Data Structures and Cloud Services with Python

Fan Wang

2020-12-28

Contents

Preface

The work-in-progress Py4Econ python examples repository.

> **bookdown site** and **bookdown pdf**.

Files are written with RMD. Materials are gathered from various projects in which python code is used for research and paper-administrative tasks. Bullet points show which python packages/functions are used to achieve various objectives. This is not a python package, but a set of example files. The pyfan package repository provides some associated programs.

Other repositories: For dynamic savings problems, see MEconoTools; For code examples, see also Matlab Example Code, R Example Code, and Stata Example Code; For intro econ with Matlab, see Intro Mathematics for Economists, and for intro stat with R, see 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

Data Structures

1.1 Numbers, Strings, Lists and Tuples

1.1.1 Numeric Basics

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import numpy as np
```

1.1.1.1 Two Digit Numbers Coding Conditional Information

We have numbers between 0 and 99, these indicate different estimation specifications, where the digit number is the estimation tolerance level, and the tens number is the minimization algorithm.

```
ls_it_esti_optsalgo = [0, 1, 10, 15, 23, 89, 90, 99, 900, 901, 999, 1000]
for it_esti_optsalgo in ls_it_esti_optsalgo:
    it_esti_optsalgo_tens = int(np.floor(it_esti_optsalgo/10))
    it_esti_optsalgo_digits = it_esti_optsalgo%10
    print(f'{it_esti_optsalgo_tens=}, {it_esti_optsalgo_digits=}')
```

```
## it_esti_optsalgo_tens=0, it_esti_optsalgo_digits=0
## it_esti_optsalgo_tens=0, it_esti_optsalgo_digits=1
## it_esti_optsalgo_tens=1, it_esti_optsalgo_digits=0
## it_esti_optsalgo_tens=1, it_esti_optsalgo_digits=5
## it_esti_optsalgo_tens=2, it_esti_optsalgo_digits=3
## it_esti_optsalgo_tens=8, it_esti_optsalgo_digits=9
## it_esti_optsalgo_tens=9, it_esti_optsalgo_digits=0
## it_esti_optsalgo_tens=9, it_esti_optsalgo_digits=9
## it_esti_optsalgo_tens=90, it_esti_optsalgo_digits=0
## it_esti_optsalgo_tens=90, it_esti_optsalgo_digits=1
## it_esti_optsalgo_tens=99, it_esti_optsalgo_digits=9
## it_esti_optsalgo_tens=100, it_esti_optsalgo_digits=0
```

1.1.2 Tuple

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import numpy as np
```

1.1.2.1 Deal Variables

Define a number of variables in one line.

```
st_cta, st_ctb, it_ctc = 'e', '20201025x_esr', 2
print(f'{st_cta=} and {st_ctb=} and {it_ctc=}')
```

```
## st_cta='e' and st_ctb='20201025x_esr' and it_ctc=2
```

1.1.2.2 Tuple Example

A tuple is created with parenthesis on the sides (or no parenthesis), not brackets on the sides. Can access values in a tuple as in list.

```
# Define Tuple, with and without parenthesis
tp_abc = ('a', 'b', 'c')
tp_abc_noparent = 'a', 'b', 'c'
print(f'{tp_abc=}')
```

```
## tp_abc=('a', 'b', 'c')
```

```
print(f'{len(tp_abc)=}')
```

```
## len(tp_abc)=3
```

```
print(f'{tp_abc_noparent=}')
```

```
## tp_abc_noparent=('a', 'b', 'c')
```

```
print(f'{(tp_abc==tp_abc_noparent)=}')
# Check Type
```

```
## (tp_abc==tp_abc_noparent)=True
```

```
print(f'{isinstance(tp_abc, list)=}')
```

```
## isinstance(tp_abc, list)=False
```

```
print(f'{isinstance(tp_abc, tuple)=}')
# select element
```

```
## isinstance(tp_abc, tuple)=True
```

```
print(f'{tp_abc[1]=}')
```

```
## tp_abc[1]='b'
```

Convert tuple to a list:

```
# convert Tuple to list
ls_abc = [i for i in tp_abc]
print(f'{ls_abc=}')
```

```
## ls_abc=['a', 'b', 'c']
```

```
print(f'{isinstance(ls_abc, list)=}')
```

```
## isinstance(ls_abc, list)=True
```

```
print(f'{isinstance(ls_abc, tuple)=}')
```

```
## isinstance(ls_abc, tuple)=False
```

Since the tuple is not mutable, we can not change values inside the tuple:

```
# define the tuple
tp_abc = ('a', 'b', 'c')
# update tuple value
try:
    tp_abc[0] = 'efg'
```

```
except TypeError as error:
    print('Caught this error: ' + repr(error))
```

```
## Caught this error: TypeError("'tuple' object does not support item assignment")
```

1.1.2.3 Function Returns Tuple and Unpack

When a function returns multiple *items* in a list, that is a tuple. Each element of the list can be accessed. And the tuple can be unpacked:

```
# Results from some function
tp_results = 'a', 123,  [1,2,3]
# Unpack the results
a_st, b_int, c_ls_int = tp_results
# Print
print(f'{tp_results=}')
```

```
## tp_results=('a', 123, [1, 2, 3])
```

```
print(f'{a_st=}')
```

```
## a_st='a'
```

```
print(f'{b_int=}')
```

```
## b_int=123
```

```
print(f'{c_ls_int=}')
```

```
## c_ls_int=[1, 2, 3]
```

Unpack only a subset of the elements in the tuple:

```
# Unpack only one
a_st_self_m1, _ , _ = tp_results
# Alternative shorter
a_st_self_m2, *_  = tp_results
# Print
print(f'{a_st_self_m1=}')
```

```
## a_st_self_m1='a'
```

```
print(f'{a_st_self_m2=}')
```

```
## a_st_self_m2='a'
```

see unpack the first two elements in list/tuple.

1.1.3 List

> Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import numpy as np
```

1.1.3.1 Loop Through a List get Value and Index

There is a list, loop through it, get out both the index and the value of each indexed element together.

```
ls_ob_combo_type = ["e", "20201025x_esr_list_tKap_mlt_ce1a2", ["esti_param.kappa_ce9901", "esti_para
for it_idx, ob_val in enumerate(ls_ob_combo_type):
    print(f'{it_idx=} and {ob_val=}')
```

```
## it_idx=0 and ob_val='e'
## it_idx=1 and ob_val='20201025x_esr_list_tKap_mlt_ce1a2'
```

```
## it_idx=2 and ob_val=['esti_param.kappa_ce9901', 'esti_param.kappa_ce0209']
## it_idx=3 and ob_val=1
## it_idx=4 and ob_val='C1E31M3S3=1'
```

1.1.3.2 Parse Elements of a List

A list has multiple elements, deal them out.

```
list_test = ['C1E126M4S3', 2]
[compesti_short_name, esti_top_which] = list_test
print(f'{compesti_short_name=} and {esti_top_which=}')
```

```
## compesti_short_name='C1E126M4S3' and esti_top_which=2
```

1.1.3.3 Check if Any Element of a List is another List

There is a list with two elements, there is another list with say four elements. Check if any of the two element is in the list with four elements.

```
ar_int_list_a1 = [1,2]
ar_int_list_a2 = [2,1111]
ar_int_list_a3 = [2111,1111]
ar_int_list_b = [1,2,3,11,999]
ar_int_list_c = [2]

check_any_a1_in_b =  any(item in ar_int_list_a1 for item in ar_int_list_b)
check_any_a2_in_b =  any(item in ar_int_list_a2 for item in ar_int_list_b)
check_any_a3_in_b =  any(item in ar_int_list_a3 for item in ar_int_list_b)
check_any_a1_in_c =  any(item in ar_int_list_a1 for item in ar_int_list_c)

print(f'{check_any_a1_in_b=}')
```

```
## check_any_a1_in_b=True
```

```
print(f'{check_any_a2_in_b=}')
```

```
## check_any_a2_in_b=True
```

```
print(f'{check_any_a3_in_b=}')
```

```
## check_any_a3_in_b=False
```

```
print(f'{check_any_a1_in_c=}')
```

```
## check_any_a1_in_c=True
```

1.1.3.4 Convert a List to a String List

Given a list of string and numeric values, convert to a list of string values. The MAP function is like apply in R.

- How to concatenate items in a list to a single string?

