl_rho-1))
ar_overall = ar_p1*ar_p2
fl_overall = fl_N_agg - sum(ar_overall)
print(fl_overall)
```

```
## [1] -598.2559
```

Implement the non-linear problem's evaluation using apply over all N individuals.

```
# Define Implicit Function
ffi_nonlin_dplyrdo <- function(fl_A, fl_alpha, fl_N, ar_A, ar_alpha, fl_N_agg, fl_rho){
  # ar_A_alpha[1] is A
  # ar_A_alpha[2] is alpha

  # # Test Parameters
  # fl_N = 100
  # fl_rho = -1
  # fl_N_q = 10

  # Apply Function
  ar_p1_s1 = exp((fl_A - ar_A)*fl_rho)
  ar_p1_s2 = (fl_alpha/ar_alpha)
  ar_p1_s3 = (1/(ar_alpha*fl_rho - 1))
  ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
  ar_p2 = fl_N^((fl_alpha*fl_rho-1)/(ar_alpha*fl_rho-1))
  ar_overall = ar_p1*ar_p2
  fl_overall = fl_N_agg - sum(ar_overall)

  return(fl_overall)
}

# Parameters
fl_rho = -1

# Evaluate Function
print(ffi_nonlin_dplyrdo(mt_nN_by_nQ_A_alpha[1,1],
                         mt_nN_by_nQ_A_alpha[1,2],
                         mt_nN_by_nQ_A_alpha[1,3]*fl_N_agg,
                         ar_nN_A, ar_nN_alpha, fl_N_agg, fl_rho))
```

```
## [1] 81.86645
```

```
for (i in seq(1,dim(mt_nN_by_nQ_A_alpha)[1])){
  fl_eval = ffi_nonlin_dplyrdo(mt_nN_by_nQ_A_alpha[i,1],
                               mt_nN_by_nQ_A_alpha[i,2],
                               mt_nN_by_nQ_A_alpha[i,3]*fl_N_agg,
                               ar_nN_A, ar_nN_alpha, fl_N_agg, fl_rho)
  print(fl_eval)
}
```

```
## [1] 81.86645
## [1] 54.48885
## [1] -65.5619
## [1] -598.2559
## [1] -3154.072
```

fl_A	fl_alpha	fl_N	dplyr_eval
-2	0.1	0.0666667	81.86645
-1	0.3	0.1333333	54.48885
0	0.5	0.2000000	-65.56190
1	0.7	0.2666667	-598.25595
2	0.9	0.3333333	-3154.07226

3.1.1.4 Evaluate Nonlinear Function using dplyr mutate

```
# Define Implicit Function
ffi_nonlin_dplyrdo <- function(fl_A, fl_alpha, fl_N, ar_A, ar_alpha, fl_N_agg, fl_rho){

  # Test Parameters
  # ar_A = ar_nN_A
  # ar_alpha = ar_nN_alpha
  # fl_N = 100
  # fl_rho = -1
  # fl_N_q = 10

  # Apply Function
  ar_p1_s1 = exp((fl_A - ar_A)*fl_rho)
  ar_p1_s2 = (fl_alpha/ar_alpha)
  ar_p1_s3 = (1/(ar_alpha*fl_rho - 1))
  ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
  ar_p2 = (fl_N*fl_N_agg)^((fl_alpha*fl_rho-1)/(ar_alpha*fl_rho-1))
  ar_overall = ar_p1*ar_p2
  fl_overall = fl_N_agg - sum(ar_overall)

  return(fl_overall)
}

# fl_A, fl_alpha are from columns of tb_nN_by_nQ_A_alpha
tb_nN_by_nQ_A_alpha = tb_nN_by_nQ_A_alpha %>% rowwise() %>%
                        mutate(dplyr_eval = ffi_nonlin_dplyrdo(fl_A, fl_alpha, fl_N,
                                                               ar_nN_A, ar_nN_alpha,
                                                               fl_N_agg, fl_rho))
# Show
kable(tb_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

3.1.2 Evaluate Choices Across States

> Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

See the ff_opti_bisect_pmap_multi function from Fan's *REconTools* Package, which provides a resuable function based on the algorithm worked out here.

We want evaluate linear function $0 = f(z_{ij}, x_i, y_i, \mathbf{X}, \mathbf{Y}, c, d)$. There are i functions that have i specific x and y. For each i function, we evaluate along a grid of feasible values for z, over $j \in J$ grid points, potentially looking for the j that is closest to the root. $\mathbf{X}$ and $\mathbf{Y}$ are arrays common across the i equations, and c and d are constants.

The evaluation strategy is the following, given min and max for z that are specific for each j, and given common number of grid points, generate a matrix of z_{ij}. Suppose there the number of i is I, and the number of grid points for j is J.

1. Generate a $J \cdot I$ by 3 matrix where the columns are z, x, y as tibble
2. Follow this Mutate to evaluate the $f(\cdot)$ function.

3. Add two categorical columns for grid levels and wich i, i and j index. Plot Mutate output evaluated column categorized by i as color and j as x-axis.

3.1.2.1 Set up Input Arrays

There is a function that takes $M = Q + P$ inputs, we want to evaluate this function N times. Each time, there are M inputs, where all but Q of the M inputs, meaning P of the M inputs, are the same. In particular, $P = Q * N$.

$$M = Q + P = Q + Q * N$$

Now we need to expand this by the number of choice grid. Each row, representing one equation, is expanded by the number of choice grids. We are graphically searching, or rather brute force searching, which means if we have 100 individuals, we want to plot out the nonlinear equation for each of these lines, and show graphically where each line crosses zero. We achieve this, by evaluating the equation for each of the 100 individuals along a grid of feasible choices.

In this problem here, the feasible choices are shared across individuals.

```
# Parameters
fl_rho = 0.20
svr_id_var = 'INDI_ID'

# it_child_count = N, the number of children
it_N_child_cnt = 4
# it_heter_param = Q, number of parameters that are heterogeneous across children
it_Q_hetpa_cnt = 2

# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
ar_nP_A_alpha = c(ar_nN_A, ar_nN_alpha)

# N by Q varying parameters
mt_nN_by_nQ_A_alpha = cbind(ar_nN_A, ar_nN_alpha)

# Choice Grid for nutritional feasible choices for each
fl_N_agg = 100
fl_N_min = 0
it_N_choice_cnt_ttest = 3
it_N_choice_cnt_dense = 100
ar_N_choices_ttest = seq(fl_N_min, fl_N_agg, length.out = it_N_choice_cnt_ttest)
ar_N_choices_dense = seq(fl_N_min, fl_N_agg, length.out = it_N_choice_cnt_dense)

# Mesh Expand
tb_states_choices <- as_tibble(mt_nN_by_nQ_A_alpha) %>% rowid_to_column(var=svr_id_var)
tb_states_choices_ttest <- tb_states_choices %>% expand_grid(choices = ar_N_choices_ttest)
tb_states_choices_dense <- tb_states_choices %>% expand_grid(choices = ar_N_choices_dense)

# display
summary(tb_states_choices_dense)
```

```
##     INDI_ID        ar_nN_A     ar_nN_alpha     choices
##  Min.   :1.00   Min.   :-2   Min.   :0.1   Min.   :  0
##  1st Qu.:1.75   1st Qu.:-1   1st Qu.:0.3   1st Qu.: 25
##  Median :2.50   Median : 0   Median :0.5   Median : 50
##  Mean   :2.50   Mean   : 0   Mean   :0.5   Mean   : 50
##  3rd Qu.:3.25   3rd Qu.: 1   3rd Qu.:0.7   3rd Qu.: 75
##  Max.   :4.00   Max.   : 2   Max.   :0.9   Max.   :100
```

INDI_ID	ar_nN_A	ar_nN_alpha	choices
1	-2.0000000	0.1000000	0
1	-2.0000000	0.1000000	50
1	-2.0000000	0.1000000	100
2	-0.6666667	0.3666667	0
2	-0.6666667	0.3666667	50
2	-0.6666667	0.3666667	100
3	0.6666667	0.6333333	0
3	0.6666667	0.6333333	50
3	0.6666667	0.6333333	100
4	2.0000000	0.9000000	0
4	2.0000000	0.9000000	50
4	2.0000000	0.9000000	100

```
kable(tb_states_choices_ttest) %>%
  kable_styling_fc()
```

3.1.2.2 Apply Same Function all Rows, Some Inputs Row-specific, other Shared

There are two types of inputs, row-specific inputs, and inputs that should be applied for each row. The Function just requires all of these inputs, it does not know what is row-specific and what is common for all row. Dplyr recognizes which parameter inputs already existing in the piped dataframe/tibble, given rowwise, those will be row-specific inputs. Additional function parameters that do not exist in dataframe as variable names, but that are pre-defined scalars or arrays will be applied to all rows.

- **?** string variable name of input where functions are evaluated, these are already contained in the dataframe, existing variable names, row specific, rowwise computation over these, each rowwise calculation using different rows: *fl_A*, *fl_alpha*, *fl_N*
- **?** scalar and array values that are applied to every rowwise calculation, all rowwise calculations using the same scalars and arrays:*ar_A*, *ar_alpha*, *fl_N_agg*, *fl_rho*
- **?** string output variable name

The function looks within group, finds min/max etc that are relevant.

```
# Convert Matrix to Tibble
ar_st_col_names = c(svr_id_var,'fl_A', 'fl_alpha')
tb_states_choices <- tb_states_choices %>% rename_all(~c(ar_st_col_names))
ar_st_col_names = c(svr_id_var,'fl_A', 'fl_alpha', 'fl_N')
tb_states_choices_ttest <- tb_states_choices_ttest %>% rename_all(~c(ar_st_col_names))
tb_states_choices_dense <- tb_states_choices_dense %>% rename_all(~c(ar_st_col_names))

# Define Implicit Function
ffi_nonlin_dplyrdo <- function(fl_A, fl_alpha, fl_N, ar_A, ar_alpha, fl_N_agg, fl_rho){
  # scalar value that are row-specific, in dataframe already: *fl_A*, *fl_alpha*, *fl_N*
  # array and scalars not in dataframe, common all rows: *ar_A*, *ar_alpha*, *fl_N_agg*, *fl_rho*

  # Test Parameters
  # ar_A = ar_nN_A
  # ar_alpha = ar_nN_alpha
  # fl_N = 100
  # fl_rho = -1
  # fl_N_q = 10

  # Apply Function
  ar_p1_s1 = exp((fl_A - ar_A)*fl_rho)
  ar_p1_s2 = (fl_alpha/ar_alpha)
```

INDI_ID	fl_A	fl_alpha	fl_N	dplyr_eval
1	-2.0000000	0.1000000	0	100.00000
1	-2.0000000	0.1000000	50	-5666.95576
1	-2.0000000	0.1000000	100	-12880.28392
2	-0.6666667	0.3666667	0	100.00000
2	-0.6666667	0.3666667	50	-595.73454
2	-0.6666667	0.3666667	100	-1394.70698
3	0.6666667	0.6333333	0	100.00000
3	0.6666667	0.6333333	50	-106.51058
3	0.6666667	0.6333333	100	-323.94216
4	2.0000000	0.9000000	0	100.00000
4	2.0000000	0.9000000	50	22.55577
4	2.0000000	0.9000000	100	-51.97161

```
  ar_p1_s3 = (1/(ar_alpha*fl_rho - 1))
  ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
  ar_p2 = fl_N^((fl_alpha*fl_rho-1)/(ar_alpha*fl_rho-1))
  ar_overall = ar_p1*ar_p2
  fl_overall = fl_N_agg - sum(ar_overall)

  return(fl_overall)
}
```

3.1.2.2.1 3 Points and Denser Dataframs and Define Function

3.1.2.2.2 Evaluate at Three Choice Points and Show Table

In the example below, just show results evaluating over three choice points and show table.

```
# fl_A, fl_alpha are from columns of tb_nN_by_nQ_A_alpha
tb_states_choices_ttest_eval = tb_states_choices_ttest %>% rowwise() %>%
                        mutate(dplyr_eval = ffi_nonlin_dplyrdo(fl_A, fl_alpha, fl_N,
                                                               ar_nN_A, ar_nN_alpha,
                                                               fl_N_agg, fl_rho))
# Show
kable(tb_states_choices_ttest_eval) %>%
  kable_styling_fc()
```

3.1.2.2.3 Evaluate at Many Choice Points and Show Graphically

Same as above, but now we evaluate the function over the individuals at many choice points so that we can graph things out.

```
# fl_A, fl_alpha are from columns of tb_nN_by_nQ_A_alpha
tb_states_choices_dense_eval = tb_states_choices_dense %>% rowwise() %>%
                        mutate(dplyr_eval = ffi_nonlin_dplyrdo(fl_A, fl_alpha, fl_N,
                                                               ar_nN_A, ar_nN_alpha,
                                                               fl_N_agg, fl_rho))
```

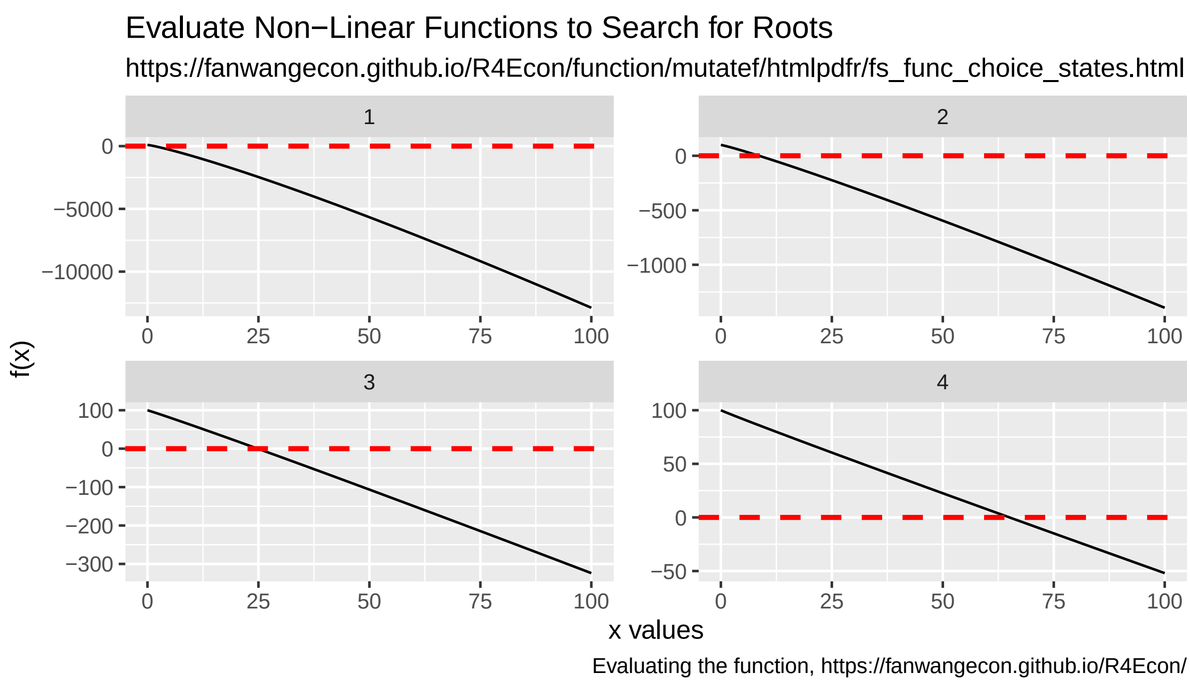
```
# Labeling
st_title <- paste0('Evaluate Non-Linear Functions to Search for Roots')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/function/mutatef/htmlpdfr/fs_func_choice_states.html')
st_caption <- paste0('Evaluating the function, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'x values'
st_y_label <- 'f(x)'

# Show
dim(tb_states_choices_dense_eval)
```

```
## [1] 400   5
```

```r
summary(tb_states_choices_dense_eval)
```

```
##     INDI_ID          fl_A        fl_alpha         fl_N         dplyr_eval
##  Min.   :1.00   Min.   :-2   Min.   :0.1   Min.   :  0   Min.   :-12880.28
##  1st Qu.:1.75   1st Qu.:-1   1st Qu.:0.3   1st Qu.: 25   1st Qu.: -1167.29
##  Median :2.50   Median : 0   Median :0.5   Median : 50   Median :  -202.42
##  Mean   :2.50   Mean   : 0   Mean   :0.5   Mean   : 50   Mean   : -1645.65
##  3rd Qu.:3.25   3rd Qu.: 1   3rd Qu.:0.7   3rd Qu.: 75   3rd Qu.:     0.96
##  Max.   :4.00   Max.   : 2   Max.   :0.9   Max.   :100   Max.   :   100.00
```

```r
lineplot <- tb_states_choices_dense_eval %>%
    ggplot(aes(x=fl_N, y=dplyr_eval)) +
        geom_line() +
        facet_wrap( . ~ INDI_ID, scales = "free") +
        geom_hline(yintercept=0, linetype="dashed",
               color = "red", size=1) +
        labs(title = st_title,
             subtitle = st_subtitle,
             x = st_x_label,
             y = st_y_label,
             caption = st_caption)
print(lineplot)
```



3.2 Dataframe Do Anything

3.2.1 (Mx1 by N) to (MxQ by N+1)

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Case One: There is a dataframe with M rows, based on these m specific information, generate dataframes for each m. Stack these indivdiual dataframes together and merge original m specific information in as well. The number of rows for each m is Q_m, each m could have different number of expansion rows.

ID	Q	sd	mean
1	1	0.010	1000
2	3	100.005	1000
3	4	200.000	1000

Generate a panel with M individuals, each individual is observed for different spans of times (*uncount*). Before expanding, generate individual specific normal distribution standard deviation. All individuals share the same mean, but have increasing standard deviations.

3.2.1.1 Generate Dataframe with M Rows.

This is the first step, generate M rows of data, to be expanded. Each row contains the number of normal draws to make and the mean and the standard deviation for normal daraws that are m specific.

```
# Parameter Setups
it_M <- 3
it_Q_max <- 5
fl_rnorm_mu <- 1000
ar_rnorm_sd <- seq(0.01, 200, length.out=it_M)
ar_it_q <- sample.int(it_Q_max, it_M, replace=TRUE)

# N by Q varying parameters
mt_data = cbind(ar_it_q, ar_rnorm_sd)
tb_M <- as_tibble(mt_data) %>% rowid_to_column(var = "ID") %>%
                rename(sd = ar_rnorm_sd, Q = ar_it_q) %>%
                mutate(mean = fl_rnorm_mu)

# display
kable(tb_M) %>%
  kable_styling_fc()
```

3.2.1.2 Random Normal Draw Expansion

The steps are:

1. do anything
2. use ".$" sign to refer to variable names, or [['name']]
3. unnest
4. left_join expanded and original

Note these all give the same results

Use dot dollar to get variables

```
# Generate $Q_m$ individual specific incomes, expanded different number of times for each m
tb_income <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(.$Q, mean=.$mean, sd=.$sd)) %>%
  unnest(c(income))

# Merge back with tb_M
tb_income_full_dd <- tb_income %>%
  left_join(tb_M)

# display
kable(tb_income) %>%
  kable_styling_fc()
```

```
kable(tb_income_full_dd) %>%
  kable_styling_fc()
```

ID	income
1	999.9803
2	1070.1391
2	952.7185
2	893.2123
3	956.4050
3	794.7991
3	854.2218
3	874.9921

ID	income	Q	sd	mean
1	999.9803	1	0.010	1000
2	1070.1391	3	100.005	1000
2	952.7185	3	100.005	1000
2	893.2123	3	100.005	1000
3	956.4050	4	200.000	1000
3	794.7991	4	200.000	1000
3	854.2218	4	200.000	1000
3	874.9921	4	200.000	1000

3.2.2 (MxP by N) to (Mx1 by 1)

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

3.2.2.1 Wages from Many Countries and Country-specific GINI

There is a Panel with M individuals and each individual has Q records/rows. A function generate an individual specific outcome given the Q individual specific inputs, along with shared parameters/values stored as variables that contain common values for each of the M individuals.

For example, suppose we have a dataframe of individual wage information from different countries (the number of countries is M). Each row is an individual from one country, giving us $Q \cdot M$ observations of wages.

We want to generate country specific gini based on the individual wage data for each country in the dataframe. Additionally, perhaps the gini formula requires not just individual wages but some additional parameters or shared dataframes as inputs. We will use the ff_dist_gini_vector_pos.html function from REconTools.

First, we simulate a dataframe with M countries, and up to Q people in each country. The countries share the same mean income, but have different standard deviations.

```
# Parameter Setups
it_M <- 10
it_Q_max <- 100
fl_rnorm_mu <- 1
ar_rnorm_sd <- seq(0.01, 0.2, length.out=it_M)
set.seed('789')
ar_it_q <- sample.int(it_Q_max, it_M, replace=TRUE)

# N by Q varying parameters
mt_data <- cbind(ar_it_q, ar_rnorm_sd)
tb_M <- as_tibble(mt_data) %>% rowid_to_column(var = "ID") %>%
                rename(sd = ar_rnorm_sd,
                       Q = ar_it_q) %>%
                mutate(mean = fl_rnorm_mu) %>%
                select(ID, Q,
                       mean, sd)
```

M=10 countries (ID is country ID), observation per country (Q), mean and s.d. of wages each country

ID	Q	mean	sd
1	45	1	0.0100000
2	12	1	0.0311111
3	42	1	0.0522222
4	26	1	0.0733333
5	99	1	0.0944444
6	37	1	0.1155556
7	100	1	0.1366667
8	43	1	0.1577778
9	67	1	0.1788889
10	70	1	0.2000000

```
# Show table
kable(tb_M, caption = paste0("M=", it_M,
  " countries (ID is country ID), observation per country (Q)",
  ", mean and s.d. of wages each country")) %>%
  kable_styling_fc()
```

Second, we now expand the dataframe so that each country has not just one row, but Q_i of observations (i is country), or randomly drawn income based on the country-specific income distribution. Note that there are three ways of referring to variable names with dot, which are all shown below:

1. We can explicitly refer to names
2. We can use the dollar dot structure to use string variable names in do anything.
3. We can use dot bracket, this is the only option that works with string variable names

```
# A. Normal Draw Expansion, Explicitly Name
set.seed('123')
tb_income_norm_dot_dollar <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(.$Q, mean=.$mean, sd=.$sd)) %>%
  unnest(c(income)) %>%
  left_join(tb_M, by="ID")

# Normal Draw Expansion again, dot dollar differently with string variable name
set.seed('123')
tb_income_norm_dollar_dot <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(`$`(., 'Q'), mean = `$`(., 'mean'), sd = `$`(., 'sd'))) %>%
  unnest(c(income)) %>%
  left_join(tb_M, by="ID")

# Normal Draw Expansion again, dot double bracket
set.seed('123')
svr_mean <- 'mean'
svr_sd <- 'sd'
svr_Q <- 'Q'
tb_income_norm_dot_bracket_db <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(.[[svr_Q]], mean = .[[svr_mean]], sd = .[[svr_sd]])) %>%
  unnest(c(income)) %>%
  left_join(tb_M, by="ID")
```

Third, we print the first set of rows of the dataframe, and also summarize income by country groups.

```
# Show dataframe dimension
print(dim(tb_income_norm_dot_bracket_db))
```

```
## [1] 541   5
```

ID = country ID, wage draws

ID	income	Q	mean	sd
1	0.9943952	45	1	0.01
1	0.9976982	45	1	0.01
1	1.0155871	45	1	0.01
1	1.0007051	45	1	0.01
1	1.0012929	45	1	0.01
1	1.0171506	45	1	0.01
1	1.0046092	45	1	0.01
1	0.9873494	45	1	0.01
1	0.9931315	45	1	0.01
1	0.9955434	45	1	0.01
1	1.0122408	45	1	0.01
1	1.0035981	45	1	0.01
1	1.0040077	45	1	0.01
1	1.0011068	45	1	0.01
1	0.9944416	45	1	0.01
1	1.0178691	45	1	0.01
1	1.0049785	45	1	0.01
1	0.9803338	45	1	0.01
1	1.0070136	45	1	0.01
1	0.9952721	45	1	0.01

```r
# Show first 20 rows
kable(head(tb_income_norm_dot_bracket_db, 20),
  caption = "ID = country ID, wage draws"
  ) %>% kable_styling_fc()
```

```r
# Display country-specific summaries
REconTools::ff_summ_bygroup(tb_income_norm_dot_bracket_db, c("ID"), "income")$df_table_grp_stats
```

Fourth, there is only one input for the gini function *ar_pos*. Note that the gini are not very large even with large SD, because these are normal distributions. By Construction, most peple are in the middle. So with almost zero standard deviation, we have perfect equality, as standard deviation increases, inequality increases, but still pretty equal overall, there is no fat upper tail.

```r
# Gini by Group
tb_gini_norm <- tb_income_norm_dot_bracket_db %>% group_by(ID) %>%
  do(inc_gini_norm = REconTools::ff_dist_gini_vector_pos(.$income)) %>%
  unnest(c(inc_gini_norm)) %>%
  left_join(tb_M, by="ID")

# display
kable(tb_gini_norm,
  caption = paste0(
    "Country-specific wage GINI based on income draws",
    ", ID=country-ID, Q=sample-size-per-country",
    ", mean=true-income-mean, sd=true-income-sd"
  )) %>%
  kable_styling_fc()
```

3.2.3 (MxP by N) to (MxQ by N+Z)

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

There is a dataframe composed of M mini-dataframes. Group by a variable that identifies each unique

Country-specific wage GINI based on income draws, ID=country-ID, Q=sample-size-per-country, mean=true-income-mean, sd=true-income-sd

ID	inc_gini_norm	Q	mean	sd
1	0.0052111	45	1	0.0100000
2	0.0137174	12	1	0.0311111
3	0.0245939	42	1	0.0522222
4	0.0303468	26	1	0.0733333
5	0.0527628	99	1	0.0944444
6	0.0544053	37	1	0.1155556
7	0.0786986	100	1	0.1366667
8	0.0818873	43	1	0.1577778
9	0.1014639	67	1	0.1788889
10	0.0903825	70	1	0.2000000

sub-dataframe, and use the sub-dataframes with P rows as inputs to a function.

The function outputs Q *by* Z rows and columns of results, stack the results. The output file has MxQ rows and the Z columns of additional results should be appended.

3.2.3.1 Generate the MxP by N Dataframe

M Grouping characteristics, P rows for each group, and N Variables.

1. M are individuals
2. P are dates
3. A wage variable for individual wage at each date. And a savings varaible as well.

```
# Define
it_M <- 3
it_P <- 5
svr_m <- 'group_m'
svr_mp <- 'info_mp'

# dataframe
set.seed(123)
df_panel_skeleton <- as_tibble(matrix(it_P, nrow=it_M, ncol=1)) %>%
  rowid_to_column(var = svr_m) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_m)) %>% mutate(!!sym(svr_mp) := row_number()) %>%
  ungroup() %>%
  rowwise() %>% mutate(wage = rnorm(1, 100, 10),
                       savings = rnorm(1, 200, 30)) %>%
  ungroup() %>%
  rowid_to_column(var = "id_ji")

# Print
kable(df_panel_skeleton) %>% kable_styling_fc()
```

3.2.3.2 Subgroup Compute and Expand

Use the M sub-dataframes, generate Q by Z result for each of the M groups. Stack all results together.

Base on all the wages for each individual, generate individual specific mean and standard deviations. Do this for three things, the wage variable, the savings variable, and the sum of wage and savings:

1. $Z=2$: 2 columns, mean and standard deviation
2. $Q=3$: 3 rows, statistics based on wage, savings, and the sum of both

First, here is the processing function that takes the dataframe as input, with a parameter for rounding:

id_ji	group_m	info_mp	wage	savings
1	1	1	94.39524	253.6074
2	1	2	97.69823	214.9355
3	1	3	115.58708	141.0015
4	1	4	100.70508	221.0407
5	1	5	101.29288	185.8163
6	2	1	117.15065	167.9653
7	2	2	104.60916	193.4608
8	2	3	87.34939	169.2199
9	2	4	93.13147	178.1333
10	2	5	95.54338	181.2488
11	3	1	112.24082	149.3992
12	3	2	103.59814	225.1336
13	3	3	104.00771	204.6012
14	3	4	101.10683	165.8559
15	3	5	94.44159	237.6144

```r
# define function
ffi_subset_mean_sd <- function(df_sub, it_round=1) {
  #' A function that generates mean and sd for several variables
  #'
  #' @description
  #' Assume there are two variables in df_sub wage and savings
  #'
  #' @param df_sub dataframe where each individual row is a different
  #' data point, over which we compute mean and sd, Assum there are two
  #' variables, savings and wage
  #' @param it_round integer rounding for resulting dataframe
  #' @return a dataframe where each row is aggregate for a different type
  #' of variablea and each column is a different statistics

  fl_wage_mn = mean(df_sub$wage)
  fl_wage_sd = sd(df_sub$wage)

  fl_save_mn = mean(df_sub$savings)
  fl_save_sd = sd(df_sub$savings)

  fl_wgsv_mn = mean(df_sub$wage + df_sub$savings)
  fl_wgsv_sd = sd(df_sub$wage + df_sub$savings)

  ar_mn <- c(fl_wage_mn, fl_save_mn, fl_wgsv_mn)
  ar_sd <- c(fl_wage_sd, fl_save_sd, fl_wgsv_sd)
  ar_st_row_lab <- c('wage', 'savings', 'wage_and_savings')

  mt_stats <- cbind(ar_mn, ar_sd)
  mt_stats <- round(mt_stats, it_round)

  ar_st_varnames <- c('mean', 'sd', 'variables')
  df_combine <- as_tibble(mt_stats) %>%
    add_column(ar_st_row_lab) %>%
    rename_all(~c(ar_st_varnames)) %>%
    select(variables, 'mean', 'sd') %>%
    rowid_to_column(var = "id_q")

  return(df_combine)
}
```

id_mq	group_m	id_q	variables	mean	sd
1	1	1	wage	101.94	8.11
2	1	2	savings	203.28	42.33
3	1	3	wage_and_savings	305.22	34.83
4	2	1	wage	99.56	11.63
5	2	2	savings	178.01	10.34
6	2	3	wage_and_savings	277.56	15.48
7	3	1	wage	103.08	6.39
8	3	2	savings	196.52	37.86
9	3	3	wage_and_savings	299.60	33.50

```
# testing function
ffi_subset_mean_sd(df_panel_skeleton %>% filter(!!sym(svr_m)==1))
```

Second, call *ffi_subset_mean_sd* function for each of the groups indexed by j and stack results together with j index:

1. group by
2. call function
3. unnest

```
# run group stats and stack dataframes
df_outputs <- df_panel_skeleton %>% group_by(!!sym(svr_m)) %>%
  do(df_stats = ffi_subset_mean_sd(., it_round=2)) %>%
  unnest() %>%
  rowid_to_column(var = "id_mq")
# print
kable(df_outputs) %>% kable_styling_fc()
```

In the resulting file, we went from a matrix with MxP rows to a matrix with MxQ Rows.

3.3 Apply and pmap

3.3.1 Apply and Sapply

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

- r apply matrix to function row by row
- r evaluate function on grid
- Apply a function to every row of a matrix or a data frame
- r apply
- r sapply
- sapply over matrix row by row
- function as parameters using formulas
- do

We want evaluate linear function f(x_i, y_i, ar_x, ar_y, c, d), where c and d are constants, and ar_x and ar_y are arrays, both fixed. x_i and y_i vary over each row of matrix. More specifically, we have a functions, this function takes inputs that are individual specific. We would like to evaluate this function concurrently across N individuals.

The function is such that across the N individuals, some of the function parameter inputs are the same, but others are different. If we are looking at demand for a particular product, the prices of all products enter the demand equation for each product, but the product's own price enters also in a different way.

The objective is either to just evaluate this function across N individuals, or this is a part of a nonlinear solution system.

ar_nN_A	ar_nN_alpha
-2	0.1
-1	0.3
0	0.5
1	0.7
2	0.9

What is the relationship between apply, lapply and vectorization? see Is the "*apply" family really not vectorized?.

3.3.1.1 Set up Input Arrays

There is a function that takes $M = Q + P$ inputs, we want to evaluate this function N times. Each time, there are M inputs, where all but Q of the M inputs, meaning P of the M inputs, are the same. In particular, $P = Q * N$.

$$M = Q + P = Q + Q * N$$

```
# it_child_count = N, the number of children
it_N_child_cnt = 5
# it_heter_param = Q, number of parameters that are
# heterogeneous across children
it_Q_hetpa_cnt = 2

# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
ar_nP_A_alpha = c(ar_nN_A, ar_nN_alpha)

# N by Q varying parameters
mt_nN_by_nQ_A_alpha = cbind(ar_nN_A, ar_nN_alpha)

# display
kable(mt_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

3.3.1.2 Using apply

3.3.1.2.1 Named Function First we use the apply function, we have to hard-code the arrays that are fixed for each of the N individuals. Then apply allows us to loop over the matrix that is N by Q, each row one at a time, from 1 to N.

```
# Define Implicit Function
ffi_linear_hardcode <- function(ar_A_alpha){
  # ar_A_alpha[1] is A
  # ar_A_alpha[2] is alpha

  fl_out = sum(ar_A_alpha[1]*ar_nN_A +
               1/(ar_A_alpha[2] + 1/ar_nN_alpha))

  return(fl_out)
}

# Evaluate function row by row
ar_func_apply = apply(mt_nN_by_nQ_A_alpha, 1, ffi_linear_hardcode)
```

3.3.1.2.2 Anonymous Function

- apply over matrix

Apply with anonymous function generating a list of arrays of different lengths. In the example below, we want to drawn N sets of random uniform numbers, but for each set the number of draws we want to have is Q_i. Furthermore, we want to rescale the random uniform draws so that they all become proportions that sum u pto one for each i, but then we multply each row's values by the row specific aggregates.

The anonymous function has hard coded parameters. Using an anonymous function here allows for parameters to be provided inside the function that are shared across each looped evaluation. This is perhaps more convenient than sapply with additional parameters.

```
set.seed(1039)

# Define the number of draws each row and total amount
it_N <- 4
fl_unif_min <- 1
fl_unif_max <- 2
mt_draw_define <- cbind(sample(it_N, it_N, replace=TRUE),
                        runif(it_N, min=1, max=10))
tb_draw_define <- as_tibble(mt_draw_define) %>%
  rowid_to_column(var = "draw_group")
print(tb_draw_define)

# apply row by row, anonymous function has hard
# coded min and max
ls_ar_draws_shares_lvls =
  apply(tb_draw_define,
        1,
        function(row) {
          it_draw <- row[2]
          fl_sum <- row[3]
          ar_unif <- runif(it_draw,
                           min=fl_unif_min,
                           max=fl_unif_max)
          ar_share <- ar_unif/sum(ar_unif)
          ar_levels <- ar_share*fl_sum
          return(list(ar_share=ar_share,
                      ar_levels=ar_levels))
        })

# Show Results as list
print(ls_ar_draws_shares_lvls)
```

```
## [[1]]
## [[1]]$ar_share
## [1] 0.2783638 0.2224140 0.2797840 0.2194381
##
## [[1]]$ar_levels
## [1] 1.492414 1.192446 1.500028 1.176491
##
##
## [[2]]
## [[2]]$ar_share
## [1] 0.5052919 0.4947081
##
## [[2]]$ar_levels
## [1] 3.866528 3.785541
##
##
## [[3]]
```

ar_share	ar_levels
0.2783638, 0.2224140, 0.2797840, 0.2194381	1.492414, 1.192446, 1.500028, 1.176491
0.5052919, 0.4947081	3.866528, 3.785541
1	9.572211
0.4211426, 0.2909812, 0.2878762	4.051971, 2.799640, 2.769765

```
## [[3]]$ar_share
## [1] 1
##
## [[3]]$ar_levels
##       V2
## 9.572211
##
##
## [[4]]
## [[4]]$ar_share
## [1] 0.4211426 0.2909812 0.2878762
##
## [[4]]$ar_levels
## [1] 4.051971 2.799640 2.769765
```

Above, our results is a list of lists. We can convert this to a table. If all results are scalar, would be regular table where each cell has a single scalar value.

```
# Show results as table
kable(as_tibble(do.call(rbind, ls_ar_draws_shares_lvls))) %>%
  kable_styling_fc()
```

We will try to do the same thing as above, but now the output will be a stacked dataframe. Note that within each element of the apply row by row loop, we are generating two variables *ar_share* and *ar_levels*. We will not generate a dataframe with multiple columns, storing *ar_share*, *ar_levels* as well as information on *min*, *max*, number of draws and rescale total sum.

```
set.seed(1039)
# apply row by row, anonymous function has hard coded min and max
ls_mt_draws_shares_lvls =
  apply(tb_draw_define, 1, function(row) {

    it_draw_group <- row[1]
    it_draw <- row[2]
    fl_sum <- row[3]

    ar_unif <- runif(it_draw,
                     min=fl_unif_min,
                     max=fl_unif_max)
    ar_share <- ar_unif/sum(ar_unif)
    ar_levels <- ar_share*fl_sum

    mt_all_res <- cbind(it_draw_group, it_draw, fl_sum,
                        ar_unif, ar_share, ar_levels)
    colnames(mt_all_res) <-
      c('draw_group', 'draw_count', 'sum',
        'unif_draw', 'share', 'rescale')
    rownames(mt_all_res) <- NULL

    return(mt_all_res)
  })
mt_draws_shares_lvls_all <- do.call(rbind, ls_mt_draws_shares_lvls)
# Show Results
```

draw_group	draw_count	sum	unif_draw	share	rescale
1	4	5.361378	1.125668	0.1988606	1.066167
1	4	5.361378	1.668536	0.2947638	1.580340
1	4	5.361378	1.419382	0.2507483	1.344356
1	4	5.361378	1.447001	0.2556274	1.370515
2	2	7.652069	1.484598	0.4605236	3.523959
2	2	7.652069	1.739119	0.5394764	4.128110
3	1	9.572211	1.952468	1.0000000	9.572211
4	3	9.621375	1.957931	0.3609352	3.472693
4	3	9.621375	1.926995	0.3552324	3.417824
4	3	9.621375	1.539678	0.2838324	2.730858

```r
kable(mt_draws_shares_lvls_all) %>% kable_styling_fc()
```

3.3.1.3 Using sapply

3.3.1.3.1 Named Function

- r convert matrix to list
- Convert a matrix to a list of vectors in R

Sapply allows us to not have tohard code in the A and alpha arrays. But Sapply works over List or Vector, not Matrix. So we have to convert the N by Q matrix to a N element list Now update the function with sapply.

```r
ls_ar_nN_by_nQ_A_alpha = as.list(data.frame(t(mt_nN_by_nQ_A_alpha)))

# Define Implicit Function
ffi_linear_sapply <- function(ar_A_alpha, ar_A, ar_alpha){
  # ar_A_alpha[1] is A
  # ar_A_alpha[2] is alpha

  fl_out = sum(ar_A_alpha[1]*ar_nN_A +
                 1/(ar_A_alpha[2] + 1/ar_nN_alpha))

  return(fl_out)
}

# Evaluate function row by row
ar_func_sapply = sapply(ls_ar_nN_by_nQ_A_alpha, ffi_linear_sapply,
                        ar_A=ar_nN_A, ar_alpha=ar_nN_alpha)
```

3.3.1.3.2 Anonymous Function, list of arrays as output

- sapply anonymous function
- r anoymous function multiple lines

Sapply with anonymous function generating a list of arrays of different lengths. In the example below, we want to drawn N sets of random uniform numbers, but for each set the number of draws we want to have is Q_i. Furthermore, we want to rescale the random uniform draws so that they all become proportions that sum u pto one for each i.

```r
it_N <- 4
fl_unif_min <- 1
fl_unif_max <- 2

# Generate using runif without anonymous function
set.seed(1039)
ls_ar_draws = sapply(seq(it_N),
```

```
                      runif,
                      min=fl_unif_min, max=fl_unif_max)
print(ls_ar_draws)
```

```
## [[1]]
## [1] 1.125668
##
## [[2]]
## [1] 1.668536 1.419382
##
## [[3]]
## [1] 1.447001 1.484598 1.739119
##
## [[4]]
## [1] 1.952468 1.957931 1.926995 1.539678
```

```
# Generate Using Anonymous Function
set.seed(1039)
ls_ar_draws_shares = sapply(seq(it_N),
                             function(n, min, max) {
                               ar_unif <- runif(n,min,max)
                               ar_share <- ar_unif/sum(ar_unif)
                               return(ar_share)
                             },
                             min=fl_unif_min, max=fl_unif_max)
# Print Share
print(ls_ar_draws_shares)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 0.5403432 0.4596568
##
## [[3]]
## [1] 0.3098027 0.3178522 0.3723451
##
## [[4]]
## [1] 0.2646671 0.2654076 0.2612141 0.2087113
```

```
# Sapply with anonymous function to check sums
sapply(seq(it_N), function(x) {sum(ls_ar_draws[[x]])})
```

```
## [1] 1.125668 3.087918 4.670717 7.377071
```

```
sapply(seq(it_N), function(x) {sum(ls_ar_draws_shares[[x]])})
```

```
## [1] 1 1 1 1
```

3.3.1.3.3 Anonymous Function, matrix as output Below, we provide another example with sapply, we generate probabilities for discrete random variables that follow the binomial distribution. We do this for twice, with "chance of success" set at different values.

The output in this case is a matrix, where each column stores the output from a different dbinom call.

```
# First, generate the results without sapply
ar_binomprob <- matrix(c(0.1, 0.9), nrow=2, ncol=1)
# Second, generate the results with sapply
# dbinom call: dbinom(x, size, prob, log = FALSE)
# The function requires x, size, and prob.
# we provide x and size, and each element of ar_binomprob
```

	eval_lin_apply	eval_lin_sapply
X1	2.346356	2.346356
X2	2.094273	2.094273
X3	1.895316	1.895316
X4	1.733708	1.733708
X5	1.599477	1.599477

```
# will be a different prob.
mt_dbinom <- sapply(ar_binomprob, dbinom, x=seq(0,4), size=4)
# Third compare results
print(paste0('binomial p=', ar_binomprob[1]))
```

```
## [1] "binomial p=0.1"
```

```
print(dbinom(seq(0,4), 4, ar_binomprob[1]))
```

```
## [1] 0.6561 0.2916 0.0486 0.0036 0.0001
```

```
print(mt_dbinom[,1])
```

```
## [1] 0.6561 0.2916 0.0486 0.0036 0.0001
```

```
print(paste0('binomial p=', ar_binomprob[2]))
```

```
## [1] "binomial p=0.9"
```

```
print(dbinom(seq(0,4), 4, ar_binomprob[2]))
```

```
## [1] 0.0001 0.0036 0.0486 0.2916 0.6561
```

```
print(mt_dbinom[,2])
```

```
## [1] 0.0001 0.0036 0.0486 0.2916 0.6561
```

3.3.1.4 Compare Results

```
# Show overall Results
mt_results <- cbind(ar_func_apply, ar_func_sapply)
colnames(mt_results) <- c('eval_lin_apply', 'eval_lin_sapply')
kable(mt_results) %>% kable_styling_fc()
```

3.3.2 Mutate Evaluate Functions

> Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Apply a function over rows of a matrix using mutate, rowwise, etc.

3.3.2.1 Set up Input Arrays

There is a function that takes $M = Q + P$ inputs, we want to evaluate this function N times. Each time, there are M inputs, where all but Q of the M inputs, meaning P of the M inputs, are the same. In particular, $P = Q * N$.

$$M = Q + P = Q + Q * N$$

```
# it_child_count = N, the number of children
it_N_child_cnt = 5
# it_heter_param = Q, number of parameters that are
# heterogeneous across children
it_Q_hetpa_cnt = 2
```

ar_nN_A	ar_nN_alpha
-2	0.1
-1	0.3
0	0.5
1	0.7
2	0.9

fl_A	fl_alpha
-2	0.1
-1	0.3
0	0.5
1	0.7
2	0.9

```
# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
ar_nP_A_alpha = c(ar_nN_A, ar_nN_alpha)

# N by Q varying parameters
mt_nN_by_nQ_A_alpha = cbind(ar_nN_A, ar_nN_alpha)

# display
kable(mt_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

```
# Convert Matrix to Tibble
ar_st_col_names = c('fl_A', 'fl_alpha')
tb_nN_by_nQ_A_alpha <- as_tibble(mt_nN_by_nQ_A_alpha) %>%
  rename_all(~c(ar_st_col_names))
# Show
kable(tb_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

3.3.2.2 mutate rowwise

- dplyr mutate own function
- dplyr all row function
- dplyr do function
- apply function each row dplyr
- applying a function to every row of a table using dplyr
- dplyr rowwise

```
# Define Implicit Function
ffi_linear_dplyrdo <- function(fl_A, fl_alpha, ar_nN_A, ar_nN_alpha){
  # ar_A_alpha[1] is A
  # ar_A_alpha[2] is alpha

  print(paste0('cur row, fl_A=', fl_A, ', fl_alpha=', fl_alpha))
  fl_out = sum(fl_A*ar_nN_A + 1/(fl_alpha + 1/ar_nN_alpha))

  return(fl_out)
}

# Evaluate function row by row of tibble
# fl_A, fl_alpha are from columns of tb_nN_by_nQ_A_alpha
tb_nN_by_nQ_A_alpha_show <- tb_nN_by_nQ_A_alpha %>%
```

fl_A	fl_alpha	dplyr_eval
-2	0.1	2.346356
-1	0.3	2.094273
0	0.5	1.895316
1	0.7	1.733708
2	0.9	1.599477

fl_A	fl_alpha	dplyr_eval
-2	0.1	2.346356
-1	0.3	2.094273
0	0.5	1.895316
1	0.7	1.733708
2	0.9	1.599477


```
  rowwise() %>%
  mutate(dplyr_eval =
           ffi_linear_dplyrdo(fl_A, fl_alpha, ar_nN_A, ar_nN_alpha))
```

```
## [1] "cur row, fl_A=-2, fl_alpha=0.1"
## [1] "cur row, fl_A=-1, fl_alpha=0.3"
## [1] "cur row, fl_A=0, fl_alpha=0.5"
## [1] "cur row, fl_A=1, fl_alpha=0.7"
## [1] "cur row, fl_A=2, fl_alpha=0.9"
```

```
# Show
kable(tb_nN_by_nQ_A_alpha_show) %>%
  kable_styling_fc()
```

same as before, still rowwise, but hard code some inputs:

```
# Define function, fixed inputs are not parameters, but
# defined earlier as a part of the function
# ar_nN_A, ar_nN_alpha are fixed, not parameters
ffi_linear_dplyrdo_func <- function(fl_A, fl_alpha){
  fl_out <- sum(fl_A*ar_nN_A + 1/(fl_alpha + 1/ar_nN_alpha))
  return(fl_out)
}

# Evaluate function row by row of tibble
tbfunc_A_nN_by_nQ_A_alpha_rowwise = tb_nN_by_nQ_A_alpha %>% rowwise() %>%
  mutate(dplyr_eval = ffi_linear_dplyrdo_func(fl_A, fl_alpha))
# Show
kable(tbfunc_A_nN_by_nQ_A_alpha_rowwise) %>%
  kable_styling_fc()
```

3.3.2.3 mutate with pmap

Apparently *rowwise()* is not a good idea, and *pmap* should be used, below is the *pmap* solution to the problem. Which does seem nicer. Crucially, don't have to define input parameter names, automatically I think they are matching up to the names in the function

- dplyr mutate pass function
- r function quosure string multiple
- r function multiple parameters as one string
- dplyr mutate anonymous function
- quosure style lambda
- pmap tibble rows
- dplyr pwalk

fl_A	fl_alpha	dplyr_eval_pmap
-2	0.1	2.346356
-1	0.3	2.094273
0	0.5	1.895316
1	0.7	1.733708
2	0.9	1.599477

```
# Define function, fixed inputs are not parameters, but defined
# earlier as a part of the function Rorate fl_alpha and fl_A name
# compared to before to make sure pmap tracks by names
ffi_linear_dplyrdo_func <- function(fl_alpha, fl_A){
  fl_out <- sum(fl_A*ar_nN_A + 1/(fl_alpha + 1/ar_nN_alpha))
  return(fl_out)
}

# Evaluate a function row by row of dataframe, generate list,
# then to vector
tb_nN_by_nQ_A_alpha %>% pmap(ffi_linear_dplyrdo_func) %>% unlist()
```

```
## [1] 2.346356 2.094273 1.895316 1.733708 1.599477
```

```
# Same as above, but in line line and save output as new column
# in dataframe note this ONLY works if the tibble only has variables
# that are inputs for the function if tibble contains additional
#  variables, those should be droppd, or only the ones needed selected,
# inside the pmap call below.
tbfunc_A_nN_by_nQ_A_alpha_pmap <- tb_nN_by_nQ_A_alpha %>%
  mutate(dplyr_eval_pmap =
           unlist(
             pmap(tb_nN_by_nQ_A_alpha, ffi_linear_dplyrdo_func)
           )
  )

# Show
kable(tbfunc_A_nN_by_nQ_A_alpha_pmap) %>%
  kable_styling_fc()
```

3.3.2.4 rowwise and do

Now, we have three types of parameters, for something like a bisection type calculation. We will supply the program with a function with some hard-coded value inside, and as parameters, we will have one parameter which is a row in the current matrix, and another parameter which is a sclar values. The three types of parameters are dealt with sparately:

1. parameters that are fixed for all bisection iterations, but differ for each row

- these are hard-coded into the function

2. parameters that are fixed for all bisection iterations, but are shared across rows

- these are the first parameter of the function, a list

3. parameters that differ for each iteration, but differ acoss iterations

- second scalar value parameter for the function
- dplyr mutate function applow to each row dot notation
- note rowwise might be bad according to Hadley, should use pmap?

```
ffi_linear_dplyrdo_fdot <- function(ls_row, fl_param){
  # Type 1 Param = ar_nN_A, ar_nN_alpha
```

fl_A	fl_alpha	dplyr_eval_flex
-2	0.1	2.346356
-1	0.3	2.094273
0	0.5	1.895316
1	0.7	1.733708
2	0.9	1.599477

```
  # Type 2 Param = ls_row$fl_A, ls_row$fl_alpha
  # Type 3 Param = fl_param

  fl_out <- (sum(ls_row$fl_A*ar_nN_A +
                  1/(ls_row$fl_alpha + 1/ar_nN_alpha))) + fl_param
  return(fl_out)
}

cur_func <- ffi_linear_dplyrdo_fdot
fl_param <- 0
dplyr_eval_flex <- tb_nN_by_nQ_A_alpha %>% rowwise() %>%
  do(dplyr_eval_flex = cur_func(., fl_param)) %>%
  unnest(dplyr_eval_flex)
tbfunc_B_nN_by_nQ_A_alpha <- tb_nN_by_nQ_A_alpha %>% add_column(dplyr_eval_flex)
# Show
kable(tbfunc_B_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

3.3.2.5 Compare Apply and Mutate Results

```
# Show overall Results
mt_results <- cbind(tb_nN_by_nQ_A_alpha_show['dplyr_eval'],
                    tbfunc_A_nN_by_nQ_A_alpha_rowwise['dplyr_eval'],
                    tbfunc_A_nN_by_nQ_A_alpha_pmap['dplyr_eval_pmap'],
                    tbfunc_B_nN_by_nQ_A_alpha['dplyr_eval_flex'],
                    mt_nN_by_nQ_A_alpha)
colnames(mt_results) <- c('eval_dplyr_mutate',
                          'eval_dplyr_mutate_hcode',
                          'eval_dplyr_mutate_pmap',
                          'eval_dplyr_mutate_flex',
                          'A_child', 'alpha_child')
kable(mt_results) %>%
  kable_styling_fc_wide()
```

eval_dplyr_mutate	eval_dplyr_mutate_hcode	eval_dplyr_mutate_pmap	eval_dplyr_mutate_flex	A_child	alpha_child
2.346356	2.346356	2.346356	2.346356	-2	0.1
2.094273	2.094273	2.094273	2.094273	-1	0.3
1.895316	1.895316	1.895316	1.895316	0	0.5
1.733708	1.733708	1.733708	1.733708	1	0.7
1.599477	1.599477	1.599477	1.599477	2	0.9

Chapter 4

Multi-dimensional Data Structures

4.1 Generate, Gather, Bind and Join

4.1.1 Generate Panel Structure

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

4.1.1.1 Balanced Panel Skeleton

There are N individuals, each could be observed M times. In the example below, there are 3 students, each observed over 4 dates. This just uses the uncount function from *tidyr*.

```
# Define
it_N <- 3
it_M <- 5
svr_id <- 'student_id'
svr_date <- 'class_day'

# dataframe
df_panel_skeleton <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number()) %>%
  ungroup()

# Print
kable(df_panel_skeleton) %>%
  kable_styling_fc()
```

4.1.1.2 Panel of Children with Height Growth

Given N individuals, each with G observations. There is an initial height variable and height grows every year. There are growth variables, variables for cumulative growth and variables for height at each age for each child.

Individuals are defined by gender (1 = female), race (1=asian), and birth height. Within individual yearly information includes height at each year of age.

```
# Define
it_N <- 5
it_M <- 3
svr_id <- 'indi_id'
svr_gender <- 'female'
svr_asian <- 'asian'
```

student_id	class_day
1	1
1	2
1	3
1	4
1	5
2	1
2	2
2	3
2	4
2	5
3	1
3	2
3	3
3	4
3	5

```
svr_age <- 'year_of_age'
# Define Height Related Variables
svr_brthgt <- 'birth_height'
svr_hgtgrow <- 'hgt_growth'
svr_hgtgrow_cumu <- 'hgt_growcumu'
svr_height <- 'height'

# panel dataframe following
set.seed(123)
df_panel_indiage <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  mutate(!!sym(svr_gender) := rbinom(n(), 1, 0.5),
         !!sym(svr_asian) := rbinom(n(), 1, 0.5),
         !!sym(svr_brthgt) := rnorm(n(), mean=60,sd=3)) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_gender), !!sym(svr_asian), !!sym(svr_brthgt)) %>%
  mutate(!!sym(svr_age) := row_number(),
         !!sym(svr_hgtgrow) := runif(n(), min=5, max=15),
         !!sym(svr_hgtgrow_cumu) := cumsum(!!sym(svr_hgtgrow)),
         !!sym(svr_height) := !!sym(svr_brthgt) + !!sym(svr_hgtgrow_cumu))  %>%
  ungroup()

# Add Height Index
kable(df_panel_indiage) %>% kable_styling_fc()
```

4.1.1.3 Create Group IDs

Given the dataframe just created, generate group IDs for each Gender and Race Groups. Given that both are binary, there can only be 4 unique groups.

```
# group id
svr_group_id <- 'female_asian_id'
# Define
ls_svr_group_vars <- c('female', 'asian')

# panel dataframe following
df_panel_indiage_id <- df_panel_indiage %>%
  arrange(!!!syms(ls_svr_group_vars)) %>%
  group_by(!!!syms(ls_svr_group_vars)) %>%
  mutate(!!sym(svr_group_id) := (row_number()==1)*1) %>%
  ungroup() %>%
```

female	asian	birth_height	year_of_age	hgt_growth	hgt_growcumu	height
0	0	65.14520	1	13.895393	13.895393	79.04059
0	0	65.14520	2	11.928034	25.823427	90.96862
0	0	65.14520	3	11.405068	37.228495	102.37369
1	1	61.38275	1	11.907053	11.907053	73.28980
1	1	61.38275	2	12.954674	24.861727	86.24448
1	1	61.38275	3	5.246137	30.107864	91.49061
0	1	56.20482	1	14.942698	14.942698	71.14751
0	1	56.20482	2	11.557058	26.499756	82.70457
0	1	56.20482	3	12.085305	38.585060	94.78988
1	1	57.93944	1	6.471137	6.471137	64.41058
1	1	57.93944	2	14.630242	21.101379	79.04082
1	1	57.93944	3	14.022991	35.124369	93.06381
1	0	58.66301	1	10.440660	10.440660	69.10367
1	0	58.66301	2	10.941420	21.382081	80.04509
1	0	58.66301	3	7.891597	29.273678	87.93669

```r
  mutate(!!sym(svr_group_id) := cumsum(!!sym(svr_group_id))) %>%
  select(one_of(svr_group_id, ls_svr_group_vars), everything())

# Add Height Index
kable(df_panel_indiage_id) %>%
  kable_styling_fc_wide()
```

female_asian_id	female	asian	birth_height	year_of_age	hgt_growth	hgt_growcumu	height
1	0	0	65.14520	1	13.895393	13.895393	79.04059
1	0	0	65.14520	2	11.928034	25.823427	90.96862
1	0	0	65.14520	3	11.405068	37.228495	102.37369
2	0	1	56.20482	1	14.942698	14.942698	71.14751
2	0	1	56.20482	2	11.557058	26.499756	82.70457
2	0	1	56.20482	3	12.085305	38.585060	94.78988
3	1	0	58.66301	1	10.440660	10.440660	69.10367
3	1	0	58.66301	2	10.941420	21.382081	80.04509
3	1	0	58.66301	3	7.891597	29.273678	87.93669
4	1	1	61.38275	1	11.907053	11.907053	73.28980
4	1	1	61.38275	2	12.954674	24.861727	86.24448
4	1	1	61.38275	3	5.246137	30.107864	91.49061
4	1	1	57.93944	1	6.471137	6.471137	64.41058
4	1	1	57.93944	2	14.630242	21.101379	79.04082
4	1	1	57.93944	3	14.022991	35.124369	93.06381

4.1.2 Join Datasets

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

4.1.2.1 Join Panel with Multiple Keys

We have two datasets, one for student enrollment, panel over time, but some students do not show up on some dates. The other is a skeleton panel with all student ID and all dates. Often we need to join dataframes together, and we need to join by the student ID and the panel time Key at the same time. When students show up, there is a quiz score for that day, so the joined panel should have as data column quiz score

Student count is N, total dates are M. First we generate two panels below, then we join by both keys using *left_join*. First, define dataframes:

sid	classday
1	1
1	2
1	3
2	1
2	2
2	3
3	1
3	2
3	3
4	1
4	2
4	3

sid	date_in_class	dayquizscore
1	1	89.88726
2	1	96.53929
2	2	65.59195
2	3	99.47356
4	2	97.36936

```r
# Define
it_N <- 4
it_M <- 3
svr_id <- 'sid'
svr_date <- 'classday'
svr_attend <- 'date_in_class'

# Panel Skeleton
df_panel_balanced_skeleton <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number()) %>%
  ungroup()
# Print
kable(df_panel_balanced_skeleton) %>%
  kable_styling_fc()
```

```r
# Smaller Panel of Random Days in School
set.seed(456)
df_panel_attend <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number()) %>%
  ungroup() %>% mutate(in_class = case_when(rnorm(n(),mean=0,sd=1) < 0 ~ 1, TRUE ~ 0)) %>%
  filter(in_class == 1) %>% select(!!sym(svr_id), !!sym(svr_date)) %>%
  rename(!!sym(svr_attend) := !!sym(svr_date)) %>%
  mutate(dayquizscore = rnorm(n(),mean=80,sd=10))
# Print
kable(df_panel_attend) %>%
  kable_styling_fc()
```

Second, now join dataframes:

```r
# Join with explicit names
df_quiz_joined_multikey <- df_panel_balanced_skeleton %>%
  left_join(df_panel_attend,
```

sid	classday	dayquizscore
1	1	89.88726
1	2	NA
1	3	NA
2	1	96.53929
2	2	65.59195
2	3	99.47356
3	1	NA
3	2	NA
3	3	NA
4	1	NA
4	2	97.36936
4	3	NA

sid	classday	dayquizscore
1	1	89.88726
1	2	NA
1	3	NA
2	1	96.53929
2	2	65.59195
2	3	99.47356
3	1	NA
3	2	NA
3	3	NA
4	1	NA
4	2	97.36936
4	3	NA

```
            by=(c('sid'='sid', 'classday'='date_in_class')))

# Join with setname strings
df_quiz_joined_multikey_setnames <- df_panel_balanced_skeleton %>%
  left_join(df_panel_attend, by=setNames(c('sid', 'date_in_class'), c('sid', 'classday')))

# Print
kable(df_quiz_joined_multikey) %>%
  kable_styling_fc()
```

```
kable(df_quiz_joined_multikey_setnames) %>%
  kable_styling_fc()
```

4.1.2.2 Stack Panel Frames Together

There are multiple panel dataframe, each for different subsets of dates. All variable names and units of observations are identical. Use DPLYR bind_rows.

```
# Define
it_N <- 2 # Number of individuals
it_M <- 3 # Number of Months
svr_id <- 'sid'
svr_date <- 'date'

# Panel First Half of Year
df_panel_m1tom3 <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number()) %>%
```

sid	date
1	1
1	2
1	3
2	1
2	2
2	3

sid	date
1	4
1	5
1	6
2	4
2	5
2	6

```
  ungroup()

# Panel Second Half of Year
df_panel_m4tom6 <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number() + 3) %>%
  ungroup()

# Bind Rows
df_panel_m1tm6 <- bind_rows(df_panel_m1tom3, df_panel_m4tom6) %>% arrange(!!!syms(c(svr_id, svr_date

# Print
kable(df_panel_m1tom3) %>%
  kable_styling_fc()
```

```
kable(df_panel_m4tom6) %>%
  kable_styling_fc()
```

```
kable(df_panel_m1tm6) %>%
  kable_styling_fc()
```

sid	date
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
2	3
2	4
2	5
2	6

4.1.3 Gather Files

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

4.1.3.1 Stack CSV Files Together Extract and Select Variables

There are multiple csv files, each was simulated with a different combination of parameters, each file has the same columns and perhaps even the same number of rows. We want to combine the files together, and provide correct attributes to rows from each table stacked, based on each underlying csv file's file name.

This is necessary, for example, when running computational exercises across EC2 instances in batch array and files are saved to different S3 folders. Need to gather parallel computational results together in a single file after syncing files locally with S3.

In the csv folder under this section, there are four subfolder, each containing 3 files with identical file structures.

We want to find the relevant csv files from these directories, and stack the results together.

1. File search search string, search in all subfolders, the search string contains file prefix that is common across files that need to be gathered.
2. Extract path folder hierarchy, each layer of folder is a different variable
3. Stack files together, with variables for file name and folder name
4. Extract from file name the component that is not in the search string, keep as separate variable
5. Follow specific rules about how file suffix is constructed to obtain additional variables.
6. Keep only a subset of columns of interest.

First, search and find all files with certain prefix.

```
# can search in multiple paths, second path here has no relevant contents
spt_roots <- c('C:/Users/fan/R4Econ/panel/basic/_file/csv',
               'C:/Users/fan/R4Econ/panel/basic/_file/tex')
# can skip file names with certain strings
spn_skip <- c('A3420')
# prefix search patther,
st_search_str <- 'solu_19E1NEp99r99x_ITG_PE_cev_*'

# Search and get all Path
ls_sfls <- list.files(path=spt_roots,
                      recursive=T,
                      pattern=st_search_str,
                      full.names=T)

# Skip path if contains words in skip list
if(!missing(spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(spn_skip, collapse = "|"), ls_sfls)]
}
```

Second, show all the files found, show their full path, the file name and the two folder names above the file name.

```
# Loop and print found files
it_folders_names_to_keep = 2
for (spt_file in ls_sfls) {
    ls_srt_folders_name_keep <- tail(strsplit(spt_file, "/")[[1]], n=it_folders_names_to_keep+1)
    snm_file_name <- tail(ls_srt_folders_name_keep, 1)
    ls_srt_folders_keep <- head(ls_srt_folders_name_keep, it_folders_names_to_keep)
    print(paste0('path:', spt_file))
    print(snm_file_name)
    print(ls_srt_folders_keep)
}
```

```
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev-2000/solu_19E1NEp99r99x_ITG_PE_cev_c0_cev
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0.csv"
## [1] "csv"      "cev-2000"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev-2000/solu_19E1NEp99r99x_ITG_PE_cev_c0_cev
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840.csv"
## [1] "csv"      "cev-2000"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev-947/solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0.csv"
## [1] "csv"     "cev-947"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev-947/solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840.csv"
## [1] "csv"     "cev-947"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev2000/solu_19E1NEp99r99x_ITG_PE_cev_c19_cev
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0.csv"
## [1] "csv"     "cev2000"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev2000/solu_19E1NEp99r99x_ITG_PE_cev_c19_cev
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840.csv"
## [1] "csv"     "cev2000"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev947/solu_19E1NEp99r99x_ITG_PE_cev_c14_cev9
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A0.csv"
## [1] "csv"    "cev947"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev947/solu_19E1NEp99r99x_ITG_PE_cev_c14_cev9
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A6840.csv"
## [1] "csv"    "cev947"
```

Third, create a dataframe with the folder and file names:

```
# String matrix empty
mt_st_paths_names <- matrix(data=NA, nrow=length(ls_sfls), ncol=4)

# Loop and print found files
it_folders_names_to_keep = 2
it_file_counter = 0
for (spt_file in ls_sfls) {
    # row counter
    it_file_counter = it_file_counter + 1

    # get file paths
    ls_srt_folders_name_keep <- tail(strsplit(spt_file, "/")[[1]], n=it_folders_names_to_keep+1)
    snm_file_name <- tail(ls_srt_folders_name_keep, 1)
    ls_srt_folders_keep <- head(ls_srt_folders_name_keep, it_folders_names_to_keep)

    # store
    # tools::file_path_sans_ext to drop suffix
    mt_st_paths_names[it_file_counter,1] = tools::file_path_sans_ext(snm_file_name)
    mt_st_paths_names[it_file_counter,2] = ls_srt_folders_keep[1]
    mt_st_paths_names[it_file_counter,3] = ls_srt_folders_keep[2]
    mt_st_paths_names[it_file_counter,4] = spt_file
}

# Column Names
ar_st_varnames <- c('fileid','name','folder1','folder2', 'fullpath')

# Combine to tibble, add name col1, col2, etc.
tb_csv_info <- as_tibble(mt_st_paths_names) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~c(ar_st_varnames))

# Display
```

fileid	name	folder1	folder2
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	csv	cev-2000
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	csv	cev-2000
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	csv	cev-947
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	csv	cev-947
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	csv	cev2000
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	csv	cev2000
7	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A0	csv	cev947
8	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A6840	csv	cev947

```
kable(tb_csv_info[,1:4]) %>% kable_styling_fc()
```

Fourth, create a dataframe by expanding each row with the datafile loaded in, use apply with anonymous function.

```
# Generate a list of dataframes
ls_df_loaded_files =
  apply(tb_csv_info,
        1,
        function(row) {
          # Loading file
          spn_full_path <- row[5]
          mt_csv = read.csv(file = spn_full_path)
          # dataframe
          it_fileid <- row[1]
          snm_filename <- row[2]
          srt_folder_level2 <- row[3]
          srt_folder_level1 <- row[4]

          tb_combine = as_tibble(mt_csv) %>%
            na.omit %>%
            rowid_to_column(var = "statesid") %>%
            mutate(fileid = it_fileid,
                   filename = snm_filename,
                   folder_lvl1 = srt_folder_level1,
                   folder_lvl2 = srt_folder_level2) %>%
            select(fileid, filename, folder_lvl1, folder_lvl2,
                   statesid, everything())
          # return
          return(tb_combine)
        })

# Stack dataframes together
df_all_files = do.call(bind_rows, ls_df_loaded_files)

# show stacked table
kable(df_all_files[seq(1,601,50),1:6]) %>% kable_styling_fc_wide()
```

Fifth, get additional information from the file name and file folder. Extract those as separate variables. The file names is dash connected, with various information. First, split just the final element of the string file name out, which is *A###*. Then, also extract the number next to N as a separate numeric column. Additional *folder_lvl1* separate out the numeric number from the initial word *cev*.

Split "solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A###" to "solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000" and "A###":

```
# separate last eleemtnafter underscore
df_all_files_finalA <- df_all_files %>%
   separate(filename, into = c("filename_main", "prod_type_st"),
```

fileid	filename	folder_lvl1	folder_lvl2	statesid	EjV
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	cev-2000	csv	1	-28.8586860
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	cev-2000	csv	51	-0.2106603
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	cev-2000	csv	3	-28.8586860
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	cev-2000	csv	53	-0.0642997
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	cev-947	csv	5	-5.8826609
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	cev-947	csv	55	0.0353187
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	cev-947	csv	7	-2.7046907
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	cev-947	csv	57	0.1094474
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	cev2000	csv	9	-2.9782236
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	cev2000	csv	59	0.3389275
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	cev2000	csv	11	-1.7229647
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	cev2000	csv	61	-14.6880377
7	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A0	cev947	csv	13	-1.7623279

```
            sep="_(?=[^_]+$)",
            remove = FALSE) %>%
   select(fileid, filename, filename_main, prod_type_st, folder_lvl1, folder_lvl2,
          statesid, everything())
# show stacked table
kable(df_all_files_finalA[seq(1,601,50),1:10]) %>% kable_styling_fc_wide()
```

fileid	filename	filename_main	prod_type_st	folder_lvl1	folder_lvl2	statesid	EjV	k_tt	b_tt
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000	A0	cev-2000	csv	1	-28.8586860	0.000000	0.000000
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000	A0	cev-2000	csv	51	-0.2106603	0.000000	78.005300
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000	A6840	cev-2000	csv	3	-28.8586860	0.000000	0.000000
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000	A6840	cev-2000	csv	53	-0.0642997	0.000000	84.215368
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947	A0	cev-947	csv	5	-5.8826609	0.000000	3.869909
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947	A0	cev-947	csv	55	0.0353187	0.000000	90.611739
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947	A6840	cev-947	csv	7	-2.7046907	0.000000	7.855916
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947	A6840	cev-947	csv	57	0.1094474	0.000000	90.611739
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000	A0	cev2000	csv	9	-2.9782236	0.000000	7.855916
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000	A0	cev2000	csv	59	0.3389275	0.000000	97.200000
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000	A6840	cev2000	csv	11	-1.7229647	0.000000	11.961502
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000	A6840	cev2000	csv	61	-14.6880377	1.990694	-1.879215
7	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A0	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947	A0	cev947	csv	13	-1.7623279	0.000000	16.190257

Split "A###" to "A" and "A###". Additionally, also split *cev####* to *cev* and *####*, allow for positive and negative numbers. See regular expression 101 helper

```
# string and number separation
df_all_files_finalB <- df_all_files_finalA %>%
  separate(prod_type_st,
           into = c("prod_type_st_prefix", "prod_type_lvl"),
           sep="(?<=[A-Za-z])(?=[-0-9])", # positive or negative numbers
           remove=FALSE) %>%
  separate(folder_lvl1,
           into = c("cev_prefix", "cev_lvl"),
           sep="(?<=[A-Za-z])(?=[-0-9])", # positive or negative numbers
           remove=FALSE) %>%
  mutate(cev_st = folder_lvl1,
         prod_type_lvl = as.numeric(prod_type_lvl),
         cev_lvl = as.numeric(cev_lvl)/10000) %>%
  select(fileid,
         prod_type_st, prod_type_lvl,
         cev_st, cev_lvl,
         statesid, EjV,
         filename, folder_lvl1, folder_lvl2)
# Ordering, sort by cev_lvl, then prod_type_lvl, then stateid
df_all_files_finalB <- df_all_files_finalB %>%
  arrange(cev_lvl, prod_type_lvl, statesid)
# show stacked table
kable(df_all_files_finalB[seq(1,49*16,49),1:7]) %>% kable_styling_fc_wide()
```

fileid	prod_type_st	prod_type_lvl	cev_st	cev_lvl	statesid	EjV
1	A0	0	cev-2000	-0.2000	1	-28.8586860
1	A0	0	cev-2000	-0.2000	50	-0.2106603
2	A6840	6840	cev-2000	-0.2000	1	-28.8586860
2	A6840	6840	cev-2000	-0.2000	50	-0.1311749
3	A0	0	cev-947	-0.0947	1	-28.0399281
3	A0	0	cev-947	-0.0947	50	-0.0911499
4	A6840	6840	cev-947	-0.0947	1	-28.0399281
4	A6840	6840	cev-947	-0.0947	50	-0.0134719
7	A0	0	cev947	0.0947	1	-26.8243673
7	A0	0	cev947	0.0947	50	0.0857474
8	A6840	6840	cev947	0.0947	1	-26.8243673
8	A6840	6840	cev947	0.0947	50	0.1608382
5	A0	0	cev2000	0.2000	1	-26.2512036
5	A0	0	cev2000	0.2000	50	0.1694524
6	A6840	6840	cev2000	0.2000	1	-26.2512036
6	A6840	6840	cev2000	0.2000	50	0.2432677

4.2 Wide and Long

4.2.1 Long Table to Wide Table

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Using the pivot_wider function in tidyr to reshape panel or other data structures

4.2.1.1 Compute Wide Table Cumulative Student Attendance based on Long Table Roster

There are N students in class, but only a subset of them attend class each day. If student id_i is in class on day Q, the teacher records on a sheet the date and the student ID. So if the student has been in class 10 times, the teacher has ten rows of recorded data for the student with two columns: column one is the student ID, and column two is the date on which the student was in class.

Suppose there were 50 students, who on average attended exactly 10 classes each during the semester, this means we have $10 \cdot 50$ rows of data, with differing numbers of rows for each student. This is shown as *df_panel_attend_date* generated below.

Now we want to generate a new dataframe, where each row is a date, and each column is a student. The values in the new dataframe shows, at the Q^{th} day (row), how many classes student i has attended so far. The following results is also in a REconTools Function. This is shown as *df_attend_cumu_by_day* generated below.

First, generate the raw data structure, *df_panel_attend_date*:

```
# Define
it_N <- 3
it_M <- 5
svr_id <- 'student_id'

# from : support/rand/fs_rand_draws.Rmd
set.seed(222)
df_panel_attend_date <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(date = row_number()) %>%
  ungroup() %>%
```

student_id	date_in_class
1	2
1	4
2	1
2	2
2	5
3	2
3	3
3	5

student_id	date_in_class	attended
1	2	1
1	4	1
2	1	1
2	2	1
2	5	1
3	2	1
3	3	1
3	5	1

```
  mutate(in_class = case_when(rnorm(n(),mean=0,sd=1) < 0 ~ 1, TRUE ~ 0)) %>%
  filter(in_class == 1) %>% select(!!sym(svr_id), date) %>%
  rename(date_in_class = date)

# Print
kable(df_panel_attend_date) %>%
  kable_styling_fc()
```

Second, we create a attendance column that is all 1. This is not useful for the long table, but useful for our conversion to wide.

```
# Define
svr_id <- 'student_id'
svr_date <- 'date_in_class'
st_idcol_prefix <- 'sid_'

# Generate cumulative enrollment counts by date
df_panel_attend_date_addone <- df_panel_attend_date %>% mutate(attended = 1)
kable(df_panel_attend_date_addone) %>%
  kable_styling_fc()
```

Third, we convert the long table to wide. The unit of observation is student-day for the long table, and day for the wide table.

```
# Pivot Wide
df_panel_attend_date_wider <- df_panel_attend_date_addone %>%
  arrange(student_id) %>%
  pivot_wider(names_from = svr_id,
              values_from = attended)

# Sort and rename
# rename see: https://fanwangecon.github.io/R4Econ/amto/tibble/fs_tib_basics.html
ar_unique_ids <- sort(unique(df_panel_attend_date %>% pull(!!sym(svr_id))))
df_panel_attend_date_wider_sort <- df_panel_attend_date_wider %>%
    arrange(!!sym(svr_date)) %>%
    rename_at(vars(num_range('',ar_unique_ids))
              , list(~paste0(st_idcol_prefix, . , ''))
              )
```

date_in_class	sid_1	sid_2	sid_3
1	NA	1	NA
2	1	1	1
3	NA	NA	1
4	1	NA	NA
5	NA	1	1

date_in_class	sid_1	sid_2	sid_3
1	NA	1	NA
2	1	1	1
3	NA	NA	1
4	1	NA	NA
5	NA	1	1

```
kable(df_panel_attend_date_wider_sort) %>%
  kable_styling_fc()
```

Fourth, we could achieve what we have above by specifying more parameters in the *pivot_wider* function.

```
# Include name_prefix
df_panel_attend_date_wider_sort <- df_panel_attend_date_addone %>%
  arrange(student_id) %>%
  pivot_wider(id_cols = c("date_in_class"),
              names_from = svr_id,
              names_prefix = "sid_",
              values_from = attended) %>%
  arrange(date_in_class)
# Print
kable(df_panel_attend_date_wider_sort) %>%
  kable_styling_fc()
```

Fifth, sum across rows for each student (column) to get cumulative attendance for each student on each date.

```
# replace NA and cumusum again
# see: R4Econ/support/function/fs_func_multivar for renaming and replacing
df_attend_cumu_by_day <- df_panel_attend_date_wider_sort %>%
  mutate_at(vars(contains(st_idcol_prefix)), list(~replace_na(., 0))) %>%
  mutate_at(vars(contains(st_idcol_prefix)), list(~cumsum(.)))

kable(df_attend_cumu_by_day) %>%
  kable_styling_fc()
```

Finally, the structure above is also a function in Fan's REconTools Package, here the function is tested:

```
# Parameters
df <- df_panel_attend_date
svr_id_i <- 'student_id'
svr_id_t <- 'date_in_class'
st_idcol_prefix <- 'sid_'
```

date_in_class	sid_1	sid_2	sid_3
1	0	1	0
2	1	2	1
3	1	2	2
4	2	2	2
5	2	3	3

Attend and score info

student_id	date_in_class	attended	score	other_var_1	other_var_2
1	2	1	64.40	1	2
1	4	1	67.70	1	2
2	1	1	85.59	1	2
2	2	1	70.71	1	2
2	5	1	71.29	1	2
3	2	1	87.15	1	2
3	3	1	74.61	1	2
3	5	1	57.35	1	2

```r
# Invoke Function
ls_df_rosterwide <- ff_panel_expand_longrosterwide(df, svr_id_t, svr_id_i, st_idcol_prefix)
df_roster_wide_func <- ls_df_rosterwide$df_roster_wide
df_roster_wide_cumu_func <- ls_df_rosterwide$df_roster_wide_cumu

# Print
print(df_roster_wide_func)
print(df_roster_wide_cumu_func)
```

4.2.1.2 Panel Long Attendance Roster and Score Card to Wide

In the prior example, at each date, we only had information about attendance, however, we might also know the exam score on each day when the student attends school. In the long table, this appears, in addition to *attended*, as an additional variable *score*. When we convert from long to wide, we will have 3 new columns for attendance and also 3 new columns for score. The 3 columns are for the three students, there will be five rows for the five days. Each row in the wide table is the attendance and score information for each day.

First, we add a random score column to the long dataframe created prior. Also add two other additional columns.

```r
# Create score column
set.seed(123)
df_panel_attend_score_date <- df_panel_attend_date_addone %>%
  mutate(score = rnorm(dim(df_panel_attend_date_addone)[1], mean=70, sd=10)) %>%
  mutate(score = round(score, 2),
         other_var_1 = 1, other_var_2 = 2)

# Print
kable(df_panel_attend_score_date, caption="Attend and score info") %>%
  kable_styling_fc()
```

Second, convert both attended and score columns to wide at the same time. Note that we add "sid" in front of the index for each student. Note that *id_cols* picks the columns to keep in addition to the names_from and values_from columns. In this case, we are not keeping *other_var_1* and *other_var_2*.

```r
# Convert to wide
df_panel_attend_score_date_wide <- df_panel_attend_score_date %>%
  arrange(student_id) %>%
  pivot_wider(id_cols  = c("date_in_class"),
              names_from = svr_id,
              names_prefix = "sid",
              names_sep = "_",
              values_from = c(attended, score)) %>%
  arrange(date_in_class)

# Print
```

```
kable(df_panel_attend_score_date_wide, caption="Attend and score wide") %>%
  kable_styling_fc_wide()
```

Attend and score wide

date_in_class	attended_sid1	attended_sid2	attended_sid3	score_sid1	score_sid2	score_sid3
1	NA	1	NA	NA	85.59	NA
2	1	1	1	64.4	70.71	87.15
3	NA	NA	1	NA	NA	74.61
4	1	NA	NA	67.7	NA	NA
5	NA	1	1	NA	71.29	57.35

4.2.2 Wide to Long

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Using the pivot_wider function in tidyr to reshape panel or other data structures

4.2.2.1 Wide to long panel, single variable

We have a matrix of values, the values are ev. Each row corresponds to a different value of the a variable, each column represents a different value of the z variable.

Based on this matrix, we create a table where each unit of observation is for a specific a and z variable combination. So the matrix is turned from wide to long.

The resulting long table has 5 variables

1. a: values of the a variable, the original matrix row labels
2. ai: an index from 1, indicating the original matrix row index
3. z: values of the z variable, the original matrix column lables
4. zi: an index from 1, indicating hte original matrix column index

First, we create the matrix.

```
# Generate A Matrix
set.seed(123)
ar_a <- c(1.1,5.1)
ar_z <- seq(-2.5, 2.53, length.out=11)
mt_ev = matrix(rnorm(length(ar_a)*length(ar_z)),
  nrow=length(ar_a), ncol=length(ar_z))

# Name Matrix
rownames(mt_ev) <- paste0('ai', seq(1:length(ar_a)))
colnames(mt_ev) <- paste0('zi', seq(1:length(ar_z)))

# to tibble
tb_ev <- as_tibble(mt_ev) %>% rowid_to_column(var = "ai")

# Print
print(mt_ev)
```

```
##            zi1        zi2       zi3        zi4        zi5       zi6       zi7
## ai1 -0.5604756 1.55870831 0.1292877  0.4609162 -0.6868529 1.2240818 0.4007715
## ai2 -0.2301775 0.07050839 1.7150650 -1.2650612 -0.4456620 0.3598138 0.1106827
##            zi8        zi9       zi10       zi11
## ai1 -0.5558411  0.4978505  0.7013559 -1.0678237
## ai2  1.7869131 -1.9666172 -0.4727914 -0.2179749
```

Wide table

ai	zi1	zi2	zi3	zi4	zi5	zi6	zi7	zi8	zi9
1	-0.5604756	1.5587083	0.1292877	0.4609162	-0.6868529	1.2240818	0.4007715	-0.5558411	0.4978505
2	-0.2301775	0.0705084	1.7150650	-1.2650612	-0.4456620	0.3598138	0.1106827	1.7869131	-1.9666172

```r
# Display
kable(tb_ev, caption = "Wide table") %>% kable_styling_fc()
```

Second, we convert the table wide to long.

```r
# longer
tb_ev_long <- tb_ev %>%
  pivot_longer(cols = starts_with('zi'),
               names_to = c('zi'),
               names_pattern = paste0("zi(.*)"),
               values_to = "ev") %>%
  mutate(zi = as.numeric(zi))

# Merge with a and z values
tb_ev_long <- tb_ev_long %>%
  left_join(as_tibble(ar_a) %>%
              rowid_to_column(var = "ai") %>%
              rename(a = value)
              , by = 'ai') %>%
  left_join(as_tibble(ar_z) %>%
              rowid_to_column(var = "zi") %>%
              rename(z = value),
            by = 'zi') %>%
  select(a,ai,z,zi,ev)

# Display
kable(tb_ev_long, caption = "Long table") %>% kable_styling_fc()
```

4.2.2.2 Wide to long panel, multiple variables

We have a dataset where each row contains data from a different year. We have four variables, observed wage, simulated wage, observed labor quantities, and simulated labor quantities.

We generate reshape this file to have four variables:

1. year
2. categorical for wage or quantity
3. categorical for observed or simulated
4. a numerical column with wage and quantity values

This is different then the situation prior, because we are need to convert to long two different numerical variables that will be in the same long variable, but differentiated by two categorical variables (rather than one).

First, we create the matrix.

```r
# Generate A Matrix
set.seed(123)
ar_year <- c(1995, 1997, 1999)
ar_vars <- c("wage_model", "quant_model", "wage_simu", "quant_simu")
mt_equi = matrix(rnorm(length(ar_year)*length(ar_vars)),
  nrow=length(ar_year), ncol=length(ar_vars))

# Name Matrix
```

Long table

a	ai	z	zi	ev
1.1	1	-2.500	1	-0.5604756
1.1	1	-1.997	2	1.5587083
1.1	1	-1.494	3	0.1292877
1.1	1	-0.991	4	0.4609162
1.1	1	-0.488	5	-0.6868529
1.1	1	0.015	6	1.2240818
1.1	1	0.518	7	0.4007715
1.1	1	1.021	8	-0.5558411
1.1	1	1.524	9	0.4978505
1.1	1	2.027	10	0.7013559
1.1	1	2.530	11	-1.0678237
5.1	2	-2.500	1	-0.2301775
5.1	2	-1.997	2	0.0705084
5.1	2	-1.494	3	1.7150650
5.1	2	-0.991	4	-1.2650612
5.1	2	-0.488	5	-0.4456620
5.1	2	0.015	6	0.3598138
5.1	2	0.518	7	0.1106827
5.1	2	1.021	8	1.7869131
5.1	2	1.524	9	-1.9666172
5.1	2	2.027	10	-0.4727914
5.1	2	2.530	11	-0.2179749

Wide table

year	wage_model	quant_model	wage_simu	quant_simu
1995	-0.5604756	0.0705084	0.4609162	-0.4456620
1997	-0.2301775	0.1292877	-1.2650612	1.2240818
1999	1.5587083	1.7150650	-0.6868529	0.3598138

```
rownames(mt_equi) <- ar_year
colnames(mt_equi) <- ar_vars

# to tibble
tb_equi <- as_tibble(mt_equi, rownames = "year")

# Print
print(mt_equi)
```

```
##      wage_model quant_model  wage_simu quant_simu
## 1995 -0.5604756  0.07050839  0.4609162 -0.4456620
## 1997 -0.2301775  0.12928774 -1.2650612  1.2240818
## 1999  1.5587083  1.71506499 -0.6868529  0.3598138
```

```
# Display
kable(tb_equi, caption = "Wide table") %>% kable_styling_fc()
```

Second, we convert the table wide to long. We select columns that includes either wage or quant, see tidyselect Select variables that match a pattern for additional verbs for how to select variables.

```
# longer
tb_equi_long <- tb_equi %>%
  pivot_longer(cols = matches('wage|quant'),
               names_to = c('variable', 'source'),
               names_pattern = paste0("(.*)_(.*)"),
```

Long table, Two Variables

year	variable	source	value
1995	wage	model	-0.5604756
1995	quant	model	0.0705084
1995	wage	simu	0.4609162
1995	quant	simu	-0.4456620
1997	wage	model	-0.2301775
1997	quant	model	0.1292877
1997	wage	simu	-1.2650612
1997	quant	simu	1.2240818
1999	wage	model	1.5587083
1999	quant	model	1.7150650
1999	wage	simu	-0.6868529
1999	quant	simu	0.3598138

```
               values_to = "value")

# Display
kable(tb_equi_long, caption = "Long table, Two Variables") %>% kable_styling_fc()
```

4.3 Within Panel Comparisons and Statistics

4.3.1 Find Closest Neighbor on Grid

> Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Using the pivot_wider function in tidyr to reshape panel or other data structures

4.3.1.1 Closest Neighbor on Grid

There is a dataframe that provides $V(coh, a, cev)$ levels. There is another dataframe with $\hat{V}(coh, a)$, for each coh, a, find the cev that such that the difference between $\hat{V}(coh, a)$ and $V(coh, a, cev)$ is minimized.