```
ls_spec_key = ['ng_s_d', 'esti_test_11_simu', 2, 3]
ls_st_spec_key = list(map(str, ls_spec_key))
print(ls_st_spec_key)
```

```
## ['ng_s_d', 'esti_test_11_simu', '2', '3']
```

Additionally, append some common element to each element of the string using MAP.

```
ls_spec_key = ['ng_s_d', 'esti_test_11_simu', 2, 3]
ls_st_spec_key = list(map(lambda x: 'add++' + str(x), ls_spec_key))
print(ls_st_spec_key)
```

```
## ['add++ng_s_d', 'add++esti_test_11_simu', 'add++2', 'add++3']
```

Equivalently, via list comprehension

```
ls_spec_key = ['ng_s_d', 'esti_test_11_simu', 2, 3]
ls_st_spec_key = ['list_comprehension' + str(spec_key) for spec_key in ls_spec_key]
print(ls_st_spec_key)
```

```
## ['list_comprehensionng_s_d', 'list_comprehensionesti_test_11_simu', 'list_comprehension2', 'list_
```

1.1.3.5 Concatenate a List to a String with Separator

Given a list of strings and numeric data types, concatenate list to a string with some separator. Also in reverse, generate a list by breaking a string joined by some separator.

```
ls_spec_key = ['ng_s_d', 'esti_test_11_simu', 2, 3]
st_separator = '='
st_spec_key = st_separator.join(list(map(lambda x : str(x), ls_spec_key)))
print(st_spec_key)
```

```
## ng_s_d=esti_test_11_simu=2=3
```

Now break string apart:

```
st_spec_key = '='.join(list(map(lambda x : '$' + str(x) + '$', ['ng_s_d', 'esti_test_11_simu', 2, 3]
print(st_spec_key.split('='))
```

```
## ['$ng_s_d$', '$esti_test_11_simu$', '$2$', '$3$']
```

1.1.3.6 Add Nth Element to List when Nth Element Does not Exist

There is a list with 2 elements, check if the list has 3 elements, if not, add another element.

```
ls_string_A = ['c', '20180918']
ls_string_B = ['c', '20180918', ['esti_param.alpha_k']]

for ls_string in [ls_string_A, ls_string_B]:
  if len(ls_string) == 2:
    ls_string.insert(2, None)

  print(ls_string)
```

```
## ['c', '20180918', None]
## ['c', '20180918', ['esti_param.alpha_k']]
```

1.1.3.7 Check If List Has N Elements of None for Some Elements

In the example below, for A, B and C, do something, for D and E do something else.

```
ls_string_A = ['c', '20180918']
ls_string_B = ['c', '20180918', None]
ls_string_C = ['c', '20180918', None, None]
ls_string_D = ['c', '20180918', ['esti_param.alpha_k'], None]
ls_string_E = ['c', '20180918', ['esti_param.alpha_k'], 5]

for ls_string in [ls_string_A, ls_string_B, ls_string_C, ls_string_D, ls_string_E]:
  if len(ls_string) >= 3 and ls_string[2] is not None:
    print(ls_string)
  else:
    print(ls_string[0:2])
```

```
## ['c', '20180918']
```

```
## ['c', '20180918']
## ['c', '20180918']
## ['c', '20180918', ['esti_param.alpha_k'], None]
## ['c', '20180918', ['esti_param.alpha_k'], 5]
```

1.1.3.8 Add a Default Value to Nth Element of List

There is a string list with potential potentially three elements. But sometimes the input only has two elements. Provide default third element value if third element is NONE or if the string list only has two elements.

```
ls_string_A = ['c', '20180918']
ls_string_B = ['c', '20180918', None]
ls_string_C = ['c', '20180918', ['esti_param.alpha_k']]

for ls_string in [ls_string_A, ls_string_B, ls_string_C]:

  if len(ls_string) <= 2:
    # Deals with situation A
    ls_string.append(['esti_param.alpha_k'])
  elif ls_string[2] is None:
    # Deals with situation B
    ls_string[2] = ['esti_param.alpha_k']
  else:
    # Situation C
    pass

  print(ls_string)
```

```
## ['c', '20180918', ['esti_param.alpha_k']]
## ['c', '20180918', ['esti_param.alpha_k']]
## ['c', '20180918', ['esti_param.alpha_k']]
```

Now do the same thing for a numeric list:

```
ls_string_A = [11, 22]
ls_string_B = [11, 22, None]
ls_string_C = [11, 22, 33]

for ls_string in [ls_string_A, ls_string_B, ls_string_C]:

  if len(ls_string) <= 2:
    # Deals with situation A
    ls_string.append(33)
  elif ls_string[2] is None:
    # Deals with situation B
    ls_string[2] = 33
  else:
    # Situation C
    pass

  print(ls_string)
```

```
## [11, 22, 33]
## [11, 22, 33]
## [11, 22, 33]
```

1.1.4 Strings

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import numpy as np
import string as string
import random as random
import pprint
```

1.1.4.1 List of Array Count Frequency

There is a list of strings, with repeating values, count the frequence of the each unique string.

```
ls_st_status = ["success", "running", "running", "running", "finished", "pending", "pending"]
ls_freq = [ [f'{ls_st_status.count(st_status)} of {len(ls_st_status)} {st_status}']  for st_status i
pprint.pprint(ls_freq)
```

```
## [['3 of 7 running'],
##  ['2 of 7 pending'],
##  ['1 of 7 success'],
##  ['1 of 7 finished']]
```

1.1.4.2 Get Substring

Given string, get substring after a word.

```
st_func_stack_code = 'dc_ls_combo_type = pyfan_amto_lsdcconvert.ff_ls2dc(ls_combo_type,'
st_search_break = 'ff_ls2dc('
st_string_after = st_func_stack_code.split(st_search_break)[1]
st_search_break = ','
st_string_after = st_func_stack_code.split(st_search_break)[1]
print(f'{st_string_after=}')
```

```
## st_string_after=''
```

1.1.4.3 Generate Random Strings

Generate some random strings:

```
random.seed(123)
it_word_length = 5
st_rand_word = ''.join(random.choice(string.ascii_lowercase) for i in range(it_word_length))
st_rand_word = st_rand_word.capitalize()
print(f'{st_rand_word=}')
```

```
## st_rand_word='Bicyn'
```

Generate a block or random text and then convert it to a one list of strings:

```
random.seed(123)
it_words_count = 15
it_word_length = 5
st_rand_word_block = ''.join(random.choice(string.ascii_lowercase) for ctr in range(it_word_length*i
ls_st_rand_word = [st_rand_word_block[ctr: ctr + it_word_length].capitalize()
                   for ctr in range(0, len(st_rand_word_block), it_word_length)]
print(f'{ls_st_rand_word=}')
```

```
## ls_st_rand_word=['Bicyn', 'Idbmr', 'Rkkbf', 'Ekrkw', 'Hfany', 'Ctmca', 'Kxodb', 'Cveez', 'Ajnsp',
```

Reshape the array of words to a matrix:

```
mt_st_rand_word = np.reshape(ls_st_rand_word, [3,5])
print(f'{mt_st_rand_word=}')
```

```
## mt_st_rand_word=array([['Bicyn', 'Idbmr', 'Rkkbf', 'Ekrkw', 'Hfany'],
##        ['Ctmca', 'Kxodb', 'Cveez', 'Ajnsp', 'Ipbyj'],
##        ['Kqzpg', 'Tuqsz', 'Kamyu', 'Qnvru', 'Zvtpq']], dtype='<U5')
```

```
print(f'{mt_st_rand_word.shape=}')
```

```
## mt_st_rand_word.shape=(3, 5)
```

```
print(f'{type(mt_st_rand_word)=}')
```

```
## type(mt_st_rand_word)=<class 'numpy.ndarray'>
```

1.1.4.4 Add String Suffix to Numeric Array

Given an numeric array, add string, for example to generate sequencial column names with suffix c:

```
ar_st_colnames = [ 's' + str(it_col) for it_col in np.array(range(1, 3))]
print(ar_st_colnames)
```

```
## ['s1', 's2']
```

1.1.4.5 Search if Names Include Strings

Given a list of strings, loop but skip if string contains elements string list.

```
# define string
ls_st_ignore = ['abc', 'efg', 'xyz']
ls_st_loop = ['ab cefg sdf', '12345', 'xyz', 'abc xyz', 'good morning']

# zip and loop and replace
for st_loop in ls_st_loop:
  if sum([st_ignore in st_loop for st_ignore in ls_st_ignore]):
    print('skip:', st_loop)
  else:
    print('not skip:', st_loop)
```

```
## skip: ab cefg sdf
## not skip: 12345
## skip: xyz
## skip: abc xyz
## not skip: good morning
```

1.1.4.6 Replace a Set of Strings in String

Replace terms in string

```
# define string
st_full = """
abc is a great efg, probably xyz. Yes, xyz is great, like efg.
eft good, EFG capitalized, efg good again.
A B C or abc or ABC. Interesting xyz.
"""

# define new and old
ls_st_old = ['abc', 'efg', 'xyz']
ls_st_new = ['123', '456', '789']

# zip and loop and replace
for old, new in zip(ls_st_old, ls_st_new):
  st_full = st_full.replace(old, new)

# print
print(st_full)
```

```
##
## 123 is a great 456, probably 789. Yes, 789 is great, like 456.
## eft good, EFG capitalized, 456 good again.
## A B C or 123 or ABC. Interesting 789.
```

1.1.4.7 Wrap String with Fixed Width

Given a long string, wrap it into multiple lines with fixed width.

```
import textwrap

# A long Path
st_path = """
C:/Users/fan/Documents/Dropbox (UH-ECON)/Project Emily Minority Survey/EthLang/reg_lang_abi_cls_mino
"""

# Wrap text with tight width
st_wrapped = textwrap.fill(st_path, width = 20)
print(st_wrapped)
```

```
##  C:/Users/fan/Docume
## nts/Dropbox (UH-
## ECON)/Project Emily
## Minority Survey/EthL
## ang/reg_lang_abi_cls
## _mino/tab3_fm/attain
## _m_vs_f/tab3_mand_ta
## lk_m2c_hfracle02.tex
```

Combine Strings that are wrapped and not Wrapped

```
# Paths
st_path_a = "C:/Users/fan/Documents/Dropbox (UH-ECON)/Project Emily Minority Survey/EthLang/reg_lang
st_path_b = 'C:/Users/fan/R4Econ/support/development/fs_packaging.html'

# Combine Strings and Wrap
str_dc_records = 'First Path:'.upper() + '\n' + \
                 textwrap.fill(st_path_a, width=25) + '\n\n' + \
                 'Second Path:'.upper() + '\n' + \
                 textwrap.fill(st_path_b, width=25)

# Print
print(str_dc_records)
```

```
## FIRST PATH:
## C:/Users/fan/Documents/Dr
## opbox (UH-ECON)/Project
## Emily Minority Survey/Eth
## Lang/reg_lang_abi_cls_min
## o/tab3_fm/attain_m_vs_f/t
## ab3_mand_talk_m2c_hfracle
## 02.tex
##
## SECOND PATH:
## C:/Users/fan/R4Econ/suppo
## rt/development/fs_packagi
## ng.html
```

1.1.4.8 Change Round for Lists of String Estimates

Here we have two strings in a list, with point estimates and corresponding standard errors. Estimates are separated by commas. We want to change the number of decimal points shown and set appropriate roundings. Several steps: (1) split string by comma (2) Loop over (3) extract numerical elements (4) recover

```
it_round_decimal = 1
ls_st_all_estimates = ["84.506***, 91.758***, 107.950***, 115.879***, 133.560***\n",
                       "(7.796), (4.848), (4.111), (5.044), (6.961)\n",
                       "68.180***, 47.921***, 47.127***, 51.366***, 41.764***\n",
                       "(8.986), (5.368), (4.995), (5.099), (8.637)\n"]

for st_all_estimates in ls_st_all_estimates:

    # delete linebreak at end of line
    st_all_estimates = st_all_estimates.replace("\n", "")

    # split
    ls_st_estimates = st_all_estimates.split(",")

    # Loop over each value separated by commas
    for it_esti_ctr, st_esti in enumerate(ls_st_estimates):

        # Default update is to keep current
        st_esti_update = st_esti

        # If estimates, might have stars
        st_esti_numeric = st_esti.strip()
        st_esti_numeric = st_esti_numeric.replace("*", "")
        st_esti_numeric = st_esti_numeric.replace("(", "")
        st_esti_numeric = st_esti_numeric.replace(")", "")

        # Decimal Rounding
        fl_esti_rounded = round(float(st_esti_numeric), it_round_decimal)
        st_esti_rounded = f'{fl_esti_rounded:.{it_round_decimal}f}'

        # Replace
        print(f'{st_esti=} + {st_esti_numeric=} + {st_esti_rounded=}')
        st_esti_rounded = st_esti.replace(st_esti_numeric, st_esti_rounded)

        # Update List
        ls_st_estimates[it_esti_ctr] = st_esti_rounded

    # Flatten comman
    st_text_out = ','.join(ls_st_estimates)
    print(f'\n{st_text_out=}\n')
    print()
```

```
## st_esti='84.506***' + st_esti_numeric='84.506' + st_esti_rounded='84.5'
## st_esti=' 91.758***' + st_esti_numeric='91.758' + st_esti_rounded='91.8'
## st_esti=' 107.950***' + st_esti_numeric='107.950' + st_esti_rounded='108.0'
## st_esti=' 115.879***' + st_esti_numeric='115.879' + st_esti_rounded='115.9'
## st_esti=' 133.560***' + st_esti_numeric='133.560' + st_esti_rounded='133.6'
##
## st_text_out='84.5***, 91.8***, 108.0***, 115.9***, 133.6***'
##
##
## st_esti='(7.796)' + st_esti_numeric='7.796' + st_esti_rounded='7.8'
```

```
## st_esti=' (4.848)' + st_esti_numeric='4.848' + st_esti_rounded='4.8'
## st_esti=' (4.111)' + st_esti_numeric='4.111' + st_esti_rounded='4.1'
## st_esti=' (5.044)' + st_esti_numeric='5.044' + st_esti_rounded='5.0'
## st_esti=' (6.961)' + st_esti_numeric='6.961' + st_esti_rounded='7.0'
##
## st_text_out='(7.8), (4.8), (4.1), (5.0), (7.0)'
##
##
## st_esti='68.180***' + st_esti_numeric='68.180' + st_esti_rounded='68.2'
## st_esti=' 47.921***' + st_esti_numeric='47.921' + st_esti_rounded='47.9'
## st_esti=' 47.127***' + st_esti_numeric='47.127' + st_esti_rounded='47.1'
## st_esti=' 51.366***' + st_esti_numeric='51.366' + st_esti_rounded='51.4'
## st_esti=' 41.764***' + st_esti_numeric='41.764' + st_esti_rounded='41.8'
##
## st_text_out='68.2***, 47.9***, 47.1***, 51.4***, 41.8***'
##
##
## st_esti='(8.986)' + st_esti_numeric='8.986' + st_esti_rounded='9.0'
## st_esti=' (5.368)' + st_esti_numeric='5.368' + st_esti_rounded='5.4'
## st_esti=' (4.995)' + st_esti_numeric='4.995' + st_esti_rounded='5.0'
## st_esti=' (5.099)' + st_esti_numeric='5.099' + st_esti_rounded='5.1'
## st_esti=' (8.637)' + st_esti_numeric='8.637' + st_esti_rounded='8.6'
##
## st_text_out='(9.0), (5.4), (5.0), (5.1), (8.6)'
```

1.2 Dictionary

1.2.1 Dictionary

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import pprint
import copy as copy
```

1.2.1.1 Loop Through a Dictionary

Given a dictionary, loop through all of its elements

```
dc_speckey_dict = {0: 'mpoly_1',
                   1: 'ng_s_t',
                   2: 'ng_s_d',
                   3: 'ng_p_t',
                   4: 'ng_p_d'}
for speckey_key, speckey_val in dc_speckey_dict.items():
    print('speckey_key:' + str(speckey_key) + ', speckey_val:' + speckey_val)
```

```
## speckey_key:0, speckey_val:mpoly_1
## speckey_key:1, speckey_val:ng_s_t
## speckey_key:2, speckey_val:ng_s_d
## speckey_key:3, speckey_val:ng_p_t
## speckey_key:4, speckey_val:ng_p_d
```

1.2.1.2 Convert a List to a Dictionary

There is a list with different types of elements. Use the name of the list as the key, with value index added in as a part of the key, the value is the value of the dictionary.