V and $\hat{V}$ information are stored in a dataframe in the csv folder in the current directory. In fact, we have one V surface, but multiple $\hat{V}$ files, so we want to do the find closest neighbor exercise for each one of the several $\hat{V}$ files.

The structure is as follows: (1) Load in the V file, where coh, a, cev are all variable attributes. (2) Merge with one $\hat{V}$ file. (3) Take the difference between the V and $\hat{V}$ columns, and take the absolute value of the difference. (4) Group by coh, a, and sort to get the smallest absolute difference among the cev possibilities, and slice out the row for the smallest. (5) Now We have $V(coh, a, cev^{\star}(coh, a))$. (6) Do this for each of the several $\hat{V}$ files. (7) Stack the results from 1 through 6 together, generate a column that identifies which simulation/exercise/counterfactual each of the $\hat{V}$ file comes from. (8) Visualize by plotting as subplot different a, coh is x-axis, different $\hat{V}$ outcome are different lines, and $cev^{\star}\left(coh, a, \hat{V}\right)$ is the y-axis outcome.

First, load the CEV file.

```
# folder
spt_root <- c('C:/Users/fan/R4Econ/panel/join/_file/csv')
# cev surface file, the V file
snm_cev_surface <- 'e_19E1NEp99r99_ITG_PE_cev_subsettest.csv'
mt_cev_surface <- read.csv(file = file.path(spt_root, snm_cev_surface))
tb_cev_surface <- as_tibble(mt_cev_surface) %>%
  rename(EjVcev = EjV)
```

Second, loop over the V hat files, join V with V hat:

```
ls_tb_cev_surfhat = vector(mode = "list", length = 4)
for (it_simu_counter in c(1,2,3,4)) {

    # conditionally change file names
    if (it_simu_counter == 1) {
        st_counter <- '19E1NEp99r99'
    } else if (it_simu_counter == 2) {
        st_counter <- '19E1NEp02r99'
    } else if (it_simu_counter == 3) {
        st_counter <- '19E1NEp02per02ger99'
    } else if (it_simu_counter == 4) {
        st_counter <- '19E1NEp02r02'
    }
    snm_v_hat <- paste0('e_', st_counter, '_ITG_PE_subsettest.csv')

    # Overall path to files
    mt_v_hat <- read.csv(file = file.path(spt_root, snm_v_hat))
    tb_v_hat <- as_tibble(mt_v_hat) %>%
      select(prod_type_lvl, statesid, EjV)

    # Merge file using key
    tb_cev_surfhat <- tb_cev_surface %>%
      left_join(tb_v_hat, by=(c('prod_type_lvl'='prod_type_lvl',
                                'statesid'='statesid'))) %>%
      arrange(statesid, prod_type_lvl, cev_lvl) %>%
      mutate(counter_policy = st_counter)

    # Store to list
    ls_tb_cev_surfhat[[it_simu_counter]] <- tb_cev_surfhat
}

# Display
kable(ls_tb_cev_surfhat[[1]][seq(1, 40, 5),]) %>% kable_styling_fc_wide()
```

X	cev_st	cev_lvl	prod_type_st	prod_type_lvl	statesid	cash_tt	EjVcev	EjV	counter_policy
1	cev-2000	-0.2000	A0	0	526	32.84747	-1.0479929	-0.7957419	19E1NEp99r99
1501	cev-947	-0.0947	A0	0	526	32.84747	-0.9079859	-0.7957419	19E1NEp99r99
3001	cev105	0.0105	A0	0	526	32.84747	-0.7880156	-0.7957419	19E1NEp99r99
4501	cev1157	0.1157	A0	0	526	32.84747	-0.6803586	-0.7957419	19E1NEp99r99
51	cev-2000	-0.2000	A2504	2504	526	32.90371	-1.0002921	-0.7504785	19E1NEp99r99
1551	cev-947	-0.0947	A2504	2504	526	32.90371	-0.8613743	-0.7504785	19E1NEp99r99
3051	cev105	0.0105	A2504	2504	526	32.90371	-0.7423281	-0.7504785	19E1NEp99r99
4551	cev1157	0.1157	A2504	2504	526	32.90371	-0.6354620	-0.7504785	19E1NEp99r99

Third, sort each file, and keep only the best match rows that minimize the absolute distance between *EjV* and *EjVcev*.

```
ls_tb_cev_matched = vector(mode = "list", length = 4)
for (it_simu_counter in c(1,2,3,4)) {

    # Load merged file
    tb_cev_surfhat <- ls_tb_cev_surfhat[[it_simu_counter]]

    # Difference Column
    tb_cev_surfhat <- tb_cev_surfhat %>%
      mutate(EjVcev_gap = abs(EjVcev - EjV))

    # Group by, Arrange and Slice, get lowest gap
    tb_cev_matched <- tb_cev_surfhat %>%
      arrange(statesid, prod_type_lvl, EjVcev_gap) %>%
```

```
      group_by(statesid, prod_type_lvl) %>%
      slice_head(n=1)

    # Store to list
    ls_tb_cev_matched[[it_simu_counter]] <- tb_cev_matched
}

# Display
kable(ls_tb_cev_matched[[2]][seq(1, 30, 1),]) %>% kable_styling_fc_wide()
```

X	cev_st	cev_lvl	prod_type_st	prod_type_lvl	statesid	cash_tt	EjVcev	EjV	counter_policy	EjVcev_gap
3001	cev105	0.0105	A0	0	526	32.847471	-0.7880156	-0.7928034	19E1NEp02r99	0.0047878
3051	cev105	0.0105	A2504	2504	526	32.903714	-0.7423281	-0.7480617	19E1NEp02r99	0.0057336
3101	cev105	0.0105	A4145	4145	526	32.948970	-0.7082006	-0.7145418	19E1NEp02r99	0.0063412
3151	cev105	0.0105	A5633	5633	526	32.996952	-0.6753576	-0.6818996	19E1NEp02r99	0.0065420
3201	cev105	0.0105	A7274	7274	526	33.058832	-0.6368297	-0.6431710	19E1NEp02r99	0.0063413
3251	cev105	0.0105	A9779	9779	526	33.175241	-0.5711706	-0.5774648	19E1NEp02r99	0.0062942
3002	cev105	0.0105	A0	0	555	53.346587	-0.2985944	-0.3041922	19E1NEp02r99	0.0055978
3052	cev105	0.0105	A2504	2504	555	53.815772	-0.2617572	-0.2680026	19E1NEp02r99	0.0062454
3102	cev105	0.0105	A4145	4145	555	54.193302	-0.2340822	-0.2406142	19E1NEp02r99	0.0065320
3152	cev105	0.0105	A5633	5633	555	54.593579	-0.2067964	-0.2134634	19E1NEp02r99	0.0066670
3202	cev105	0.0105	A7274	7274	555	55.109790	-0.1740126	-0.1806320	19E1NEp02r99	0.0066194
3252	cev105	0.0105	A9779	9779	555	56.080888	-0.1169470	-0.1236111	19E1NEp02r99	0.0066641
3603	cev526	0.0526	A0	0	905	1.533025	-5.2530406	-5.2486887	19E1NEp02r99	0.0043519
3353	cev315	0.0315	A2504	2504	905	1.714498	-4.5517474	-4.5408560	19E1NEp02r99	0.0108913
3403	cev315	0.0315	A4145	4145	905	1.860521	-4.1039608	-4.1072736	19E1NEp02r99	0.0033128
3453	cev315	0.0315	A5633	5633	905	2.015341	-3.7465733	-3.7611842	19E1NEp02r99	0.0146109
3503	cev315	0.0315	A7274	7274	905	2.215003	-3.4101025	-3.4235413	19E1NEp02r99	0.0134388
3553	cev315	0.0315	A9779	9779	905	2.590608	-2.9413469	-2.9535570	19E1NEp02r99	0.0122101
3004	cev105	0.0105	A0	0	953	20.125381	-1.3249909	-1.3290865	19E1NEp02r99	0.0040957
3054	cev105	0.0105	A2504	2504	953	20.306854	-1.2476021	-1.2531860	19E1NEp02r99	0.0055839
3104	cev105	0.0105	A4145	4145	953	20.452876	-1.1916003	-1.1975215	19E1NEp02r99	0.0059211
3154	cev105	0.0105	A5633	5633	953	20.607697	-1.1383665	-1.1444048	19E1NEp02r99	0.0060383
3204	cev105	0.0105	A7274	7274	953	20.807359	-1.0766095	-1.0823344	19E1NEp02r99	0.0057250
3254	cev105	0.0105	A9779	9779	953	21.182964	-0.9729832	-0.9781408	19E1NEp02r99	0.0051576
3005	cev105	0.0105	A0	0	1017	63.774766	-0.1284542	-0.1342653	19E1NEp02r99	0.0058110
3055	cev105	0.0105	A2504	2504	1017	64.298911	-0.0967695	-0.1031112	19E1NEp02r99	0.0063417
3105	cev105	0.0105	A4145	4145	1017	64.720664	-0.0728485	-0.0793940	19E1NEp02r99	0.0065454
3155	cev105	0.0105	A5633	5633	1017	65.167829	-0.0490898	-0.0557238	19E1NEp02r99	0.0066341
3205	cev105	0.0105	A7274	7274	1017	65.744507	-0.0203378	-0.0269149	19E1NEp02r99	0.0065772
3255	cev105	0.0105	A9779	9779	1017	66.829359	0.0299397	0.0233507	19E1NEp02r99	0.0065890

Fourth, row_bind results together.

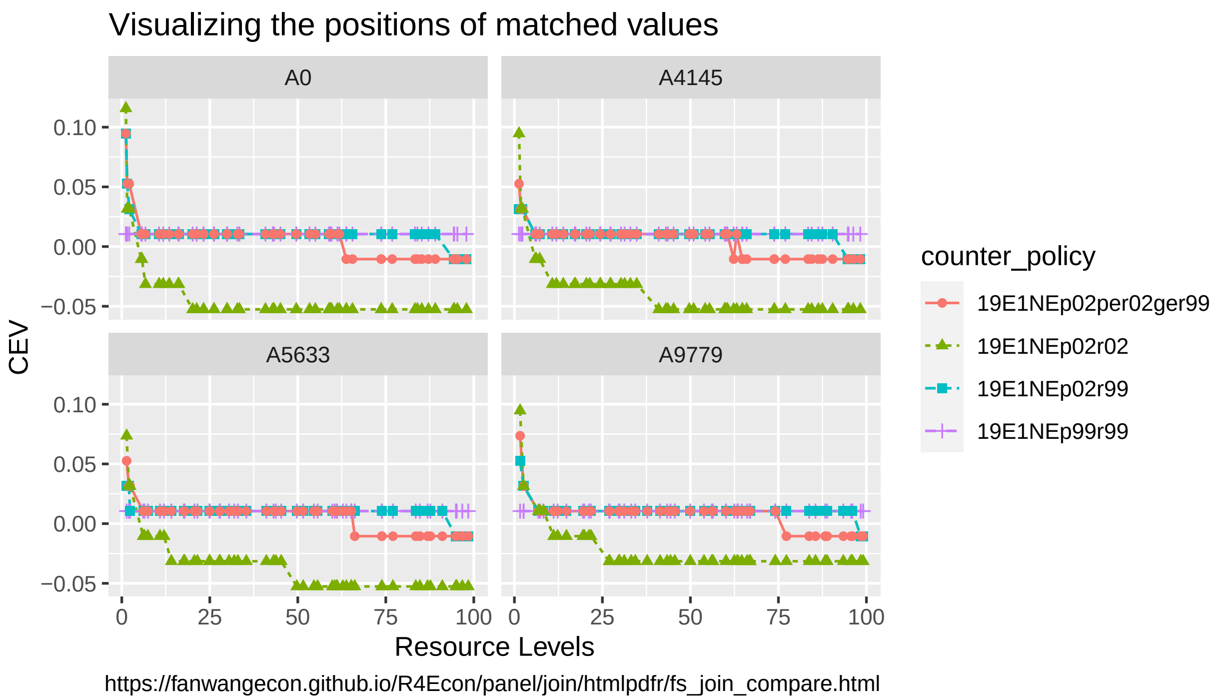
```
# Single dataframe with all results
tb_cev_matched_all_counter <- do.call(bind_rows, ls_tb_cev_matched)
# check size
print(dim(tb_cev_matched_all_counter))
```

```
## [1] 1200   11
```

Fifth, visualize results

```
# select four from the productivity types
ar_prod_type_lvl_unique <- unique(tb_cev_matched_all_counter %>% pull(prod_type_lvl))
ar_prod_type_lvl_selected <- ar_prod_type_lvl_unique[round(seq(1, length(ar_prod_type_lvl_unique), l
# graph
lineplot <- tb_cev_matched_all_counter %>%
    filter(prod_type_lvl %in% ar_prod_type_lvl_selected) %>%
    group_by(prod_type_st, cash_tt) %>%
    ggplot(aes(x=cash_tt, y=cev_lvl,
               colour=counter_policy, linetype=counter_policy, shape=counter_policy)) +
        facet_wrap( ~ prod_type_st) +
        geom_line() +
        geom_point() +
        labs(title = 'Visualizing the positions of matched values',
             x = 'Resource Levels',
             y = 'CEV',
```

```
            caption = paste0('https://fanwangecon.github.io/',
                             'R4Econ/panel/join/htmlpdfr/fs_join_compare.html'))
print(lineplot)
```



4.3.2 Within Panel Cross-time and Cross-group Statistics

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

4.3.2.1 Comparing Three Countries over Time

Given three time series for three "countries", we compute percentage change from initial year for each country, and compare relative values within each timer period versus one country.

First, we generate the core data inputs. We assume that the output here would be the data structure we face prior to generating the figures we are interested in. We use data from the attitude dataset, but re-interpret what the columns are. We work with data from three "countries" at the same time, which generalizes also to the two countries case.

```
# Load data, and treat index as "year"
# pretend data to be country-data
df_attitude <- as_tibble(attitude) %>%
  rowid_to_column(var = "year") %>%
  select(year, rating, complaints, learning) %>%
  rename(stats_usa = rating,
         stats_canada = complaints,
         stats_uk = learning)

# Wide to Long
df_attitude <- df_attitude %>%
  pivot_longer(cols = starts_with('stats_'),
               names_to = c('country'),
               names_pattern = paste0("stats_(.*)"),
               values_to = "rating")
```

year	country	rating
1	usa	43
1	canada	51
1	uk	39
2	usa	63
2	canada	64
2	uk	54
3	usa	71
3	canada	70
3	uk	69
4	usa	61

year	country	rating	ratings_ratio_vs_countrycanada
1	canada	51	1.0000000
1	uk	39	0.7647059
1	usa	43	0.8431373
2	canada	64	1.0000000
2	uk	54	0.8437500
2	usa	63	0.9843750
3	canada	70	1.0000000
3	uk	69	0.9857143
3	usa	71	1.0142857
4	canada	63	1.0000000

```
# Print
kable(df_attitude[1:10,]) %>% kable_styling_fc()
```

Second, we generate additional data inputs. Specifically, we also generate ratios of values with respect to he “first” country, within each year.

```
# Sort and get list of countries
ar_countries_sorted <- df_attitude %>%
  ungroup() %>% distinct(country) %>% arrange(country) %>%
  pull(country)
st_ratio_var <- paste0('ratings_ratio_vs_country', ar_countries_sorted[1])

# Generate ratio over the first location
df_attitude <- df_attitude %>%
  arrange(year, country) %>% group_by(year) %>%
  mutate(!!sym(st_ratio_var) := rating/first(rating))

# Print
kable(df_attitude[1:10,]) %>% kable_styling_fc()
```

Third, we now generate ratios of values with respect to the first year, within each country.

```
# Sort and get list of countries
ar_years_sorted <- df_attitude %>%
  ungroup() %>% distinct(year) %>% arrange(year) %>%
  pull(year)
st_ratio_var <- paste0('ratings_ratio_vs_year', ar_years_sorted[1])

# Generate ratio over the first location
df_attitude <- df_attitude %>%
  arrange(country, year) %>% group_by(country) %>%
  mutate(!!sym(st_ratio_var) := rating/first(rating))

# Print
```

year	country	rating	ratings_ratio_vs_countrycanada	ratings_ratio_vs_year1
1	canada	51	1.0000000	1.000000
2	canada	64	1.0000000	1.254902
3	canada	70	1.0000000	1.372549
1	uk	39	0.7647059	1.000000
2	uk	54	0.8437500	1.384615
3	uk	69	0.9857143	1.769231
1	usa	43	0.8431373	1.000000
2	usa	63	0.9843750	1.465116
3	usa	71	1.0142857	1.651163

```
# Within each country, we show the first 3 years
kable(df_attitude %>%
  group_by(country) %>%
  slice_min(order_by = year, n = 3)
  ) %>% kable_styling_fc()
```

4.4 Join and Merge Files Together by Keys

4.4.1 Mesh Join

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

4.4.1.1 Expand Multiple Rows with the Same Expansion

File A is at `ID x Week x DayOfWeek` level, file B is at `ID x DayOfWeek x Product`. Product is the product ID bought, could also store other info on product as additional variables. We want to create a file that is at `ID x Week x DayOfweek x Product` level.

The idea is that products bought made on Monday by household 1, for example are always the same, and file A records a "shopping-record", which week, which day each household went shopping.

We do not store in file A what was bought because a particular household always buys the same thing on the same day of the week. We store data in A and B separately to save space since all products by the same household on the same day of week is always identical.

However, we need to join the two files together compute how many units of each product all households bought throughout some timeframe.

Step 1, construct File A, by fully messing ID, Week and Day of Week. In the simulated file below, household 1 shopped 3 times, twice on 3rd day of week, once on 2nd day of week, across two weeks. Household 2 shopped once, on the 3rd day of week.

```
# Mesh
ar_st_varnames <- c('hh','week','dayofweek')
ar_it_ids <- c(1,2)
ar_it_weeks <- c(1,2)
ar_it_daysofweek <- c(1,2,3)
df_idwkday_mesh <- tidyr::expand_grid(
  ar_it_ids, ar_it_weeks, ar_it_daysofweek) %>%
  rename_all(~c(ar_st_varnames))

# Randomly drop a subset of rows
# Different subset of ID and Week for each DayOfWeek.
it_M <- 4
set.seed(456)
df_idwkday_mesh <- df_idwkday_mesh[sample(dim(df_idwkday_mesh)[1], it_M, replace=FALSE),] %>%
  arrange(!!!syms(ar_st_varnames))
```

File A (ID x Week x DayOfWeek)

hh	week	dayofweek
1	1	3
1	2	2
1	2	3
2	2	3

File B (ID x DayOfWeek x Product)

hh	dayofweek	product
1	1	12
1	2	14
1	3	10
1	3	13
1	3	14
2	1	12
2	1	13
2	2	11

```
# Display
st_caption <- "File A (ID x Week x DayOfWeek)"
kable(df_idwkday_mesh, caption=st_caption) %>% kable_styling_fc()
```

Step 2, construct File B. We have shopping list for the 1st household on shopping from 1st, 2nd, and 3rd days of a week. We have a shopping list for 2nd household only for shopping on the 1st and 2nd day.

```
# Generate day of week specific product file
ar_st_varnames <- c('hh', 'dayofweek', 'product')
ar_it_product <- c(10,11,12,13,14)
df_dayproduct_mesh <- tidyr::expand_grid(
  ar_it_ids, ar_it_daysofweek, ar_it_product) %>%
  rename_all(~c(ar_st_varnames))

# Make each day product list not identical
it_M <- 8
set.seed(123)
df_dayproduct_mesh <- df_dayproduct_mesh[sample(dim(df_dayproduct_mesh)[1], it_M, replace=FALSE),] %
  arrange(!!!syms(ar_st_varnames))

# Display
st_caption <- "File B (ID x DayOfWeek x Product)"
kable(df_dayproduct_mesh, caption=st_caption) %>% kable_styling_fc()
```

Step 3. we combine files A and B together via dplyr::left_join.

Given the sample files we have constructed we have:

- multiple items in shopping list for household 1 on day 3
- no shopping list for household 2 on day 3
- shopping list available on days that do not appear on shopping days tracking list

When we left_join, we do not include in combined file shopping list from days for households not in the tracking list. Note that from the output below, we achieved several things:

- the day 3 shopping list for household 1 is merged in twice, to household's trips on day 3 in both week 1 and 2, rows expanded because 3 items bought on each day
- the day 2 shopping list for household 1 is merged in once, there are no row-expansion, since there was one item bought on this shopping list

File C, left-join (ID x Week x DayOfweek x Product)

hh	week	dayofweek	product
1	1	3	10
1	1	3	13
1	1	3	14
1	2	2	14
1	2	3	10
1	2	3	13
1	2	3	14
2	2	3	NA

File C, full-join (ID x Week x DayOfweek x Product)

hh	week	dayofweek	product
1	1	3	10
1	1	3	13
1	1	3	14
1	2	2	14
1	2	3	10
1	2	3	13
1	2	3	14
2	2	3	NA
1	NA	1	12
2	NA	1	12
2	NA	1	13
2	NA	2	11

- the day 3 shopping list for household 2 is not merged in, since the shopping list does not exist, but the row remains.

```
# left join
df_left_join <- df_idwkday_mesh %>%
  left_join(df_dayproduct_mesh,
  by= c('hh'='hh', 'dayofweek'='dayofweek'))
# Display left-join
st_caption <- "File C, left-join (ID x Week x DayOfweek x Product)"
kable(df_left_join, caption=st_caption) %>% kable_styling_fc()
```

Step 4, now, we also try dplyr::full_join. Note that the full-join result is not what we want, it added shopping list by household to the file, but these shopping lists were un-realized, since the households did not shop in any week on those days. So our desired result is achieved by dplyr::left_join.

```
# full join
df_full_join <- df_idwkday_mesh %>%
  full_join(df_dayproduct_mesh,
  by= c('hh'='hh', 'dayofweek'='dayofweek'))
# Display full-join
st_caption <- "File C, full-join (ID x Week x DayOfweek x Product)"
kable(df_full_join, caption=st_caption) %>% kable_styling_fc()
```

Chapter 5

Linear Regression

5.1 Linear and Polynomial Fitting

5.1.1 Fit Curves Through Points

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

5.1.1.1 Polynomial Fit with N Sets of Points

There are three points defined by their x and y coordinates. We draw these points randomly, and we find a curve that fits through them. The fit is exact. If there are more than three sets of points, we will not be able to fit exactly. In the illustration before, first, we draw 3 sets of x and y points, then we draw 4 and 5 sets, and we compare the prediction results.

Note that we are not generating data from a particular set of quadratic (polynomial) parameters, we are just drawing random values. When we draw exactly three pairs of random x and y values, we can find some polynomial that provides an exact fit through the three points in 2d space.

We define the function here.

```
#' Test polynomial fit to random draw x and y points
#'
#' @description Draw random sets of x and y points, fit a polynomial curve through
#' and compare predictions of y and actual y values.
#'
#' @param it_xy_pairs The number of x and y pair points
#' @param it_seed The random seed for drawing values
#' @param it_poly_fit An integer value of the order of polynomial
#' @param fl_mean The mean of the the random normal draw
#' @param fl_sd The standard deviation of the random normal draw
#' @returns
#' \itemize{
#'   \item rs_lm_poly - polynomial fit estimation results
#'   \item df_data_predict - N by 3 where N = \code{it_xy_pairs} and
#' columns are x, y, y-predict, and residual
#'   \item td_table_return - Display version of df_data_predict with title
#' }
#' @import stats, tibble, dplyr
#' @author Fan Wang, \url{http://fanwangecon.github.io}
ffi_lm_quad_fit <- function(it_xy_pairs = 3, it_seed = 123,
                            it_poly_fit = 2, fl_mean = 1, fl_sd = 1,
                            verbose = FALSE) {

  # 1. Generate three pairs of random numbers
```

Quadratic Fit of 3 Sets of Random (X,Y) Points

x	y	ar_y_predict	res
0.4395244	1.070508	1.070508	0
0.7698225	1.129288	1.129288	0
2.5587083	2.715065	2.715065	0

```
  set.seed(it_seed)
  mt_rnorm <- matrix(
    rnorm(it_xy_pairs * 2, mean = fl_mean, sd = fl_sd),
    nrow = it_xy_pairs, ncol = 2
  )
  colnames(mt_rnorm) <- c("x", "y")
  rownames(mt_rnorm) <- paste0("p", seq(1, it_xy_pairs))
  df_rnorm <- as_tibble(mt_rnorm)

  # 2. Quadratic fit using ORTHOGONAL POLYNOMIAL
  # For predictions, lm(y ~ x + I(x^2)) and lm(y ~ poly(x, 2)) are the same,
  # but they have different parameters because x is transformed by poly().
  rs_lm_quad <- stats::lm(y ~ poly(x, it_poly_fit), data = df_rnorm)
  if (verbose) print(stats::summary.lm(rs_lm_quad))

  # 3. Fit prediction
  ar_y_predict <- stats::predict(rs_lm_quad)
  df_data_predict <- cbind(df_rnorm, ar_y_predict) %>%
    mutate(res = ar_y_predict - y)
  if (verbose) print(df_data_predict)

  # 4. show values
  st_poly_order <- "Quadratic"
  if (it_poly_fit != 2) {
    st_poly_order <- paste0(it_poly_fit, "th order")
  }
  td_table_return <- kable(df_data_predict,
    caption = paste0(
      st_poly_order, " Fit of ", it_xy_pairs, " Sets of Random (X,Y) Points"
    )
  ) %>%
    kable_styling_fc()

  return(list(
    rs_lm_quad = rs_lm_quad,
    df_data_predict = df_data_predict,
    td_table_return = td_table_return
  ))
}
```

In the first example below, we simulate 3 set of points and estimate quadratic exact fit.

```
ls_ffi_lm_quad_fit <-
  ffi_lm_quad_fit(
    it_xy_pairs = 3, it_seed = 123,
    it_poly_fit = 2, fl_mean = 1, fl_sd = 1
  )
ls_ffi_lm_quad_fit$td_table_return
```

In the second example below, we simulate 4 set of points and estimate a quadratic non-exact fit.

Quadratic Fit of 4 Sets of Random (X,Y) Points

x	y	ar_y_predict	res
0.2150918	0.9324684	0.9373687	0.0049003
0.7204856	0.3664796	1.5207575	1.1542779
0.8385421	0.0722760	-0.0080226	-0.0802987
0.7094034	2.7107710	1.6318915	-1.0788795

3th order Fit of 4 Sets of Random (X,Y) Points

x	y	ar_y_predict	res
0.2150918	0.9324684	0.9324684	0
0.7204856	0.3664796	0.3664796	0
0.8385421	0.0722760	0.0722760	0
0.7094034	2.7107710	2.7107710	0

```
ls_ffi_lm_quad_fit <-
  ffi_lm_quad_fit(
    it_xy_pairs = 4, it_seed = 345,
    it_poly_fit = 2, fl_mean = 1, fl_sd = 1
  )
ls_ffi_lm_quad_fit$td_table_return
```

In the third example below, we simulate the same 4 sets of points as in the prior example, but now use a cubic polynomial to fit the data exactly.

```
ls_ffi_lm_cubic_fit <-
  ffi_lm_quad_fit(
    it_xy_pairs = 4, it_seed = 345,
    it_poly_fit = 3, fl_mean = 1, fl_sd = 1
  )
ls_ffi_lm_cubic_fit$td_table_return
```

5.1.2 Polynomial Time Series

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

5.1.2.1 Analytical Solution for Time-Series Polynomial Coefficients

This file is developed in support of Bhalotra, Fernandez, and Wang (2021), where we use polynomials to model share parameter changes over time in the context of a nested-CES problem.

5.1.2.1.1 Mth Order Polynomial and its Derivative

We have a polynomial of Mth order:

$$y\left(t\right)=a_{0}+a_{1}\cdot t+a_{2}\cdot t^{2}+...+a_{M}\cdot t^{M}$$

Taking derivative of of y with respect to t, we have:

$$\begin{aligned}
\frac{d^{M-0}y}{dt^{M-0}}&=a_{M-0}\cdot\frac{(M-0)!}{0!}\cdot t^{-0+0}\\
\frac{d^{M-1}y}{dt^{M-1}}&=a_{M-0}\cdot\frac{(M-0)!}{1!}\cdot t^{-0+1}+a_{M-1}\cdot\frac{(M-1)!}{0!}\cdot t^{-1+1}\\
\frac{d^{M-2}y}{dt^{M-2}}&=a_{M-0}\cdot\frac{(M-0)!}{2!}\cdot t^{-0+2}+a_{M-1}\cdot\frac{(M-1)!}{1!}\cdot t^{-1+2}+a_{M-2}\cdot\frac{(M-2)!}{0!}\cdot t^{-2+2}\\
\frac{d^{M-3}y}{dt^{M-3}}&=a_{M-0}\cdot\frac{(M-0)!}{3!}\cdot t^{-0+3}+a_{M-1}\cdot\frac{(M-1)!}{2!}\cdot t^{-1+3}+a_{M-2}\cdot\frac{(M-2)!}{1!}\cdot t^{-2+3}+a_{M-3}\cdot\frac{(M-3)!}{0!}\cdot t^{-3+3}
\end{aligned}$$

Given the structure above, we have the following formulation for polynomial derivative, where we have a polynomial of M^{th} order and we are interested in the N^{th} derivative:

$$\frac{d^N y(M)}{dt^N} = \sum_{i=0}^{M-N} \left(a_{M-i} \cdot \frac{(M-i)!}{(M-N-i)!} \cdot t^{M-N-i} \right)$$

Note that the M^{th} derivative of a M^{th} order polynomial is equal to $(a_M \cdot M!)$

5.1.2.1.2 Repeatedly Taking Differences of Differences Suppose the data generating process is a M^{th} order polynomial, then differencing of time-series observables can be used to identify polynomial coefficients.

In the simplest case of a 1st order polynomial with $Y_t = a_0 + a_1 \cdot t$, given 2 periods of data, $t = 0, t = 1$, we have $a_1 = Y_2 - Y_1$, and $a_0 = Y_0$. The rate of change in Y over t captures coefficient a_1.

For a second order polynomial, while the first derivative is time-varying, the second derivative, acceleration, is invariant over t. Similarly, for any M^{th} order polynomial, the M^{th} derivative is time-invariant.

Because of this time-invariance of the M^{th} derivative, the differencing idea can be used to identify a_M, the coefficient for the highest polynomial term for the M^{th} order polynomial.

Now we difference observable y_t overtime. We difference the differences, difference the differences of the differences, and difference the differences of the differences of the differences, etc. It turns out that the difference is a summation over all observations $\{y_t\}_{t=1}^{T}$, with the number of times each y_t term appearing following Pascal's Triangle.

Specifically:

- 1st difference is $y_1 - y_0$
- 2nd difference is $(y_2 - y_1) - (y_1 - y_0) = y_2 - 2y_1 + y_0$
- 3rd difference is $((y_3 - y_2) - (y_2 - y_1)) - ((y_2 - y_1) - (y_1 - y_0)) = y_3 - 3y_2 + 3y_1 - y_0$
- 4th difference is $(((y_4 - y_3) - (y_3 - y_2)) - ((y_3 - y_2) - (y_2 - y_1))) - (((y_3 - y_2) - (y_2 - y_1)) - ((y_2 - y_1) - (y_1 - y_0))) = y_4 - 4y_3 + 6y_2 - 4y_1 + y_0$
- 5th difference is $((((y_5-y_4)-(y_4-y_3))-((y_4-y_3)-(y_3-y_2)))-(((y_4-y_3)-(y_3-y_2))-((y_3-y_2)-(y_2-y_1))))-((((y_4-y_3)-(y_3-y_2))-((y_3-y_2)-(y_2-y_1)))-(((y_3-y_2)-(y_2-y_1))-((y_2-y_1)-(y_1-y_0)))) = y_5 - 5y_4 + 10y_3 - 10y_2 + 5y_1 - y_0$

Note that the pattern has alternating signs, and the coefficients are binomial. We have the following formula:

$$\Delta^M = \sum_{i=0}^{M} \left((-1)^i \cdot \frac{M!}{(M-i)!i!} \right) \cdot y_{(M-i)}$$

When there are T periods of data, and we are interested in the $T-1$ difference, the differencing formula is unique. However, for less than $T-1$ difference, we can use alternative consecutive data segments. Specifically, given T periods of data from $t = 1$ to $t = T$, we have the notation Δ_τ^M where τ is the starting time. We have, for $M \leq T - 1$:

$$\Delta_\tau^M \left(\{y_t\}_{t=1}^{T} \right) = \sum_{i=0}^{M} \left((-1)^i \cdot \frac{M!}{(M-i)!i!} \right) \cdot y_{(\tau+(M-i))} \text{ for } 1 \leq \tau \leq T-M$$

5.1.2.1.3 Solutions for Polynomial Coefficients with Differences of Differences Intuitively, for a M^{th} order polynomial, the coefficient on the highest polynomial term is a function of the $(M-1)^{th}$ difference. Coefficients of lower polynomial terms, $m < M$, are function of the $(m-1)^{th}$ difference along with higher order polynomial coefficients already computed.

Formally, we have, for a M^{th} order polynomial, a vector of $\{a_m\}_{m=0}^{M}$ $M+1$ polynomial coefficients. For formula for the coefficient for the largest polynomial is:

$$a_M = \sum_{i=0}^{M} \left((-1)^i \left((M-i)!i! \right)^{-1} \right) y_{(M-i)}$$

Given this, we have also, given T periods of data from $t = 1$ to $t = T$:

$$a_{M-1} = \sum_{i=0}^{M-1} \left((-1)^i \left((M-1-i)!i! \right)^{-1} \right) \cdot \left(y_{(\tau+(M-1-i))} - a_M \cdot t^M \right) \text{ for } 1 \leq \tau \leq T-M-1$$

Using one formula, given a_{m+1}, we have:

$$a_m = \sum_{i=0}^{m} \left((-1)^i \left((m-i)!i! \right)^{-1} \right) \cdot \left(y_{(\tau+(m-i))} - \sum_{j=0}^{M-m-1} a_{M-j} \cdot t^{M-j} \right) \text{ for } 1 \leq \tau \leq T-m$$

5.1.2.1.4 Identifying Polynomial Coefficients with Differences for Third Order Polynomial

To illustrate, we test the formulas with a 3rd order polynomial, and derive some 3rd-order specific formulas.

For data from a 3rd order polynomial data generating process, we can use the 3rd difference to identify the coefficient in front of x^3. With this, we can iteratively to lower polynomials and identify all relevant coefficients.

Specifically, using equations from the two sections above, we have:

$$\begin{aligned} \frac{d^3 y(3)}{dt^3} &= y_3 - 3y_2 + 3y_1 - y_0 \\ \frac{d^3 y(3)}{dt^3} &= 3! \cdot a_3 \end{aligned}$$

Combining the two equations we have, that a_3 is the 3rd difference divided by 6:

$$\begin{aligned} 3! \cdot a_3 &= y_3 - 3y_2 + 3y_1 - y_0 \\ a_3 &= \frac{y_3 - 3y_2 + 3y_1 - y_0}{3 \cdot 2} \end{aligned}$$

For the linear and cubic terms, we have:

$$\begin{aligned} \frac{d^2 y(M=3)}{dt^2} &= 3 \cdot a_2 + 6 \cdot a_3 \cdot t \\ \frac{d^1 y(M=3)}{dt^1} &= a_1 + 2 \cdot a_2 + 3 \cdot a_3 \cdot t^2 \end{aligned}$$

Note that the 3rd derivative of a 3rd order polynomial is a constant, but the 2nd derivative of a 3rd order polynomial is not. This means that to use the second difference to identify a_2 parameter, we first have to difference out from y_t the impact of the 3rd polynomial term, which we can because we know a_3 now.

Differencing out the 3rd term, we have now the 2nd derivative of a 2nd order polynomial:

$$\frac{d^2 \left(y(M=3) - a_3 \cdot t^3 \right)}{dt^2} = \frac{d^2 \left(\widehat{y}(M=2) \right)}{dt^2} ,$$

where $\widehat{y}(M,2) = y(M) - a_3 \cdot t^3$.

So this means we have:

$$
\begin{aligned}
\frac{d^2\widehat{y}(3,2)}{dt^2} &= (1\cdot y_2 - 2\cdot y_1 + 1\cdot y_0) - a_3\cdot(1\cdot 2^3 - 2\cdot 1^3 + 1\cdot 0^3)\\
&= (1\cdot y_2 - 2\cdot y_1 + 1\cdot y_0) - a_3\cdot(2^3 - 2)\\
\frac{d^2\widehat{y}(3,2)}{dt^2} &= 2!\cdot a_2
\end{aligned}
$$

Given the value for a_3, we have:

$$
\begin{aligned}
2!\cdot a_2 &= (1\cdot y_2 - 2\cdot y_1 + 1\cdot y_0) - a_3\cdot 6\\
2!\cdot a_2 &= (1\cdot y_2 - 2\cdot y_1 + 1\cdot y_0) - \frac{y_3 - 3y_2 + 3y_1 - y_0}{3\cdot 2}\cdot(2^3-2)\\
2!\cdot a_2 &= (y_2 - 2y_1 + y_0) - (y_3 - 3y_2 + 3y_1 - y_0)\\
2!\cdot a_2 &= -y_3 + 4y_2 - 5y_1 + 2y_0\\
a_2 &= \frac{-y_3 + 4y_2 - 5y_1 + 2y_0}{2}
\end{aligned}
$$

Following the same strategy, we can also find a_1. Let $\widehat{y}\,(M,1) = y\,(M) - a_3\cdot t^3 - a_2\cdot t^2$

$$
\begin{aligned}
\frac{d^2\widehat{y}(3,1)}{dt^2} &= (y_1 - y_0) - a_3\cdot(1^3 - 0^3) - a_2\cdot(1^2 - 0^2)\\
&= (y_1 - y_0) - a_3 - a_2\\
\frac{d^2\widehat{y}(3,1)}{dt^2} &= 1!\cdot a_1
\end{aligned}
$$

Hence:

$$
\begin{aligned}
a_1 &= (y_1 - y_0) - a_3\cdot(1^3 - 0^3) - a_2\cdot(1^2 - 0^2)\\
&= (y_1 - y_0) - a_3 - a_2\\
&= (y_1 - y_0) - \frac{y_3 - 3y_2 + 3y_1 - y_0}{3\cdot 2} - \frac{-y_3 + 4y_2 - 5y_1 + 2y_0}{2}\\
&= \frac{6y_1 - 6y_0}{6} - \frac{y_3 - 3y_2 + 3y_1 - y_0}{6} - \frac{-3y_3 + 12y_2 - 15y_1 + 6y_0}{6}\\
&= \frac{6y_1 - 6y_0 - y_3 + 3y_2 - 3y_1 + y_0 + 3y_3 - 12y_2 + 15y_1 - 6y_0}{6}\\
&= \frac{2y_3 - 9y_2 + 18y_1 - 11y_0}{6}
\end{aligned}
$$

Finally, we know that $a_0 = y_0$. We have now analytical expressions for each of the 4 polynomial coefficients for a 3rd order polynomial. Given data from the data-generating process, these would back out the underlying parameters of the data generating process using data from four periods at $t = 0, 1, 2, 3$.

5.1.2.1.5 Third Order Polynomial Simulation and Solving for Parameters Now we generated a time-series of values and solve back for the underlying polynomial coefficients.

```r
# polynomial coefficients
set.seed(123)
ar_coef_poly <- rnorm(4)
# time right hand side matrix
ar_t <- 0:3
ar_power <- 0:3
mt_t_data <- do.call(rbind, lapply(ar_power, function(power) {
  ar_t^power
}))
# Final matrix, each row is an observation, or time.
mt_t_data <- t(mt_t_data)
```

C1=Y, each row is time, t=0, incremental by 1, each column a polynomial term from 0th to higher.

-0.5604756	1	0	0	0
0.8385636	1	1	1	1
5.7780698	1	2	4	8
14.6810933	1	3	9	27

Solving for polynomial coefficients.

Coefficient Counter	Polynomial Terms	Solved Coefficient Given Y	Actual DGP Coefficient
1	Constant	-0.560475646552213	-0.560475646552213
2	Linear	-0.230177489483281	-0.23017748948328
3	Quadratic	1.55870831414913	1.55870831414912
4	Cubic	0.0705083914245757	0.070508391424576

```
# General model prediction
ar_y <- mt_t_data %*% matrix(ar_coef_poly, ncol = 1, nrow = 4)
# Prediction and Input time matrix
mt_all_data <- cbind(ar_y, mt_t_data)
st_cap <- paste0(
  "C1=Y, each row is time, t=0, incremental by 1, ",
  "each column a polynomial term from 0th to higher."
)
kable(mt_all_data, caption = st_cap) %>% kable_styling_fc()
```

Backing out coefficients using the formulas for 3rd order polynomial derived above, we have:

```
# The constant term
alpha_0 <- ar_y[1]
# The cubic term
alpha_3 <- as.numeric((t(ar_y) %*% c(-1, +3, -3, +1))/(3*2))
# The quadratic term, difference cubic out, alpha_2_1t3 = alpha_2_2t4
ar_y_hat <- ar_y - alpha_3*ar_t^3
alpha_2_1t3 <- as.numeric((t(ar_y_hat[1:3]) %*% c(1, -2, +1))/(2))
alpha_2_2t4 <- as.numeric((t(ar_y_hat[2:4]) %*% c(1, -2, +1))/(2))
alpha_2 <- alpha_2_1t3
# The linear term, difference cubic out and quadratic
ar_y_hat <- ar_y - alpha_3*ar_t^3 - alpha_2*ar_t^2
alpha_1_1t2 <- as.numeric((t(ar_y_hat[1:2]) %*% c(-1, +1))/(1))
alpha_1_2t3 <- as.numeric((t(ar_y_hat[2:3]) %*% c(-1, +1))/(1))
alpha_1_3t4 <- as.numeric((t(ar_y_hat[3:4]) %*% c(-1, +1))/(1))
alpha_1 <- alpha_1_1t2
# Collect results
ar_names <- c("Constant", "Linear", "Quadratic", "Cubic")
ar_alpha_solved <- c(alpha_0, alpha_1, alpha_2, alpha_3)
mt_alpha <- cbind(ar_names, ar_alpha_solved, ar_coef_poly)
# Display
ar_st_varnames <- c('Coefficient Counter', 'Polynomial Terms', 'Solved Coefficient Given Y', 'Actual
tb_alpha <- as_tibble(mt_alpha) %>%
  rowid_to_column(var = "polynomial_term_coef") %>%
  rename_all(~c(ar_st_varnames))
# Display
st_cap = paste0('Solving for polynomial coefficients.')
kable(tb_alpha, caption = st_cap) %>% kable_styling_fc()
```

Note that given that the data is exact output from DGP, and we have the same number of data and parameters, parameters are exactly identified. However, this is only really true for the a_3 parameter, which requires all four periods of data. - For a_2, it is over-identified, we can arrived at it, given a_3, either with difference of differences using data from $t = 1, 2, 3$ or using data from $t = 2, 3, 4$. - For a_1, it is also over-identified, given a_3 and a_2. The difference of $t = 1, 2$, $t = 2, 3$ or $t = 3, 4$ can all identify a_1.

Note also that the solution above can be found by running a linear regression as well. The point of doing the exercise here and showing analytically how layers of differences of differences identify each polynomial coefficient is to show what in the underlying variation of the data is identifying each of the polynomial term.

In effect, all identification is based on the fact that the M^{th} order polynomial's M^{th} derivative is a constant, it is invariant over t. This is the core assumption, or restriction of the otherwise highly flexible polynomial functional form. With this core invariance at the max degree derivative condition, all other parameters are obtained through the simple act of differencing.

5.2 OLS and IV

Back to **Fan**'s R4Econ Homepage **Table of Content**

5.2.1 OLS and IV Regression

> Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

IV regression using AER package. Option to store all results in dataframe row for combining results from other estimations together. Produce Row Statistics.

5.2.1.1 Construct Program

```
# IV regression function
# The code below uses the AER library's regresison function
# All results are stored in a single row as data_frame
# This functoin could work with dplyr do
# var.y is single outcome, vars.x, vars.c and vars.z are vectors of endogenous variables, controls a
regf.iv <- function(var.y, vars.x,
                    vars.c, vars.z, df, transpose=TRUE) {

  # A. Set-Up Equation
  str.vars.x <- paste(vars.x, collapse='+')
  str.vars.c <- paste(vars.c, collapse='+')

  df <- df %>%
    select(one_of(var.y, vars.x, vars.c, vars.z)) %>%
    drop_na() %>% filter_all(all_vars(!is.infinite(.)))

  if (length(vars.z) >= 1) {
    #     library(AER)
    str.vars.z <- paste(vars.z, collapse='+')
    equa.iv <- paste(var.y,
                     paste(paste(str.vars.x, str.vars.c, sep='+'),
                           paste(str.vars.z, str.vars.c, sep='+'),
                           sep='|'),
                     sep='~')
    #     print(equa.iv)

    # B. IV Regression
    ivreg.summ <- summary(ivreg(as.formula(equa.iv), data=df),
                          vcov = sandwich, df = Inf, diagnostics = TRUE)

    # C. Statistics from IV Regression
    #     ivreg.summ$coef
```

```
    #     ivreg.summ$diagnostics

    # D. Combine Regression Results into a Matrix
    df.results <- suppressWarnings(suppressMessages(
      as_tibble(ivreg.summ$coef, rownames='rownames') %>%
        full_join(as_tibble(ivreg.summ$diagnostics, rownames='rownames')) %>%
        full_join(tibble(rownames=c('vars'),
                         var.y=var.y,
                         vars.x=str.vars.x,
                         vars.z=str.vars.z,
                         vars.c=str.vars.c))))
  } else {

    # OLS regression
    equa.ols <- paste(var.y,
                      paste(paste(vars.x, collapse='+'),
                            paste(vars.c, collapse='+'), sep='+'),
                      sep='~')

    lmreg.summ <- summary(lm(as.formula(equa.ols), data=df))

    lm.diagnostics <- as_tibble(
      list(df1=lmreg.summ$df[[1]],
           df2=lmreg.summ$df[[2]],
           df3=lmreg.summ$df[[3]],
           sigma=lmreg.summ$sigma,
           r.squared=lmreg.summ$r.squared,
           adj.r.squared=lmreg.summ$adj.r.squared)) %>%
      gather(variable, value) %>%
      rename(rownames = variable) %>%
      rename(v = value)

    df.results <- suppressWarnings(suppressMessages(
      as_tibble(lmreg.summ$coef, rownames='rownames') %>%
        full_join(lm.diagnostics) %>%
        full_join(tibble(rownames=c('vars'),
                         var.y=var.y,
                         vars.x=str.vars.x,
                         vars.c=str.vars.c))))
  }

  # E. Flatten Matrix, All IV results as a single tibble
  # row to be combined with other IV results
  df.row.results <- df.results %>%
    gather(variable, value, -rownames) %>%
    drop_na() %>%
    unite(esti.val, rownames, variable) %>%
    mutate(esti.val = gsub(' ', '', esti.val))

  if (transpose) {
    df.row.results <- df.row.results %>% spread(esti.val, value)
  }

  # F. Return
  return(data.frame(df.row.results))
}
```

esti.val	value
(Intercept)_Estimate	52.1186286658655
prot_Estimate	0.37447238635789
sexMale_Estimate	0.611043720578337
hgt0_Estimate	0.148513781160842
wgt0_Estimate	0.00150560230505629
(Intercept)_Std.Error	1.57770483608693
prot_Std.Error	0.004181211911338
sexMale_Std.Error	0.118396259120663
hgt0_Std.Error	0.0393807494783184
wgt0_Std.Error	0.000187123663624396
(Intercept)_tvalue	33.0344608660336
prot_tvalue	89.5607288744324
sexMale_tvalue	5.16100529794268
hgt0_tvalue	3.77122790013451
wgt0_tvalue	8.04602836377986
(Intercept)_Pr(>|t|)	9.92126150965951e-233
prot_Pr(>|t|)	0
sexMale_Pr(>|t|)	2.48105505495376e-07
hgt0_Pr(>|t|)	0.000162939618371172
wgt0_Pr(>|t|)	9.05257561534482e-16
df1_v	5
df2_v	18958
df3_v	5
sigma_v	8.06197784622979
r.squared_v	0.319078711001326
adj.r.squared_v	0.318935041565942
vars_var.y	hgt
vars_vars.x	prot
vars_vars.c	sex+hgt0+wgt0

5.2.1.2 Program Testing

Load Data

```
# Library
library(tidyverse)
library(AER)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')
```

```
# One Instruments
var.y <- c('hgt')
vars.x <- c('prot')
vars.z <- NULL
vars.c <- c('sex', 'hgt0', 'wgt0')
# Regression
regf.iv(var.y, vars.x, vars.c, vars.z, df, transpose=FALSE) %>%
  kable() %>%
  kable_styling_fc()
```

5.2.1.2.1 Example No Instrument, OLS

esti.val	value
(Intercept)_Estimate	43.4301969117894
prot_Estimate	0.130833343849431
sexMale_Estimate	0.868121847262594
hgt0_Estimate	0.412093881816612
wgt0_Estimate	0.000858630042618959
(Intercept)_Std.Error	1.82489550970971
prot_Std.Error	0.0192036220809207
sexMale_Std.Error	0.133730167005418
hgt0_Std.Error	0.0459431912925973
wgt0_Std.Error	0.000226910577025037
(Intercept)_zvalue	23.7987307660689
prot_zvalue	6.81295139521715
sexMale_zvalue	6.49159323361512
hgt0_zvalue	8.96963990141912
wgt0_zvalue	3.78400184723085
(Intercept)_Pr(>|z|)	3.44237661591474e-125
prot_Pr(>|z|)	9.56164541652958e-12
sexMale_Pr(>|z|)	8.49333228164569e-11
hgt0_Pr(>|z|)	2.97485394504032e-19
wgt0_Pr(>|z|)	0.000154326676599558
Weakinstruments_df1	1
Wu-Hausman_df1	1
Sargan_df1	0
Weakinstruments_df2	16394
Wu-Hausman_df2	16393
Weakinstruments_statistic	935.817456612075
Wu-Hausman_statistic	123.595856606734
Weakinstruments_p-value	6.3971492917806e-200
Wu-Hausman_p-value	1.30703637796418e-28
vars_var.y	hgt
vars_vars.x	prot
vars_vars.z	momEdu
vars_vars.c	sex+hgt0+wgt0

```
# One Instrucments
var.y <- c('hgt')
vars.x <- c('prot')
vars.z <- c('momEdu')
vars.c <- c('sex', 'hgt0', 'wgt0')
# Regression
regf.iv(var.y, vars.x, vars.c, vars.z, df, transpose=FALSE) %>%
  kable() %>%
  kable_styling_fc()
```

5.2.1.2.2 Example 1 Insturment

```
# Multiple Instrucments
var.y <- c('hgt')
vars.x <- c('prot')
vars.z <- c('momEdu', 'wealthIdx', 'p.A.prot', 'p.A.nProt')
vars.c <- c('sex', 'hgt0', 'wgt0')
# Regression
regf.iv(var.y, vars.x, vars.c, vars.z, df, transpose=FALSE) %>%
```

esti.val	value
(Intercept)__Estimate	42.243761355532
prot__Estimate	0.266999451947032
sexMale__Estimate	0.695548488813011
hgt0__Estimate	0.424954881262903
wgt0__Estimate	0.000486951420329 74
(Intercept)__Std.Error	1.8535668678938
prot__Std.Error	0.0154939347964086
sexMale__Std.Error	0.133157977814372
hgt0__Std.Error	0.0463195803786077
wgt0__Std.Error	0.000224867994873511
(Intercept)__zvalue	22.7905246297013
prot__zvalue	17.2325142357589
sexMale__zvalue	5.22348341593647
hgt0__zvalue	9.17441129192881
wgt0__zvalue	2.16549901022443
(Intercept)__Pr(>|z|)	5.6929407426237e-115
prot__Pr(>|z|)	1.51424021933765e-66
sexMale__Pr(>|z|)	1.75588197501936e-07
hgt0__Pr(>|z|)	4.54048595586446e-20
wgt0__Pr(>|z|)	0.0303494911144483
Weakinstruments__df1	4
Wu-Hausman__df1	1
Sargan__df1	3
Weakinstruments__df2	14914
Wu-Hausman__df2	14916
Weakinstruments__statistic	274.147084958342
Wu-Hausman__statistic	17.7562545747099
Sargan__statistic	463.729664547247
Weakinstruments__p-value	8.6173195623464e-228
Wu-Hausman__p-value	2.52567249124201e-05
Sargan__p-value	3.45452874915773e-100
vars__var.y	hgt
vars__vars.x	prot
vars__vars.z	momEdu+wealthIdx+p.A.prot+p.A.nProt
vars__vars.c	sex+hgt0+wgt0

```
  kable() %>%
  kable_styling_fc()
```

5.2.1.2.3 Example Multiple Instrucments

```
# Multiple Instrucments
var.y <- c('hgt')
vars.x <- c('prot', 'cal')
vars.z <- c('momEdu', 'wealthIdx', 'p.A.prot', 'p.A.nProt')
vars.c <- c('sex', 'hgt0', 'wgt0')
# Regression
regf.iv(var.y, vars.x, vars.c, vars.z, df, transpose=FALSE) %>%
  kable() %>%
  kable_styling_fc()
```

5.2.1.2.4 Example Multiple Endogenous Variables

<table>
<tr><th>esti.val</th><th>value</th></tr>
<tr><td>(Intercept)_Estimate</td><td>44.0243196254372</td></tr>
<tr><td>prot_Estimate</td><td>-1.40256232471058</td></tr>
<tr><td>cal_Estimate</td><td>0.0651048957501503</td></tr>
<tr><td>sexMale_Estimate</td><td>0.120832787571903</td></tr>
<tr><td>hgt0_Estimate</td><td>0.286525437984394</td></tr>
<tr><td>wgt0_Estimate</td><td>0.000850481389651284</td></tr>
<tr><td>(Intercept)_Std.Error</td><td>2.75354847245259</td></tr>
<tr><td>prot_Std.Error</td><td>0.198640060272442</td></tr>
<tr><td>cal_Std.Error</td><td>0.00758881298876464</td></tr>
<tr><td>sexMale_Std.Error</td><td>0.209984580636194</td></tr>
<tr><td>hgt0_Std.Error</td><td>0.0707828182891216</td></tr>
<tr><td>wgt0_Std.Error</td><td>0.000337112104445047</td></tr>
<tr><td>(Intercept)_zvalue</td><td>15.9882130515846</td></tr>
<tr><td>prot_zvalue</td><td>-7.06082309271814</td></tr>
<tr><td>cal_zvalue</td><td>8.57906181724851</td></tr>
<tr><td>sexMale_zvalue</td><td>0.575436478268133</td></tr>
<tr><td>hgt0_zvalue</td><td>4.04795181810993</td></tr>
<tr><td>wgt0_zvalue</td><td>2.52284441417891</td></tr>
<tr><td>(Intercept)_Pr(>|z|)</td><td>1.54396598289425e-57</td></tr>
<tr><td>prot_Pr(>|z|)</td><td>1.65519210798224e-12</td></tr>
<tr><td>cal_Pr(>|z|)</td><td>9.56500647777971e-18</td></tr>
<tr><td>sexMale_Pr(>|z|)</td><td>0.564996139463126</td></tr>
<tr><td>hgt0_Pr(>|z|)</td><td>5.16677787150118e-05</td></tr>
<tr><td>wgt0_Pr(>|z|)</td><td>0.011640989283946</td></tr>
<tr><td>Weakinstruments(prot)_df1</td><td>4</td></tr>
<tr><td>Weakinstruments(cal)_df1</td><td>4</td></tr>
<tr><td>Wu-Hausman_df1</td><td>2</td></tr>
<tr><td>Sargan_df1</td><td>2</td></tr>
<tr><td>Weakinstruments(prot)_df2</td><td>14914</td></tr>
<tr><td>Weakinstruments(cal)_df2</td><td>14914</td></tr>
<tr><td>Wu-Hausman_df2</td><td>14914</td></tr>
<tr><td>Weakinstruments(prot)_statistic</td><td>274.147084958342</td></tr>
<tr><td>Weakinstruments(cal)_statistic</td><td>315.036848606229</td></tr>
<tr><td>Wu-Hausman_statistic</td><td>94.7020085425631</td></tr>
<tr><td>Sargan_statistic</td><td>122.0819796289</td></tr>
<tr><td>Weakinstruments(prot)_p-value</td><td>8.6173195623464e-228</td></tr>
<tr><td>Weakinstruments(cal)_p-value</td><td>1.18918641221312e-260</td></tr>
<tr><td>Wu-Hausman_p-value</td><td>1.35024050402095e-41</td></tr>
<tr><td>Sargan_p-value</td><td>3.09196773720141e-27</td></tr>
<tr><td>vars_var.y</td><td>hgt</td></tr>
<tr><td>vars_vars.x</td><td>prot+cal</td></tr>
<tr><td>vars_vars.z</td><td>momEdu+wealthIdx+p.A.prot+p.A.nProt</td></tr>
<tr><td>vars_vars.c</td><td>sex+hgt0+wgt0</td></tr>
</table>

5.2.1.2.5 Examples Line by Line

The examples are just to test the code with different types of variables.

```
# Selecting Variables
var.y <- c('hgt')
vars.x <- c('prot', 'cal')
vars.z <- c('momEdu', 'wealthIdx', 'p.A.prot', 'p.A.nProt')
vars.c <- c('sex', 'hgt0', 'wgt0')
```

```
# A. create Equation
str.vars.x <- paste(vars.x, collapse='+')
str.vars.c <- paste(vars.c, collapse='+')
str.vars.z <- paste(vars.z, collapse='+')
print(str.vars.x)
```

```
## [1] "prot+cal"
```

```
print(str.vars.c)
```

```
## [1] "sex+hgt0+wgt0"
```

```
print(str.vars.z)
```

```
## [1] "momEdu+wealthIdx+p.A.prot+p.A.nProt"
```

```
equa.iv <- paste(var.y,
                 paste(paste(str.vars.x, str.vars.c, sep='+'),
                       paste(str.vars.z, str.vars.c, sep='+'),
                       sep='|'),
                 sep='~')
print(equa.iv)
```

```
## [1] "hgt~prot+cal+sex+hgt0+wgt0|momEdu+wealthIdx+p.A.prot+p.A.nProt+sex+hgt0+wgt0"
```

```
# B. regression
res.ivreg <- ivreg(as.formula(equa.iv), data=df)
coef(res.ivreg)
```

```
##   (Intercept)          prot           cal       sexMale          hgt0          wgt0
## 44.0243196254 -1.4025623247  0.0651048958  0.1208327876  0.2865254380  0.0008504814
```

```
# C. Regression Summary
ivreg.summ <- summary(res.ivreg, vcov = sandwich, df = Inf, diagnostics = TRUE)

ivreg.summ$coef
```

```
##                  Estimate   Std. Error    z value     Pr(>|z|)
## (Intercept) 44.0243196254 2.7535484725 15.9882131 1.543966e-57
## prot        -1.4025623247 0.1986400603 -7.0608231 1.655192e-12
## cal          0.0651048958 0.0075888130  8.5790618 9.565006e-18
## sexMale      0.1208327876 0.2099845806  0.5754365 5.649961e-01
## hgt0         0.2865254380 0.0707828183  4.0479518 5.166778e-05
## wgt0         0.0008504814 0.0003371121  2.5228444 1.164099e-02
## attr(,"df")
## [1] 0
## attr(,"nobs")
## [1] 14922
```

```
ivreg.summ$diagnostics
```

```
##                          df1   df2 statistic       p-value
## Weak instruments (prot)    4 14914 274.14708 8.617320e-228
## Weak instruments (cal)     4 14914 315.03685 1.189186e-260
## Wu-Hausman                 2 14914  94.70201  1.350241e-41
```

```
## Sargan                         2    NA 122.08198  3.091968e-27
```

```
# D. Combine Regression Results into a Matrix
df.results <- suppressMessages(as_tibble(ivreg.summ$coef, rownames='rownames') %>%
    full_join(as_tibble(ivreg.summ$diagnostics, rownames='rownames')) %>%
    full_join(tibble(rownames=c('vars'),
                     var.y=var.y,
                     vars.x=str.vars.x,
                     vars.z=str.vars.z,
                     vars.c=str.vars.c)))
# E. Flatten Matrix, All IV results as a single tibble row to be combined with other IV results
df.row.results <- df.results %>%
    gather(variable, value, -rownames) %>%
    drop_na() %>%
    unite(esti.val, rownames, variable) %>%
    mutate(esti.val = gsub(' ', '', esti.val))
```

```
# F. Results as Single Colum
# df.row.results
```

```
# G. Results as Single Row
# df.row.results
```

```
# t(df.row.results %>% spread(esti.val, value)) %>%
#   kable() %>%
#   kable_styling_fc_wide()
```

5.2.2 IV Loop over RHS

> Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Regression with a Variety of Outcome Variables and Right Hand Side Variables. There are M outcome variables, and there are N alternative right hand side variables. Regress each M outcome variable and each N alternative right hand side variable, with some common sets of controls and perhaps shared instruments. The output file is a M by N matrix of coefficients, with proper variable names and row names. The matrix stores coefficients for this key endogenous variable.

- Dependency: *R4Econ/linreg/ivreg/ivregdfrow.R*

5.2.2.1 Construct Program

The program relies on double lapply. lapply is used for convenience, not speed.

```
ff_reg_mbyn <- function(list.vars.y, list.vars.x,
                        vars.c, vars.z, df,
                        return_all = FALSE,
                        stats_ends = 'value', time = FALSE) {

  # regf.iv() function is from C:\Users\fan\R4Econ\linreg\ivreg\ivregdfrow.R
  if (time) {
    start_time <- Sys.time()
  }

  if (return_all) {
    df.reg.out.all <-
      bind_rows(lapply(list.vars.x,
                       function(x) (
                         bind_rows(
                           lapply(list.vars.y, regf.iv,
                                  vars.x=x, vars.c=vars.c, vars.z=vars.z, df=df))
```

```r
                        )))

  } else {
    df.reg.out.all <-
      (lapply(list.vars.x,
              function(x) (
                bind_rows(
                  lapply(list.vars.y, regf.iv,
                         vars.x=x, vars.c=vars.c, vars.z=vars.z, df=df)) %>%
                  select(vars_var.y, starts_with(x)) %>%
                  select(vars_var.y, ends_with(stats_ends))
              ))) %>% reduce(full_join)
  }

  if (time) {
    end_time <- Sys.time()
    print(paste0('Estimation for all ys and xs took (seconds):',
                 end_time - start_time))
  }

  return(df.reg.out.all)
}
```

5.2.2.2 Prepare Data

```r
# Library
library(tidyverse)
library(AER)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')

# Source Dependency
source('C:/Users/fan/R4Econ/linreg/ivreg/ivregdfrow.R')

# Setting
options(repr.matrix.max.rows=50, repr.matrix.max.cols=50)
```

Parameters.

```r
var.y1 <- c('hgt')
var.y2 <- c('wgt')
var.y3 <- c('vil.id')
list.vars.y <- c(var.y1, var.y2, var.y3)

var.x1 <- c('prot')
var.x2 <- c('cal')
var.x3 <- c('wealthIdx')
var.x4 <- c('p.A.prot')
var.x5 <- c('p.A.nProt')
list.vars.x <- c(var.x1, var.x2, var.x3, var.x4, var.x5)

vars.z <- c('indi.id')
vars.c <- c('sex', 'wgt0', 'hgt0', 'svymthRound')
```

5.2.2.3 Program Testing

```
vars.z <- NULL
suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = FALSE,
              stats_ends = 'value'))) %>%
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_tvalue	cal_tvalue	wealthIdx_tvalue	p.A.prot_tvalue	p.A.nProt_tvalue
hgt	18.8756010031766	23.4421863484656	13.5088996182178	3.83682180045522	32.5448257554849
wgt	16.359112505607	17.368603130933	14.1390521528125	1.36958319982295	12.0961557911473
vil.id	-14.9385580468905	-19.6150110809449	34.0972558327354	8.45943342783161	17.7801422421398

5.2.2.3.1 Test Program OLS Z-Stat

```
vars.z <- c('indi.id')
suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = FALSE,
              stats_ends = 'value'))) %>%
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_zvalue	cal_zvalue	wealthIdx_zvalue	p.A.prot_zvalue	p.A.nProt_zvalue
hgt	8.87674929306359	12.0739764946734	4.62589553677888	26.6373587567245	32.1162192385715
wgt	5.60385871757577	6.12251870088371	5.17869536991513	11.9295584469951	12.3509307017258
vil.id	-9.22106223346427	-13.0586007975956	-51.5866689219473	-29.9627476577358	-38.3528894620791

5.2.2.3.2 Test Program IV T-stat

```
vars.z <- NULL
suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = FALSE,
              stats_ends = 'Estimate'))) %>%
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_Estimate	cal_Estimate	wealthIdx_Estimate	p.A.prot_Estimate	p.A.nProt_Estimate
hgt	0.0494310938067476	0.00243408846205617	0.210456554881893	3.86952250259533e-05	0.00542428867316432
wgt	16.5557424523601	0.69907250036464	106.678721085982	0.00521731297924599	0.77951423205071
vil.id	-0.0758835879205561	-0.00395676177098467	0.451733304543376	0.000149388430455129	0.00526237555580908

5.2.2.3.3 Test Program OLS Coefficient

```
vars.z <- c('indi.id')
suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = FALSE,
              stats_ends = 'Estimate'))) %>%
```

```
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_Estimate	cal_Estimate	wealthIdx_Estimate	p.A.prot_Estimate	p.A.nProt_Estimate
hgt	0.859205733631884	0.0238724384575233	0.144503490136916	0.00148073028434634	0.0141317656200728
wgt	98.9428234201429	2.71948246216963	69.1816142882735	0.221916473012471	2.1185694049434
vil.id	-6.02451379136l3	-0.168054407187466	-1.91414470908346	-0.00520794333267228	-0.0494468877742103

5.2.2.3.4 Test Program IV coefficient

```
vars.z <- NULL
t(suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = TRUE,
              stats_ends = 'Estimate')))) %>%
  kable() %>%
  kable_styling_fc_wide()
```

5.2.2.3.5 Test Program OLS Return All

```
vars.z <- c('indi.id')
t(suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = TRUE,
              stats_ends = 'Estimate')))) %>%
  kable() %>%
  kable_styling_fc_wide()
```

5.2.2.3.6 Test Program IV Return All

5.2.2.4 Program Line by Line

Set Up Parameters

```
vars.z <- c('indi.id')
vars.z <- NULL
vars.c <- c('sex', 'wgt0', 'hgt0', 'svymthRound')
```

```
df.reg.out <- as_tibble(
  bind_rows(lapply(list.vars.y, regf.iv,
                   vars.x=var.x1, vars.c=vars.c, vars.z=vars.z, df=df)))
```

5.2.2.4.1 Lapply

```
lapply(list.vars.y, function(y) (mean(df[[var.x1]], na.rm=TRUE) +
                                  mean(df[[y]], na.rm=TRUE)))
```

5.2.2.4.2 Nested Lapply Test

```
## [[1]]
## [1] 98.3272
##
## [[2]]
## [1] 13626.51
##
## [[3]]
## [1] 26.11226
```

```
lapplytwice <- lapply(
  list.vars.x, function(x) (
    lapply(list.vars.y, function(y) (mean(df[[x]], na.rm=TRUE) +
                                      mean(df[[y]], na.rm=TRUE)))))
# lapplytwice
```

```
df.reg.out.all <- bind_rows(
  lapply(list.vars.x,
         function(x) (
           bind_rows(
             lapply(list.vars.y, regf.iv,
                    vars.x=x, vars.c=vars.c, vars.z=vars.z, df=df))
         )))
```

```
# df.reg.out.all %>%
#   kable() %>%
#   kable_styling_fc_wide()
```

5.2.2.4.3 Nested Lapply All

```
df.reg.out.all <-
  (lapply(list.vars.x,
          function(x) (
            bind_rows(lapply(list.vars.y, regf.iv,
                             vars.x=x, vars.c=vars.c, vars.z=vars.z, df=df)) %>%
              select(vars_var.y, starts_with(x)) %>%
              select(vars_var.y, ends_with('value'))
          ))) %>% reduce(full_join)
```

```
df.reg.out.all %>%
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_tvalue	cal_tvalue	wealthIdx_tvalue	p.A.prot_tvalue	p.A.nProt_tvalue
hgt	18.8756010031766	23.4421863484656	13.5088996182178	3.83682180045522	32.5448257554849
wgt	16.359112505607	17.368603130933	14.1390521528125	1.36958319982295	12.0961557911473
vil.id	-14.9385580468905	-19.6150110809449	34.0972558327354	8.45943342783161	17.7801422421398

5.2.2.4.4 Nested Lapply Select

5.3 Decomposition

5.3.1 Decompose RHS

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

One runs a number of regressions. With different outcomes, and various right hand side variables.

What is the remaining variation in the left hand side variable if right hand side variable one by one is set to the average of the observed values.

- Dependency: *R4Econ/linreg/ivreg/ivregdfrow.R*

The code below does not work with categorical variables (except for dummies). Dummy variable inputs need to be converted to zero/one first. The examples are just to test the code with different types of variables.

```
# Library
library(tidyverse)
library(AER)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')

# Source Dependency
source('C:/Users/fan/R4Econ/linreg/ivreg/ivregdfrow.R')
```

Data Cleaning.

```
# Convert Variable for Sex which is categorical to Numeric
df <- df
df$male <- (as.numeric(factor(df$sex)) - 1)
summary(factor(df$sex))
```

```
## Female   Male
##  16446  18619
```

```
summary(df$male)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   0.531   1.000   1.000
```

```
df.use <- df %>% filter(S.country == 'Guatemala') %>%
  filter(svymthRound %in% c(12, 18, 24))
dim(df.use)
```

```
## [1] 2022   16
```

Setting Up Parameters.

```
# Define Left Hand Side Variab les
var.y1 <- c('hgt')
var.y2 <- c('wgt')
vars.y <- c(var.y1, var.y2)
# Define Right Hand Side Variables
vars.x <- c('prot')
vars.c <- c('male', 'wgt0', 'hgt0', 'svymthRound')
# vars.z <- c('p.A.prot')
vars.z <- c('vil.id')
# vars.z <- NULL
vars.xc <- c(vars.x, vars.c)

# Other variables to keep
vars.other.keep <- c('S.country', 'vil.id', 'indi.id', 'svymthRound')

# Decompose sequence
vars.tomean.first <- c('male', 'hgt0')
var.tomean.first.name.suffix <- '_mh02m'
vars.tomean.second <- c(vars.tomean.first, 'hgt0', 'wgt0')
var.tomean.second.name.suffix <- '_mh0me2m'
vars.tomean.third <- c(vars.tomean.second, 'prot')
var.tomean.third.name.suffix <- '_mh0mep2m'
vars.tomean.fourth <- c(vars.tomean.third, 'svymthRound')
var.tomean.fourth.name.suffix <- '_mh0mepm2m'
list.vars.tomean = list(
#                         vars.tomean.first,
                        vars.tomean.second,
                        vars.tomean.third,
                        vars.tomean.fourth
                        )
list.vars.tomean.name.suffix <- list(
#                                     var.tomean.first.name.suffix,
                                     var.tomean.second.name.suffix,
                                     var.tomean.third.name.suffix,
                                     var.tomean.fourth.name.suffix
                                    )
```

5.3.1.1 Obtain Regression Coefficients from somewhere

```
# Regressions
# regf.iv from C:\Users\fan\R4Econ\linreg\ivreg\ivregdfrow.R
df.reg.out <- as_tibble(
  bind_rows(lapply(vars.y, regf.iv,
                   vars.x=vars.x, vars.c=vars.c, vars.z=vars.z, df=df)))
# Regressions
# reg1 <- regf.iv(var.y = var.y1, vars.x, vars.c, vars.z, df.use)
# reg2 <- regf.iv(var.y = var.y2, vars.x, vars.c, vars.z, df.use)
# df.reg.out <- as_tibble(bind_rows(reg1, reg2))
```

```
# df.reg.out
```

```
# Select Variables
str.esti.suffix <- '_Estimate'
arr.esti.name <- paste0(vars.xc, str.esti.suffix)
str.outcome.name <- 'vars_var.y'
arr.columns2select <- c(arr.esti.name, str.outcome.name)
arr.columns2select
```

	prot_Estimate	male_Estimate	wgt0_Estimate	hgt0_Estimate	svymthRound_Estimate
hgt	-0.2714772	1.244735	0.0004430	0.6834853	1.133919
wgt	-59.0727542	489.852902	0.7696158	75.4867897	250.778883

S.country	vil.id	indi.id	svymthRound	prot	male	wgt0	hgt0	variable	value
Guatemala	3	1352	18	13.3	1	2545.2	47.4	hgt	70.2
Guatemala	3	1352	24	46.3	1	2545.2	47.4	hgt	75.8
Guatemala	3	1354	12	1.0	1	3634.3	51.2	hgt	66.3
Guatemala	3	1354	18	9.8	1	3634.3	51.2	hgt	69.2
Guatemala	3	1354	24	15.4	1	3634.3	51.2	hgt	75.3
Guatemala	3	1356	12	8.6	1	3911.8	51.9	hgt	68.1
Guatemala	3	1356	18	17.8	1	3911.8	51.9	hgt	74.1
Guatemala	3	1356	24	30.5	1	3911.8	51.9	hgt	77.1
Guatemala	3	1357	12	1.0	1	3791.4	52.6	hgt	71.5
Guatemala	3	1357	18	12.7	1	3791.4	52.6	hgt	77.8

```
## [1] "prot_Estimate"        "male_Estimate"        "wgt0_Estimate"
## [4] "hgt0_Estimate"        "svymthRound_Estimate" "vars_var.y"
```

```
# Generate dataframe for coefficients
df.coef <- df.reg.out[,c(arr.columns2select)] %>%
  mutate_at(vars(arr.esti.name), as.numeric) %>% column_to_rownames(str.outcome.name)
df.coef %>%
  kable() %>%
  kable_styling_fc()
```

```
str(df.coef)
```

```
## 'data.frame':    2 obs. of  5 variables:
##  $ prot_Estimate       : num  -0.271 -59.073
##  $ male_Estimate       : num  1.24 489.85
##  $ wgt0_Estimate       : num  0.000443 0.769616
##  $ hgt0_Estimate       : num  0.683 75.487
##  $ svymthRound_Estimate: num  1.13 250.78
```

5.3.1.2 Decomposition Step 1

```
# Decomposition Step 1: gather
df.decompose_step1 <- df.use %>%
                        filter(svymthRound %in% c(12, 18, 24)) %>%
                        select(one_of(c(vars.other.keep, vars.xc, vars.y))) %>%
                        drop_na() %>%
                        gather(variable, value, -one_of(c(vars.other.keep, vars.xc)))
options(repr.matrix.max.rows=20, repr.matrix.max.cols=20)
dim(df.decompose_step1)
```

```
## [1] 1382   10
```

```
head(df.decompose_step1, 10) %>%
  kable() %>%
  kable_styling_fc()
```

5.3.1.3 Decomposition Step 2

```
# Decomposition Step 2: mutate_at(vars, funs(mean = mean(.)))
# the xc averaging could have taken place earlier, no difference in mean across variables
df.decompose_step2 <- df.decompose_step1 %>%
                        group_by(variable) %>%
```

```
                        mutate_at(vars(c(vars.xc, 'value')), funs(mean = mean(.))) %>%
                        ungroup()

options(repr.matrix.max.rows=20, repr.matrix.max.cols=20)
dim(df.decompose_step2)
```

```
## [1] 1382   16
```

```
head(df.decompose_step2,10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

S.country	vil.id	indi.id	svymthRound	prot	male	wgt0	hgt0	variable	value	prot_mean	male_mean	wgt0_mean	hgt0_mean	svymthRound_mean	value_mean
Guatemala	3	1352	18	13.3	1	2545.2	47.4	hgt	70.2	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1352	24	46.3	1	2545.2	47.4	hgt	75.8	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1354	12	1.0	1	3634.3	51.2	hgt	66.3	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1354	18	9.8	1	3634.3	51.2	hgt	69.2	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1354	24	15.4	1	3634.3	51.2	hgt	75.3	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1356	12	8.6	1	3911.8	51.9	hgt	68.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1356	18	17.8	1	3911.8	51.9	hgt	74.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1356	24	30.5	1	3911.8	51.9	hgt	77.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1357	12	1.0	1	3791.4	52.6	hgt	71.5	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1357	18	12.7	1	3791.4	52.6	hgt	77.8	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216

5.3.1.4 Decomposition Step 3 Non-Loop

```
ff_lr_decompose_valadj <- function(df, df.coef, vars.tomean, str.esti.suffix) {
  new_value <- (df$value +
                rowSums((df[paste0(vars.tomean, '_mean')] - df[vars.tomean])
                        *df.coef[df$variable, paste0(vars.tomean, str.esti.suffix)]))
  return(new_value)
}
```

5.3.1.5 Decomposition Step 3 With Loop

```
df.decompose_step3 <- df.decompose_step2
for (i in 1:length(list.vars.tomean)) {
    var.decomp.cur <- (paste0('value', list.vars.tomean.name.suffix[[i]]))
    vars.tomean <- list.vars.tomean[[i]]
    var.decomp.cur
    df.decompose_step3 <- df.decompose_step3 %>%
      mutate((!!var.decomp.cur) :=
               ff_lr_decompose_valadj(., df.coef, vars.tomean, str.esti.suffix))

}

dim(df.decompose_step3)
```

```
## [1] 1382   19
```

```
head(df.decompose_step3, 10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

S.country	vil.id	indi.id	svymthRound	prot	male	wgt0	hgt0	variable	value	prot_mean	male_mean	wgt0_mean	hgt0_mean	svymthRound_mean	value_mean	value_mh0me2m	value_mh0mep2m	value_mh0mepm2m
Guatemala	3	1352	18	13.3	1	2545.2	47.4	hgt	70.2	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	73.19390	71.19903	71.68148
Guatemala	3	1352	24	46.3	1	2545.2	47.4	hgt	75.8	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	78.79390	85.75778	79.43671
Guatemala	3	1354	12	1.0	1	3634.3	51.2	hgt	66.3	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	63.61689	58.28285	65.56882
Guatemala	3	1354	18	9.8	1	3634.3	51.2	hgt	69.2	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	66.51689	63.57185	64.05430
Guatemala	3	1354	24	15.4	1	3634.3	51.2	hgt	75.3	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	72.61689	71.19213	64.87106
Guatemala	3	1356	12	8.6	1	3911.8	51.9	hgt	68.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	64.33707	61.06626	68.35222
Guatemala	3	1356	18	17.8	1	3911.8	51.9	hgt	74.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	70.33707	69.56385	70.04630
Guatemala	3	1356	24	30.5	1	3911.8	51.9	hgt	77.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	73.33707	76.01161	69.69055
Guatemala	3	1357	12	1.0	1	3791.4	52.6	hgt	71.5	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	66.83353	61.49949	68.78545
Guatemala	3	1357	18	12.7	1	3791.4	52.6	hgt	77.8	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	73.13353	70.97578	71.45823

```r
df.decompose_step3 %>%
        select(variable, contains('value')) %>%
        group_by(variable) %>%
        summarize_all(funs(mean = mean, var = var)) %>%
        select(matches('value')) %>% select(ends_with("_var")) %>%
        mutate_if(is.numeric, funs( frac = (./value_var))) %>%
        mutate_if(is.numeric, round, 3) %>%
  kable() %>%
  kable_styling_fc_wide()
```

value_var	value_mean_var	value_mh0me2m_var	value_mh0mep2m_var	value_mh0mepm2m_var	value_var_frac	value_mean_var_frac	value_mh0me2m_var_frac	value_mh0mep2m_var_frac	value_mh0mepm2m_var_frac
21.864	NA	25.35	49.047	23.06	1	NA	1.159	2.243	1.055
2965693.245	NA	2949187.64	4192769.518	3147506.60	1	NA	0.994	1.414	1.061

5.3.1.7 Graphical Results

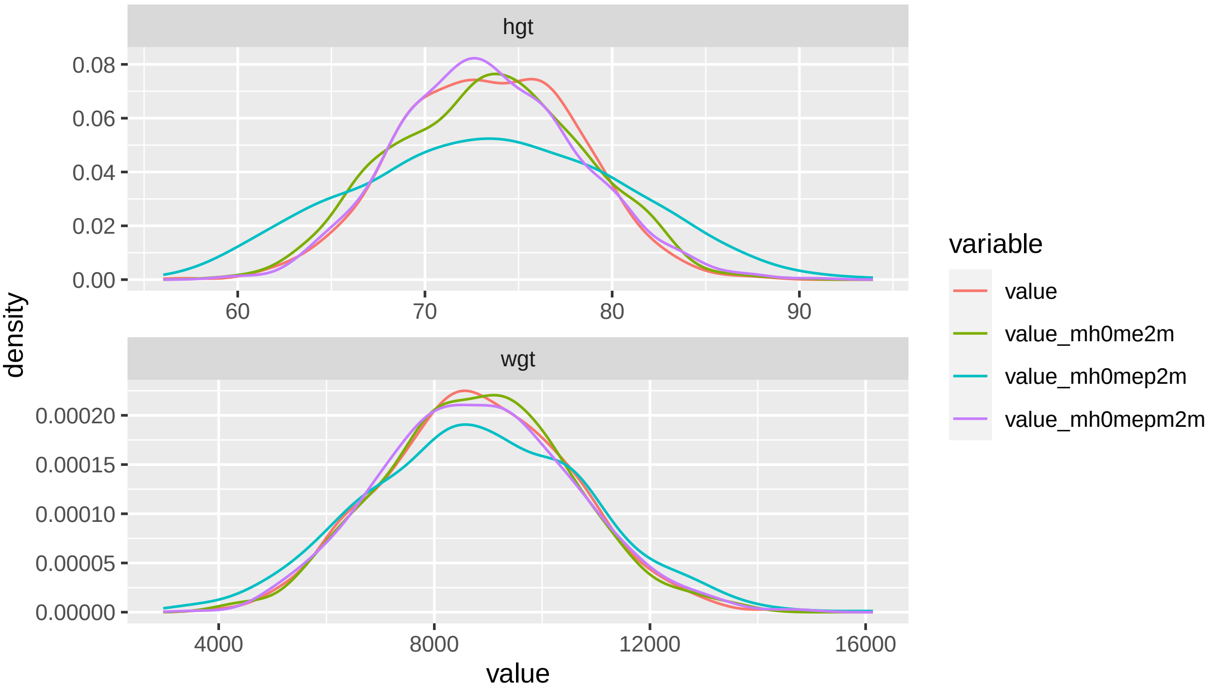
Graphically, difficult to pick up exact differences in variance, a 50 percent reduction in variance visually does not look like 50 percent. Intuitively, we are kind of seeing standard deviation, not variance on the graph if we think abou the x-scale.

```r
head(df.decompose_step3 %>%
    select(variable, contains('value'), -value_mean), 10) %>%
  kable() %>%
  kable_styling_fc()
```

```r
df.decompose_step3 %>%
    select(variable, contains('value'), -value_mean) %>%
    rename(outcome = variable) %>%
    gather(variable, value, -outcome) %>%
    ggplot(aes(x=value, color = variable, fill = variable)) +
        geom_line(stat = "density") +
        facet_wrap(~ outcome, scales='free', nrow=2)
```

variable	value	value_mh0me2m	value_mh0mep2m	value_mh0mepm2m
hgt	70.2	73.19390	71.19903	71.68148
hgt	75.8	78.79390	85.75778	79.43671
hgt	66.3	63.61689	58.28285	65.56882
hgt	69.2	66.51689	63.57185	64.05430
hgt	75.3	72.61689	71.19213	64.87106
hgt	68.1	64.33707	61.06626	68.35222
hgt	74.1	70.33707	69.56385	70.04630
hgt	77.1	73.33707	76.01161	69.69055
hgt	71.5	66.83353	61.49949	68.78545
hgt	77.8	73.13353	70.97578	71.45823

variable	value_mean	pred_new_mean	value_sd	pred_new_sd
hgt	73.41216	73.41216	4.675867	4.534947
wgt	8807.87656	8807.87656	1722.118824	1695.221845



5.3.1.8 Additional Decomposition Testings

```
head(df.decompose_step2[vars.tomean.first],3)
head(df.decompose_step2[paste0(vars.tomean.first, '_mean')], 3)
head(df.coef[df.decompose_step2$variable,
             paste0(vars.tomean.first, str.esti.suffix)], 3)
df.decompose.tomean.first <- df.decompose_step2 %>%
    mutate(pred_new = df.decompose_step2$value +
        rowSums((df.decompose_step2[paste0(vars.tomean.first, '_mean')]
                 - df.decompose_step2[vars.tomean.first])
             *df.coef[df.decompose_step2$variable,
                      paste0(vars.tomean.first, str.esti.suffix)])) %>%
        select(variable, value, pred_new)
head(df.decompose.tomean.first, 10)
df.decompose.tomean.first %>%
        group_by(variable) %>%
        summarize_all(funs(mean = mean, sd = sd))  %>%
  kable() %>%
  kable_styling_fc()
```

Note the r-square from regression above matches up with the 1 - ratio below. This is the proper decomposition method that is equivalent to r2.