```
# List
ls_combo_type = ["e", "20201025x_esr_list_tKap_mlt_ce1a2", ["esti_param.kappa_ce9901", "esti_param.k
```

```
# List name as string variable
st_ls_name = f'{ls_combo_type=}'.split('=')[0]
# Convert to dict
dc_from_list = {st_ls_name + '_i' + str(it_idx) + 'o' + str(len(ls_combo_type)) : ob_val
                for it_idx, ob_val in enumerate(ls_combo_type)}
# Print
pprint.pprint(dc_from_list, width=1)
```

```
## {'ls_combo_type_i0o5': 'e',
##  'ls_combo_type_i1o5': '20201025x_esr_list_tKap_mlt_ce1a2',
##  'ls_combo_type_i2o5': ['esti_param.kappa_ce9901',
##                         'esti_param.kappa_ce0209'],
##  'ls_combo_type_i3o5': 1,
##  'ls_combo_type_i4o5': 'C1E31M3S3=1'}
```

1.2.1.3 Select One Key-value Pair

Given a dictionary, select a single key-value pair, based on either the key or the value.

```
# select by key
ls_it_keys = [0, 4]
dc_speckey_dict_select_by_key = {it_key: dc_speckey_dict[it_key] for it_key in ls_it_keys}
print(f'{dc_speckey_dict_select_by_key=}')
# select by value
```

```
## dc_speckey_dict_select_by_key={0: 'mpoly_1', 4: 'ng_p_d'}
```

```
ls_st_keys = ['ng_s_d', 'ng_p_d']
dc_speckey_dict_select_by_val = {it_key: st_val for it_key, st_val in dc_speckey_dict.items()
                                 if st_val in ls_st_keys}
print(f'{dc_speckey_dict_select_by_val=}')
```

```
## dc_speckey_dict_select_by_val={2: 'ng_s_d', 4: 'ng_p_d'}
```

See Get key by value in dictionary.

1.2.1.4 Copying Dictionary and Updating Copied Dictionary

First, below, it looks as if the default dictionary has been copied, and that the updates to the dictionary will only impact the *dc_invoke_main_args*, but that is not the case:

```
# list update
dc_invoke_main_args_default = {'speckey': 'ng_s_t',
                               'ge': False,
                               'multiprocess': False,
                               'estimate': False,
                               'graph_panda_list_name': 'min_graphs',
                               'save_directory_main': 'simu',
                               'log_file': False,
                               'log_file_suffix': ''}
dc_invoke_main_args = dc_invoke_main_args_default
dc_invoke_main_args['speckey'] = 'b_ge_s_t_bis'
dc_invoke_main_args['ge'] = True
print(f'speckey in dc_invoke_main_args is {dc_invoke_main_args["speckey"]}.')
```

```
## speckey in dc_invoke_main_args is b_ge_s_t_bis.
```

```
print(f'speckey in dc_invoke_main_args_default is {dc_invoke_main_args_default["speckey"]}.')
```

```
## speckey in dc_invoke_main_args_default is b_ge_s_t_bis.
```

Now this has the intended result. After updating the deep-copied dictionary, the key-values in the original dictionary are preserved:

```
# list update
dc_invoke_main_args_default = {'speckey': 'ng_s_t',
                               'ge': False,
                               'multiprocess': False,
                               'estimate': False,
                               'graph_panda_list_name': 'min_graphs',
                               'save_directory_main': 'simu',
                               'log_file': False,
                               'log_file_suffix': ''}
# deep copy and update
dc_invoke_main_args = copy.deepcopy(dc_invoke_main_args_default)
dc_invoke_main_args['speckey'] = 'b_ge_s_t_bis'
dc_invoke_main_args['ge'] = True
print(f'speckey in dc_invoke_main_args_default is {dc_invoke_main_args_default["speckey"]}.')
```

```
## speckey in dc_invoke_main_args_default is ng_s_t.
```

```
print(f'speckey in dc_invoke_main_args is {dc_invoke_main_args["speckey"]}.')
# deep copy and update again
```

```
## speckey in dc_invoke_main_args is b_ge_s_t_bis.
```

```
dc_invoke_main_args = copy.deepcopy(dc_invoke_main_args_default)
dc_invoke_main_args['speckey'] = 'b_ge_s_t_bis_new'
dc_invoke_main_args['ge'] = False
print(f'speckey in dc_invoke_main_args is {dc_invoke_main_args["speckey"]}.')
```

```
## speckey in dc_invoke_main_args is b_ge_s_t_bis_new.
```

- copy and deepcopy
- Deep copy of a dict in python

1.2.1.5 Create a List of Dictionaries

```
import datetime
import pprint
ls_dc_exa =  [
    {"file": "mat_matlab",
     "title": "One Variable Graphs and Tables",
     "description": "Frequency table, bar chart and histogram",
     "val": 1,
     "date": datetime.date(2020, 5, 2)},
    {"file": "mat_two",
     "title": "Second file",
     "description": "Second file.",
     "val": [1, 2, 3],
     "date": datetime.date(2020, 5, 2)},
    {"file": "mat_algebra_rules",
     "title": "Opening a Dataset",
     "description": "Opening a Dataset.",
     "val": 1.1,
     "date": datetime.date(2018, 12, 1)}
]
pprint.pprint(ls_dc_exa, width=1)
```

```
## [{'date': datetime.date(2020, 5, 2),
##   'description': 'Frequency '
##                  'table, '
##                  'bar '
##                  'chart '
```

```
##                 'and '
##                 'histogram',
##   'file': 'mat_matlab',
##   'title': 'One '
##            'Variable '
##            'Graphs '
##            'and '
##            'Tables',
##   'val': 1},
##  {'date': datetime.date(2020, 5, 2),
##   'description': 'Second '
##                  'file.',
##   'file': 'mat_two',
##   'title': 'Second '
##            'file',
##   'val': [1,
##           2,
##           3]},
##  {'date': datetime.date(2018, 12, 1),
##   'description': 'Opening '
##                  'a '
##                  'Dataset.',
##   'file': 'mat_algebra_rules',
##   'title': 'Opening '
##            'a '
##            'Dataset',
##   'val': 1.1}]
```

1.2.1.6 Iteratively Add to A Dictionary

Iteratively add additional Key and Value pairs to a dictionary.

```
ls_snm_tex = ["file1.tex", "file2.tex", "file3.tex"]
ls_snm_pdf = ["file1.pdf", "file2.pdf", "file3.pdf"]

dc_tex_pdf = {}
for tex, pdf in zip(ls_snm_tex, ls_snm_pdf):
  dc_tex_pdf[tex] = pdf

pprint.pprint(dc_tex_pdf, width=1)
```

```
## {'file1.tex': 'file1.pdf',
##  'file2.tex': 'file2.pdf',
##  'file3.tex': 'file3.pdf'}
```

1.2.1.7 Select by Keys Dictionaries from list of Dictionaries

Given a list of dictionary, search if key name is in list:

```
# string to search through
ls_str_file_ids = ['mat_matlab', 'mat_algebra_rules']
# select subset
ls_dc_selected = [dc_exa
                  for dc_exa in ls_dc_exa
                  if dc_exa['file'] in ls_str_file_ids]
# print
pprint.pprint(ls_dc_selected, width=1)
```