```
df.decompose_step2 %>%
    mutate(pred_new = df.decompose_step2$value +
        rowSums((df.decompose_step2[paste0(vars.tomean.second, '_mean')]
                 - df.decompose_step2[vars.tomean.second])
             *df.coef[df.decompose_step2$variable,
                      paste0(vars.tomean.second, str.esti.suffix)])) %>%
        select(variable, value, pred_new) %>%
        group_by(variable) %>%
```

variable	value_mean	pred_new_mean	value_var	pred_new_var	ratio
hgt	73.41216	73.41216	2.186374e+01	25.3504	1.1594724
wgt	8807.87656	8807.87656	2.965693e+06	2949187.6357	0.9944345

```
      summarize_all(funs(mean = mean, var = var)) %>%
      mutate(ratio = (pred_new_var/value_var))  %>%
  kable() %>%
  kable_styling_fc()
```

Chapter 6

Nonlinear and Other Regressions

6.1 Logit Regression

6.1.1 Binary Logit

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Data Preparation

```
df_mtcars <- mtcars

# X-variables to use on RHS
ls_st_xs <- c('mpg', 'qsec')
ls_st_xs <- c('mpg')
ls_st_xs <- c('qsec')
ls_st_xs <- c('wt')
ls_st_xs <- c('mpg', 'wt', 'vs')

svr_binary <- 'hpLowHigh'
svr_binary_lb0 <- 'LowHP'
svr_binary_lb1 <- 'HighHP'
svr_outcome <- 'am'
sdt_name <- 'mtcars'

# Discretize hp
df_mtcars <- df_mtcars %>%
    mutate(!!sym(svr_binary) := cut(hp,
                               breaks=c(-Inf, 210, Inf),
                               labels=c(svr_binary_lb0, svr_binary_lb1)))
```

6.1.1.1 Logit Regresion and Prediction

logit regression with glm, and predict using estimation data. Prediction and estimation with one variable.

- LOGIT REGRESSION R DATA ANALYSIS EXAMPLES
- Generalized Linear Models

```
# Regress
rs_logit <- glm(as.formula(paste(svr_outcome, "~", paste(ls_st_xs, collapse="+")))
                ,data = df_mtcars, family = "binomial")
summary(rs_logit)
```

```
##
## Call:
```

```
## glm(formula = as.formula(paste(svr_outcome, "~", paste(ls_st_xs,
##     collapse = "+"))), family = "binomial", data = df_mtcars)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 22.53287   13.93556   1.617   0.1059
## mpg         -0.01919    0.33693  -0.057   0.9546
## wt          -6.68827    3.02783  -2.209   0.0272 *
## vs          -4.38343    2.86743  -1.529   0.1263
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 41.381  on 30  degrees of freedom
## Residual deviance: 13.073  on 27  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 21.073
##
## Number of Fisher Scoring iterations: 7
```

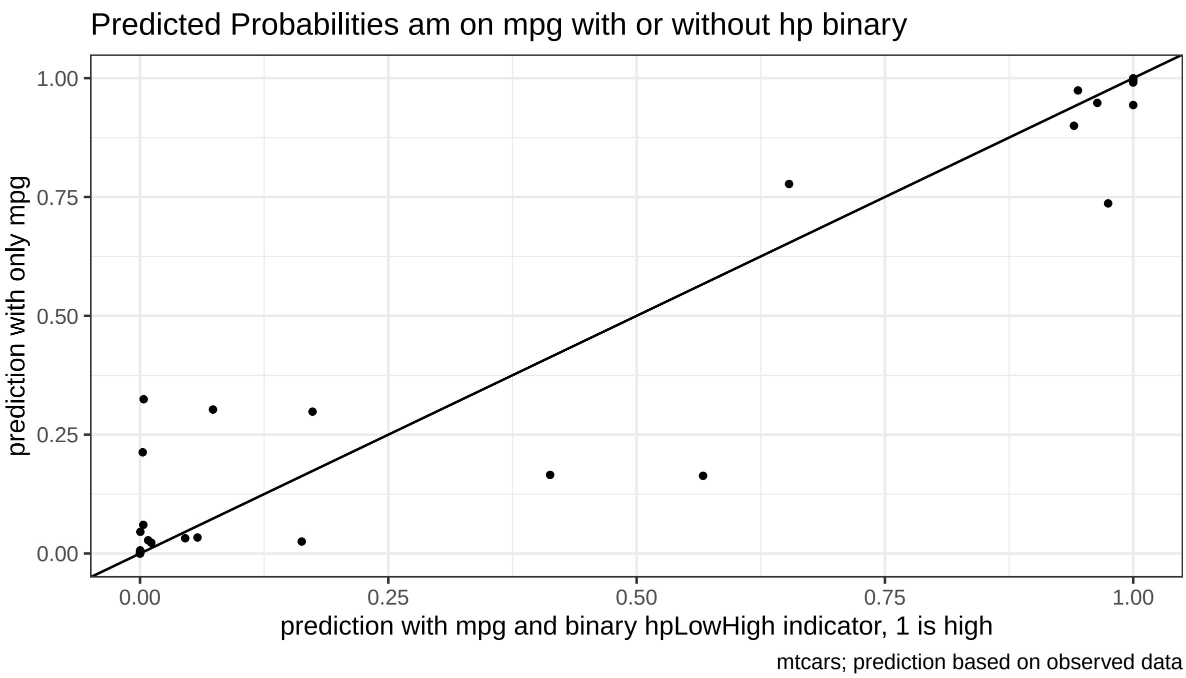
```
# Predcit Using Regression Data
df_mtcars$p_mpg <- predict(rs_logit, newdata = df_mtcars, type = "response")
```

6.1.1.1.1 Prediction with Observed Binary Input Logit regression with a continuous variable and a binary variable. Predict outcome with observed continuous variable as well as observed binary input variable.

```
# Regress
rs_logit_bi <- glm(as.formula(paste(svr_outcome,
                                    "~ factor(", svr_binary,") + ",
                                    paste(ls_st_xs, collapse="+")))
                   , data = df_mtcars, family = "binomial")
summary(rs_logit_bi)
```

```
##
## Call:
## glm(formula = as.formula(paste(svr_outcome, "~ factor(", svr_binary,
##     ") + ", paste(ls_st_xs, collapse = "+"))), family = "binomial",
##     data = df_mtcars)
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)               3.8614    18.0138   0.214   0.8303
## factor(hpLowHigh)HighHP   6.9559     5.5134   1.262   0.2071
## mpg                       0.8874     0.8941   0.993   0.3209
## wt                       -6.6834     3.3355  -2.004   0.0451 *
## vs                       -5.8324     4.2498  -1.372   0.1699
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 41.3808  on 30  degrees of freedom
## Residual deviance:  8.9646  on 26  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 18.965
##
## Number of Fisher Scoring iterations: 9
```

```r
# Predcit Using Regresion Data
df_mtcars$p_mpg_hp <- predict(rs_logit_bi, newdata = df_mtcars, type = "response")

# Predicted Probabilities am on mgp with or without hp binary
scatter <- ggplot(df_mtcars, aes(x=p_mpg_hp, y=p_mpg)) +
      geom_point(size=1) +
      # geom_smooth(method=lm) + # Trend line
      geom_abline(intercept = 0, slope = 1) + # 45 degree line
      labs(title = paste0('Predicted Probabilities ', svr_outcome, ' on ', ls_st_xs, ' with or witho
           x = paste0('prediction with ', ls_st_xs, ' and binary ', svr_binary, ' indicator, 1 is hi
           y = paste0('prediction with only ', ls_st_xs),
           caption = 'mtcars; prediction based on observed data') +
      theme_bw()
print(scatter)
```



6.1.1.1.2 Prediction with Binary set to 0 and 1 Now generate two predictions. One set where binary input is equal to 0, and another where the binary inputs are equal to 1. Ignore whether in data binary input is equal to 0 or 1. Use the same regression results as what was just derived.

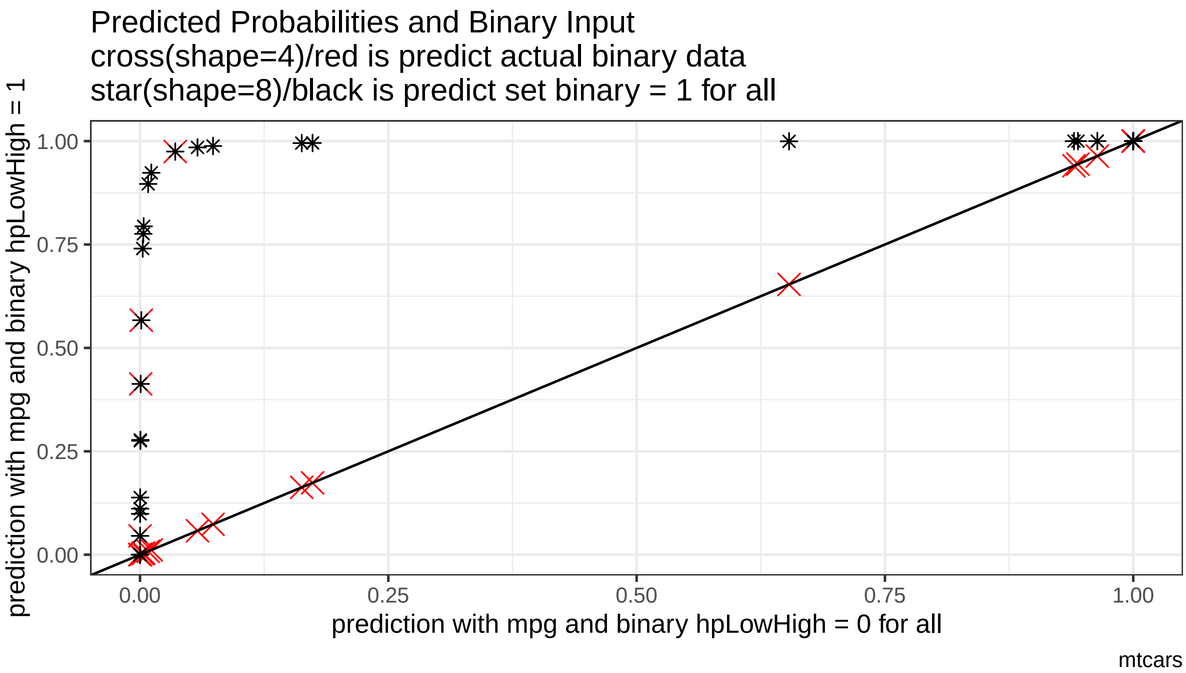
Note that given the example here, the probability changes a lot when we

```r
# Previous regression results
summary(rs_logit_bi)
```

```
##
## Call:
## glm(formula = as.formula(paste(svr_outcome, "~ factor(", svr_binary,
##     ") + ", paste(ls_st_xs, collapse = "+"))), family = "binomial",
##     data = df_mtcars)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                3.8614    18.0138   0.214   0.8303
## factor(hpLowHigh)HighHP    6.9559     5.5134   1.262   0.2071
## mpg                        0.8874     0.8941   0.993   0.3209
```

```
## wt                         -6.6834     3.3355  -2.004   0.0451 *
## vs                         -5.8324     4.2498  -1.372   0.1699
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 41.3808  on 30  degrees of freedom
## Residual deviance:  8.9646  on 26  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 18.965
##
## Number of Fisher Scoring iterations: 9
```

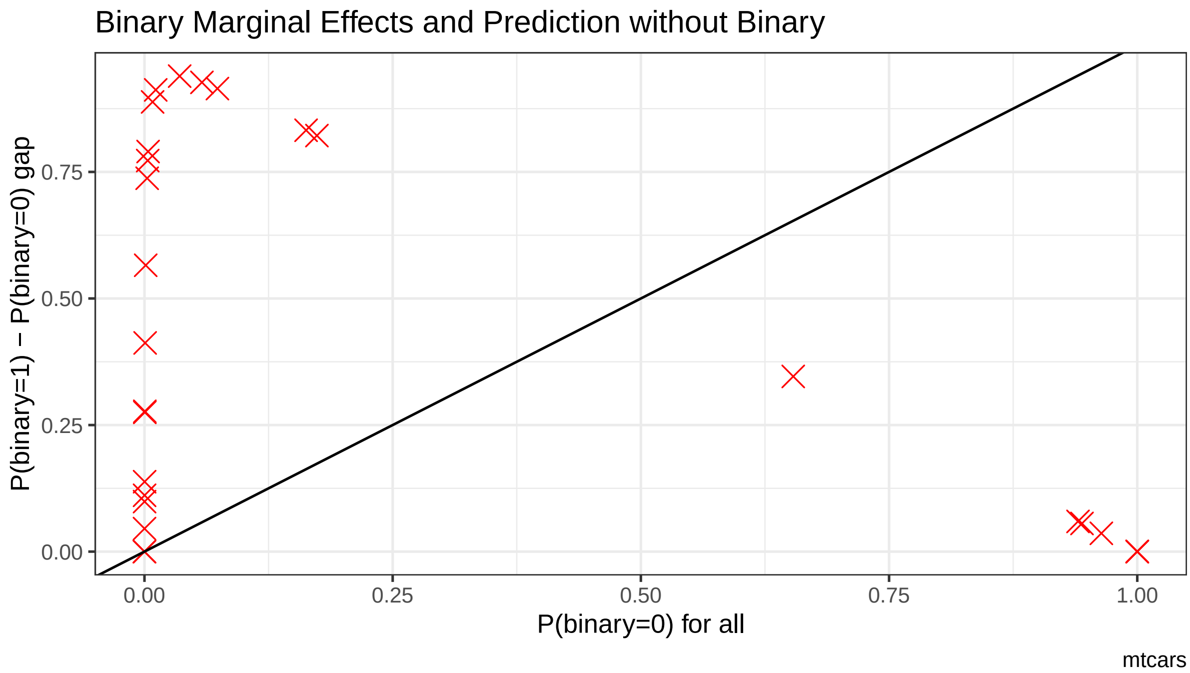
```
# Two different dataframes, mutate the binary regressor
df_mtcars_bi0 <- df_mtcars %>% mutate(!!sym(svr_binary) := svr_binary_lb0)
df_mtcars_bi1 <- df_mtcars %>% mutate(!!sym(svr_binary) := svr_binary_lb1)

# Predcit Using Regresion Data
df_mtcars$p_mpg_hp_bi0 <- predict(rs_logit_bi, newdata = df_mtcars_bi0, type = "response")
df_mtcars$p_mpg_hp_bi1 <- predict(rs_logit_bi, newdata = df_mtcars_bi1, type = "response")

# Predicted Probabilities and Binary Input
scatter <- ggplot(df_mtcars, aes(x=p_mpg_hp_bi0)) +
      geom_point(aes(y=p_mpg_hp), size=4, shape=4, color="red") +
      geom_point(aes(y=p_mpg_hp_bi1), size=2, shape=8) +
      # geom_smooth(method=lm) + # Trend line
      geom_abline(intercept = 0, slope = 1) + # 45 degree line
      labs(title = paste0('Predicted Probabilities and Binary Input',
                          '\ncross(shape=4)/red is predict actual binary data',
                          '\nstar(shape=8)/black is predict set binary = 1 for all'),
            x = paste0('prediction with ', ls_st_xs, ' and binary ', svr_binary, ' = 0 for all'),
            y = paste0('prediction with ', ls_st_xs, ' and binary ', svr_binary, ' = 1'),
           caption = paste0(sdt_name)) +
      theme_bw()
print(scatter)
```



6.1.1.1.3 Prediction with Binary set to 0 and 1 Difference What is the difference in probability between binary = 0 vs binary = 1. How does that relate to the probability of outcome of interest when binary = 0 for all.

In the binary logit case, the relationship will be hump–shaped by construction between A_i and α_i. In the exponential wage cases, the relationship is convex upwards.

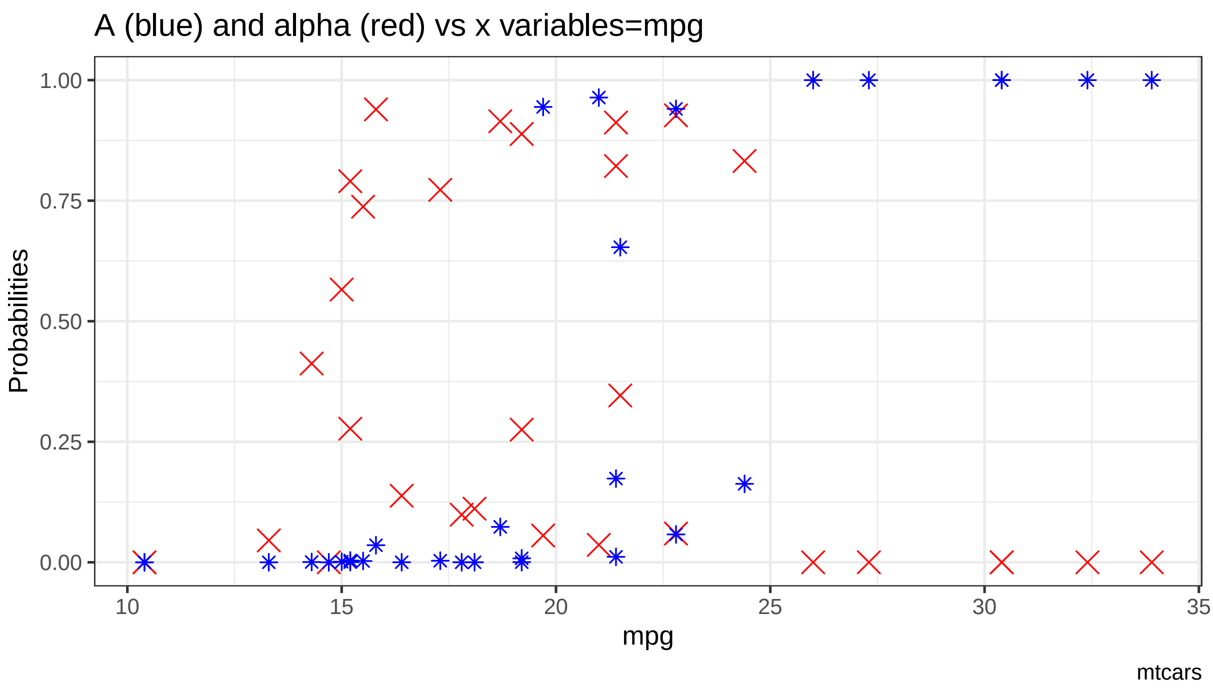
```
# Generate Gap Variable
df_mtcars <- df_mtcars %>% mutate(alpha_i = p_mpg_hp_bi1 - p_mpg_hp_bi0) %>%
                mutate(A_i = p_mpg_hp_bi0)

# Binary Marginal Effects and Prediction without Binary
scatter <- ggplot(df_mtcars, aes(x=A_i)) +
      geom_point(aes(y=alpha_i), size=4, shape=4, color="red") +
      geom_abline(intercept = 0, slope = 1) + # 45 degree line
      labs(title = paste0('Binary Marginal Effects and Prediction without Binary'),
           x = 'P(binary=0) for all',
           y = 'P(binary=1) - P(binary=0) gap',
           caption = paste0(sdt_name)) +
      theme_bw()
print(scatter)
```



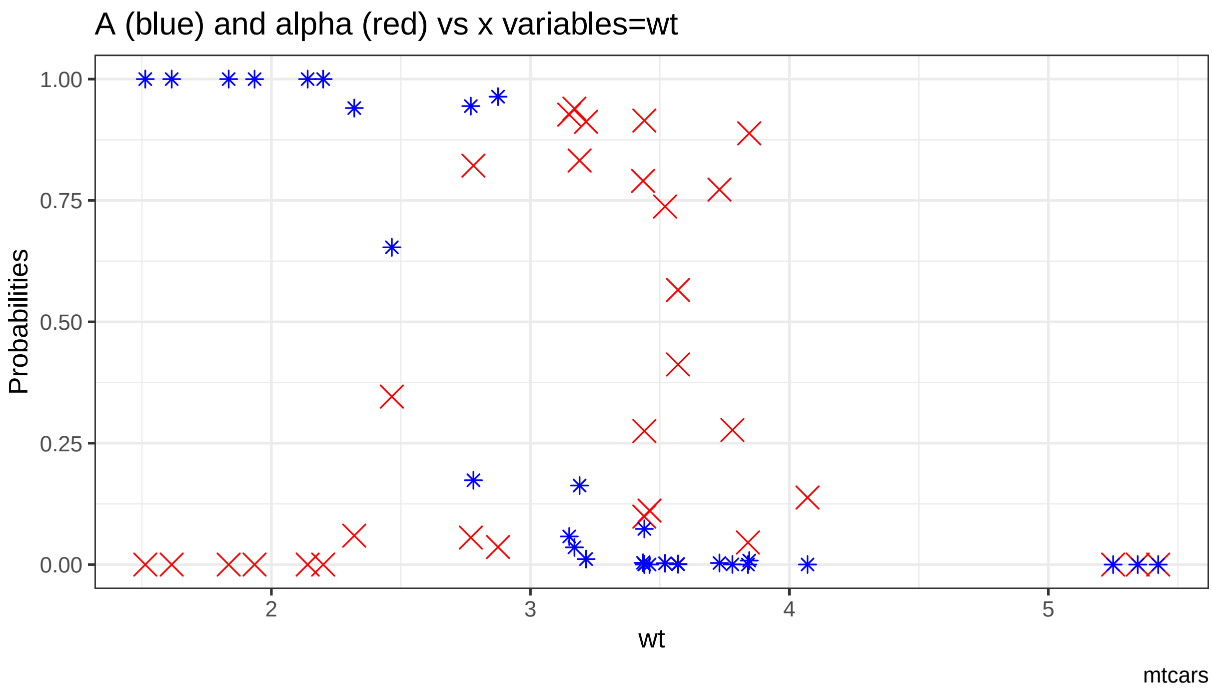
6.1.1.1.4 X variables and A and alpha Given the x-variables included in the logit regression, how do they relate to A_i and alpha_i

```
# Generate Gap Variable
df_mtcars <- df_mtcars %>% mutate(alpha_i = p_mpg_hp_bi1 - p_mpg_hp_bi0) %>%
                mutate(A_i = p_mpg_hp_bi0)

# Binary Marginal Effects and Prediction without Binary
ggplot.A.alpha.x <- function(svr_x, df,
                             svr_alpha = 'alpha_i', svr_A = "A_i"){

  scatter <- ggplot(df, aes(x=!!sym(svr_x))) +
        geom_point(aes(y=alpha_i), size=4, shape=4, color="red") +
        geom_point(aes(y=A_i), size=2, shape=8, color="blue") +
        geom_abline(intercept = 0, slope = 1) + # 45 degree line
        labs(title = paste0('A (blue) and alpha (red) vs x variables=', svr_x),
             x = svr_x,
             y = 'Probabilities',
             caption = paste0(sdt_name)) +
        theme_bw()

return(scatter)
}

# Plot over multiple
lapply(ls_st_xs,
       ggplot.A.alpha.x,
       df = df_mtcars)
```

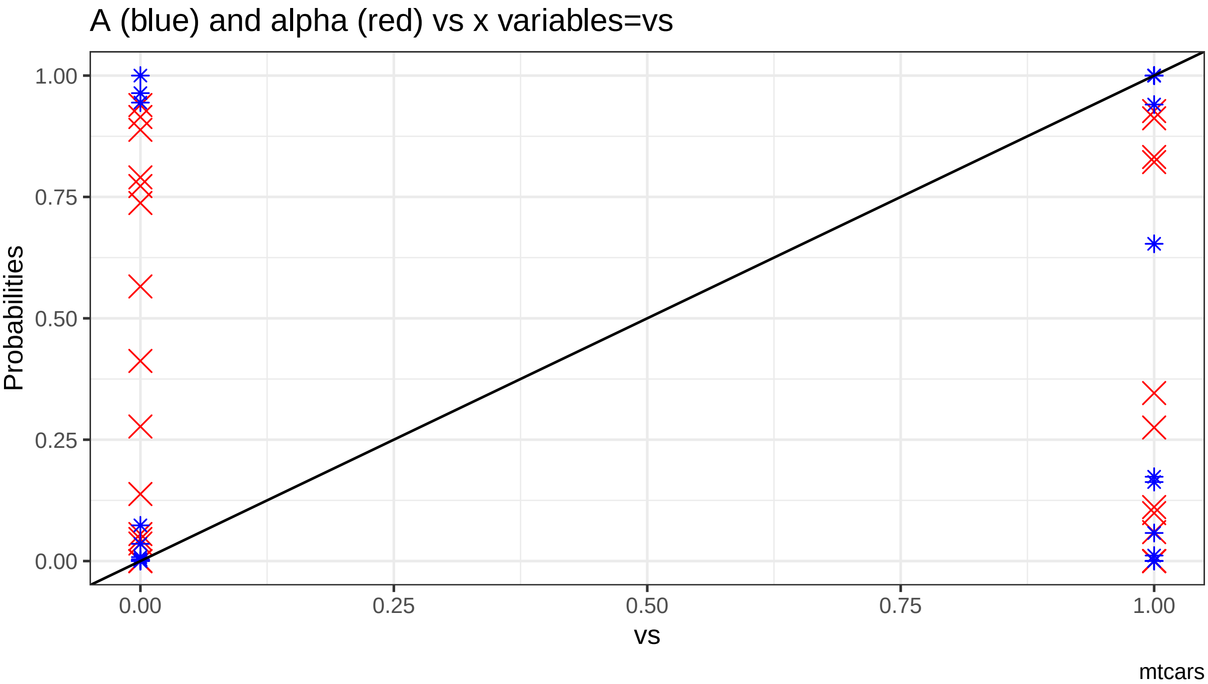
```
## [[1]]
```


A (blue) and alpha (red) vs x variables=mpg
Probabilities
1.00
0.75
0.50
0.25
0.00
10
15
20
25
30
35
mpg
mtcars

```
##
## [[2]]
```


A (blue) and alpha (red) vs x variables=wt
Probabilities
1.00
0.75
0.50
0.25
0.00
2
3
4
5
wt
mtcars

```
##
## [[3]]
```



6.1.2 Logistic Choice Model with Aggregate Shares

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

6.1.2.1 Logistic Choices, Wages and Market Shares

6.1.2.1.1 Wage and Aggregate Share of Workers *Note*: See Fit Prices Given Quantities Logistic Choice with Aggregate Data for solving/estimating for wages given aggregate shares.

Individual i can choose among $M+1$ options, the M options, are for example, alternative job possibilities. The $+1$ refers to a leisure category. The value associated with each one of the choice alternatives is:

$$V_{itm} = \widehat{V}_{tm} + \epsilon_{itm} = \alpha_m + \beta \cdot \text{WAGE}_{tm} + \epsilon_{itm}$$

Note that β captures the effect of occupation-specific wage, β is the same across occupational groups. Each occupational group has its own intercept α_m, and ϵ_{im} is the individual and occupation specific extreme value error. The non-error component of the value of leisure is normalized to 0 ($\exp(0) = 1$).

The discrete choice problem is solved by a comparison across alternatives. o_i is the individual optimal choice $o_i = \arg\max_m \left(V_{i,m}\right)$

Choice probabilities are functions of wages. The probability that individual i chooses occupation m is (ignoring the t-subscripts):

$$P(o = m) = \frac{\exp\left(\widehat{V}_m\right)}{1 + \sum_{\hat{m}=1}^{M} \exp\left(\widehat{V}_{\hat{m}}\right)}$$

The log ratio of probability of choosing any of the M occupation alternatives and leisure is:

$$\log P\left(o = m\right) - \log P\left(o = 0\right) = \log\left(\frac{P\left(o = m\right)}{P\left(o = 0\right)}\right) = \log\left(\frac{\exp\left(\widehat{V}_m\right)}{1 + \sum_{\hat{m}=1}^{M} \exp\left(\widehat{V}_{\hat{m}}\right)} \cdot \frac{1 + \sum_{\hat{m}=1}^{M} \exp\left(\widehat{V}_{\hat{m}}\right)}{1}\right)$$

This means that the log of the probability ratio is linear in wages:

$$\log\left(\frac{P(o=m)}{P(o=0)}\right) = \alpha_m + \beta \cdot \text{WAGE}_m$$

Supose we have $M-1$, either work or leisure. Suppose we have aggregate data from 1 time period, with relative work to leisure share (aggregate-share data, "market-share" data) and a single measure of wage, then we have 1 equation with two parameters, α and β can not be jointly identified. α and β, neither of which is time-varying, are exactly identified if we have data from two periods with variations in wage across the periods.

6.1.2.1.2 Simulate Market Share In this section, we now simulate the above model, with $M=2$ and data over three periods, and estimate α and β via OLS. Note that $m=0$ is leisure.

First, simulate the data.

```
# set seed
set.seed(123)
# T periods, and M occupations (+1 leisure)
it_T <- 3
it_M <- 2
# define alpha and beta parameters
ar_alpha <- runif(it_M) + 0
fl_beta <- runif(1) + 0
# wage matrix, no wage for leisure
mt_wages <- matrix(runif(it_T*it_M) + 1, nrow=it_T, ncol=it_M)
colnames(mt_wages) <- paste0('m', seq(1,it_M))
rownames(mt_wages) <- paste0('t', seq(1,it_T))
# Define a probability assignment function
ffi_logit_prob_alpha_beta <- function(ar_alpha, fl_beta, mt_wages) {
  # Dimensions
  it_T <- dim(mt_wages)[1]
  it_M <- dim(mt_wages)[2]
  # Aggregate probabilities
  mt_prob_o <- matrix(data=NA, nrow=it_T, ncol=it_M+1)
  colnames(mt_prob_o) <- paste0('m', seq(0,it_M))
  rownames(mt_prob_o) <- paste0('t', seq(1,it_T))
  # Generate Probabilities
  for (it_t in seq(1, it_T)) {
    # get current period wages
    ar_wage_at_t <- mt_wages[it_t, ]
    # Value without shocks/errors, M+1
    ar_V_hat_at_t <- c(0, ar_alpha + fl_beta*ar_wage_at_t)
    # Probabilities across M+1
    fl_prob_denominator <- sum(exp(ar_V_hat_at_t))
    ar_prob_at_t <- exp(ar_V_hat_at_t)/fl_prob_denominator
    # Fill in
    mt_prob_o[it_t,] <- ar_prob_at_t
  }
  return(mt_prob_o)
}
# Show probabilities
mt_prob_o <- ffi_logit_prob_alpha_beta(ar_alpha, fl_beta, mt_wages)
st_caption <- 'Occupation aggregate participation probabilities across time'
kable(mt_prob_o, caption=st_caption) %>% kable_styling_fc()
```

```
# mt_prob_o
```

6.1.2.1.3 Create Regression Data Inputs Second, generate relative market shares of various work occupations, columns 2 through $M+1$, with respect to leisure, column 1. If there are M categories and

Occupation aggregate participation probabilities across time

	m0	m1	m2
t1	0.1251712	0.3604563	0.5143725
t2	0.1147084	0.3381795	0.5471121
t3	0.1390180	0.2842316	0.5767504

Log of relative participation probabilities against leisure

	m1	m2
t1	1.057688	1.413265
t2	1.081184	1.562261
t3	0.715186	1.422806

T time periods, there are $M \times T$ observations (rows). Flatten the structure so that row-groups are the M categories, and we have time-specific information within row groups.

```
# A Matrix with share from 1:M columns
mt_prob_rela_m2leisure <-
  matrix(data=mt_prob_o[1:it_T, 2:(it_M+1)], nrow=it_T, ncol=it_M)
colnames(mt_prob_rela_m2leisure) <- paste0('m', seq(1,it_M))
rownames(mt_prob_rela_m2leisure) <- paste0('t', seq(1,it_T))
# Divide 1:M by leisure
mt_prob_rela_m2leisure <- mt_prob_rela_m2leisure/mt_prob_o[1:it_T, 1]
# Take Logs, log(Pm/Po)
mt_prob_rela_m2leisure_log <- log(mt_prob_rela_m2leisure)
# Flatten to single column
ar_prob_ols_output <- matrix(data=mt_prob_rela_m2leisure_log, nrow=it_T*it_M, ncol=1)
# Show probabilities
st_caption <- 'Log of relative participation probabilities against leisure'
kable(mt_prob_rela_m2leisure_log, caption=st_caption) %>% kable_styling_fc()
```

Third, construct the estimation input matrices. If there are M categories and T time periods, there are $M \times T$ observations (rows). Flatten in the same way as for outcomes. There are M indicator vectors and 1 wage vector for data inputs:

- α is a fixed effect that is m-specific, so we have as Data, M indicator vectors.
- The wage variable is shared by all, so a single data column

```
# Regression input matrix
mt_prob_ols_input <- matrix(data=NA, nrow=it_T*it_M, ncol=it_M+1)
colnames(mt_prob_ols_input) <- paste0('m', seq(1,dim(mt_prob_ols_input)[2]))
rownames(mt_prob_ols_input) <- paste0('rowMcrsT', seq(1,dim(mt_prob_ols_input)[1]))
# Generate index position in ols input matrix for M indicators
mt_m_indicators <- matrix(data=NA, nrow=it_T*it_M, ncol=it_M)
colnames(mt_prob_ols_input) <- paste0('m', seq(1,dim(mt_prob_ols_input)[2]))
rownames(mt_prob_ols_input) <- paste0('rowMcrsT', seq(1,dim(mt_prob_ols_input)[1]))
# loop over columns
for (it_indix in seq(1, it_M)) {
  # loop over rows
  for (it_rowMcrsT in seq(1, it_T*it_M)) {
    if ((it_rowMcrsT >= (it_indix-1)*(it_T) + 1) && it_rowMcrsT <= it_indix*(it_T)){
      mt_m_indicators[it_rowMcrsT, it_indix] <- 1
    } else {
      mt_m_indicators[it_rowMcrsT, it_indix] <- 0
    }
  }
}
# Indicators are the earlier columns
```

LHS=Log Probability Ratios (column 1); RHS=Indicators of occupations and wages, OLS inputs (other columns)

log_pm_over_po	m1	m2	wages
1.057688	1	0	1.883017
1.081184	1	0	1.940467
0.715186	1	0	1.045556
1.413265	0	1	1.528105
1.562261	0	1	1.892419
1.422806	0	1	1.551435

```
mt_prob_ols_input[, 1:it_M] <- mt_m_indicators
# Wage is the last column
mt_prob_ols_input[, it_M+1] <- matrix(data=mt_wages, nrow=it_T*it_M, ncol=1)
```

Fourth, combine left-hand-side and right-hand-side regression input data structures/

```
# Construct data structure
mt_all_inputs <- cbind(ar_prob_ols_output, mt_prob_ols_input)
colnames(mt_all_inputs)[1] <- 'log_pm_over_po'
colnames(mt_all_inputs)[dim(mt_all_inputs)[2]] <- 'wages'
tb_all_inputs <- as_tibble(mt_all_inputs)
# Show data
st_caption <- 'LHS=Log Probability Ratios (column 1); RHS=Indicators of occupations and wages, OLS i
kable(tb_all_inputs, caption=st_caption) %>% kable_styling_fc()
```

6.1.2.1.4 Estimate Wage Coefficients

Fifth, estimate the OLS equation, and compare estimate to true parameters.

```
# Regression
fit_ols_agg_prob <- lm(log_pm_over_po ~ . -1, data = tb_all_inputs)
summary(fit_ols_agg_prob)
```

```
##
## Call:
## lm(formula = log_pm_over_po ~ . - 1, data = tb_all_inputs)
##
## Residuals:
##          1          2          3          4          5          6
##  4.027e-17  2.276e-17 -6.303e-17 -6.481e-17 -1.631e-16  2.279e-16
##
## Coefficients:
##        Estimate Std. Error   t value Pr(>|t|)
## m1    2.876e-01  3.784e-16 7.599e+14   <2e-16 ***
## m2    7.883e-01  3.859e-16 2.043e+15   <2e-16 ***
## wages 4.090e-01  2.250e-16 1.818e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.721e-16 on 3 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.043e+32 on 3 and 3 DF,  p-value: < 2.2e-16
```

```
# alpha estimates
ar_coefficients <- fit_ols_agg_prob$coefficients
ar_alpha_esti <- ar_coefficients[1:(length(ar_coefficients)-1)]
fl_beta_esti <- ar_coefficients[length(ar_coefficients)]
# Compare estimates and true
```

Predicted probabilities based on estimates

	m0	m1	m2
t1	0.1251712	0.3604563	0.5143725
t2	0.1147084	0.3381795	0.5471121
t3	0.1390180	0.2842316	0.5767504

Compare differences in probability predictions based on estimates and true probabilities

	m0	m1	m2
t1	0	0	0
t2	0	0	0
t3	0	0	0

```r
print(paste0('ar_alpha_esti=', ar_alpha_esti))
```

```
## [1] "ar_alpha_esti=0.287577520124614" "ar_alpha_esti=0.788305135443806"
```

```r
print(paste0('ar_alpha=', ar_alpha))
```

```
## [1] "ar_alpha=0.287577520124614" "ar_alpha=0.788305135443807"
```

```r
print(paste0('fl_beta_esti=', fl_beta_esti))
```

```
## [1] "fl_beta_esti=0.4089769218117"
```

```r
print(paste0('fl_beta=', fl_beta))
```

```
## [1] "fl_beta=0.4089769218117"
```

```r
# Simulate given estimated parameters using earlier function
mt_prob_o_esti <- ffi_logit_prob_alpha_beta(ar_alpha_esti, fl_beta_esti, mt_wages)
# Results
st_caption <- 'Predicted probabilities based on estimates'
kable(mt_prob_o_esti, caption=st_caption) %>% kable_styling_fc()
```

```r
# Results
st_caption <- 'Compare differences in probability predictions based on estimates and true probabilit
kable(mt_prob_o_esti-mt_prob_o, caption=st_caption) %>% kable_styling_fc()
```

6.1.2.2 Logistic Choices with Time-specific Observables

In our example here, there are choice alternatives (m) and time periods (t). In terms of data inputs, in the prior example, we had on the Right

In the example in the prior section, the data structure included:

1. m-specific variables: indicators
2. m- and t-specific variables: wages

In the new data structure for this section, we have:

1. m-specific variables: indicators
2. m- and t-specific variables: wages
3. t-specific variables: characteristics that are homogeneous across groups, but varying over time. This will include both a trend variable capturing time patterns, as well as an observable that is different over time.

Suppose there is some information that impacts individual's willingness to stay at home. Note that leisure is the same as work from home. For example, there could be technological improvements that makes home-production less time-consuming, and we might have a variable that captures the efficiency of home-technology. Captured by a time trend, there might also be changes in social attitudes, which could be harder to measure.

Suppose we still have:

$$V_{itm} = \widehat{V}_{tm} + \epsilon_{itm} = \alpha_m + \beta \cdot \text{WAGE}_{tm} + \epsilon_{itm}$$

But now, for the leisure category we have:

$$V_{it0} = \widehat{V}_{t0} + \epsilon_{it0} = \alpha_0 + \phi \cdot t + \theta \cdot \text{HOMETECH}_t + \epsilon_{it0}$$

Because only the differences in value across choices matter, so we can continue with the prior normalization, in difference, we have:

$$V_{itm} - V_{it0} = (\alpha_m - \alpha_0) + (\beta \cdot \text{WAGE}_{tm} - \phi \cdot t - \theta \cdot \text{HOMETECH}_t) + (\epsilon_{itm} - \epsilon_{it0})$$

Note that:

- α_0 is not identifiable.
- If the HOMETECH or time variables are multiplied by negative one, the signs changes.
- Since only the differences in value matter, the coefficients for these variables represent the net changes on alternative value difference due to changes in time-trend or the HOMETECH variable.

6.1.2.2.1 Simulate Market Share We simulate market share data now, with information over several more periods, and the HOMETECH variable, along with the time trend.

First, simulate the data.

```
# set seed
set.seed(123)
# T periods, and M occupations (+1 leisure/home)
it_T <- 5
it_M <- 4
# define alpha and beta parameters
ar_alpha <- runif(it_M)*0.5
fl_beta <- runif(1)*0.25
# also negative time-trend from home category perspective, culturally increasingly accepting of work
fl_phi <- -runif(1)*0.50
# home-tech is negative from home category perspective, higher home-tech, more chance to work
fl_theta <- -runif(1)*1
# wage matrix, no wage for leisure
mt_wages <- matrix(runif(it_T*it_M) + 1, nrow=it_T, ncol=it_M)
# HOMETECH changes, random and sorted in ascending order, increasing over time
ar_hometech <- sort(runif(it_T))
colnames(mt_wages) <- paste0('m', seq(1,it_M))
rownames(mt_wages) <- paste0('t', seq(1,it_T))
# Define a probability assignment function
ffi_logit_prob2_alpha_beta <- function(ar_alpha, fl_beta, fl_phi, fl_theta, ar_hometech, mt_wages) {
  # Dimensions
  it_T <- dim(mt_wages)[1]
  it_M <- dim(mt_wages)[2]
  # Aggregate probabilities
  mt_prob_o <- matrix(data=NA, nrow=it_T, ncol=it_M+1)
  colnames(mt_prob_o) <- paste0('m', seq(0,it_M))
  rownames(mt_prob_o) <- paste0('t', seq(1,it_T))
  # Generate Probabilities
  for (it_t in seq(1, it_T)) {
    # get current period wages
    ar_wage_at_t <- mt_wages[it_t, ]
    # Value without shocks/errors, M+1
    ar_V_hat_at_t <- c(0, ar_alpha + fl_beta*ar_wage_at_t - fl_phi*it_t - fl_theta*ar_hometech[it_t]
```

Occupation aggregate participation probabilities across time: time-varying observables, time- and occupation-specific wages, occupation-specific intercepts; note reduction in m0, home-activity share over time

	m0	m1	m2	m3	m4
t1	0.1032335	0.2056532	0.2511476	0.1789233	0.2610423
t2	0.0932896	0.1891478	0.2441729	0.1906952	0.2826945
t3	0.0805830	0.1920307	0.2269798	0.2293885	0.2710180
t4	0.0633282	0.2043484	0.2589892	0.2137280	0.2596062
t5	0.0653460	0.1978795	0.2420864	0.2224408	0.2722473

```
    # Probabilities across M+1
    fl_prob_denominator <- sum(exp(ar_V_hat_at_t))
    ar_prob_at_t <- exp(ar_V_hat_at_t)/fl_prob_denominator
    # Fill in
    mt_prob_o[it_t,] <- ar_prob_at_t
  }
  return(mt_prob_o)
}
# Show probabilities
mt_prob_o2 <- ffi_logit_prob2_alpha_beta(ar_alpha, fl_beta, fl_phi, fl_theta, ar_hometech, mt_wages)
st_caption <- 'Occupation aggregate participation probabilities across time: time-varying observable
kable(mt_prob_o2, caption=st_caption) %>% kable_styling_fc()
```

```
# mt_prob_o
```

6.1.2.2.2 Create Regression Data Inputs Second, generate relative market shares of various work occupations as prior

```
mt_prob_rela_m2leisure2 <- matrix(data=mt_prob_o2[1:it_T, 2:(it_M+1)], nrow=it_T, ncol=it_M)
ar_prob_ols_output2 <- matrix(data=log(mt_prob_rela_m2leisure2/mt_prob_o2[1:it_T, 1]), nrow=it_T*it_
```

Third, construct the estimation input matrices as before. There are M indicator vectors, a time variable, a HOMETECH variable, and 1 wage vector for data inputs:

```
# Regression input matrix
mt_prob_ols_input2 <- matrix(data=NA, nrow=it_T*it_M, ncol=it_M+2+1)
colnames(mt_prob_ols_input2) <- paste0('m', seq(1,dim(mt_prob_ols_input2)[2]))
rownames(mt_prob_ols_input2) <- paste0('rowMcrsT', seq(1,dim(mt_prob_ols_input2)[1]))
# Generate index position in ols input matrix for M indicators
mt_m_indicators <- matrix(data=NA, nrow=it_T*it_M, ncol=it_M)
for (it_indix in seq(1, it_M)) {
  for (it_rowMcrsT in seq(1, it_T*it_M)) {
    if ((it_rowMcrsT >= (it_indix-1)*(it_T) + 1) && it_rowMcrsT <= it_indix*(it_T)){
      mt_m_indicators[it_rowMcrsT, it_indix] <- 1
    } else {
      mt_m_indicators[it_rowMcrsT, it_indix] <- 0
    }
  }
}
# Indicators are the earlier columns
mt_prob_ols_input2[, 1:it_M] <- mt_m_indicators
# Time variable column
ar_time <- matrix(seq(1,it_T), nrow=it_T*it_M, ncol=1)
mt_prob_ols_input2[, it_M+1] <- ar_time
# Home Tech Column
ar_hometech_mesh <- matrix(ar_hometech, nrow=it_T*it_M, ncol=1)
mt_prob_ols_input2[, it_M+2] <- ar_hometech_mesh
```

LHS=Log Probability Ratios (column 1); RHS=Indicators of occupations and other variables, OLS inputs (other columns)

log_pm_over_po	m1	m2	m3	m4	time	hometech	wages
0.6891981	1	0	0	0	1	0.1471136	1.892419
0.7068206	1	0	0	0	2	0.2891597	1.551435
0.8683678	1	0	0	0	3	0.5941420	1.456615
1.1714953	1	0	0	0	4	0.9022990	1.956833
1.1079617	1	0	0	0	5	0.9630242	1.453334
0.8890474	0	1	0	0	1	0.1471136	1.677571
0.9621685	0	1	0	0	2	0.2891597	1.572633
1.0355731	0	1	0	0	3	0.5941420	1.102925
1.4084555	0	1	0	0	4	0.9022990	1.899825
1.3095984	0	1	0	0	5	0.9630242	1.246088
0.5499640	0	0	1	0	1	0.1471136	1.042059
0.7149683	0	0	1	0	2	0.2891597	1.327921
1.0461296	0	0	1	0	3	0.5941420	1.954504
1.2163730	0	0	1	0	4	0.9022990	1.889539
1.2249646	0	0	1	0	5	0.9630242	1.692803
0.9276892	0	0	0	1	1	0.1471136	1.640507
1.1086584	0	0	0	1	2	0.2891597	1.994270
1.2128974	0	0	0	1	3	0.5941420	1.655706
1.4108350	0	0	0	1	4	0.9022990	1.708530
1.4270142	0	0	0	1	5	0.9630242	1.544066

```r
# Wage is the last column
mt_prob_ols_input2[, it_M+3] <- matrix(data=mt_wages, nrow=it_T*it_M, ncol=1)
```

Fourth, combine left-hand-side and right-hand-side regression input data structures/

```r
# Construct data structure
mt_all_inputs2 <- cbind(ar_prob_ols_output2, mt_prob_ols_input2)
colnames(mt_all_inputs2)[1] <- 'log_pm_over_po'
colnames(mt_all_inputs2)[dim(mt_all_inputs2)[2]-2] <- 'time'
colnames(mt_all_inputs2)[dim(mt_all_inputs2)[2]-1] <- 'hometech'
colnames(mt_all_inputs2)[dim(mt_all_inputs2)[2]] <- 'wages'
tb_all_inputs2 <- as_tibble(mt_all_inputs2)
# Show data
st_caption <- 'LHS=Log Probability Ratios (column 1); RHS=Indicators of occupations and other variab
kable(tb_all_inputs2, caption=st_caption) %>% kable_styling_fc()
```

6.1.2.2.3 Estimate Wage Coefficients, HOMETECH Effects and Time Trend

Similar to before, estimate with OLS.

```r
# Regression
fit_ols_agg_prob2 <- lm(log_pm_over_po ~ . -1, data = tb_all_inputs2)
summary(fit_ols_agg_prob2)
```

```
##
## Call:
## lm(formula = log_pm_over_po ~ . - 1, data = tb_all_inputs2)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.615e-16 -9.020e-17 -1.217e-17  9.375e-17  2.984e-16
##
## Coefficients:
##           Estimate Std. Error   t value Pr(>|t|)
```

```
## m1       1.438e-01  3.364e-16 4.274e+14   <2e-16 ***
## m2       3.942e-01  3.111e-16 1.267e+15   <2e-16 ***
## m3       2.045e-01  3.238e-16 6.315e+14   <2e-16 ***
## m4       4.415e-01  3.437e-16 1.284e+15   <2e-16 ***
## time     2.278e-02  1.666e-16 1.367e+14   <2e-16 ***
## hometech 5.281e-01  7.335e-16 7.200e+14   <2e-16 ***
## wages    2.351e-01  1.717e-16 1.370e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.851e-16 on 13 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 9.716e+31 on 7 and 13 DF,  p-value: < 2.2e-16
```

```
# alpha estimates
ar_coefficients2 <- fit_ols_agg_prob2$coefficients
ar_alpha_esti2 <- ar_coefficients2[1:(length(ar_coefficients2)-3)]
# time -1 because of the sign structure
fl_phi_esti2 <- -1*ar_coefficients2[length(ar_coefficients2)-2]
# time -1 because of the sign structure
fl_theta_esti2 <- -1*ar_coefficients2[length(ar_coefficients2)-1]
fl_beta_esti2 <- ar_coefficients2[length(ar_coefficients2)]
# Compare estimates and true
print(paste0('ar_alpha_esti2=', ar_alpha_esti2))
```

```
## [1] "ar_alpha_esti2=0.143788760062308" "ar_alpha_esti2=0.394152567721904"
## [3] "ar_alpha_esti2=0.20448846090585"  "ar_alpha_esti2=0.441508702002466"
```

```
print(paste0('ar_alpha=', ar_alpha))
```

```
## [1] "ar_alpha=0.143788760062307" "ar_alpha=0.394152567721903"
## [3] "ar_alpha=0.20448846090585"  "ar_alpha=0.441508702002466"
```

```
print(paste0('fl_theta_esti2=', fl_theta_esti2))
```

```
## [1] "fl_theta_esti2=-0.528105488047005"
```

```
print(paste0('fl_theta=', fl_theta))
```

```
## [1] "fl_theta=-0.528105488047004"
```

```
print(paste0('fl_phi_esti2=', fl_phi_esti2))
```

```
## [1] "fl_phi_esti2=-0.0227782496949655"
```

```
print(paste0('fl_phi=', fl_phi))
```

```
## [1] "fl_phi=-0.0227782496949658"
```

```
print(paste0('fl_beta_esti2=', fl_beta_esti2))
```

```
## [1] "fl_beta_esti2=0.235116821073461"
```

```
print(paste0('fl_beta=', fl_beta))
```

```
## [1] "fl_beta=0.235116821073461"
```

```
# Simulate given estimated parameters using earlier function
mt_prob_o2_esti <- ffi_logit_prob2_alpha_beta(ar_alpha_esti2, fl_beta_esti2, fl_phi_esti2, fl_theta_
# Results
st_caption <- 'Predicted probabilities based on estimates'
kable(mt_prob_o2_esti, caption=st_caption) %>% kable_styling_fc()
```

```
# Results
st_caption <- 'Compare differences in probability predictions based on estimates and true probabilit
```

Predicted probabilities based on estimates

	m0	m1	m2	m3	m4
t1	0.1032335	0.2056532	0.2511476	0.1789233	0.2610423
t2	0.0932896	0.1891478	0.2441729	0.1906952	0.2826945
t3	0.0805830	0.1920307	0.2269798	0.2293885	0.2710180
t4	0.0633282	0.2043484	0.2589892	0.2137280	0.2596062
t5	0.0653460	0.1978795	0.2420864	0.2224408	0.2722473

Compare differences in probability predictions based on estimates and true probabilities

	m0	m1	m2	m3	m4
t1	0	0	0	0	0
t2	0	0	0	0	0
t3	0	0	0	0	0
t4	0	0	0	0	0
t5	0	0	0	0	0

```
kable(mt_prob_o2_esti-mt_prob_o2, caption=st_caption) %>% kable_styling_fc()
```

6.1.2.3 Logistic Choices and Multiple Types of Workers

In the prior two examples, we had the same type of worker, choosing among multiple occupations. There could be different types of workers, unskilled and skilled workers, both of whom can choose among the same set of job possibilities. If the two types of workers do not share any parameters in the occupation-choice problems, then we can estimate their problems separately. However, perhaps on the stay-at-home front, they share the same time trend, which captures overall cultural changes to work vs not-work. And perhaps the wage parameter is the same. On other fronts, they are different with different parameters as well as data inputs.

So we have:

$$V_{istm} - V_{ist0} = (\alpha_{sm} - \alpha_{s0}) + (\beta \cdot \text{WAGE}_{stm} - \phi \cdot t - \theta_s \cdot \text{HOMETECH}_{st}) + (\epsilon_{istm} - \epsilon_{ist0})$$

In the new data structure for this section, we have:

1. k- and m-specific variables: indicators
2. k- and m- and t-specific variable: wages
3. t-specific variable: dates
4. k- and t-specific variable: hometech

6.1.2.3.1 Simulate Market Share In the simulation example, store all K types and T time periods as rows, and the M occupational types as columns.

```
# set seed
set.seed(123)
# K skill levels, T periods, and M occupations (+1 leisure/home)
it_K <- 2
it_T <- 3
it_M <- 4
# define alpha and beta parameters
mt_alpha <- matrix(runif(it_K*it_M)*0.5, nrow=it_K, ncol=it_M)
colnames(mt_alpha) <- paste0('m', seq(1,dim(mt_alpha)[2]))
rownames(mt_alpha) <- paste0('k', seq(1,dim(mt_alpha)[1]))
fl_beta <- runif(1)*0.25
# also negative time-trend from home category perspective, culturally increasingly accepting of work
fl_phi <- -runif(1)*0.50
# home-tech is negative from home category perspective, higher home-tech, more chance to work
```

```
ar_theta <- -runif(it_K)*1
# wage matrix, no wage for leisure
mt_wages <- matrix(runif(it_K*it_T*it_M) + 1, nrow=it_K*it_T, ncol=it_M)
colnames(mt_wages) <- paste0('m', seq(1,it_M))
rownames(mt_wages) <- paste0('kxt', seq(1,it_K*it_T))
# HOMETECH changes, random and sorted in ascending order, increasing over time, specific to each k
mt_hometech <- sapply(seq(1,it_K), function(i){sort(runif(it_T))})
colnames(mt_hometech) <- paste0('k', seq(1,dim(mt_hometech)[2]))
rownames(mt_hometech) <- paste0('t', seq(1,dim(mt_hometech)[1]))
# Define a probability assignment function
ffi_logit_prob3_alpha_beta <- function(it_K, it_T, it_M,
  mt_alpha, fl_beta, fl_phi, ar_theta, mt_hometech, mt_wages) {

  # Aggregate probabilities
  mt_prob_o <- matrix(data=NA, nrow=it_K*it_T, ncol=it_M+3)
  colnames(mt_prob_o) <- c('skillgroup', 'time', paste0('m', seq(0,it_M)))
  rownames(mt_prob_o) <- paste0('kxt', seq(1,it_K*it_T))
  # Generate Probabilities
  it_kxt_ctr <- 0
  for (it_k in seq(1, it_K)) {
    for (it_t in seq(1, it_T)) {
      # Counter updating
      it_kxt_ctr <- it_kxt_ctr + 1
      # get current period wages
      ar_wage_at_t <- mt_wages[it_kxt_ctr, ]
      # Value without shocks/errors, M+1
      ar_V_hat_at_t <- c(0, mt_alpha[it_k,] + fl_beta*ar_wage_at_t - fl_phi*it_t - ar_theta[it_k]*mt
      # Probabilities across M+1
      fl_prob_denominator <- sum(exp(ar_V_hat_at_t))
      ar_prob_at_t <- exp(ar_V_hat_at_t)/fl_prob_denominator
      # Fill in
      mt_prob_o[it_kxt_ctr,1] <- it_k
      mt_prob_o[it_kxt_ctr,2] <- it_t
      mt_prob_o[it_kxt_ctr,3:dim(mt_prob_o)[2]] <- ar_prob_at_t
      # row name
      rownames(mt_prob_o)[it_kxt_ctr] <- paste0('rk', it_k, 't', it_t)
    }
  }
  return(mt_prob_o)

}
# Show probabilities
mt_prob_o3_full <- ffi_logit_prob3_alpha_beta(it_K, it_T, it_M,
  mt_alpha, fl_beta, fl_phi, ar_theta, mt_hometech, mt_wages)
# Selected only probability columns
mt_prob_o3 <- mt_prob_o3_full[, 3:dim(mt_prob_o3_full)[2]]
st_caption <- 'Occupation aggregate participation probabilities across time: skill- and time-varying
kable(mt_prob_o3_full, caption=st_caption) %>% kable_styling_fc()
```

```
# mt_prob_o
```

6.1.2.3.2 Create Regression Data Inputs Second, generate relative market shares of various work occupations as prior.

```
mt_prob_rela_m2leisure3 <- mt_prob_o3[, 2:(it_M+1)]/mt_prob_o3[, 1]
# mt_log_prob_rela_m2leisure3 <- log(mt_prob_rela_m2leisure3/mt_prob_o3[, 1])
# kable(log(mt_prob_rela_m2leisure3/mt_prob_o3[, 1]), caption=st_caption) %>% kable_styling_fc()
```

Occupation aggregate participation probabilities across time: skill- and time-varying observables, skill and time- and occupation-specific wages, skill and occupation-specific intercepts; common wage coefficient and time coefficient; note reduction in m0, home-activity share over time

	skillgroup	time	m0	m1	m2	m3	m4
rk1t1	1	1	0.0887175	0.1995146	0.2020237	0.2757014	0.2340428
rk1t2	1	2	0.0646539	0.1984822	0.2223035	0.2803052	0.2342551
rk1t3	1	3	0.0358582	0.1975615	0.2339696	0.2909961	0.2416147
rk2t1	2	1	0.0942785	0.2435583	0.2481846	0.1610703	0.2529083
rk2t2	2	2	0.0780946	0.2411637	0.2669887	0.1673420	0.2464110
rk2t3	2	3	0.0572989	0.2348487	0.2807791	0.1643585	0.2627147

Third, construct the estimation input matrices as before. There are $K \times M$ indicator vectors, a time variable, a HOMETECH variable, and 1 wage vector for data inputs. The indicator vectors are specific interecept for each type of worker (skilled or not) and for each occupation. Note:

1. k and m specific indicators
2. k specific hometech coefficients
3. common time coefficient
4. common coefficient for wage

```
# Regression input matrix
# it_K*it_M+it_K+1+1: 1. it_K*it_M for all indicators; 2. it_k for HOMETECH; 3. 1 for time; 4. 1 for
mt_prob_ols_input3 <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=it_K*it_M+it_K+1+1)
colnames(mt_prob_ols_input3) <- paste0('m', seq(1,dim(mt_prob_ols_input3)[2]))
rownames(mt_prob_ols_input3) <- paste0('rowkxMxT', seq(1,dim(mt_prob_ols_input3)[1]))

# LHS variable meshed store
ar_prob_ols_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=1)
# RHS variables meshed store
mt_indi_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=it_K*it_M)
ar_time_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=1)
ar_wage_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=1)
mt_hometech_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=it_K)

# Loop over columns
it_kxm_ctr <- 0
for (it_r_k in seq(1, it_K)) {
  for (it_r_m in seq(1, it_M)) {
    # Column counter
    it_kxm_ctr <- it_kxm_ctr + 1

    # Update name of indicator column
    colnames(mt_prob_ols_input3)[it_kxm_ctr] <- paste0('i_k', it_r_k, 'm', it_r_m)
    # Start and end row for the indictor function mt_indi_mesh_kmt
    it_indi_str_row <- (it_kxm_ctr-1)*it_T
    it_indi_end_row <- (it_kxm_ctr+0)*it_T

    # Update names of the hometech column
    colnames(mt_prob_ols_input3)[it_K*it_M + it_r_k] <- paste0('hometech_k', it_r_k)
    # Start and end row for the indictor function mt_hometech_mesh_kmt
    it_hometech_str_row <- (it_r_k-1)*it_M*it_T
    it_hometech_end_row <- (it_r_k+0)*it_M*it_T

    # Loop over rows
    it_rowKxT <- 0
    it_rowKxTxM <- 0
    for (it_k in seq(1, it_K)) {
```

```
      for (it_m in seq(1, it_M)) {
        for (it_t in seq(1, it_T)) {

          # KxT group counter
          it_rowKxT <- it_T*(it_k-1) + it_t

          # Row counter
          it_rowKxTxM <- it_rowKxTxM + 1

          # Indicator matrix, heterogeneous by K and M
          if ((it_rowKxTxM > it_indi_str_row) && (it_rowKxTxM <= it_indi_end_row)){
            mt_indi_mesh_kmt[it_rowKxTxM, it_kxm_ctr] <- 1
          } else {
            mt_indi_mesh_kmt[it_rowKxTxM, it_kxm_ctr] <- 0
          }

          # HOMETECH specific matrix, heterogeneous by K, homogeneous by M
          if (it_r_m == 1) {
            if ((it_rowKxTxM > it_hometech_str_row) && (it_rowKxTxM <= it_hometech_end_row)){
              mt_hometech_mesh_kmt[it_rowKxTxM, it_r_k] <- mt_hometech[it_t, it_k]
            } else {
              mt_hometech_mesh_kmt[it_rowKxTxM, it_r_k] <- 0
            }
          }

          # Only need to do once, homogeneous across K, M and T
          if (it_kxm_ctr == 1) {
            rownames(mt_prob_ols_input3)[it_rowKxTxM] <- paste0('ik', it_k, 'm', it_m, 't', it_t)
            # RHS
            ar_time_mesh_kmt[it_rowKxTxM] <- it_t
            ar_wage_mesh_kmt[it_rowKxTxM] <- mt_wages[it_rowKxT, it_m]
            # LHS, log of probability
            ar_prob_ols_mesh_kmt[it_rowKxTxM] <- log(mt_prob_rela_m2leisure3[it_rowKxT, it_m])
          }
        }
      }
    }
  }
}
# Indicators are the earlier columns
mt_prob_ols_input3[, 1:(it_K*it_M)] <- mt_indi_mesh_kmt
# Time variable column
it_col_start <- (it_K*it_M) + 1
it_col_end <- it_col_start + it_K - 1
mt_prob_ols_input3[, it_col_start:it_col_end] <- mt_hometech_mesh_kmt
# Home Tech Column
mt_prob_ols_input3[, it_col_end+1] <- ar_time_mesh_kmt
# Wage is the last column
mt_prob_ols_input3[, it_col_end+2] <- ar_wage_mesh_kmt
# kable out
# kable(mt_prob_ols_input3) %>% kable_styling_fc()
```

Fourth, combine left-hand-side and right-hand-side regression input data structures/

```
# Construct data structure
mt_all_inputs3 <- cbind(ar_prob_ols_mesh_kmt, mt_prob_ols_input3)
colnames(mt_all_inputs3)[1] <- 'log_pm_over_po'
colnames(mt_all_inputs3)[dim(mt_all_inputs3)[2]-1] <- 'time'
```

```r
colnames(mt_all_inputs3)[dim(mt_all_inputs3)[2]] <- 'wages'
tb_all_inputs3 <- as_tibble(mt_all_inputs3)
# Show data
st_caption <- 'LHS=Log Probability Ratios (column 1); RHS=Indicator, time-trends and data with diffe
kable(tb_all_inputs3, caption=st_caption) %>% kable_styling_fc_wide()
```

LHS=Log Probability Ratios (column 1); RHS=Indicator, time-trends and data with different assumptions on whether coefficients with be skill specific (hometech), occuption and skill specific (indictor), or homogeneous across skills and occupations (time and wage), OLS inputs (other columns)

log_pm_over_po	i_k1m1	i_k1m2	i_k1m3	i_k1m4	i_k2m1	i_k2m2	i_k2m3	i_k2m4	hometech_k1	hometech_k2	time	wages
0.8104303	1	0	0	0	0	0	0	0	0.2164079	0.0000000	1	1.677571
1.1216510	1	0	0	0	0	0	0	0	0.3181810	0.0000000	2	1.572633
1.7064781	1	0	0	0	0	0	0	0	0.7584595	0.0000000	3	1.102925
0.8229277	0	1	0	0	0	0	0	0	0.2164079	0.0000000	1	1.327921
1.2349948	0	1	0	0	0	0	0	0	0.3181810	0.0000000	2	1.954504
1.8756195	0	1	0	0	0	0	0	0	0.7584595	0.0000000	3	1.889539
1.1338609	0	0	1	0	0	0	0	0	0.2164079	0.0000000	1	1.655706
1.4668305	0	0	1	0	0	0	0	0	0.3181810	0.0000000	2	1.708530
2.0937381	0	0	1	0	0	0	0	0	0.7584595	0.0000000	3	1.544066
0.9700465	0	0	0	1	0	0	0	0	0.2164079	0.0000000	1	1.963024
1.2873623	0	0	0	1	0	0	0	0	0.3181810	0.0000000	2	1.902299
1.9077727	0	0	0	1	0	0	0	0	0.7584595	0.0000000	3	1.690705
0.9491036	0	0	0	0	1	0	0	0	0.0000000	0.1428000	1	1.899825
1.1275553	0	0	0	0	1	0	0	0	0.0000000	0.2316258	2	1.246088
1.4106597	0	0	0	0	1	0	0	0	0.0000000	0.4145463	3	1.042059
0.9679200	0	0	0	0	0	1	0	0	0.0000000	0.1428000	1	1.692803
1.2292855	0	0	0	0	0	1	0	0	0.0000000	0.2316258	2	1.640507
1.5892864	0	0	0	0	0	1	0	0	0.0000000	0.4145463	3	1.994270
0.5355882	0	0	0	0	0	0	1	0	0.0000000	0.1428000	1	1.594142
0.7621188	0	0	0	0	0	0	1	0	0.0000000	0.2316258	2	1.289160
1.0537680	0	0	0	0	0	0	1	0	0.0000000	0.4145463	3	1.147114
0.9867739	0	0	0	0	0	0	0	1	0.0000000	0.1428000	1	1.795467
1.1490801	0	0	0	0	0	0	0	1	0.0000000	0.2316258	2	1.024614
1.5227867	0	0	0	0	0	0	0	1	0.0000000	0.4145463	3	1.477796

6.1.2.3.3 Estimate Wage and Time Coefficients, Skill Specific Fixed Effects and HOMETECH Coefficient

Similar to before, estimate with OLS, and show that predictions based on OLS estimates match with the true parameters. Predicted probabilities are the same as observed probabilities.

```r
# Regression
fit_ols_agg_prob3 <- lm(log_pm_over_po ~ . -1, data = tb_all_inputs3)
summary(fit_ols_agg_prob3)
```

```
##
## Call:
## lm(formula = log_pm_over_po ~ . - 1, data = tb_all_inputs3)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.827e-16 -9.177e-17  6.850e-18  7.622e-17  4.148e-16
##
## Coefficients:
##              Estimate Std. Error   t value Pr(>|t|)
## i_k1m1      1.438e-01  3.474e-16 4.140e+14   <2e-16 ***
## i_k1m2      2.045e-01  3.902e-16 5.241e+14   <2e-16 ***
## i_k1m3      4.702e-01  3.762e-16 1.250e+15   <2e-16 ***
## i_k1m4      2.641e-01  4.108e-16 6.428e+14   <2e-16 ***
## i_k2m1      3.942e-01  3.479e-16 1.133e+15   <2e-16 ***
## i_k2m2      4.415e-01  4.091e-16 1.079e+15   <2e-16 ***
## i_k2m3      2.278e-02  3.397e-16 6.705e+13   <2e-16 ***
## i_k2m4      4.462e-01  3.537e-16 1.262e+15   <2e-16 ***
## hometech_k1 9.568e-01  6.264e-16 1.527e+15   <2e-16 ***
## hometech_k2 4.533e-01  1.352e-15 3.353e+14   <2e-16 ***
## time        2.283e-01  1.775e-16 1.286e+15   <2e-16 ***
## wages       1.379e-01  1.818e-16 7.581e+14   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.962e-16 on 12 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 8.731e+31 on 12 and 12 DF,  p-value: < 2.2e-16
```

```
# Parse coefficients
ls_coefficients_esti <- vector(mode = "list", length = 0)
ls_coefficients_true <- vector(mode = "list", length = 0)
ar_coefficients3 <- fit_ols_agg_prob3$coefficients
it_col_end <- 0
for (it_coef_grp in c(1,2,3,4)) {
  it_col_str <- it_col_end + 1
  if (it_coef_grp == 1) {
    it_grp_coef_cnt <- (it_K*it_M)
    st_coef_name <- 'indi_km'
    ar_esti_true <- mt_alpha
    it_sign <- +1
  } else if (it_coef_grp == 2) {
    it_grp_coef_cnt <- it_K
    st_coef_name <- 'hometech_k'
    ar_esti_true <- ar_theta
    it_sign <- -1
  } else if (it_coef_grp == 3) {
    it_grp_coef_cnt <- 1
    st_coef_name <- 'time'
    ar_esti_true <- fl_phi
    it_sign <- -1
  } else if (it_coef_grp == 4) {
    it_grp_coef_cnt <- 1
    st_coef_name <- 'wage'
    ar_esti_true <- fl_beta
    it_sign <- +1
  }

  # select
  it_col_end <- it_col_end + it_grp_coef_cnt
  ar_esti_curgroup <- ar_coefficients3[it_col_str:it_col_end]
  if (it_coef_grp == 1) {
      ar_esti_curgroup <- t(matrix(data=ar_esti_curgroup, nrow=it_M, ncol=it_K))
  }
  # store
  ls_coefficients_esti[[st_coef_name]] <- it_sign*ar_esti_curgroup
  ls_coefficients_true[[st_coef_name]] <- ar_esti_true
}

# Compare estimates and true
print(ls_coefficients_esti)
```

```
## $indi_km
##           [,1]      [,2]       [,3]      [,4]
## [1,] 0.1437888 0.2044885 0.47023364 0.2640527
## [2,] 0.3941526 0.4415087 0.02277825 0.4462095
##
## $hometech_k
## hometech_k1 hometech_k2
##  -0.9568333  -0.4533342
##
```

Predicted probabilities based on estimates

	skillgroup	time	m0	m1	m2	m3	m4
rk1t1	1	1	0.0887175	0.1995146	0.2020237	0.2757014	0.2340428
rk1t2	1	2	0.0646539	0.1984822	0.2223035	0.2803052	0.2342551
rk1t3	1	3	0.0358582	0.1975615	0.2339696	0.2909961	0.2416147
rk2t1	2	1	0.0942785	0.2435583	0.2481846	0.1610703	0.2529083
rk2t2	2	2	0.0780946	0.2411637	0.2669887	0.1673420	0.2464110
rk2t3	2	3	0.0572989	0.2348487	0.2807791	0.1643585	0.2627147

Compare differences in probability predictions based on estimates and true probabilities

	skillgroup	time	m0	m1	m2	m3	m4
rk1t1	0	0	0	0	0	0	0
rk1t2	0	0	0	0	0	0	0
rk1t3	0	0	0	0	0	0	0
rk2t1	0	0	0	0	0	0	0
rk2t2	0	0	0	0	0	0	0
rk2t3	0	0	0	0	0	0	0

```
## $time
##       time
## -0.2283074
##
## $wage
##     wages
## 0.1378588
```

```r
print(ls_coefficients_true)
```

```
## $indi_km
##           m1        m2         m3        m4
## k1 0.1437888 0.2044885 0.47023364 0.2640527
## k2 0.3941526 0.4415087 0.02277825 0.4462095
##
## $hometech_k
## [1] -0.9568333 -0.4533342
##
## $time
## [1] -0.2283074
##
## $wage
## [1] 0.1378588
```

```r
# Simulate given estimated parameters using earlier function
mt_prob_o3_full_esti <- ffi_logit_prob3_alpha_beta(it_K, it_T, it_M,
  ls_coefficients_esti[['indi_km']],
  ls_coefficients_esti[['wage']],
  ls_coefficients_esti[['time']],
  ls_coefficients_esti[['hometech_k']],
  mt_hometech, mt_wages)
# Results
st_caption <- 'Predicted probabilities based on estimates'
kable(mt_prob_o3_full_esti, caption=st_caption) %>% kable_styling_fc()
```

```r
# Results
st_caption <- 'Compare differences in probability predictions based on estimates and true probabilit
kable(mt_prob_o3_full_esti-mt_prob_o3_full, caption=st_caption) %>% kable_styling_fc()
```

6.1.3 Prices from Aggregate Shares in Logistic Choice Model

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

6.1.3.1 Observed Shares and Wages

In Estimate Logistic Choice Model with Aggregate Shares, we described and developed the multinomial logistic model with choice over alternatives.

The scenario here is that we have estimated the logistic choice model using data from some prior years. We know that for another set of years, model parameters, and in particular, the effect of alternative-specific prices are the same. We have market share information, as well as all other observables from other years, however, no price for alternatives. The goal is to use existing model parameters and the aggregate shares to back out the alternative-specific prices that would explain the data.

We know that values for choice-specific alternatives, with p_m as the alternative-specific price/wage, are:

$$V_{im} = \alpha_m + \beta \cdot w_m + \epsilon_{im}$$

Choice probabilities are functions of wages. The probability that individual i chooses alternative m is:

$$P(o = m) = P_m = \frac{\exp(\alpha_m + \beta \cdot w_m)}{1 + \sum_{\widehat{m}=1}^{M} \exp(\alpha_{\widehat{m}} + \beta \cdot w_{\widehat{m}})}$$

We observe $P(o = m)$, we know α_m across alternatives, and we know already β. We do not know w_m across alternatives. Fitting means adjusting $\{w_m\}_{m=1}^{M}$ to fit $\{P(o = m)\}_{m=1}^{M}$ observed.

Moving terms around and cross multiplying, we have:

$$\begin{aligned}
\exp(\alpha_m + \beta \cdot w_m) &= P_m + P_m \sum_{\widehat{m}=1}^{M} \exp(\alpha_{\widehat{m}} + \beta \cdot w_{\widehat{m}}) \\
e^{\alpha_m} \exp(\beta \cdot w_m) &= P_m + P_m \sum_{\widehat{m}=1}^{M} e^{\alpha_{\widehat{m}}} \exp(\beta \cdot w_{\widehat{m}})
\end{aligned}$$

This can be viewed as a linear equation, let $\exp(\alpha_m) = A_m$, which is known, and let $\exp(\beta \cdot w_m) = \theta_m$, which is a function of the known parameter β and unknown price w_m. We have:

$$P_m = A_m \cdot \theta_m - P_m \sum_{\widehat{m}=1}^{M} A_{\widehat{m}} \cdot \theta_{\widehat{m}}$$

Suppose $M = 3$, and we label the categories as m, r, a. Note that implicitly we have an outside option category that we are normalizing against. We have then:

$$\begin{aligned}
P_m &= +A_m (1 - P_m) \cdot \theta_m - A_r P_m \cdot \theta_r - A_a P_m \cdot \theta_a \\
P_r &= -A_m P_r \cdot \theta_m + A_r (1 - P_r) \cdot \theta_r - A_a P_r \cdot \theta_a \\
P_a &= -A_m P_a \cdot \theta_m - A_r P_a \cdot \theta_r + A_a (1 - P_a) \cdot \theta_a
\end{aligned}$$

Above, we have a system of equations, with three unknown parameters. Regressing the left-hand-side aggregate share vectors against the matrix of right-hand values composed of A and P values generates θ values, which then map one to one to the wages.

An important issue to note is that the "backing-out" procedure does not work with any arbitrary probabilities. Note that $\exp(\beta \cdot w_m) > 0$. The estimated unknown θ_m will indeed be positive if the probabilities for example sum up to less than 1, however if the probabilities on the left hand side sum to greater than 1, then $\theta_m < 0$ is possible, which leads to no solutions.

Wages across occupations

	m1	m2	m3
t1	1.940467	1.045556	1.528105

Participation probabilities across categories

	m0	m1	m2	m3
t1	0.0506663	0.374767	0.2805663	0.2940004

Additionally, note that while the procedure here is correct, we can also obtain the wages that can explaine observed probabilities simply by using the log-odds ratio equations. Doing that requires first computing the appropriate log-odds, which requires positive probabilities on the outside option category.

6.1.3.2 Simulate Market Share

In this section, we now simulate the above model, with $M = 3$ and data over three periods, and estimate α and β via OLS. Note that $m = 0$ is leisure. This is identical what is what the simulate market share section from Estimate Logistic Choice Model with Aggregate Shares.

First, wages across alternatives.

```
# set seed
set.seed(123)
# T periods, and M occupations (+1 leisure)
it_M <- 3
# define alpha and beta parameters
ar_alpha <- runif(it_M) + 0
fl_beta <- runif(1) + 0
# wage matrix, no wage for leisure
mt_wages <- matrix(runif(1 * it_M) + 1, nrow = 1, ncol = it_M)
colnames(mt_wages) <- paste0("m", seq(1, it_M))
rownames(mt_wages) <- paste0("t", seq(1, 1))
# Show wages
st_caption <- "Wages across occupations"
kable(mt_wages, caption = st_caption) %>% kable_styling_fc()
```

Second, shares across alternatives (and outside option).

```
# Define a probability assignment function
ffi_logit_prob_alpha_beta_1t <- function(ar_alpha, fl_beta, mt_wages) {
  # Dimensions
  it_M <- dim(mt_wages)[2]
  # Aggregate probabilities
  mt_prob_o <- matrix(data = NA, nrow = 1, ncol = it_M + 1)
  colnames(mt_prob_o) <- paste0("m", seq(0, it_M))
  rownames(mt_prob_o) <- paste0("t", seq(1, 1))
  # Generate Probabilities
  # Value without shocks/errors, M+1
  ar_V_hat <- c(0, ar_alpha + fl_beta * mt_wages[1, ])
  # Probabilities across M+1
  mt_prob_o[1, ] <- exp(ar_V_hat) / sum(exp(ar_V_hat))
  return(mt_prob_o)
}
# Show probabilities
ar_prob_o <- ffi_logit_prob_alpha_beta_1t(ar_alpha, fl_beta, mt_wages)
st_caption <- "Participation probabilities across categories"
kable(ar_prob_o, caption = st_caption) %>% kable_styling_fc()
```

RHS fit wages matrix

	mValNoWage1	mValNoWage2	mValNoWage3
mProb1	0.8335569	-0.8243618	-0.5641281
mProb2	-0.3740494	1.5825131	-0.4223301
mProb3	-0.3919596	-0.6467025	1.0627249

6.1.3.3 Create Inputs for Wages Fit/Estimation

See the linearized structure above, where the LHS is a vector of non-outside-option alternative probabilities. And the RHS is M by M, where each row is multiplying by a different occupation-specific probability, and each column is a different non-wage component of the category specific value (without the error term).

Create the right-hand-side matrix.

```r
# A Matrix with share from 1:M columns
mt_rhs_input <- matrix(data = NA, nrow = it_M, ncol = it_M)
colnames(mt_rhs_input) <- paste0("mValNoWage", seq(1, it_M))
rownames(mt_rhs_input) <- paste0("mProb", seq(1, it_M))

# Loop over rows
for (it_m_r in seq(1, it_M)) {
  # +1 to skip the outside-option category
  P_m <- ar_prob_o[it_m_r + 1]
  # Loop over columns
  for (it_m_c in seq(1, it_M)) {
    # Column value for non-wage component of the category-specific value
    A_m <- exp(ar_alpha[it_m_c])
    # Diagonal or not
    if (it_m_r == it_m_c) {
      fl_rhs_val <- A_m * (1 - P_m)
    } else {
      fl_rhs_val <- -1 * A_m * P_m
    }
    # Fill value
    mt_rhs_input[it_m_r, it_m_c] <- fl_rhs_val
  }
}

# Show rhs matrix
st_caption <- "RHS fit wages matrix"
kable(mt_rhs_input, caption = st_caption) %>% kable_styling_fc()
```

Add in the LHS probability column.

```r
# Construct data structure, LHS and RHS, LHS first oclumn
mt_all_inputs <- cbind(ar_prob_o[2:length(ar_prob_o)], mt_rhs_input)
colnames(mt_all_inputs)[1] <- "prob_o"
tb_all_inputs <- as_tibble(mt_all_inputs)
# Show data
st_caption <- "col1=prob in non-outside options; other columns, RHS matrix"
kable(tb_all_inputs, caption=st_caption) %>% kable_styling_fc()
```

6.1.3.4 Solve/Estimate for Wages that Explain Shares

Given the RHS matrix just generated estimate the wages, and check that the match the wages used to simulate the probabilities.

col1=prob in non-outside options; other columns, RHS matrix

prob_o	mValNoWage1	mValNoWage2	mValNoWage3
0.3747670	0.8335569	-0.8243618	-0.5641281
0.2805663	-0.3740494	1.5825131	-0.4223301
0.2940004	-0.3919596	-0.6467025	1.0627249

```
# Regression
fit_ols_agg_prob_to_wages <- lm(prob_o ~ . - 1, data = tb_all_inputs)
summary(fit_ols_agg_prob_to_wages)
```

```
##
## Call:
## lm(formula = prob_o ~ . - 1, data = tb_all_inputs)
##
## Residuals:
## ALL 3 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## mValNoWage1    5.548        NaN     NaN      NaN
## mValNoWage2    2.517        NaN     NaN      NaN
## mValNoWage3    3.855        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 3 and 0 DF,  p-value: NA
```

```
# alpha estimates
ar_coefficients <- fit_ols_agg_prob_to_wages$coefficients
ar_wages_esti <- log(ar_coefficients)/fl_beta
# Compare estimates and true
print(paste0("ar_coefficients=", ar_coefficients))
```

```
## [1] "ar_coefficients=5.54816023677087" "ar_coefficients=2.51744522035287"
## [3] "ar_coefficients=3.85489489133316"
```

```
print(paste0("ar_wages_esti=", ar_wages_esti))
```

```
## [1] "ar_wages_esti=1.94046728429384" "ar_wages_esti=1.04555649938993"
## [3] "ar_wages_esti=1.528105488047"
```

```
print(paste0("mt_wages=", mt_wages))
```

```
## [1] "mt_wages=1.94046728429385" "mt_wages=1.04555649938993" "mt_wages=1.528105488047"
```

6.2 Quantile Regression

6.2.1 Quantile Regression Basics

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

6.2.1.1 Estimate Mean and Conditional Quantile Coefficients using mtcars dataset

Here, we conduct tests for using the quantreg package, using the built-in mtcars dataset.

First, estimate the mean (OLS) regression:

```r
fit_mean <- lm(mpg ~ disp + hp + factor(am) + factor(vs), data = mtcars)
summary(fit_mean)
```

```
##
## Call:
## lm(formula = mpg ~ disp + hp + factor(am) + factor(vs), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8493 -1.9540  0.4649  1.4002  5.4239
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 25.664864   2.976827   8.622 4.23e-09 ***
## disp        -0.008676   0.010053  -0.863  0.39602
## hp          -0.040770   0.014085  -2.895  0.00759 **
## factor(am)1  4.624025   1.498995   3.085  0.00479 **
## factor(vs)1  1.454075   1.709450   0.851  0.40275
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.801 on 26 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.8187, Adjusted R-squared:  0.7908
## F-statistic: 29.35 on 4 and 26 DF,  p-value: 2.664e-09
```

Now estimate conditional quantile regressions (not that this remains linear) at various quantiles, standard error obtained via bootstrap. Note that there is a gradient in the quantile hp coefficients as well as disp. disp sign reverses, also the coefficient on factor am is different by quantiles:

```r
ls_fl_quantiles <- c(0.25, 0.50, 0.75)
fit_quantiles <- rq(mpg ~ disp + hp + factor(am),
               tau = ls_fl_quantiles,
               data = mtcars)
summary(fit_quantiles, se = "boot")
```

```
##
## Call: rq(formula = mpg ~ disp + hp + factor(am), tau = ls_fl_quantiles,
##     data = mtcars)
##
## tau: [1] 0.25
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) 25.34665  1.85938   13.63181  0.00000
## disp        -0.02441  0.00858   -2.84514  0.00837
## hp          -0.01672  0.01655   -1.00987  0.32152
## factor(am)1  1.64389  1.51309    1.08644  0.28689
##
## Call: rq(formula = mpg ~ disp + hp + factor(am), tau = ls_fl_quantiles,
##     data = mtcars)
##
## tau: [1] 0.5
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) 27.49722  1.91907   14.32839  0.00000
## disp        -0.02253  0.01640   -1.37347  0.18090
## hp          -0.02713  0.02469   -1.09910  0.28143
```

```
## factor(am)1  3.37328  1.92135    1.75568  0.09048
##
## Call: rq(formula = mpg ~ disp + hp + factor(am), tau = ls_fl_quantiles,
##     data = mtcars)
##
## tau: [1] 0.75
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) 28.52841  1.46520   19.47068  0.00000
## disp         0.00420  0.01299    0.32307  0.74913
## hp          -0.06815  0.01612   -4.22901  0.00024
## factor(am)1  8.03935  2.46073    3.26706  0.00296
```

6.2.1.2 Test Conditional Quantile Coefficients if Different

Use the rq.anova function frm the quantile regression packge to conduct WALD test. Remember WALD test says given unrestricted model's estimates, test where null is that the coefficients satisfy some linear restrictions.

To test, use the returned object from running rq with different numbers of quantiles, and set the option *joint* to true or false. When joint is true: "equality of slopes should be done as joint tests on all slope parameters", when joint is false: "separate tests on each of the slope parameters should be reported". A slope parameter refers to one of the RHS variables.

Note that quantile tests are "parallel line" tests. Meaning that we should except to have different x-intercepts for each quantile, because they represents the levels of the conditional shocks distributions. However, if quantile coefficients for the slopes are all the same, then there are no quantile specific effects, mean effects would be sufficient.

see:

- anova.rq() in quantreg package in R

6.2.1.2.1 Test Statistical Difference between 25th and 50th Conditional Quantiles

Given the quantile estimates above, the difference between 0.25 and 0.50 quantiles exists, but are they sufficiently large to be statistically different? What is the p-value? Reviewing the results below, they are not statistically different.

First, joint = TRUE. This is not testing if the coefficien on disp is the same as the coefficient on hp. This is testing jointly if the coefficients for different quantiles of disp, and different quantiles of hp are the same for each RHS variable.

```
ls_fl_quantiles <- c(0.25, 0.50)
fit_quantiles <- rq(mpg ~ disp + hp + factor(am),
               tau = ls_fl_quantiles,
               data = mtcars)
anova(fit_quantiles, test = "Wald", joint=TRUE)
```

```
## Quantile Regression Analysis of Deviance Table
##
## Model: mpg ~ disp + hp + factor(am)
## Joint Test of Equality of Slopes: tau in {  0.25 0.5  }
##
##   Df Resid Df F value Pr(>F)
## 1  3       59   0.705 0.5529
```

Second, joint = False:

```
anova(fit_quantiles, test = "Wald", joint=FALSE)
```

```
## Quantile Regression Analysis of Deviance Table
##
```

```
## Model: mpg ~ disp + hp + factor(am)
## Tests of Equality of Distinct Slopes: tau in {  0.25 0.5  }
##
##              Df Resid Df F value Pr(>F)
## disp          1       61  0.0307 0.8615
## hp            1       61  0.5418 0.4645
## factor(am)1  1       61  0.9959 0.3222
```

6.2.1.2.2 Test Statistical Difference between 25th, 50th, and 75th Conditional Quantiles

The 1st quartile and median do not seem to be statistically different, now include the 3rd quartile. As seen earlier, the quartiles jointly show a gradient. Now, we can see that idisp, hp and am are separately have statistically different

First, joint = TRUE:

```
ls_fl_quantiles <- c(0.25, 0.50, 0.75)
fit_quantiles <- rq(mpg ~ disp + hp + factor(am),
                tau = ls_fl_quantiles,
                data = mtcars)
anova(fit_quantiles, test = "Wald", joint=TRUE)
```

```
## Quantile Regression Analysis of Deviance Table
##
## Model: mpg ~ disp + hp + factor(am)
## Joint Test of Equality of Slopes: tau in {  0.25 0.5 0.75  }
##
##   Df Resid Df F value   Pr(>F)
## 1  6       87   3.292 0.005752 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Second, joint = False:

```
anova(fit_quantiles, test = "Wald", joint=FALSE)
```

```
## Quantile Regression Analysis of Deviance Table
##
## Model: mpg ~ disp + hp + factor(am)
## Tests of Equality of Distinct Slopes: tau in {  0.25 0.5 0.75  }
##
##              Df Resid Df F value   Pr(>F)
## disp          2       91  5.4482 0.005823 **
## hp            2       91  7.2035 0.001247 **
## factor(am)1  2       91  6.8592 0.001680 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Chapter 7

Optimization

7.1 Grid Based Optimization

7.1.1 Find Maximum By Iterating Over Grids

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

7.1.1.1 Single Parameter Optimization

We have a function $f(\mu)$, we know that $a \le \mu \le b$, and we want to find the value of μ that maximizes $f(\mu)$ within the bounds. The same idea here is used in various aspects of solving the dynamic equilibrium borrowing and savings problem in Wang (2022) (preprint pdf).

First, we create a simple quadratic function. the minimum of the function is where $\mu = -2$

```
# Define Function
ffi_quad_func <- function(fl_mu) {
  1 + (fl_mu + 2)^2
}
# Test Function
print(paste0("ffi_quad_func(-3)=", ffi_quad_func(-3)))
```

```
## [1] "ffi_quad_func(-3)=2"
```

```
print(paste0("ffi_quad_func(-2)=", ffi_quad_func(-2)))
```

```
## [1] "ffi_quad_func(-2)=1"
```

```
print(paste0("ffi_quad_func(-1)=", ffi_quad_func(-1)))
```

```
## [1] "ffi_quad_func(-1)=2"
```

Second, we develop the maximizer function given grid.

```
# Function
ffi_find_min <- function(fl_min = -4, fl_max = 2, it_grid_len = 7) {

  # Construct grid where to evaluate the function
  ar_fl_mu <- seq(fl_min, fl_max, length.out = it_grid_len)

  # Evaluate likelihood
  ar_obj <- sapply(ar_fl_mu, ffi_quad_func)

  # Find min grid
  it_min_idx <- which.min(ar_obj)
  fl_min_val <- ar_obj[it_min_idx]
```

```
  # Find lower and upper bound
  fl_min_new <- ar_fl_mu[max(it_min_idx - 1, 1)]
  fl_max_new <- ar_fl_mu[min(it_min_idx + 1, it_grid_len)]

  # return
  return(list(
    fl_min_val = fl_min_val,
    fl_min_new = fl_min_new,
    fl_max_new = fl_max_new
  ))
}
# Test Function
print("ffi_find_min(-3,-1,10)")
```

```
## [1] "ffi_find_min(-3,-1,10)"
```

```
print(ffi_find_min(-3, -1, 10))
```

```
## $fl_min_val
## [1] 1.012346
##
## $fl_min_new
## [1] -2.333333
##
## $fl_max_new
## [1] -1.888889
```

```
# Test function if lower bound is actual min
print("ffi_find_min(-2,-1,10)")
```

```
## [1] "ffi_find_min(-2,-1,10)"
```

```
print(ffi_find_min(-2, -1, 10))
```

```
## $fl_min_val
## [1] 1
##
## $fl_min_new
## [1] -2
##
## $fl_max_new
## [1] -1.888889
```

```
# Test function if upper bound is actual min
print("ffi_find_min(-3,-2,10)")
```

```
## [1] "ffi_find_min(-3,-2,10)"
```

```
print(ffi_find_min(-3, -2, 10))
```

```
## $fl_min_val
## [1] 1
##
## $fl_min_new
## [1] -2.111111
##
## $fl_max_new
## [1] -2
```

Third, we iterately zoom-in to ever finer grid around the point in the last grid where the objective function had the lowest value.

```
# Initialize min and max and tolerance criteria
fl_min_cur <- -10
fl_max_cur <- 10
it_grid_len <- 10
fl_tol <- 1e-5
it_max_iter <- 5

# Initialize initial gaps etc
fl_gap <- 1e5
fl_min_val_last <- 1e5
it_iter <- 0

# Iteratively loop over grid to find the maximum by zooming in
while ((fl_gap > fl_tol) && it_iter <= it_max_iter) {

  # Iterator counts up
  it_iter <- it_iter + 1
  print(paste0("it_iter=", it_iter))

  # build array
  ls_find_min <- ffi_find_min(
    fl_min = fl_min_cur, fl_max = fl_max_cur, it_grid_len = it_grid_len
  )

  # Min objective value current
  fl_min_val <- ls_find_min$fl_min_val
  # Find new lower and upper bound
  fl_min_cur <- ls_find_min$fl_min_new
  fl_max_cur <- ls_find_min$fl_max_new
  print(paste0("fl_min_cur=", fl_min_cur))
  print(paste0("fl_max_cur=", fl_max_cur))

  # Compare
  fl_gap <- abs(fl_min_val - fl_min_val_last)
  fl_min_val_last <- fl_min_val
  print(paste0("fl_gap=", fl_gap))
}
```

```
## [1] "it_iter=1"
## [1] "fl_min_cur=-3.33333333333333"
## [1] "fl_max_cur=1.11111111111111"
## [1] "fl_gap=99998.2098765432"
## [1] "it_iter=2"
## [1] "fl_min_cur=-2.34567901234568"
## [1] "fl_max_cur=-1.35802469135802"
## [1] "fl_gap=0.768175582990399"
## [1] "it_iter=3"
## [1] "fl_min_cur=-2.12620027434842"
## [1] "fl_max_cur=-1.90672153635117"
## [1] "fl_gap=0.0216769123947906"
## [1] "it_iter=4"
## [1] "fl_min_cur=-2.02865416857186"
## [1] "fl_max_cur=-1.97988111568358"
## [1] "fl_gap=0.00025274863560476"
## [1] "it_iter=5"
## [1] "fl_min_cur=-2.00697725617707"
## [1] "fl_max_cur=-1.99613879997968"
## [1] "fl_gap=1.57853178373024e-05"
```

```
## [1] "it_iter=6"
## [1] "fl_min_cur=-2.00095589162296"
## [1] "fl_max_cur=-1.99854734580132"
## [1] "fl_gap=2.36575822887275e-06"
```

7.1.2 Bisection

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

See the ff_opti_bisect_pmap_multi function from Fan's *REconTools* Package, which provides a resuable function based on the algorithm worked out here.

The bisection specific code does not need to do much.

- list variables in file for grouping, each group is an individual for whom we want to calculate optimal choice for using bisection.
- string variable name of input where functions are evaluated, these are already contained in the dataframe, existing variable names, row specific, rowwise computation over these, each rowwise calculation using different rows.
- scalar and array values that are applied to every rowwise calculation, all rowwise calculations using the same scalars and arrays.
- string output variable name

This is how I implement the bisection algorithm, when we know the bounding minimum and maximum to be below and above zero already.

1. Evaluate $f_a^0 = f(a^0)$ and $f_b^0 = f(b^0)$, min and max points.
2. Evaluate at $f_p^0 = f(p^0)$, where $p_0 = \frac{a^0+b^0}{2}$.
3. if $f_a^i \cdot f_p^i < 0$, then $b_{i+1} = p_i$, else, $a_{i+1} = p_i$ and $f_a^{i+1} = p_i$.
4. iteratre until convergence.

Generate New columns of a and b as we iteratre, do not need to store p, p is temporary. Evaluate the function below which we have already tested, but now, in the dataframe before generating all permutations, *tb_states_choices*, now the *fl_N* element will be changing with each iteration, it will be row specific. *fl_N* are first min and max, then each subsequent ps.

7.1.2.1 Initialize Matrix

Prepare Input Data:

```
# Parameters
fl_rho = 0.20
svr_id_var = 'INDI_ID'

# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = 4)
ar_nN_alpha = seq(0.1, 0.9, length.out = 4)

# Choice Grid for nutritional feasible choices for each
fl_N_agg = 100
fl_N_min = 0

# Mesh Expand
tb_states_choices <- as_tibble(cbind(ar_nN_A, ar_nN_alpha)) %>%
  rowid_to_column(var=svr_id_var)

# Convert Matrix to Tibble
ar_st_col_names = c(svr_id_var,'fl_A', 'fl_alpha')
tb_states_choices <- tb_states_choices %>% rename_all(~c(ar_st_col_names))
```

Prepare Function:

```
# Define Implicit Function
ffi_nonlin_dplyrdo <- function(fl_A, fl_alpha, fl_N, ar_A, ar_alpha, fl_N_agg, fl_rho){

  ar_p1_s1 = exp((fl_A - ar_A)*fl_rho)
  ar_p1_s2 = (fl_alpha/ar_alpha)
  ar_p1_s3 = (1/(ar_alpha*fl_rho - 1))
  ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
  ar_p2 = fl_N^((fl_alpha*fl_rho-1)/(ar_alpha*fl_rho-1))
  ar_overall = ar_p1*ar_p2
  fl_overall = fl_N_agg - sum(ar_overall)

  return(fl_overall)
}
```

Initialize the matrix with a_0 and b_0, the initial min and max points:

```
# common prefix to make reshaping easier
st_bisec_prefix <- 'bisec_'
svr_a_lst <- paste0(st_bisec_prefix, 'a_0')
svr_b_lst <- paste0(st_bisec_prefix, 'b_0')
svr_fa_lst <- paste0(st_bisec_prefix, 'fa_0')
svr_fb_lst <- paste0(st_bisec_prefix, 'fb_0')

# Add initial a and b
tb_states_choices_bisec <- tb_states_choices %>%
  mutate(!!sym(svr_a_lst) := fl_N_min, !!sym(svr_b_lst) := fl_N_agg)

# Evaluate function f(a_0) and f(b_0)
tb_states_choices_bisec <- tb_states_choices_bisec %>%
  rowwise() %>%
  mutate(!!sym(svr_fa_lst) := ffi_nonlin_dplyrdo(fl_A, fl_alpha, !!sym(svr_a_lst),
                                                ar_nN_A, ar_nN_alpha,
                                                fl_N_agg, fl_rho),
         !!sym(svr_fb_lst) := ffi_nonlin_dplyrdo(fl_A, fl_alpha, !!sym(svr_b_lst),
                                                ar_nN_A, ar_nN_alpha,
                                                fl_N_agg, fl_rho))
# Summarize
dim(tb_states_choices_bisec)
```

```
## [1] 4 7
```

```
# summary(tb_states_choices_bisec)
```

7.1.2.2 Iterate and Solve for f(p), update f(a) and f(b)

Implement the DPLYR based Concurrent bisection algorithm.

```
# fl_tol = float tolerance criteria
# it_tol = number of interations to allow at most
fl_tol <- 10^-2
it_tol <- 100

# fl_p_dist2zr = distance to zero to initalize
fl_p_dist2zr <- 1000
it_cur <- 0
while (it_cur <= it_tol && fl_p_dist2zr >= fl_tol ) {

  it_cur <- it_cur + 1

  # New Variables
```

```
  svr_a_cur <- paste0(st_bisec_prefix, 'a_', it_cur)
  svr_b_cur <- paste0(st_bisec_prefix, 'b_', it_cur)
  svr_fa_cur <- paste0(st_bisec_prefix, 'fa_', it_cur)
  svr_fb_cur <- paste0(st_bisec_prefix, 'fb_', it_cur)

  # Evaluate function f(a_0) and f(b_0)
  # 1. generate p
  # 2. generate f_p
  # 3. generate f_p*f_a
  tb_states_choices_bisec <- tb_states_choices_bisec %>%
    rowwise() %>%
    mutate(p = ((!!sym(svr_a_lst) + !!sym(svr_b_lst))/2)) %>%
    mutate(f_p = ffi_nonlin_dplyrdo(fl_A, fl_alpha, p,
                                    ar_nN_A, ar_nN_alpha,
                                    fl_N_agg, fl_rho)) %>%
    mutate(f_p_t_f_a = f_p*!!sym(svr_fa_lst))
  # fl_p_dist2zr = sum(abs(p))
  fl_p_dist2zr <- mean(abs(tb_states_choices_bisec %>% pull(f_p)))

  # Update a and b
  tb_states_choices_bisec <- tb_states_choices_bisec %>%
    mutate(!!sym(svr_a_cur) :=
             case_when(f_p_t_f_a < 0 ~ !!sym(svr_a_lst),
                       TRUE ~ p)) %>%
    mutate(!!sym(svr_b_cur) :=
             case_when(f_p_t_f_a < 0 ~ p,
                       TRUE ~ !!sym(svr_b_lst)))
  # Update f(a) and f(b)
  tb_states_choices_bisec <- tb_states_choices_bisec %>%
    mutate(!!sym(svr_fa_cur) :=
             case_when(f_p_t_f_a < 0 ~ !!sym(svr_fa_lst),
                       TRUE ~ f_p)) %>%
    mutate(!!sym(svr_fb_cur) :=
             case_when(f_p_t_f_a < 0 ~ f_p,
                       TRUE ~ !!sym(svr_fb_lst)))
  # Save from last
  svr_a_lst <- svr_a_cur
  svr_b_lst <- svr_b_cur
  svr_fa_lst <- svr_fa_cur
  svr_fb_lst <- svr_fb_cur

  # Summar current round
  print(paste0('it_cur:', it_cur, ', fl_p_dist2zr:', fl_p_dist2zr))
  summary(tb_states_choices_bisec %>%
            select(one_of(svr_a_cur, svr_b_cur, svr_fa_cur, svr_fb_cur)))
}
```

```
## [1] "it_cur:1, fl_p_dist2zr:1597.93916362849"
## [1] "it_cur:2, fl_p_dist2zr:676.06602535902"
## [1] "it_cur:3, fl_p_dist2zr:286.850590132782"
## [1] "it_cur:4, fl_p_dist2zr:117.225493866655"
## [1] "it_cur:5, fl_p_dist2zr:37.570593471664"
## [1] "it_cur:6, fl_p_dist2zr:4.60826664896022"
## [1] "it_cur:7, fl_p_dist2zr:14.4217689135683"
## [1] "it_cur:8, fl_p_dist2zr:8.38950830086659"
## [1] "it_cur:9, fl_p_dist2zr:3.93347761455868"
## [1] "it_cur:10, fl_p_dist2zr:1.88261338941038"
## [1] "it_cur:11, fl_p_dist2zr:0.744478952222305"
```

INDI_ID	1.000000e+00	2.0000000	3.0000000	4.0000000
fl_A	-2.000000e+00	-0.6666667	0.6666667	2.0000000
fl_alpha	1.000000e-01	0.3666667	0.6333333	0.9000000
bisec_a_0	0.000000e+00	0.0000000	0.0000000	0.0000000
bisec_b_0	1.000000e+02	100.0000000	100.0000000	100.0000000
bisec_fa_0	1.000000e+02	100.0000000	100.0000000	100.0000000
bisec_fb_0	-1.288028e+04	-1394.7069782	-323.9421599	-51.9716069
p	1.544952e+00	8.5838318	24.8359680	65.0367737
f_p	-7.637200e-03	-0.0052211	-0.0016162	-0.0009405
f_p_t_f_a	-3.800000e-04	-0.0000237	-0.0000025	-0.0000002
bisec_a_1	0.000000e+00	0.0000000	0.0000000	50.0000000
bisec_b_1	5.000000e+01	50.0000000	50.0000000	100.0000000
bisec_fa_1	1.000000e+02	100.0000000	100.0000000	22.5557704
bisec_fb_1	-5.666956e+03	-595.7345364	-106.5105843	-51.9716069
bisec_a_2	0.000000e+00	0.0000000	0.0000000	50.0000000
bisec_b_2	2.500000e+01	25.0000000	25.0000000	75.0000000
bisec_fa_2	1.000000e+02	100.0000000	100.0000000	22.5557704
bisec_fb_2	-2.464562e+03	-224.1460032	-0.6857375	-14.8701831
bisec_a_3	0.000000e+00	0.0000000	12.5000000	62.5000000
bisec_b_3	1.250000e+01	12.5000000	25.0000000	75.0000000
bisec_fa_3	1.000000e+02	100.0000000	50.8640414	3.7940196
bisec_fb_3	-1.041574e+03	-51.1700464	-0.6857375	-14.8701831
bisec_a_4	0.000000e+00	6.2500000	18.7500000	62.5000000
bisec_b_4	6.250000e+00	12.5000000	25.0000000	68.7500000
bisec_fa_4	1.000000e+02	29.4271641	25.2510409	3.7940196

```
## [1] "it_cur:12, fl_p_dist2zr:0.187061801237917"
## [1] "it_cur:13, fl_p_dist2zr:0.117844913432613"
## [1] "it_cur:14, fl_p_dist2zr:0.0275365951418891"
## [1] "it_cur:15, fl_p_dist2zr:0.0515488156908255"
## [1] "it_cur:16, fl_p_dist2zr:0.0191152349149135"
## [1] "it_cur:17, fl_p_dist2zr:0.00385372194545752"
```

7.1.2.3 Reshape Wide to long to Wide

To view results easily, how iterations improved to help us find the roots, convert table from wide to long. Pivot twice. This allows us to easily graph out how bisection is working out iterationby iteration.

Here, we will first show what the raw table looks like, the wide only table, and then show the long version, and finally the version that is medium wide.

7.1.2.3.1 Table One–Very Wide Show what the *tb_states_choices_bisec* looks like.

Variables are formatted like: *bisec_xx_yy*, where yy is the iteration indicator, and xx is either a, b, fa, or fb.

```
kable(head(t(tb_states_choices_bisec), 25)) %>%
  kable_styling_fc()
```

```
# str(tb_states_choices_bisec)
```

7.1.2.3.2 Table Two–Very Wide to Very Long We want to treat the iteration count information that is the suffix of variable names as a variable by itself. Additionally, we want to treat the a,b,fa,fb as a variable. Structuring the data very long like this allows for easy graphing and other types of analysis. Rather than dealing with many many variables, we have only 3 core variables that store bisection iteration information.

Here we use the very nice *pivot_longer* function. Note that to achieve this, we put a common prefix in front of the variables we wanted to convert to long. THis is helpful, because we can easily identify which

INDI_ID	fl_A	fl_alpha	varname	biseciter	value
1	-2	0.1	a	0	0.000
1	-2	0.1	b	0	100.000
1	-2	0.1	fa	0	100.000
1	-2	0.1	fb	0	-12880.284
1	-2	0.1	a	1	0.000
1	-2	0.1	b	1	50.000
1	-2	0.1	fa	1	100.000
1	-2	0.1	fb	1	-5666.956
1	-2	0.1	a	2	0.000
1	-2	0.1	b	2	25.000
1	-2	0.1	fa	2	100.000
1	-2	0.1	fb	2	-2464.562
1	-2	0.1	a	3	0.000
1	-2	0.1	b	3	12.500
1	-2	0.1	fa	3	100.000

variables need to be reshaped.

```r
# New variables
svr_bisect_iter <- 'biseciter'
svr_abfafb_long_name <- 'varname'
svr_number_col <- 'value'
svr_id_bisect_iter <- paste0(svr_id_var, '_bisect_ier')

# Pivot wide to very long
tb_states_choices_bisec_long <- tb_states_choices_bisec %>%
  pivot_longer(
    cols = starts_with(st_bisec_prefix),
    names_to = c(svr_abfafb_long_name, svr_bisect_iter),
    names_pattern = paste0(st_bisec_prefix, "(.*)_(.*)"),
    values_to = svr_number_col
  )

# Print
# summary(tb_states_choices_bisec_long)
kable(head(tb_states_choices_bisec_long %>%
             select(-one_of('p','f_p','f_p_t_f_a')), 15)) %>%
  kable_styling_fc()
```

```r
kable(tail(tb_states_choices_bisec_long %>%
             select(-one_of('p','f_p','f_p_t_f_a')), 15)) %>%
  kable_styling_fc()
```

7.1.2.3.3 Table Two–Very Very Long to Wider Again But the previous results are too long, with the a, b, fa, and fb all in one column as different categories, they are really not different categories, they are in fact different types of variables. So we want to spread those four categories of this variable into four columns, each one representing the a, b, fa, and fb values. The rows would then be uniquly identified by the iteration counter and individual ID.

```r
# Pivot wide to very long to a little wide
tb_states_choices_bisec_wider <- tb_states_choices_bisec_long %>%
  pivot_wider(
    names_from = !!sym(svr_abfafb_long_name),
    values_from = svr_number_col
  )

# Print
```

INDI_ID	fl_A	fl_alpha	varname	biseciter	value
4	2	0.9	b	14	65.0390625
4	2	0.9	fa	14	0.0047633
4	2	0.9	fb	14	-0.0043628
4	2	0.9	a	15	65.0360107
4	2	0.9	b	15	65.0390625
4	2	0.9	fa	15	0.0002003
4	2	0.9	fb	15	-0.0043628
4	2	0.9	a	16	65.0360107
4	2	0.9	b	16	65.0375366
4	2	0.9	fa	16	0.0002003
4	2	0.9	fb	16	-0.0020812
4	2	0.9	a	17	65.0360107
4	2	0.9	b	17	65.0367737
4	2	0.9	fa	17	0.0002003
4	2	0.9	fb	17	-0.0009405

```
# summary(tb_states_choices_bisec_wider)
kable(head(tb_states_choices_bisec_wider %>%
             select(-one_of('p','f_p','f_p_t_f_a')), 10)) %>%
  kable_styling_fc_wide()
```

INDI_ID	fl_A	fl_alpha	biseciter	a	b	fa	fb
1	-2	0.1	0	0.000000	100.0000	100.00000	-12880.283918
1	-2	0.1	1	0.000000	50.0000	100.00000	-5666.955763
1	-2	0.1	2	0.000000	25.0000	100.00000	-2464.562178
1	-2	0.1	3	0.000000	12.5000	100.00000	-1041.574253
1	-2	0.1	4	0.000000	6.2500	100.00000	-408.674764
1	-2	0.1	5	0.000000	3.1250	100.00000	-126.904283
1	-2	0.1	6	0.000000	1.5625	100.00000	-1.328965
1	-2	0.1	7	0.781250	1.5625	54.69612	-1.328965
1	-2	0.1	8	1.171875	1.5625	27.46061	-1.328965
1	-2	0.1	9	1.367188	1.5625	13.23495	-1.328965

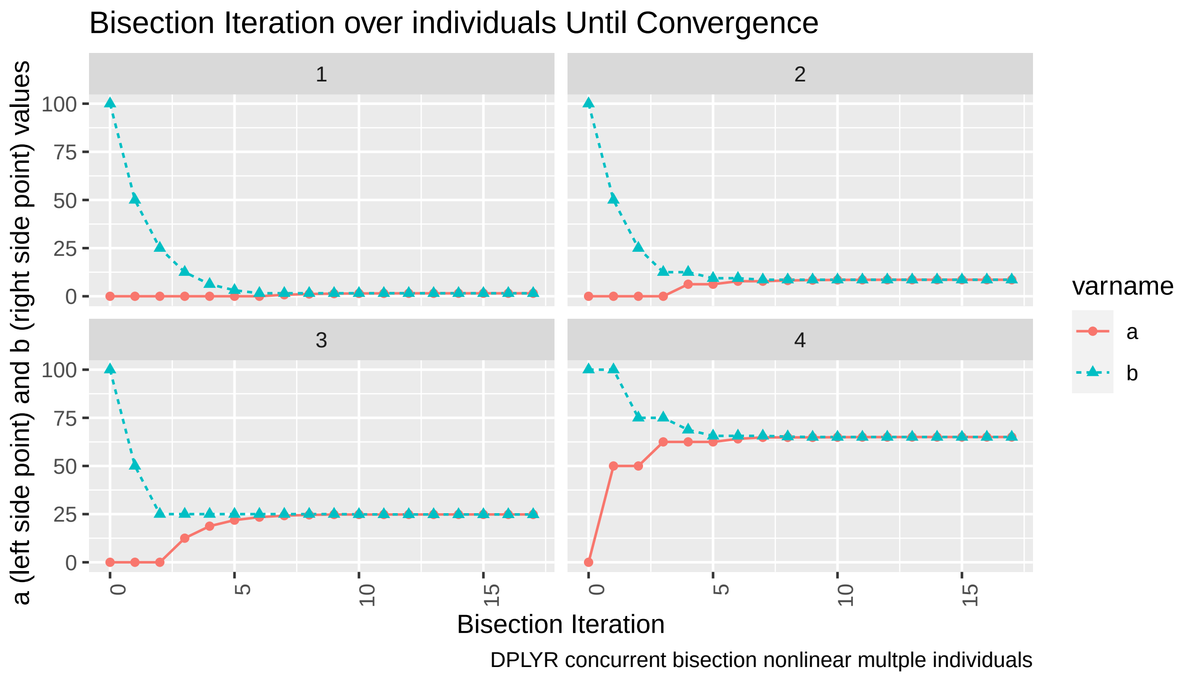
```
kable(head(tb_states_choices_bisec_wider %>%
             select(-one_of('p','f_p','f_p_t_f_a')), 10)) %>%
  kable_styling_fc_wide()
```

INDI_ID	fl_A	fl_alpha	biseciter	a	b	fa	fb
1	-2	0.1	0	0.000000	100.0000	100.00000	-12880.283918
1	-2	0.1	1	0.000000	50.0000	100.00000	-5666.955763
1	-2	0.1	2	0.000000	25.0000	100.00000	-2464.562178
1	-2	0.1	3	0.000000	12.5000	100.00000	-1041.574253
1	-2	0.1	4	0.000000	6.2500	100.00000	-408.674764
1	-2	0.1	5	0.000000	3.1250	100.00000	-126.904283
1	-2	0.1	6	0.000000	1.5625	100.00000	-1.328965
1	-2	0.1	7	0.781250	1.5625	54.69612	-1.328965
1	-2	0.1	8	1.171875	1.5625	27.46061	-1.328965
1	-2	0.1	9	1.367188	1.5625	13.23495	-1.328965

7.1.2.4 Graph Bisection Iteration Results

Actually we want to graph based on the long results, not the wider. Wider easier to view in table.

```
# Graph results
lineplot <- tb_states_choices_bisec_long %>%
    mutate(!!sym(svr_bisect_iter) := as.numeric(!!sym(svr_bisect_iter))) %>%
    filter(!!sym(svr_abfafb_long_name) %in% c('a', 'b')) %>%
    ggplot(aes(x=!!sym(svr_bisect_iter), y=!!sym(svr_number_col),
               colour=!!sym(svr_abfafb_long_name),
               linetype=!!sym(svr_abfafb_long_name),
               shape=!!sym(svr_abfafb_long_name))) +
        facet_wrap( ~ INDI_ID) +
        geom_line() +
        geom_point() +
        labs(title = 'Bisection Iteration over individuals Until Convergence',
             x = 'Bisection Iteration',
             y = 'a (left side point) and b (right side point) values',
             caption = 'DPLYR concurrent bisection nonlinear multple individuals') +
      theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(lineplot)
```



Chapter 8

Mathematics

8.1 Basics

8.1.1 Polynomial Formulas for Points

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

8.1.1.1 Formulas for Quadratic Parameters and Three Points

There are three points defined by their x and y coordinates, find a curve that fits through them exactly.

First, we have three equations with three unknowns, A, B, and C.

$$\begin{aligned}
y_1 &= A + B \cdot x_1 + C \cdot x_1^2 \\
y_2 &= A + B \cdot x_2 + C \cdot x_2^2 \\
y_3 &= A + B \cdot x_3 + C \cdot x_3^2
\end{aligned}$$

Second, we rewrite the problem in matrix form.

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Third, we solve the system of equations for the unknown vector $[A, B, C]$, via elementary row operations. The code below uses matlab to arrive at symbolic analytical solutions for for $[A, B, C]$.

```
clc
clear

% Define inputs
syms x11 x12 x21 x22 x31 x32 y1 y2 y3
mt_sm_z = [1,x11,x12;1,x21,x22;1,x31,x32];
ar_sm_y = [y1;y2;y3];

% Solve analytically
ar_sm_solu = linsolve(mt_sm_z, ar_sm_y)

% Randomly draw x and y values
rng(1234);
mt_rand = rand(3,2);
mt_rand = [0.1915, 0.6221, 0.4377;
  0.7854, 0.7800, 0.2726]';
```

```
[fl_x1, fl_x2, fl_x3] = deal(mt_rand(1,1), mt_rand(2,1), mt_rand(3,1));
[fl_y1, fl_y2, fl_y3] = deal(mt_rand(1,2), mt_rand(2,2), mt_rand(3,2));
[fl_x11, fl_x21, fl_x31] = deal(fl_x1, fl_x2, fl_x3);
[fl_x12, fl_x22, fl_x32] = deal(fl_x1^2, fl_x2^2, fl_x3^2);

% Numerically evaluate coefficients
ar_fl_solu = double(subs(ar_sm_solu, ...
  {x11, x12, x21, x22, x31, x32, y1, y2, y3}, ...
  {fl_x11,fl_x12,fl_x21,fl_x22,fl_x31,fl_x32,fl_y1,fl_y2,fl_y3}));
disp(['ar_fl_solu=', num2str(ar_fl_solu')])

% Y predictions
mt_fl_z = [1,fl_x11,fl_x12;1,fl_x21,fl_x22;1,fl_x31,fl_x32];
ar_fl_y_pred = mt_fl_z*ar_fl_solu;
ar_fl_x_actual = [fl_x1;fl_x2;fl_x3];
ar_fl_y_actual = [fl_y1;fl_y2;fl_y3];

% Compare results
tb_test = array2table([ar_fl_x_actual';ar_fl_y_actual';ar_fl_y_pred']');
cl_col_names = ["x_actual","y_actual", "y_predict"];
cl_row_names = strcat('obs_', string((1:3)));
tb_test.Properties.VariableNames = matlab.lang.makeValidName(cl_col_names);
tb_test.Properties.RowNames = matlab.lang.makeValidName(cl_row_names);
display(tb_test);
```

Fourth, the solutions are as follows.

$$A = \frac{x_1 x_2^2 y_3 - x_1^2 x_2 y_3 - x_1 x_3^2 y_2 + x_1^2 x_3 y_2 + x_2 x_3^2 y_1 - x_2^2 x_3 y_1}{x_1 x_2^2 - x_1^2 x_2 - x_1 x_3^2 + x_1^2 x_3 + x_2 x_3^2 - x_2^2 x_3}$$

$$B = \frac{-\left(x_1^2 y_2 - x_1^2 y_3 - x_2^2 y_1 + x_2^2 y_3 + x_3^2 y_1 - x_3^2 y_2\right)}{x_1 x_2^2 - x_1^2 x_2 - x_1 x_3^2 + x_1^2 x_3 + x_2 x_3^2 - x_2^2 x_3}$$

$$C = \frac{x_1 y_2 - x_1 y_3 - x_2 y_1 + x_2 y_3 + x_3 y_1 - x_3 y_2}{x_1 x_2^2 - x_1^2 x_2 - x_1 x_3^2 + x_1^2 x_3 + x_2 x_3^2 - x_2^2 x_3}$$

Fifth, given three pairs randomly drawn x and y points, we use the formulas just derived to find the parameters for the quadratic polynomial.

```
# Inputs X and Y
set.seed(123)
# Draw Randomly
mt_rnorm <- matrix(rnorm(6, mean = 1, sd = 1), nrow = 3, ncol = 2)
# # Fixed Values
# mt_rnorm <- matrix(c(
#   0.1915, 0.6221, 0.4377,
#   0.7854, 0.7800, 0.2726
# ), nrow = 3, ncol = 2)
colnames(mt_rnorm) <- c("x", "y")
x1 <- mt_rnorm[1, 1]
x2 <- mt_rnorm[2, 1]
x3 <- mt_rnorm[3, 1]
y1 <- mt_rnorm[1, 2]
y2 <- mt_rnorm[2, 2]
y3 <- mt_rnorm[3, 2]

# X quadratic
x11 <- x1
```

```r
x12 <- x1**2
x21 <- x2
x22 <- x2**2
x31 <- x3
x32 <- x3**2

# Shared denominator
fl_denominator <- (x11 * x22 - x12 * x21
  - x11 * x32 + x12 * x31
  + x21 * x32 - x22 * x31)

# Solve for A, B, and C exact fit quadratic coefficients
fl_A <- (x11 * x22 * y3 - x12 * x21 * y3
  - x11 * x32 * y2 + x12 * x31 * y2
  + x21 * x32 * y1 - x22 * x31 * y1) / fl_denominator

fl_B <- -(x12 * y2 - x12 * y3
  - x22 * y1 + x22 * y3
  + x32 * y1 - x32 * y2) / fl_denominator

fl_C <- (x11 * y2 - x11 * y3
  - x21 * y1 + x21 * y3
  + x31 * y1 - x31 * y2) / fl_denominator

# Display
st_display <- paste0(
  "A(intercept)=", round(fl_A, 3),
  ", B(lin)=", round(fl_B, 3),
  ", C(quad)=", round(fl_C, 3)
)
print(st_display)
```

```
## [1] "A(intercept)=1.105, B(lin)=-0.226, C(quad)=0.334"
```

Sixth, to check that the estimates are correct, we derive results from running quadratic estimation with the three points of data drawn. We use both polynomial and orthogonal polynomials below.

```r
# Estimation results
df_rnorm <- as_tibble(mt_rnorm)
# Linear and quadratic terms
rs_lm_quad <- stats::lm(y ~ x + I(x^2), data = df_rnorm)
print(stats::summary.lm(rs_lm_quad))
```

```
##
## Call:
## stats::lm(formula = y ~ x + I(x^2), data = df_rnorm)
##
## Residuals:
## ALL 3 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.1054        NaN     NaN      NaN
## x            -0.2264        NaN     NaN      NaN
## I(x^2)        0.3343        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 2 and 0 DF,  p-value: NA
```

Quadratic Fit of 3 Sets of Random (X,Y) Points

x	y	ar_pred_sym	ar_pred_lm	as_pred_lm_otho	res
0.4395244	1.070508	1.070508	1.070508	1.070508	0
0.7698225	1.129288	1.129288	1.129288	1.129288	0
2.5587083	2.715065	2.715065	2.715065	2.715065	0

```
# Using orthogonal polynomials
# vs. rs_lm_quad: different parameters, but same predictions
rs_lm_quad_otho <- stats::lm(y ~ poly(x, 2), data = df_rnorm)
print(stats::summary.lm(rs_lm_quad_otho))
```

```
##
## Call:
## stats::lm(formula = y ~ poly(x, 2), data = df_rnorm)
##
## Residuals:
## ALL 3 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.6383        NaN     NaN      NaN
## poly(x, 2)1   1.3109        NaN     NaN      NaN
## poly(x, 2)2   0.1499        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 2 and 0 DF,  p-value: NA
```

Seventh, now we compare between the predications based on analytical solutions and lm regression.

```
# Matrix of input values
mt_vals_xs <- t(
  matrix(c(1, x1, x1**2, 1, x2, x2**2, 1, x3, x3**2),
    nrow = 3, ncol = 3
  )
)

# Predictions from LM poly prediction
ar_pred_lm <- mt_vals_xs %*% as.numeric(rs_lm_quad$coefficients)
as_pred_lm_otho <- stats::predict(rs_lm_quad_otho)

# Predictions based on analytical solutions
ar_pred_sym <- mt_vals_xs %*% c(fl_A, fl_B, fl_C)

# Combine results
kable(
  cbind(
    df_rnorm, ar_pred_sym,
    ar_pred_lm, as_pred_lm_otho
  ) %>%
    mutate(res = ar_pred_lm - y),
  caption = paste0(
    "Quadratic Fit of 3 Sets of Random (X,Y) Points"
  )
) %>% kable_styling_fc()
```

8.1.1.2 Formulas for Quadratic Parameters and Differences

Similar to the prior example, we remain interested in quadratic parameters, however, we no longer observe three sets of x and y values, instead, we observe differences in the y values. Specifically, we observe Δy_2 and Δy_3, not y_3, y_2, and y_1. We do still observe x_1, x_2, and x_3. We can no longer identify A, but we can still identify B and C.

$$
\begin{aligned}
y_2 - y_1 = \Delta y_2 = B \cdot (x_2 - x_1) + C \cdot (x_2^2 - x_1^2) \\
y_3 - y_2 = \Delta y_3 = B \cdot (x_3 - x_2) + C \cdot (x_3^2 - x_2^2)
\end{aligned}
$$

First, we rewrite the problem in matrix form.

$$
\begin{bmatrix}
\underbrace{x_2 - x_1}_{T} & \underbrace{x_2^2 - x_1^2}_{U} \\
\underbrace{x_3 - x_2}_{V} & \underbrace{x_3^2 - x_2^2}_{W}
\end{bmatrix}
\cdot
\begin{bmatrix} B \\ C \end{bmatrix}
=
\begin{bmatrix}
\underbrace{y_2 - y_1}_{R} \\
\underbrace{y_3 - y_2}_{S}
\end{bmatrix}
$$

Second, we solve the system of equations for the unknown vector $[B, C]$, via elementary row operations. The code below uses matlab to arrive at symbolic analytical solutions for for $[B, C]$. Derivations of the solutions are also shown here: Reduced Row Echelon Form with 2 Equations and 2 Unknowns.

```
clc
clear

% Define inputs
syms T U V W R S
mt_sm_z = [T, U; V, W];
ar_sm_y = [R; S];

% Solve analytically
ar_sm_solu = linsolve(mt_sm_z, ar_sm_y)

% Randomly draw x and y values
rng(1234);
mt_rand = rand(3,2);
% Use below to check not-exact fit, gap actual and predict of y
mt_rand = [0.1915, 0.6221, 0.4377;
           0.7854, 0.7800, 0.2726]';
% Use below to check for exact fit 2nd 3rd points same
% mt_rand = [0.1915, 0.6221, 0.6221;
%  0.7854, 0.7800, 0.7800]';
[fl_x1, fl_x2, fl_x3] = deal(mt_rand(1,1), mt_rand(2,1), mt_rand(3,1));
[fl_y1, fl_y2, fl_y3] = deal(mt_rand(1,2), mt_rand(2,2), mt_rand(3,2));
[fl_x11, fl_x21, fl_x31] = deal(fl_x1, fl_x2, fl_x3);
[fl_x12, fl_x22, fl_x32] = deal(fl_x1^2, fl_x2^2, fl_x3^2);

% Define values of U V Q and S
fl_T = fl_x21 - fl_x11;
fl_U = fl_x22 - fl_x12;
fl_V = fl_x31 - fl_x21;
fl_W = fl_x32 - fl_x22;
fl_R = fl_y2 - fl_y1;
fl_S = fl_y3 - fl_y2;

% Numerically evaluate coefficients
ar_fl_solu = double(subs(ar_sm_solu, ...
  {T, U, V, W, R, S}, ...
  {fl_T, fl_U, fl_V, fl_W, fl_R, fl_S}));
disp(['ar_fl_solu=', num2str(ar_fl_solu')])
```

```
% Y difference predictions
mt_fl_z = [fl_T, fl_U;fl_V, fl_W];
ar_fl_y_diff_pred = mt_fl_z*ar_fl_solu;
ar_fl_x_diff_actual = [fl_T;fl_V];
ar_fl_x_sqr_diff_actual = [fl_U;fl_W];
ar_fl_y_diff_actual = [fl_R;fl_S];

% Compare results
tb_test = array2table( ...
  [ar_fl_x_diff_actual'; ar_fl_x_sqr_diff_actual'; ...
   ar_fl_y_diff_actual'; ar_fl_y_diff_pred']');
cl_col_names = ["x_diff_actual", "x_sqr_diff_actual", ...
                "y_diff_actual", "y_diff_predict"];
cl_row_names = strcat('diff_obs_', string((1:2)));
tb_test.Properties.VariableNames = matlab.lang.makeValidName(cl_col_names);
tb_test.Properties.RowNames = matlab.lang.makeValidName(cl_row_names);
display(tb_test);
```

Third, the solutions for the intercept and quadratic terms based on differences in y values and based on functions of the all x values are as follows.

$$
\begin{aligned}
B &= \frac{RW - SU}{TW - UV} \\
&= \frac{(y_2 - y_1) \cdot (x_3^2 - x_2^2) - (y_3 - y_2) \cdot (x_2^2 - x_1^2)}{(x_2 - x_1) \cdot (x_3^2 - x_2^2) - (x_2^2 - x_1^2) \cdot (x_3 - x_2)}
\end{aligned}
$$

$$
\begin{aligned}
C &= \frac{ST - RV}{TW - UV} \\
&= \frac{(y_3 - y_2) \cdot (x_2 - x_1) - (y_2 - y_1) \cdot (x_3 - x_2)}{(x_2 - x_1) \cdot (x_3^2 - x_2^2) - (x_2^2 - x_1^2) \cdot (x_3 - x_2)}
\end{aligned}
$$

Fourth, given three pairs randomly drawn x and y points, we use the formulas just derived to find the parameters for the best-fitting slope and quadratic parameters given observed y differences.

```
# Inputs X and Y
set.seed(123)
# Draw Randomly
mt_rnorm <- matrix(rnorm(6, mean = 1, sd = 1), nrow = 3, ncol = 2)
# # Three fixed and different set of points
# mt_rnorm <- matrix(c(
#   0.1915, 0.6221, 0.4377,
#   0.7854, 0.7800, 0.2726
# ), nrow = 3, ncol = 2)

colnames(mt_rnorm) <- c("x", "y")
x1 <- mt_rnorm[1, 1]
x2 <- mt_rnorm[2, 1]
x3 <- mt_rnorm[3, 1]
y1 <- mt_rnorm[1, 2]
y2 <- mt_rnorm[2, 2]
y3 <- mt_rnorm[3, 2]

# X quadratic
x11 <- x1
x12 <- x1**2
x21 <- x2
x22 <- x2**2
```

```
x31 <- x3
x32 <- x3**2

# Define U and V, as well as Q and S
fl_T <- x21 - x11;
fl_U <- x22 - x12;
fl_V <- x31 - x21;
fl_W <- x32 - x22;
fl_R <- y2 -  y1;
fl_S <- y3 -  y2;

# Shared denominator
fl_denominator <- (fl_T*fl_W - fl_U*fl_V)

# Solve for A and B coefficients (not exact fit)
fl_B <- (fl_R * fl_W - fl_S * fl_U) / fl_denominator
fl_C <- (fl_S * fl_T - fl_R * fl_V) / fl_denominator

# Display
st_display <- paste0(
  "B(lin)=", round(fl_B, 3),
  ", C(quad)=", round(fl_C, 3)
)
print(st_display)
```

```
## [1] "B(lin)=-0.226, C(quad)=0.334"
```

Fifth, to check that the estimates are correct, we derive results from running linear estimation with the three points of data drawn, using all level information. As prior, we use both polynomial and orthogonal polynomials below. Note that we can not run this regression in practice because we do not observe levels, just differences.

```
# Estimation results
df_rnorm <- as_tibble(mt_rnorm)
# Linear and quadratic terms
rs_lm_quad <- stats::lm(y ~ x + I(x^2), data = df_rnorm)
print(stats::summary.lm(rs_lm_quad))
```

```
##
## Call:
## stats::lm(formula = y ~ x + I(x^2), data = df_rnorm)
##
## Residuals:
## ALL 3 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.1054        NaN     NaN      NaN
## x            -0.2264        NaN     NaN      NaN
## I(x^2)        0.3343        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 2 and 0 DF,  p-value: NA
```

```
# Using orthogonal polynomials
# vs. rs_lm_quad: different parameters, but same predictions
rs_lm_quad_otho <- stats::lm(y ~ poly(x, 2), data = df_rnorm)
print(stats::summary.lm(rs_lm_quad_otho))
```

Quadratic Fit, given 3 Sets of Random (X,Y) Points' Differences in Y

	x	y	ar_pred_sym	ar_pred_lm	as_pred_lm_otho	res
2	0.3302982	0.0587793	0.0587793	0.0587793	0.0587793	0
3	1.7888858	1.5857773	1.5857773	1.5857773	1.5857773	0

```
##
## Call:
## stats::lm(formula = y ~ poly(x, 2), data = df_rnorm)
##
## Residuals:
## ALL 3 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.6383        NaN     NaN      NaN
## poly(x, 2)1   1.3109        NaN     NaN      NaN
## poly(x, 2)2   0.1499        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 2 and 0 DF,  p-value: NA
```

Sixth, now we compare between the predications based on analytical solutions using differences in y, and lm regression. The fit of differences in y are exact.

```
# Matrix of input values
mt_vals_xs <- t(
  matrix(c(x2 - x1, x2^2 - x1^2, x3 - x2, x3^2 - x2^2),
    nrow = 2, ncol = 2
  )
)

# Predictions from LM poly prediction
ar_pred_lm <- mt_vals_xs %*% as.vector(rs_lm_quad$coefficients)[2:3]
as_pred_lm_otho_lvl <- stats::predict(rs_lm_quad_otho)
as_pred_lm_otho <- t(t(diff(as_pred_lm_otho_lvl)))

# Predictions based on analytical solutions
ar_pred_sym <- mt_vals_xs %*% c(fl_B, fl_C)

# Combine results
kable(
  cbind(
    as_tibble(apply(df_rnorm, 2, diff)), ar_pred_sym,
    ar_pred_lm, as_pred_lm_otho
  ) %>%
    mutate(res = ar_pred_lm - y),
  caption = paste0(
    "Quadratic Fit, given 3 Sets of Random (X,Y) Points' Differences in Y"
  )
) %>% kable_styling_fc()
```

8.1.1.3 Formulas for Linear Fit with Three Points

We have three points, what are the optimizing intercept and slope parameters? We solve the following problem to minimize the sum-squared error given linear fit with the three points.

First, the minimizing objective function is:

$$O(A,B) = \left\{ \left(y_1 - A - Bx_1\right)^2 + \left(y_2 - A - Bx_2\right)^2 + \left(y_3 - A - Bx_3\right)^2 \right\}$$

Second, we solve for $min_{A,B} O(A,B)$. We take derivative of $O(A,B)$ with respect to A and B:

$$\begin{aligned}
\frac{\partial O}{\partial A} &= -\left(y_1 - A - Bx_1\right) - \left(y_2 - A - Bx_2\right) - \left(y_3 - A - Bx_3\right) \\
&= A + Bx_1 - y_1 + A + Bx_2 - y_2 + A + Bx_3 - y_3 \\
&= 3A + \left(x_1 + x_2 + x_3\right) B - \left(y_1 + y_2 + y_3\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial O}{\partial B} &= -x_1 \left(y_1 - A - Bx_1\right) - x_2 \left(y_2 - A - Bx_2\right) - x_3 \left(y_3 - A - Bx_3\right) \\
&= \left(x_1 + x_2 + x_3\right) A + \left(x_1^2 + x_2^2 + x_3^2\right) B - \left(x_1 y_1 + x_2 y_2 + x_3 y_3\right)
\end{aligned}$$

Third, at the optimizing minimum (note quadratic), we now have two equations with two unknowns:

$$\begin{aligned}
\left(y_1 + y_2 + y_3\right) &= 3A + \left(x_1 + x_2 + x_3\right) B \\
\left(x_1 y_1 + x_2 y_2 + x_3 y_3\right) &= \left(x_1 + x_2 + x_3\right) A + \left(x_1^2 + x_2^2 + x_3^2\right) B
\end{aligned}$$

Fourth, we rewrite the problem in matrix form.

$$\begin{bmatrix} 3 & \underbrace{x_1 + x_2 + x_3}_{U} \\ \underbrace{x_1 + x_2 + x_3}_{U} & \underbrace{x_1^2 + x_2^2 + x_3^2}_{V} \end{bmatrix} \cdot \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \underbrace{y_1 + y_2 + y_3}_{Q} \\ \underbrace{x_1 y_1 + x_2 y_2 + x_3 y_3}_{S} \end{bmatrix}$$

Fifth, we solve the system of equations for the unknown vector $[A, B]$, via elementary row operations. The code below uses matlab to arrive at symbolic analytical solutions for for $[A, B]$.

```
clc
clear

% Define inputs
syms U V Q S
mt_sm_z = [3, U; U, V];
ar_sm_y = [Q; S];

% Solve analytically
ar_sm_solu = linsolve(mt_sm_z, ar_sm_y)

% Randomly draw x and y values
rng(1234);
mt_rand = rand(3,2);
% Use below to check not-exact fit, gap actual and predict of y
mt_rand = [0.1915, 0.6221, 0.4377;
  0.7854, 0.7800, 0.2726]';
% Use below to check for exact fit 2nd 3rd points same
% mt_rand = [0.1915, 0.6221, 0.6221;
%  0.7854, 0.7800, 0.7800]';
[fl_x1, fl_x2, fl_x3] = deal(mt_rand(1,1), mt_rand(2,1), mt_rand(3,1));
[fl_y1, fl_y2, fl_y3] = deal(mt_rand(1,2), mt_rand(2,2), mt_rand(3,2));
[fl_x11, fl_x21, fl_x31] = deal(fl_x1, fl_x2, fl_x3);
[fl_x12, fl_x22, fl_x32] = deal(fl_x1^2, fl_x2^2, fl_x3^2);

% Define values of U V Q and S
fl_U = fl_x11 + fl_x21 + fl_x31;
fl_V = fl_x12 + fl_x22 + fl_x32;
```

```
fl_Q = fl_y1 + fl_y2 + fl_y3;
fl_S = fl_y1*fl_x11 + fl_y2*fl_x21 + fl_y3*fl_x31;

% Numerically evaluate coefficients
ar_fl_solu = double(subs(ar_sm_solu, ...
  {U, V, Q, S}, ...
  {fl_U, fl_V, fl_Q, fl_S}));
disp(['ar_fl_solu=', num2str(ar_fl_solu')])

% Y predictions
mt_fl_z = [1,fl_x11;1,fl_x21;1,fl_x31];
ar_fl_y_pred = mt_fl_z*ar_fl_solu;
ar_fl_x_actual = [fl_x1;fl_x2;fl_x3];
ar_fl_y_actual = [fl_y1;fl_y2;fl_y3];

% Compare results
tb_test = array2table([ar_fl_x_actual';ar_fl_y_actual';ar_fl_y_pred']');
cl_col_names = ["x_actual","y_actual", "y_predict"];
cl_row_names = strcat('obs_', string((1:3)));
tb_test.Properties.VariableNames = matlab.lang.makeValidName(cl_col_names);
tb_test.Properties.RowNames = matlab.lang.makeValidName(cl_row_names);
display(tb_test);
```

Sixth, the solutions are as follows.

$$\begin{aligned}
A &= \frac{QV - SU}{-U^2 + 3V} \\
&= \frac{(y_1 + y_2 + y_3) \cdot (x_1^2 + x_2^2 + x_3^2) - (x_1 y_1 + x_2 y_2 + x_3 y_3) \cdot (x_1 + x_2 + x_3)}{3 \cdot (x_1^2 + x_2^2 + x_3^2) - (x_1 + x_2 + x_3)^2}
\end{aligned}$$

$$\begin{aligned}
B &= \frac{3S - QU}{-U^2 + 3V} \\
&= \frac{3\,(x_1 y_1 + x_2 y_2 + x_3 y_3) - (y_1 + y_2 + y_3) \cdot (x_1 + x_2 + x_3)}{3 \cdot (x_1^2 + x_2^2 + x_3^2) - (x_1 + x_2 + x_3)^2}
\end{aligned}$$

Seventh, given three pairs randomly drawn x and y points, we use the formulas just derived to find the parameters for the best-fitting y-intercept and slope values.

```
# Inputs X and Y
set.seed(123)
# Draw Randomly
mt_rnorm <- matrix(rnorm(6, mean = 1, sd = 1), nrow = 3, ncol = 2)
# # Three fixed and different set of points
# mt_rnorm <- matrix(c(
#   0.1915, 0.6221, 0.4377,
#   0.7854, 0.7800, 0.2726
# ), nrow = 3, ncol = 2)
# # Below, the 2nd and 3rd points are the same
# mt_rnorm <- matrix(c(
#   0.1915, 0.6221, 0.6221,
#   0.7854, 0.7800, 0.7800
# ), nrow = 3, ncol = 2)

colnames(mt_rnorm) <- c("x", "y")
x1 <- mt_rnorm[1, 1]
x2 <- mt_rnorm[2, 1]
x3 <- mt_rnorm[3, 1]
```

```r
y1 <- mt_rnorm[1, 2]
y2 <- mt_rnorm[2, 2]
y3 <- mt_rnorm[3, 2]

# X quadratic
x11 <- x1
x12 <- x1**2
x21 <- x2
x22 <- x2**2
x31 <- x3
x32 <- x3**2

# Define U and V, as well as Q and S
fl_U <- x11 + x21 + x31
fl_V <- x12 + x22 + x32
fl_Q <- y1 + y2 + y3
fl_S <- x11*y1 + x21*y2 + x31*y3

# Shared denominator
fl_denominator <- (3*fl_V - fl_U^2)

# Solve for A and B coefficients (not exact fit)
fl_A <- (fl_Q * fl_V - fl_S * fl_U) / fl_denominator

fl_B <- (3 * fl_S - fl_Q * fl_U) / fl_denominator

# Display
st_display <- paste0(
  "A(intercept)=", round(fl_A, 3),
  ", B(lin)=", round(fl_B, 3)
)
print(st_display)
```

```
## [1] "A(intercept)=0.617, B(lin)=0.813"
```

Eighth, to check that the estimates are correct, we derive results from running linear estimation with the three points of data drawn. We use both polynomial and orthogonal polynomials below.

```r
# Estimation results
df_rnorm <- as_tibble(mt_rnorm)
# Linear and quadratic terms
rs_lm_quad <- stats::lm(y ~ x, data = df_rnorm)
print(stats::summary.lm(rs_lm_quad))
```

```
##
## Call:
## stats::lm(formula = y ~ x, data = df_rnorm)
##
## Residuals:
##        1        2        3
##  0.09601 -0.11374  0.01773
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.61718    0.14533   4.247   0.1472
## x            0.81297    0.09296   8.746   0.0725 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.1499 on 1 degrees of freedom
## Multiple R-squared:  0.9871, Adjusted R-squared:  0.9742
## F-statistic: 76.48 on 1 and 1 DF,  p-value: 0.07248
```

```
# Using orthogonal polynomials
# vs. rs_lm_quad: different parameters, but same predictions
rs_lm_quad_otho <- stats::lm(y ~ poly(x, 1), data = df_rnorm)
print(stats::summary.lm(rs_lm_quad_otho))
```

```
##
## Call:
## stats::lm(formula = y ~ poly(x, 1), data = df_rnorm)
##
## Residuals:
##        1        2        3
##  0.09601 -0.11374  0.01773
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.63829    0.08654  18.931   0.0336 *
## poly(x, 1)   1.31089    0.14989   8.746   0.0725 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1499 on 1 degrees of freedom
## Multiple R-squared:  0.9871, Adjusted R-squared:  0.9742
## F-statistic: 76.48 on 1 and 1 DF,  p-value: 0.07248
```

Ninth, now we compare between the predications based on analytical solutions and lm regression. Again, note that the fit is not exact.

```
# Matrix of input values
mt_vals_xs <- t(
  matrix(c(1, x1, 1, x2, 1, x3),
    nrow = 2, ncol = 3
  )
)

# Predictions from LM poly prediction
ar_pred_lm <- mt_vals_xs %*% as.vector(rs_lm_quad$coefficients)
as_pred_lm_otho <- stats::predict(rs_lm_quad_otho)

# Predictions based on analytical solutions
ar_pred_sym <- mt_vals_xs %*% c(fl_A, fl_B)

# Combine results
kable(
  cbind(
    df_rnorm, ar_pred_sym,
    ar_pred_lm, as_pred_lm_otho
  ) %>%
    mutate(res = ar_pred_lm - y),
  caption = paste0(
    "Linear Fit of 3 Sets of Random (X,Y) Points"
  )
) %>% kable_styling_fc()
```

Linear Fit of 3 Sets of Random (X,Y) Points

x	y	ar_pred_sym	ar_pred_lm	as_pred_lm_otho	res
0.4395244	1.070508	0.9744999	0.9744999	0.9744999	-0.0960085
0.7698225	1.129288	1.2430232	1.2430232	1.2430232	0.1137354
2.5587083	2.715065	2.6973381	2.6973381	2.6973381	-0.0177269

8.1.2 Rescale a Parameter with Fixed Min and Max

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

8.1.2.1 Rescale a Fraction Between 0 and 1

We have ratio $0 < \theta < 1$, we want to multiply θ to get closer to 0 or to 1, but to not exceed the bounds of 0 and 1. This is achieved by the following $\widehat{\theta}\left(\theta, \lambda\right)$ function, where we adjust λ between negative and positive infinities to move θ closer to 0 (as $\lambda \rightarrow -\infty$) or θ closer to 1 (as $\lambda \rightarrow \infty$).

$$\widehat{\theta}\left(\theta, \lambda\right) = \theta \cdot \left(\frac{\exp\left(\lambda\right)}{1 + \theta \cdot \left(\exp\left(\lambda\right) - 1\right)}\right)$$

Given this function form, when $\lambda = 0$, we are back to θ, and $0 \le \widehat{\theta}\left(\lambda\right) \le 1$, which allows $\widehat{\theta}$ to be used as the "chance of success" parameter as we freely vary λ.

$$\begin{aligned}
\widehat{\theta}\left(\theta, \lambda = 0\right) &= \theta \cdot \left(\frac{1}{1 + \theta \cdot \left(1 - 1\right)}\right) = \theta \\
\lim_{\lambda \rightarrow -\infty} \widehat{\theta}\left(\theta, \lambda\right) &= \theta \cdot \left(\frac{0}{1 + \theta \cdot \left(0 - 1\right)}\right) = 0 \\
\lim_{\lambda \rightarrow \infty} \widehat{\theta}\left(\theta, \lambda\right) &= \theta \cdot \left(\frac{\exp(\lambda)}{\theta \cdot \exp(\lambda)}\right) = 1
\end{aligned}$$

To test this, first, we write out the rescaling function.

```
# Construct the formula
ffi_theta_lambda_0t1 <- function(theta, lambda) {
  if (is.finite(exp(lambda))) {
    theta * (exp(lambda) / (1 + theta * (exp(lambda) - 1)))
  } else {
    # If lambda is large, exp(lambda)=inf, ratio above becomes 1
    1
  }
}
# Test the function
print(ffi_theta_lambda_0t1(0.5, 1e1))
```

```
## [1] 0.9999546
```

```
print(ffi_theta_lambda_0t1(0.5, 1e2))
```

```
## [1] 1
```

```
print(ffi_theta_lambda_0t1(0.5, -1e3))
```

```
## [1] 0
```

Second, given theta, we evaluate the function with differing lambda values.

```
# Create Function
ffi_fixtheta_varylambda_0t1 <-
  function(ar_lambda_pos =
```

Theta-hat rescaling , given different lambda rescalers.

theta_hat	lambda	theta
0.00005	-10.0000000	0.5
0.01340	-4.2986623	0.5
0.13613	-1.8478498	0.5
0.31124	-0.7943282	0.5
0.50000	0.0000000	0.5
0.68876	0.7943282	0.5
0.86387	1.8478498	0.5
0.98660	4.2986623	0.5
0.99995	10.0000000	0.5

```
             1e1^(seq(-0.1, 1, length.out = 4)),
           theta = 0.5) {

    # Construct lambda vector
    ar_lambda <- sort(unique((c(-ar_lambda_pos, 0, ar_lambda_pos))))
    # sapply
    ar_theta_hat <- sapply(ar_lambda, ffi_theta_lambda_0t1, theta = theta)
    # Create table
    ar_st_varnames <- c("theta_hat", "lambda")
    tb_theta_hat_lambda <- as_tibble(
      cbind(round(ar_theta_hat, 5), ar_lambda)
    ) %>%
      rename_all(~ c(ar_st_varnames)) %>%
      mutate(theta = theta)
    # return
    return(tb_theta_hat_lambda)
  }

# Test function
tb_theta_hat_lambda <- ffi_fixtheta_varylambda_0t1()
# Print
kable(tb_theta_hat_lambda,
  caption = paste(
    "Theta-hat rescaling",
    ", given different lambda rescalers.",
    separator = " "
  )
) %>% kable_styling_fc()
```

Third, we run the function we just created for two three different θ levels, and we stack the results together.

```
# Evaluate at differing thetas
ar_lambda_pos <- 1e1^(seq(-0.1, 1, length.out = 2))
tb_theta_hat_lambda_low <- ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.1)
tb_theta_hat_lambda_mid <- ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.5)
tb_theta_hat_lambda_hgh <- ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.9)
# Combine
tb_theta_hat_lambda_combo <- bind_rows(
  tb_theta_hat_lambda_low,
  tb_theta_hat_lambda_mid,
  tb_theta_hat_lambda_hgh
)
# Print
```

Theta-hat rescaling , with multiple theta values , given different lambda rescalers.

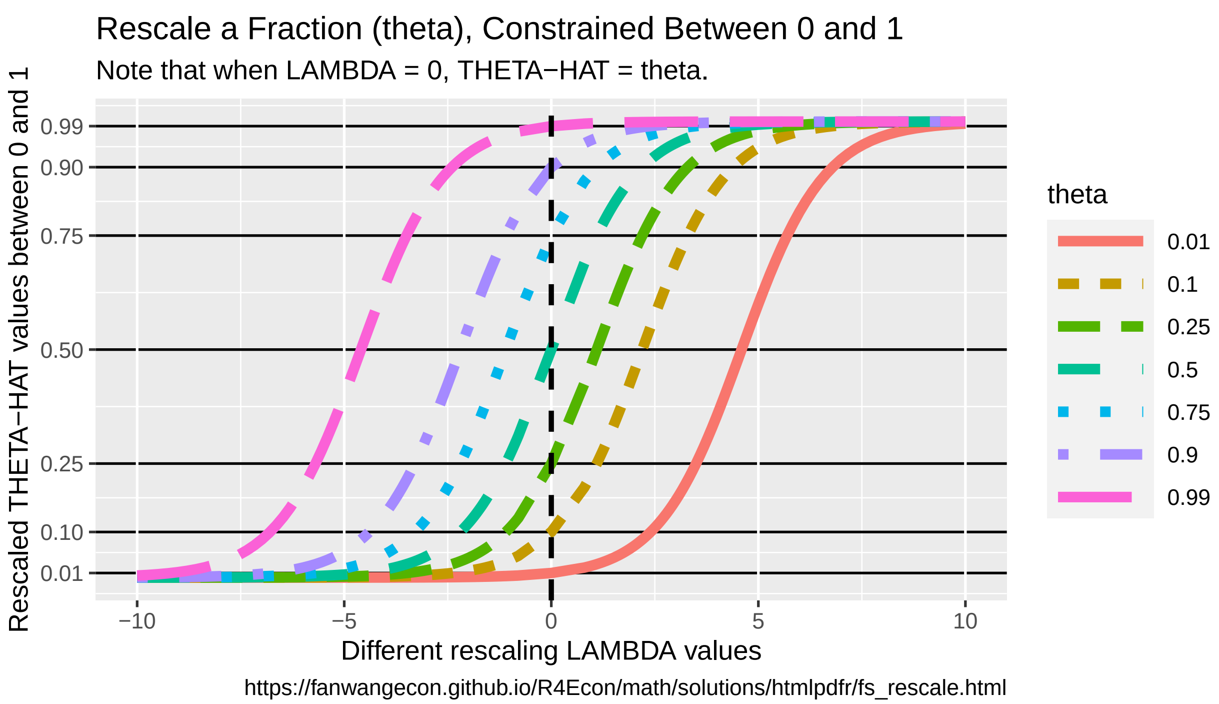
theta_hat	lambda	theta
0.00001	-10.0000000	0.1
0.04781	-0.7943282	0.1
0.10000	0.0000000	0.1
0.19736	0.7943282	0.1
0.99959	10.0000000	0.1
0.00005	-10.0000000	0.5
0.31124	-0.7943282	0.5
0.50000	0.0000000	0.5
0.68876	0.7943282	0.5
0.99995	10.0000000	0.5
0.00041	-10.0000000	0.9
0.80264	-0.7943282	0.9
0.90000	0.0000000	0.9
0.95219	0.7943282	0.9
0.99999	10.0000000	0.9

```r
kable(tb_theta_hat_lambda_combo,
  caption = paste(
    "Theta-hat rescaling",
    ", with multiple theta values",
    ", given different lambda rescalers.",
    separator = " "
  )
) %>% kable_styling_fc()
```

Fourth, we visualize the results from above. We generate a denser x-grid for this purpose, and we evaluate at 9 different theta values from $\theta = 0.01$ to $\theta = 0.99$. We can see in the graph that all $0 < \widehat{\theta} < 1$.

```r
# Generate a denser result
ar_lambda_pos <- 1e1^(seq(-0.1, 1, length.out = 100))
tb_theta_hat_lambda_combo <- bind_rows(
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.01),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.10),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.25),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.50),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.75),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.90),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.99)
)
# Labeling
st_title <- paste0("Rescale a Fraction (theta), Constrained Between 0 and 1")
st_subtitle <- paste0(
  "Note that when LAMBDA = 0, THETA-HAT = theta."
)
st_caption <- paste0(
  "https://fanwangecon.github.io/",
  "R4Econ/math/solutions/htmlpdfr/fs_rescale.html"
)
st_x_label <- "Different rescaling LAMBDA values"
st_y_label <- "Rescaled THETA-HAT values between 0 and 1"
ar_y_breaks <- c(0.01, 0.10, 0.25, 0.50, 0.75, 0.90, 0.99)
# Graph
tb_theta_hat_lambda_combo %>%
  mutate(theta = as.factor(theta)) %>%
```

```
ggplot(aes(
  x = lambda, y = theta_hat,
  color = theta, linetype = theta
)) +
geom_line(size = 2) +
geom_vline(
  xintercept = 0,
  linetype = "dashed", color = "black", size = 1
) +
labs(
  title = st_title,
  subtitle = st_subtitle,
  x = st_x_label,
  y = st_y_label,
  caption = st_caption
) +
scale_y_continuous(
  breaks = ar_y_breaks,
  limits = c(0, 1)
) +
theme(
  panel.grid.major.y = element_line(
    color = "black",
    size = 0.5,
    linetype = 1
  ),
  legend.key.width = unit(3, "line")
)
```



8.1.2.2 Fit Three Points with Ends Along 45 Degree Line

Given e < x < f, use f(x) to rescale x, such that f(e)=e, f(f)=f, but f(z)=$\alpha \cdot z$ for one particular z between e and f, with $\alpha > 1$. And in this case, assume that $\alpha \cdot z < f$. Note that this is case where we have three points, and the starting and the ending points are along the 45 degree line.

We can fit these three points using the Quadratic function exactly. In another word, there is a unique quadratic function that crosses these three points. Note the quadratic function is either concave or convex through the entire domain.

First, as an example, suppose that $e = 0$, $f = 10$, $z = 2$, and $\alpha = 1.5$. Using a quadratic to fit:

$$y(x) = a \cdot x^2 + b \cdot x + c$$

We have three equations:

$$0 = a \cdot 0 + b \cdot 0 + c2 \cdot 1.5 = a \cdot 2^2 + b \cdot 2 + c10 = a \cdot 10^2 + b \cdot 10 + c$$

Given these, we have, $c = 0$, and subsequently, 2 equations and 2 unknowns:

$$3 = a \cdot 4 + b \cdot 210 = a \cdot 100 + b \cdot 10$$

Hence:

$$a = \frac{3 - 2b}{4} 10 = \frac{3 - 2b}{4} \cdot 100 + b \cdot 1010 = 75 - 50b + 10b$$

And finally:

$$a = \frac{3 - 2 * 1.625}{4} = -0.0625b = \frac{65}{40} = 1.625c = 0$$

Generate the a, b and c points above for the quadratic function:

```
# set values
e <- 0
f <- 10
z <- 2
alpha <- 1.5
# apply formulas from above
a <- -0.0625
b <- 1.625
c <- 0
# grid of values beween a and b, 11 points covering z = 2
ar_x <- seq(e, f, length.out = 11)
# rescale
ar_grid_quad <- a * ar_x^2 + b * ar_x + c
# show values
kable(print(as_tibble(cbind(ar_x, ar_grid_quad))),
  caption = paste0(
    "Quadratic Fit of Three Equations and Three Unknowns\n",
    "Satisfies: f(0)=0, f(10)=10, f(2)=3"
  )
) %>%
  kable_styling_fc()
```

Second, as another example, suppose that $e = 0$, $f = 3.5$, $z = 0.5$, and $\alpha = 1.5$. Using a quadratic to fit these, we have three equations:

$$0 = a \cdot 0 + b \cdot 0 + c0.75 = a \cdot 0.5^2 + b \cdot 0.5 + c3.5 = a \cdot 3.5^2 + b \cdot 3.5 + c$$

Given these, we have, $c = 0$, and subsequently, 2 equations and 2 unknowns:

$$0.75 = a \cdot 0.25 + b \cdot 0.53.5 = a \cdot 12.25 + b \cdot 3.5$$

Hence:

$$a = \frac{0.75 - 0.5b}{0.25} 3.5 = \frac{0.75 - 0.5b}{0.25} \cdot 12.25 + b \cdot 3.53.5 = 36.75 - 24.5b + 3.5b$$

And finally:

$$a = \frac{0.75 - 0.5 * 1.58333}{0.25} = -0.1666b = \frac{36.75 - 3.5}{24.5 - 3.5} = 1.58333c = 0$$

Quadratic Fit of Three Equations and Three Unknowns Satisfies: f(0)=0, f(10)=10, f(2)=3

ar_x	ar_grid_quad
0	0.0000
1	1.5625
2	3.0000
3	4.3125
4	5.5000
5	6.5625
6	7.5000
7	8.3125
8	9.0000
9	9.5625
10	10.0000

Generate the a, b and c points above for the quadratic function:

```r
# set values
e <- 0
f <- 3.5
z <- 0.5
alpha <- 1.5
# apply formulas from above
a <- -0.16666666
b <- 1.583333333
c <- 0
# grid of values beween a and b, 11 points covering z = 2
ar_x <- seq(e, f, length.out = 100000)
# rescale
ar_grid_quad <- a * ar_x^2 + b * ar_x + c
# show values
# cbind(ar_x, ar_grid_quad)
ar_x[which.min(abs(ar_grid_quad - 0.75))]
```

```
## [1] 0.500015
```

The exercises above are special cases of the formula we derive on this page: Formulas for Quadratic Parameters and Three Points.

8.1.3 Log with Different Bases and Exponents

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

8.1.3.1 Log of Bases that Are not 10, 2 and e

What is y below, with arbitrary base x? It is $y = \frac{\log(z)}{\log(x)}$, because:

$$\begin{aligned}
x^y &= z \\
x^{\frac{\log(z)}{\log(x)}} &= z \\
\log\left(x^{\frac{\log(z)}{\log(x)}}\right) &= \log(z) \\
\frac{\log(z)}{\log(x)} \log(x) &= \log(z) \\
\frac{\log(z)}{\log(x)} &= \frac{\log(z)}{\log(x)}
\end{aligned}$$

Given these, we can compute the exponents, y, for non-standard bases, x, given the value for z.

```r
# base 1.1
x <- 1.1
y <- 5.5
z <- x^y
# given z and knowing x, and what is y?
y_solved <- log(z) / log(x)
# dispaly
print(paste0("y_solved=", y_solved, ", y=", y))
```

```
## [1] "y_solved=5.5, y=5.5"
```

8.1.3.2 Rescale Bounded Model Parameters to Unconstrained with Exponentiation

We have a parameter to be estimated, the parameter's values can range between positive 1 and negative infinity. We want to use an estimator that us unconstrained. Use exponentiation to rescale the parameter so that is become unconstrained, use different bases so that the speed at which the parameter value approaches its bounds can be controlled.

While y is not bounded, $f(y;x)$ is bounded:

$$
\begin{aligned}
&f(y;x) = 1 - x^y \\
&\text{where } x > 1 \text{ and } -\infty < y < \infty \\
&\text{then, } 1 > f(y;x) > -\infty
\end{aligned}
$$

With $x > 1$, as y increases $f(y;x)$ decreases:

$$\frac{df(y;x)}{dy} = -x^y \log(x) < 0 \text{ when } x > 1$$

x controls the speed at which $f(y)$ approaches its bounds. In the simulation below, we try a number of different bases, at higher bases (2, e=2.71, 10), as y value changes $f(y)$ shifts too quickly to the bounds. But a base value of $x = 1.03$ or $x = 1.04$ would work well in an unbounded estimation routine that still generates parameters within bounds, which is below 1 in the case here.

```r
# Vector of unbounded values, high and low
ar_y_vals <- sort(rnorm(20, 0, 20))
# Different base values
ar_bases <- c(1.01, 1.02, 1.03, 1.04, 1.1, 2, 2.71, 10)
# Transform back to f(y) scale with different bases
mt_f_of_y_vary_x <- matrix(NA,
  nrow = length(ar_y_vals),
  ncol = 1 + length(ar_bases)
)
ar_st_varnames <- c("yvalidx", "y_vals", paste0("base", ar_bases))
mt_f_of_y_vary_x[, 1] <- ar_y_vals
for (it_base in seq(1, length(ar_bases))) {
  fl_base <- ar_bases[it_base]
  ar_f_y <- 1 - fl_base^ar_y_vals
  mt_f_of_y_vary_x[, 1 + it_base] <- ar_f_y
}
# To tibble
tb_f_of_y_vary_x <- as_tibble(mt_f_of_y_vary_x) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~ c(ar_st_varnames))
# Print
kable(tb_f_of_y_vary_x) %>% kable_styling_fc_wide()
```

8.1.3.3 Positive Exponents

Define exponents to consider and x-values to consider.

yvalidx	y_vals	base1.01	base1.02	base1.03	base1.04	base1.1	base2	base2.71	base10
1	-39.332343	0.3238699	0.5410820	0.6873331	0.7861847	0.9764534	1.000000e+00	1.000000e+00	1.000000e+00
2	-33.733866	0.2851361	0.4872768	0.6310642	0.7336826	0.9598519	1.000000e+00	1.000000e+00	1.000000e+00
3	-25.301225	0.2225652	0.3940942	0.5266281	0.6292888	0.9103161	1.000000e+00	1.000000e+00	1.000000e+00
4	-21.356474	0.1914429	0.3448652	0.4680851	0.5672591	0.8693835	9.999996e-01	1.000000e+00	1.000000e+00
5	-20.520089	0.1846858	0.3339241	0.4547709	0.5528282	0.8585450	9.999993e-01	1.000000e+00	1.000000e+00
6	-14.577825	0.1350246	0.2507475	0.3500781	0.4354649	0.7507790	9.999591e-01	9.999995e-01	1.000000e+00
7	-13.737057	0.1277579	0.2381685	0.3337238	0.4165387	0.7299860	9.999268e-01	9.999989e-01	1.000000e+00
8	-12.500785	0.1169619	0.2192876	0.3089259	0.3875511	0.6962202	9.998275e-01	9.999961e-01	1.000000e+00
9	-11.116823	0.1047176	0.1975954	0.2800690	0.3533886	0.6533870	9.995497e-01	9.999846e-01	1.000000e+00
10	-9.455828	0.0897979	0.1707638	0.2438405	0.3098624	0.5939328	9.985760e-01	9.999195e-01	1.000000e+00
11	-8.913239	0.0848705	0.1618059	0.2316152	0.2950184	0.5723809	9.979258e-01	9.998617e-01	1.000000e+00
12	-4.359498	0.0424511	0.0827081	0.1209043	0.1571638	0.3399928	9.512853e-01	9.870440e-01	9.999563e-01
13	2.213654	-0.0222710	-0.0448112	-0.0676212	-0.0907015	-0.2348923	-3.638487e+00	-8.087498e+00	-1.625514e+02
14	7.196276	-0.0742313	-0.1531591	-0.2370300	-0.3261011	-0.9855152	-1.456544e+02	-1.304470e+03	-1.571363e+07
15	8.015429	-0.0830230	-0.1720174	-0.2673479	-0.3693975	-1.1467434	-2.577525e+02	-2.953164e+03	-1.036165e+08
16	9.218324	-0.0960638	-0.2002706	-0.3132206	-0.4355517	-1.4075271	-5.946513e+02	-9.799568e+03	-1.653195e+09
17	9.957010	-0.1041497	-0.2179571	-0.3422097	-0.4777505	-1.5831365	-9.929363e+02	-2.046719e+04	-9.057525e+09
18	14.027118	-0.1497844	-0.3201875	-0.5138027	-0.7335192	-2.8073261	-1.669388e+04	-1.183889e+06	-1.064432e+14
19	24.481636	-0.2758344	-0.6238514	-1.0619411	-1.6121855	-9.3124213	-2.342646e+07	-3.979207e+10	-3.031349e+24
20	35.738263	-0.4270474	-1.0293418	-1.8759419	-3.0620190	-29.1510650	-5.731774e+10	-2.975571e+15	-5.473470e+35

```r
# positive value exponents
ar_exponents_posv <- c(0.05, 0.5, 1, 1.5)
# positive and negative values of the base
ar_baseval_pos <- seq(1e-10, 1.5, length.out = 1000)
# base to power
mt_x2a_val <- matrix(data = NA, nrow = length(ar_exponents_posv), ncol = length(ar_baseval_pos))
# Generate values
it_row_ctr <- 0
for (fl_exponents_posv in ar_exponents_posv) {
  it_row_ctr <- it_row_ctr + 1
  mt_x2a_val[it_row_ctr, ] <- ar_baseval_pos^fl_exponents_posv
}
```

Note that the smaller exponents functions are higher when $x < 1$, but lower when $x > 1$.

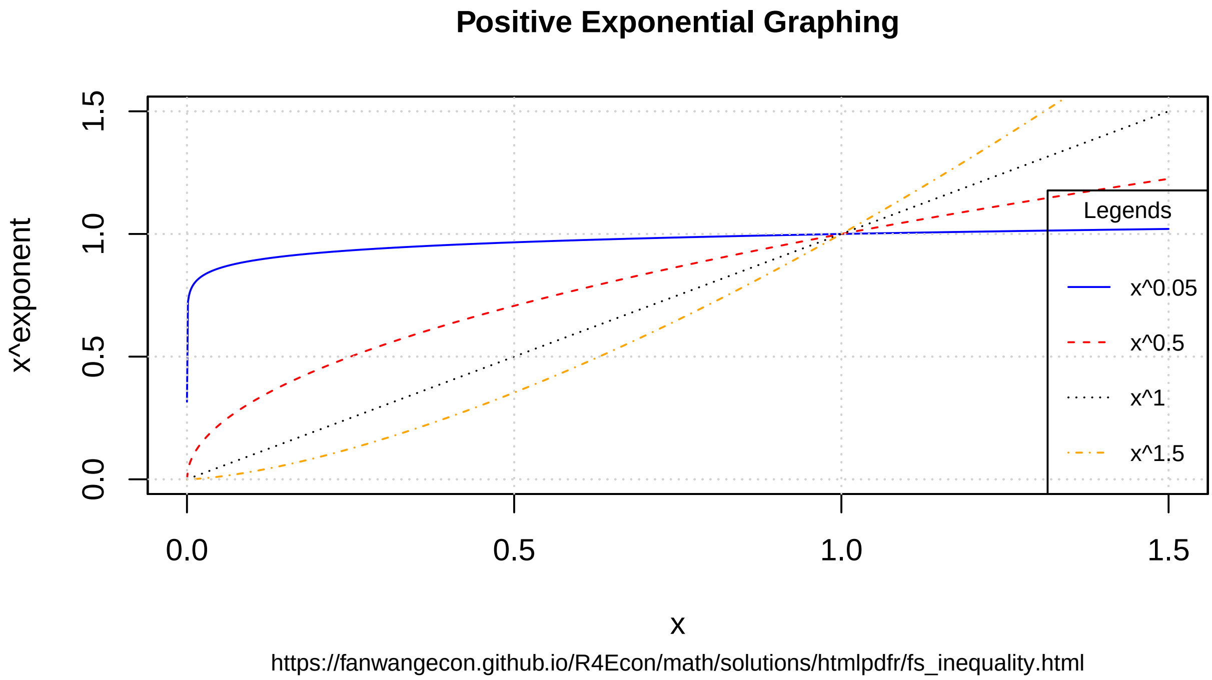
$$\text{if } b > a > 0, \text{ then, } \left(x^a - x^b\right) > 0, \text{ for all } 1 > x > 0$$

$$\text{if } b > a > 0, \text{ then, } \left(x^a - x^b\right) < 0, \text{ for all } x > 1$$

Note we also have: $\lim_{a\to 0} x^a = 1$ and $\lim_{a\to 1} x^a = x$ bounds. When $a > 1$, function becomes convex.

```r
# x and bounds
ar_xlim <- c(min(ar_baseval_pos), max(ar_baseval_pos))
ar_ylim <- c(0, 1.5)
# function line
st_line_1_y_legend <- paste0("x^", ar_exponents_posv[1])
st_line_2_y_legend <- paste0("x^", ar_exponents_posv[2])
st_line_3_y_legend <- paste0("x^", ar_exponents_posv[3])
st_line_4_y_legend <- paste0("x^", ar_exponents_posv[4])
# Color and line
st_point_1_pch <- 10
st_point_1_cex <- 2
ar_colors <- c("blue", "red", "black", "orange")
ar_ltys <- c("solid", "dashed", "dotted", "dotdash")
# Graph and combine
for (it_graph in c(1, 2, 3, 4)) {
  if (it_graph != 1) {
    par(new = T)
  }
  ar_y_current <- mt_x2a_val[it_graph, ]
  plot(ar_baseval_pos, ar_y_current,
    type = "l",
```

```
    col = ar_colors[it_graph], lty = ar_ltys[it_graph],
    pch = 10, cex = 2, xlim = ar_xlim, ylim = ar_ylim, panel.first = grid(),
    ylab = "", xlab = "", yaxt = "n", xaxt = "n", ann = FALSE
  )
  plot_line <- recordPlot()
}
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg <- 1
fl_ces_fig_small <- 0.75
# R Legend
st_title <- paste0("Positive Exponential Graphing")
st_subtitle <- paste0(
  "https://fanwangecon.github.io/",
  "R4Econ/math/solutions/htmlpdfr/fs_inequality.html"
)
st_x_label <- "x"
st_y_label <- "x^exponent"
title(
  main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
  cex.lab = fl_ces_fig_reg,
  cex.main = fl_ces_fig_reg,
  cex.sub = fl_ces_fig_small
)
axis(1, cex.axis = fl_ces_fig_reg)
axis(2, cex.axis = fl_ces_fig_reg)
grid()
# Legend sizing CEX
legend("bottomright",
  inset = c(0, 0),
  xpd = TRUE,
  c(st_line_1_y_legend, st_line_2_y_legend, st_line_3_y_legend, st_line_4_y_legend),
  col = c(ar_colors[1], ar_colors[2], ar_colors[3], ar_colors[4]),
  cex = fl_ces_fig_small,
  lty = c(ar_ltys[1], ar_ltys[2], ar_ltys[3], ar_ltys[4]),
  title = "Legends",
  y.intersp = 2
)
```



8.1.3.4 Negative Exponents

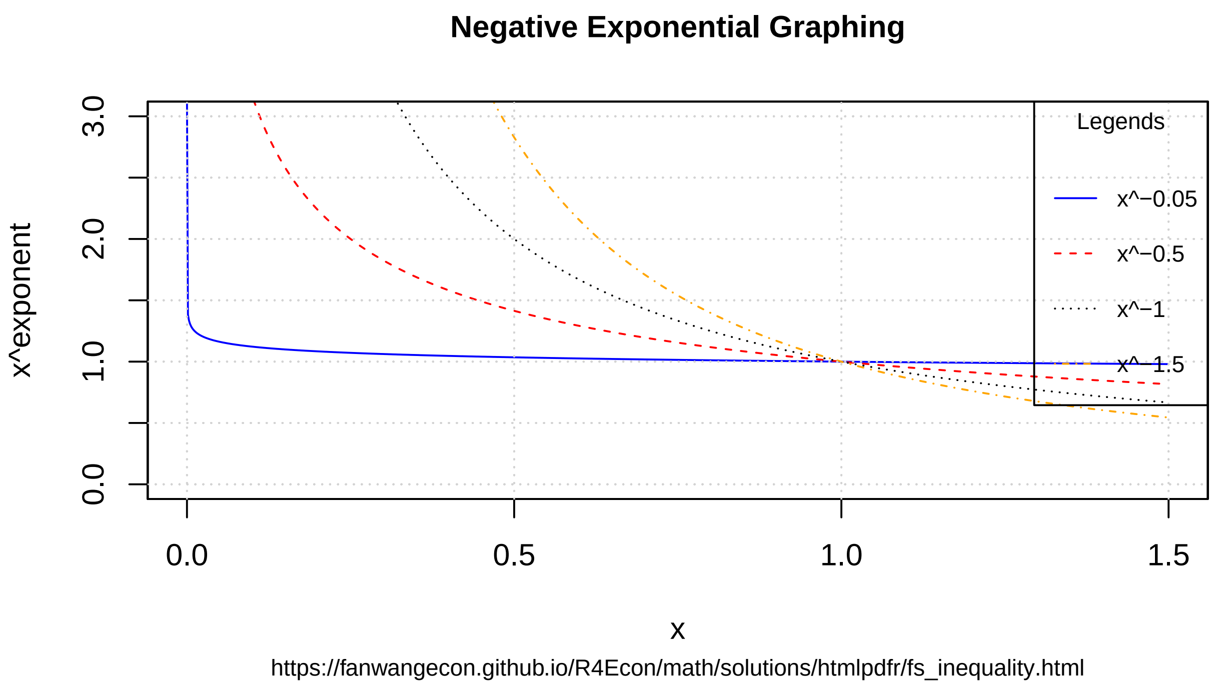
Similar to above, but now with negative exonents.

```
# positive value exponents
ar_exponents_posv <- -c(0.05, 0.5, 1, 1.5)
# positive and negative values of the base
ar_baseval_pos <- seq(1e-10, 1.5, length.out = 1000)
# base to power
mt_x2a_val <- matrix(data = NA, nrow = length(ar_exponents_posv), ncol = length(ar_baseval_pos))
# Generate values
it_row_ctr <- 0
for (fl_exponents_posv in ar_exponents_posv) {
  it_row_ctr <- it_row_ctr + 1
  mt_x2a_val[it_row_ctr, ] <- ar_baseval_pos^fl_exponents_posv
}
```

For positive exponents, when $x < 1$, $x^a < 1$, when $x > 1$, $x^a > 1$. For negative exponents, when $x < 1$, $x^a > 1$, and when $x > 1$, $x^a < 1$. Large positive exponents generate small values when $x < 1$, and large negative exponents generate very large values when $x < 1$.