```
## [{'date': datetime.date(2020, 5, 2),
##   'description': 'Frequency '
```

```
##                  'table, '
##                  'bar '
##                  'chart '
##                  'and '
##                  'histogram',
##   'file': 'mat_matlab',
##   'title': 'One '
##            'Variable '
##            'Graphs '
##            'and '
##            'Tables',
##   'val': 1},
##  {'date': datetime.date(2018, 12, 1),
##   'description': 'Opening '
##                  'a '
##                  'Dataset.',
##   'file': 'mat_algebra_rules',
##   'title': 'Opening '
##            'a '
##            'Dataset',
##   'val': 1.1}]
```

Search and Select by Multiple Keys in Dictionary. Using two keys below:

```
# string to search through
ls_str_file_ids = ['mat_matlab', 'mat_algebra_rules']
# select subset
ls_dc_selected = [dc_exa
                  for dc_exa in ls_dc_exa
                  if ((dc_exa['file'] in ls_str_file_ids)
                      and
                      (dc_exa['val']== 1))]
# print
pprint.pprint(ls_dc_selected, width=1)
```

```
## [{'date': datetime.date(2020, 5, 2),
##   'description': 'Frequency '
##                  'table, '
##                  'bar '
##                  'chart '
##                  'and '
##                  'histogram',
##   'file': 'mat_matlab',
##   'title': 'One '
##            'Variable '
##            'Graphs '
##            'and '
##            'Tables',
##   'val': 1}]
```

1.2.1.8 Drop Element of Dictionary

Drop element of a dictionary inside a list:

```
# Dictionary
dc_test = [{"file": "mat_matlab_1",
            "title": "One Variable Graphs and Tables",
            "description": "Frequency table, bar chart and histogram",
            "val": 1,
            "date": datetime.date(2020, 5, 2)},
```

```
           {"file": "mat_matlab_2",
            "val": "mat_matlab_2"}]

# Drop
del dc_test[0]['val']
del dc_test[0]['file']
del dc_test[0]['description']
del dc_test[1]['val']

# Print
pprint.pprint(dc_test, width=1)
```

```
## [{'date': datetime.date(2020, 5, 2),
##   'title': 'One '
##            'Variable '
##            'Graphs '
##            'and '
##            'Tables'},
##  {'file': 'mat_matlab_2'}]
```

1.3 Numpy Arrays

1.3.1 Generate Matrix from Arrays

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import numpy as np
```

1.3.1.1 Generate a Random Matrix

Generate a matrix with random numbers and arbitrary number of rows and columns. Several types of matrix below:

1. uniform random
2. integer random
3. integer random resorted (shuffled)
4. integer random redrawn (with replacements)

Set size:

```
it_rows = 2;
it_cols = 3;
np.random.seed(123)
```

uniform random:

```
# A random matrix of uniform draws
mt_rand_unif = np.random.rand(it_rows, it_cols)
print(mt_rand_unif)
```

```
## [[0.69646919 0.28613933 0.22685145]
##  [0.55131477 0.71946897 0.42310646]]
```

integer random:

```
# A random matrix of integers
it_max_int = 10
mt_rand_integer = np.random.randint(it_max_int, size=(it_rows, it_cols))
print(mt_rand_integer)
```

```
## [[6 1 0]
```

```
##  [1 9 0]]
```

integer random resorted (shuffled):

```
# A sequence of numbers, 1 to matrix size, resorted, unique
it_mat_size = it_rows*it_cols
ar_seq = np.arange(it_mat_size)
ar_idx_resort = np.random.choice(np.arange(it_mat_size), it_mat_size, replace = False)
ar_seq_rand_sorted = ar_seq[ar_idx_resort]
mt_seq_rand_sorted = ar_seq_rand_sorted.reshape((it_rows, it_cols))
print(mt_seq_rand_sorted)
# achieve the same objective with a shuffle
```

```
## [[5 4 2]
##  [3 1 0]]
```

```
np.random.shuffle(ar_seq)
mt_seq_rand_shuffle = ar_seq.reshape((it_rows, it_cols))
print(mt_seq_rand_shuffle)
```

```
## [[2 1 3]
##  [5 0 4]]
```

integer random redrawn (with replacements):

```
# A sequence of numbers, 1 to matrix size, resorted, nonunique, REPLACE = TRUE
it_mat_size = it_rows*it_cols
ar_seq = np.arange(it_mat_size)
ar_idx_resort_withreplacement = np.random.choice(np.arange(it_mat_size), it_mat_size, replace = True
ar_seq_rand_sorted_withreplacement = ar_seq[ar_idx_resort_withreplacement]
mt_seq_rand_sorted_withreplacement = ar_seq_rand_sorted_withreplacement.reshape((it_rows, it_cols))
print(mt_seq_rand_sorted_withreplacement)
```

```
## [[3 2 4]
##  [2 4 0]]
```

1.3.1.2 Stack Arrays to Matrix

Given various arrays, generate a matrix by stacking equi-length arrays as columns

```
# three arrays
ar_a = [1,2,3]
ar_b = [3,4,5]
ar_c = [11,4,1]

# Concatenate to matrix
mt_abc = np.column_stack([ar_a, ar_b, ar_c])
print(mt_abc)
```

```
## [[ 1  3 11]
##  [ 2  4  4]
##  [ 3  5  1]]
```

Chapter 2

Pandas

2.1 Panda Basics

2.1.1 Generate Matrix from Arrays

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import numpy as np
import pandas as pd
import random as random
import string as string
```

2.1.1.1 Single Arrays to Matrix

Given various arrays, generate a matrix

```
np.random.seed(123)
# Concatenate to matrix
mt_abc = np.column_stack(np.random.randint(10, size=(5, 3)))
# Matrix to data frame with columns and row names
df_abc = pd.DataFrame(data=mt_abc,
            index=[ 'r' + str(it_col) for it_col in np.array(range(1, mt_abc.shape[0]+1))],
            columns=[ 'c' + str(it_col) for it_col in np.array(range(1, mt_abc.shape[1]+1))])
# Print
print(df_abc)
```

```
##     c1  c2  c3  c4  c5
## r1   2   1   6   1   0
## r2   2   3   1   9   9
## r3   6   9   0   0   3
```

2.1.1.2 Generate a Testing Dataframe with String and Numeric Values

Generate a test dataframe with string and numeric variables. For testing purposes.

```
# Seed
np.random.seed(456)
random.seed(456)

# Numeric matrix 3 rows 4 columns
mt_numeric = np.random.randint(10, size=(3, 4))

# String block 5 letters per word, 3 rows and 3 columns of words
st_rand_word_block = ''.join(random.choice(string.ascii_lowercase) for ctr in range(5*3*3))
```

```
ls_st_rand_word = [st_rand_word_block[ctr: ctr + 5].capitalize() for ctr in range(0, len(st_rand_wor
mt_string = np.reshape(ls_st_rand_word, [3,3])

# Combine string and numeric matrix
mt_data = np.column_stack([mt_numeric, mt_string])

# Matrix to dataframe
df_data = pd.DataFrame(data=mt_data,
                       index=[ 'r' + str(it_col) for it_col in np.array(range(1, mt_data.shape[0]+1)
                       columns=[ 'c' + str(it_col) for it_col in np.array(range(1, mt_data.shape[1]+

# Print table
print(df_data)
```

```
##    c1 c2 c3 c4     c5     c6     c7
## r1  5  9  4  5  Xoonm  Zubtx  Zqdkp
## r2  7  1  8  3  Ydcpw  Obiee  Gfxmq
## r3  5  2  4  2  Tzrwu  Srwvp  Kcsrb
```

2.1.2 Select Rows and Columns from Dataframe

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import numpy as np
import pandas as pd
import random as random
import string as string
```

2.1.2.1 Generate a Testing Dataframe

Generate a testing dataframe for selection and other tests.

```
# Seed
np.random.seed(999)
random.seed(999)
# Numeric matrix 3 rows 4 columns
mt_numeric = np.random.randint(10, size=(5, 4))
st_rand_word_block = ''.join(random.choice(string.ascii_lowercase) for ctr in range(5*5*3))
mt_string = np.reshape([st_rand_word_block[ctr: ctr + 5].capitalize() for ctr in range(0, len(st_ran
mt_data = np.column_stack([mt_numeric, mt_string])

# Matrix to dataframe
df_data = pd.DataFrame(data=mt_data,
                       index=[ 'r' + str(it_col) for it_col in np.array(range(1, mt_data.shape[0]+1)
                       columns=[ 'c' + str(it_col) for it_col in np.array(range(1, mt_data.shape[1]+

# Replace values
df_data = df_data.replace(['Zvcss', 'Dugei', 'Ciagu'], 'Zqovt')

# Print table
print(df_data)
```

```
##    c1 c2 c3 c4     c5     c6     c7
## r1  0  5  1  8  Zqovt  Rppez  Ukuzu
## r2  1  9  3  0  Zqovt  Sbwyi  Mzhum
## r3  5  8  8  0  Zqovt  Qgfvk  Fcrto
## r4  5  2  5  7  Wxlev  Upoax  Bhdxu
## r5  4  6  2  7  Hmziq  Lbyfo  Dntrz
```