```
# x and bounds
ar_xlim <- c(min(ar_baseval_pos), max(ar_baseval_pos))
ar_ylim <- c(0, 3)
# function line
st_line_1_y_legend <- paste0("x^", ar_exponents_posv[1])
st_line_2_y_legend <- paste0("x^", ar_exponents_posv[2])
st_line_3_y_legend <- paste0("x^", ar_exponents_posv[3])
st_line_4_y_legend <- paste0("x^", ar_exponents_posv[4])
# Color and line
st_point_1_pch <- 10
st_point_1_cex <- 2
ar_colors <- c("blue", "red", "black", "orange")
ar_ltys <- c("solid", "dashed", "dotted", "dotdash")
# Graph and combine
for (it_graph in c(1, 2, 3, 4)) {
  if (it_graph != 1) {
    par(new = T)
```

```
  }
  ar_y_current <- mt_x2a_val[it_graph, ]
  plot(ar_baseval_pos, ar_y_current,
    type = "l",
    col = ar_colors[it_graph], lty = ar_ltys[it_graph],
    pch = 10, cex = 2, xlim = ar_xlim, ylim = ar_ylim, panel.first = grid(),
    ylab = "", xlab = "", yaxt = "n", xaxt = "n", ann = FALSE
  )
  plot_line <- recordPlot()
}
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg <- 1
fl_ces_fig_small <- 0.75
# R Legend
st_title <- paste0("Negative Exponential Graphing")
st_subtitle <- paste0(
  "https://fanwangecon.github.io/",
  "R4Econ/math/solutions/htmlpdfr/fs_inequality.html"
)
st_x_label <- "x"
st_y_label <- "x^exponent"
title(
  main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
  cex.lab = fl_ces_fig_reg,
  cex.main = fl_ces_fig_reg,
  cex.sub = fl_ces_fig_small
)
axis(1, cex.axis = fl_ces_fig_reg)
axis(2, cex.axis = fl_ces_fig_reg)
grid()
# Legend sizing CEX
legend("topright",
  inset = c(0, 0),
  xpd = TRUE,
  c(st_line_1_y_legend, st_line_2_y_legend, st_line_3_y_legend, st_line_4_y_legend),
  col = c(ar_colors[1], ar_colors[2], ar_colors[3], ar_colors[4]),
  cex = fl_ces_fig_small,
  lty = c(ar_ltys[1], ar_ltys[2], ar_ltys[3], ar_ltys[4]),
  title = "Legends",
  y.intersp = 2
)
```



8.1.3.5 Inequality and Exponents

Suppose we have the inequality $0 < a < b$, if we apply positive exponents to them, the direction of the inequality will stay the same: If $0 < a < b$, then $0 < a^{|\alpha|} < b^{|\alpha|}$ if $\alpha < 0$. Think about the graphs above, think of a and b as points along the x-axis, note that positive exponents are strictly increasing (although some concavely and some convexly) along the x-axis. Comparing x^{α} at $0 < b < a$ anywhere along the x-axis has still has $b^{\alpha} < a^{\alpha}$.

In contrast, if $0 < a < b$, then $a^{-|\alpha|} > b^{-|\alpha|} > 0$ if $\alpha < 0$. Sign flips. Visually from above, the sign-flipping happens because negative exponential is strictly decreasing along $x > 0$.

8.1.4 Find Nearest

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

8.1.4.1 Neart Point Along a Line Through Orgin to Another Point

We first have X_1 an Y_1, given these, we are able to generate $R = \frac{Y_2}{X_2}$, a ratio. We want to iteratively update X_1 and Y_1, where 1 subscript indicates the first iteration, but we only know the ratio. Think of R as a line through the origin with R as the slope.

We generate X_2 and Y_2 by finding the point along the R as slope origin line that is the closest to the X_1 and Y_1. At the resulting point, R will be respected, and it will differ least in distance to the earlier iteration's X_1 and Y_1 points.

1. The slope of the diagonal line is $-\frac{X_2}{Y_2} = -\frac{1}{R}$
2. The diagonal line must cross X_1 and Y_1, solve for this line's y-intercept
3. Solve for the intersection of the diagonal line and the origin line with R as slope

Implementing step (2):

$$
\begin{aligned}
Y_1 &= I - \frac{1}{R} \cdot X_1 \\
I &= Y_1 + \frac{X_1}{R} \\
I &= \frac{Y_1 \cdot R + X_1}{R}
\end{aligned}
$$

Implementing step (3):

$$
\begin{aligned}
Y &= \frac{Y_1 \cdot R + X_1}{R} - \frac{1}{R} \cdot X \\
Y &= R \cdot X \\
R \cdot X &= \frac{Y_1 \cdot R + X_1}{R} - \frac{1}{R} \cdot X \\
\left(R + \frac{1}{R}\right) \cdot X &= \frac{Y_1 \cdot R + X_1}{R} \\
\frac{R^2 + 1}{R} \cdot X &= \frac{Y_1 \cdot R + X_1}{R} \\
X &= \frac{Y_1 \cdot R + X_1}{R} \cdot \frac{R}{R^2 + 1}
\end{aligned}
$$

And we have:

$$
\begin{aligned}
X &= \frac{Y_1 \cdot R + X_1}{R^2 + 1} \\
Y &= \frac{Y_1 \cdot R^2 + X_1 \cdot R}{R^2 + 1}
\end{aligned}
$$

Visualize the results:

```r
# Set random parameter Values for X1, Y1, and X2/Y2 ratio
set.seed(3)
fl_x1 <- runif(1) * 10
fl_y1 <- runif(1) * 10
fl_r <- runif(1) * 5

# Diaganol
fl_diag_slope <- -1 / fl_r
fl_diag_yintercept <- (fl_y1 * fl_r + fl_x1) / fl_r

# Closest point
fl_x2 <- (fl_y1 * fl_r + fl_x1) / (fl_r^2 + 1)
fl_y2 <- (fl_y1 * fl_r^2 + fl_x1 * fl_r) / (fl_r^2 + 1)

# Print state
print(paste("x1=", fl_x1, "x2=", fl_x2, "R=", fl_r, sep = " "))
```

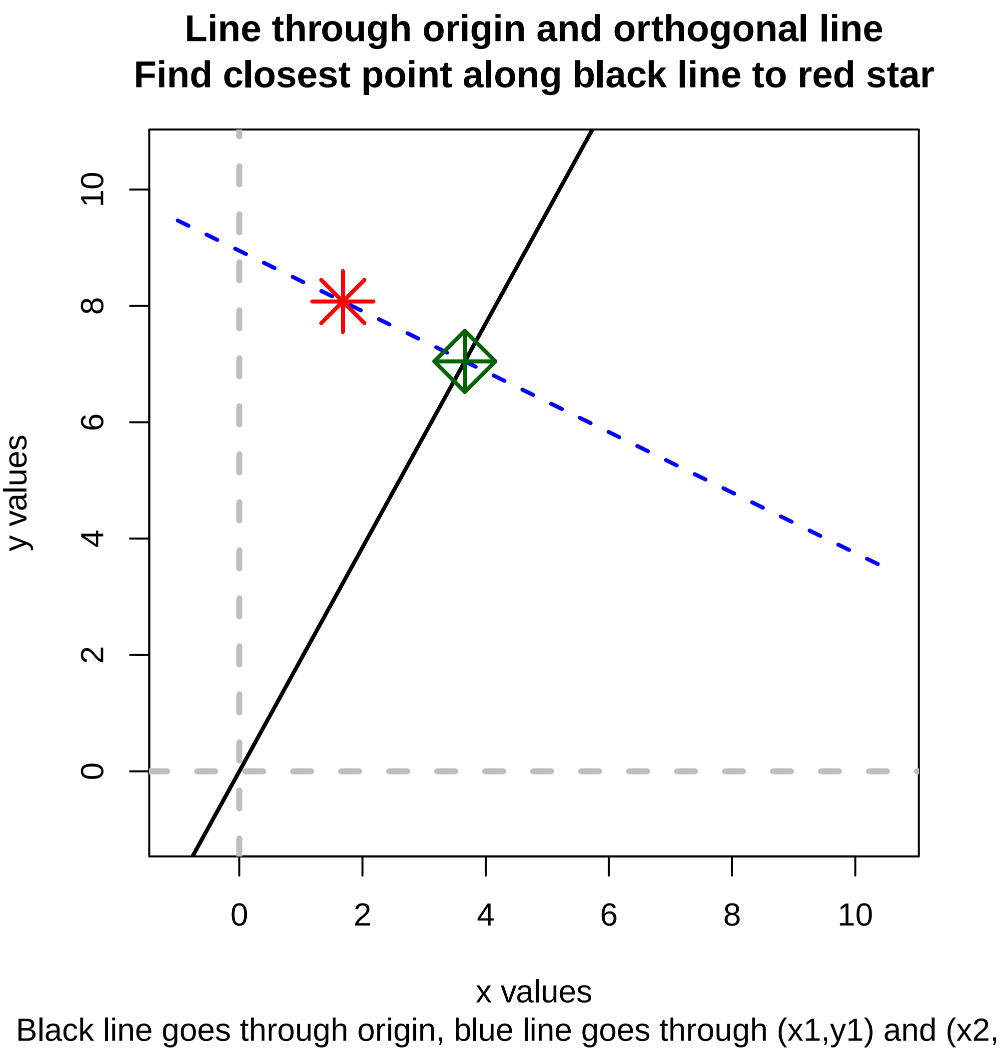
```
## [1] "x1= 1.68041526339948 x2= 3.6609038475849 R= 1.92471175687388"
```

```r
print(paste("x2=", fl_x2, "y2=", fl_y2, sep = " "))
```

```
## [1] "x2= 3.6609038475849 y2= 7.04618467623146"
```

```r
# X and y lims
ar_xylim <- c(-1, max(fl_x1, fl_y2) * 1.5)

# Visualize
par(mfrow = c(1, 1))
# Line through origin
curve(0 + fl_r * x, ar_xylim[1], ar_xylim[2],
  col = "black", lwd = 2, lty = 1,
  ylim = ar_xylim,
  ylab = "", xlab = ""
)
# Diaganol line
curve(fl_diag_yintercept + fl_diag_slope * x,
  add = TRUE,
  col = "blue", lwd = 2, lty = 2,
  ylim = ar_xylim,
```

```r
  ylab = "", xlab = ""
)
# Point
points(fl_x1, fl_y1,
  add = TRUE,
  pch = 8, col = "red", cex = 3, lwd = 2,
  ylab = "", xlab = ""
)
points(fl_x2, fl_y2,
  add = TRUE,
  pch = 9, col = "darkgreen", cex = 3, lwd = 2,
  ylab = "", xlab = ""
)
# Origin lines
abline(
  v = 0, h = 0,
  col = "gray", lwd = 3, lty = 2
)

# Titles
title(
  main = paste0(
    "Line through origin and orthogonal line\n",
    "Find closest point along black line to red star"
  ),
  sub = paste0(
    "Black line goes through origin,",
    " blue line goes through (x1,y1) and (x2, y2)"
  ),
  xlab = "x values", ylab = "y values"
)
```



8.1.5 Linear Scalar f(x)=0 Solutions

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

8.1.5.1 Ratio

Here are some common ratios.

8.1.5.1.1 Unif Draw Min and Max Ratio We want to draw numbers such that we have some mean b, and that the possible maximum and minimum value drawn are at most a times apart. Given b and a, solve for x.

$$f(x) = \frac{b+x}{b-x} - a = 0$$

$$b \cdot a - x \cdot a = b + xb \cdot a - b = x + x \cdot ab\,(a-1) = x\,(a+1)\,x = \frac{b\,(a-1)}{a+1}$$

Uniformly draw

```
b <- 100
a <- 2
x <- (b*(a-1))/(a+1)
ar_unif_draws <- runif(100, min=b-x, max=b+x)
```

```
fl_max_min_ratio <- max(ar_unif_draws)/min(ar_unif_draws)
cat('fl_max_min_ratio =', fl_max_min_ratio, 'is close to a =', a, '\n')
```

```
## fl_max_min_ratio = 1.976291 is close to a = 2
```

8.2 Production Functions

8.2.1 Nested CES Production Function

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

8.2.1.1 The Nested CES Problem

We have the following production function with four inputs x_1, x_2, y_1 and y_2. There are three ρ parameters ρ_x, ρ_y and ρ_o that correspond to inner-nest and outter nest elasticity of substitution between inputs.

The firm's expenditure minimization problem has the following objective:

$$\min_{x_1,x_2,y_1,y_2} \left(x_1 \cdot p_{x_1} + x_2 \cdot p_{x_2} + y_1 \cdot p_{y_1} + y_2 \cdot p_{y_2}\right)$$

The production quantity constraint is, using a constant-returns doubly-nested production function:

$$Y = Z \cdot \left(\beta_{o_1}\left(\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}\right)^{\rho_o} + \beta_{o_2}\left(\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{1}{\rho_y}}\right)^{\rho_o}\right)^{\frac{1}{\rho_o}}$$

Note that we are assuming constant-returns to scale in a competitive setting, so firms do not make profits. We solve for expenditure minimization rather than profit maximization.

8.2.1.1.1 Marginal Product of Labor A key object to consider is the marginal product of input (labor or capital). Taking derivative of output Y with respect to input x_1, we have:

$$\frac{\partial Y}{\partial x_1} = \left[\frac{1}{\rho_o} Z\left(\beta_{o_1}\left(\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}\right)^{\rho_o} + \beta_{o_2}\left(\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{1}{\rho_y}}\right)^{\rho_o}\right)^{\frac{1}{\rho_o}-1}\right] \cdot \left[\rho_o \beta_{o_1}\left(\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}\right)^{\rho_o -}\right.$$

What is the relationship between the marginal product of labor and the wage? Let λ be the lagrange multiplier for the overall problem:

$$p_{x_1} = \lambda \cdot \left(\frac{\partial Y}{\partial x_1}\right)$$

8.2.1.2 Denesting the Nested CES Problem

Rather than solving the problem above directly in an expenditure minimization, we can divide the problem above into three parts, the **X Problem**, the **Y Problem** and the **Z Problem**.

8.2.1.2.1 Three Denested Sub-problems, X, Y and O Problems **The X problem**:

$$\min_{x_1,x_2} \left(x_1 \cdot p_{x_1} + x_2 \cdot p_{x_2}\right)$$

$$O_x = \left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}$$

The Y problem:

$$\min_{y_1,y_2} \left(y_1 \cdot p_{y_1} + y_2 \cdot p_{y_2} \right)$$

$$O_y = \left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y} \right)^{\frac{1}{\rho_y}}$$

The O problem:

$$\min_{o_1,o_2} \left(O_x \cdot p_{o_1} + O_y \cdot p_{o_2} \right)$$

$$Y = Z \cdot \left(\beta_{o_1} O_x^{\rho_o} + \beta_{o_2} O_y^{\rho_o} \right)^{\frac{1}{\rho_o}}$$

8.2.1.2.2 Marginal Product of Labor for De-nested Problem We can also take the derivative of the output requirement for the X problem with respect to x_1, we have:

$$\frac{\partial O_x}{\partial x_1} = \left[\frac{1}{\rho_x} \left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x} \right)^{\frac{1}{\rho_x}-1} \right] \cdot \left[\rho_x \beta_{x_1} x_1^{\rho_x - 1} \right]$$

Which simplifies a little bit to:

$$\frac{\partial O_x}{\partial x_1} = \left[\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x} \right)^{\frac{1}{\rho_x}-1} \right] \cdot \left[\beta_{x_1} x_1^{\rho_x - 1} \right]$$

What is the relationship between the marginal product of labor and the wage for the problem in the subnest? Let λ_x be the lagrange multiplier for the lagrange multiplier specific to the subnest:

$$p_{x_1} = \lambda_x \cdot \left(\frac{\partial O_x}{\partial x_1} \right)$$

This means that we have the following FOC from solving the expenditure minimization problem:

$$p_{x_1} = \lambda_x \cdot \left[\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x} \right)^{\frac{1}{\rho_x}-1} \right] \cdot \left[\beta_{x_1} x_1^{\rho_x - 1} \right]$$

8.2.1.3 Solving the Nested-CES Problem

Conceptually, we can solve the nested-ces problem in two stages. First, given aggregate prices, solve for optimal aggregate inputs. Second, given the aggregate inputs, which are output requirements for inner nests, solve for optimal choices within nests.

There are two functions of interest, one function provides A

8.2.1.4 Identification

8.2.1.4.1 Relative Marginal Product Note that $\frac{1}{\rho} - 1 = \frac{1}{\rho} - \frac{\rho}{\rho} = \frac{1-\rho}{\rho}$, and $x^{\frac{1-\rho}{\rho}} = \left(x^{\frac{1}{\rho}} \right)^{1-\rho}$.

Relative marginal product within the same sub-nest:

$$\frac{p_{x_1}}{p_{x_2}} = \frac{\frac{\partial Y}{\partial x_1}}{\frac{\partial Y}{\partial x_2}} = \frac{\rho_x \beta_{x_1} x_1^{\rho_x - 1}}{\rho_x \beta_{x_2} x_2^{\rho_x - 1}} = \frac{\beta_{x_1}}{\beta_{x_2}} \cdot \left(\frac{x_1}{x_2} \right)^{\rho_x - 1}$$

Relative marginal product across subnests:

$$
\begin{aligned}
\frac{\frac{\partial Y}{\partial x_1}}{\frac{\partial Y}{\partial y_1}} &= \frac{p_{x_1}}{p_{y_1}} \\
\frac{\frac{\partial Y}{\partial x_1}}{\frac{\partial Y}{\partial y_1}} &= \frac{\left[\rho_o \beta_{o_1} \left(\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}\right)^{\rho_o - 1}\right] \cdot \left[\frac{1}{\rho_x}\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}-1}\right] \cdot \left[\rho_x \beta_{x_1} x_1^{\rho_x - 1}\right]}{\left[\rho_o \beta_{o_2} \left(\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{1}{\rho_y}}\right)^{\rho_o - 1}\right] \cdot \left[\frac{1}{\rho_y}\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{1}{\rho_y}-1}\right] \cdot \left[\rho_y \beta_{y_1} y_1^{\rho_y - 1}\right]} \\
&= \frac{\left[\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{\rho_o - \rho_x}{\rho_x}}\right]}{\left[\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{\rho_o - \rho_y}{\rho_y}}\right]} \cdot \left[\frac{\beta_{o_1}\beta_{x_1}}{\beta_{o_2}\beta_{y_1}}\right] \cdot \frac{\left[x_1^{\rho_x - 1}\right]}{\left[y_1^{\rho_y - 1}\right]}
\end{aligned}
$$

Note that in the equation above, the first term is the same across for the relative MPL across all within subnest terms.

There are four derivative ratios. First:

$$
\frac{p_{x_1}}{p_{y_1}} = \frac{\left[\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{\rho_o - \rho_x}{\rho_x}}\right]}{\left[\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{\rho_o - \rho_y}{\rho_y}}\right]} \cdot \left[\frac{\beta_{o_1}\beta_{x_1}}{\beta_{o_2}\beta_{y_1}}\right] \cdot \frac{\left[x_1^{\rho_x - 1}\right]}{\left[y_1^{\rho_y - 1}\right]}
$$

Second:

$$
\frac{p_{x_1}}{p_{y_2}} = \frac{\left[\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{\rho_o - \rho_x}{\rho_x}}\right]}{\left[\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{\rho_o - \rho_y}{\rho_y}}\right]} \cdot \left[\frac{\beta_{o_1}\beta_{x_1}}{\beta_{o_2}\beta_{y_2}}\right] \cdot \frac{\left[x_1^{\rho_x - 1}\right]}{\left[y_2^{\rho_y - 1}\right]}
$$

Third:

$$
\frac{p_{x_2}}{p_{y_1}} = \frac{\left[\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{\rho_o - \rho_x}{\rho_x}}\right]}{\left[\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{\rho_o - \rho_y}{\rho_y}}\right]} \cdot \left[\frac{\beta_{o_1}\beta_{x_2}}{\beta_{o_2}\beta_{y_1}}\right] \cdot \frac{\left[x_2^{\rho_x - 1}\right]}{\left[y_1^{\rho_y - 1}\right]}
$$

Fourth:

$$
\frac{p_{x_2}}{p_{y_2}} = \frac{\left[\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{\rho_o - \rho_x}{\rho_x}}\right]}{\left[\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{\rho_o - \rho_y}{\rho_y}}\right]} \cdot \left[\frac{\beta_{o_1}\beta_{x_2}}{\beta_{o_2}\beta_{y_2}}\right] \cdot \frac{\left[x_2^{\rho_x - 1}\right]}{\left[y_2^{\rho_y - 1}\right]}
$$

Note that we have overall seven unknowns, three share parameters, and three elasticity parameters, and a output and productivity ratio. It looks like we have six equations, but only perhaps 3? Three of the above can not be used.

8.2.1.4.2 Identification with Aggregate Data over two Periods We say: (1) given the level of nested structure you have, what is the number of restrictions you have to impose on share or elasticity in order to fully identify the model.

The identification of the three share and elasticity parameters can be achieved by using the following two equations over three periods.

$$
\begin{aligned}
\log\left(\frac{p_{x_1}}{p_{x_2}}\right) &= \log\left(\frac{\beta_{x_1}}{\beta_{x_2}}\right) + (\rho_x - 1) \cdot \log\left(\frac{x_1}{x_2}\right) \\
\log\left(\frac{p_{y_1}}{p_{y_2}}\right) &= \log\left(\frac{\beta_{y_1}}{\beta_{y_2}}\right) + (\rho_y - 1) \cdot \log\left(\frac{y_1}{y_2}\right) \\
\log\left(\frac{p_{x_1}}{p_{y_1}}\right) &= \log\left(\frac{\beta_{o_1}}{\beta_{o_2}}\left[\frac{\beta_{x_1} \cdot x_1^{\rho_x - 1} \cdot O_y^{\rho_y}}{\beta_{y_1} \cdot y_1^{\rho_y - 1} \cdot O_x^{\rho_x}}\right]\right) + \rho_o \cdot \log\left[\frac{O_x}{O_y}\right]
\end{aligned}
$$

Note that the contents in the square brackets are data given the results from the other equations.

The identification of the inner-nest elasticity and share parameters is based on inner-nest quantity and relative price relationships. The relative price across nests, and the relative aggregate quantity across nets, then pin down the elasticity and share parameters across nests. In another word, within nest information on relative prices and quantity contain no information on higher nest parameters, but higher nest parameters are a function of lower nest parameters.

Note that for the higher nest, the intercept term is fully flexibly determined by outter nest share parameters, however, the specific translation between outter nest intercept and share values is scaled by inner-nest estimates and aggregate outputs.

Where

$$O_x = \left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}$$
$$O_y = \left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{1}{\rho_y}}$$

8.2.2 Latent Dynamic Health Production Function

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

8.2.2.1 Latent Health and Observed Health

First, output and input relationship is described generally by:

$$Y_t^* = H(X_t, Z_t, Y_t^*) + \epsilon_t \,,$$

where we assume the separability between the error term and equation f.

Additionally, assume that:

$$\epsilon \sim \mathrm{N}\left(\mu, \sigma\right) \,.$$

We do not observe Y_t^*, however, we observe discretized integers $Y_{i,t}$:

$$Y_{i,t} \equiv \left\{ j \in \mathcal{N} : G_{j-1} \le Y_{i,t}^* \le G_j, \text{ for } 0 < j \le J+1) \right\} \,.$$

Our parameters of interest are: - The $H(X_t, Z_t, Y^{*,t})$ function. - What are $\left\{G_j^*\right\}_{j=1}^J$, assume that $G_0^* = -\infty$ and $G_{J+1}^* = \infty$.

If the function is known, we understand the dyanmic evolution of health, and can analyze the effects of changing input on health status for the given population or for alternative populations with different distributions of current inputs.

8.2.2.2 Standard Estimation with Observed Health Status

Suppose we observe the dataset: $\{Y_i, X_i, Z_i\}_{i=1}^N$. Suppose that $E[\epsilon|X, Z] = 0$.

$$P(Y = j|X, Z) = \Phi\left(\frac{G_j - H(X, Z) - \mu}{\sigma}\right) - \Phi\left(\frac{G_{j-1} - H(X, Z) - \mu}{\sigma}\right)$$

Note that we can construct the log-likelihood, to estimate thresholds and the parameters of for example lienarized $f()$ function. Note that we can not identify μ and also can not identify σ. Note that this is ordered-logit/ordered-probit framework. In this framework, would assume homogeneous parameters across individuals.

8.2.2.3 Observed Probabilities

We now observe the probabilities of observing the discrete Y_i values given X and Z. From the data, we have

$$\{P(Y=j|X,Z)\}_{j=1}^{J} .$$

Additionally, we observe probabilities when input changes. These allow us to estimate/identify the model with individual specific parameters and thresholds, and given the nature of the question, do not have to worry about correlation between error and input changes.

$$Y_t^* = H_i(X_t, Z_t, Y_{t-1}^*) + \epsilon .$$

We have the individual-specific H_i function. Our thresholds now can also be individual-specific $\left\{G_{ji}\right\}_{j=1}^{J}$.

8.2.2.4 Single Contemporaneous Input Model

Suppose there is no dynamics and we only observe one input. For each individual, we know:

$$\{P(Y=j|Z)\}_{j=1}^{J} .$$

Assuming separability between the error term and the observed component of the model, we have:

$$Y^* = H_i(Z) + \epsilon .$$

8.2.2.4.1 Probability given Current Inputs For each individual, it might seem like that we can normalize given their current X choices so that this is equal to 0, but we can not. So the probability of observering aparticular discrete health outcome is equal to:

$$P(Y=j|Z) = \Phi(G_{ji} - H_i(Z)) - \Phi(G_{j-1,i} - H_i(Z))$$

Alternatively, this can be written as:

$$\sum_{\hat{j}=1}^{j} \left(P(Y=\hat{j}|Z) \right) = \Phi(G_{ji} - H_i(Z))$$

For any j, just using information given current choices, we can not identify $\left\{G_{ji}\right\}_{j=1}^{J}$, because we do not know what $H_i(Z)$ is.

8.2.2.4.2 Probability at Two Input Levels Now, we observe both:

$$\{P(Y=j|Z)\}_{j=1}^{J} , \text{ and } \{P(Y=j|Z+1)\}_{j=1}^{J}$$

Since the individual thresholds, $\left\{G_{ji}\right\}_{j=1}^{J}$, applies to both probabilities, now we are able to identify the change in the H_i from Z to $Z+1$. Specifically, because:

$$\Phi^{-1}\left(\sum_{\hat{j}=1}^{j} \left(P(Y=\hat{j}|Z) \right) \right) = G_{ji} - H_i(Z)$$

Hence:

$$\Phi^{-1}\left(\sum_{\hat{j}=1}^{j} \left(P(Y=\hat{j}|Z) \right) \right) - \Phi^{-1}\left(\sum_{\hat{j}=1}^{j} \left(P(Y=\hat{j}|Z+1) \right) \right) = (G_{ji} - H_i(Z)) - (G_{ji} - H_i(Z+1)) = H_i(Z+1) - H_i(Z), \forall j \in$$

Note that the above will equal to the same number for any j under the assumption that the error term is distributed normal.

Define the difference as:

$$\beta_i^+ = H_i(Z+1) - H_i(Z)$$

If H_i is linear in Z, then, $H_i(Z) = \beta_i^+ \cdot Z$. If rather than adding 1 to Z we evaluate probabilities after subtracting 1 from Z and compute following the above procedure β_i^-, we would have $\beta_i^+ = \beta_i^- = \beta_i$.

Having identified the β_i parameter, which is individual-specific. We can now identify all $\left\{G_{ji}\right\}_{j=1}^{J}$:

$$G_{ji} = \Phi^{-1}\left(\sum_{\hat{j}=1}^{j}\left(P(Y=\hat{j}|Z)\right)\right) + Z \cdot \beta_i$$

We observe Z and the probability, and we have just found β_i, so the thresholds are now known.

To conclude, we have Result 1 below:

> Given a latent health production function with a single observed input, a normal additive error term, and given linearity of input and latent health outcome relationship, to T identify individual specific threshold and the effect of input on the latent outcome, we need to observe, for each individual, $\{P(Y=j|Z)\}_{j=1}^{J}$ and $P(Y=1|Z+1)$.

As a corrolary of Result 1, Result 2 is:

> More flexibly, given deviations in outcomes of β_i^+, 0, and β_i^-, at $Z-1$, Z, and $Z+1$, we can fit a quadratic function such that $H_i(Z) = \beta_0 + \beta_1 \cdot Z + \beta_2 \cdot Z^2$.

8.2.2.5 Mixture of Normals

For the exercse above, for $Z+1$, we only needed $P(Y=1|Z+1)$. Suppose we observe both $\{P(Y=j|Z), P(Y=j|Z+1)\}_{j=1}^{J}$ for both Z and $Z+1$, then we could allow for a mixture of normal for the error term. Another way to think about this is if we obtain very different $\widehat{\beta}_i$ depending on which $P(Y=j|Z+1)$ is observed for which j, we can potentially explain the observed data with more flexible distributional assumptions. A mixture of normal can be assumed, rather than a normal distribution. Adding a mixture introduces three additional parameter, the mean of the second normal, its standard deviation, as well as its weight.

Given these, suppose we have $J \times 2$ probabilities, and $J=4$ ($J=5$ is one minus the rest), this means we have 8 probabilities available to use, and we have 4 threshold parameters, one β_i parameter, and the 3 parameters associated with the 2nd normal of the mixture.

Under the mixture of two normals, we have J equations:

$$\sum_{\hat{j}=1}^{j}\left(P(Y=\hat{j}|Z)\right) = (1-\omega_i)\cdot\Phi\left(G_{ji}\right) + \omega_i \cdot \Phi\left(\frac{G_{ji}-\mu_{\epsilon_2}}{\sigma_{\epsilon_2}}\right), \forall j \in \{1,...,J\}$$

And J additional equations:

$$\sum_{\hat{j}=1}^{j}\left(P(Y=\hat{j}|Z+1)\right) = (1-\omega_i)\cdot\Phi\left(G_{ji}-\beta_i\right) + \omega_i \cdot \Phi\left(\frac{G_{ji}-\beta_i-\mu_{\epsilon_2}}{\sigma_{\epsilon_2}}\right), \forall j \in \{1,...,J\}$$

When $J=4$, we solve the above system of 8 equations with the following 8 unknowns:

$$\left\{G_{1i}, G_{2i}, G_{3i}, G_{4i}, \beta_i, \omega_i, \mu_{\epsilon_2}, \sigma_{\epsilon_2}\right\}$$

This is Result 3.

Additionally normals can be added into the mixture if there are additional moments to fit when other inputs change for example.

8.2.2.6 Multiple Inputs

Suppose we have two inputs, and still assuming separability between the error term and the observed component of the model, we have:

$$Y^* = H_i(Z, X) + \epsilon\,.$$

Suppose we have probabilities

$$\{P(Y = j|Z, X), P(Y = j|Z + 1, X), P(Y = j|Z, X + 1), P(Y = j|Z + 1, X + 1)\}_{j=1}^{J}\ .$$

We can follow the prior procedures to identify

$$\beta_i = H_i(Z + 1, X) - H_i(Z, X)\alpha_i = H_i(Z, X + 1) - H_i(Z, X)\zeta_i = H_i(Z + 1, X + 1) - H_i(Z, X)$$

Given different parametric assumptions on H_i, we can back out different underlying production function parameters. To illustrate, suppose we have:

$$H_i = \overline{\alpha_i} \cdot X + \overline{\beta_i} \cdot Z + \overline{\zeta_i} \cdot X \cdot Z$$

Then, we can back out the underlying production function parameters with the following three equations and three unknowns:

$$\beta_i = \overline{\beta_i} + \bar{\zeta}_i \cdot X\alpha_i = \overline{\alpha_i} + \bar{\zeta}_i \cdot Z\zeta_i = \overline{\alpha_i} + \overline{\beta_i} + \overline{\zeta_i} \cdot (X + Z + 1)$$

This is Result 4.

8.2.2.7 Dynamics and Contemporaneous Input

Now we generalize the structure above, to account for dynamics. Our model still has only one input, but we also know the lagged input. For each individual, we know:

$$\{P(Y_t = j|Z_t, Y_{t-1})\}_{j=1}^{J}\ .$$

Suppose that we have the following dynamic relationship in the latent variable and linear input/output relationship:

$$Y_t^* = \rho \cdot Y_{t-1}^* + \beta_i \cdot Z_t + \epsilon\,.$$

Note that the ρ is not individual-specific, but β_i is.

8.2.2.7.1 Identifying β_i Despite the inclusion of dynamics, we identify β_i in the same way as for the problem without dynamics. This is possible because the $\rho \cdot Y_{t-1}^*$ is invariant across probabilities of arriving in different health status tomorrow whether input is Z or $Z + 1$. Specifically, we have, for each individual:

$$\Phi^{-1}\left(\sum_{\hat{j}=1}^{j}\Big(P(Y_t = \hat{j}|Z_t, Y_{t-1}^*)\Big)\right) - \Phi^{-1}\left(\sum_{\hat{j}=1}^{j}\Big(P(Y_t = \hat{j}|Z + 1, Y_{t-1}^*)\Big)\right) = (G_{ji} - \rho \cdot Y_{t-1}^* - \beta_i \cdot Z_t) - (G_{ji} - \rho \cdot Y_{t-1}^* - \beta_i \cdot (Z_t + 1)$$

Below, we try to identify persistence ρ and thresholds $\left\{G_{ji}\right\}_{j=1}^{J}$

8.2.2.7.2 Identify Gaps In Thresholds The presence of potential lagged persistence, however, means that we can no longer directly obtain the individual-specific threshold coefficients, $\left\{G_{ji}\right\}_{j=1}^{J}$, as prior.

For each individual, conditional on the same lagged outcome, we have probabilities of arriving at different j health status next period:

$$G_{ji} + \rho \cdot Y^{*}_{t-1} = \Gamma_{ji} = \Phi^{-1}\left(\sum_{\hat{j}=1}^{j}\left(P(Y_t = \hat{j}|Z_t, Y^{*}_{t-1})\right)\right) + \beta_i \cdot Z_t$$

Differencing, we have:

$$\forall (j' - 1) = j \geq 1,\ \Gamma_{j',i} - \Gamma_{j,i} = (G_{j',i} + \rho \cdot Y^{*}_{t-1}) - (G_{ji} + \rho \cdot Y^{*}_{t-1}) = G_{j',i} - G_{ji} = \Delta G_{j',i}$$

So we know the individual specific threshold gaps, but we do not know $G_{j=1,i}$. We know, Γ_{ji}, from the right-hand side in the equation above, but can not distinguish the left-hand side components. Importantly, we do not observe the latent lagged variable $Y^{*}_{t-1} \in \mathcal{R}$, but only observed the discretized value $Y_{t-1} \in \{1, .., J\}$.

8.2.2.7.3 Linear Restrictions on Threshold and Lagged Latent Values given ρ From the last section, we have, for $j = 1$:

$$\Gamma_{j=1,i} = G_{j=1,i} + \rho \cdot Y^{*}_{t-1}$$

Note that $j = 1$ is not for the lagged choice, but relates to the probability of going to j in t.

Note that the $G_{j=1,i}$ is a bound on feasible values for Y^{*}_{t-1} if we observe $Y_{t-1} = 1$. Suppose we observed $Y_{t-1} = 1$, we have two equations:

$$G_{j=1,i} = \Gamma_{j=1,i} - \rho \cdot Y^{*}_{t-1} G_{j=1,i} > 0 + 1 \times Y^{*}_{t-1}$$

Think of $G_{j=1,i}$ as the Y-axis value and Y^{*}_{t-1} as the X-axis value. The first equation is a downward sloping line whose slope is controled by $\rho \in [0, 1)$, and whose Y-intercept we already know. The second equation says that we can restrict our attention to the area to the top left of the 45 degree upward sloping line through the origin. With these two equations, while we do not know the exactly values of threshold $G_{j=1,i}$ and the latent lagged value $Y^{*}_{j=1,i}$, we have substantially restricted the sub-set of jointly valid values that is consistent with observed probabilities given ρ.

Given ρ, and observing $Y_{t-1} = 1$, note that the two equations above provides a minimum threshold and a maximum latent value.

$$G^{\min}_{j=1,i}(Y_{t-1} = 1) = \frac{\Gamma_{j=1,i}}{1-\rho} Y^{*,\max}_{t-1}(Y_{t-1} = 1) = \frac{\Gamma_{j=1,i}}{1+\rho}$$

If we observe for the lagged discrete value $Y_{t-1} = j > 1$, then rather than having 2 equations as above, we have 3:

$$G_{j,i} = \Gamma_{j,i} - \rho \cdot Y^{*}_{t-1} G_{j,i} > 0 + 1 \times Y^{*}_{t-1} G_{j,i} < (\Gamma_{j,i} - \Gamma_{j-1,i}) + 1 \times Y^{*}_{t-1}$$

Which gives us bounds on the lagged latent value:

$$Y^{*,\max}_{t-1}(Y_{t-1} = j) = \frac{\Gamma_{j,i}}{1+\rho} Y^{*,\min}_{t-1}(Y_{t-1} = j) = \frac{\Gamma_{j-1,i}}{1+\rho}$$

Note that we are explicitly using the information provided by the lagged discrete health status, because that's what we are relying on to generate the inequality restrictions.

8.2.2.7.4 The Latent Distribution

ρ and $\beta_i \cdot Z$ jointly determine the distribution of $f(Y^*)$. For simplificty, suppose that an individual who uses Z level of input at t uses the same level of input in all past periods, then we know that (given that ϵ is normalized as a standard normal):

$$Y^* \sim \mathcal{N}\left(\mu_{Y^*} = \left(\frac{\beta_i \cdot Z}{1-\rho}\right), \sigma^2_{Y^*} = \left(\frac{1}{1-\rho^2}\right)\right)$$

Regardless of where threshold cutoff values are, we can compute the chance of observing latent lagged value between some range $P(a < Y^*_{t-1} < b)$.

8.2.2.7.5 Estimating Persistence ρ

The intuition for looking for the persistence parameter is that even though given ρ, there is a continuum of threshold and lagged latent values that can explain observed probabilities. However, values of ρ controls, as just shown, the distribution of the latent variable. So as ρ changes, the chance for observing different ranges of latent values changes.

So given that we observe $Y_{t-1} = j$, we look for ρ to maximize the probability of observing $Y_{t-1} = j$ (without needing to know what the threshold values should be):

$$\hat{\rho}^* = \arg\max_{\hat{\rho}} \left(\int^{\frac{\Gamma_{j,i}}{1+\hat{\rho}}}_{\frac{\Gamma_{j-1,i}}{1+\hat{\rho}}} \phi\left(\frac{Y^* - \left(\frac{\beta_i \cdot Z}{1-\hat{\rho}}\right)}{\left(\frac{1}{1-\hat{\rho}^2}\right)} \right) dY^* \right)$$

Note that:

1. $\Gamma_{j,i}$ and $\Gamma_{j-1,i}$ are based on observed probabilities
2. the integration bounds comes from the lagged discrete outcome which generates the inequality conditions
3. Discrete thresholds are unrelated to the distribution of the latent variable.

Maximizer $\hat{\rho}$ is an individual-specific maximizer. Not entirely clear if $\hat{\rho}^* = \rho$. Perhaps in expectation $E[\hat{\rho}] = \rho$, when we average ovr the estimate from multiple individuals.

8.3 Inequality Models

8.3.1 GINI Discrete Sample

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

This works out how the ff_dist_gini_vector_pos function works from Fan's *REconTools* Package.

8.3.1.1 GINI Formula for Discrete Sample

There is an vector values (all positive). This could be height information for N individuals. It could also be income information for N individuals. Calculate the GINI coefficient treating the given vector as population. This is not an estimation exercise where we want to estimate population GINI based on a sample. The given array is the population. The population is discrete, and only has these N individuals in the length n vector.

Note that when the sample size is small, there is a limit to inequality using the formula defined below given each N. So for small N, can not really compare inequality across arrays with different N, can only compare arrays with the same N.

The GINI formula used here is:

$$GINI = 1 - \frac{2}{N+1} \cdot \left(\sum_{i=1}^{N} \sum_{j=1}^{i} x_j\right) \cdot \left(\sum_{i=1}^{N} x_i\right)^{-1}$$

Derive the formula in the steps below.

Step 1 Area Formula

$$\Gamma = \sum_{i=1}^{N} \frac{1}{N} \cdot \left(\sum_{j=1}^{i} \left(\frac{x_j}{\sum_{\hat{j}=1}^{N} x_{\hat{j}}} \right) \right)$$

Step 2 Total Area Given Perfect equality

With perfect equality $x_i = a$ for all i, so need to divide by that.

$$\Gamma^{\text{equal}} = \sum_{i=1}^{N} \frac{1}{N} \cdot \left(\sum_{j=1}^{i} \left(\frac{a}{\sum_{\hat{j}=1}^{N} a} \right) \right) = \frac{N+1}{N} \cdot \frac{1}{2}$$

As the number of elements of the vecotr increases:

$$\lim_{N \to \infty} \Gamma^{\text{equal}} = \lim_{N \to \infty} \frac{N+1}{N} \cdot \frac{1}{2} = \frac{1}{2}$$

Step 3 Arriving at Finite Vector GINI Formula

Given what we have from above, we obtain the GINI formula, divide by total area below 45 degree line.

$$GINI = 1 - \left(\sum_{i=1}^{N} \sum_{j=1}^{i} x_j \right) \cdot \left(N \cdot \sum_{i=1}^{N} x_i \right)^{-1} \cdot \left(\frac{N+1}{N} \cdot \frac{1}{2} \right)^{-1} = 1 - \frac{2}{N+1} \cdot \left(\sum_{i=1}^{N} \sum_{j=1}^{i} x_j \right) \cdot \left(\sum_{i=1}^{N} x_i \right)^{-1}$$

Step 4 Maximum Inequality given N

Suppose $x_i = 0$ for all $i < N$, then:

$$GINI^{x_i=0 \text{ except } i=N} = 1 - \frac{2}{N+1} \cdot X_N \cdot (X_N)^{-1} = 1 - \frac{2}{N+1}$$

$$\lim_{N \to \infty} GINI^{x_i=0 \text{ except } i=N} = 1 - \lim_{N \to \infty} \frac{2}{N+1} = 1$$

Note that for small N, for example if $N = 10$, even when one person holds all income, all others have 0 income, the formula will not produce GINI is zero, but that GINI is equal to $\frac{2}{11} \approx 0.1818$. If $N = 2$, inequality is at most, $\frac{2}{3} \approx 0.667$.

$$MostUnequalGINI(N) = 1 - \frac{2}{N+1} = \frac{N-1}{N+1}$$

8.3.1.1.1 Implement GINI Formula for Discrete Sample The **GINI** formula just derived is trivial to compute.

1. scalar: $\frac{2}{N+1}$
2. cumsum: $\sum_{j=1}^{i} x_j$
3. sum of cumsum: $\left(\sum_{i=1}^{N} \sum_{j=1}^{i} x_j \right)$
4. sum: $\sum_{i=1}^{N} X_i$

There are no package dependencies. Define the formula here:

```
# Formula, directly implement the GINI formula Following Step 4 above
ffi_dist_gini_vector_pos_test <- function(ar_pos) {
  # Check length and given warning
  it_n <- length(ar_pos)
  if (it_n <= 100)  warning('Data vector has n=',it_n,', max-inequality/max-gini=',(it_n-1)/(it_n +
  # Sort
```

```
  ar_pos <- sort(ar_pos)
  # formula implement
  fl_gini <- 1 - ((2/(it_n+1)) * sum(cumsum(ar_pos))*(sum(ar_pos))^(-1))
  return(fl_gini)
}
```

Generate a number of examples Arrays for testing

```
# Example Arrays of data
ar_equal_n1 = c(1)
ar_ineql_n1 = c(100)

ar_equal_n2 = c(1,1)
ar_ineql_alittle_n2 = c(1,2)
ar_ineql_somewht_n2 = c(1,2^3)
ar_ineql_alotine_n2 = c(1,2^5)
ar_ineql_veryvry_n2 = c(1,2^8)
ar_ineql_mostmst_n2 = c(1,2^13)

ar_equal_n10 = c(2,2,2,2,2,2, 2, 2, 2, 2)
ar_ineql_some_n10 = c(1,2,3,5,8,13,21,34,55,89)
ar_ineql_very_n10 = c(1,2^2,3^2,5^2,8^2,13^2,21^2,34^2,55^2,89^2)
ar_ineql_extr_n10 = c(1,2^2,3^3,5^4,8^5,13^6,21^7,34^8,55^9,89^10)
```

Now test the example arrays above using the function based no our formula:

```
##
## Small N=1 Hard-Code

## ar_equal_n1: 0

## ar_ineql_n1: 0

##
## Small N=2 Hard-Code, converge to 1/3, see formula above

## ar_ineql_alittle_n2: 0.1111111

## ar_ineql_somewht_n2: 0.2592593

## ar_ineql_alotine_n2: 0.3131313

## ar_ineql_veryvry_n2: 0.3307393

##
## Small N=10 Hard-Code, convege to 9/11=0.8181, see formula above

## ar_equal_n10: 0

## ar_ineql_some_n10: 0.5395514

## ar_ineql_very_n10: 0.7059554

## ar_ineql_extr_n10: 0.8181549
```

8.3.2 GINI Formula for Discrete Random Varialbe

For a discrete random variable, we are two arrays, an array of x values, and an array of $f(x)$ probability mass at each x value. Suppose the x values are unique/non-repeating. This is also Implemented in MEconTools with the ff_disc_rand_var_gini function.

Generate two arrays for x and $f(x)$, we will use the binomial distribution:

```
ar_choice_unique_sorted <- seq(0, 100, by=1)
ar_choice_prob <- dbinom(ar_choice_unique_sorted, 100, 0.01)
```

Generate mean and cumulative mean at each point:

```
# 1. to normalize, get mean (binomial so mean is p*N=50)
fl_mean <- sum(ar_choice_unique_sorted*ar_choice_prob);
# 2. get cumulative mean at each point
ar_mean_cumsum <- cumsum(ar_choice_unique_sorted*ar_choice_prob);
```

Normalizing and area calculation, following the same principle as above:

```
# 3. Share of wealth (income etc) accounted for up to this sorted type
ar_height <- ar_mean_cumsum/fl_mean;
# 4. The total area, is the each height times each width summed up
fl_area_drm <- sum(ar_choice_prob*ar_height);
```

Finally GINI coefficient:

```
# 5. area below 45 degree line might not be 1/2, depends on discretness
fl_area_below45 <- sum(ar_choice_prob*(cumsum(ar_choice_prob)/sum(ar_choice_prob)))

# 6. Gini is the distance between
fl_gini_index <- (fl_area_below45-fl_area_drm)/fl_area_below45
print(paste0('fl_gini_index=', fl_gini_index))
```

```
## [1] "fl_gini_index=0.468573066002754"
```

8.3.2.1 Discrete Random Variable as Function

Organizing the code above as a function, and testing results out with the binomial distribution as an example.

For the binomial distribution, if the probability of success is very close to zero, that means nearly all mass is at lose all or nearly losing all. There will be non-zero but very small mass at higher levels of wins. Hence this should mean extreme inequality. GINI index should be close to 1. Alternatively, GINI index should be close to 0 when we have near 100 percent chance of success, then all mass is at winning all, perfect equality.

```
# Combining the code from above
ffi_dist_gini_random_var_pos_test <- function(ar_x_sorted, ar_prob_of_x) {
  #' @param ar_x_sorted sorted array of values
  #' @param ar_prob_of_x probability mass for each element along `ar_x_sorted`, sums to 1

  # 1. to normalize, get mean (binomial so mean is p*N=50)
  fl_mean <- sum(ar_x_sorted*ar_prob_of_x);
  # 2. get cumulative mean at each point
  ar_mean_cumsum <- cumsum(ar_x_sorted*ar_prob_of_x);
  # 3. Share of wealth (income etc) accounted for up to this sorted type
  ar_height <- ar_mean_cumsum/fl_mean;
  # 4. The total area, is the each height times each width summed up
  fl_area_drm <- sum(ar_prob_of_x*ar_height);
  # 5. area below 45 degree line might not be 1/2, depends on discretness
  fl_area_below45 <- sum(ar_prob_of_x*(cumsum(ar_prob_of_x)/sum(ar_prob_of_x)))
  # 6. Gini is the distance between
  fl_gini_index <- (fl_area_below45-fl_area_drm)/fl_area_below45

  return(fl_gini_index)
}
```

Testing the function with the Binomial Distribution:

```
for (fl_binom_success_prob in seq(0.0001,0.9999,length.out=10)) {
  ar_x_sorted <- seq(0, 100, by=1)
  ar_prob_of_x <- dbinom(ar_x_sorted, 100, fl_binom_success_prob)
  fl_gini_index <- ffi_dist_gini_random_var_pos_test(ar_x_sorted, ar_prob_of_x)
```

```
  st_print <- paste0('binom p(success)=', fl_binom_success_prob ,
                     ', the fl_gini_index=', fl_gini_index)
  print(st_print)
}
```

```
## [1] "binom p(success)=1e-04, the fl_gini_index=0.990048846889393"
## [1] "binom p(success)=0.111188888888889, the fl_gini_index=0.147061101509638"
## [1] "binom p(success)=0.222277777777778, the fl_gini_index=0.0989942681255604"
## [1] "binom p(success)=0.333366666666667, the fl_gini_index=0.0753059593394789"
## [1] "binom p(success)=0.444455555555556, the fl_gini_index=0.0596495299535176"
## [1] "binom p(success)=0.555544444444444, the fl_gini_index=0.0476678493269222"
## [1] "binom p(success)=0.666633333333333, the fl_gini_index=0.0375168214334586"
## [1] "binom p(success)=0.777722222222222, the fl_gini_index=0.0280646430085938"
## [1] "binom p(success)=0.888811111111111, the fl_gini_index=0.0180312757603542"
## [1] "binom p(success)=0.9999, the fl_gini_index=9.90197372312816e-07"
```

8.3.2.2 Compare Discrete Sample and Discrete Random Variable Functions for GINI

ff_dist_gini_random_var provides the GINI implementation for a discrete random variable. The procedure is the same as prior, except now each element of the "x" array has element specific weights associated with it. The function can handle unsorted array with non-unique values.

Test and compare ff_dist_gini_random_var provides the GINI implementation for a discrete random variable and ff_dist_gini_vector_pos.

There is a vector of values from 1 to 100, in ascending order. What is the equal-weighted gini, the gini result when smaller numbers have higher weights, and when larger numbers have higher weights?

First, generate the relevant values.

```
# array
ar_x <- seq(1, 100, length.out = 30)
# prob array
ar_prob_x_unif <- rep.int(1, length(ar_x))/sum(rep.int(1, length(ar_x)))
# prob higher at lower values
ar_prob_x_lowval_highwgt <- rev(cumsum(ar_prob_x_unif))/sum(cumsum(ar_prob_x_unif))
# prob higher at lower values
ar_prob_x_highval_highwgt <- (cumsum(ar_prob_x_unif))/sum(cumsum(ar_prob_x_unif))
# show
kable(cbind(ar_x, ar_prob_x_unif, ar_prob_x_lowval_highwgt, ar_prob_x_highval_highwgt)) %>%
  kable_styling_fc()
```

Second, generate GINI values. What should happen?

1. The ff_dist_gini_random_var and ff_dist_gini_vector_pos results should be the same when the uniform distribution is used.
2. GINI should be higher, more inequality, if there is higher weights on the lower values.
3. GINI should be lower, more equality, if there is higher weight on the higher values.

```
ff_dist_gini_vector_pos(ar_x)
```

```
## [1] 0.3267327
```

```
ff_dist_gini_random_var(ar_x, ar_prob_x_unif)
```

```
## [1] 0.3267327
```

```
ff_dist_gini_random_var(ar_x, ar_prob_x_lowval_highwgt)
```

```
## [1] 0.4010343
```

```
ff_dist_gini_random_var(ar_x, ar_prob_x_highval_highwgt)
```

```
## [1] 0.1926849
```

ar_x	ar_prob_x_unif	ar_prob_x_lowval_highwgt	ar_prob_x_highval_highwgt
1.000000	0.0333333	0.0645161	0.0021505
4.413793	0.0333333	0.0623656	0.0043011
7.827586	0.0333333	0.0602151	0.0064516
11.241379	0.0333333	0.0580645	0.0086022
14.655172	0.0333333	0.0559140	0.0107527
18.068966	0.0333333	0.0537634	0.0129032
21.482759	0.0333333	0.0516129	0.0150538
24.896552	0.0333333	0.0494624	0.0172043
28.310345	0.0333333	0.0473118	0.0193548
31.724138	0.0333333	0.0451613	0.0215054
35.137931	0.0333333	0.0430108	0.0236559
38.551724	0.0333333	0.0408602	0.0258065
41.965517	0.0333333	0.0387097	0.0279570
45.379310	0.0333333	0.0365591	0.0301075
48.793103	0.0333333	0.0344086	0.0322581
52.206897	0.0333333	0.0322581	0.0344086
55.620690	0.0333333	0.0301075	0.0365591
59.034483	0.0333333	0.0279570	0.0387097
62.448276	0.0333333	0.0258065	0.0408602
65.862069	0.0333333	0.0236559	0.0430108
69.275862	0.0333333	0.0215054	0.0451613
72.689655	0.0333333	0.0193548	0.0473118
76.103448	0.0333333	0.0172043	0.0494624
79.517241	0.0333333	0.0150538	0.0516129
82.931034	0.0333333	0.0129032	0.0537634
86.344828	0.0333333	0.0107527	0.0559140
89.758621	0.0333333	0.0086022	0.0580645
93.172414	0.0333333	0.0064516	0.0602151
96.586207	0.0333333	0.0043011	0.0623656
100.000000	0.0333333	0.0021505	0.0645161

8.3.3 Atkinson Inequality Index

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

8.3.3.1 Atkinson Inequality Measures

Atkinson (JET, 1970) studies five standard inequality measures. Atkinson finds that given the same income data across countries, different inequality measure lead to different rankings of which country is more unequal. Atkinson develops an measure of inequality that changes depending on an inequality aversion parameter.

$$\text{Atkinson Inequality} = A\left(\left\{Y_i\right\}_{i=1}^N, \lambda\right) = 1 - \left(\sum_{i=1}^N \frac{1}{N}\left(\frac{Y_i}{\sum_{j=1}^N\left(\frac{Y_j}{N}\right)}\right)^\lambda\right)^{\frac{1}{\lambda}} \in [0,1]$$

$A\left(\left\{Y_i\right\}_{i=1}^N, \lambda\right)$ equals to zero is perfect equality. 1 is Perfect inequality. If $\lambda = 1$, the inequality measure is always equal to 0 because the planner does not care abouot inequality anymore.

8.3.3.2 Atkinson Inequality Function

Programming up the equation above, we have, given a sample of data:

```
# Formula
ffi_atkinson_ineq <- function(ar_data, fl_rho) {
  ar_data_demean <- ar_data/mean(ar_data)
  it_len <- length(ar_data_demean)
  fl_atkinson <- 1 - sum(ar_data_demean^{fl_rho}*(1/it_len))^(1/fl_rho)
  return(fl_atkinson)
}
```

When each element of the data array has weight, we have:

```
# Formula
ffi_atkinson_random_var_ineq <- function(ar_data, ar_prob_data, fl_rho) {
  #' @param ar_data array sorted array values
  #' @param ar_prob_data array probability mass for each element along `ar_data`, sums to 1
  #' @param fl_rho float inequality aversion parameter fl_rho = 1 for planner
  #' without inequality aversion. fl_rho = -infinity for fully inequality averse.

  fl_mean <- sum(ar_data*ar_prob_data);
  fl_atkinson <- 1 - (sum(ar_prob_data*(ar_data^{fl_rho}))^(1/fl_rho))/fl_mean
  return(fl_atkinson)
}
```

8.3.3.3 Atkinson Inequality Examples

Given a vector of observables, compute the Atkinson inequality measure given different inequality aversion.

8.3.3.3.1 Data Samples and Weighted Data

The ρ preference vector.

```
# Preference Vector
ar_rho <- 1 - (10^(c(seq(-2.0,2.0, length.out=30))))
ar_rho <- unique(ar_rho)
mt_rho <- matrix(ar_rho, nrow=length(ar_rho), ncol=1)
```

Sampled version for N sampled points, random, uniform and one-rich.

```
# Random normal Data Vector (not equal outcomes)
set.seed(123)
it_sample_N <- 30
fl_rnorm_mean <- 100
fl_rnorm_sd <- 6
ar_data_rnorm <- rnorm(it_sample_N, mean=fl_rnorm_mean, sd=fl_rnorm_sd)
# Uniform Data Vector (Equal)
ar_data_unif <- rep(1, length(ar_data_rnorm))

# One Rich (last person has income equal to the sum of all others*100)
ar_data_onerich <- rep(0.1, length(ar_data_rnorm))
ar_data_onerich[length(ar_data_onerich)] = sum(head(ar_data_onerich,-1))*10
```

Given the same distributions, random, uniform and one-rich, generate discrete random variable versions below. We approximate continuous normal with discrete binomial:

```
# Use binomial to approximate normal
fl_p_binom <- 1 - fl_rnorm_sd^2/fl_rnorm_mean
fl_n_binom <- round(fl_rnorm_mean^2/(fl_rnorm_mean - fl_rnorm_sd^2))
fl_binom_mean <- fl_n_binom*fl_p_binom
fl_binom_sd <- sqrt(fl_n_binom*fl_p_binom*(1-fl_p_binom))
# drv = discrete random variable
ar_drv_rbinom_xval <- seq(1, fl_n_binom)
ar_drv_rbinom_prob <- dbinom(ar_drv_rbinom_xval, size=fl_n_binom, prob=fl_p_binom)
# ignore weight at x=0
ar_drv_rbinom_prob <- ar_drv_rbinom_prob/sum(ar_drv_rbinom_prob)
```

Additionally, for the one-rich vector created earlier, now consider several probability mass over them, change the weight assigned to the richest person.

```
# This should be the same as the unweighted version
ar_drv_onerich_prob_unif <- rep(1/it_sample_N, it_sample_N)
# This puts almost no weight on the last rich person
# richlswgt = rich less weight
ar_drv_onerich_prob_richlswgt <- ar_drv_onerich_prob_unif
ar_drv_onerich_prob_richlswgt[it_sample_N] <- (1/it_sample_N)*0.1
ar_drv_onerich_prob_richlswgt <- ar_drv_onerich_prob_richlswgt/sum(ar_drv_onerich_prob_richlswgt)
# This puts more weight on the rich person
# richmrwgt = rich more weight
ar_drv_onerich_prob_richmrwgt <- ar_drv_onerich_prob_unif
ar_drv_onerich_prob_richmrwgt[it_sample_N] <- (1/it_sample_N)*10
ar_drv_onerich_prob_richmrwgt <- ar_drv_onerich_prob_richmrwgt/sum(ar_drv_onerich_prob_richmrwgt)
```

8.3.3.3.2 Testing Atkinson Index at Single rho Value Atkinson index with $\rho = -1$, which is a planner with some aversion towards inequality, this is equivalent to CRRA=2.

Testing with normal sample draw vs normal approximated with binomial discrete random variable:

```
# ATK = 0.05372126
ffi_atkinson_ineq(ar_data_rnorm, -1)
```

```
## [1] 0.00335581
```

```
# ATK = 0.03443246
ffi_atkinson_random_var_ineq(ar_drv_rbinom_xval, ar_drv_rbinom_prob, -1)
```

```
## [1] 0.003642523
```

```
# ATK = 0
ffi_atkinson_ineq(ar_data_unif, -1)
```

```
## [1] 0
```

Testing with

```
# ATK = 0.89, sample
ffi_atkinson_ineq(ar_data_onerich, -1)
```

```
## [1] 0.9027248
```

```
# ATK = 0.89, drv, uniform weight
ffi_atkinson_random_var_ineq(ar_data_onerich, ar_drv_onerich_prob_unif, -1)
```

```
## [1] 0.9027248
```

```
# ATK = 0.49, drv, less weight on rich
ffi_atkinson_random_var_ineq(ar_data_onerich, ar_drv_onerich_prob_richlswgt, -1)
```

```
## [1] 0.4965518
```

```
# ATK = 0.97, drv, more weight on rich
ffi_atkinson_random_var_ineq(ar_data_onerich, ar_drv_onerich_prob_richmrwgt, -1)
```

```
## [1] 0.9821147
```

Create vector of inequality aversion parameters and graph legends.

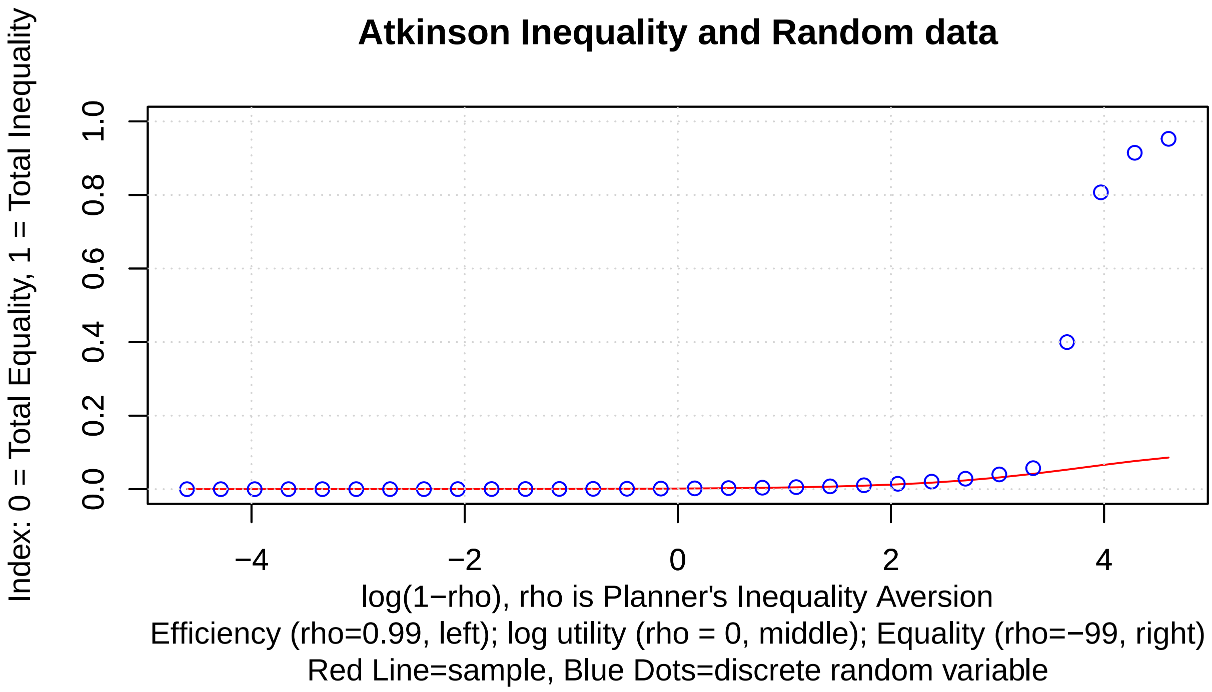
```
ar_log_1_minus_rho <- log(1-ar_rho)
st_x_label <- 'log(1-rho), rho is Planner\'s Inequality Aversion\nEfficiency (rho=0.99, left); log u
st_y_label <- 'Index: 0 = Total Equality, 1 = Total Inequality'
```

8.3.3.3.3 Atkinson Inequality and Normally Distributed Data

How does Atkinson Inequality measure change with respect to a vector of normal random data as inequality aversion shifts? Note that in the example below, the bionomial approximated version is very similar to the normally drawn sample version. until rho beocmes very negative.

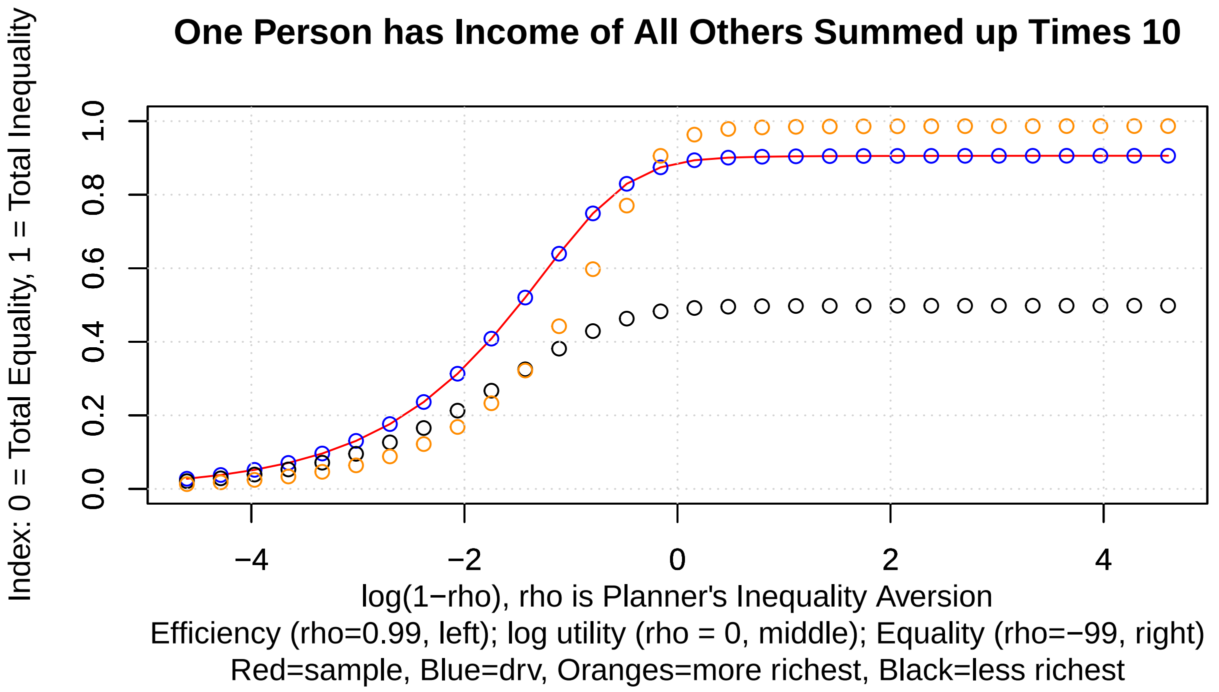
At very negative rho values, the binomial approximation has very tiny, but positive mass for all small values starting from 0, the normally drawn sample has no mass at those points. The Atkinson inequality planner increasingly only cares about the individual with the lowest value from the binomial approximated version, and given the low value of those individuals compared to others, despite having no mass, the inequality index is almost 1.

```
ar_ylim = c(0,1)
# First line
par(new=FALSE)
ar_atkinson_sample <- apply(mt_rho, 1, function(row){
  ffi_atkinson_ineq(ar_data_rnorm, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_sample,
     ylim = ar_ylim, xlab = st_x_label, ylab = st_y_label,
     type="l", col = 'red')
# Second line
par(new=T)
ar_atkinson_drv <- apply(mt_rho, 1, function(row){
  ffi_atkinson_random_var_ineq(ar_drv_rbinom_xval, ar_drv_rbinom_prob, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_drv,
     ylim = ar_ylim, xlab = '', ylab = '',
     type="p", col = 'blue')
# Title
title(main = 'Atkinson Inequality and Random data',
      sub = 'Red Line=sample, Blue Dots=discrete random variable')
grid()
```



8.3.3.3.4 Atkinson Inequality with an Extremely Wealthy Individual Now with the one person has the wealth of all others in the vector times 10.

```
# First line
par(new=FALSE)
ar_atkinson <- apply(mt_rho, 1, function(row){ffi_atkinson_ineq(
  ar_data_onerich, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson,
     ylim = ar_ylim, xlab = st_x_label, ylab = st_y_label,
     type="l", col = 'red')
# Second line
par(new=T)
ar_atkinson_drv <- apply(mt_rho, 1, function(row){ffi_atkinson_random_var_ineq(
  ar_data_onerich, ar_drv_onerich_prob_unif, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_drv,
     ylim = ar_ylim, xlab = '', ylab = '',
     type="p", col = 'blue')
# Third line
par(new=T)
ar_atkinson_drv_richlswgt <- apply(mt_rho, 1, function(row){ffi_atkinson_random_var_ineq(
  ar_data_onerich, ar_drv_onerich_prob_richlswgt, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_drv_richlswgt,
     ylim = ar_ylim, xlab = '', ylab = '',
     type="p", col = 'black')
# Fourth line
par(new=T)
ar_atkinson_drv_richmrwgt <- apply(mt_rho, 1, function(row){ffi_atkinson_random_var_ineq(
  ar_data_onerich, ar_drv_onerich_prob_richmrwgt, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_drv_richmrwgt,
     ylim = ar_ylim, xlab = '', ylab = '',
     type="p", col = 'darkorange')
# Title
title(main = 'One Person has Income of All Others Summed up Times 10',
      sub  = 'Red=sample, Blue=drv, Oranges=more richest, Black=less richest')
grid()
```

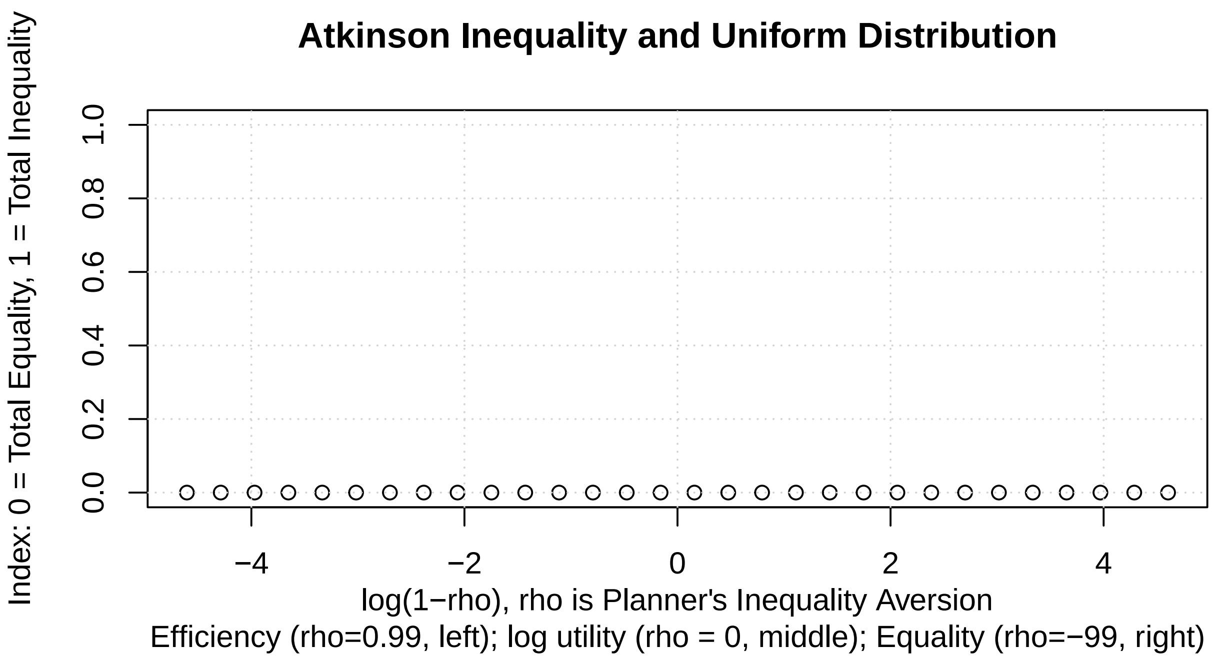


8.3.3.3.5 Atkinson Inequality with an Uniform Distribution The Uniform Results, since allocations are uniform, zero for all.

```
par(new=FALSE)
ffi_atkinson_ineq(ar_data_unif, -1)
```

```
## [1] 0
```

```
ar_atkinson <- apply(mt_rho, 1, function(row){ffi_atkinson_ineq(ar_data_unif, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson, ylim = ar_ylim, xlab = st_x_label, ylab = st_y_label)
title(main = 'Atkinson Inequality and Uniform Distribution')
grid()
```



8.3.3.4 Analyzing Equation Mechanics

How does the Aktinson Family utility function work? THe Atkinson Family Utility has the following functional form.

$$V^{\text{social}} = \left(\alpha \cdot A^{\lambda} + \beta \cdot B^{\lambda}\right)^{\frac{1}{\lambda}}$$

Several key issues here:

1. V^{social} is the utility of some social planner
2. A and B are allocations for Alex and Ben.
3. α and β are biases that a social planner has for Alex and Ben: $\alpha + \beta = 1$, $\alpha > 0$, and $\beta > 0$
4. $-\infty < \lambda \leq 1$ is a measure of inequality aversion
    - $\lambda = 1$ is when the planner cares about weighted total allocations (efficient, Utilitarian)
    - $\lambda = -\infty$ is when the planner cares about only the minimum between A and B allocations (equality, Rawlsian)

What if only care about Alex? Clearly, if the planner only cares about Ben, $\beta = 1$, then:

$$V^{\text{social}} = \left(B^{\lambda}\right)^{\frac{1}{\lambda}} = B$$

Clearly, regardless of the value of λ, as B increases V increases. What Happens to V when A or B increases? What is the derivative of V with respect to A or B?

$$\frac{\partial V}{\partial A} = \frac{1}{\lambda}\left(\alpha A^{\lambda} + \beta B^{\lambda}\right)^{\frac{1}{\lambda}-1} \cdot \lambda \alpha A^{\lambda-1}$$

$$\frac{\partial V}{\partial A} = \left(\alpha A^{\lambda} + \beta B^{\lambda}\right)^{\frac{1-\lambda}{\lambda}} \cdot \alpha A^{\lambda-1} > 0$$

Note that $\frac{\partial V}{\partial A} > 0$. When $\lambda < 0$, $Z^{\lambda} > 0$. For example $10^{-2} = \frac{1}{100}$. And For example $0.1^{\frac{3}{-2}} = \frac{1}{0.1^{1.5}}$. Still Positive.

While the overall V increases with increasing A, but if we did not have the outter power term, the situation is different. In particular, when $\lambda < 0$:

$$\text{if } \lambda < 0 \ \text{ then } \ \frac{d\left(\alpha A^{\lambda} + \beta B^{\lambda}\right)}{dA} = \alpha \lambda A^{\lambda-1} < 0$$

Without the outter $\frac{1}{\lambda}$ power, negative λ would lead to decreasing weighted sum. But:

$$\text{if } \lambda < 0 \ \text{ then } \ \frac{dG^{\frac{1}{\lambda}}}{dG} = \frac{1}{\lambda} \cdot G^{\frac{1-\lambda}{\lambda}} < 0$$

so when G is increasing and $\lambda < 0$, V would decrease. But when $G(A, B)$ is decreasing, as is the case with increasing A when $\lambda < 0$, V will actually increase. This confirms that $\frac{\partial V}{\partial A} > 0$ for $\lambda < 0$. The result is symmetric for $\lambda > 0$.

8.3.3.5 Indifference Curve Graph

Given V^{*}, we can show the combinations of A and B points that provide the same utility. We want to be able to potentially draw multiple indifference curves at the same time. Note that indifference curves are defined by α, λ only. Each indifference curve is a set of A and B coordinates. So to generate multiple indifference curves means to generate many sets of A, B associated with different planner preferences, and then these could be graphed out.

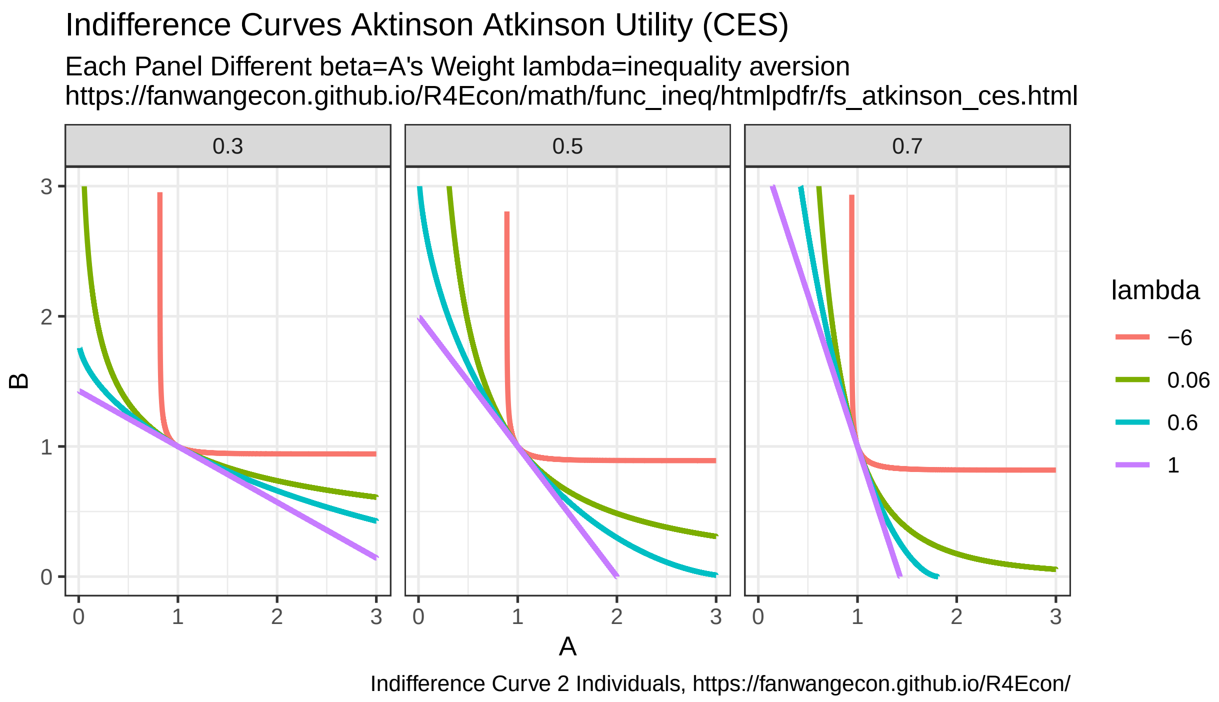
```
# A as x-axis, need bounds on A
fl_A_min = 0.01
fl_A_max = 3
it_A_grid = 50000
```

```
# Define parameters
# ar_lambda <- 1 - (10^(c(seq(-2,2, length.out=3))))
ar_lambda <- c(1, 0.6, 0.06, -6)
ar_beta <- seq(0.25, 0.75, length.out = 3)
ar_beta <- c(0.3, 0.5, 0.7)
ar_v_star <- seq(1, 2, length.out = 1)
tb_pref <- as_tibble(cbind(ar_lambda)) %>%
  expand_grid(ar_beta) %>% expand_grid(ar_v_star) %>%
  rename_all(~c('lambda', 'beta', 'vstar')) %>%
  rowid_to_column(var = "indiff_id")

# Generate indifference points with apply and anonymous function
# tb_pref, whatever is selected from it, must be all numeric
# if there are strings, would cause conversion error.
ls_df_indiff <- apply(tb_pref, 1, function(x){
  indiff_id <- x[1]
  lambda <- x[2]
  beta <- x[3]
  vstar <- x[4]
  ar_fl_A_indiff <- seq(fl_A_min, fl_A_max, length.out=it_A_grid)
  ar_fl_B_indiff <- (((vstar^lambda) -
                      (beta*ar_fl_A_indiff^(lambda)))/(1-beta))^(1/lambda)
  mt_A_B_indiff <- cbind(indiff_id, lambda, beta, vstar,
                         ar_fl_A_indiff, ar_fl_B_indiff)
  colnames(mt_A_B_indiff) <- c('indiff_id', 'lambda', 'beta', 'vstar',
                               'indiff_A', 'indiff_B')
  tb_A_B_indiff <- as_tibble(mt_A_B_indiff) %>%
    rowid_to_column(var = "A_grid_id") %>%
    filter(indiff_B >= 0 & indiff_B <= max(ar_fl_A_indiff))
  return(tb_A_B_indiff)
})
df_indiff <- do.call(rbind, ls_df_indiff) %>% drop_na()
```

Note that many more A grid points are needed to fully plot out the leontief line.

```
# Labeling
st_title <- paste0('Indifference Curves Aktinson Atkinson Utility (CES)')
st_subtitle <- paste0('Each Panel Different beta=A\'s Weight lambda=inequality aversion\n',
                      'https://fanwangecon.github.io/',
                      'R4Econ/math/func_ineq/htmlpdfr/fs_atkinson_ces.html')
st_caption <- paste0('Indifference Curve 2 Individuals, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'A'
st_y_label <- 'B'

# Graphing
plt_indiff <-
  df_indiff %>% mutate(lambda = as_factor(lambda),
                       beta = as_factor(beta),
                       vstar = as_factor(vstar)) %>%
  ggplot(aes(x=indiff_A, y=indiff_B,
             colour=lambda)) +
  facet_wrap( ~ beta) +
  geom_line(size=1) +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption) +
  theme_bw()
```

```r
# show
print(plt_indiff)
```



Chapter 9

Statistics

9.1 Random Draws

9.1.1 Randomly Perturbing a Parameter

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

9.1.1.1 Perturbation Normally with Log-Normal Magnitude Scaling Value

During estimation, we have some starting estimation parameter value. We want to do multi-start estimation by perturbing initial starting points. The perturbing process follows the rules specified below, implemented in the function below.

1. Select from 0 to 1, 0 closest to the existing parameter value, 1 very far from it.
2. Log normal distribution with 1st quartile = 0.185, and mu of normal = 0. The value from (1) correspond to a cumulative mass point of this log normal distribution.
3. Draw a value randomly from standard normal
4. Transform the randomly drawn value to current parameter scale with inverse z-score, the resulting value is the parameter of interest.

Test and implement the ideas above.

```
# Step 0
ar_fl_original_param <- c(-100, -10, -1, -0.1, 0, 0.1, 1, 10, 100)
ar_fl_original_param <- c(-10, -0.1, 0, 0.1, 10)
# Step 1
ar_zero_to_one_select <- seq(1e-3, 1 - 1e-3, length.out = 11)
# Step 2
# Assume mean of normal = 0, with sdlog = 2, 25th percentile is 0.185
fl_sdlog <- 2.5
fl_p25_logn <- qlnorm(0.25, meanlog = 0, sdlog = fl_sdlog)
# Step 3
# random draw, for now fix at positive number, which means to "randomly" expand
fl_draw_znorm <- 1
# Step 4
mt_collect <- matrix(
  data = NA,
  nrow = length(ar_zero_to_one_select),
  ncol = length(ar_fl_original_param)
)
it_col_ctr <- 0
for (fl_original_param in ar_fl_original_param) {
  it_col_ctr <- it_col_ctr + 1
  # inverse z-score
```

zero_one_scalar	ori_val=-10	ori_val=-0.1	ori_val=0	ori_val=0.1	ori_val=10
0.0010	-10.00441	-0.1000441	0	0.1000441	10.00441
0.1008	-10.41068	-0.1041068	0	0.1041068	10.41068
0.2006	-11.22616	-0.1122616	0	0.1122616	11.22616
0.3004	-12.70326	-0.1270326	0	0.1270326	12.70326
0.4002	-15.31489	-0.1531489	0	0.1531489	15.31489
0.5000	-20.00000	-0.2000000	0	0.2000000	20.00000
0.5998	-28.81508	-0.2881508	0	0.2881508	28.81508
0.6996	-46.99235	-0.4699235	0	0.4699235	46.99235
0.7994	-91.55561	-0.9155561	0	0.9155561	91.55561
0.8992	-253.49612	-2.5349612	0	2.5349612	253.49612
0.9990	-22665.67969	-226.6567969	0	226.6567969	22665.67969

```
  ar_logn_coef_of_var <- qlnorm(1-ar_zero_to_one_select, meanlog = 0, sdlog = fl_sdlog)
  ar_logn_sd <- fl_original_param/ar_logn_coef_of_var
  ar_param_perturbed <- fl_draw_znorm * ar_logn_sd + fl_original_param
  # fill matrix
  mt_collect[, it_col_ctr] <- (ar_param_perturbed)
}
# Out to table
ar_st_varnames <- c("zero_one_scalar", paste0("ori_val=", ar_fl_original_param))
# Combine to tibble, add name col1, col2, etc.
tb_collect <- as_tibble(cbind(ar_zero_to_one_select, mt_collect)) %>%
  rename_all(~ c(ar_st_varnames))
# Display
kable(tb_collect) %>% kable_styling_fc()
```

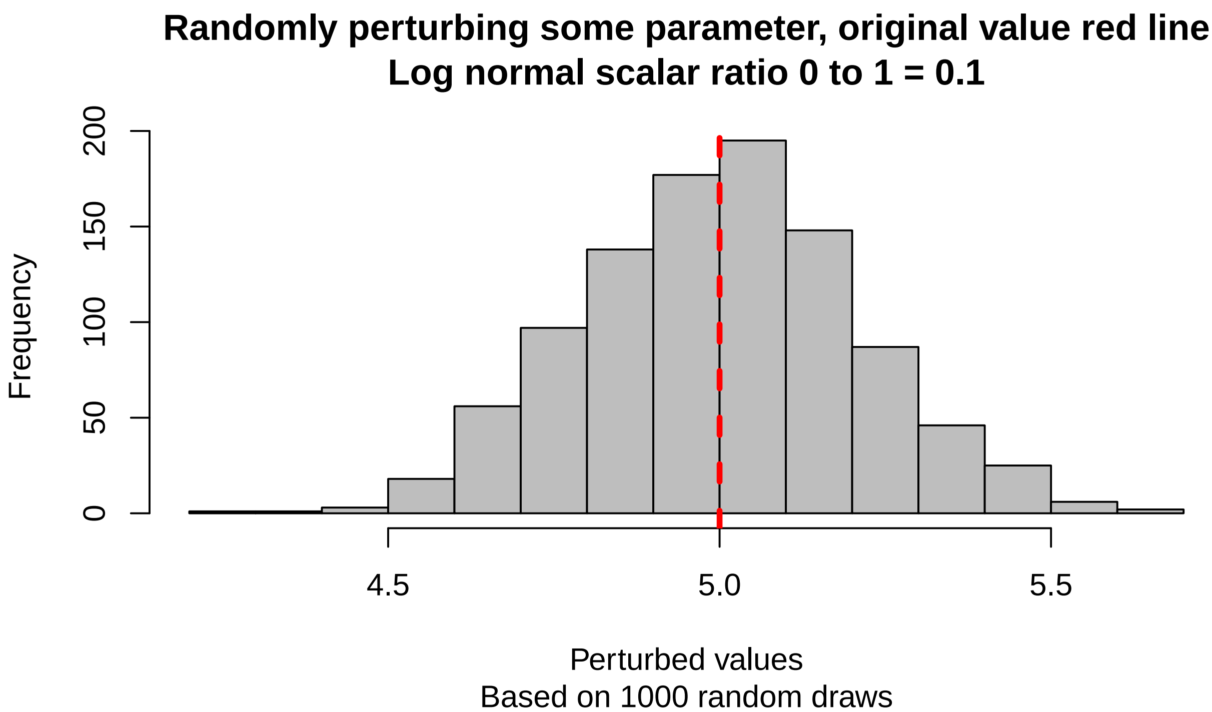
Implement the above idea with a function.

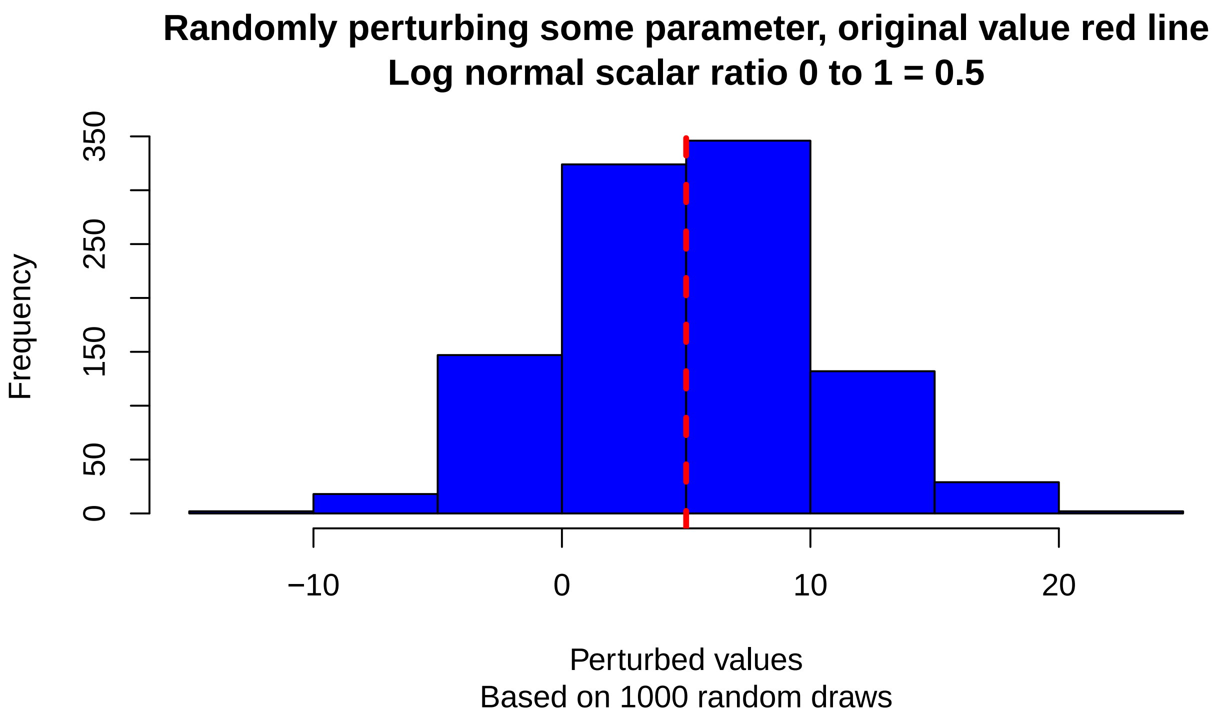
```
ffi_param_logn_perturber <- function(
  param_original=5, scaler_0t1=0.5,
  it_rand_seed=1, fl_sdlog=2.5, fl_min_quantile=1e-3) {
  #' @param float original current parameter value to be perturbed
  #' @param scaler_0t1 float, must be between 0 to 1, 0 means don't scale much, 1 mean a lot
  #' @param it_rand_seed integer randomly sperturbing seed
  #' @param fl_sdlog float the sdlog parameter
  #' @param fl_min_quantile float minimum quantile point (and 1 - max) to allow for selecting 0 and

  # Draw randomly
  set.seed(it_rand_seed)
  fl_draw_znorm <- rnorm(1)
  # logn value at quantile
  scaler_0t1 <- scaler_0t1*(1-fl_min_quantile*2) + fl_min_quantile
  logn_coef_of_var <- qlnorm(1-scaler_0t1, meanlog = 0, sdlog = fl_sdlog)
  # Coefficient of variation
  ar_logn_sd <- param_original/logn_coef_of_var
  # Invert z-score
  param_perturbed <- fl_draw_znorm * ar_logn_sd + param_original

  return(param_perturbed)
}
```

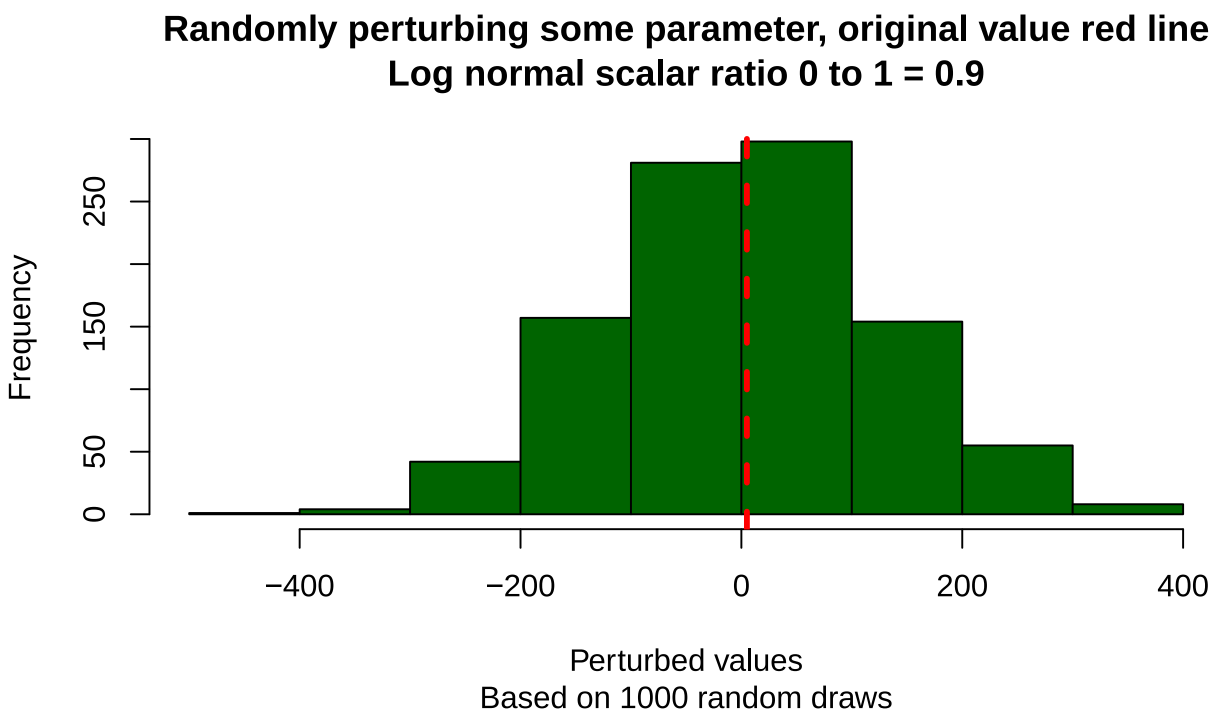
Test the function with differently randomly drawn parameters, and visualize.