2.1.2.2 Select Rows Based on Column/Variable Values

There is a dataframe with many rows, select a subset of rows where a particular column/variable's value is equal to some value.

```
# Concatenate to matrix
df_data_subset = df_data.loc[df_data['c5'] == 'Zqovt']
# Print
print(df_data_subset)
```

```
##    c1 c2 c3 c4     c5     c6     c7
## r1  0  5  1  8  Zqovt  Rppez  Ukuzu
## r2  1  9  3  0  Zqovt  Sbwyi  Mzhum
## r3  5  8  8  0  Zqovt  Qgfvk  Fcrto
```

See How to select rows from a DataFrame based on column values.

2.1.3 Pandas Importing and Exporting

> Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

2.1.3.1 Export a Dataframe to CSV in User Download with Automatic File Name

During debugging and testing, a large dataframe is generated, but certain operation produces error. To fully debug, drop into debugger on error in PyCharm, and use console to generate a dataframe of just the matrix at issue. Now export this dataframe to csv in the fastest way possible.

1. Find user home path, generate a download subdirectory if it does not exist.
2. Export the current dataframe to csv in that file, with auto row and column names.
3. The dataframe will be named after the current variable array name, and will have a time suffix added.

Replace the mt_abc line below, use a different name that should appear in the saved file name.

```
# Import Pathlib and panddas
import pandas as pd
import numpy as np
from pathlib import Path
import time

# replace mt_abc line by the matrix currently used
mt_abc = np.column_stack(np.random.randint(10, size=(5, 3)))
# Save results to C:\Users\fan\Downloads\PythonDebug, generate if does not exist.
srt_pydebug = Path.joinpath(Path.home(), "Downloads", "PythonDebug")
srt_pydebug.mkdir(parents=True, exist_ok=True)
# Matrix to data frame with columns and row names
df2export = pd.DataFrame(data=mt_abc,
                         index=['r' + str(it_col) for it_col in np.array(range(1, mt_abc.shape[0] +
                         columns=['c' + str(it_col) for it_col in np.array(range(1, mt_abc.shape[1]
# Export File Name
spn_csv_path = Path.joinpath(srt_pydebug, f'{mt_abc=}'.split('=')[0] + '-' + time.strftime("%Y%m%d-%
# export
df2export.to_csv(spn_csv_path, sep=",")

# print
print(f'{srt_pydebug=}')
```

```
## srt_pydebug=WindowsPath('C:/Users/fan/Downloads/PythonDebug')
```

```
print(f'{spn_csv_path=}')
```

```
## spn_csv_path=WindowsPath('C:/Users/fan/Downloads/PythonDebug/mt_abc-20201228-220300.csv')
```

Chapter 3

Functions

3.1 Function Arguments and Returns

3.1.1 Data Types

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

3.1.1.1 Check Parameter Type

There are parameters of a function, depending on the parameter type, execute the program differently. If integer, behave one way if string behave in another way.

```
# define the function
def get_speckey_dict(gn_speckey=None):
    if isinstance(gn_speckey, str):
        print(f'{gn_speckey=} is a string')
    elif isinstance(gn_speckey, int):
        print(f'{gn_speckey=} is an integer')
    else:
        raise TypeError(f'{gn_speckey=} was not a string or an integer')
# Call function with integer
get_speckey_dict(1)
# Call function with string
```

```
## gn_speckey=1 is an integer
```

```
get_speckey_dict('abc')
# Call function with a list
```

```
## gn_speckey='abc' is a string
```

```
try:
    get_speckey_dict(['abc'])
except TypeError as e:
    print(f'Exception{e=}')
```

```
## Exceptione=TypeError("gn_speckey=['abc'] was not a string or an integer")
```

3.1.1.2 Check if Parameter is String or Integer In Interval

There is a function that takes a string or an integer between certain values. Execute if either of these two conditions are satisfied, do not if neither is satisfied. Below, print if string or an int between 1 and 11.

```
# condition check function
def check_condition(gn_invoke_set):

  bl_is_str = isinstance(gn_invoke_set, str)
  bl_is_int = isinstance(gn_invoke_set, int)
  if bl_is_int:
    bl_between = (gn_invoke_set >= 1 and gn_invoke_set <= 11)
  else:
    bl_between = False

  if bl_between or bl_is_str:
      print(f'{gn_invoke_set=}')
  else:
      print(f'{gn_invoke_set=} is not string or an integer between 1 and 11')

# call with string or integer
check_condition('string')
```

```
## gn_invoke_set='string'
```

```
check_condition(11)
```

```
## gn_invoke_set=11
```

```
check_condition(1)
```

```
## gn_invoke_set=1
```

```
check_condition(199)
```

```
## gn_invoke_set=199 is not string or an integer between 1 and 11
```

```
check_condition(['abc'])
```

```
## gn_invoke_set=['abc'] is not string or an integer between 1 and 11
```

3.1.2 Function Arguments

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import pprint
```

3.1.2.1 Mutable Argument Default

If a parameter is a list, that is mutable, or a dict. Default values for the mutable parameter should be None, with the actual default value provided inside the function.

```
# Mutable dict as parameters
def ffi_tab_txt2tex(dc_fmd_sym_sig=None, dc_tex_sig_sym=None):
    if dc_fmd_sym_sig is None:
        # defaults
        dc_fmd_sym_sig = {'***': 1e-2, '**': 5e-2, '*': 1e-1}
    if dc_tex_sig_sym is None:
        # defaults
        dc_tex_sig_sym = {'1e-3': '\\sym{***}',
                          '1e-2': '\\sym{**}',
                          '5e-2': '\\sym{*}',
                          '1e-1': '\\dagger'}

    # print
    print(f'{dc_fmd_sym_sig=}')
```

```
    print(f'{dc_tex_sig_sym=}')

# Call function with default
ffi_tab_txt2tex()
# Call function update the first dict
```

```
## dc_fmd_sym_sig={'***': 0.01, '**': 0.05, '*': 0.1}
## dc_tex_sig_sym={'1e-3': '\\sym{***}', '1e-2': '\\sym{**}', '5e-2': '\\sym{*}', '1e-1': '\\dagger'
ffi_tab_txt2tex(dc_fmd_sym_sig = {'***': 1e-3, '**': 1e-2, '*': 5e-2})

## dc_fmd_sym_sig={'***': 0.001, '**': 0.01, '*': 0.05}
## dc_tex_sig_sym={'1e-3': '\\sym{***}', '1e-2': '\\sym{**}', '5e-2': '\\sym{*}', '1e-1': '\\dagger'
```

see "Least Astonishment" and the Mutable Default Argument.

3.1.2.2 Python Dictionary As Argument via kwargs

There is a python function that outputs a dictionary with key and value pairs that specify key aspects of how a model should be solved. For example, one of the parameters could specify the *vcpu* requirement. This *vcpu* requirement might change, and so it should be easy to update this key with alternative values.

These are accomplished in the following manner. Define the full key-value pair list, with default values for several dictionaries, with model simulation, support, and compute parameters for example. These lists could be updated with some default alternative combinations, or alternatively, it could be updated with externally provided dictionary with both updated values for existing keys, or even additional key value pairs.

First, we create a function that processes and outputs default parameters, it has two inputs, *it_default_group* to specify pre-fixed adjustments from defaults, and *kwargs* that allows for arbitrarily modifications and additions to parameter dictionary.