```
# Start image
# Loop over different scalars
param_original <- 5
ar_scaler_0t1 <- c(0.1, 0.5, 0.9)
```

```
ar_color_strs <- c("gray", "blue", "darkgreen")
it_scalar_ctr <- 0
for (scaler_0t1 in ar_scaler_0t1) {
  it_scalar_ctr <- it_scalar_ctr + 1

  # Generate differently perturbed parameters
  ar_param_perturbed <- c()
  for (it_rand_seed in seq(1, 1000)) {
    param_perturbed <- ffi_param_logn_perturber(
      param_original, scaler_0t1, it_rand_seed=it_rand_seed)
    ar_param_perturbed <- c(ar_param_perturbed, param_perturbed)
  }

  # Line through origin
  par(mfrow = c(1, 1))
  hist(ar_param_perturbed, col = ar_color_strs[it_scalar_ctr],
       ylab = "", xlab = "", main = "")
  # Original parameter line
  abline(
    v = param_original,
    col = "red", lwd = 3, lty = 2
  )
  # Titles
  title(
    main = paste0(
      "Randomly perturbing some parameter, original value red line\n",
      "Log normal scalar ratio 0 to 1 = ", scaler_0t1
    ),
    sub = paste0(
      "Based on 1000 random draws"
    ),
    xlab = "Perturbed values", ylab = "Frequency"
  )
}
```







9.2 Distributions

9.2.1 Integrate Over Normal Guassian Process Shock

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Some Common parameters

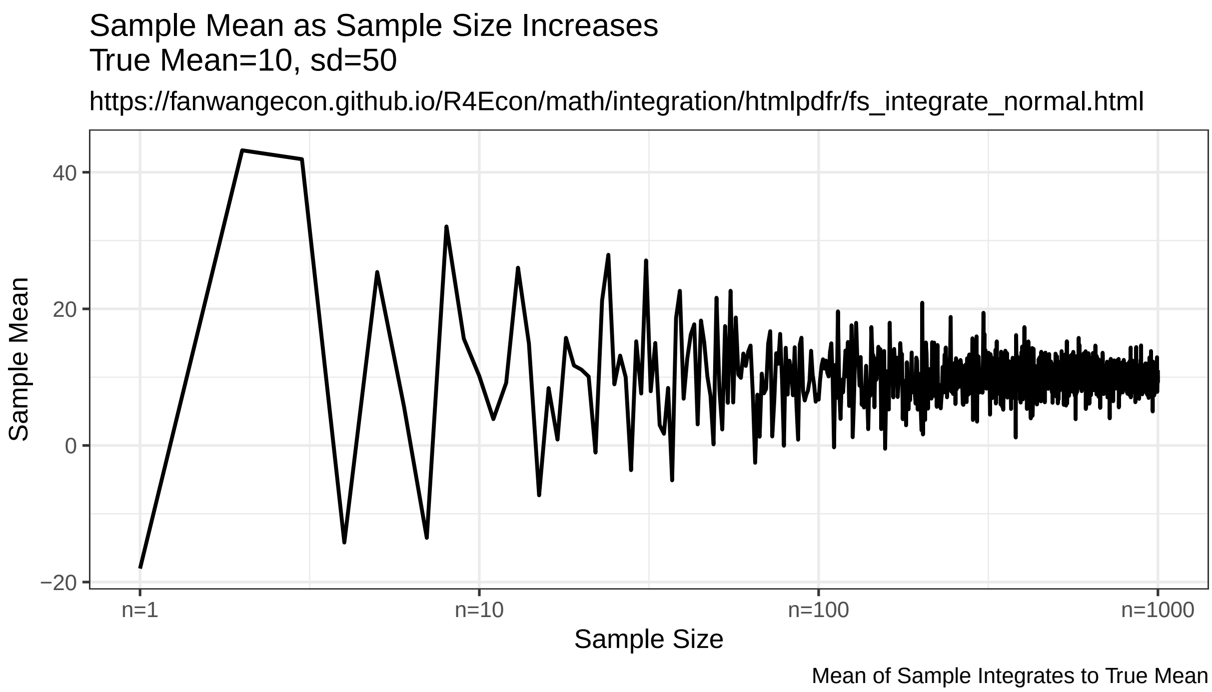
```
fl_eps_mean = 10
fl_eps_sd = 50
```

```
fl_cdf_min = 0.000001
fl_cdf_max = 0.999999
ar_it_draws <- seq(1, 1000)
```

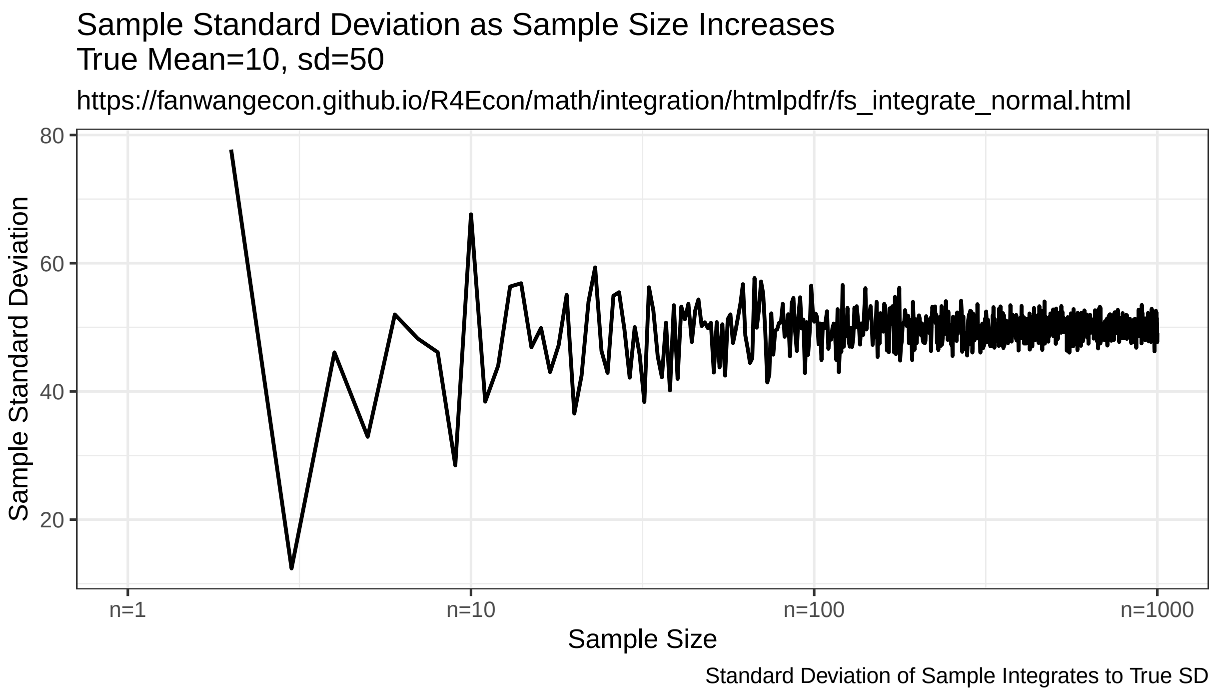
9.2.1.1 Randomly Sample and Integrate (Monte Carlo Integration)

Compare randomly drawn normal shock mean and known mean. How does simulated mean change with draws. Actual integral equals to 10, as sample size increases, the sample mean approaches the integration results, but this is expensive, even with ten thousand draws, not very exact.

```
# Simulate Draws
set.seed(123)
ar_fl_means <-
  sapply(ar_it_draws, function(x)
    return(mean(rnorm(x[1], mean=fl_eps_mean, sd=fl_eps_sd))))
ar_fl_sd <-
  sapply(ar_it_draws, function(x)
    return(sd(rnorm(x[1], mean=fl_eps_mean, sd=fl_eps_sd))))

mt_sample_means <- cbind(ar_it_draws, ar_fl_means, ar_fl_sd)
colnames(mt_sample_means) <- c('draw_count', 'mean', 'sd')
tb_sample_means <- as_tibble(mt_sample_means)

# Graph
# x-labels
x.labels <- c('n=1', 'n=10', 'n=100', 'n=1000')
x.breaks <- c(1, 10, 100, 1000)

# Shared Subtitle
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/math/integration/htmlpdfr/fs_integrate_normal.html')

# Shared Labels
slb_title_shr = paste0('as Sample Size Increases\n',
                       'True Mean=', fl_eps_mean,', sd=',fl_eps_sd)
slb_xtitle = paste0('Sample Size')

# Graph Results--Draw
plt_mean <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=mean)) +
  geom_line(size=0.75) +
  labs(title = paste0('Sample Mean ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Sample Mean',
       caption = 'Mean of Sample Integrates to True Mean') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_mean)
```




```r
plt_sd <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=sd)) +
  geom_line(size=0.75) +
  labs(title = paste0('Sample Standard Deviation ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Sample Standard Deviation',
       caption = 'Standard Deviation of Sample Integrates to True SD') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_sd)
```

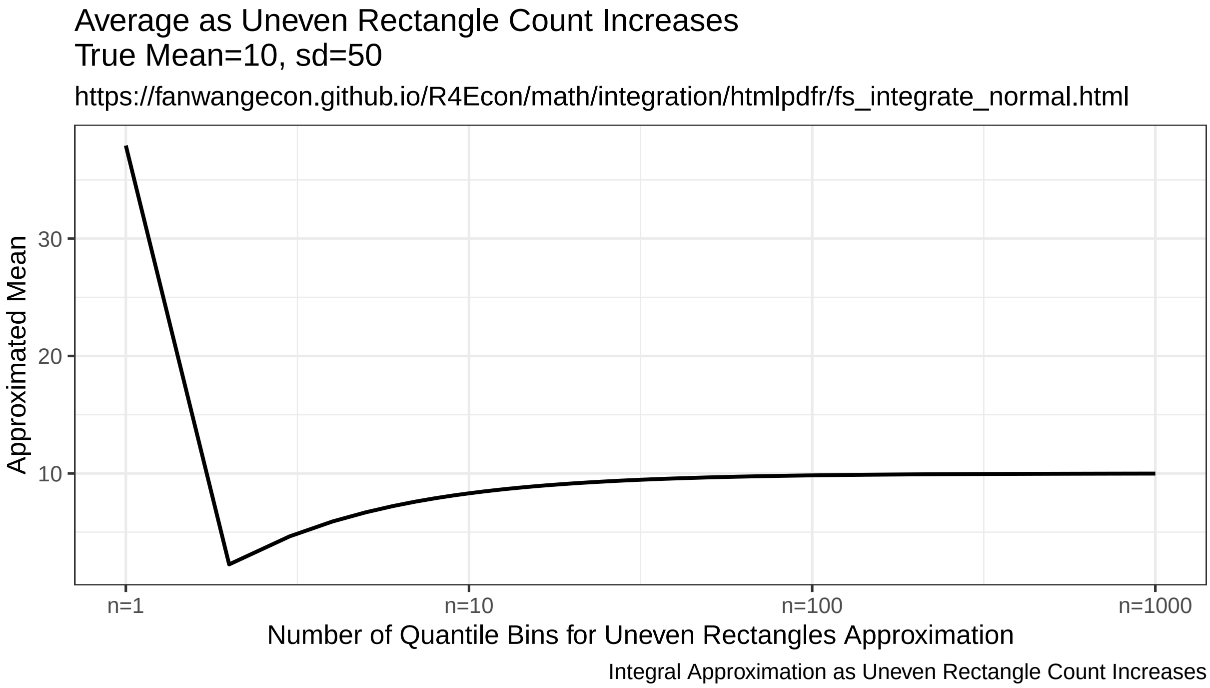


9.2.1.2 Integration By Symmetric Uneven Rectangle

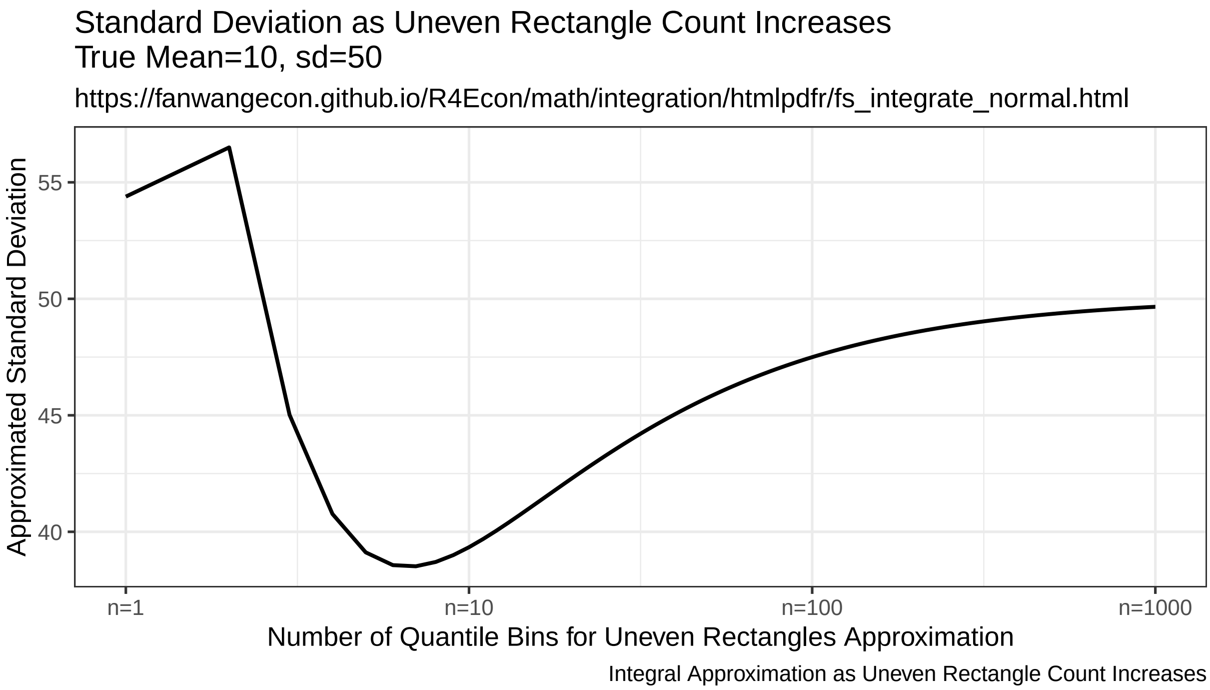
Draw on even grid from close to 0 to close to 1. Get the corresponding x points to these quantile levels. Distance between x points are not equi-distance but increasing and symmetric away from the mean. Under this approach, each rectangle aims to approximate the same area.

Resulting integration is rectangle based, but rectangle width differ. The rectangles have wider width as they move away from the mean, and thinner width close to the mean. This is much more stable than the random draw method, but note that it converges somewhat slowly to true values as well.

```r
mt_fl_means <-
  sapply(ar_it_draws, function(x) {

    fl_prob_break = (fl_cdf_max - fl_cdf_min)/(x[1])
    ar_eps_bounds <- qnorm(seq(fl_cdf_min, fl_cdf_max,
                               by=(fl_cdf_max - fl_cdf_min)/(x[1])),
                           mean = fl_eps_mean, sd = fl_eps_sd)
    ar_eps_val <- (tail(ar_eps_bounds, -1) + head(ar_eps_bounds, -1))/2
    ar_eps_prb <- rep(fl_prob_break/(fl_cdf_max - fl_cdf_min), x[1])
    ar_eps_fev <- dnorm(ar_eps_val,
                        mean = fl_eps_mean, sd = fl_eps_sd)

    fl_cdf_total_approx <- sum(ar_eps_fev*diff(ar_eps_bounds))
    fl_mean_approx <- sum(ar_eps_val*(ar_eps_fev*diff(ar_eps_bounds)))
    fl_sd_approx <- sqrt(sum((ar_eps_val-fl_mean_approx)^2*(ar_eps_fev*diff(ar_eps_bounds))))

    return(list(cdf=fl_cdf_total_approx, mean=fl_mean_approx, sd=fl_sd_approx))
  })

mt_sample_means <- cbind(ar_it_draws, as_tibble(t(mt_fl_means)) %>% unnest())
colnames(mt_sample_means) <- c('draw_count', 'cdf', 'mean', 'sd')
tb_sample_means <- as_tibble(mt_sample_means)

# Graph
# x-labels
x.labels <- c('n=1', 'n=10', 'n=100', 'n=1000')
x.breaks <- c(1, 10, 100, 1000)

# Shared Labels
slb_title_shr = paste0('as Uneven Rectangle Count Increases\n',
                       'True Mean=', fl_eps_mean,', sd=',fl_eps_sd)
slb_xtitle = paste0('Number of Quantile Bins for Uneven Rectangles Approximation')

# Graph Results--Draw
plt_mean <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=mean)) +
  geom_line(size=0.75) +
  labs(title = paste0('Average ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Approximated Mean',
       caption = 'Integral Approximation as Uneven Rectangle Count Increases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_mean)
```

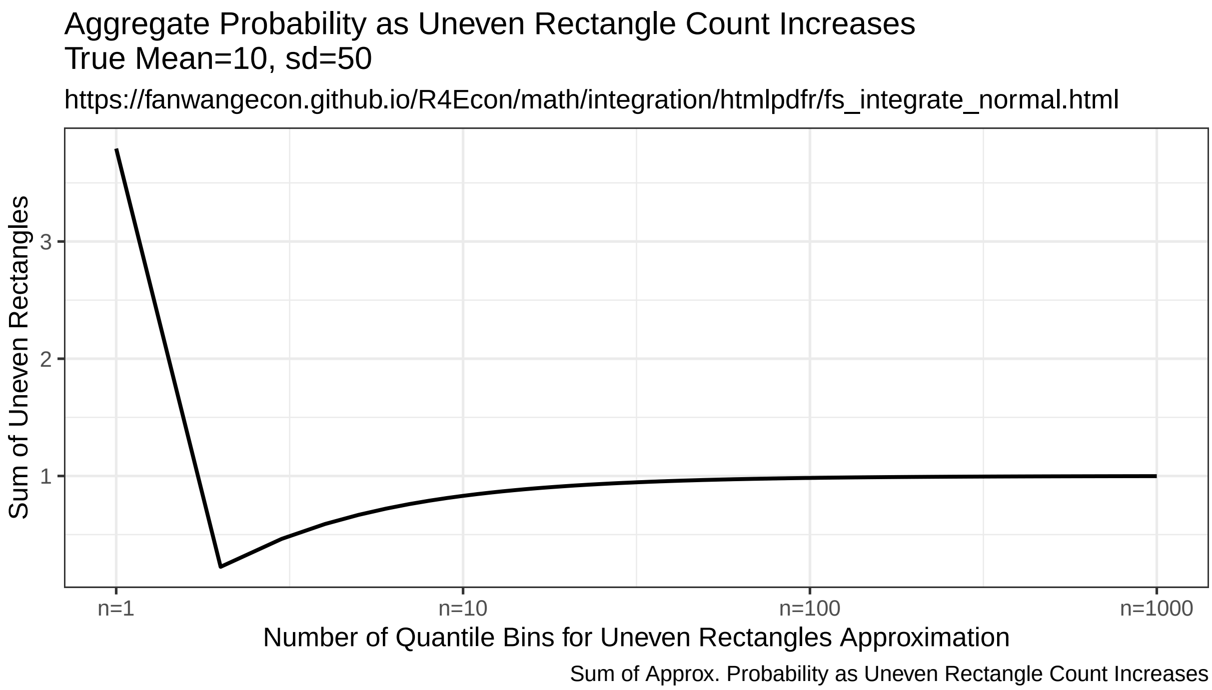



```
plt_sd <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=sd)) +
  geom_line(size=0.75) +
  labs(title = paste0('Standard Deviation ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Approximated Standard Deviation',
       caption = 'Integral Approximation as Uneven Rectangle Count Increases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_sd)
```




```
plt_cdf <- tb_sample_means %>%
```

```
  ggplot(aes(x=draw_count, y=cdf)) +
  geom_line(size=0.75) +
  labs(title = paste0('Aggregate Probability ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Sum of Uneven Rectangles',
       caption = 'Sum of Approx. Probability as Uneven Rectangle Count Increases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_cdf)
```

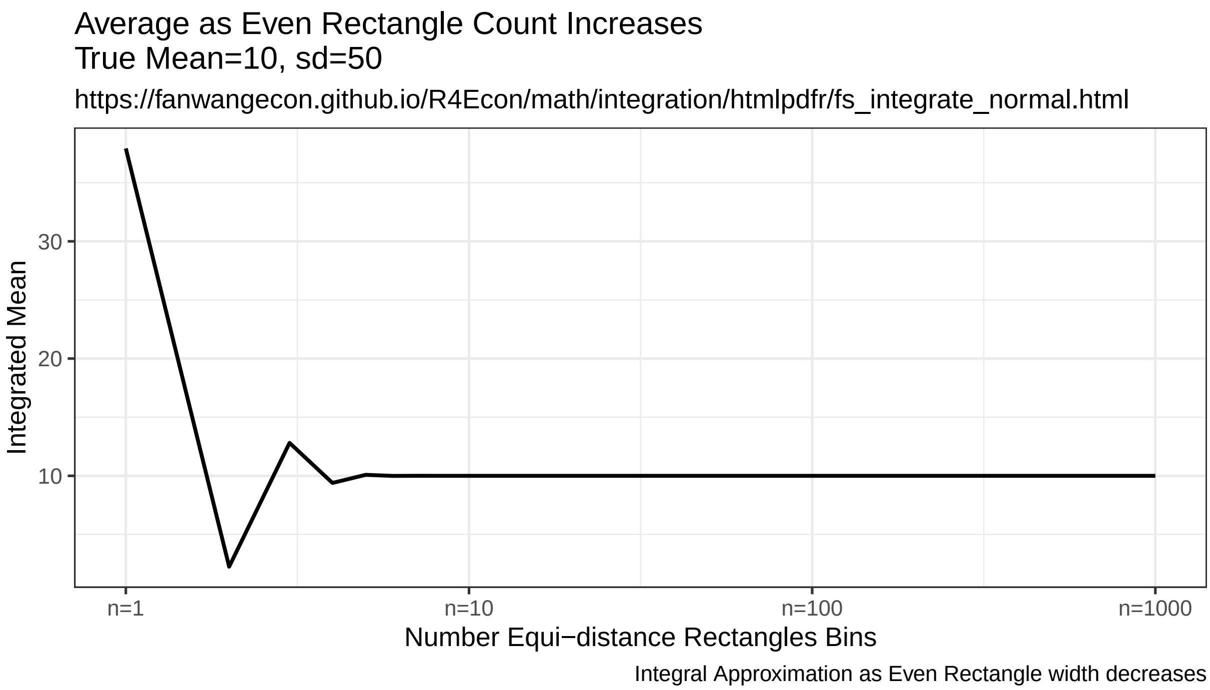


9.2.1.3 Integration By Constant Width Rectangle (Trapezoidal rule)

This is implementing even width recentagle, even along x-axix. Rectangle width are the same, height is $f(x)$. This is even width, but uneven area. Note that this method approximates the true answer much better and more quickly than the prior methods.

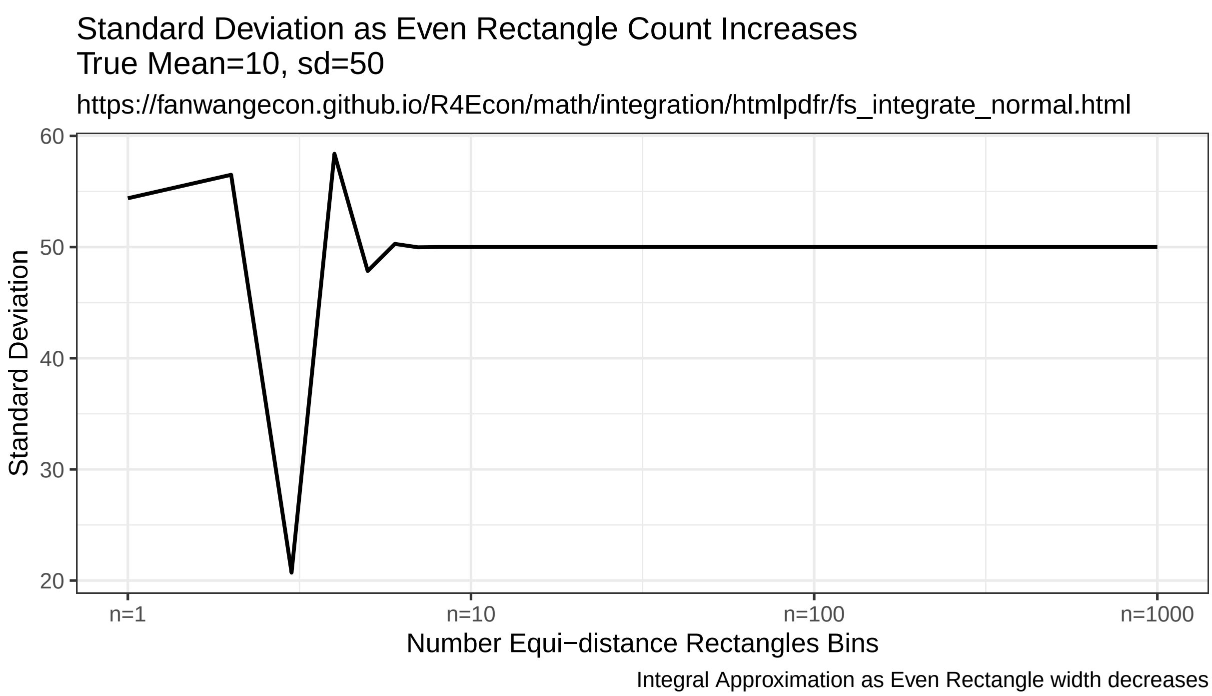
```
mt_fl_means <-
  sapply(ar_it_draws, function(x) {

    fl_eps_min <- qnorm(fl_cdf_min, mean = fl_eps_mean, sd = fl_eps_sd)
    fl_eps_max <- qnorm(fl_cdf_max, mean = fl_eps_mean, sd = fl_eps_sd)
    fl_gap <- (fl_eps_max-fl_eps_min)/(x[1])
    ar_eps_bounds <- seq(fl_eps_min, fl_eps_max, by=fl_gap)
    ar_eps_val <- (tail(ar_eps_bounds, -1) + head(ar_eps_bounds, -1))/2
    ar_eps_prb <- dnorm(ar_eps_val, mean = fl_eps_mean, sd = fl_eps_sd)*fl_gap

    fl_cdf_total_approx <- sum(ar_eps_prb)
    fl_mean_approx <- sum(ar_eps_val*ar_eps_prb)
    fl_sd_approx <- sqrt(sum((ar_eps_val-fl_mean_approx)^2*ar_eps_prb))

    return(list(cdf=fl_cdf_total_approx, mean=fl_mean_approx, sd=fl_sd_approx))
  })

mt_sample_means <- cbind(ar_it_draws, as_tibble(t(mt_fl_means)) %>% unnest())
```

```
colnames(mt_sample_means) <- c('draw_count', 'cdf', 'mean', 'sd')
tb_sample_means <- as_tibble(mt_sample_means)

# Graph
# x-labels
x.labels <- c('n=1', 'n=10', 'n=100', 'n=1000')
x.breaks <- c(1, 10, 100, 1000)

# Shared Labels
slb_title_shr = paste0('as Even Rectangle Count Increases\n',
                       'True Mean=', fl_eps_mean,', sd=',fl_eps_sd)
slb_xtitle = paste0('Number Equi-distance Rectangles Bins')

# Graph Results--Draw
plt_mean <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=mean)) +
  geom_line(size=0.75) +
  labs(title = paste0('Average ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Integrated Mean',
       caption = 'Integral Approximation as Even Rectangle width decreases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_mean)
```

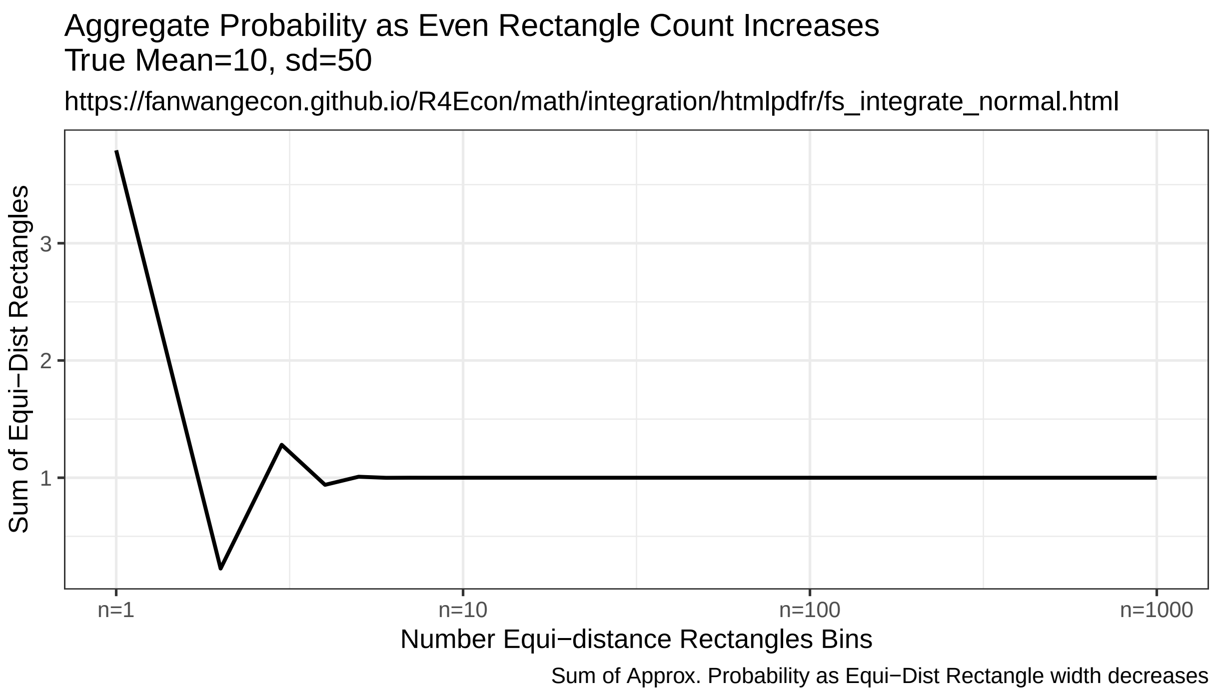



```
plt_sd <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=sd)) +
  geom_line(size=0.75) +
  labs(title = paste0('Standard Deviation ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Standard Deviation',
       caption = 'Integral Approximation as Even Rectangle width decreases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
```

```
  theme_bw()
print(plt_sd)
```




```
plt_cdf <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=cdf)) +
  geom_line(size=0.75) +
  labs(title = paste0('Aggregate Probability ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Sum of Equi-Dist Rectangles',
       caption = 'Sum of Approx. Probability as Equi-Dist Rectangle width decreases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_cdf)
```



9.3 Discrete Random Variable

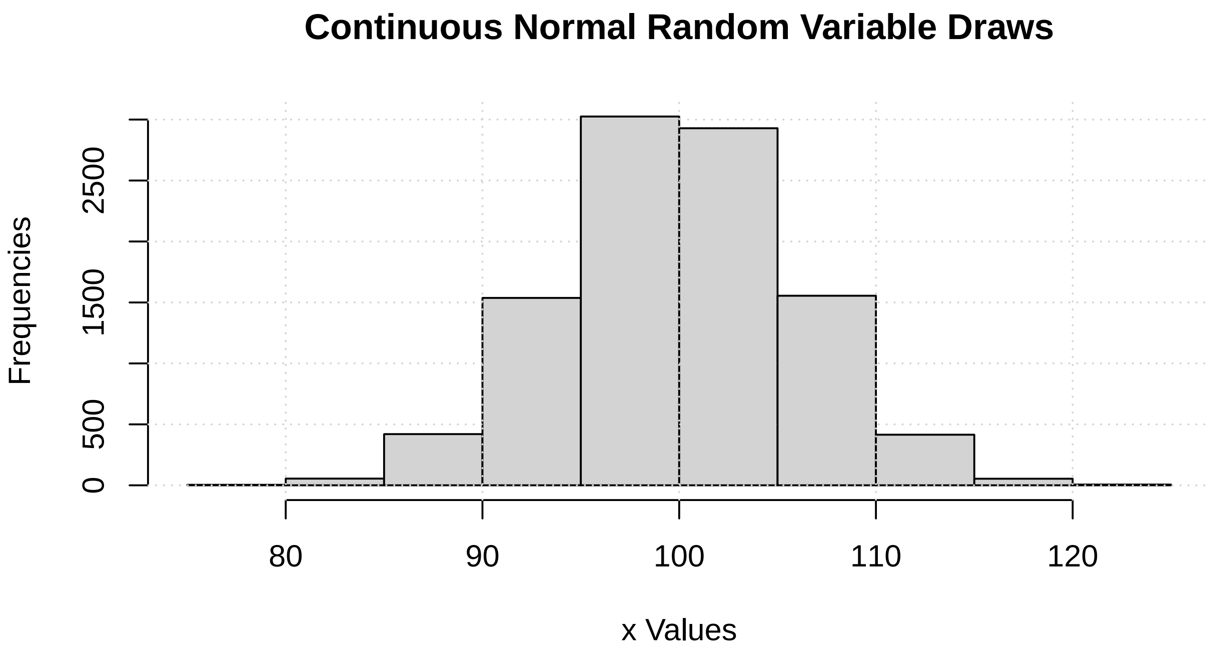
9.3.1 Discrete Approximation of Continuous Random Variables

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

9.3.1.1 Use Binomial Discrete Random Variable to Approximate Continuous Normal

First, draw from a Continuous Random Variable. Sample N draws from a normal random variable.

```
# Random normal Data Vector (not equal outcomes)
set.seed(123)
it_sample_N <- 10000
fl_rnorm_mean <- 100
fl_rnorm_sd <- 6
ar_data_rnorm <- rnorm(it_sample_N, mean = fl_rnorm_mean, sd = fl_rnorm_sd)
# Visualize
par(new = FALSE)
hist(ar_data_rnorm, xlab = "x Values", ylab = "Frequencies", main = "")
title(main = "Continuous Normal Random Variable Draws")
grid()
```



We use the binomial to approximate the normal distribution. Let μ and σ be the mean and standard deviations of the normal random variable, and n and p be the number of "trials" and the "probability-of-success" for the binomial distribution. We know that these relationships are approximately true, :

$$\mu = n \cdot p$$
$$n = \frac{\mu}{p}$$
$$\sigma^2 = n \cdot p \cdot (1-p) = \mu \cdot (1-p)$$

Given these, we have can translate between the normal random variable's parameters and the binomial discrete random variable's parameters:

$$p = 1 - \frac{\sigma^2}{\mu}$$
$$n = \frac{\mu}{1 - \frac{\sigma^2}{\mu}} = \frac{\mu}{\frac{\mu - \sigma^2}{\mu}} = \frac{\mu^2}{\mu - \sigma^2}$$

There are two important aspects to note here:

1. Since p must be positive, this means $\frac{\sigma^2}{\mu} < 1$ and $\sigma^2 < \mu$, which is the condition for the above transformation to work.
2. The binomial discrete random variable will have non-zero mass for very small probability events at the left-tail. These very low outcome events are highly unlikely to be observed or drawn from sampling the continuous random variable. The presence of these left-tail values might impact the computation of certain statistics, for example the Atkinson Index for highly inequality averse planners.

Create a function for converting between normal and binomial parameters:

```
ffi_binom_approx_nomr <- function(fl_rnorm_mean, fl_rnorm_sd) {
  #' @param fl_rnorm_mean float normal mean
  #' @param fl_rnorm_sd float normal standard deviation
  if (fl_rnorm_mean <= fl_rnorm_sd^2) {
    stop("Normal mean must be larger than the variance for conversion")
  } else {
    # Use binomial to approximate normal
    fl_p_binom <- 1 - fl_rnorm_sd^2 / fl_rnorm_mean
    fl_n_binom <- round(fl_rnorm_mean^2 / (fl_rnorm_mean - fl_rnorm_sd^2))
```

```
    fl_binom_mean <- fl_n_binom * fl_p_binom
    fl_binom_sd <- sqrt(fl_n_binom * fl_p_binom * (1 - fl_p_binom))
    # return
    return(list(
      fl_p_binom = fl_p_binom, fl_n_binom = fl_n_binom,
      fl_binom_mean = fl_binom_mean, fl_binom_sd = fl_binom_sd
    ))
  }
}
```

Call the function to generate binomial parameters and generate the resulting binomial discrete random variable:

```
# with these parameters, does not work
# ls_binom_params <- ffi_binom_approx_nomr(fl_rnorm_mean = 10, fl_rnorm_sd = 3)
# Call function with parameters, defined earlier
ls_binom_params <- ffi_binom_approx_nomr(fl_rnorm_mean, fl_rnorm_sd)
fl_binom_mean <- ls_binom_params$fl_binom_mean
fl_binom_sd <- ls_binom_params$fl_binom_sd
fl_n_binom <- ls_binom_params$fl_n_binom
fl_p_binom <- ls_binom_params$fl_p_binom
# Mean and sd, note that these are the same as values defined earlier
print(paste0("BINOMI mean=",
             ls_binom_params$fl_binom_mean,
             ", fl_rnorm_mean=",
             fl_rnorm_mean))
```

```
## [1] "BINOMI mean=99.84, fl_rnorm_mean=100"
```

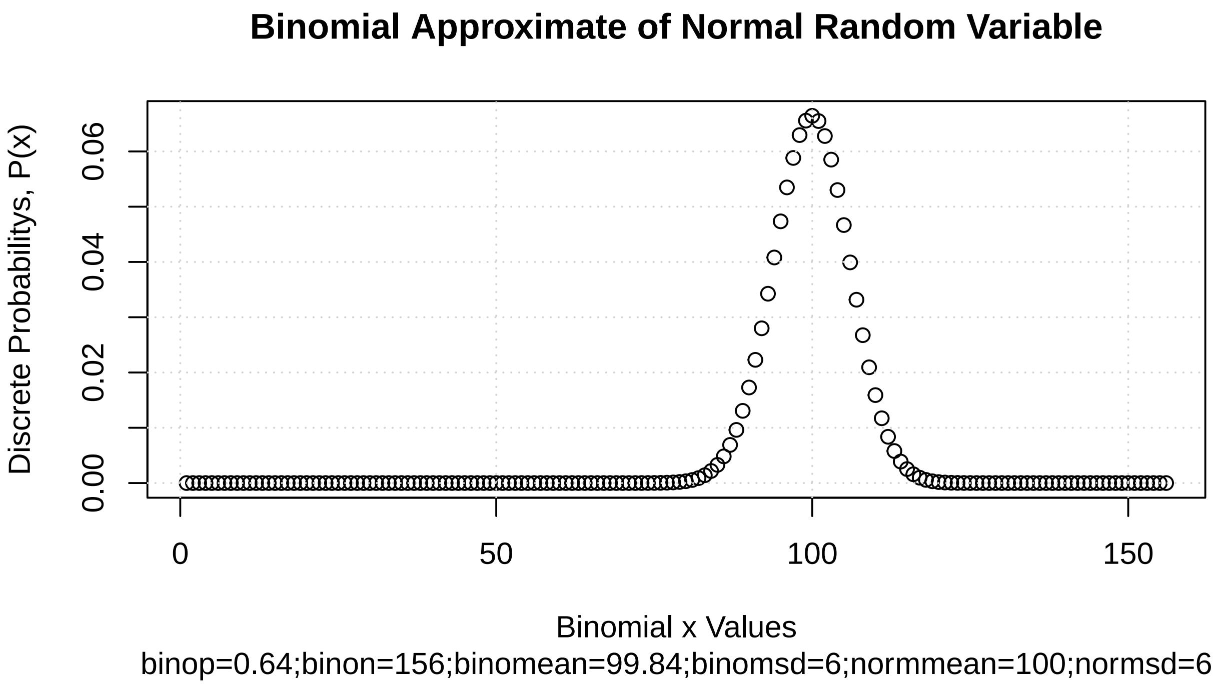
```
print(paste0("BINOMI sd=", ls_binom_params$fl_binom_sd,
             ", fl_binom_sd=", fl_binom_sd))
```

```
## [1] "BINOMI sd=5.99519807846246, fl_binom_sd=5.99519807846246"
```

```
# drv = discrete random variable
ar_drv_rbinom_xval <- seq(1, fl_n_binom)
ar_drv_rbinom_prob <- dbinom(ar_drv_rbinom_xval,
  size = fl_n_binom, prob = fl_p_binom
)
# ignore weight at x=0
ar_drv_rbinom_prob <- ar_drv_rbinom_prob / sum(ar_drv_rbinom_prob)
```

Visualize the binomial discrete random variable:

```
# graph
par(new = FALSE)
ar_ylim <- c(0, 1)
plot(ar_drv_rbinom_xval, ar_drv_rbinom_prob,
  xlab = "Binomial x Values", ylab = "Discrete Probabilitys, P(x)"
)
title(
  main = paste0("Binomial Approximate of Normal Random Variable"),
  sub = paste0(
    "binop=", round(fl_p_binom, 2),
    ";binon=", round(fl_n_binom, 2),
    ";binomean=", round(fl_binom_mean, 2),
    ";binomsd=", round(fl_binom_sd, 2),
    ";normmean=", round(fl_rnorm_mean, 2), ";normsd=", round(fl_rnorm_sd, 2)
  )
)
grid()
```


Binomial Approximate of Normal Random Variable
Discrete Probabilitys, P(x)
0.00
0.02
0.04
0.06
0
50
100
150
Binomial x Values
binop=0.64;binon=156;binomean=99.84;binomsd=6;normmean=100;normsd=6

Chapter 10

Tables and Graphs

10.1 R Base Plots

10.1.1 Plot Curve, Line and Points

> Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Work with the R plot function.

10.1.1.1 One Point, One Line and Two Curves

- r curve on top of plot
- r plot specify pch lty both scatter and line
- r legend outside

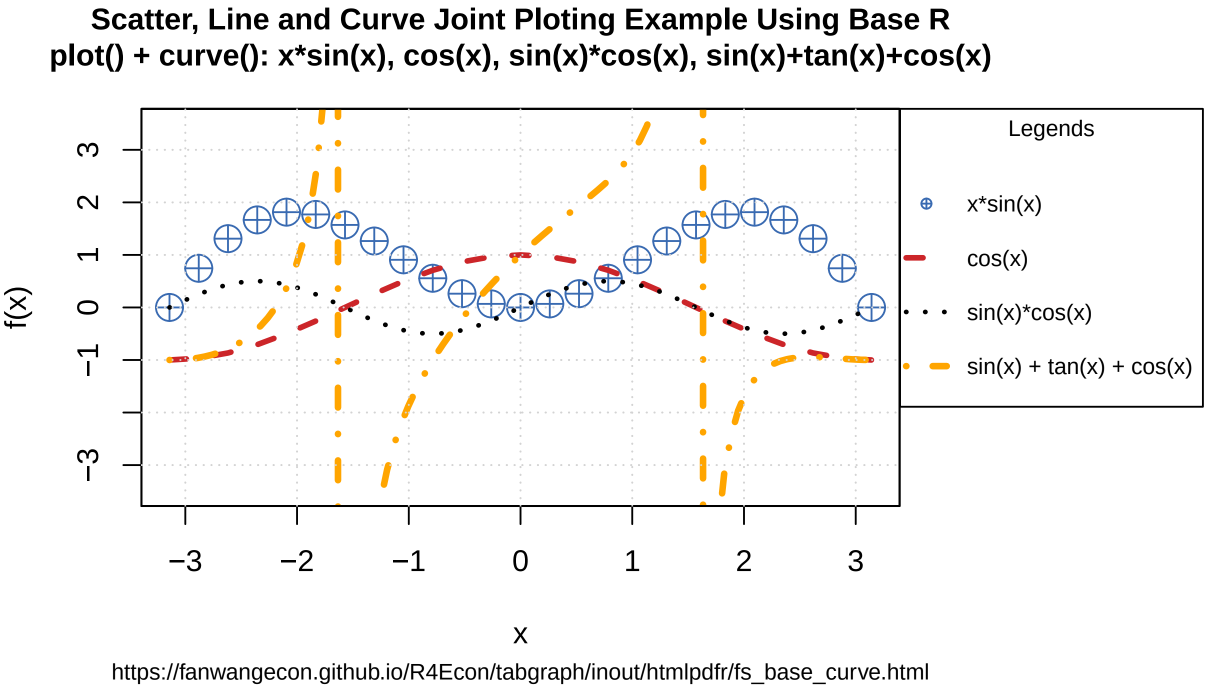
Jointly plot:

- 1 scatter plot
- 1 line plot
- 2 function curve plots

```
#######################################################
# First, Some common Labels:
#######################################################
# Labeling
st_title <- paste0('Scatter, Line and Curve Joint Ploting Example Using Base R\n',
                   'plot() + curve(): x*sin(x), cos(x), sin(x)*cos(x), sin(x)+tan(x)+cos(x)')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/inout/htmlpdfr/fs_base_curve.html')
st_x_label <- 'x'
st_y_label <- 'f(x)'

#######################################################
# Second, Generate the Graphs Functions and data points:
#######################################################
# x only used for Point 1 and Line 1
x <- seq(-1*pi, 1*pi, length.out=25)
# Line (Point) 1: Generate X and Y
y1 <- x*sin(x)
st_point_1_y_legend <- 'x*sin(x)'
# Line 2: Line Plot
y2 <- cos(x)
st_line_2_y_legend <- 'cos(x)'
# Line 3: Function
```

```
fc_sin_cos_diff <- function(x) sin(x)*cos(x)
st_line_3_y_legend <- 'sin(x)*cos(x)'
# Line 4: Function
fc_sin_cos_tan <- function(x) sin(x) + cos(x) + tan(x)
st_line_4_y_legend <- 'sin(x) + tan(x) + cos(x)'

#######################################################
# Third, set:
# - point shape and size: *pch* and *cex*
# - line type and width: *lty* and *lwd*
#######################################################
# http://www.sthda.com/english/wiki/r-plot-pch-symbols-the-different-point-shapes-available-in-r
# http://www.sthda.com/english/wiki/line-types-in-r-lty
# for colors, see: https://fanwangecon.github.io/M4Econ/graph/tools/fs_color.html
st_point_1_blue <- rgb(57/255,106/255,177/255)
st_line_2_red <- rgb(204/255, 37/255, 41/255,)
st_line_3_black <- 'black'
st_line_4_purple <- 'orange'

# point type
st_point_1_pch <- 10
# point size
st_point_1_cex <- 2

# line type
st_line_2_lty <- 'dashed'
st_line_3_lty <- 'dotted'
st_line_4_lty <- 'dotdash'
# line width
st_line_2_lwd <- 3
st_line_3_lwd <- 2.5
st_line_4_lwd <- 3.5

#######################################################
# Fourth: Share xlim and ylim
#######################################################
ar_xlim = c(min(x), max(x))
ar_ylim = c(-3.5, 3.5)

#######################################################
# Fifth: the legend will be long, will place it to the right of figure,
#######################################################
par(new=FALSE, mar=c(5, 4, 4, 10))

#######################################################
# Sixth, the four objects and do not print yet:
#######################################################
# pdf(NULL)
# Graph Scatter 1
plot(x, y1, type="p",
     col = st_point_1_blue,
     pch = st_point_1_pch, cex = st_point_1_cex,
     xlim = ar_xlim, ylim = ar_ylim,
     panel.first = grid(),
     ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
pl_scatter_1 <- recordPlot()
```

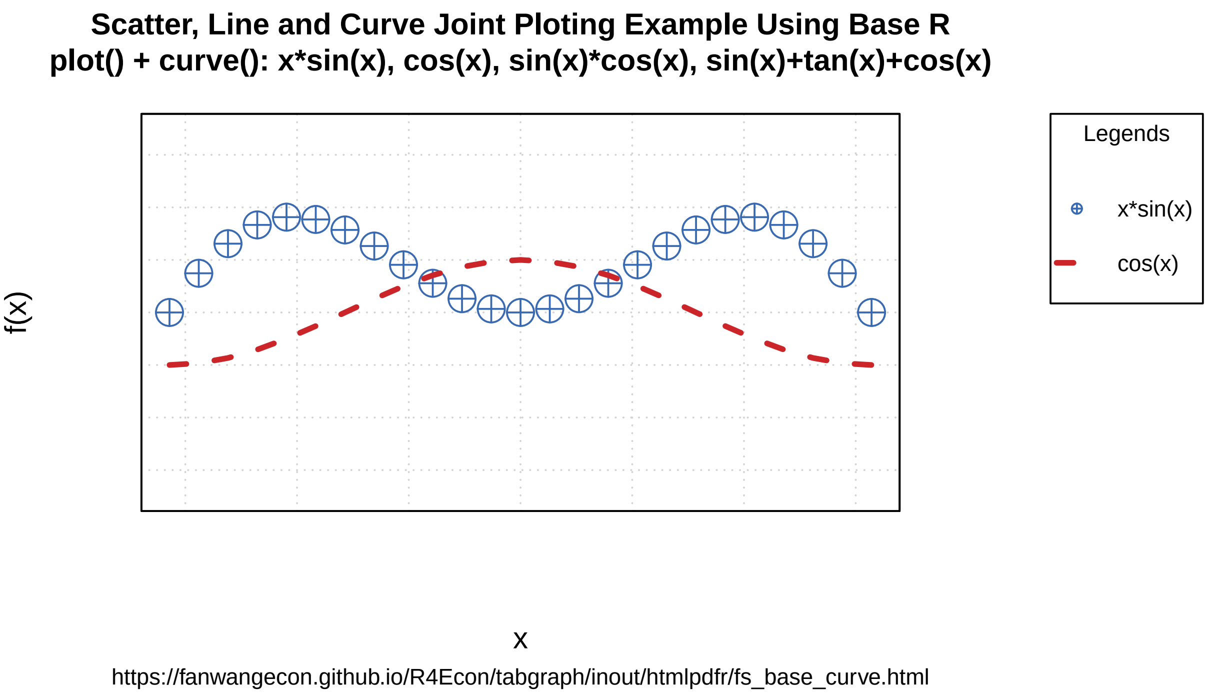
```
# Graph Line 2
par(new=T)
plot(x, y2, type="l",
     col = st_line_2_red,
     lwd = st_line_2_lwd, lty = st_line_2_lty,
     xlim = ar_xlim, ylim = ar_ylim,
     ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
pl_12 <- recordPlot()

# Graph Curve 3
par(new=T)
curve(fc_sin_cos_diff,
      col = st_line_3_black,
      lwd = st_line_3_lwd, lty = st_line_3_lty,
      from = ar_xlim[1], to = ar_xlim[2], ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
pl_123 <- recordPlot()

# Graph Curve 4
par(new=T)
curve(fc_sin_cos_tan,
      col = st_line_4_purple,
      lwd = st_line_4_lwd, lty = st_line_4_lty,
      from = ar_xlim[1], to = ar_xlim[2], ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
pl_1234 <- recordPlot()
# invisible(dev.off())

#######################################################
# Seventh, Set Title and Legend and Plot Jointly
#######################################################
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg = 1
fl_ces_fig_small = 0.75

# R Legend
title(main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
      cex.lab=fl_ces_fig_reg,
      cex.main=fl_ces_fig_reg,
      cex.sub=fl_ces_fig_small)
axis(1, cex.axis=fl_ces_fig_reg)
axis(2, cex.axis=fl_ces_fig_reg)
grid()

# Legend sizing CEX
legend("topright",
       inset=c(-0.4,0),
       xpd=TRUE,
       c(st_point_1_y_legend, st_line_2_y_legend, st_line_3_y_legend, st_line_4_y_legend),
       col = c(st_point_1_blue, st_line_2_red, st_line_3_black, st_line_4_purple),
       pch = c(st_point_1_pch, NA, NA, NA),
       cex = fl_ces_fig_small,
       lty = c(NA, st_line_2_lty, st_line_3_lty, st_line_4_lty),
       lwd = c(NA, st_line_2_lwd, st_line_3_lwd,st_line_4_lwd),
       title = 'Legends',
       y.intersp=2)
```




```
# record final plot
pl_1234_final <- recordPlot()
```

We used recordplot() earlier. So now we can print just the first two constructed plots.

```
#######################################################
# Eighth, Plot just the first two saved lines
#######################################################
# mar: margin, bottom, left, top, right
pl_12
# R Legend
par(new=T)
title(main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
      cex.lab = fl_ces_fig_reg,
      cex.main = fl_ces_fig_reg,
      cex.sub = fl_ces_fig_small)

# Legend sizing CEX
par(new=T)
legend("topright",
       inset=c(-0.4,0),
       xpd=TRUE,
       c(st_point_1_y_legend, st_line_2_y_legend),
       col = c(st_point_1_blue, st_line_2_red),
       pch = c(st_point_1_pch, NA),
       cex = fl_ces_fig_small,
       lty = c(NA, st_line_2_lty),
       lwd = c(NA, st_line_2_lwd),
       title = 'Legends',
       y.intersp=2)
```



10.2 ggplot Line Related Plots

10.2.1 ggplot Line Plot Basics

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

10.2.1.1 Two Time Series

Given three time series, we plot them jointly.

First, we construct a dataframe.

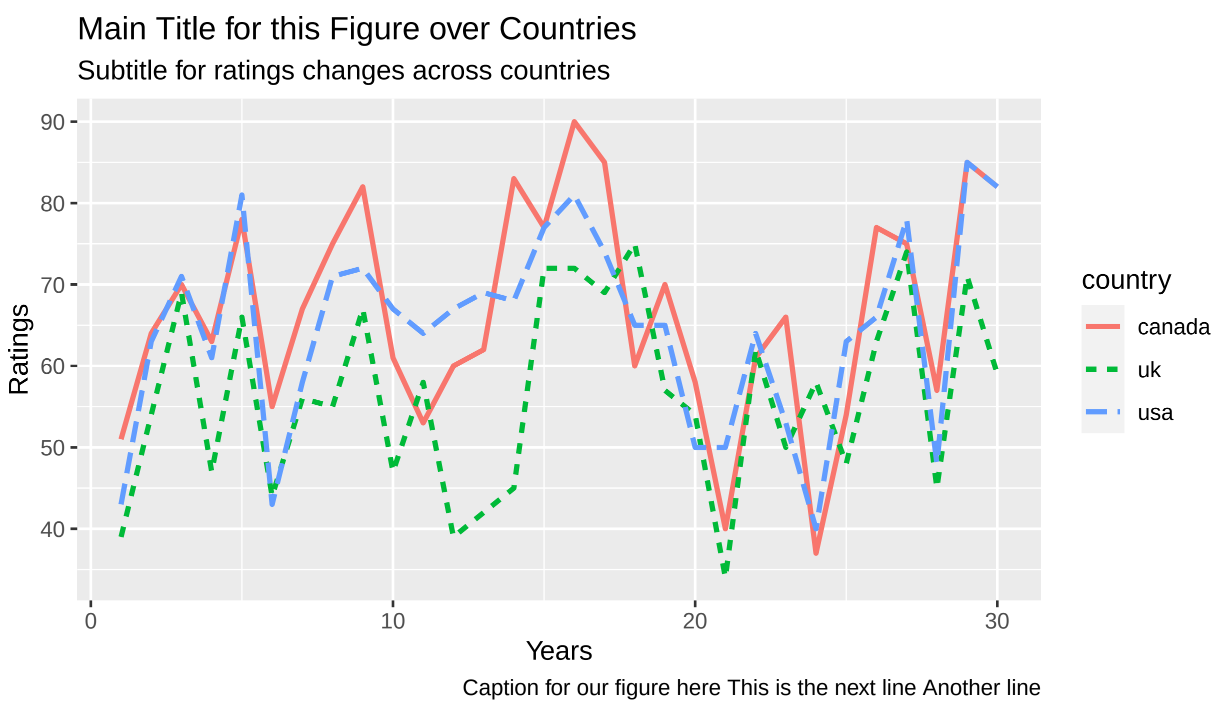
```
# Load data, and treat index as "year"
# pretend data to be country-data
df_attitude <- as_tibble(attitude) %>%
  rowid_to_column(var = "year") %>%
  select(year, rating, complaints, learning) %>%
  rename(stats_usa = rating,
         stats_canada = complaints,
         stats_uk = learning)

# Wide to Long
df_attitude <- df_attitude %>%
  pivot_longer(cols = starts_with('stats_'),
               names_to = c('country'),
               names_pattern = paste0("stats_(.*)"),
               values_to = "rating")

# Print
kable(df_attitude[1:10,]) %>% kable_styling_fc()
```

Second, we generate a basic visualizations with default values.

```
# basic chart with two lines
pl_lines_basic <- df_attitude %>%
  ggplot(aes(x=year, y=rating,
```

year	country	rating
1	usa	43
1	canada	51
1	uk	39
2	usa	63
2	canada	64
2	uk	54
3	usa	71
3	canada	70
3	uk	69
4	usa	61

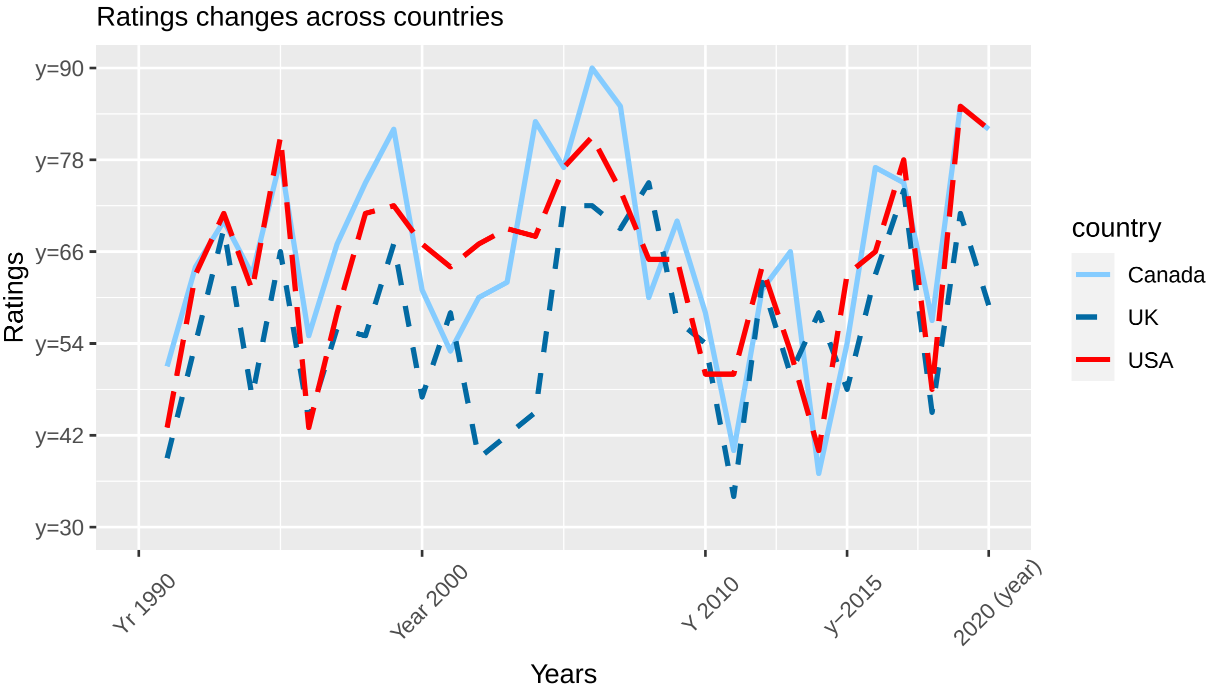
```
    color=country, linetype=country)) +
  geom_line(size = 1) +
  labs(x = paste0("Years"),
       y = paste0("Ratings"),
       title = paste(
        "Main Title for this Figure over",
        "Countries", sep=" "),
       subtitle = paste(
        "Subtitle for ratings changes across",
        "countries", sep=" "),
       caption = paste(
        "Caption for our figure here ",
        "This is the next line ",
        "Another line", sep=""))

# print figure
print(pl_lines_basic)
```



Third, we generate a more customized visualization with customized: (1) colors and shapes for lines; (2) x- and y-axis limits, labels, and breaks; (3) customized legend position.

```
# basic chart with two lines
pl_lines <- df_attitude %>%
  ggplot(aes(x=year, y=rating,
    color=country, linetype=country, shape=country)) +
  geom_line(size=1)

# Titles
st_x = "Years"
st_y = "Ratings"
st_subtitle = "Ratings changes across countries"
pl_lines <- pl_lines +
  labs(
    x = st_x,
    y = st_y,
    subtitle = st_subtitle)

# Figure improvements
# set shapes and colors
ar_st_labels <- c(
  bquote("Canada"),
  bquote("UK"),
  bquote("USA"))

ar_st_colours <- c("#85ccff", "#026aa3", "red")
ar_st_linetypes <- c("solid", "dashed", "longdash")
pl_lines <- pl_lines +
  scale_colour_manual(values = ar_st_colours, labels = ar_st_labels) +
  scale_shape_discrete(labels = ar_st_labels) +
  scale_linetype_manual(values = ar_st_linetypes, labels = ar_st_labels)

# Axis
x_labels <- c("Yr 1990", "Year 2000", "Y 2010", "y-2015", "2020 (year)")
x_breaks <- c(0, 10, 20, 25, 30)
x_min <- 0
x_max <- 30

y_breaks <- seq(30, 90, length.out=6)
y_labels <- paste0('y=', y_breaks)
y_min <- 30
y_max <- 90

pl_lines <- pl_lines +
  scale_x_continuous(
    labels = x_labels, breaks = x_breaks,
    limits = c(x_min, x_max)
  ) +
  theme(axis.text.x = element_text(
    # Adjust x-label angle
    angle = 45,
    # Adjust x-label distance to x-axis (up vs down)
    hjust = 0.4,
    # Adjust x-label left vs right wwith respect ot break point
    vjust = 0.5)) +
  scale_y_continuous(
    labels = y_labels, breaks = y_breaks,
    limits = c(y_min, y_max)
  )
```

```
# print figure
print(pl_lines)
```



10.2.2 ggplot Line Advanced

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

10.2.2.1 Continuous Y and X Variables, Three Categories, One is Subplot

Visualize one continuous variable, along the x-axis, given three categorical variables, with 12 combined categories $3 \times 2 \times 2 = 12$:

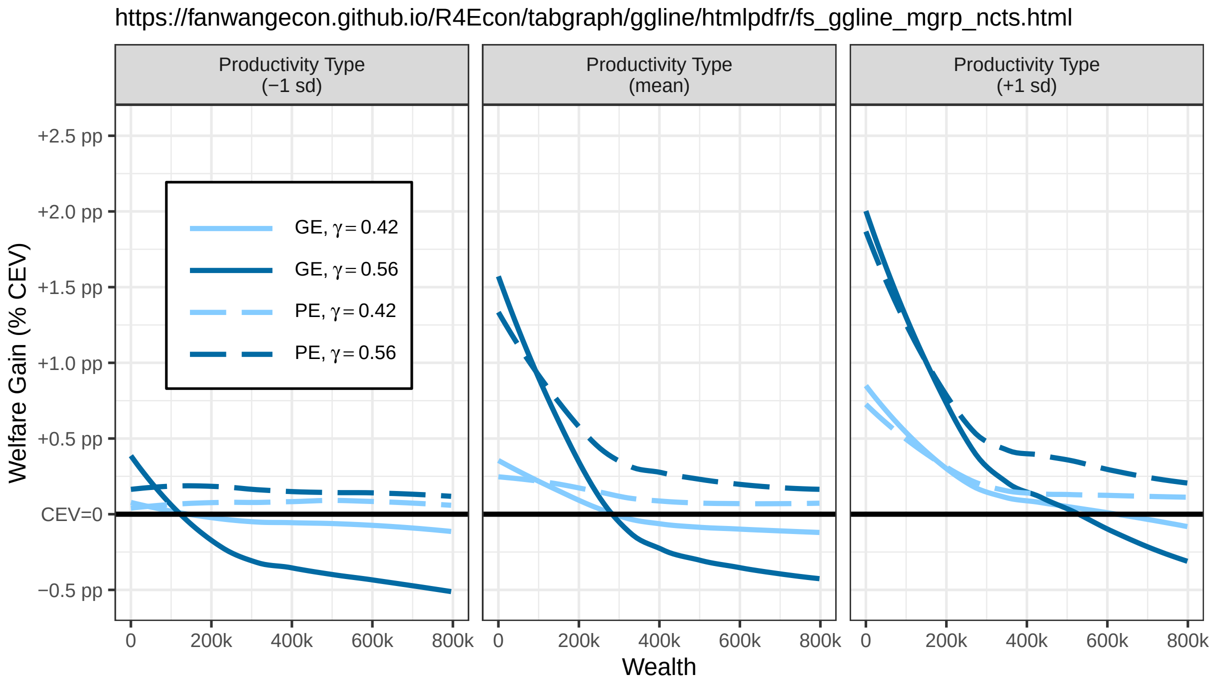
- one as subplot (productivity type), 3 unique values
- one as line-color (gamma levels), 2 unique values
- one as line-type (GE vs PE), 2 unique values

The outcome is continuous CEV, generated for results with different productivity types (subplot), generated for PE vs GE (linetype), and at different parameter specifications (lower and higher gamma). X-axis is continuous. The graphs rely on this csv file cev_data.csv.

```
# Libraries
# library(tidyverse)

# Load in CSV
bl_save_img <- TRUE
spt_csv_root <- c("C:/Users/fan/R4Econ/tabgraph/ggline/_file/")
spt_img_root <- c("C:/Users/fan/R4Econ/tabgraph/ggline/_file/")
spn_cev_data <- paste0(spt_csv_root, "cev_data.csv")
spn_cev_graph <- paste0(spt_img_root, "cev_graph.png")
spn_cev_graph_eps <- paste0(spt_img_root, "cev_graph.eps")
df_cev_graph <- as_tibble(read.csv(spn_cev_data)) %>% select(-X)

# Dataset subsetting ------

# Line Patterns and Colors ------
```

```
# ar_st_age_group_leg_labels <- c("\nGE\n\u03B3=0.42\n", "\nGE\n\u03B3=0.56\n",
#                                  "\nPE\n\u03B3=0.42\n", "\nPE\n\u03B3=0.42\n")
ar_st_age_group_leg_labels <- c(
  bquote("GE," ~ gamma == .(0.42)),
  bquote("GE," ~ gamma == .(0.56)),
  bquote("PE," ~ gamma == .(0.42)),
  bquote("PE," ~ gamma == .(0.56))
)
ar_st_colours <- c("#85ccff", "#026aa3", "#85ccff", "#026aa3")
ar_st_linetypes <- c("solid", "solid", "longdash", "longdash")

# Labels and Other Strings -------
st_title <- ""
st_x <- "Wealth"
st_y <- "Welfare Gain (% CEV)"
st_subtitle <- paste0(
  "https://fanwangecon.github.io/",
  "R4Econ/tabgraph/ggline/htmlpdfr/fs_ggline_mgrp_ncts.html"
)

# ar_st_age_group_leg_labels <- c("C\u2013Optimal", "V\u2013Optimal")

prod_type_recode <- c(
  "Productivity Type\n(-1 sd)" = "8993",
  "Productivity Type\n(mean)" = "10189",
  "Productivity Type\n(+1 sd)" = "12244"
)

x_labels <- c("0", "200k", "400k", "600k", "800k")
x_breaks <- c(
  0,
  5,
  10,
  15,
  20
)
x_min <- 0
x_max <- 20

# y_labels <- c('-0.01',
#               '\u2191\u2191\nWelfare\nGain\n\nCEV=0\n\nWelfare\nLoss\n\u2193\u2193',
#               '+0.01', '+0.02', '+0.03', '+0.04','+0.05')
y_labels <- c(
  "-0.5 pp",
  "CEV=0",
  "+0.5 pp", "+1.0 pp", "+1.5 pp", "+2.0 pp", "+2.5 pp"
)
y_breaks <- c(-0.01, 0, 0.01, 0.02, 0.03, 0.04, 0.05)
y_min <- -0.011
y_max <- 0.051

# data change -------
df_cev_graph <- df_cev_graph %>%
  filter(across(counter_policy, ~ grepl("70|42", .))) %>%
  mutate(prod_type_lvl = as.factor(prod_type_lvl)) %>%
  mutate(prod_type_lvl = fct_recode(prod_type_lvl, !!!prod_type_recode))
```

```
# graph ------
pl_cev <- df_cev_graph %>%
  group_by(prod_type_st, cash_tt) %>%
  ggplot(aes(
    x = cash_tt, y = cev_lvl,
    colour = counter_policy, linetype = counter_policy, shape = counter_policy
  )) +
  facet_wrap(~prod_type_lvl, nrow = 1) +
  geom_smooth(method = "auto", se = FALSE, fullrange = FALSE, level = 0.95)

# labels
pl_cev <- pl_cev +
  labs(
    x = st_x,
    y = st_y,
    subtitle = st_subtitle
  )

# set shapes and colors
pl_cev <- pl_cev +
  scale_colour_manual(values = ar_st_colours, labels = ar_st_age_group_leg_labels) +
  scale_shape_discrete(labels = ar_st_age_group_leg_labels) +
  scale_linetype_manual(values = ar_st_linetypes, labels = ar_st_age_group_leg_labels) +
  scale_x_continuous(
    labels = x_labels, breaks = x_breaks,
    limits = c(x_min, x_max)
  ) +
  scale_y_continuous(
    labels = y_labels, breaks = y_breaks,
    limits = c(y_min, y_max)
  )

# Horizontal line
pl_cev <- pl_cev +
  geom_hline(yintercept = 0, linetype = "solid", colour = "black", size = 1)
# geom_hline(yintercept=0, linetype='dotted', colour="black", size=2)

# theme
pl_cev <- pl_cev +
  theme_bw() +
  theme(
    text = element_text(size = 10),
    legend.title = element_blank(),
    legend.position = c(0.16, 0.65),
    legend.background = element_rect(
      fill = "white",
      colour = "black",
      linetype = "solid"
    ),
    legend.key.width = unit(1.5, "cm")
  )

# Print Images to Screen -----
print(pl_cev)
```




```
# Save Image Outputs -----
if (bl_save_img) {
  png(spn_cev_graph,
    width = 160,
    height = 105, units = "mm",
    res = 150, pointsize = 7
  )
  ggsave(
    spn_cev_graph_eps,
    plot = last_plot(),
    device = "eps",
    path = NULL,
    scale = 1,
    width = 200,
    height = 100,
    units = c("mm"),
    dpi = 150,
    limitsize = TRUE
  )
  print(pl_cev)
  dev.off()
}
```

```
## pdf
##   2
```

10.2.2.2 Continuous Y and X Variables, Two Categories, One is Subplot

In contrast to the first line plot, in this second example, we use both varying line color as well as line shape and scatter type to distinguish categories of one categorical variable. Visualize one continuous variable, along the x-axis, given three categorical variables, with 10 combined categories $2 \times 5 = 10$:

- one as subplot (GE vs PE), 2 unique values
- one with line-color, line-color and scatter shape joint variation (counterfactual type), 5 unique values

The outcome is change in male and female labor participation gaps, generated under partial and general equilibrium (subplot), generated for different counterfactual policies (linetype). X-axis is calendar year.

Features:

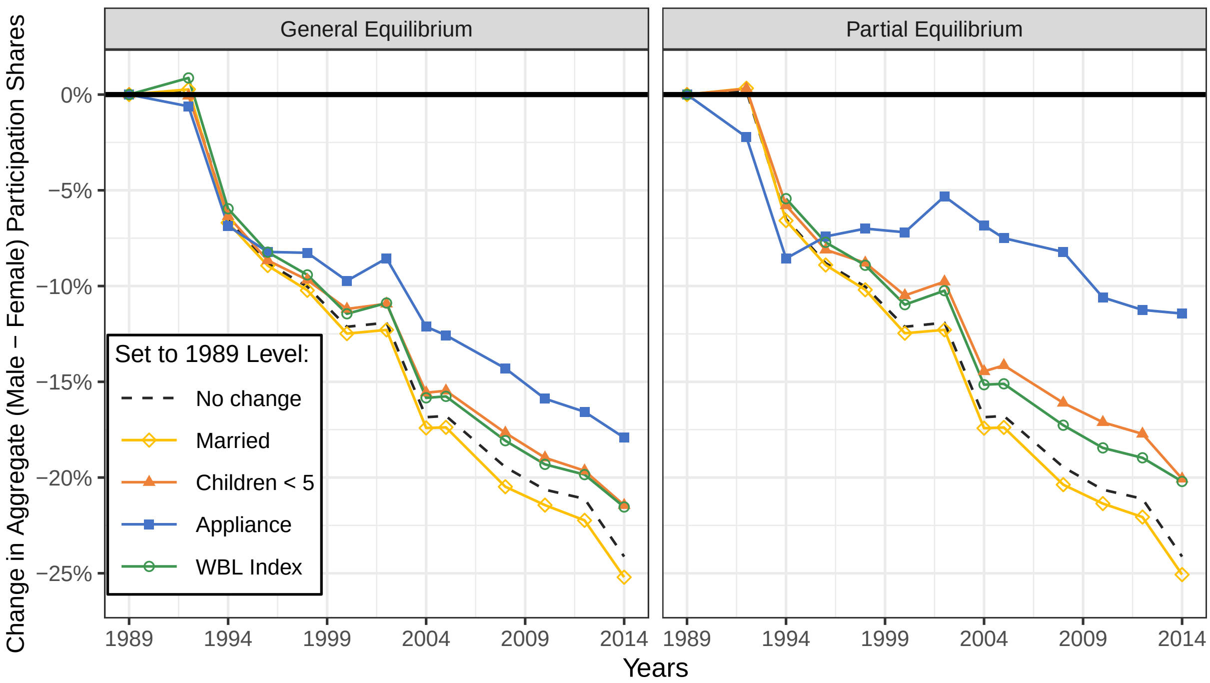
- Calendar year as x-axis
- Line + scatter with varying line patterns and scatter shapes
- Scatter shapes
- Show five lines together, with 2 lines stand out more, and 4 lines overall different than 1
- Legend box area with longer legend text, transparent and no border

For data processing, converts all possible numerical variables to numeric.

```
# Load in CSV
bl_save_img <- TRUE
spt_csv_root <- c("C:/Users/fan/R4Econ/tabgraph/ggline/_file/")
spt_img_root <- spt_csv_root
spn_flfp_sklocc_data <- paste0(spt_csv_root, "flfp_data.csv")
spn_flfp_sklocc_graph <- paste0(spt_img_root, "flfp_sam2fshr_graph.png")
spn_flfp_sklocc_graph_eps <- paste0(spt_img_root, "flfp_sam2fshr_graph.eps")

# Load data
# Convert all convertable numeric columns from string to numeric
# https://stackoverflow.com/a/49054046/8280804
is_all_numeric <- function(x) {
  !any(is.na(suppressWarnings(as.numeric(na.omit(x))))) & is.character(x)
}
df_flfp <- as_tibble(read.csv(spn_flfp_sklocc_data)) %>%
  mutate_if(is_all_numeric, as.numeric) %>%
  filter(year <= 2014)

# Dataset subsetting ------

# Line Patterns and Colors ------
ctr_var_recode <- c(
  "No change" = "1",
  "Married" = "31",
  "Children < 5" = "32",
  "Appliance" = "33",
  "WBL Index" = "34"
)

# https://www.rgbtohex.net/
ar_st_colours <- c("#262626", "#FFC001", "#ED8137", "#4472C4", "#3E9651")
# http://www.sthda.com/english/wiki/ggplot2-line-types-how-to-change-line-types-of-a-graph-in-r-soft
ar_st_linetypes <- c("dashed", "solid", "solid", "solid", "solid")
# http://sape.inf.usi.ch/quick-reference/ggplot2/shape
# 32 is invisible shape
ar_it_shapes <- c(32, 5, 17, 15, 1)

# Labels and Other Strings -------
st_title <- ""
st_x <- "Years"
st_y <- "Change in Aggregate (Male - Female) Participation Shares"
st_subtitle <- paste0(
  "https://fanwangecon.github.io/",
  "R4Econ/tabgraph/ggline/htmlpdf/fs_ggline_mgrp_ncts.html"
)

# ge_pe_recode <- c(
#   "General Equilibrium\n(Adjust Wages)" = "GE",
#   "Partial Equilibrium\n(Wage as Observed)" = "PE"
# )
```

```
ge_pe_recode <- c(
  "General Equilibrium" = "GE",
  "Partial Equilibrium" = "PE"
)
# x.breaks <- c(1989, seq(1992, 2004, by = 2), 2005, seq(2008, 2014, by = 2))
# x.labels <- paste(x.breaks[1:13])
x.breaks <- seq(1989, 2014, by = 5)
x.labels <- paste(x.breaks[1:6])
x.min <- 1989
x.max <- 2014

y.breaks <- round(seq(-0.30, 0.05, by = 0.05), 2)
y.labels <- paste0(paste(y.breaks[1:length(y.breaks)] * 100), "%")

y.min <- -0.26
y.max <- 0.01

# data change -------
df_flfp_sklocc_graph <- df_flfp %>%
  filter(ctr_var_idx %in% c(1, 31, 32, 33, 34) & category == "C001") %>%
  mutate(
    ge_pe = as.factor(ge_pe),
    ctr_var_idx = as.factor(ctr_var_idx)
  ) %>%
  mutate(ge_pe = fct_recode(ge_pe, !!!ge_pe_recode)) %>%
  mutate(ctr_var_idx = fct_recode(ctr_var_idx, !!!ctr_var_recode)) %>%
  select(year, ctr_var_idx, ge_pe, part_yeargender_shr_m2f_dfv1st)

# graph ------
pl_flfp_agg <- df_flfp_sklocc_graph %>%
  ggplot(aes(
    x = year, y = part_yeargender_shr_m2f_dfv1st,
    colour = ctr_var_idx, linetype = ctr_var_idx, shape = ctr_var_idx
  )) +
  facet_wrap(~ge_pe, nrow = 1) +
  geom_line() +
  geom_point()

# labels
pl_flfp_agg <- pl_flfp_agg +
  labs(
    x = st_x,
    y = st_y
  )
# subtitle = st_subtitle

# set shapes and colors
# scale_colour_manual(values = ar_st_colours, labels = ctr_var_recode) +
# scale_shape_manual(values=ar_it_shapes, labels = ctr_var_recode) +
# scale_linetype_manual(values = ar_st_linetypes, labels = ctr_var_recode) +
pl_flfp_agg <- pl_flfp_agg +
  scale_colour_manual(values = ar_st_colours) +
  scale_shape_manual(values = ar_it_shapes) +
  scale_linetype_manual(values = ar_st_linetypes) +
  scale_x_continuous(
    labels = x.labels, breaks = x.breaks,
    limits = c(x.min, x.max)
```

```
  ) +
  scale_y_continuous(
    labels = y.labels, breaks = y.breaks,
    limits = c(y.min, y.max)
  )

# Horizontal line
pl_flfp_agg <- pl_flfp_agg +
  geom_hline(yintercept = 0, linetype = "solid", colour = "black", size = 1)
# geom_hline(yintercept=0, linetype='dotted', colour="black", size=2)

# theme
pl_flfp_agg <- pl_flfp_agg +
  theme_bw() +
  theme(
    text = element_text(size = 11),
    legend.title = element_text(size = 10),
    legend.margin = margin(c(0.1, 0.1, 0.1, 0.1), unit = "cm"),
    legend.position = c(0.10, 0.27),
    legend.background = element_rect(
      fill = "white",
      colour = "black",
      linetype = "solid"
    ),
    legend.key.width = unit(1.0, "cm"),
    axis.title.y = element_text(size = 10)
  ) +
  guides(
    color = guide_legend(title = "Set to 1989 Level:"),
    linetype = guide_legend(title = "Set to 1989 Level:"),
    shape = guide_legend(title = "Set to 1989 Level:")
  )

# Print Images to Screen
print(pl_flfp_agg)
```



```
# Save Image Outputs -----
if (bl_save_img) {
  png(spn_flfp_sklocc_graph,
    width = 200,
    height = 100, units = "mm",
    res = 150, pointsize = 7
  )
  ggsave(
    spn_flfp_sklocc_graph_eps,
    plot = last_plot(),
    device = "eps",
    path = NULL,
    scale = 1,
    width = 200,
    height = 100,
    units = c("mm"),
    dpi = 150,
    limitsize = TRUE
  )
  print(pl_flfp_agg)
  dev.off()
}
```

```
## pdf
##   2
```

10.2.2.3 Continuous Y and X Variables, Four Categories, Three for Subplot

In contrast to the line plot above, in this third example, we have three categorical variables that will be visualized via plots and subplots. We have four categoricalv variables overall, for the fourth categorical variable, as in the second example, we continue to use both varying line color as well as line shape and scatter type to distinguish categories of this fourth categorical variable. Visualize one continuous variable, along the x-axis, given four categorical variables, with 60 combined categories $3 \times 2 \times 2 \times 3 = 36$:

- one as plot, generate three different plots, 3 unique values, achieved by saving a function and running the function three times with variable conditioning.
- one as *facet_grid* row group, 2 unique values.
- one as *facet_grid* column group, 2 unique values.
- one with line-color, line-color and scatter shape joint variation (counterfactual type), 3 unique values

Following the example above, continue to analyze female labor participation. Generated under partial and general equilibrium (subplot), and skill and occupational groups. , generated for different counterfactual policies (linetype). X-axis is calendar year.

Features:

- *facet_grid*: Multiple rows and columns for faceting, row and column labels
- No spacing for empty title line
- Graph as function with simple variable and parameter adjustment
- No minor grid
- Do not show ylabel.

First define the graphing function:

```
# The graphing function with limited parameter options.
ff_grhlfp_gepeedu_byocc <-
  function(bl_save_img = TRUE,
           st_occ = "Manual",
           y_breaks = round(seq(0.08, 0.18, by = 0.02), 2),
           y_min = 0.08,
```

```
        y_max = 0.19,
        ar_leg_position = c(0.29, 0.50),
        it_width = 160, it_height = 105,
        st_subtitle = paste0(
          "https://fanwangecon.github.io/",
          "R4Econ/tabgraph/ggline/htmlpdfr/fs_ggline_mgrp_ncts.html"
        )) {

  # Load in CSV
  spt_csv_root <- c("C:/Users/fan/R4Econ/tabgraph/ggline/_file/")
  spt_img_root <- spt_csv_root
  spn_flfp_sklocc_data <- paste0(spt_csv_root, "flfp_data.csv")
  spn_flfp_sklocc_graph <- paste0(
    spt_img_root,
    paste0("flfp_gepe_colhigh_", tolower(st_occ), "_graph.png")
  )
  spn_flfp_sklocc_graph_eps <- paste0(
    spt_img_root,
    paste0("flfp_gepe_colhigh_", tolower(st_occ), "_graph.eps")
  )

  # Load data
  # Convert all convertable numeric columns from string to numeric
  # https://stackoverflow.com/a/49054046/8280804
  is_all_numeric <- function(x) {
    !any(is.na(suppressWarnings(as.numeric(na.omit(x))))) & is.character(x)
  }
  df_flfp <- as_tibble(read.csv(spn_flfp_sklocc_data)) %>%
    mutate_if(is_all_numeric, as.numeric) %>%
    filter(year <= 2014)

  # Dataset subsetting ------

  # Line Patterns and Colors ------
  ctr_var_recode <- c(
    "Prediction no Counterfactual" = "1",
    "Married at 1989 Levels" = "31",
    "Children < 5 at 1989 Levels" = "32",
    "Appliance at 1989 Levels" = "33",
    "WBL Index at 1989 Levels" = "34"
  )

  # https://www.rgbtohex.net/
  # ar_st_colours <- c("#262626", "#FFC001", "#ED8137", "#4472C4", "#3E9651")
  ar_st_colours <- c("#262626", "#ED8137", "#4472C4")
  # http://www.sthda.com/english/wiki/ggplot2-line-types-how-to-change-line-types-of-a-graph-in-r-
  ar_st_linetypes <- c("dashed", "solid", "solid")
  # http://sape.inf.usi.ch/quick-reference/ggplot2/shape
  # 32 is invisible shape
  # ar_it_shapes <- c(32, 5, 17, 15, 1)
  ar_it_shapes <- c(32, 17, 15)

  # Labels and Other Strings -------
  st_x <- "Years"
  st_y <- paste0("Female ", st_occ, " Occupation Participation Shares")

  # ge_pe_recode <- c(
  #   "General Equilibrium\n(Adjust Wages)" = "GE",
```

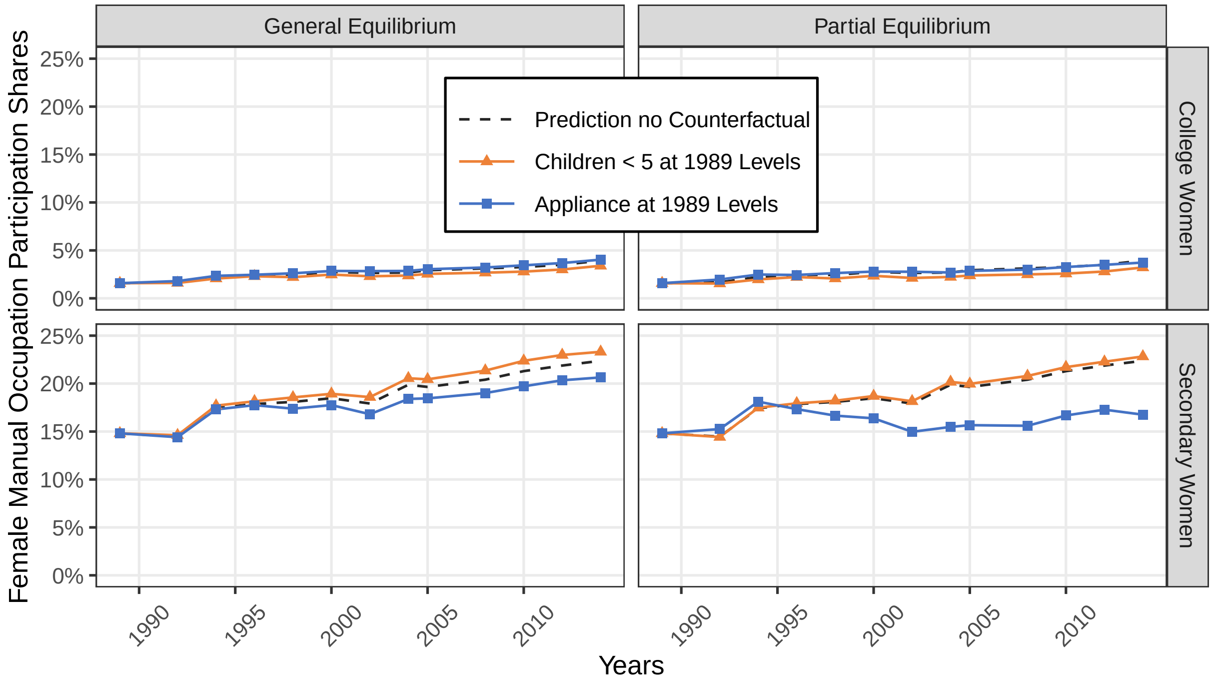
```
#   "Partial Equilibrium\n(Wage as Observed)" = "PE"
# )
ge_pe_recode <- c(
  "General Equilibrium" = "GE",
  "Partial Equilibrium" = "PE"
)
# ge_pe_recode <- c(
#   "GE" = "GE",
#   "PE" = "PE"
# )

skilled_unskilled_recode <- c(
  "College Women" = "skilled",
  "Secondary Women" = "unskilled"
)

# x_breaks <- seq(1989, 2014, by = 5)
x_breaks <- c(1990, 1995, 2000, 2005, 2010)
x_labels <- paste(x_breaks[1:length(x_breaks)])
x_min <- 1989
x_max <- 2014

# y_breaks <- round(seq(0.08, 0.18, by = 0.02), 2)
y_labels <- paste0(paste(y_breaks[1:length(y_breaks)] * 100), "%")

# y_min <- 0.08
# y_max <- 0.19

# data change -------
df_flfp_sklocc_graph <- df_flfp %>%
  filter(ctr_var_idx %in% c(1, 32, 33) &
    gender == "Female" &
    occupation %in% c(st_occ)) %>%
  mutate(
    ge_pe = as.factor(ge_pe),
    ctr_var_idx = as.factor(ctr_var_idx)
  ) %>%
  mutate(
    ge_pe = fct_recode(ge_pe, !!!ge_pe_recode),
    skill = fct_recode(skill, !!!skilled_unskilled_recode),
    ctr_var_idx = fct_recode(ctr_var_idx, !!!ctr_var_recode)
  ) %>%
  select(year, skill, occupation, ctr_var_idx, ge_pe, genskl_part_share)

# graph ------
pl_flfp_sklocc <- df_flfp_sklocc_graph %>%
  ggplot(aes(
    x = year, y = genskl_part_share,
    colour = ctr_var_idx, linetype = ctr_var_idx, shape = ctr_var_idx
  )) +
  facet_grid(skill ~ ge_pe) +
  geom_line() +
  geom_point()

# labels
if (st_subtitle == "") {
  pl_flfp_sklocc <- pl_flfp_sklocc +
```

```
      labs(
        x = st_x,
        y = st_y
      )
  } else {
    pl_flfp_sklocc <- pl_flfp_sklocc +
      labs(
        x = st_x,
        y = st_y,
        subtitle = st_subtitle
      )
  }

  # set shapes and colors
  pl_flfp_sklocc <- pl_flfp_sklocc +
    scale_colour_manual(values = ar_st_colours) +
    scale_shape_manual(values = ar_it_shapes) +
    scale_linetype_manual(values = ar_st_linetypes) +
    scale_x_continuous(
      labels = x_labels, breaks = x_breaks,
      limits = c(x_min, x_max)
    ) +
    scale_y_continuous(
      labels = y_labels, breaks = y_breaks,
      limits = c(y_min, y_max)
    )

  # theme
  pl_flfp_sklocc <- pl_flfp_sklocc +
    theme_bw() +
    theme(
      text = element_text(size = 11),
      panel.grid.minor = element_blank(),
      legend.title = element_blank(),
      legend.position = ar_leg_position,
      legend.margin = margin(c(0.1, 0.1, 0.1, 0.1), unit = "cm"),
      legend.background = element_rect(
        fill = "white",
        colour = "black",
        linetype = "solid"
      ),
      legend.key.width = unit(1.0, "cm"),
      axis.text.x = element_text(angle = 45, vjust = 0.1, hjust = 0.1)
      # axis.text.y = element_text(angle = 90, hjust = 0.4)
    )
  # element_text(angle = 90, hjust = 0.4)
  # axis.title.y = element_blank(), # no y-label

  # Save Image Outputs -----
  if (bl_save_img) {
    ggsave(
      spn_flfp_sklocc_graph,
      plot = last_plot(),
      device = "png",
      path = NULL,
      scale = 1,
      width = it_width,
```

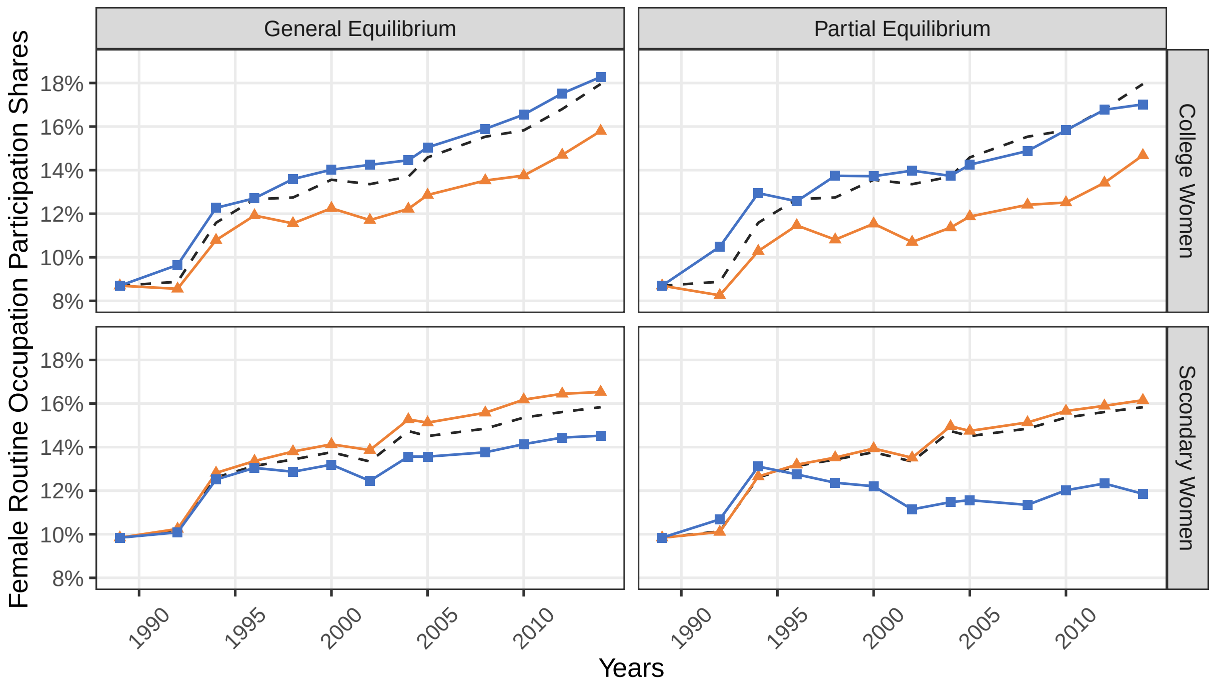
```
        height = it_height,
        units = c("mm"),
        dpi = 150,
        limitsize = TRUE
      )
      ggsave(
        spn_flfp_sklocc_graph_eps,
        plot = last_plot(),
        device = "eps",
        path = NULL,
        scale = 1,
        width = it_width,
        height = it_height,
        units = c("mm"),
        dpi = 150,
        limitsize = TRUE
      )
       # dev.off()
    }

    return(pl_flfp_sklocc)
  }
```

Second, run the function, for Manual, Routine and Analytical Works Separately.

```
it_width <- 100
it_height <- 100
st_subtitle <- paste0(
  "https://fanwangecon.github.io/",
  "R4Econ/tabgraph/ggline/htmlpdfr/fs_ggline_mgrp_ncts.html"
)
st_subtitle <- ""
# Manual,
pl_flfp_sklocc_manual <- ff_grhlfp_gepeedu_byocc(
  bl_save_img = TRUE,
  st_occ = "Manual",
  y_breaks = round(seq(0.00, 0.25, by = 0.05), 2),
  y_min = 0.00, y_max = 0.25,
  ar_leg_position = c(0.50, 0.80),
  it_width = it_width, it_height = it_height,
  st_subtitle = st_subtitle
)
print(pl_flfp_sklocc_manual)
```

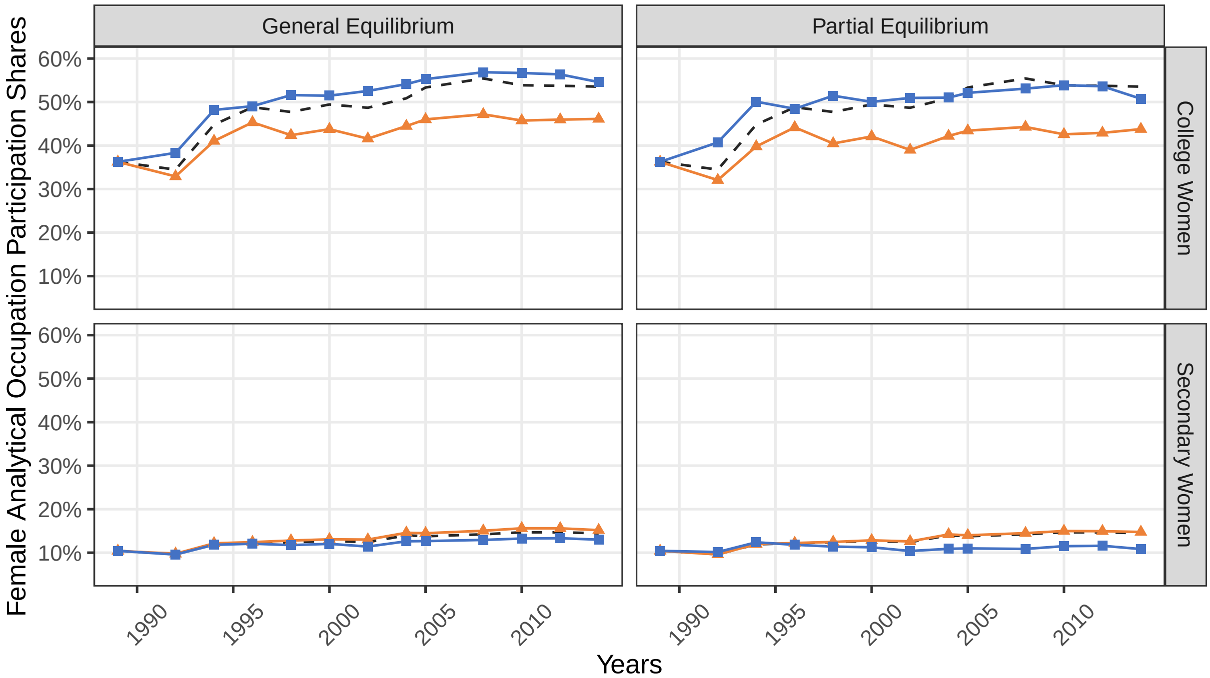



```
# Routine
pl_flfp_sklocc_routine <- ff_grhlfp_gepeedu_byocc(
  bl_save_img = TRUE,
  st_occ = "Routine",
  y_breaks = round(seq(0.08, 0.18, by = 0.02), 2),
  y_min = 0.08, y_max = 0.19,
  ar_leg_position = "none",
  it_width = it_width, it_height = it_height,
  st_subtitle = st_subtitle
)
print(pl_flfp_sklocc_routine)
```




```
# Analytical
pl_flfp_sklocc_analytical <- ff_grhlfp_gepeedu_byocc(
```

```
  bl_save_img = TRUE,
  st_occ = "Analytical",
  y_breaks = round(seq(0.10, 0.60, by = 0.10), 2),
  y_min = 0.05, y_max = 0.60,
  ar_leg_position = "none",
  it_width = it_width, it_height = it_height,
  st_subtitle = st_subtitle
)
print(pl_flfp_sklocc_analytical)
```



10.2.3 Time-series Plots with Shaded Areas

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

10.2.3.1 Single Time-series with Single Type of Shade

We will construct three country-specific fake GDP time-series (converted from the attitude dataset). Then we will randomly select a subset of months and shade these months. This will generate a "recession" plot, where recession periods are shaded.

One of the assumption will be that we have data at discrete intervals, and the shaded areas will take mid-points.

First, we repeat the basic time-series data construction found in R4Econ.fs_ggline_basic.

```
# Load data, and treat index as "year", pretend data to be country-data
df_gdp <- as_tibble(attitude) %>%
    rowid_to_column(var = "year") %>%
    select(year, rating, complaints, learning) %>%
    rename(stats_usa = rating, stats_uk = learning) %>%
    pivot_longer(
        cols = starts_with("stats_"),
        names_to = c("country"),
        names_pattern = paste0("stats_(.*)"),
        values_to = "gdp"
    )
```

year	complaints	country	gdp
1	51	usa	43
1	51	uk	39
2	64	usa	63
2	64	uk	54
3	70	usa	71
3	70	uk	69
4	63	usa	61
4	63	uk	47
5	78	usa	81
5	78	uk	66

```
# Print
kable(df_gdp[1:10, ]) %>% kable_styling_fc()
```

Second, we select a subset of period to shade. We generate a random subset of non-overlapping consecutive numbers following what is outlined in R4Econ.fs_ary_generate.

```
# Number of random starting index
it_start_idx <- min(df_gdp$year)
it_end_idx <- max(df_gdp$year)
it_startdraws_max <- 6
it_duramax <- 2
it_backward_win <- 0.3
it_forward_win <- 0.3

# Random seed
set.seed(1234)
# Draw random index between min and max
ar_it_start_idx <- sort(sample(
    seq(it_start_idx, it_end_idx),
    it_startdraws_max,
    replace = FALSE
))
# Draw random durations, replace = TRUE because can repeat
ar_it_duration <- sample(it_duramax, it_startdraws_max, replace = TRUE)

# Check space between starts
ar_it_startgap <- diff(ar_it_start_idx)
ar_it_dura_lenm1 <- ar_it_duration[1:(length(ar_it_duration) - 1)]
# Adjust durations
ar_it_dura_bd <- pmin(ar_it_startgap - 2, ar_it_dura_lenm1)
ar_it_duration[1:(length(ar_it_duration) - 1)] <- ar_it_dura_bd

# Drop consecutive starts
ar_bl_dura_nonneg <- which(ar_it_duration >= 0)
ar_it_start_idx <- ar_it_start_idx[ar_bl_dura_nonneg]
ar_it_duration <- ar_it_duration[ar_bl_dura_nonneg]

# Print
print(glue::glue(
    "random starts + duration: ",
    "{ar_it_start_idx} + {ar_it_duration}"
))
```

```
## random starts + duration: 5 + 1
## random starts + duration: 12 + 1
```

```
## random starts + duration: 16 + 1
## random starts + duration: 22 + 2
## random starts + duration: 26 + 0
## random starts + duration: 28 + 2
```

Third, convert integers to half-point mid-distance, unless exceed lower or upper bounds, and build start and end points. We also construct back and forward window around

```
# Offset by half of an integer
ar_fl_start_time <- ar_it_start_idx - 0.5
ar_fl_end_time <- ar_it_start_idx + ar_it_duration + 0.5

# Bound by min and max
ar_fl_end_time <- pmin(ar_fl_end_time, it_end_idx)
ar_fl_start_time <- pmax(ar_fl_start_time, it_start_idx)

# Backward window
ar_fl_end_time_win_backward <- ar_fl_start_time
ar_fl_start_time_win_backward <- pmax(
    ar_fl_start_time - it_backward_win, it_start_idx
)

# Forward window
ar_fl_end_time_win_forward <- pmin(
    ar_fl_end_time + it_forward_win, it_end_idx
)
ar_fl_start_time_win_forward <- ar_fl_end_time

# Print
print(glue::glue(
    "random start-time vs end-time: ",
    "{ar_fl_start_time} + {ar_fl_end_time}"
))
```

```
## random start-time vs end-time: 4.5 + 6.5
## random start-time vs end-time: 11.5 + 13.5
## random start-time vs end-time: 15.5 + 17.5
## random start-time vs end-time: 21.5 + 24.5
## random start-time vs end-time: 25.5 + 26.5
## random start-time vs end-time: 27.5 + 30
```

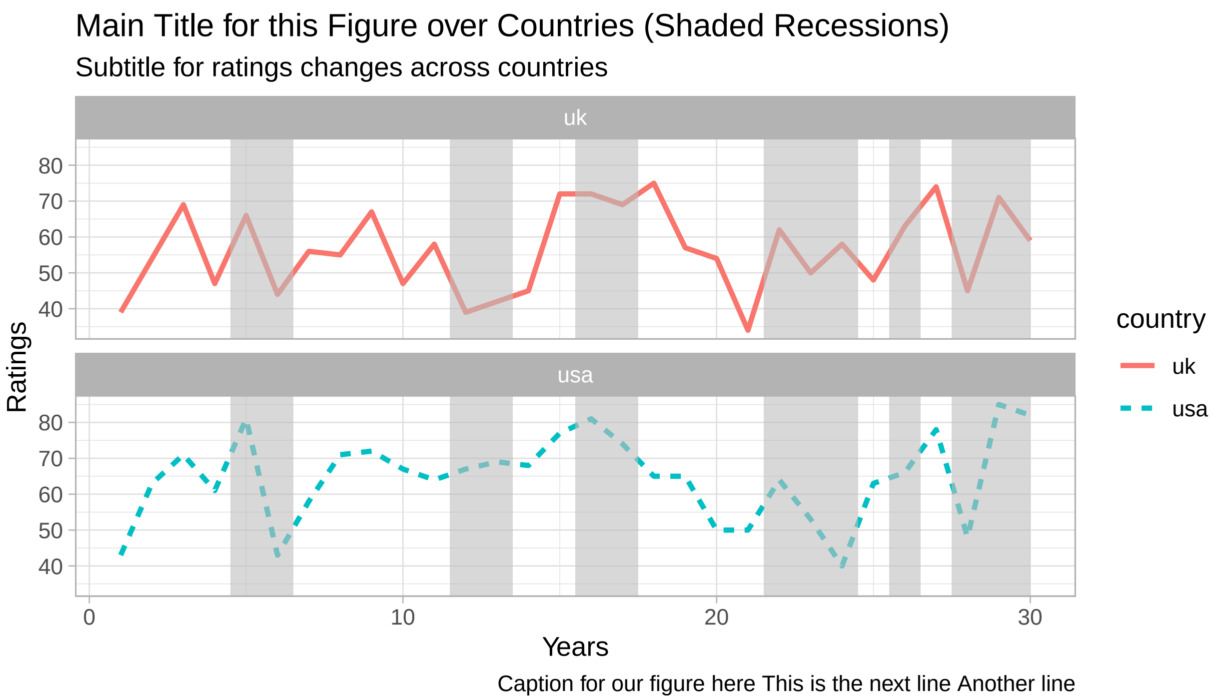
Fourth, we construct a dataframe with variables as start and end of each non-overlapping recessions. We have a main window, and a lower and upper window bounds as well.

```
# Variable names
# w1 = backward, w2 = main, w3 = forward, use w1, w2, w3 to facilate legend fill sorting
ar_st_varnames <- c(
    "recess_id",
    "year_start_w2", "year_end_w2",
    "year_start_w1", "year_end_w1",
    "year_start_w3", "year_end_w3"
)

# Recession index file
df_recession <- as_tibble(cbind(
    ar_fl_start_time, ar_fl_end_time,
    ar_fl_start_time_win_backward, ar_fl_end_time_win_backward,
    ar_fl_start_time_win_forward, ar_fl_end_time_win_forward
)) %>%
    rowid_to_column() %>%
```

```r
    rename_all(~ c(ar_st_varnames))

# Reshape from Wide to Long
df_recession <- df_recession %>%
    pivot_longer(
        cols = starts_with("year"),
        names_to = c("time", "window"),
        names_pattern = "year_(.*)_(.*)",
        values_to = "year"
    ) %>%
    pivot_wider(
        id_cols = c("recess_id", "window"),
        names_from = time,
        names_prefix = "year_",
        values_from = year
    )
```

Fifth, visualize time-series with shaded areas for “recessions”. Note that we are considering here “recessions” that are not country-specific.

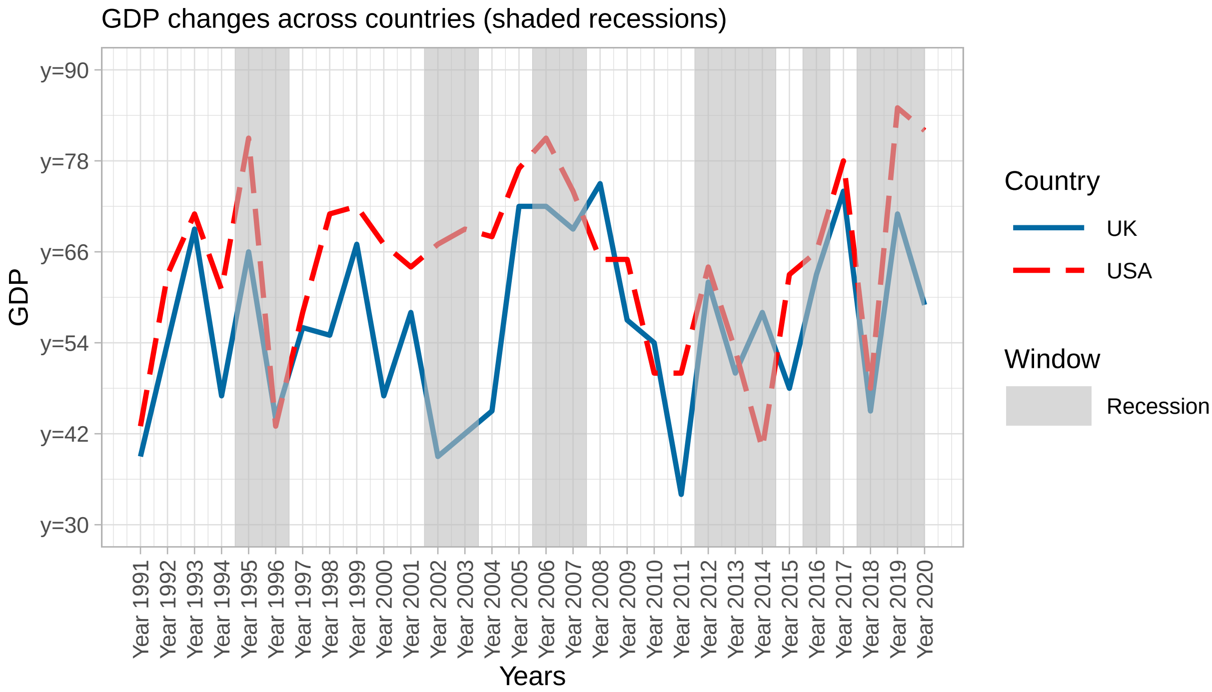
```r
# basic chart with two lines
pl_lines_basic <- ggplot() +
    geom_line(data = df_gdp, aes(
        x = year, y = gdp,
        color = country, linetype = country
    ), size = 1) +
    geom_rect(data = df_recession %>%
        filter(window == "w2"), aes(
        xmin = year_start, xmax = year_end,
        ymin = -Inf, ymax = Inf
    ), alpha = 0.6, fill = "gray") +
    labs(
        x = paste0("Years"),
        y = paste0("Ratings"),
        title = paste(
            "Main Title for this Figure over Countries (Shaded Recessions)",
            sep = " "
        ),
        subtitle = paste(
            "Subtitle for ratings changes across",
            "countries",
            sep = " "
        ),
        caption = paste(
            "Caption for our figure here ",
            "This is the next line ",
            "Another line",
            sep = ""
        )
    ) +
    theme_light() +
    facet_wrap(~country, ncol = 1)

# print figure
print(pl_lines_basic)
```



Sixth, we generate a more customized visualization with customized: (1) colors and shapes for lines as well as for windows; (2) x- and y-axis limits, labels, and breaks; (3) customized legend position.

```
# Window color
st_win_leg_title <- "Window"
st_win_color <- "gray"
st_win_label <- "Recession"
# basic chart with two lines
pl_lines <- ggplot() +
    geom_line(data = df_gdp, aes(
        x = year, y = gdp,
        color = country, linetype = country
    ), size = 1) +
    geom_rect(data = df_recession %>%
        filter(window == "w2"), aes(
        xmin = year_start, xmax = year_end,
        ymin = -Inf, ymax = Inf,
        fill = st_win_color
    ), alpha = 0.6) +
    theme_light()

# Titles
st_x <- "Years"
st_y <- "GDP"
st_subtitle <- "GDP changes across countries (shaded recessions)"
pl_lines <- pl_lines +
    labs(
        x = st_x,
        y = st_y,
        subtitle = st_subtitle
    )

# Figure improvements
# set shapes and colors
st_line_leg_title <- "Country"
ar_st_labels <- c(
```

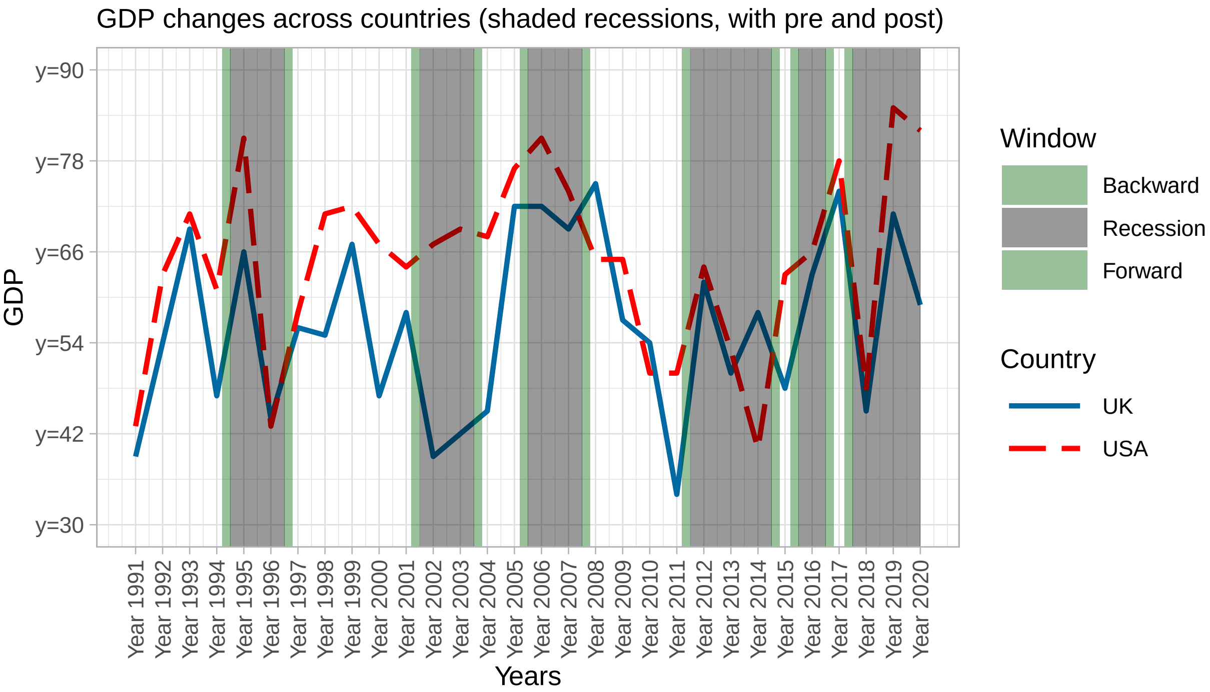
```
    bquote("UK"),
    bquote("USA")
)

ar_st_colours <- c("#026aa3", "red")
ar_st_linetypes <- c("solid", "longdash")
pl_lines <- pl_lines +
    scale_colour_manual(values = ar_st_colours, labels = ar_st_labels) +
    scale_shape_discrete(labels = ar_st_labels) +
    scale_linetype_manual(values = ar_st_linetypes, labels = ar_st_labels) +
    scale_fill_manual(values = c(st_win_color), labels = c(st_win_label)) +
    labs(
        fill = st_win_leg_title,
        colour = st_line_leg_title, linetype = st_line_leg_title
    ) +
    theme(legend.key.width = unit(2.5, "line"))

# Axis
x_breaks <- seq(1, 30, length.out = 30)
x_labels <- paste0("Year ", x_breaks + 1990)

x_min <- 1
x_max <- 30

y_breaks <- seq(30, 90, length.out = 6)
y_labels <- paste0("y=", y_breaks)
y_min <- 30
y_max <- 90

pl_lines <- pl_lines +
    scale_x_continuous(
        labels = x_labels, breaks = x_breaks,
        limits = c(x_min, x_max)
    ) +
    theme(axis.text.x = element_text(
        # Adjust x-label angle
        angle = 90,
        # Adjust x-label distance to x-axis (up vs down)
        hjust = 0.4,
        # Adjust x-label left vs right wwith respect ot break point
        vjust = 0.5
    )) +
    scale_y_continuous(
        labels = y_labels, breaks = y_breaks,
        limits = c(y_min, y_max)
    )

# print figure
print(pl_lines)
```



Finally, we generate a figure with three fill colors for the three windows, main, backward, and forward windows. We reuse various parameters used in the prior block.

```
# Window color
st_win_leg_title <- "Window"
# basic chart with two lines
pl_lines <- ggplot() +
    geom_line(data = df_gdp, aes(
        x = year, y = gdp,
        color = country, linetype = country
    ), size = 1) +
    geom_rect(data = df_recession, aes(
        xmin = year_start, xmax = year_end,
        ymin = -Inf, ymax = Inf,
        fill = window
    ), alpha = 0.4) +
    theme_light()

# Titles
st_x <- "Years"
st_y <- "GDP"
st_subtitle <- "GDP changes across countries (shaded recessions, with pre and post)"
pl_lines <- pl_lines +
    labs(
        x = st_x,
        y = st_y,
        subtitle = st_subtitle
    )

# Figure improvements
# fill label and colors
ar_st_win_color <- c("darkgreen", "black", "darkgreen")
ar_st_win_label <- c("Backward", "Recession", "Forward")
# set shapes and colors
st_line_leg_title <- "Country"
ar_st_labels <- c(
    bquote("UK"),
```

```
    bquote("USA")
)

ar_st_colours <- c("#026aa3", "red")
ar_st_linetypes <- c("solid", "longdash")
pl_lines <- pl_lines +
    scale_colour_manual(values = ar_st_colours, labels = ar_st_labels) +
    scale_shape_discrete(labels = ar_st_labels) +
    scale_linetype_manual(values = ar_st_linetypes, labels = ar_st_labels) +
    scale_fill_manual(values = c(ar_st_win_color), labels = c(ar_st_win_label)) +
    labs(
        fill = st_win_leg_title,
        colour = st_line_leg_title, linetype = st_line_leg_title
    ) +
    theme(legend.key.width = unit(2.5, "line"))

# Axis
x_breaks <- seq(1, 30, length.out = 30)
x_labels <- paste0("Year ", x_breaks + 1990)

x_min <- 1
x_max <- 30

y_breaks <- seq(30, 90, length.out = 6)
y_labels <- paste0("y=", y_breaks)
y_min <- 30
y_max <- 90

pl_lines <- pl_lines +
    scale_x_continuous(
        labels = x_labels, breaks = x_breaks,
        limits = c(x_min, x_max)
    ) +
    theme(axis.text.x = element_text(
        # Adjust x-label angle
        angle = 90,
        # Adjust x-label distance to x-axis (up vs down)
        hjust = 0.4,
        # Adjust x-label left vs right wwith respect ot break point
        vjust = 0.5
    )) +
    scale_y_continuous(
        labels = y_labels, breaks = y_breaks,
        limits = c(y_min, y_max)
    )

# print figure
print(pl_lines)
```



10.3 ggplot Scatter Related Plots

10.3.1 ggplot Scatter Plot

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

10.3.1.1 Scatter Plot with Unique Shape, Color, and Label for Each

1. y-axis: horsepower
2. x-axis: time for 1/4 Miles (QSEC)
3. filter: select to display six cars as six scattered points

First, select the relevant variables and filter.

```
# Include row-name (car-names) as a variable
tb_carnames <- rownames_to_column(mtcars, var = "car_name")  %>% as_tibble()
# Select only six observations for scatter plot
set.seed(789)
it_cars_select <- 8
tb_carnames_selected <- tb_carnames[sample(dim(tb_carnames)[1], it_cars_select, replace=FALSE), ]
# Select only car name and a few variables
tb_carnames_selected <- tb_carnames_selected %>%
  select(car_name, hp, qsec) %>%
  mutate(car_name = factor(car_name))
```

Second, add styling for each point:

```
# https://www.rgbtohex.net/
# https://fanwangecon.github.io/M4Econ/graph/tools/htmlpdfm/fs_color.html
# ar_st_colours <- c(
#   "#262626", "#922428",
#   "#6b4c9a", "#535154",
#   "#3e9651", "#396ab1",
#   "#cc2529", "#ED8137")
ar_st_colours <- c(
  "#922428", "#922428",
```

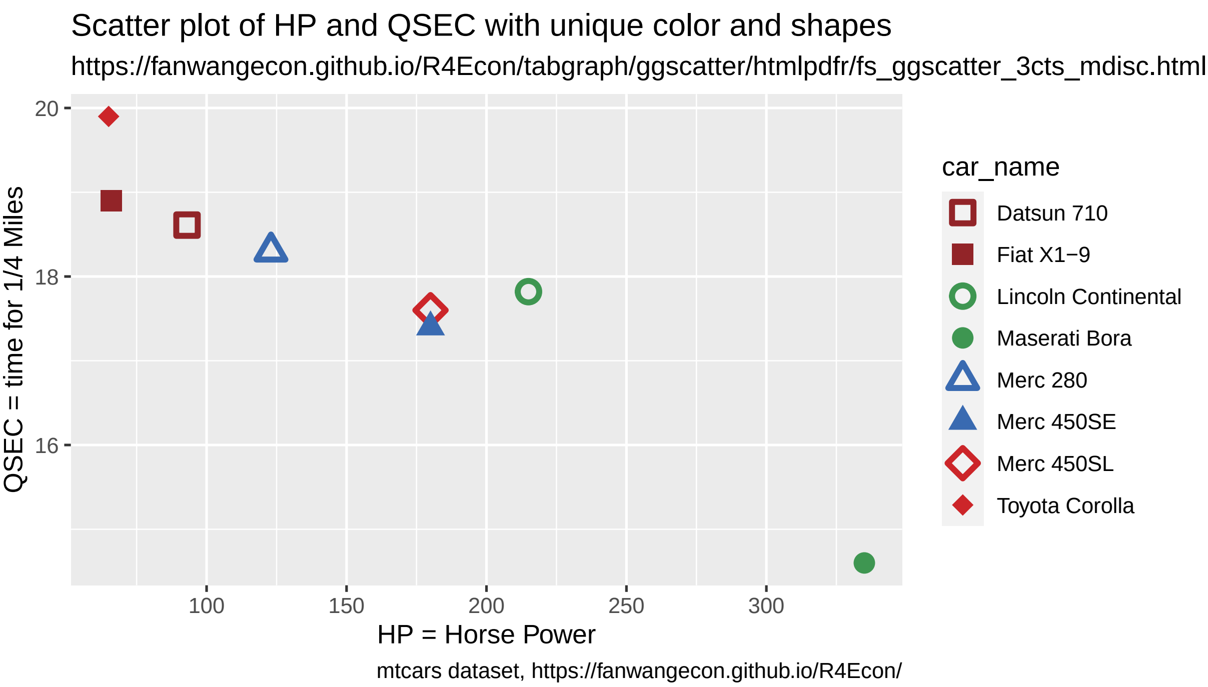
```
  "#3e9651", "#3e9651",
  "#396ab1", "#396ab1",
  "#cc2529", "#cc2529")
# http://sape.inf.usi.ch/quick-reference/ggplot2/shape
# 32 is invisible shape
# ar_it_shapes <- c(32, 5, 17, 15, 1)
ar_it_shapes <- c(
  0, 15, # square
  1, 16, # circle
  2, 17, # triangle
  5, 18 # diamond
  )
```

Third, draw a scatter plot, with defaults.

```
# Labeling
st_title <- paste0('Scatter plot of HP and QSEC with unique color and shapes')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/ggscatter/htmlpdfr/fs_ggscatter_3cts_mdisc.html')
st_caption <- paste0('mtcars dataset, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'HP = Horse Power'
st_y_label <- 'QSEC = time for 1/4 Miles'
# Graphing
plt_mtcars_scatter <- tb_carnames_selected %>%
  ggplot(aes(x=hp, y=qsec,
             colour = car_name, shape = car_name,
             label=car_name)) +
  geom_point(size=3, stroke = 1.75) +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption)
  # geom_text(color='black', size = 3.5, check_overlap = TRUE)
# Display preliminary
# print(plt_mtcars_scatter)
```

Fourth, add in color and shape for each point based on our specifications.

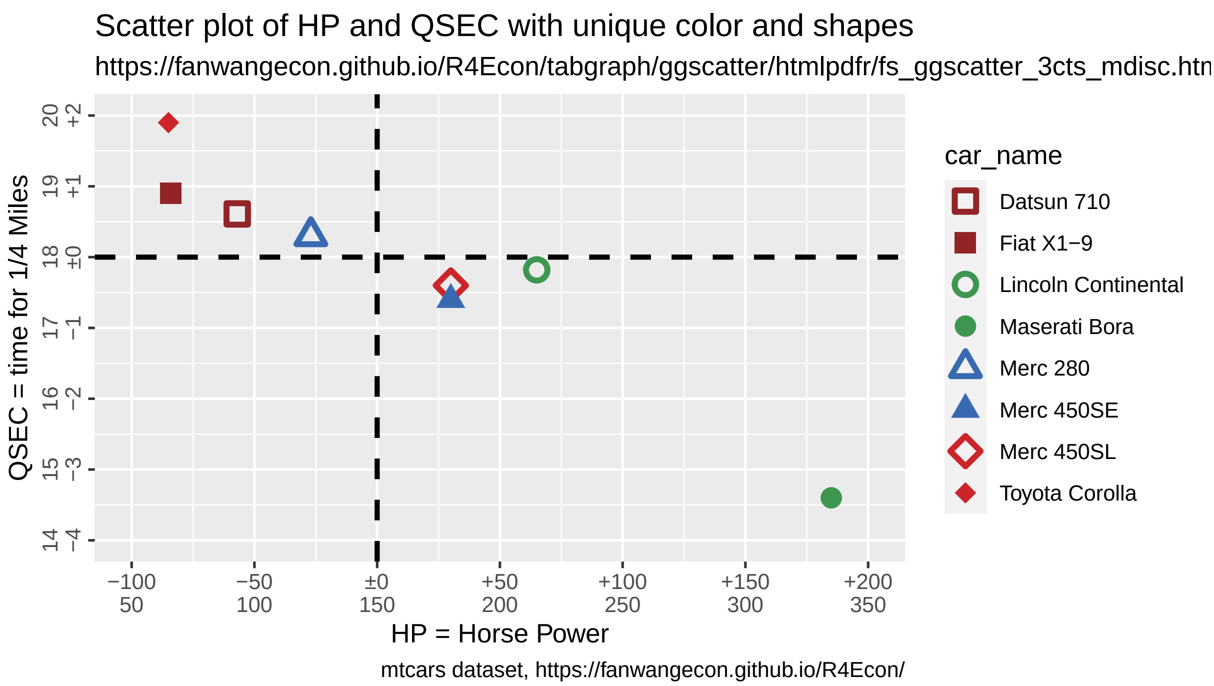
```
plt_mtcars_scatter <- plt_mtcars_scatter +
  scale_colour_manual(values = ar_st_colours) +
  scale_shape_manual(values = ar_it_shapes)
# Display preliminary
print(plt_mtcars_scatter)
```



Fifth, axis control, add-in mid-lines and additional layer of axis to show differences from added mid-lines. Add two layers of y and x labels, so that we have levels as well as deviations from the horizontal and vertical lines. Have two layers of labels so that have levels and deviations from levels.

```
# A. Y-line and X-line
fl_y_line_val <- 18
fl_x_line_val <- 150

# B. X labels
x_breaks <- c(50, 100, 150, 200, 250, 300, 350)
# x labels layer 2
x_breaks_devi <- x_breaks - fl_x_line_val
st_x_breaks_devi <- paste0(x_breaks_devi)
st_x_breaks_devi[x_breaks_devi>0] <- paste0("+", st_x_breaks_devi[x_breaks_devi>0])
st_x_breaks_devi[x_breaks_devi==0] <- paste0("±", st_x_breaks_devi[x_breaks_devi==0])
# x labels layer 1 and 2 joined
x_labels <- paste0(st_x_breaks_devi[1:length(x_breaks)], '\n', x_breaks[1:length(x_breaks)])
# x-bounds
x_min <- 50
x_max <- 350

# C. Y labels layer 1
y_breaks <- seq(14, 20, by=1)
# Y labels layer 2
y_breaks_devi <- y_breaks - fl_y_line_val
st_y_breaks_devi <- paste0(y_breaks_devi)
st_y_breaks_devi[y_breaks_devi>0] <- paste0("+", st_y_breaks_devi[y_breaks_devi>0])
st_y_breaks_devi[y_breaks_devi==0] <- paste0("±", st_y_breaks_devi[y_breaks_devi==0])
# Y labels layer 1 and 2 joined
y_labels <- paste0(y_breaks[1:length(y_breaks)], '\n', st_y_breaks_devi[1:length(y_breaks)])
# y-bounds
y_min <- 14
y_max <- 20

# D. Add custom axis
plt_mtcars_scatter <- plt_mtcars_scatter +
  geom_hline(yintercept=fl_y_line_val, linetype="dashed", color="black", size=1) +
```

```
  geom_vline(xintercept=fl_x_line_val, linetype="dashed", color="black", size=1) +
  scale_x_continuous(
    labels = x_labels, breaks = x_breaks,
    limits = c(x_min, x_max)
  ) +
  scale_y_continuous(
    labels = y_labels, breaks = y_breaks,
    limits = c(y_min, y_max)
  )

# E. Rotate Text
plt_mtcars_scatter <- plt_mtcars_scatter +
  theme(axis.text.y = element_text(angle = 90, hjust = 0.5, vjust = 0.5))

# F. print
print(plt_mtcars_scatter)
```



10.3.1.2 Three Continuous Variables and Two Categorical Variables

We will generate a graph that is very similar to the graph shown for fs_tib_factors, with the addition that scatter color and shape will be for two separate variables, and with the addition that scatter size will be for an additional continuous variable.

We have three continuous variables:

1. y-axis: time for 1/4 Miles (QSEC)
2. x-axis: horsepower
3. scatter-size: miles per gallon (mpg)

We have two categorical ariables:

1. color: vs engine shape (vshaped or straight)
2. shape: am shift type (auto or manual)

First, Load in the mtcars dataset and convert to categorical variables to factor with labels.

```
# First make sure these are factors
tb_mtcars <- as_tibble(mtcars) %>%
```

```
  mutate(vs = as_factor(vs), am = as_factor(am))
# Second Label the Factors
am_levels <- c(auto_shift = "0", manual_shift = "1")
vs_levels <- c(vshaped_engine = "0", straight_engine = "1")
tb_mtcars <- tb_mtcars %>%
  mutate(vs = fct_recode(vs, !!!vs_levels),
         am = fct_recode(am, !!!am_levels))
```

Second, generate the core graph, a scatterplot with a nonlinear trendline. Note that in the example below color and shpae only apply to the jitter scatter, but not the trendline graph.

```
# Graphing
plt_mtcars_scatter <-
  ggplot(tb_mtcars, aes(x=hp, y=qsec)) +
  geom_jitter(aes(size=mpg, colour=vs, shape=am), width = 0.15) +
  geom_smooth(span = 0.50, se=FALSE) +
  theme_bw()
```

Third, control Color and Shape Information. There will be two colors and two shapes. See all shape listing.

```
# Color controls
ar_st_colors <- c("#33cc33", "#F8766D")
ar_st_colors_label <- c("v-shaped", "straight")
fl_legend_color_symbol_size <- 5
st_leg_color_lab <- "Engine-Shape"
# Shape controls
ar_it_shapes <- c(9, 15)
ar_st_shapes_label <- c("auto", "manuel")
fl_legend_shape_symbol_size <- 5
st_leg_shape_lab <- "Transmission"
```

Fourth, control the size of the scatter, which will be the MPG variable.

```
# Control scatter point size
fl_min_size <- 3
fl_max_size <- 6
ar_size_range <- c(fl_min_size, fl_max_size)
st_leg_size_lab <- "MPG"
```

Fifth, control graph strings.

```
# Labeling
st_title <- paste0('Distribution of HP and QSEC from mtcars')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/ggscatter/htmlpdfr/fs_ggscatter_3cts_mdisc.html')
st_caption <- paste0('mtcars dataset, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'HP = Horse Power'
st_y_label <- 'QSEC = time for 1/4 Miles'
```

Sixth, combine graphical components.

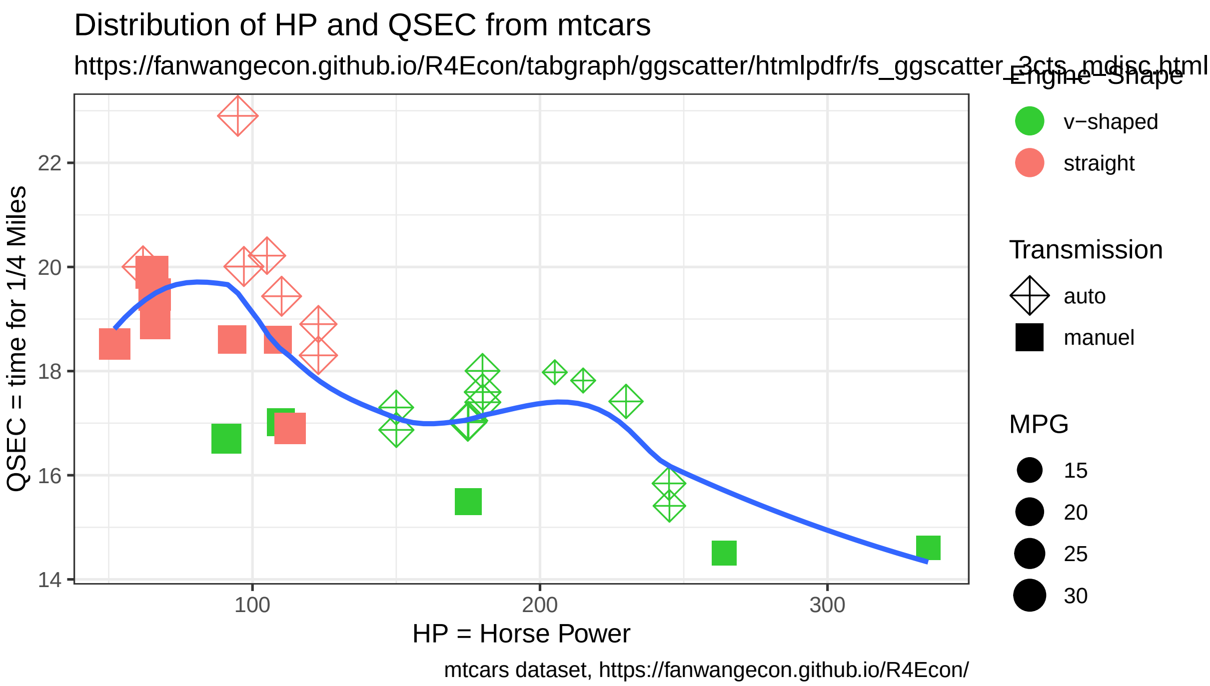
```
# Add titles and labels
plt_mtcars_scatter <- plt_mtcars_scatter +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption)
# Color, shape and size controls
plt_mtcars_scatter <- plt_mtcars_scatter +
  scale_colour_manual(values=ar_st_colors, labels=ar_st_colors_label) +
  scale_shape_manual(values=ar_it_shapes, labels=ar_st_shapes_label) +
  scale_size_continuous(range = ar_size_range)
```

Eigth, replace default legends titles for color, shape and size.

```
# replace the default labels for each legend segment
plt_mtcars_scatter <- plt_mtcars_scatter +
  labs(colour = st_leg_color_lab,
       shape = st_leg_shape_lab,
       size = st_leg_size_lab)
```

Ninth, additional controls for the graph.

```
# Control the order of legend display
# Show color, show shape, then show size.
plt_mtcars_scatter <- plt_mtcars_scatter + guides(
  colour = guide_legend(order = 1, override.aes = list(size = fl_legend_color_symbol_size)),
  shape = guide_legend(order = 2, override.aes = list(size = fl_legend_shape_symbol_size)),
  size = guide_legend(order = 3))
# show
print(plt_mtcars_scatter)
```



10.3.2 ggplot Multiple Scatter-Lines with Facet Wrap

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

10.3.2.1 Multiple Scatter-Lines with Facet Wrap

Two subplots, for auto and manual transitions. The x-axis is horse-power, the y-axis shows QSEC. Different colors represent v-shaped and straight-engines.

1. y-axis: time for 1/4 Miles (QSEC)
2. x-axis: horsepower (hp)
3. facet-wrap: auto or manual (am)
4. colored line and point shapes: vshaped or straight engine (vs)

First, Load in the mtcars dataset and convert to categorical variables to factor with labels.

```
# First make sure these are factors
tb_mtcars <- as_tibble(mtcars) %>%
  mutate(vs = as_factor(vs), am = as_factor(am))
# Second Label the Factors
am_levels <- c(auto_shift = "0", manual_shift = "1")
vs_levels <- c("vshaped engine" = "0", "straight engine" = "1")
tb_mtcars <- tb_mtcars %>%
  mutate(vs = fct_recode(vs, !!!vs_levels),
         am = fct_recode(am, !!!am_levels))
```

Second, generate the core graph, a line plot and facet wrapping over the *am* variable. Note that *vs* variable has different color as well as line type and shape

```
# Graphing
plt_mtcars_scatter <-
  ggplot(tb_mtcars, aes(x=hp, y=qsec,
                        colour=am, shape=am, linetype=am)) +
  geom_smooth(se = FALSE, lwd = 1.5) + # Lwd = line width
  geom_point(size = 5, stroke = 2) + # stroke = point shape width
  facet_wrap(~ vs,
             scales = "free_x",
             nrow = 1, ncol = 2,
             labeller = label_wrap_gen(multi_line=FALSE))
```

Third, control Color, Shape and Line-type Information. There will be two colors, two shapes and two linetypes. See all shape listing and linetype listing., See all shape listing.

```
# Color controls
ar_st_colors <- c("#33cc33", "#F8766D")
ar_st_colors_label <- c("auto", "manual")
fl_legend_color_symbol_size <- 5
st_leg_color_lab <- "Transmission"
# Shape controls
ar_it_shapes <- c(1, 5)
ar_st_shapes_label <- c("auto", "manual")
fl_legend_shape_symbol_size <- 5
st_leg_shape_lab <- "Transmission"
# Line-Type controls
ar_st_linetypes <- c('solid', 'dashed')
ar_st_linetypes_label <- c("auto", "manual")
fl_legend_linetype_symbol_size <- 5
st_leg_linetype_lab <- "Transmission"
```

Fourth, manaully specify an x-axis.

```
# x labeling and axis control
ar_st_x_labels <- c('50 hp', '150 hp', '250 hp', '350 hp')
ar_fl_x_breaks <- c(50, 150, 250, 350)
ar_fl_x_limits <- c(40, 360)
# y labeling and axis control
ar_st_y_labels <- c('15 QSEC', '18', '21', '24 QSEC')
ar_fl_y_breaks <- c(15, 18, 21, 24)
ar_fl_y_limits <- c(13.5, 25.5)
```

Fifth, control graph strings.

```
# Labeling
st_title <- paste0('How QSEC varies by Horse-power, by Engine and Transmission Types')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/multiplot/htmlpdfr/fs_ggscatter_facet_wrap.html')
st_caption <- paste0('mtcars dataset, ',
```

```
                      'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'HP = Horse Power'
st_y_label <- 'QSEC = time for 1/4 Miles'
```

Sixth, combine graphical components.

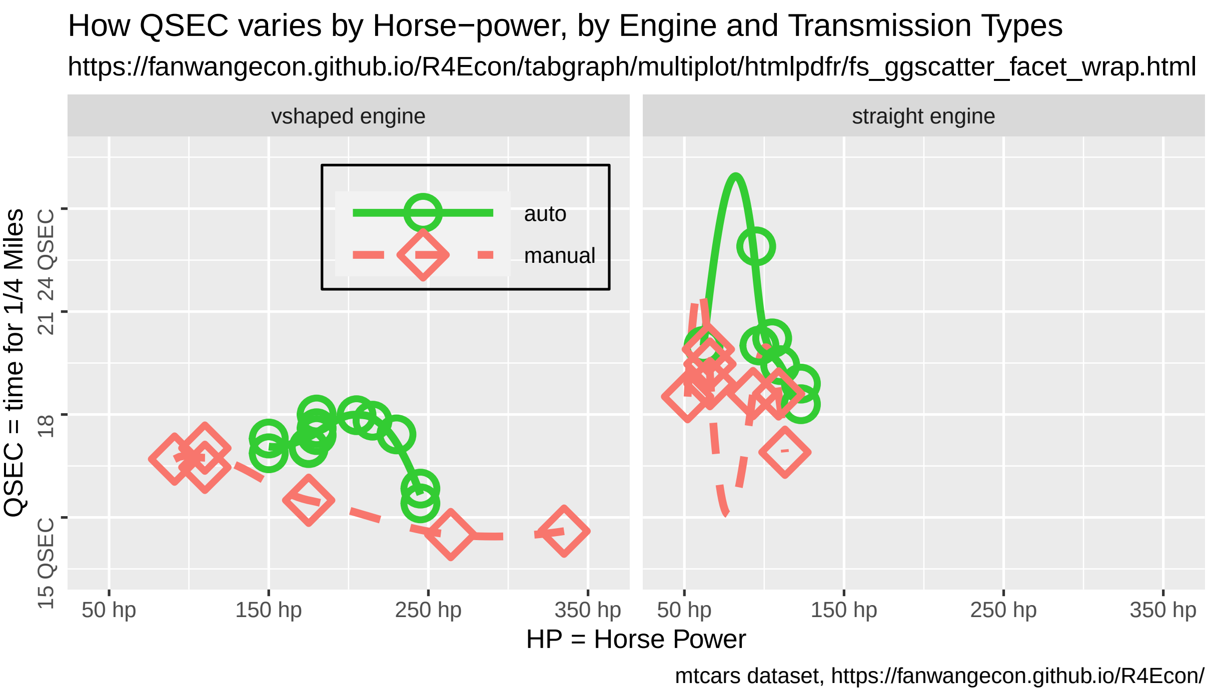
```
# Add titles and labels
plt_mtcars_scatter <- plt_mtcars_scatter +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption)

# x and y-axis ticks controls
plt_mtcars_scatter <- plt_mtcars_scatter +
  scale_x_continuous(labels = ar_st_x_labels,
                     breaks = ar_fl_x_breaks,
                     limits = ar_fl_x_limits) +
  scale_y_continuous(labels = ar_st_y_labels,
                     breaks = ar_fl_y_breaks,
                     limits = ar_fl_y_limits)

# Color, shape and linetype controls
plt_mtcars_scatter <- plt_mtcars_scatter +
  scale_colour_manual(values=ar_st_colors, labels=ar_st_colors_label) +
  scale_shape_manual(values=ar_it_shapes, labels=ar_st_shapes_label) +
  scale_linetype_manual(values=ar_st_linetypes, labels=ar_st_linetypes_label)
```

Seventh, replace default legends, and set figure font overall etc.

```
# has legend theme
theme_custom <- theme(
  text = element_text(size = 11),
  axis.text.y = element_text(angle = 90),
  legend.title = element_blank(),
  legend.position = c(0.35, 0.80),
  legend.key.width = unit(5, "line"),
  legend.background =
    element_rect(fill = "transparent", colour = "black", linetype='solid'))
# no legend theme (no y)
theme_custom_blank <- theme(
  text = element_text(size = 12),
  legend.title = element_blank(),
  legend.position = "none",
  axis.title.y=element_blank(),
  axis.text.y=element_blank(),
  axis.ticks.y=element_blank())
```

Eighth, graph out.

```
# replace the default labels for each legend segment
plt_mtcars_scatter <- plt_mtcars_scatter + theme_custom
# show
print(plt_mtcars_scatter)
```



10.3.2.2 Divide Facet Wrapped Plot into Subplots

Given the facet-wrapped plot just generated, now save alternative plot versions, where each subplot is saved by itself. Will simply use the code from above, but call inside lapply over different am categories.

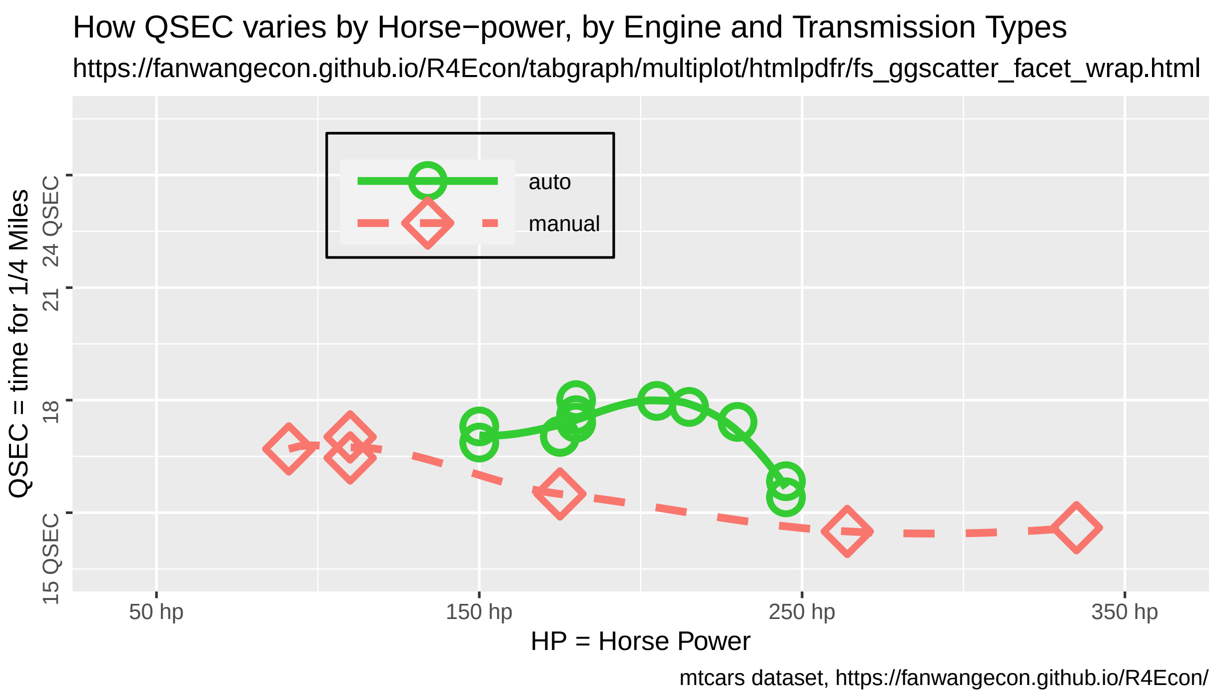
Below generate a matrix with multiple data columns, then use apply over each row of the matrix and select different columns of values for each subplot generation.

First generate with legend versions, then without legend versions. These are versions that would be used to more freely compose graph together.

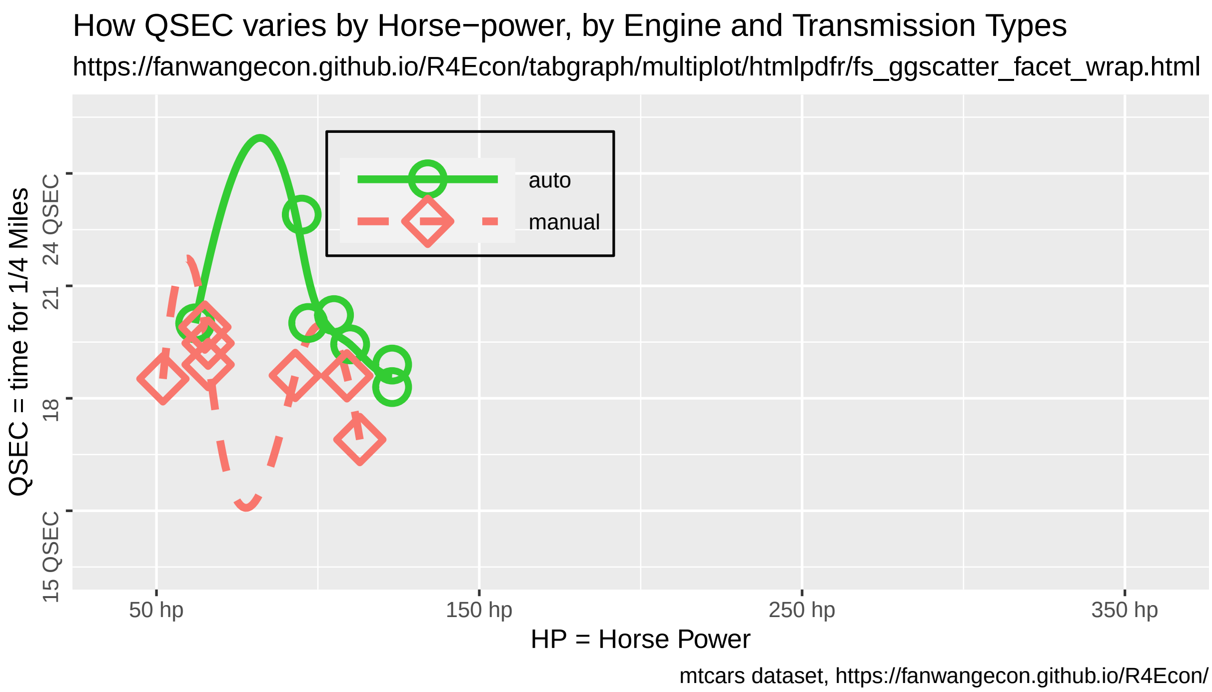
```
for (it_subplot_as_own_vsr in c(1,2)) {

  if (it_subplot_as_own_vsr == 1) {
    theme_custom_use <- theme_custom
    # st_file_suffix <- '_haslegend'
    # it_width <- 100
  } else if (it_subplot_as_own_vsr == 2) {
    theme_custom_use <- theme_custom_blank
    # st_file_suffix <- '_nolegend'
    # it_width <- 88
  }

  # unique vs as matrix
  # ar_uniques <-sort(unique(tb_mtcars$vs))
  # mt_unique_vs <- matrix(data=ar_uniques, nrow=length(ar_uniques), ncol=1)
  mt_unique_vs <- tb_mtcars %>% group_by(vs) %>%
    summarize(mpg=mean(mpg)) %>% ungroup()
  # apply over
  ls_plots <- apply(mt_unique_vs, 1, function(ar_vs_cate_row) {
    # 1. Graph main
    plt_mtcars_scatter <-
      ggplot(tb_mtcars %>% filter(vs == ar_vs_cate_row[1]),
             aes(x=hp, y=qsec,
                 colour=am, shape=am, linetype=am)) +
      geom_smooth(se = FALSE, lwd = 1.5) + # Lwd = line width
      geom_point(size = 5, stroke = 2)
```

```
    # 2. Add titles and labels
    plt_mtcars_scatter <- plt_mtcars_scatter +
      labs(title = st_title, subtitle = st_subtitle,
           x = st_x_label, y = st_y_label, caption = st_caption)

    # 3. x and y ticks
    plt_mtcars_scatter <- plt_mtcars_scatter +
      scale_x_continuous(labels = ar_st_x_labels, breaks = ar_fl_x_breaks, limits = ar_fl_x_limits)
      scale_y_continuous(labels = ar_st_y_labels, breaks = ar_fl_y_breaks, limits = ar_fl_y_limits)

    # 4. Color, shape and linetype controls
    plt_mtcars_scatter <- plt_mtcars_scatter +
      scale_colour_manual(values=ar_st_colors, labels=ar_st_colors_label) +
      scale_shape_manual(values=ar_it_shapes, labels=ar_st_shapes_label) +
      scale_linetype_manual(values=ar_st_linetypes, labels=ar_st_linetypes_label)

    # 5. replace the default labels for each legend segment
    plt_mtcars_scatter <- plt_mtcars_scatter + theme_custom_use
  })
  # show
  print(ls_plots)
}
```

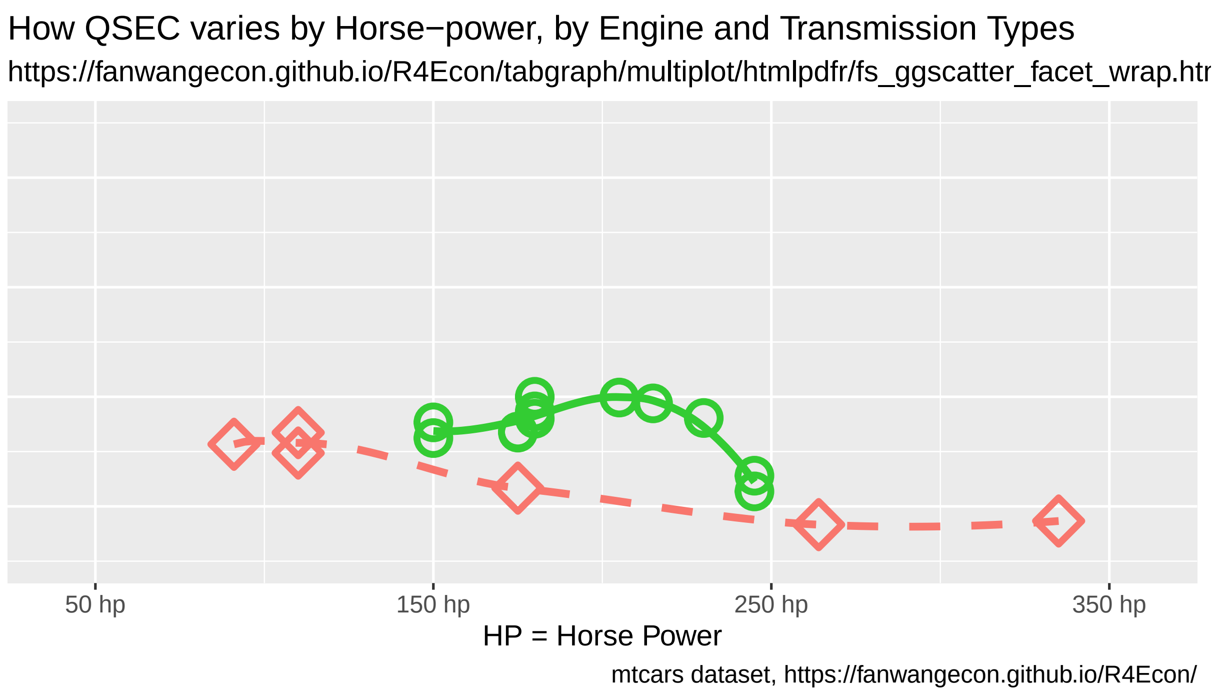
```
## [[1]]
```



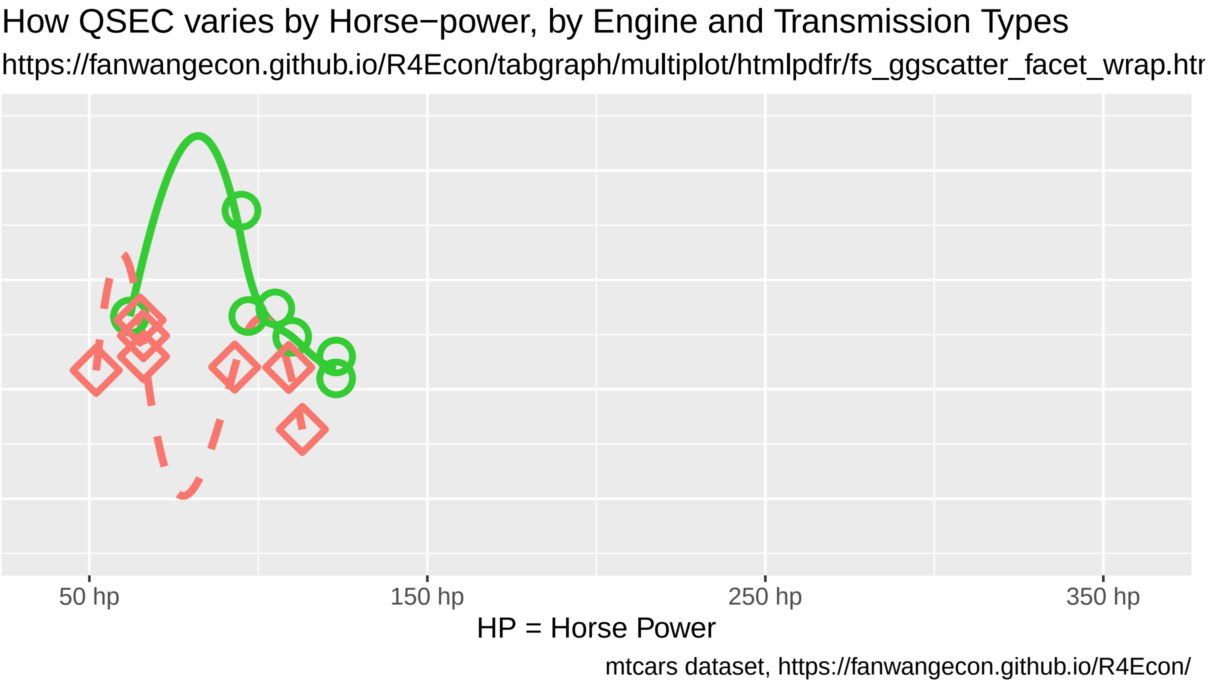

```
##
## [[2]]
```


How QSEC varies by Horse−power, by Engine and Transmission Types
https://fanwangecon.github.io/R4Econ/tabgraph/multiplot/htmlpdfr/fs_ggscatter_facet_wrap.html
auto
manual
QSEC = time for 1/4 Miles
15 QSEC
18
21
24 QSEC
50 hp
150 hp
250 hp
350 hp
HP = Horse Power
mtcars dataset, https://fanwangecon.github.io/R4Econ/

```
##
## [[1]]
```


How QSEC varies by Horse−power, by Engine and Transmission Types
https://fanwangecon.github.io/R4Econ/tabgraph/multiplot/htmlpdfr/fs_ggscatter_facet_wrap.htr
50 hp
150 hp
250 hp
350 hp
HP = Horse Power
mtcars dataset, https://fanwangecon.github.io/R4Econ/

```
##
## [[2]]
```



10.4 Write and Read Plots

10.4.1 Import and Export Images

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

Work with the R plot function.

10.4.1.1 Export Images Different Formats with Plot()

10.4.1.1.1 Generate and Record A Plot Generate a graph and recordPlot() it. The generated graph does not have legends Yet. Crucially, there are no titles, legends, axis, labels in the figures. As we stack the figures together, do not add those. Only add at the end jointly for all figure elements together to control at one spot things.

```
#######################################################
# First, Strings
#######################################################
# Labeling
st_title <- paste0('Scatter, Line and Curve Joint Ploting Example Using Base R\n',
                   'plot() + curve():sin(x)*cos(x), sin(x)+tan(x)+cos(x)')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/inout/htmlpdfr/fs_base_curve.html')
st_x_label <- 'x'
st_y_label <- 'f(x)'

#######################################################
# Second, functions
#######################################################
fc_sin_cos_diff <- function(x) sin(x)*cos(x)
st_line_3_y_legend <- 'sin(x)*cos(x)'
fc_sin_cos_tan <- function(x) sin(x) + cos(x) + tan(x)
st_line_4_y_legend <- 'sin(x) + tan(x) + cos(x)'

#######################################################
# Third, patterns
```

```
########################################################
st_line_3_black <- 'black'
st_line_4_purple <- 'orange'
# line type
st_line_3_lty <- 'dotted'
st_line_4_lty <- 'dotdash'
# line width
st_line_3_lwd <- 2.5
st_line_4_lwd <- 3.5

########################################################
# Fourth: Share xlim and ylim
########################################################
ar_xlim = c(-3, 3)
ar_ylim = c(-3.5, 3.5)

########################################################
# Fifth: Even margins
########################################################
par(new=FALSE)

########################################################
# Sixth, the four objects and do not print yet:
########################################################
# Graph Curve 3
par(new=T)
curve(fc_sin_cos_diff,
      col = st_line_3_black,
      lwd = st_line_3_lwd, lty = st_line_3_lty,
      from = ar_xlim[1], to = ar_xlim[2], ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
# Graph Curve 4
par(new=T)
curve(fc_sin_cos_tan,
      col = st_line_4_purple,
      lwd = st_line_4_lwd, lty = st_line_4_lty,
      from = ar_xlim[1], to = ar_xlim[2], ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
```

```
pl_curves_save <- recordPlot()
```

10.4.1.1.2 Generate Large Font and Small Font Versions of PLot Generate larger font version:

```
# Replay
pl_curves_save

########################################################
# Seventh, Set Title and Legend and Plot Jointly
########################################################
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg = 0.75
fl_ces_fig_leg = 0.75
fl_ces_fig_small = 0.65

# R Legend
title(main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
      cex.lab=fl_ces_fig_reg,
      cex.main=fl_ces_fig_reg,
```

```
      cex.sub=fl_ces_fig_small)
axis(1, cex.axis=fl_ces_fig_reg)
axis(2, cex.axis=fl_ces_fig_reg)
grid()

# Legend sizing CEX
legend("topleft",
       bg="transparent",
       bty = "n",
       c(st_line_3_y_legend, st_line_4_y_legend),
       col = c(st_line_3_black, st_line_4_purple),
       pch = c(NA, NA),
       cex = fl_ces_fig_leg,
       lty = c(st_line_3_lty, st_line_4_lty),
       lwd = c(st_line_3_lwd,st_line_4_lwd),
       y.intersp=2)
```

```
# record final plot
pl_curves_large <- recordPlot()
dev.off()
```

Generate smaller font version:

```
# Replay
pl_curves_save

#######################################################
# Seventh, Set Title and Legend and Plot Jointly
#######################################################
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg = 0.45
fl_ces_fig_leg = 0.45
fl_ces_fig_small = 0.25

# R Legend
title(main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
      cex.lab=fl_ces_fig_reg,
      cex.main=fl_ces_fig_reg,
      cex.sub=fl_ces_fig_small)
axis(1, cex.axis=fl_ces_fig_reg)
axis(2, cex.axis=fl_ces_fig_reg)
grid()

# Legend sizing CEX
legend("topleft",
       bg="transparent",
       bty = "n",
       c(st_line_3_y_legend, st_line_4_y_legend),
       col = c(st_line_3_black, st_line_4_purple),
       pch = c(NA, NA),
       cex = fl_ces_fig_leg,
       lty = c(st_line_3_lty, st_line_4_lty),
       lwd = c(st_line_3_lwd,st_line_4_lwd),
       y.intersp=2)
```

```
# record final plot
pl_curves_small <- recordPlot()
dev.off()
```

10.4.1.1.3 Save Plot with Varying Resolutions and Heights

Export recorded plot.

A4 paper is 8.3 x 11.7, with 1 inch margins, the remaining area is 6.3 x 9.7. For figures that should take half of the page, the height should be 4.8 inch. One third of a page should be 3.2 inch. 6.3 inch is 160mm and 3 inch is 76 mm. In the example below, use

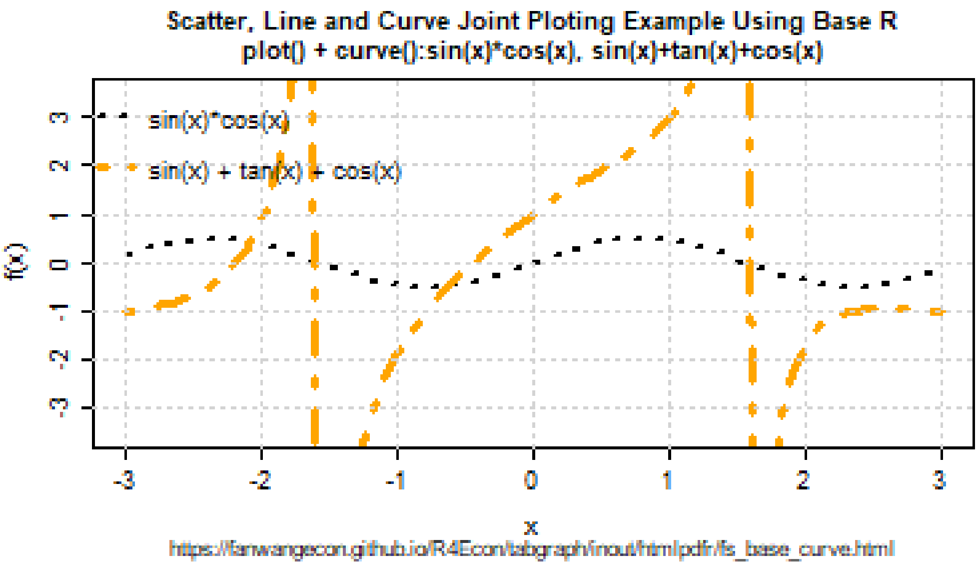
```
# Store both in within folder directory and root image directory:
# C:\Users\fan\R4Econ\tabgraph\inout\_img
# C:\Users\fan\R4Econ\_img
# need to store in both because bookdown and indi pdf path differ.
# Wrap in try because will not work underbookdown, but images already created

ls_spt_root <- c('..//..//', '')
spt_prefix <- '_img/fs_img_io_2curve'

for (spt_root in ls_spt_root) {
  # Changing pointsize will not change font sizes inside, just rescale
  # PNG 72
  try(png(paste0(spt_root, spt_prefix, "_w135h76_res72.png"),
      width = 135 , height = 76, units='mm', res = 72, pointsize=7))
  print(pl_curves_large)
  dev.off()
  # PNG 300
  try(png(paste0(spt_root, spt_prefix, "_w135h76_res300.png"),
      width = 135, height = 76, units='mm', res = 300, pointsize=7))
  print(pl_curves_large)
  dev.off()
  # PNG 300, SMALL, POINT SIZE LOWER
  try(png(paste0(spt_root, spt_prefix, "_w80h48_res300.png"),
      width = 80, height = 48, units='mm', res = 300, pointsize=7))
  print(pl_curves_small)
  dev.off()
  # PNG 300
  try(png(paste0(spt_root, spt_prefix, "_w160h100_res300.png"),
      width = 160, height = 100, units='mm', res = 300))
  print(pl_curves_large)
  dev.off()

  # EPS
  setEPS()
  try(postscript(paste0(spt_root, spt_prefix, "_fs_2curve.eps")))
  print(pl_curves_large)
  dev.off()
}
```

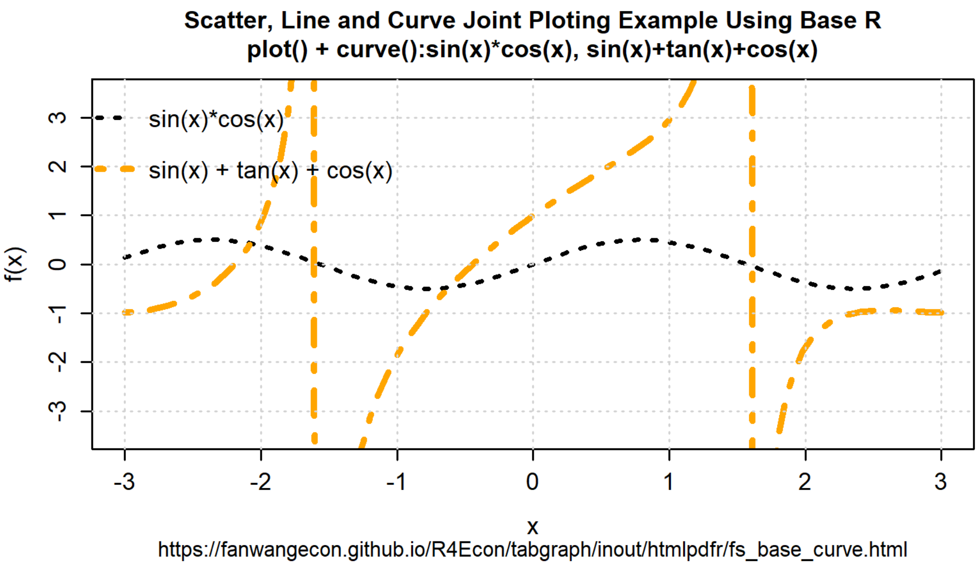
```
## Error in png(paste0(spt_root, spt_prefix, "_w135h76_res72.png"), width = 135,  :
##   unable to start png() device
## Error in png(paste0(spt_root, spt_prefix, "_w135h76_res300.png"), width = 135,  :
##   unable to start png() device
## Error in png(paste0(spt_root, spt_prefix, "_w80h48_res300.png"), width = 80,  :
##   unable to start png() device
## Error in png(paste0(spt_root, spt_prefix, "_w160h100_res300.png"), width = 160,  :
##   unable to start png() device
## Error in postscript(paste0(spt_root, spt_prefix, "_fs_2curve.eps")) :
##   cannot open file '..//..//_img/fs_img_io_2curve_fs_2curve.eps'
```

10.4.1.1.4 Low and High Resolution Figure

The standard resolution often produces very low quality images. Resolution should be increased. See figure comparison.

RES=72 (DEFAULT R) TOP, RES=300 Bottom, (Width=160, Height=81, PNG)

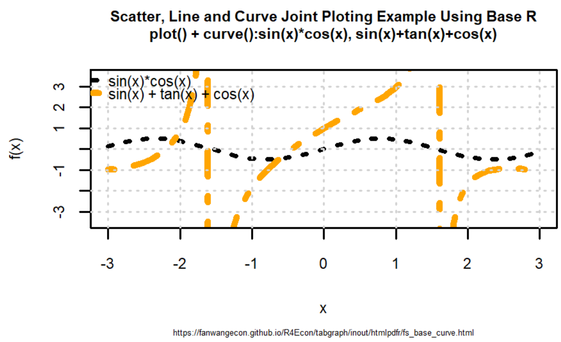


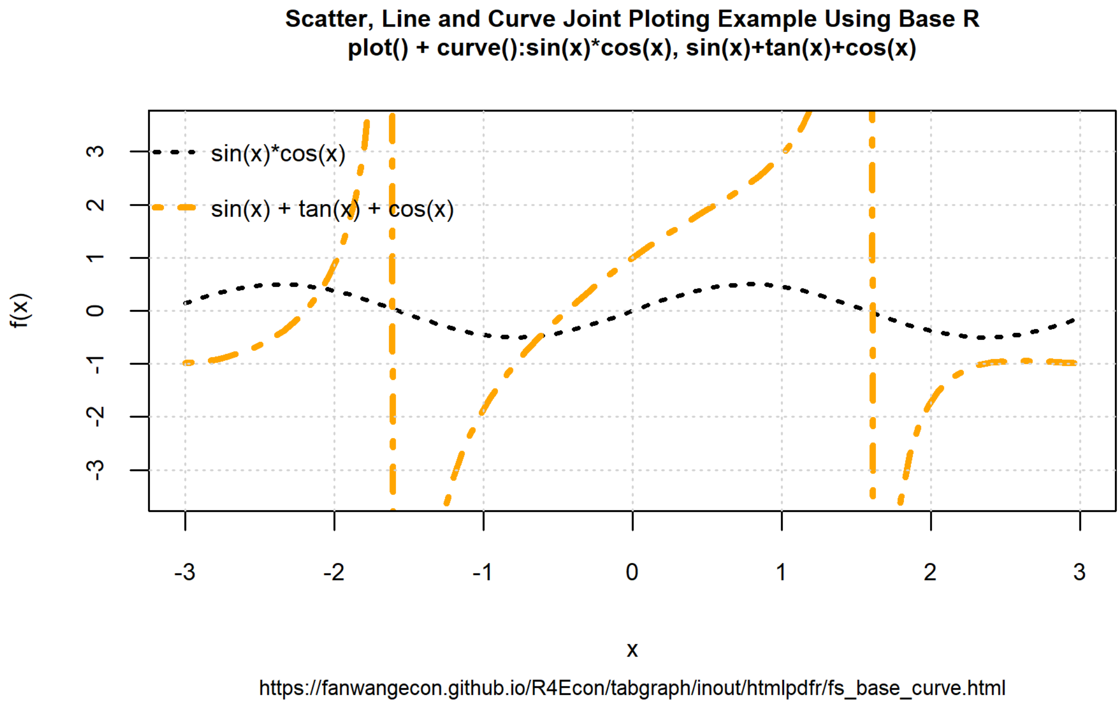


10.4.1.1.5 Smaller and Larger Figures

Smaller and larger figures with different font size comparison. Note that earlier, we generated the figure without legends, labels, etc first, recorded the figure. Then we associated the same underlying figure with differently sized titles, legends, axis, labels.

Top Small (small font saved), Bottom Large, PNG





Chapter 11

Get Data

11.1 Environmental Data

11.1.1 ECMWF ERA5 Data

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

This files uses R with the reticulate package to download ECMWF ERA5 data. See this file for instructions and tutorials for downloading the data.

11.1.1.1 Program to Download, Unzip, Convert to combined CSV, derived-utci-historical data

The data downloaded from CDS climate could become very large in size. We want to process parts of the data one part at a time, summarize and aggregate over each part, and generate a file output file with aggregate statistics over the entire time period of interest.

This code below accompalishes the following tasks:

1. download data from derived-utci-historical as ZIP
2. unzip
3. convert *nc* files to *csv* files
4. individual csv files are half year groups

Parameter Control for the code below:

1. *spt_root*: root folder where everything will be at
2. *spth_conda_env*: the conda virtual environment python path, eccodes and cdsapi packages are installed in the conda virtual environment. In the example below, the first env is: wk_ecmwf
3. *st_nc_prefix*: the downloaded individual nc files have dates and prefix before and after the date string in the nc file names. This is the string before that.
4. *st_nc_suffix*: see (3), this is the suffix
5. *ar_years*: array of years to download and aggregate over
6. *ar_months_g1*: months to download in first half year
7. *ar_months_g2*: months to download in second half year

Note: *area* below corresponds to *North, West, South, East*.