```
def gen_compesti_spec(it_default_group=None, **kwargs):
    # A. Define the default parameter keys and values
    esti_specs = {'esti_method': 'MomentsSimuStates',
                  'momsets_type': ['a', '20180805a'],
                  'esti_param_vec_count': 1,
                  'esti_max_func_eval': 10,
                  'graph_frequncy': 20}
    compute_specs = {'cpu': str(1024 * 1),
                     'memory': str(517),  # only need about 160 mb in reality
                     'workers': 1,
                     'aws_fargate': False}

    # B. For different
    compesti_specs = {**compute_specs, **esti_specs}

    # C. Update dictionaries with parameter group values
    if it_default_group == 1:
        compesti_specs_updates = {'memory': str(1024 * 55),
                                  'compute_param_vec_count': 6,
                                  'esti_param_vec_count': 640}
        compesti_specs.update(compesti_specs_updates)

    # D. Update with kward, could append new
    compesti_specs.update(kwargs)

    return compesti_specs
```

Second, we test the defaults:

```
compesti_specs = gen_compesti_spec()
pprint.pprint(compesti_specs, width=1)
```

```
## {'aws_fargate': False,
##  'cpu': '1024',
##  'esti_max_func_eval': 10,
##  'esti_method': 'MomentsSimuStates',
##  'esti_param_vec_count': 1,
##  'graph_frequncy': 20,
##  'memory': '517',
##  'momsets_type': ['a',
##                   '20180805a'],
##  'workers': 1}
```

Third, we test using default group 1, pre-fixed changes to defaults:

```
compesti_specs = gen_compesti_spec(it_default_group=1)
pprint.pprint(compesti_specs, width=1)
```

```
## {'aws_fargate': False,
##  'compute_param_vec_count': 6,
##  'cpu': '1024',
##  'esti_max_func_eval': 10,
##  'esti_method': 'MomentsSimuStates',
##  'esti_param_vec_count': 640,
##  'graph_frequncy': 20,
##  'memory': '56320',
##  'momsets_type': ['a',
##                   '20180805a'],
##  'workers': 1}
```

Fourth, we use kwargs to feed in arbitrary dictionary to update and append to existing parameter dictionary:

```
compesti_specs_updates = {'esti_method': 'MomentsSimuStateszzz',
                          'moments_type': ['a', '20180805azzz'],
                          'momsets_type': ['a', '20180805azzz'],
                          'momsets_type_uuu': ['a', '20180805azzz']}
compesti_specs = gen_compesti_spec(it_default_group=None, **compesti_specs_updates)
pprint.pprint(compesti_specs, width=1)
```

```
## {'aws_fargate': False,
##  'cpu': '1024',
##  'esti_max_func_eval': 10,
##  'esti_method': 'MomentsSimuStateszzz',
##  'esti_param_vec_count': 1,
##  'graph_frequncy': 20,
##  'memory': '517',
##  'moments_type': ['a',
##                   '20180805azzz'],
##  'momsets_type': ['a',
##                   '20180805azzz'],
##  'momsets_type_uuu': ['a',
##                       '20180805azzz'],
##  'workers': 1}
```

3.1.2.3 Named Argument List and Dictionary

Define a function with named and unnamed arguments:

```
def gen_compesti_spec_named(it_default_group, esti_method, memory=123, graph_frequncy=10):
    # A. Define the default parameter keys and values
    esti_specs = {'esti_method': 'MomentsSimuStates',
                  'momsets_type': ['a', '20180805a'],
                  'it_default_group': it_default_group,
                  'esti_param_vec_count': 1,
                  'esti_max_func_eval': 10,
                  'graph_frequncy': graph_frequncy}
    compute_specs = {'cpu': str(1024 * 1),
                     'memory': str(memory),  # only need about 160 mb in reality
                     'workers': 1,
                     'aws_fargate': False}

    # B. For different
    compesti_specs = {**compute_specs, **esti_specs}

    return compesti_specs
```

Provide inputs for the first two unnamed parameters explicitly. Then provided the two named parameters via a dictionary:

```
dc_inputs = {'memory':12345, 'graph_frequncy':2}
compesti_specs = gen_compesti_spec_named(None, 'MomentsSimuStates', **dc_inputs)
pprint.pprint(compesti_specs, width=1)
```

```
## {'aws_fargate': False,
##  'cpu': '1024',
##  'esti_max_func_eval': 10,
##  'esti_method': 'MomentsSimuStates',
##  'esti_param_vec_count': 1,
##  'graph_frequncy': 2,
##  'it_default_group': None,
##  'memory': '12345',
##  'momsets_type': ['a',
##                   '20180805a'],
##  'workers': 1}
```

3.1.3 Python Command-line Arguments Parsing

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import pprint
import argparse
```

3.1.3.1 Positional and Optional Arguments

Provide a positional and an optional argument. Position arguments are provided positions when the module is called, without prefixed by which parameter this is. Optional argument requires parameter specification.

```
# Start parser for arguments
parser = argparse.ArgumentParser()

# Positional argument 1st, will be stored as int
parser.add_argument('esrtype', type=int, help='positional argument 1st')
# Positional argument 2nd, will be stored as string
```

```
## _StoreAction(option_strings=[], dest='esrtype', nargs=None, const=None, default=None, type=<class
```

```
parser.add_argument('speckey', type=str, help='positional argument 2nd')

# Optional argument
```

```
## _StoreAction(option_strings=[], dest='speckey', nargs=None, const=None, default=None, type=<class
```

```
parser.add_argument('-A', type=str, default='opt_arg_A_default_str_val')

# Call with positional argument specified
# Note that one is bracketed, will be interpreted as int
```

```
## _StoreAction(option_strings=['-A'], dest='A', nargs=None, const=None, default='opt_arg_A_default_
```

```
print(f"Must specify posi. arg: {parser.parse_args(['1', 'mpoly_1=esti_tinytst_mpoly_13=3=3'])=}")
# Call with two positional arguments and one optional
# Note that the first positional argument becomes int, second beomce str
```

```
## Must specify posi. arg: parser.parse_args(['1', 'mpoly_1=esti_tinytst_mpoly_13=3=3'])=Namespace(A
```

```
print(f"With opt arg: {parser.parse_args(['1', '2', '-A', 'abc'])=}")
```

```
## With opt arg: parser.parse_args(['1', '2', '-A', 'abc'])=Namespace(A='abc', esrtype=1, speckey='2
```

3.1.3.2 Short and Long Parameter Name Specifications

Test below a boolean parameter that will be true or false. The default value is False. The parameter is called *boolparam* with short name *abc*. There is a variety of ways of setting the parameter to true.

```
# Start parser for arguments
parser = argparse.ArgumentParser()

# short name for the first parameter is a, full name is abc, boolean parameter
parser.add_argument('-abc', '--boolparam', action="store_true", default=False)

# default is false but turn on option so true
```

```
## _StoreTrueAction(option_strings=['-abc', '--boolparam'], dest='boolparam', nargs=0, const=True, d
```

```
print(f"default false: {parser.parse_args()=}")
```

```
## default false: parser.parse_args()=Namespace(boolparam=False)
```

```
print(f"default false, set to true, short all: {parser.parse_args(['-abc'])=}")
```

```
## default false, set to true, short all: parser.parse_args(['-abc'])=Namespace(boolparam=True)
```

```
print(f"default false, set to true, short part ab for abc: {parser.parse_args(['-ab'])=}")
```

```
## default false, set to true, short part ab for abc: parser.parse_args(['-ab'])=Namespace(boolparam
```

```
print(f"default false, set to true, short part a for abc: {parser.parse_args(['-a'])=}")
```

```
## default false, set to true, short part a for abc: parser.parse_args(['-a'])=Namespace(boolparam=T
```

```
print(f"default false, set to true, full param: {parser.parse_args(['--boolparam'])=}")
```

```
## default false, set to true, full param: parser.parse_args(['--boolparam'])=Namespace(boolparam=Tr
```

```
print(f"default false, set to true, full param: {parser.parse_args(['--boolparam'])=}")
```

```
## default false, set to true, full param: parser.parse_args(['--boolparam'])=Namespace(boolparam=Tr
```