```
#################################################
# ------------ Parameters
#################################################

# Where to store everything
spt_root <- "C:/Users/fan/Downloads/_data/"
spth_conda_env <- "C:/ProgramData/Anaconda3/envs/wk_ecmwf/python.exe"
```

```
# nc name prefix
st_nc_prefix <- "ECMWF_utci_"
st_nc_suffix <- "_v1.0_con.nc"

# Years list
# ar_years <- 2001:2019
ar_years <- c(2005, 2015)
# ar_months_g1 <- c('01','02','03','04','05','06')
ar_months_g1 <- c('01', '03')
# ar_months_g2 <- c('07','08','09','10','11','12')
ar_months_g2 <- c('07', '09')

# Area
# # China
# fl_area_north <- 53.31
# fl_area_west <- 73
# fl_area_south <- 4.15
# fl_area_east <- 135
fl_area_north <- 53
fl_area_west <- 73
fl_area_south <- 52
fl_area_east <- 74

# folder to download any nc zips to
nczippath <- spt_root
# we are changing the python api file with different requests stirngs and storing it here
pyapipath <- spt_root
# output directory for AGGREGATE CSV with all DATES from this search
csvpath <- spt_root

#################################################
# ------------ Packages
#################################################

library("ncdf4")
library("chron")
library("lattice")
library("RColorBrewer")
library("stringr")
library("tibble")
library("dplyr")
Sys.setenv(RETICULATE_PYTHON = spth_conda_env)
library("reticulate")

#################################################
# ------------ Define Loops
#################################################
for (it_yr in ar_years) {
  for (it_mth_group in c(1,2)) {
    if(it_mth_group == 1) {
      ar_months = ar_months_g1
    }
    if(it_mth_group == 2) {
      ar_months = ar_months_g2
    }

    #################################################
    # ------------ Define Python API Call
```

```
    ##################################################

    # name of zip file
    nczipname <- "derived_utci_2010_2.zip"
    unzipfolder <- "derived_utci_2010_2"

    st_file <- paste0("import cdsapi
import urllib.request
# download folder
spt_root = '", nczippath, "'
spn_dl_test_grib = spt_root + '", nczipname, "'
# request
c = cdsapi.Client()
res = c.retrieve(
    'derived-utci-historical',
    {
        'format': 'zip',
        'variable': 'Universal thermal climate index',
        'product_type': 'Consolidated dataset',
        'year': '",it_yr, "',
        'month': [
            ", paste("'", ar_months, "'", sep = "", collapse = ", "), "
        ],
        'day': [
            '01','03'
        ],
        'area'  : [", fl_area_north ,", ", fl_area_west ,", ", fl_area_south ,", ", fl_area_east ,"]
        'grid'  : [0.25, 0.25],
    },
    spn_dl_test_grib)
# show results
print('print results')
print(res)
print(type(res))")

    # st_file = "print(1+1)"

    # Store Python Api File
    fl_test_tex <- paste0(pyapipath, "api.py")
    fileConn <- file(fl_test_tex)
    writeLines(st_file, fileConn)
    close(fileConn)

    ##################################################
    # ------------ Run Python File
    ##################################################
    # Set Path
    setwd(pyapipath)
    # Run py file, api.py name just defined
    use_python(spth_conda_env)
    source_python('api.py')

    ##################################################
    # ------------ uNZIP
    ##################################################
    spn_zip <- paste0(nczippath, nczipname)
    spn_unzip_folder <- paste0(nczippath, unzipfolder)
    unzip(spn_zip, exdir=spn_unzip_folder)
```

```r
##################################################
# ------------ Find All files
##################################################
# Get all files with nc suffix in folder
ncpath <- paste0(nczippath, unzipfolder)
ls_sfls <- list.files(path=ncpath, recursive=TRUE, pattern=".nc", full.names=T)

##################################################
# ------------ Combine individual NC files to JOINT Dataframe
##################################################
# List to gather dataframes
ls_df <- vector(mode = "list", length = length(ls_sfls))
# Loop over files and convert nc to csv
it_df_ctr <- 0
for (spt_file in ls_sfls) {
  it_df_ctr <- it_df_ctr + 1

  # Get file name without Path
  snm_file_date <- sub(paste0('\\',st_nc_suffix,'$'), '', basename(spt_file))
  snm_file_date <- sub(st_nc_prefix, '', basename(snm_file_date))

  # Dates Start and End: list.files is auto sorted in ascending order
  if (it_df_ctr == 1) {
    snm_start_date <- snm_file_date
  }
  else {
    # this will give the final date
    snm_end_date <- snm_file_date
  }

  # Given this structure: ECMWF_utci_20100702_v1.0_con, sub out prefix and suffix
  print(spt_file)
  ncin <- nc_open(spt_file)

  nchist <- ncatt_get(ncin, 0, "history")

  # not using this missing value flag at the moment
  missingval <- str_match(nchist$value, "setmisstoc,\\s*(.*?)\\s* ")[,2]
  missingval <- as.numeric(missingval)

  lon <- ncvar_get(ncin, "lon")
  lat <- ncvar_get(ncin, "lat")
  tim <- ncvar_get(ncin, "time")
  tunits <- ncatt_get(ncin, "time", "units")

  nlon <- dim(lon)
  nlat <- dim(lat)
  ntim <- dim(tim)

  # convert time -- split the time units string into fields
  # tustr <- strsplit(tunits$value, " ")
  # tdstr <- strsplit(unlist(tustr)[3], "-")
  # tmonth <- as.integer(unlist(tdstr)[2])
  # tday <- as.integer(unlist(tdstr)[3])
  # tyear <- as.integer(unlist(tdstr)[1])
  # mytim <- chron(tim, origin = c(tmonth, tday, tyear))
```

```
      tmp_array <- ncvar_get(ncin, "utci")
      tmp_array <- tmp_array - 273.15

      lonlat <- as.matrix(expand.grid(lon = lon, lat = lat, hours = tim))
      temperature <- as.vector(tmp_array)
      tmp_df <- data.frame(cbind(lonlat, temperature))

      # extract a rectangle
      eps <- 1e-8
      minlat <- 22.25 - eps
      maxlat <- 23.50 + eps
      minlon <- 113.00 - eps
      maxlon <- 114.50 + eps
      # subset data
      subset_df <- tmp_df[tmp_df$lat >= minlat & tmp_df$lat <= maxlat &
                          tmp_df$lon >= minlon & tmp_df$lon <= maxlon, ]

      # add Date
      subset_df_date <- as_tibble(subset_df) %>% mutate(date = snm_file_date)

      # Add to list
      ls_df[[it_df_ctr]] <- subset_df_date

      # Close NC
      nc_close(ncin)
    }

    # List of DF to one DF
    df_all_nc <- do.call(rbind, ls_df)

    # Save File
    fname <- paste0(paste0(st_nc_prefix,
                          snm_start_date, "_to_", snm_end_date,
                          ".csv"))
    csvfile <- paste0(csvpath, fname)
    write.table(na.omit(df_all_nc), csvfile, row.names = FALSE, sep = ",")

    # Delete folders
    unlink(spn_zip, recursive=TRUE, force=TRUE)
    unlink(spn_unzip_folder, recursive=TRUE, force=TRUE)

  # end loop months groups
  }
# end loop year
}
```

Chapter 12

Coding and Development

12.1 Installation and Packages

12.1.1 R Installation and Set-Up

Install R new, or update an existing R installation.

> Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

12.1.1.1 Uninstall r

Uninstall R, RStudio and RTools from Windows "Programs and Features" menu. After uninstaller finishes, check in the libPath folders to see if there are still stuff there, delete all, delete the folders.

If R was installed in a virtual environment, delete the environment. Otherwise, use system's uninstaller. To check on this, from terminal/command-prompt for each virtual environment, type *R*, and *.libPaths()* to find paths. Do so inside conda main, outside of conda main, inside different environments, find all paths.

Inside R:

```
ls_spn_paths <- .libPaths()
print(ls_spn_paths)
```

```
## [1] "C:/Users/fan/AppData/Local/R/win-library/4.3"
## [2] "C:/Program Files/R/R-4.3.1/library"
```

```
# C:/Users/fan/Documents/R/win-library/3.6
# C:/Program Files/R/R-3.6.1/library
```

Check which conda env is installed, if there is an env installed for R.

```
# Show All installed environments
conda info --envs
```

For Linux and for unsintalling inside conda:

```
# Exit Conda
conda deactivate
# where is R installed outside of Conda
which R
# /usr/bin/R
# To remove all
sudo apt-get remove r-base
sudo apt-get remove r-base-core

# Inside Conda base
```

```
conda activate
# Conda r_env
conda activate r_envr
# Where is it installed?
which R
# /home/wangfanbsg75/anaconda3/bin/R
conda uninstall r-base
```

12.1.1.2 Install R

12.1.1.2.1 Install R for the First Time

1. download R
    - for debian: Johannes Ranke. For Linux/Debian installation, crucial to update the *source.list* to include sources that have more recent versions of R. If not, will get very old R versions that is not compatible with many packages.
    - add R to path for Windows. In Windows Path, add for example: *C:/Program Files/R/R-3.6.2/bin/x64/* and *C:/Rtools/bin*
2. Install Rtools for building R packages.
3. download R-studio
4. Open R-studio and auto-detect R
5. Install additional packages

12.1.1.2.2 Linux R Install For linux/Debian, to Install latest R:

```
# Go to get latesdebian latest r sources.list
cat /etc/apt/sources.list
# Install this First (should already be installed)
sudo apt install dirmngr

# Debian R is maintained by Johannes Ranke, copied from https://cran.r-project.org/bin/linux/debian/
apt-key adv --keyserver keys.gnupg.net --recv-key 'E19F5F87128899B192B1A2C2AD5F960A256A04AF'
# Add to source.list, for debian stretch (9)
# sudo su added for security issue as super-user
sudo su -c "sudo echo 'deb http://cloud.r-project.org/bin/linux/debian stretch-cran35/' >> /etc/apt/
# if added wrong lines, delete 3rd line
sudo sed '3d' /etc/apt/sources.list

# Update and Install R, should say updated from cloud.r
sudo apt-get update
sudo apt-get install r-base r-base-dev

# Also install these, otherwise r-packages do not install
# libxml2 seems need for tidymodels
sudo apt-get install libcurl4-openssl-dev
sudo apt-get install libssl-dev
sudo apt-get install libxml2-dev
```

12.1.1.2.3 Update R on Windows First, use the *updateR()* function from the *installr* package.

1. On windows, install the *installr* package, and use *updateR()*
2. At the end, will ask if want to move all old packages to new R directory

New R will have new package directory, could keep all in old, should copy all to new, and not keep old. Can choose to copy all old packages to new folder, but still keep old packages in prior folder as they were

```
# https://www.r-project.org/nosvn/pandoc/installr.html
install.packages('installr')
# update R from inside R (not Rstudio)
require(installr)
```

```
# this will open dialog boxes to take you through the steps.
updateR()
# Set Rstudio to the Latest R
```

Second, after updating, might go into "Apps and Features" on Windows to unstaill the previous R version.

Third, update RTools. Uninstall RTools. Installation can be very large.

Fourth, update RStudio. Upon opening RStudio, if prior installations of R have been uninstalled, RStudio will auto-detect as ask if the latest version of R should be used. Choose yes. Upon entering RStudio, if there are updates, might prompt to RStudio website to download and update.

Fifth, follow the package installation directions below to update that.

12.1.1.2.4 R Add to Path To be able to use R via command line, make sure Windows knows where the path to R.exe is.

First, find where the installed R.exe path is. Open up the R installation, and then check path as below.

```
ls_spn_paths <- .libPaths()
print(ls_spn_paths)
# "C:/Users/fan/AppData/Local/R/win-library/4.2"
# "C:/Program Files/R/R-4.2.1/library"
```

Second, given the path found, the R.exe is at "C:/Program Files/R/R-4.2.1/bin". So now, in windows, System Properties -> Advanced -> Environment Variables -> System Variables -> Path -> Edit -> New -> Paste "C:/Program Files/R/R-4.2.1/bin"

Third, now open up command-prompt/terminal/git-bash, enter R, this will take us into the R console via command-line.

```
# To exit command line:
q()
```

12.1.1.3 R Package Installations

12.1.1.3.1 Update R Install Directory After installing R, change the path sequence so that packages install for all users.

```
ls_spn_paths <- .libPaths()
print(ls_spn_paths)
# [1] "C:/Users/fan/AppData/Local/R/win-library/4.2" "C:/Program Files/R/R-4.2.1/library"
ls_spn_paths <- c(ls_spn_paths[2], ls_spn_paths[1])
.libPaths(ls_spn_paths)
ls_spn_paths <- .libPaths()
print(ls_spn_paths)
# [1] "C:/Program Files/R/R-4.2.1/library"           "C:/Users/fan/AppData/Local/R/win-library/4.2"
```

12.1.1.3.2 Install vearious directory After updating R, sometimes, old packages are not copied over to new directory, so need to reinstall all packages.

Having set the directories earlier so that packages do not install in user's personal folder, but the library folder where the R version is installed, we can find all installed packages in the *C:/Program Files/R/R-4.2.1/library* folder.

```
# Install RTools First!
# https://cran.r-project.org/bin/windows/Rtools/

# Install system tools
install.packages(c("backports"))

# Install tidyverse
```

```
install.packages(c("tidyverse", "tidymodels", "vroom"))

# Install Packaging tools
install.packages(c("devtools", "pkgdown", "roxygen2", "bookdown", "knitr", "kableExtra", "formatR",

# Install Statistics models
install.packages(c("AER", "minpack.lm"))
install.packages(c("quantreg"))

# Install Tools to Work with Other Packages
# matconv: converts matlab programs to R
install.packages(c("reticulate", "JuliaCall", "matconv"))
install.packages(c("matconv"))
# for reticulate errors, install directly from: devtools::install_github("rstudio/reticulate")

# Install Paralell Tools
install.packages(c("parallel", "doParallel", "foreach"))

# Install personal Packages
devtools::install_github("fanwangecon/REconTools")
devtools::install_github("fanwangecon/PrjOptiAlloc")

# Stata in Rmd
# devtools::install_github("Hemken/Statamarkdown")

# VScode integration and also sublime r-ide
install.packages("languageserver")
```

Temp Installs:

```
# 2020-10-19
# Temp install development version due to but
# https://github.com/rstudio/reticulate/issues/831
devtools::install_github("rstudio/reticulate")
```

12.1.1.4 R Tests

Test the following file to see if we can execute a R file. Do it inside *r_env* and inside a *r* session.

```
# # A simple file with summary statistics using tidyverse
# source('C:/Users/fan/R4Econ/summarize/dist/htmlpdfr/fst_hist_onevar.R')
# source('G:/repos/R4Econ/summarize/dist/htmlpdfr/fst_hist_onevar.R')
# # Another simple file with summary statistics using tidyverse
# source('C:/Users/fan/R4Econ/support/tibble/htmlpdfr/fs_tib_basics.R')
# source('G:/repos/R4Econ/support/tibble/htmlpdfr/fs_tib_basics.R')
# # A file involving estimation
# source('C:/Users/fan/R4Econ/optimization/cesloglin/htmlpdfr/fst_ces_plan_linlog.R')
#
# C:/Users/fan/R4Econ/summarize/dist/fst_hist_onevar.Rmd
# C:/Users/fan/R4Econ/support/tibble/fs_tib_basics.Rmd
# C:/Users/fan/R4Econ/optimization/cesloglin/fst_ces_plan_linlog.Rmd
```

12.1.1.5 R with Radian

R with Radian:

- Setup Visual Studio Code to run R on VSCode 2021
- Radian

12.1.1.6 Running R Inside VSCode

Rstudio seems laggy sometimes, nice to be able to run R from VSCode, use VSCode as an alternative editor.

- Writing R in VSCode: A Fresh Start

To Run .R scripts from inside VSCode. Here is the offical guide: R in Visual Studio Code.

1. Install R following prior steps.
2. Make sure that the package languageserver is installed, check "require(languageserver)".
3. Install R Extension for Visual Studio Code

- change setting for "r.rterm.windows" to "C:/Program Files/R/R-4.2.1/bin/R.exe"

4. Click File -> Open Folder -> Select the folder where the ".R" file to run is located at, this way, the Terminal will be directred to the folder that is currently open, rather than home default for example. By default, the terminal will open at the folder that is opened in the Explorer
5. Run individual files in the folder just opened.

12.1.1.6.1 RMD in VSCode

Steps for RMD:

1. VSCode already has default markdown editor
2. Install Markdown Preview ehance, which generates a table of content bar on the side, and has math preview correclty
3. In file, F1, and type markdown preview and open up preview

12.1.1.6.2 RMD and MARKDOWN File Associations

Change File Extension Association

- Working with RMD file, sometimes want to preview as MD file to view equations, sometimes want to view as RMD file to edit the R code. See How to make VS Code to treat other file extensions as certain language?
- change extension association between md and Rmd for Rmd files for example: "Ctrl + shift + p" and "change language mode" from one file association to another.
- Additionally, if there is a standard association we want for RMD, for it to be markdown for example, can add to JSON settings for *file.associations*.

```
"files.associations": {
     "*.Rmd": "markdown"
}
```

12.1.2 R Package Installation

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

12.1.2.1 Duplicate function names across packages

dplyr::filter and stats::filter are two functions from two popular packages that have the same name. And this leads to erros, when `dplyr::filter` is confused for `stats::filter`. We use the conflicted::conflict_prefer to resolve this issue.

This is an issue related to namespaces.

Below return the environment on the search path via rlang::search_envs():

```r
print(rlang::search_envs())
```

```
##  [[1]] $ <env: global>
##  [[2]] $ <env: .conflicts>
##  [[3]] $ <env: package:quantreg>
##  [[4]] $ <env: package:SparseM>
##  [[5]] $ <env: package:AER>
##  [[6]] $ <env: package:survival>
```

```
##  [[7]] $ <env: package:sandwich>
##  [[8]] $ <env: package:lmtest>
##  [[9]] $ <env: package:zoo>
## [[10]] $ <env: package:car>
## [[11]] $ <env: package:carData>
## [[12]] $ <env: tools:rstudio>
## [[13]] $ <env: package:stats>
## [[14]] $ <env: package:graphics>
## [[15]] $ <env: package:grDevices>
## [[16]] $ <env: package:utils>
## [[17]] $ <env: package:datasets>
## [[18]] $ <env: package:reticulate>
## [[19]] $ <env: package:conflicted>
## [[20]] $ <env: package:formatR>
## ... and 16 more environments
```

We can use tidyverse_conflicts() to "lists all the conflicts between packages in the tidyverse and other packages that you have loaded". We can see that we have problems due to `filter`, `lag`, and `group_rows`.

```
tidyverse_conflicts()
```

```
## -- Conflicts -------------------------------------------------- tidyverse_conflicts() --
## x stats::filter()          masks dplyr::filter()
## x kableExtra::group_rows() masks dplyr::group_rows()
## x stats::lag()             masks dplyr::lag()
## x car::recode()            masks dplyr::recode()
## x car::some()              masks purrr::some()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become er
```

For example, the code below fails:

```
library(stats)
library(dplyr)
as_tibble(mtcars, rownames = "car") %>% filter(car == "Valiant")

# Error message
# > as_tibble(mtcars, rownames = "car") %>% filter(car == "Valiant")
# Error: object 'car' not found
```

The code below works, because we explicitly write `dplyr::filter`:

```
library(stats)
library(dplyr)
print(as_tibble(mtcars, rownames = "car") %>% dplyr::filter(car == "Valiant"))
```

To deal with this, we use the conflicted::conflict_prefer to resolve this issue. Now we can run the `filter` function safely, knowing that it is the `dplyr::filter` function will be used.

```
library(conflicted)
conflict_prefer("filter", "dplyr", "stats")
library(stats)
library(dplyr)
print(as_tibble(mtcars, rownames = "car") %>% filter(car == "Valiant"))

# > conflict_prefer("filter", "dplyr", "stats")
# [conflicted] Will prefer dplyr::filter over stats::filter.
# > library(stats)
# > library(dplyr)
# > print(as_tibble(mtcars, rownames = "car") %>% filter(car == "Valiant"))
# # A tibble: 1 × 12
#   car       mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
#   <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
# 1 Valiant  18.1     6   225   105  2.76  3.46  20.2     1     0     3     1
```

12.2 Files In and Out

12.2.1 File Path

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

12.2.1.1 Check if File or Directory Exists on Computer

Take different string-based actions if some path exists on a computer.

```r
# Check if either of the path exists on the computer
spt_root_computer_a <- 'G:/repos/R4Econ/'
spt_root_computer_b <- 'C:/Users/fan/R4Econ/'

# Checking if directory exists
if (dir.exists(spt_root_computer_a)) {
  print(paste(spt_root_computer_a, 'exists'))
} else if (dir.exists(spt_root_computer_b)) {
  print(paste(spt_root_computer_b, 'exists', spt_root_computer_a, 'does not exist'))
} else {
  print(paste(spt_root_computer_b, spt_root_computer_a, 'both do does not exist'))
}
```

```
## [1] "C:/Users/fan/R4Econ/ exists G:/repos/R4Econ/ does not exist"
```

Now, we check if a particular file exists.

```r
# Check if either of the files exists on the computer
spn_root_computer_a <- 'G:/repos/R4Econ/index.Rmd'
spn_root_computer_b <- 'C:/Users/fan/R4Econ/index.Rmd'

# Checking if file exists
if (file.exists(spn_root_computer_a)) {
  print(paste(spn_root_computer_a, 'exists'))
} else if (file.exists(spn_root_computer_b)) {
  print(paste(spn_root_computer_b, 'exists', spn_root_computer_a, 'does not exist'))
} else {
  print(paste(spn_root_computer_b, spn_root_computer_a, 'both do does not exist'))
}
```

```
## [1] "C:/Users/fan/R4Econ/index.Rmd exists G:/repos/R4Econ/index.Rmd does not exist"
```

12.2.1.2 Compose a Path

File Path might contain information related to the file, decompose the file path, keep the final N folder names, to be possibled stored as a variable in the datafile stored inside.

```r
# Compose together a path
print(paste0('.Platform$file.sep=', .Platform$file.sep))
```

```
## [1] ".Platform$file.sep=/"
```

```r
spn_file_path = file.path("C:", "Users", "fan", "R4Econ", "amto", "tibble",
                          "fs_tib_basics.Rmd",
                          fsep = .Platform$file.sep)
# print
print(spn_file_path)
```

```
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd"
```

12.2.1.3 Substring and File Name

From path, get file name without suffix.

- r string split
- r list last element
- r get file name from path
- r get file path no name

```
st_example <- 'C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd'
st_file_wth_suffix_s <- tail(strsplit(st_example, "/")[[1]],n=1)
st_file_wno_suffix_s <- tools::file_path_sans_ext(basename(st_example))
st_fullpath_nosufx_s <- sub('\\.Rmd$', '', st_example)
st_fullpath_noname_s <- dirname(st_example)

print(paste0('st_file_wth_suffix_s:', st_file_wth_suffix_s))
```

```
## [1] "st_file_wth_suffix_s:fs_tib_basics.Rmd"
```

```
print(paste0('st_file_wno_suffix_s:', st_file_wno_suffix_s))
```

```
## [1] "st_file_wno_suffix_s:fs_tib_basics"
```

```
print(paste0('st_fullpath_nosufx_s:', st_fullpath_nosufx_s))
```

```
## [1] "st_fullpath_nosufx_s:C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics"
```

```
print(paste0('st_fullpath_noname_s:', st_fullpath_noname_s))
```

```
## [1] "st_fullpath_noname_s:C:/Users/fan/R4Econ/amto/tibble"
```

12.2.1.4 Get Subset of Path Folder Names

File Path might contain information related to the file, decompose the file path, keep the final N folder names, to be possibled stored as a variable in the datafile stored inside.

```
# Compose together a path
spn_example <- 'C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd'
# Replace default system slash, assume spn was generated by system.
ls_srt_folders_file <- strsplit(st_example, .Platform$file.sep)[[1]]
# Keep the last N layers
it_folders_names_to_keep = 2
snm_file_name <- tail(strsplit(st_example, "/")[[1]],n=1)
ls_srt_folders_keep <- head(tail(ls_srt_folders_file, it_folders_names_to_keep+1),
                            it_folders_names_to_keep)
# Show folder names
print(paste0('snm_file_name:', snm_file_name))
```

```
## [1] "snm_file_name:fs_tib_basics.Rmd"
```

```
print(paste0('last ', it_folders_names_to_keep, ' folders:'))
```

```
## [1] "last 2 folders:"
```

```
print(ls_srt_folders_keep)
```

```
## [1] "amto"   "tibble"
```

Shorter lines, to make copying easier.

```
# inputs
it_folders_names_to_keep = 2
spn_example <- 'C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd'
# copy these
ls_srt_folders_name_keep <- tail(strsplit(st_example, "/")[[1]], n=it_folders_names_to_keep+1)
snm_file_name <- tail(ls_srt_folders_name_keep, 1)
```

```r
ls_srt_folders_keep <- head(ls_srt_folders_name_keep, it_folders_names_to_keep)
# print
print(paste0('snm_file_name:', snm_file_name))
```

```
## [1] "snm_file_name:fs_tib_basics.Rmd"
```

```r
print(paste0('last ', it_folders_names_to_keep, ' folders:'))
```

```
## [1] "last 2 folders:"
```

```r
print(ls_srt_folders_keep)
```

```
## [1] "amto"   "tibble"
```

12.2.2 Text to File

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

12.2.2.1 Latex Table to File

Tabular outputs, text outputs, etc are saved as variables, which could be printed in console. They can also be saved to file for future re-used. For example, latex outputs need to be saved to file.

```r
# Load Data
dt <- mtcars[1:4, 1:6]
# Generate latex string variable
st_out_tex <- kable(dt, "latex")
print(st_out_tex)
# File out
# fileConn <- file("./../../_file/tex/tex_sample_a_tab.tex")
fileConn <- file("_file/tex/tex_sample_a_tab.tex")
writeLines(st_out_tex, fileConn)
close(fileConn)
```

12.2.2.2 Create a Text File from Strings

```r
st_file <- "\\documentclass[12pt,english]{article}
\\usepackage[bottom]{footmisc}
\\usepackage[urlcolor=blue]{hyperref}
\\begin{document}
\\title{A Latex Testing File}
\\author{\\href{http://fanwangecon.github.io/}{Fan Wang} \\thanks{See information \\href{https://fan
\\maketitle
Ipsum information dolor sit amet, consectetur adipiscing elit. Integer Latex placerat nunc orci.
\\paragraph{\\href{https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3140132}{Data}}
Village closure information is taken from a village head survey.\\footnote{Generally students went t
output:
  pdf_document:
    pandoc_args: '..//..//_output_kniti_pdf.yaml'
    includes:
      in_header: '..//..//preamble.tex'
  html_document:
    pandoc_args: '..//..//_output_kniti_html.yaml'
    includes:
      in_header: '..//..//hdga.html'
\\end{document}"

print(st_file)
```

```
## [1] "\\documentclass[12pt,english]{article}\n\\usepackage[bottom]{footmisc}\n\\usepackage[urlcolo
```

```r
fl_test_tex <- "_file/tex/test_fan.tex"
fileConn <- file(fl_test_tex)
writeLines(st_file, fileConn)
close(fileConn)
```

12.2.2.3 Open A File and Read Lines

Open and Replace Text in File:

```r
fileConn <- file(fl_test_tex, "r")
st_file_read <- readLines(fileConn)
print(st_file_read)
```

```
##  [1] "\\documentclass[12pt,english]{article}"
##  [2] "\\usepackage[bottom]{footmisc}"
##  [3] "\\usepackage[urlcolor=blue]{hyperref}"
##  [4] "\\begin{document}"
##  [5] "\\title{A Latex Testing File}"
##  [6] "\\author{\\href{http://fanwangecon.github.io/}{Fan Wang} \\thanks{See information \\href{ht
##  [7] "\\maketitle"
##  [8] "Ipsum information dolor sit amet, consectetur adipiscing elit. Integer Latex placerat nunc
##  [9] "\\paragraph{\\href{https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3140132}{Data}}"
## [10] "Village closure information is taken from a village head survey.\\footnote{Generally studen
## [11] "output:"
## [12] "  pdf_document:"
## [13] "    pandoc_args: '..//..//_output_kniti_pdf.yaml'"
## [14] "    includes:"
## [15] "      in_header: '..//..//preamble.tex'"
## [16] "  html_document:"
## [17] "    pandoc_args: '..//..//_output_kniti_html.yaml'"
## [18] "    includes:"
## [19] "      in_header: '..//..//hdga.html'"
## [20] "\\end{document}"
```

```r
close(fileConn)
```

12.2.2.4 Open a File and Replace Some Lines

Append additional strings into the file after *html_document* with proper spacings:

```r
# Read in Lines from existing file
fileConn <- file(fl_test_tex, "r")
st_file_read <- readLines(fileConn)
close(fileConn)

# Search and Replace String
st_search <- "html_document:"
st_replace <- paste0("html_document:\r\n",
                     "    toc: true\r\n",
                     "    number_sections: true\r\n",
                     "    toc_float:\r\n",
                     "      collapsed: false\r\n",
                     "      smooth_scroll: false\r\n",
                     "      toc_depth: 3")

# Search and Replace
st_file_updated <- gsub(x = st_file_read,
                        pattern = st_search,
                        replacement = st_replace)
```

```
# Print
print(st_file_updated)
```

```
##  [1] "\\documentclass[12pt,english]{article}"
##  [2] "\\usepackage[bottom]{footmisc}"
##  [3] "\\usepackage[urlcolor=blue]{hyperref}"
##  [4] "\\begin{document}"
##  [5] "\\title{A Latex Testing File}"
##  [6] "\\author{\\href{http://fanwangecon.github.io/}{Fan Wang} \\thanks{See information \\href{ht
##  [7] "\\maketitle"
##  [8] "Ipsum information dolor sit amet, consectetur adipiscing elit. Integer Latex placerat nunc
##  [9] "\\paragraph{\\href{https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3140132}{Data}}"
## [10] "Village closure information is taken from a village head survey.\\footnote{Generally studen
## [11] "output:"
## [12] "  pdf_document:"
## [13] "    pandoc_args: '..//..//_output_kniti_pdf.yaml'"
## [14] "    includes:"
## [15] "      in_header: '..//..//preamble.tex'"
## [16] "  html_document:\r\n    toc: true\r\n    number_sections: true\r\n    toc_float:\r\n      c
## [17] "    pandoc_args: '..//..//_output_kniti_html.yaml'"
## [18] "    includes:"
## [19] "      in_header: '..//..//hdga.html'"
## [20] "\\end{document}"
```

```
# Save Updated File
fl_srcrep_tex <- "_file/tex/test_fan_search_replace.tex"
fileConn_sr <- file(fl_srcrep_tex)
writeLines(st_file_updated, fileConn_sr)
close(fileConn_sr)
```

12.2.3 Rmd to HTML

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

12.2.3.1 Search and Find all Files in Repository

Search inside directories, for all files in a repository that have a particular suffix and that don't contain skip pattern list string items.

```
# Serch Folder and skip list
spt_roots <- c('C:/Users/fan/R4Econ/amto', 'C:/Users/fan/R4Econ/summarize')
spn_skip <- c('summarize', 'panel', 'support')

# Search and get all Path
ls_sfls <- list.files(path=spt_roots, recursive=T, pattern=".Rmd", full.names=T)

# Skip path if contains words in skip list
if(!missing(spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(spn_skip, collapse = "|"), ls_sfls)]
}

# Loop and print
for (spt_file in ls_sfls) {
    st_fullpath_nosufx <- tail(strsplit(spt_file, "/")[[1]],n=1)
    print(paste0(spt_file, '---', st_fullpath_nosufx))
}
```

```
## [1] "C:/Users/fan/R4Econ/amto/array/fs_ary_basics.Rmd---fs_ary_basics.Rmd"
```

```
## [1] "C:/Users/fan/R4Econ/amto/array/fs_ary_generate.Rmd---fs_ary_generate.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/array/fs_ary_mesh.Rmd---fs_ary_mesh.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/array/fs_ary_string.Rmd---fs_ary_string.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/array/main.Rmd---main.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/list/fs_lst_basics.Rmd---fs_lst_basics.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/list/main.Rmd---main.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/main.Rmd---main.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/matrix/fs_mat_demo_trans.Rmd---fs_mat_demo_trans.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/matrix/fs_mat_generate.Rmd---fs_mat_generate.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/matrix/fs_mat_linear_algebra.Rmd---fs_mat_linear_algebra.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/matrix/main.Rmd---main.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/misc/fs_parse_regex.Rmd---fs_parse_regex.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/misc/main.Rmd---main.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd---fs_tib_basics.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_factors.Rmd---fs_tib_factors.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_na.Rmd---fs_tib_na.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_random_draws.Rmd---fs_tib_random_draws.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_string.Rmd---fs_tib_string.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/main.Rmd---main.Rmd"
```

12.2.3.2 Search and Find all Git Modified or New Rmd

Search inside directories, for all files in a git repo folder that are new or have been modified. Ignore possible subset of file based on string search.

```r
# Serch Folder and skip list
spt_roots <- c('C:/Users/fan/R4Econ/amto', 'C:/Users/fan/R4Econ/development')
spn_skip <- c('summarize', 'panel', 'support')
ls_sfls <- list.files(path=spt_roots, recursive=T, pattern=".Rmd", full.names=T)
if(!missing(spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(spn_skip, collapse = "|"), ls_sfls)]
}

# Loop and print
for (spt_file in ls_sfls) {
  spg_check_git_status <- paste0('git status -s ', spt_file)
  st_git_status <- toString(system(spg_check_git_status, intern=TRUE))
  bl_modified <- grepl(' M ', st_git_status, fixed=TRUE)
  bl_anewfile <- grepl('?? ', st_git_status, fixed=TRUE)
  bl_nochange <- (st_git_status == "")

  if (bl_modified == 1) {
    print(paste0('MODIFIED: ', spt_file))
  } else if (bl_anewfile == 1) {
    print(paste0('A NEW FL: ', spt_file))
  } else {
    print(paste0('NO CHNGE: ', spt_file))
  }
}
```

```
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/array/fs_ary_basics.Rmd"
## [1] "MODIFIED: C:/Users/fan/R4Econ/amto/array/fs_ary_generate.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/array/fs_ary_mesh.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/array/fs_ary_string.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/array/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/list/fs_lst_basics.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/list/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/matrix/fs_mat_demo_trans.Rmd"
```

```
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/matrix/fs_mat_generate.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/matrix/fs_mat_linear_algebra.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/matrix/main.Rmd"
## [1] "MODIFIED: C:/Users/fan/R4Econ/amto/misc/fs_parse_regex.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/misc/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_factors.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_na.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_random_draws.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_string.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/_file/rmd/fs_rmd_pdf_html_mod.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/_file/rmd/fs_rmd_pdf_html_mod_mod.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/_file/rmd/fs_text_save_mod.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/_file/rmd/main_mod.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/fs_path.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/fs_rmd_pdf_html.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/fs_text_save.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/install/fs_install_R.Rmd"
## [1] "A NEW FL: C:/Users/fan/R4Econ/development/install/fs_packages_R.Rmd"
## [1] "MODIFIED: C:/Users/fan/R4Econ/development/install/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/main.Rmd"
## [1] "MODIFIED: C:/Users/fan/R4Econ/development/parallel/fs_parallel.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/parallel/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/python/fs_python_reticulate.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/python/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/system/fs_system_shell.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/system/main.Rmd"
```

12.2.3.3 Resave an Existing File with Different Name Different Folder

Given an existing Rmd File, Resave it with a different name (add to name suffix), and then save in a different folder:

- old file: */R4Econ/development/fs_rmd_pdf_html.Rmd*
- new file: *R4Econ/development/inout/_file/rmd/fs_rmd_pdf_html_mod.Rmd*

```
# Serch Folder and skip list
spt_roots <- c('C:/Users/fan/R4Econ/development/inout/')
spn_skip <- c('_main', '_file')
ls_sfls <- list.files(path=spt_roots, recursive=T, pattern=".Rmd", full.names=T)
if(!missing(spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(spn_skip, collapse = "|"), ls_sfls)]
}

# Loop and print
for (spt_file in ls_sfls) {
  spt_new <- paste0('_file/rmd/')
  spn_new <- paste0(spt_new, sub('\\.Rmd$', '', basename(spt_file)), '_mod.Rmd')
  print(spt_new)
  print(spn_new)

  fileConn_rd <- file(spt_file, "r")
  st_file_read <- readLines(fileConn_rd)

  fileConn_sr <- file(spn_new)
  writeLines(st_file_read, fileConn_sr)
```

```
  close(fileConn_rd)
  close(fileConn_sr)
}
```

12.2.3.3.1 Replacment Function Change Markdown Hierarchy and Add to YAML

Given an existing Rmd File, Resave it with a different name, and replace (add in) additional yaml contents.

```
spn_file = '_file/rmd/fs_rmd_pdf_html_mod.Rmd'
fileConn_sr <- file(spn_file)
st_file <- readLines(fileConn_sr)
# print(st_file)

st_search <- "html_document:
    toc: true
    number_sections: true
    toc_float:
      collapsed: false
      smooth_scroll: false
      toc_depth: 3"
st_replace <- paste0("html_document:
    toc: true
    number_sections: true
    toc_float:
      collapsed: false
      smooth_scroll: false
      toc_depth: 3\n",
                     "    toc: true\n",
                     "    number_sections: true\n",
                     "    toc_float:\n",
                     "      collapsed: false\n",
                     "      smooth_scroll: false\n",
                     "      toc_depth: 3")
st_file_updated <- gsub(x = st_file,
                        pattern = st_search,
                        replacement = st_replace)

st_search <- "../../"
st_replace <- paste0("../../../../")
st_file_updated <- gsub(x = st_file_updated,
                        pattern = st_search,
                        replacement = st_replace)

st_file_updated <- gsub(x = st_file_updated, pattern = '# ', replacement = '# ')
st_file_updated <- gsub(x = st_file_updated, pattern = '## ', replacement = '## ')
st_file_updated <- gsub(x = st_file_updated, pattern = '### ', replacement = '# ')

spn_file = '_file/rmd/fs_rmd_pdf_html_mod.Rmd'
fileConn_sr <- file(spn_file)
st_file <- writeLines(st_file_updated, fileConn_sr)
```

12.2.3.4 Search and Render Rmd File and Save HTML, PDF or R

1. Search files satisfying conditions in a folder
2. knit files to HTML (and re-run the contents of the file)
3. Save output to a different folder

```
# Specify Parameters
ar_spt_root = c('C:/Users/fan/R4Econ/amto/array/', 'C:/Users/fan/R4Econ/math/integration')
```

```
bl_recursive = TRUE
st_rmd_suffix_pattern = "*.Rmd"
ar_spn_skip <- c('basics', 'integrate', 'main', 'mesh')
ls_bool_convert <- list(bl_pdf=TRUE, bl_html=TRUE, bl_R=TRUE)
spt_out_directory <- 'C:/Users/fan/Downloads/_data'
bl_verbose <- TRUE

# Get Path
ls_sfls  <- list.files(path=ar_spt_root,
                       recursive=bl_recursive,
                       pattern=st_rmd_suffix_pattern,
                       full.names=T)

# Exclude Some Files given ar_spn_skip
if(!missing(ar_spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(ar_spn_skip, collapse = "|"), ls_sfls)]
}

# Loop over files
for (spn_file in ls_sfls) {

  # Parse File Name
  spt_file <- dirname(spn_file)
  sna_file <- tools::file_path_sans_ext(basename(spn_file))

  # Output FIles
  spn_file_pdf <- paste0(spt_file, sna_file, '.pdf')
  spn_file_html <- paste0(spt_file, sna_file, '.html')
  spn_file_R <- paste0(spt_file, sna_file, '.R')

  # render to PDF
  if (ls_bool_convert$bl_pdf) {
    if (bl_verbose) message(paste0('spn_file_pdf:',spn_file_pdf, ', PDF started'))
    rmarkdown::render(spn_file, output_format='pdf_document',
                      output_dir = spt_out_directory, output_file = sna_file)
    if (bl_verbose) message(paste0('spn_file_pdf:',spn_file_pdf, ', PDF finished'))
    spn_pdf_generated <- paste0(spt_out_directory, '/', spn_file_pdf)
  }

  # render to HTML
  if (ls_bool_convert$bl_html) {
    if (bl_verbose) message(paste0('spth_html:',spn_file_html, ', HTML started.'))
    rmarkdown::render(spn_file, output_format='html_document',
                      output_dir = spt_out_directory, output_file = sna_file)
    if (bl_verbose) message(paste0('spth_html:',spn_file_html, ', HTML finished.'))
    spn_html_generated <- paste0(spt_out_directory, '/', spn_file_html)
  }

  # purl to R
  if (ls_bool_convert$bl_R) {
    if (bl_verbose) message(paste0('purl_to:', paste0(spn_file_R, ".R")))
    knitr::purl(spn_file, output=paste0(spt_out_directory, '/', sna_file, '.R'), documentation = 1)
    spn_R_generated <- paste0(spt_out_directory, '/', sna_file, '.R')
  }

  # return(list(ls_spt_pdf_generated=ls_spt_pdf_generated,
  #             ls_spt_html_generated=ls_spt_html_generated,
  #             ls_spt_R_generated=ls_spt_R_generated))
```

```
}
```

12.3 Python with R

12.3.1 Reticulate Basics

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

12.3.1.1 Basic Python Tests with RMD

Could specify: *python, engine.path = "C:/ProgramData/Anaconda3/envs/wk_pyfan/python.exe"*, this is already set inside Rprofile: *knitr::opts_chunk$set(engine.path = "C:/ProgramData/Anaconda3/envs/wk_pyfan/python.e*

```
1+1
```

```
## 2
```

12.3.1.2 Install and Python Path

Install reticulate from github directly to get latest version: *devtools::install_github("rstudio/reticulate")*

Check python version on computer:

```
Sys.which('python')
```

```
##                                                python
## "C:\\PROGRA~3\\ANACON~1\\envs\\wk_pyfan\\python.exe"
```

After installing reticulate, load in the library: library(reticulate). With "py_config()" to see python config. First time, might generate "No non-system installation of Python could be found." and ask if want to install Miniconda. Answer NO.

Correct outputs upon checking py_config():

```
python:         C:/ProgramData/Anaconda3/python.exe
libpython:      C:/ProgramData/Anaconda3/python37.dll
pythonhome:     C:/ProgramData/Anaconda3
version:        3.7.9 (default, Aug 31 2020, 17:10:11) [MSC v.1916 64 bit (AMD64)]
Architecture:   64bit
numpy:          C:/ProgramData/Anaconda3/Lib/site-packages/numpy
numpy_version:  1.19.1

python versions found:
 C:/ProgramData/Anaconda3/python.exe
 C:/ProgramData/Anaconda3/envs/wk_cgefi/python.exe
 C:/ProgramData/Anaconda3/envs/wk_jinja/python.exe
 C:/ProgramData/Anaconda3/envs/wk_pyfan/python.exe
```

Set which python to use:

```
# Sys.setenv(RETICULATE_PYTHON = "C:/programdata/Anaconda3/python.exe")
# Sys.setenv(RETICULATE_PYTHON = "C:/ProgramData/Anaconda3/envs/wk_pyfan/python.exe")
library(reticulate)
# What is the python config
py_config()
```

```
## python:         C:/ProgramData/anaconda3/envs/wk_pyfan/python.exe
## libpython:      C:/ProgramData/anaconda3/envs/wk_pyfan/python311.dll
## pythonhome:     C:/ProgramData/anaconda3/envs/wk_pyfan
## version:        3.11.7 | packaged by conda-forge | (main, Dec 23 2023, 14:27:59) [MSC v.1937 64 b
```

```
## Architecture:   64bit
## numpy:          C:/ProgramData/anaconda3/envs/wk_pyfan/Lib/site-packages/numpy
## numpy_version:  1.26.2
##
## NOTE: Python version was forced by RETICULATE_PYTHON
```

```
# set python
# use_python("C:/programdata/Anaconda3/python.exe")
# use_python("C:/ProgramData/Anaconda3/envs/wk_pyfan/python.exe")
use_condaenv('wk_pyfan')
# Sys.which('python')
py_run_string('print(1+1)')
```

```
## 2
```

12.3.1.3 Error

12.3.1.3.1 py_call_impl error The error appeared when calling any python operations, including "1+1", resolved after installing reticulate from github: *devtools::install_github("rstudio/reticulate")*

```
Error in py_call_impl(callable, dots$args, dots$keywords) :
  TypeError: use() got an unexpected keyword argument 'warn'
```

12.4 Command Line

12.4.1 Shell and System Commands

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

12.4.1.1 Basic Shell Commands

Run basic shell commands in windows:

```
# detect current path
print(toString(shell(paste0("echo %cd%"), intern=TRUE)))
```

```
## [1] "C:\\Users\\fan\\R4Econ"
```

```
# Show directory
print(toString(shell(paste0("dir"), intern=TRUE)))
```

```
## [1] " Volume in drive C is OS,  Volume Serial Number is 123A-85A2, ,  Directory of C:\\Users\\fan
```

12.4.1.2 Run Python Inside a Conda Environment

Use shell rather than system to activate a conda environment, check python version:

```
# activate conda env
print(toString(shell(paste0("activate base & python --version"), intern=TRUE)))
```

```
## [1] ", C:\\Users\\fan\\R4Econ>conda.bat activate base , Python 3.11.5"
```

Activate conda env and run a line:

```
spg_runpython <- paste0("activate base &",
                        "python --version &",
                        "python -c ",
                        "\"st_var='this is string var';",
                        "print(f'{st_var}');",
                        "\"")
print(toString(shell(spg_runpython, intern=TRUE)))
```

```
## [1] ", C:\\Users\\fan\\R4Econ>conda.bat activate base , Python 3.11.5, this is string var"
```

12.5 Run Code in Parallel in R

12.5.1 Parallel Loop in R

Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

12.5.1.1 Setting Up and First Run

First, install several packages.

```
install.packages(c("parallel", "doParallel", "foreach"))
```

Second, we load the libraries, and check on the parallel processing capacities on the local machine.

```
# Load libraries
library(dplyr)
library(readr)
library(tibble)
library(iterators)
library(parallel)
library(foreach)
library(doParallel)

# Check number of cores
it_n_cores_computer <- parallel::detectCores()
glue::glue("Number of cores on computers:{it_n_cores_computer}")

# OUTPUT
## Number of cores on computers:20
```

Third, we might want to use less than the total number of cores available. Specifying the number of cores to be used, we can initiate a local cluster. "PSOCK" below copies everything to each worker.

```
# Start cluster
ob_cluster <- parallel::makeCluster(
  it_n_cores_computer - 2,
  type = "PSOCK"
  )
# Register cluster
doParallel::registerDoParallel(cl = ob_cluster)
```

Fourth, run first parallel task, concurrent base-10 exponentiation.

```
# c(a,b,c,d) outputs together with combine
ar_test_parallel <- foreach(
  it_power = seq(1, 10), .combine = 'c'
) %dopar% {
  return(10^(it_power))
}
glue::glue("dopar outputs: {ar_test_parallel}")

# Output
## dopar outputs: 10
## dopar outputs: 100
## dopar outputs: 1000
## dopar outputs: 10000
## dopar outputs: 1e+05
## dopar outputs: 1e+06
## dopar outputs: 1e+07
## dopar outputs: 1e+08
```

```
## dopar outputs: 1e+09
## dopar outputs: 1e+10
```

Fifth, close cluster. When work is done, close the cluster.

```
parallel::stopCluster(cl = ob_cluster)
```

12.5.1.2 Parallel Function Run with Different Parameters, Aggregate Output Files

In this example, we create a function, we run the function with different parameters, each time generating a data output file to be stored, and then review results after.

First, we create a function. In this function, we generate a random matrix, the `it_nrow` parameter controls the number of rows in this random matrix. We store this matrix as csv.

Note, for each function used, such as `as_tibble` below, we should write it as `tibble::as_tibble`, to declare package and function jointly.

```
ffi_rand2csv <- function(
    spt_path_out,
    it_nrow = 3,
    st_file_prefix = "prefix") {

    # Generate a matrix and tibble
    mt_rnorm_a <- matrix(
      rnorm(it_nrow*3, mean=0, sd=1),
      nrow=it_nrow, ncol=3)
    tb_test <- tibble::as_tibble(mt_rnorm_a)

    # File output path
    spn_output_file <- file.path(
      spt_path_out,
      paste0(st_file_prefix, '_nrow', it_nrow, '.csv'),
      fsep = .Platform$file.sep)

    # Write file out
    readr::write_csv(tb_test, spn_output_file)
    print(glue::glue(
      "File saved successfully: ", spn_output_file))

}
```

Second, we initialize the cluster.

```
# Get the number of cores
it_n_cores_computer <- parallel::detectCores()
glue::glue("Number of cores on computers:{it_n_cores_computer}")
# Start cluster
ob_cluster <- parallel::makeCluster(
  it_n_cores_computer - 2,
  type = "PSOCK"
  )
# Register cluster
doParallel::registerDoParallel(cl = ob_cluster)

# OUTPUT
## Number of cores on computers:20
```

Third, we run the function in parallel.

```
# Define shared Path

spt_root <- "C:/Users/fan/"
spt_rmd <- "R4Econ/development/parallel/_file/"
spt_path_out <- file.path(spt_root, spt_rmd, fsep = .Platform$file.sep)

# Parallel Run
foreach(
  it_nrow = seq(2, 4)
) %dopar% {
  # Run function
  ffi_rand2csv(
    spt_path_out,
    it_nrow = it_nrow,
    st_file_prefix = "ffi_para_test")
}

# Output
## [[1]]
## File saved successfully: C:/Users/fan//R4Econ/development/parallel/_file/ffi_para_test_nrow2.csv
##
## [[2]]
## File saved successfully: C:/Users/fan//R4Econ/development/parallel/_file/ffi_para_test_nrow3.csv
##
## [[3]]
## File saved successfully: C:/Users/fan//R4Econ/development/parallel/_file/ffi_para_test_nrow4.csv
```

Fourth, adapting the parallel loop to other functions. Note that:

1. In the `forach` loop ablow, we iterate over `seq(2,4)`, assigning in parallel `2`, `3`, and `4` to the parameter `it_nrow`.
2. `it_nrow` is a parameter for the `ffi_rand2csv` function, so we will generate different outputs associated with `it_nrow=2`, `it_nrow=3`, and `it_nrow=4`.
3. The code above can be adapted to other functions that one wants to run in parallel by changing only one parameter of a function. For example, suppose we want to run `ffp_demo_loc_env_inequality(spt_path_data, fl_temp_bound=fl_temp_bound)`, where `spt_path_data` is common across parallel calls, but we want to update `fl_temp_bound` for each parallel call, then we need to iterate over `fl_temp_bound`. See example below:

```
# Some path
spt_path_data <- "C:/Users/fan/"
# Parallel Run
foreach(
  fl_temp_bound = seq(-40, 40, by=1)
) %dopar% {
  # Run function
  ffp_demo_loc_env_inequality(
    spt_path_data,
    fl_temp_bound=fl_temp_bound)
}
```

Appendix A

Index and Code Links

A.1 Array, Matrix, Dataframe links

A.1.1 Section 1.1 List links

1. Multi-dimensional Named Lists: **rmd** | **r** | **pdf** | **html**
    - Initiate Empty List. Named one and two dimensional lists. List of Dataframes.
    - Collapse named and unamed list to string and print input code.
    - **r**: *deparse(substitute()) + vector(mode = "list", length = it_N) + names(list) <- paste0('e',seq()) + dimnames(ls2d)[[1]] <- paste0('r',seq()) + dimnames(ls2d)[[2]] <- paste0('c',seq())*
    - **tidyr**: *unnest()*

A.1.2 Section 1.2 Array links

1. Basic Arrays Operations in R: **rmd** | **r** | **pdf** | **html**
    - Generate N-dimensional array of NA values, label dimension elements.
    - Basic array operations in R, rep, head, tail, na, etc.
    - E notation.
    - Get N cuts from M points.
    - **r**: *sum() + prod() + rep() + array(NA, dim=c(3, 3)) + array(NA, dim=c(3, 3, 3)) + dimnames(mn)[[3]] = paste0('k=', 0:4) + head() + tail() + na_if() + Re()*
    - **purrr**: *reduce()*
2. Generate Special Arrays: **rmd** | **r** | **pdf** | **html**
    - Generate equi-distance, special log spaced array.
    - Generate probability mass function with non-unique and non-sorted value and probability arrays.
    - Generate a set of integer sequences, with gaps in between, e.g., (1,2,3), (5), (10,11).
    - **r**: *seq() + sort() + runif() + ceiling() + sample() + apply() + do.call()*
    - **stats**: *aggregate()*
3. String Operations: **rmd** | **r** | **pdf** | **html**
    - Split, concatenate, subset, replace, and substring strings.
    - Convert number to string without decimal and negative sign.
    - Concatenate numeric and string arrays as a single string.
    - Regular expression
    - **r**: *paste0() + paste0(round(runif(3),3), collapse=',') + sub() + gsub() + grepl() + sprintf()*
4. Meshgrid Matrices, Arrays and Scalars: **rmd** | **r** | **pdf** | **html**
    - Meshgrid Matrices, Arrays and Scalars to form all combination dataframe.
    - **tidyr**: *expand_grid() + expand.grid()*

A.1.3 Section 1.3 Matrix links

1. Matrix Basics: **rmd** | **r** | **pdf** | **html**

- Generate and combine NA, fixed and random matrixes. Name columns and rows.
- Sort all rows and all columns of a matrix.
- Replace values outside min and max in matrix by NA values.
- **R**: *rep() + rbind() + matrix(NA) + matrix(NA_real_) + matrix(NA_integer_) + colnames() + rownames() + t(apply(mt, 1, sort)) + apply(mt, 2, sort) + colMeans + rowMeans + which()*

2. Linear Algebra Operations: **rmd** | **r** | **pdf** | **html**

A.1.4 Section 1.4 Regular Expression, Date, etc. links

1. R String Regular Expression (Regex): **rmd** | **r** | **pdf** | **html**
   - Regular expression.
   - Find characters that that contain or not contain certain certain strings, numbers, and symbols.
   - **r**: *grepl()*

A.2 Manipulate and Summarize Dataframes links

A.2.1 Section 2.1 Variables in Dataframes links

1. Generate Tibble Dataframes from Matrix and List: **rmd** | **r** | **pdf** | **html**
   - Generate tibble data from two dimensional named lists, unlist for exporting.
   - Generate tibble dataframe, rename tibble variables, generate tibble row and column names.
   - Export tibble table to csv file with date and time stamp in file name.
   - Rename numeric sequential columns with string prefix and suffix.
   - **base**: *Sys.time() + format() + sample(LETTERS, 5, replace = TRUE) + is.list*
   - **dplyr**: *as_tibble(mt) + rename_all(~c(ar_names)) + rename_at(vars(starts_with("xx")), funs(str_replace(., "yy", "yyyy")) + rename_at(vars(num_range('',ar_it)), funs(paste0(st,.))) + rowid_to_column() + row_number() + min_rank() + dense_rank() + mutate_if()*
   - **base**: *colnames + rownames*
2. Interact and Cut Variables to Generate Categorical Variables: **rmd** | **r** | **pdf** | **html**
   - Convert rowname to variable name.
   - Generate categorical variable from a continuous variable.
   - Convert numeric variables to factor variables, generate interaction variables (joint factors), and label factors with descriptive words.
   - Graph MPG and 1/4 Miles Time (qsec) from the mtcars dataset over joint shift-type (am) and engine-type (vs) categories.
   - **r**: *cut(breaks = ar, values = ar, right = FALSE)*
   - **tibble**: *rownames_to_column()*
   - **forcats**: *as_factor() + fct_recode() + fct_cross()*
3. Randomly Draw Subsets of Rows from Matrix: **rmd** | **r** | **pdf** | **html**
   - Given matrix, randomly sample rows, or select if random value is below threshold.
   - **r**: *rnorm() + sample() + df[sample(dim(df)[1], it_M, replace=FALSE),]*
   - **dplyr**: *case_when() + mutate(var = case_when(rnorm(n(),mean=0,sd=1) < 0 ~ 1, TRUE ~ 0)) %>% filter(var == 1)*
4. Generate Variables Conditional on Other Variables, Categorical from Continuous: **rmd** | **r** | **pdf** | **html**
   - Use case_when to generate elseif conditional variables: NA, approximate difference, etc.
   - Generate Categorical Variables from Continuous Variables.
   - **dplyr**: *case_when() + na_if() + mutate(var = na_if(case_when(rnorm(n())< 0 ~ -99, TRUE ~ mpg), -99))*
   - **r**: *e-notation + all.equal() + isTRUE(all.equal(a,b,tol)) + is.na() + NA_real_ + NA_character_ + NA_integer_*
5. R Tibble Dataframe String Manipulations: **rmd** | **r** | **pdf** | **html**
   - There are multiple CEV files, each containing the same file structure but simulated
   - with different parameters, gather a subset of columns from different files, and provide
   - with correct attributes based on CSV file names.
   - **r**: *cbind(ls_st, ls_st) + as_tibble(mt_st)*

A.2.2 Section 2.2 Counting Observation links

1. R Example Counting, Tabulation, and Cross Tabulation: **rmd** | **r** | **pdf** | **html**
    - Uncount to generate panel skeleton from years in survey
    - **dplyr**: *tally() + spread() + distinct() + uncount(yr_n) + group_by() + mutate(yr = row_number() + start_yr)*

A.2.3 Section 2.3 Sorting, Indexing, Slicing links

1. Sorted Index, Interval Index and Expand Value from One Row: **rmd** | **r** | **pdf** | **html**
    - Sort and generate index for rows
    - Generate negative and positive index based on deviations
    - Populate Values from one row to other rows
    - **dplyr**: *arrange() + row_number() + mutate(lowest = min(Sepal.Length)) + case_when(row_number()==x ~ Septal.Length) + mutate(Sepal.New = Sepal.Length[Sepal.Index == 1])*
2. R Within-group Ascending and Descending Sort, Selection, and Differencing: **rmd** | **r** | **pdf** | **html**
    - Sort a dataframe by multiple variables, some in descending order.
    - Select observations with the highest M values from within N groups (top scoring students from each class).
    - **dplyr**: *arrange(a, b, desc(c)) + group_by() + lag() + lead() + slice_head(n=1)*

A.2.4 Section 2.4 Advanced Group Aggregation links

1. Cummean Test, Cumulative Mean within Group: **rmd** | **r** | **pdf** | **html**
    - There is a dataframe with a grouping variable and some statistics sorted by another within group
    - variable, calculate the cumulative mean of that variable.
    - **dplyr**: *cummean() + group_by(id, isna = is.na(val)) + mutate(val_cummean = ifelse(isna, NA, cummean(val)))*
2. Count Unique Groups and Mean within Groups: **rmd** | **r** | **pdf** | **html**
    - Unique groups defined by multiple values and count obs within group.
    - Mean, sd, observation count for non-NA within unique groups.
    - **dplyr**: *group_by() + summarise(n()) + summarise_if(is.numeric, funs(mean = mean(., na.rm = TRUE), n = sum(is.na(.)==0)))*
3. By Groups, One Variable All Statistics: **rmd** | **r** | **pdf** | **html**
    - Pick stats, overall, and by multiple groups, stats as matrix or wide row with name=(ctsvar + catevar + catelabel).
    - **tidyr**: *group_by() + summarize_at(, funs()) + rename(!!var := !!sym(var)) + mutate(!!var := paste0(var,'str',!!!syms(vars))) + gather() + unite() + spread(varcates, value)*
4. By within Individual Groups Variables, Averages: **rmd** | **r** | **pdf** | **html**
    - By Multiple within Individual Groups Variables.
    - Averages for all numeric variables within all groups of all group variables. Long to Wide to very Wide.
    - **tidyr**: *gather() + group_by() + summarise_if(is.numeric, funs(mean(., na.rm = TRUE))) + mutate(all_m_cate = paste0(variable, '_c', value)) + unite() + spread()*

A.2.5 Section 2.5 Distributional Statistics links

1. Tibble Basics: **rmd** | **r** | **pdf** | **html**
    - input multiple variables with comma separated text strings
    - quantitative/continuous and categorical/discrete variables
    - histogram and summary statistics
    - **tibble**: *ar_one <- c(107.72,101.28) + ar_two <- c(101.72,101.28) + mt_data <- cbind(ar_one, ar_two) + as_tibble(mt_data)*

A.2.6 Section 2.6 Summarize Multiple Variables links

1. Apply the Same Function over Columns and Row Groups: **rmd** | **r** | **pdf** | **html**
    - Compute row-specific quantiles, based on values across columns within each row.

- Sum values within-row across multiple columns, ignoring NA.
- Sum values within-group across multiple rows for matched columns, ignoring NA.
- Replace NA values in selected columns by alternative values.
- **r**: *rowSums() + cumsum() + gsub() + mutate_at(vars(matches()), .funs = list(gs = ~sum(.))) + mutate_at(vars(contains()), .funs = list(cumu = ~cumsum(.))) + rename_at(vars(contains()), list(~gsub("M", " ", .)))*
- **dplyr**: *group_by(across(one_of(ar_st_vars))) + mutate(across(matches(), func) + rename_at() + mutate_at() + rename_at(vars(starts_with()), funs(str_replace(., "v", "var"))) + mutate_at(vars(one_of()), list(~replace_na(., 99)))*
- **purrr**: *reduce()*

A.3 Functions links

A.3.1 Section 3.1 Dataframe Mutate links

1. Nonlinear Function of Scalars and Arrays over Rows: **rmd** | **r** | **pdf** | **html**
   - Five methods to evaluate scalar nonlinear function over matrix.
   - Evaluate non-linear function with scalar from rows and arrays as constants.
   - **r**: $.fl_A + fl_A =$*'(., 'fl_A') + .[[svr_fl_A]]*
   - **dplyr**: *rowwise() + mutate(out = funct(inputs))*
2. Evaluate Functions over Rows of Meshes Matrices: **rmd** | **r** | **pdf** | **html**
   - Mesh states and choices together and rowwise evaluate many matrixes.
   - Cumulative sum over multiple variables.
   - Rename various various with common prefix and suffix appended.
   - **r**: *ffi <- function(fl_A, ar_B)*
   - **tidyr**: *expand_grid() + rowwise() + df %>% rowwise() %>% mutate(var = ffi(fl_A, ar_B))*
   - **ggplot2**: *geom_line() + facet_wrap() + geom_hline() + facet_wrap(. ~ var_id, scales = 'free') + geom_hline(yintercept=0, linetype="dashed", color="red", size=1) +*

A.3.2 Section 3.2 Dataframe Do Anything links

1. Dataframe Row to Array (Mx1 by N) to (MxQ by N+1): **rmd** | **r** | **pdf** | **html**
   - Generate row value specific arrays of varying Length, and stack expanded dataframe.
   - Given row-specific information, generate row-specific arrays that expand matrix.
   - **dplyr**: *do() + unnest() + left_join() + df %>% group_by(ID) %>% do(inc =* $rnorm(.Q, mean =$ *.mean, sd=.$sd)) %>% unnest(c(inc))*
2. Simulate country-specific wage draws and compute country wage GINIs: Dataframe (Mx1 by N) to (MxQ by N+1) to (Mx1 by N: **rmd** | **r** | **pdf** | **html**
   - Define attributes for M groups across N variables, simulate up to Q observations for each of the M Groups, then compute M-specific statistics based on the sample of observations within each M.
   - Start with a matrix that is (Mx1 by N); Expand this to (MxQ by N+1), where, the additional column contains the MxQ specific variable; Compute statistics for each M based on the Q observations with M, and then present (Mx1 by N+1) dataframe.
   - **dplyr**: *group_by(ID) + do(inc =* $rnorm(.N, mean =$ *.mn, sd=.$sd)) + unnest(c(inc)) + left_join(df, by="ID")*
3. Dataframe Subset to Dataframe (MxP by N) to (MxQ by N+Z-1): **rmd** | **r** | **pdf** | **html**
   - Group by mini dataframes as inputs for function. Stack output dataframes with group id.
   - **dplyr**: *group_by() + do() + unnest()*

A.3.3 Section 3.3 Apply and pmap links

1. Apply and Sapply function over arrays and rows: **rmd** | **r** | **pdf** | **html**
   - Evaluate function f(x_i,y_i,c), where c is a constant and x and y vary over each row of a matrix, with index i indicating rows.
   - Get same results using apply and sapply with defined and anonymous functions.
   - Convert list of list to table.

- **r**: *do.call() + as_tibble(do.call(rbind,ls)) + apply(mt, 1, func) + sapply(ls_ar, func, ar1, ar2)*

2. Mutate rowwise, mutate pmap, and rowwise do unnest: **rmd** | **r** | **pdf** | **html**
   - Evaluate function f(x_i,y_i,c), where c is a constant and x and y vary over each row of a matrix, with index i indicating rows.
   - Get same results using various types of mutate rowwise, mutate pmap and rowwise do unnest.
   - **dplyr**: *rowwise() + do() + unnest()*
   - **purrr**: *pmap(func)*
   - **tidyr**: *unlist()*

A.4 Multi-dimensional Data Structures links

A.4.1 Section 4.1 Generate, Gather, Bind and Join links

1. R dplyr Group by Index and Generate Panel Data Structure: **rmd** | **r** | **pdf** | **html**
   - Build skeleton panel frame with N observations and T periods with gender and height.
   - Generate group Index based on a list of grouping variables.

   - **r**: *runif() + rnorm() + rbinom(n(), 1, 0.5) + cumsum()*
   - **dplyr**: *group_by() + row_number() + ungroup() + one_of() + mutate(var = (row_number()==1)*1)*
   - **tidyr**: *uncount()*
2. R DPLYR Join Multiple Dataframes Together: **rmd** | **r** | **pdf** | **html**
   - Join dataframes together with one or multiple keys. Stack dataframes together.
   - **dplyr**: *filter() + rename(!!sym(vsta) := !!sym(vstb)) + mutate(var = rnom(n())) + left_join(df, by=(c('id'='id', 'vt'='vt'))) + left_join(df, by=setNames(c('id', 'vt'), c('id', 'vt'))) + bind_rows()*
3. R Gather Data Columns from Multiple CSV Files: **rmd** | **r** | **pdf** | **html**
   - There are multiple CEV files, each containing the same file structure but simulated
   - with different parameters, gather a subset of columns from different files, and provide
   - with correct attributes based on CSV file names.
   - Separate numeric and string components of a string variable value apart.
   - **r**: *file() + writeLines() + readLines() + close() + gsub() + read.csv() + do.call(bind_rows, ls_df) + apply()*
   - **tidyr**: *separate()*
   - **regex**: *(?<=[A-Za-z])(?=[-0-9])*

A.4.2 Section 4.2 Wide and Long links

1. Convert Table from Long to Wide with dplyr: **rmd** | **r** | **pdf** | **html**
   - Long attendance roster to wide roster and calculate cumulative attendance by each day for students.
   - Convert long roster with attendance and test-scores to wide.
   - **tidyr**: *pivot_wider(id_cols = c(v1), names_from = v2, names_prefix = "id", names_sep = "_", values_from = c(v3, v4))*
   - **dplyr**: *mutate(var = case_when(rnorm(n()) < 0 ~ 1, TRUE ~ 0)) + rename_at(vars(num_range('', ar_it)), list(~paste0(st_prefix, . , ''))) + mutate_at(vars(contains(str)), list(~replace_na(., 0))) + mutate_at(vars(contains(str)), list(~cumsum(.)))*
2. Convert Table from Wide to Long with dplyr: **rmd** | **r** | **pdf** | **html**
   - Given a matrix of values with row and column labels, create a table where the unit of observation are the row and column categories, and the values in the matrix is stored in a single variable.
   - Reshape wide to long two sets of variables, two categorical variables added to wide table.
   - **tidyr**: *pivot_longer(cols = starts_with('zi'), names_to = c('zi'), names_pattern = paste0("zi(.)"), values_to = "ev") + pivot_longer(cols = matches('a line b'), names_to = c('va', 'vb'), names_pattern = paste0("(.)_(.)"), values_to ="ev")*
   - **dplyr**: *left_join()*

A.4.3 Section 4.3 Within Panel Comparisons and Statistics links

1. Find Closest Values Along Grids: **rmd** | **r** | **pdf** | **html**
   - There is an array (matrix) of values, find the index of the values closest to another value.
   - **r**: *do.call(bind_rows, ls_df)*
   - **dplyr**: *left_join(tb, by=(c('vr_a'='vr_a', 'vr_b'='vr_b')))*
2. Cross-group Within-time and Cross-time Within-group Statistics: **rmd** | **r** | **pdf** | **html**
   - Compute relative values across countries at each time, and relative values within country across time.
   - **dplyr**: *arrange(v1, v2) %>% group_by(v1) %>% mutate(stats := v3/first(v3))*

A.4.4 Section 4.4 Join and Merge Files Together by Keys links

1. Mesh join: **rmd** | **r** | **pdf** | **html**
   - Full join, expand multiple-rows of data-frame with the same set of expansion rows and columns
   - **dplyr**: *full_join()*

A.5 Linear Regression links

A.5.1 Section 5.1 Linear and Polynomial Fitting links

1. Find Best Fit of Curves Through Points: **rmd** | **r** | **pdf** | **html**
   - There are three x and y points, find the quadratic curve that fits through them exactly.
   - There are N sets of x and y points, find the Mth order polynomial fit by regressing y on poly(x, M).
   - **stats**: *lm(y ~ poly(x, 2), dataset=df) + summary.lm(rs) + predict(rs)*
2. Fit a Time Series with Polynomial and Analytical Expressions for Coefficients: **rmd** | **r** | **pdf** | **html**
   - Given a time series of data points from a polynomial data generating process, solve for the polynomial coefficients.
   - Mth derivative of Mth order polynomial is time invariant, use functions of differences of differences of differences to identify polynomial coefficients analytically.
   - **R**: *matrix multiplication*

A.5.2 Section 5.2 OLS and IV links

1. IV/OLS Regression: **rmd** | **r** | **pdf** | **html**
   - R Instrumental Variables and Ordinary Least Square Regression store all Coefficients and Diagnostics as Dataframe Row.
   - **aer**: *library(aer) + ivreg(as.formula, diagnostics = TRUE)*
2. M Outcomes and N RHS Alternatives: **rmd** | **r** | **pdf** | **html**
   - There are M outcome variables and N alternative explanatory variables. Regress all M outcome variables on N endogenous/independent right hand side variables one by one, with controls and/or IVs, collect coefficients.
   - **dplyr**: *bind_rows(lapply(listx, function(x)(bind_rows(lapply(listy, regf.iv))) + starts_with() + ends_with() + reduce(full_join)*

A.5.3 Section 5.3 Decomposition links

1. Regression Decomposition: **rmd** | **r** | **pdf** | **html**
   - Post multiple regressions, fraction of outcome variables' variances explained by multiple subsets of right hand side variables.
   - **dplyr**: *gather() + group_by(var) + mutate_at(vars, funs(mean = mean(.))) + rowSums(mat*mat) + mutate_if(is.numeric, funs(frac = (./value_var)))*

A.6 Nonlinear and Other Regressions links

A.6.1 Section 6.1 Logit Regression links

1. Logit Regression: **rmd** | **r** | **pdf** | **html**
   - Logit regression testing and prediction.
   - **stats**: *glm(as.formula(), data, family='binomial') + predict(rs, newdata, type = "response")*
2. Estimate Logistic Choice Model with Aggregate Shares: **rmd** | **r** | **pdf** | **html**
   - Aggregate share logistic OLS with K worker types, T time periods and M occupations.
   - Estimate logistic choice model with aggregate shares, allowing for occupation-specific wages and occupation-specific intercepts.
   - Estimate allowing for K and M specific intercepts, K and M specific coefficients, and homogeneous coefficients.
   - Create input matrix data structures for logistic aggregate share estimation.
   - **stats**: *lm(y ~ . -1)*
3. Fit Prices Given Quantities Logistic Choice with Aggregate Data: **rmd** | **r** | **pdf** | **html**
   - A multinomial logistic choice problem generates choice probabilities across alternatives, find the prices that explain aggregate shares.
   - **stats**: *lm(y ~ . -1)*

A.6.2 Section 6.2 Quantile Regression links

1. Quantile Regressions with Quantreg: **rmd** | **r** | **pdf** | **html**
   - Quantile regression with continuous outcomes. Estimates and tests quantile coefficients.
   - **quantreg**: *rq(mpg ~ disp + hp + factor(am), tau = c(0.25, 0.50, 0.75), data = mtcars) + anova(rq(), test = "Wald", joint=TRUE) + anova(rq(), test = "Wald", joint=FALSE)*

A.7 Optimization links

A.7.1 Section 7.1 Grid Based Optimization links

1. Find the Maximizing or Minimizing Point Given Some Objective Function: **rmd** | **r** | **pdf** | **html**
   - Find the maximizing or minimizing point given some objective function.
   - **base**: *while + min + which.min + sapply*
2. Concurrent Bisection over Dataframe Rows: **rmd** | **r** | **pdf** | **html**
   - Post multiple regressions, fraction of outcome variables' variances explained by multiple subsets of right hand side variables.
   - **tidyr**: *pivot_longer(cols = starts_with('abc'), names_to = c('a', 'b'), names_pattern = paste0('prefix', "(.)_(.)"), values_to = val) + pivot_wider(names_from = !!sym(name), values_from = val) + mutate(!!sym(abc) := case_when(efg < 0 ~ !!sym(opq), TRUE ~ iso))*
   - **gglot2**: *geom_line() + facet_wrap() + geom_hline()*

A.8 Mathematics links

A.8.1 Section 8.1 Basics links

1. Analytical Formula Fit Curves Through Points: **rmd** | **r** | **pdf** | **html**
   - There are three pairs of points, formulas for the exact quadratic curve that fits through the points.
   - There are three pairs of points, we observe only differences in y values, formulas for the linear and quadratic parameters.
   - There are three pairs of points, formulas for the linear best fit line through the points.
   - **stats**: *lm(y ~ x + I(x^2), dataset=df) + lm(y ~ poly(x, 2), dataset=df) + summary.lm(rs) + predict(rs)*
2. Quadratic and Ratio Rescaling of Parameters with Fixed Min and Max: **rmd** | **r** | **pdf** | **html**
   - For 0<theta<1, generate 0 < thetaHat(theta, lambda) < 1, where lambda is between positive and negative infinity, used to rescale theta.

- Fit a quadratic function for three points, where the starting and ending points are along the 45 degree line.

- **r**: *sort(unique()) + sapply(ar, func, param=val)*
- **ggplot2**: *geom_line() + geom_vline() + labs(title, subtitle, x, y, caption) + scale_y_continuous(breaks, limits)*

3. Rescaling Bounded Parameter to be Unbounded and Positive and Negative Exponents with Different Bases: **rmd** | **r** | **pdf** | **html**
   - Log of alternative bases, bases that are not e, 10 or 2.
   - A parameter is constrained between 1 and negative infinity, use exponentials of different bases to scale the bounded parameter to an unbounded parameter.

   - Positive exponentials are strictly increasing. Negative exponentials are strictly decreasing.
   - A positive number below 1 to a negative exponents is above 1, and a positive number above 1 to a negative exponents is below 1.
   - **graphics**: *plot(x, y) + title() + legend()*
4. Find the Closest Point Along a Line to Another Point: **rmd** | **r** | **pdf** | **html**
   - A line crosses through the origin, what is the closest point along this line to another point.
   - Graph several functions jointly with points and axis.
   - **graphics**: *par(mfrow = c(1, 1)) + curve(fc) + points(x, y) + abline(v=0, h=0)*
5. linear solve x with f(x) = 0: **rmd** | **r** | **pdf** | **html**
   - Evaluate and solve statistically relevant problems with one equation and one unknown that permit analytical solutions.

A.8.2 Section 8.2 Production Functions links

1. Nested Constant Elasticity of Substitution Production Function: **rmd** | **r** | **pdf** | **html**
   - A nested-CES production function with nest-specific elasticities.
   - Re-state the nested-CES problem as several sub-problems.
   - Marginal products and its relationship to prices in expenditure minimization.
2. Latent Dynamic Health Production Function: **rmd** | **r** | **pdf** | **html**
   - A model of latent health given lagged latent health and health inputs.
   - Find individual-specific production function coefficient given self-rated discrete health status probabilities.
   - Persistence of latent health status given observed discrete current and lagged outcomes.

A.8.3 Section 8.3 Inequality Models links

1. GINI for Discrete Samples or Discrete Random Variable: **rmd** | **r** | **pdf** | **html**
   - Given sample of data points that are discrete, compute the approximate GINI coefficient.
   - Given a discrete random variable, compute the GINI coefficient.
   - **r**: *sort() + cumsum() + sum()*
2. CES and Atkinson Inequality Index: **rmd** | **r** | **pdf** | **html**
   - Analyze how changing individual outcomes shift utility given inequality preference parameters.
   - Discrete a continuous normal random variable with a binomial discrete random variable.
   - Draw Cobb-Douglas, Utilitarian and Leontief indifference curve.
   - **r**: *apply(mt, 1, funct(x){}) + do.call(rbind, ls_mt)*
   - **tidyr**: *expand_grid()*
   - **ggplot2**: *geom_line() + facet_wrap()*
   - **econ**: *Atkinson (JET, 1970)*

A.9 Statistics links

A.9.1 Section 9.1 Random Draws links

1. Randomly Perturb Some Parameter Value with Varying Magnitudes: **rmd** | **r** | **pdf** | **html**
   - Given some existing parameter value, with an intensity value between 0 and 1, decide how to perturb the value.

- **r**: *matrix*
- **stats**: *qlnorm()*
- **graphics**: *par() + hist() + abline()*

A.9.2 Section 9.2 Distributions links

1. Integrate Normal Shocks: **rmd** | **r** | **pdf** | **html**
   - Random Sampling (Monte Carlo) integrate shocks.
   - Trapezoidal rule (symmetric rectangles) integrate normal shock.

A.9.3 Section 9.3 Discrete Random Variable links

1. Binomial Approximation of Normal: **rmd** | **r** | **pdf** | **html**
   - Approximate a continuous normal random variable with a discrete binomial random variable.
   - **r**: *hist() + plot()*
   - **stats**: *dbinom() + rnorm()*

A.10 Tables and Graphs links

A.10.1 Section 10.1 R Base Plots links

1. R Base Plot Line with Curves and Scatter: **rmd** | **r** | **pdf** | **html**
   - Plot scatter points, line plot and functional curve graphs together.
   - Set margins for legend to be outside of graph area, change line, point, label and legend sizes.
   - Generate additional lines for plots successively, record successively, and plot all steps, or initial steps results.
   - **r**: *plot() + curve() + legend() + title() + axis() + par() + recordPlot()*

A.10.2 Section 10.2 ggplot Line Related Plots links

1. ggplot2 Basic Line Plot for Multiple Time Series: **rmd** | **r** | **pdf** | **html**
   - Given three time series, present both in levels, in log levels, and as ratio
   - **ggplot**: *ggplot() + geom_line()*
2. ggplot Line Plot Multiple Categorical Variables With Continuous Variable: **rmd** | **r** | **pdf** | **html**
   - One category is subplot, one category is line-color, one category is line-type.
   - One category is subplot, one category is differentiated by line-color, line-type and scatter-shapes.
   - One category are separate plots, two categories are subplots rows and columns, one category is differentiated by line-color, line-type and scatter-shapes.
   - **ggplot**: *ggplot() + facet_wrap() + facet_grid() + geom_line() + geom_point() + geom_smooth() + geom_hline() + scale_colour_manual() + scale_shape_manual() + scale_shape_discrete() + scale_linetype_manual() + scale_x_continuous() + scale_y_continuous() + theme_bw() + theme() + guides() + theme() + ggsave()*
   - **dplyr**: *filter(vara %in% c(1, 2) & varb == "val") + mutate_if() + !any(is.na(suppressWarnings(as.numeric(n & is.character(x)*
3. Time Series with Shaded Regions, plot GDP with recessions: **rmd** | **r** | **pdf** | **html**
   - Plot several time series with multiple shaded windows.
   - Plot GDP with shaded recession window, and differentially shaded pre- and post-recession windows.
   - **r**: *sample + pmin + diff + which*
   - **ggplot**: *ggplot() + geom_line() + geom_rect(aes(xmin, xmax, ymin, ymax)) + theme_light() + scale_colour_manual() + scale_shape_discrete() + scale_linetype_manual() + scale_fill_manual()*

A.10.3 Section 10.3 ggplot Scatter Related Plots links

1. ggplot Scatter Plot Grouped or Unique Patterns and Colors: **rmd** | **r** | **pdf** | **html**
   - Scatter Plot Three Continuous Variables and Multiple Categorical Variables

- Two continuous variables for the x-axis and the y-axis, another continuous variable for size of scatter, other categorical variables for scatter shape and size.
- Scatter plot with unique pattern and color for each scatter point.
- Y and X label axis with two layers of text in levels and deviation from some mid-point values.
- **tibble**: *rownames_to_column()*
- **ggplot**: *ggplot() + geom_jitter() + geom_smooth() + geom_point(size=1, stroke=1) + scale_colour_manual() + scale_shape_discrete() + scale_linetype_manual() + scale_x_continuous() + scale_y_continuous() + theme_bw() + theme()*

2. ggplot Multiple Scatter-Lines and Facet Wrap Over Categories: **rmd** | **r** | **pdf** | **html**
   - ggplot multiple lines with scatter as points and connecting lines.
   - Facet wrap to generate subfigures for sub-categories.
   - Generate separate plots from data saved separately.
   - **r**: *apply*
   - **ggplot**: *facet_wrap() + geom_smooth() + geom_point() + facet_wrap() + scale_colour_manual() + scale_shape_manual() + scale_linetype_manual()*

A.10.4 Section 10.4 Write and Read Plots links

1. Base R Save Images At Different Sizes: **rmd** | **r** | **pdf** | **html**
   - Base R store image core, add legends/titles/labels/axis of different sizes to save figures of different sizes.
   - **r**: *png() + setEPS() + postscript() + dev.off()*

A.11 Get Data links

A.11.1 Section 11.1 Environmental Data links

1. CDS ECMWF Global Enviornmental Data Download: **rmd** | **r** | **pdf** | **html**
   - Using Python API get get ECMWF ERA5 data.
   - Dynamically modify a python API file, run python inside a Conda virtual environment with R-reticulate.
   - **r**: *file() + writeLines() + unzip() + list.files() + unlink()*
   - **r-reticulate**: *use_python() + Sys.setenv(RETICULATE_PYTHON = spth_conda_env)*

A.12 Coding and Development links

A.12.1 Section 12.1 Installation and Packages links

1. R, RTools, Rstudio Installation and Update with VSCode: **rmd** | **r** | **pdf** | **html**
   - Install and update R, RTools, and Rstudio.
   - Set-up R inside VSCode.
   - **installr**: *updateR()*
2. Handling R Packages: **rmd** | **r** | **pdf** | **html**
   - Resolve conflicts between two packages with identically named function.
   - **tidyverse**: *tidyverse_conflicts*
   - **dplyr**: *filter*
   - **stats**: *filter*
   - **conflicted**: *conflict_prefer()*

A.12.2 Section 12.2 Files In and Out links

1. Decompose File Paths to Get Folder and Files Names: **rmd** | **r** | **pdf** | **html**
   - Decompose file path and get file path folder names and file name.
   - Check if file name exists.
   - **r**: *.Platform$file.sep + tail() + strsplit() + basename() + dirname() + substring() + dir.exists() + file.exists()*
2. Save Text to File, Read Text from File, Replace Text in File: **rmd** | **r** | **pdf** | **html**
   - Save data to file, read text from file, replace text in file.

- **r**: *kable() + file() + writeLines() + readLines() + close() + gsub()*

3. Convert R Markdown File to R, PDF and HTML: **rmd** | **r** | **pdf** | **html**
    - Find all files in a folder with a particula suffix, with exclusion.
    - Convert R Markdow File to R, PDF and HTML.
    - Modify markdown pounds hierarchy.
    - **r**: *file() + writeLines() + readLines() + close() + gsub()*

A.12.3 Section 12.3 Python with R links

1. Python in R with Reticulate: **rmd** | **r** | **pdf** | **html**
    - Use Python in R with Reticulate
    - **reticulate**: *py_config() + use_condaenv() + py_run_string() + Sys.which('python')*

A.12.4 Section 12.4 Command Line links

1. System and Shell Commands in R: **rmd** | **r** | **pdf** | **html**
    - Run system executable and shell commands.
    - Activate conda environment with shell script.
    - **r**: *system() + shell()*

A.12.5 Section 12.5 Run Code in Parallel in R links

1. Run Code in Parallel in R: **rmd** | **r** | **pdf** | **html**
    - Running parallel code in R
    - **parallel**: *detectCores() + makeCluster()*
    - **doParallel**: *registerDoParallel()*
    - **foreach**: *dopar*

Bibliography

R Core Team (2019). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Wang, F. (2020). *REconTools: R Tools for Panel Data and Optimization.* R package version 0.0.0.9000.

Wickham, H. (2019). *tidyverse: Easily Install and Load the 'Tidyverse'.* R package version 1.3.0.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown.* R package version 0.18.