3.1.3.3 A List of Allowed Values

There is a parameter, only some specific values are allowed. Also provide help for each allowed option. Note added *argparse.RawTextHelpFormatter* to parse the next lines in help.

```
# Start parse
parser = argparse.ArgumentParser(description='Run ESR cmd', formatter_class=argparse.RawTextHelpForm

# A required positional argument parameter tht is int and can take eight possible values
parser.add_argument('esrtype', type=int,
                    choices=[1, 2, 3, 4, 5, 6, 7, 8],
                    help='1. Simulate at N sets of parameter combinations\n'
                         '2. Polynomial approximation surface based on (1) '
                         'for each outcome of interest, find best\n'
                         '3. Estimation at N sets of starting points with (2) as objective function\
                         '4. Gather results frorm (3), find M best.\n'
                         '5. Simulate (estimate once) at the top M best results from (4) actual mode
                         'compare objective to approximated from (3)\n'
                         '6. Gather results from (5), re-rank best of the M best from (4)\n'
                         '7. Estimate at the top M best results from (4) actual model, '
                         '(4) M best are M best seeds\n'
                         '8. Gather results from (7), re-rank best of the final results from the M b
# Print defaults
```

```
## _StoreAction(option_strings=[], dest='esrtype', nargs=None, const=None, default=None, type=<class
```

```
print(f"provide 1 for the value of the positional argument: {parser.parse_args(['1'])=}")
```

```
## provide 1 for the value of the positional argument: parser.parse_args(['1'])=Namespace(esrtype=1)
```

3.1.3.4 Boolean, Integer, String and list Parameters

How to handle parameters of different types, boolean, integer, string and list. For these four types, the same way to specify short and long parameter names. How to set the parameter types, and how to set default values for each type.

```
# Start parser for arguments
parser = argparse.ArgumentParser()

# Single letter string parameters
# Note dest name over-rides full name
parser.add_argument('-cta', '--cttaaaaa', dest="combo_type_a", default='e',
                    type=str)
# Multiple letters and integers
# Note without dest full name is dest
```

```
## _StoreAction(option_strings=['-cta', '--cttaaaaa'], dest='combo_type_a', nargs=None, const=None,
```

```
parser.add_argument('-ctb', '--combo_type_b', default='20201025',
                    type=str)
# Multiple letters and integers
# Note without dest and full name short name is parameter name
```

```
## _StoreAction(option_strings=['-ctb', '--combo_type_b'], dest='combo_type_b', nargs=None, const=No
```

```
parser.add_argument('-ctc', default=['list_tKap_mlt_ce1a2'],
                    nargs='+', type=str)

# Print defaults
```

```
## _StoreAction(option_strings=['-ctc'], dest='ctc', nargs='+', const=None, default=['list_tKap_mlt_
```

```
print(f"default false: {parser.parse_args()=}")
# change parameters
```

```
## default false: parser.parse_args()=Namespace(combo_type_a='e', combo_type_b='20201025', ctc=['lis
```

```
print(f"default false: {parser.parse_args(['-ctb', '20201111'])=}")
```

```
## default false: parser.parse_args(['-ctb', '20201111'])=Namespace(combo_type_a='e', combo_type_b='
```

see variable-argument-lists.

3.1.3.5 Parse multiple parameter types

Provide several types of parameters to a function, so that the function can be called easily container call to execute estimation. The types of parameters includes:

1. A list including parameter information
2. A string including estimation/computational controls
3. Additional parameters

```
# Start parser for arguments
parser = argparse.ArgumentParser()

# First (and only) positional argument for esrtype:
parser.add_argument('esrtype', type=int, help='positional argument')

# Optional argument
```

```
## _StoreAction(option_strings=[], dest='esrtype', nargs=None, const=None, default=None, type=<class
```

```
parser.add_argument('-s', dest='speckey', type=str,
                    default='ng_s_t=esti_tinytst_thin_1=3=3',
                    help="compute and esti keys and omments")

# abc and e of comb_type
```

```
## _StoreAction(option_strings=['-s'], dest='speckey', nargs=None, const=None, default='ng_s_t=esti_
```

```
parser.add_argument('-cta', dest="combo_type_a", default='e', type=str)
```

```
## _StoreAction(option_strings=['-cta'], dest='combo_type_a', nargs=None, const=None, default='e', t
```

```
parser.add_argument('-ctb', dest="combo_type_b", default='20201025', type=str)
```

```
## _StoreAction(option_strings=['-ctb'], dest='combo_type_b', nargs=None, const=None, default='20201
```

```
parser.add_argument('-ctc', dest="combo_type_c", default=['list_tKap_mlt_ce1a2'], nargs='+', type=st
```

```
## _StoreAction(option_strings=['-ctc'], dest='combo_type_c', nargs='+', const=None, default=['list_
```

```
parser.add_argument('-cte1', dest="combo_type_e1", default=None, type=str)
```

```
## _StoreAction(option_strings=['-cte1'], dest='combo_type_e1', nargs=None, const=None, default=None
```

```
parser.add_argument('-cte2', dest="combo_type_e2", default='mpoly_1=esti_tinytst_mpoly_13=3=3', type

# other parameters
```

```
## _StoreAction(option_strings=['-cte2'], dest='combo_type_e2', nargs=None, const=None, default='mpo
```

```
parser.add_argument('-f', dest="save_directory_main", default='esti')

# Default, must specify erstype
```

```
## _StoreAction(option_strings=['-f'], dest='save_directory_main', nargs=None, const=None, default='
```

```
print(f"default false: {parser.parse_args(['1'])=}")
# Print with the nargs+ arguments
# specified two elements, abc, and efg for nargs ctc, becomes a string list
```

```
## default false: parser.parse_args(['1'])=Namespace(combo_type_a='e', combo_type_b='20201025', comb
```

```
print(f"default false: {parser.parse_args(['1', '-ctc', 'abc', 'efg'])=}")
# one input for ctc, still generates a list
```

```
## default false: parser.parse_args(['1', '-ctc', 'abc', 'efg'])=Namespace(combo_type_a='e', combo_t
```

```
print(f"default false: {parser.parse_args(['1', '-ctc', 'abc'])=}")
```

```
## default false: parser.parse_args(['1', '-ctc', 'abc'])=Namespace(combo_type_a='e', combo_type_b='
```

3.1.4 Function Returns

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

3.1.4.1 Function with Multiple Optional Returns

There is a function that is already written, that returns some string value. Without interrupting the existing function, now add an additional return for the function. There is some conditional statement that controls whether the function returns one or two value.

In the example below, if the path contains

Create a testing function:

```
def get_val(spn_path):
    if 'G:' in spn_path:
        # this returns a tuple of length 2
        return spn_path, 'G:/repos'
    else:
        return spn_path
```

Call the function with one return, in the two calls below, the first call returns a tuple

```
# Return tuple
tp_st_return = get_val("G:/Dropbox (UH-ECON)")
# Unpack tuple return
st_path_a, st_path_b = get_val("G:/Dropbox (UH-ECON)")
# Single element return
st_return = get_val("C:/Dropbox (UH-ECON)")
# Print
print(f'{tp_st_return=} and {st_return=}')
```

```
## tp_st_return=('G:/Dropbox (UH-ECON)', 'G:/repos') and st_return='C:/Dropbox (UH-ECON)'
```

```
print(f'{st_path_a=} and {st_path_b=}')
```

```
## st_path_a='G:/Dropbox (UH-ECON)' and st_path_b='G:/repos'
```

```
print(f'{isinstance(tp_st_return, str)=}')
```

```
## isinstance(tp_st_return, str)=False
```

```
print(f'{isinstance(tp_st_return, tuple)=}')
```

```
## isinstance(tp_st_return, tuple)=True
```

```
print(f'{isinstance(st_return, str)=}')
```

```
## isinstance(st_return, str)=True
```

```
print(f'{isinstance(st_return, tuple)=}')
```

```
## isinstance(st_return, tuple)=False
```

3.2 Exceptions

3.2.1 Exception Handling

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

3.2.1.1 A function That Raises an Error with Try Statement Catching it

Below, we have a function that will raise a *TypeError* unless we provide an integer input. The function is called with integer input and then called with a string input. The string input is wrapped in a try and except call, where the exception catches the TypeError and prints it.

```
# define the function
def ffi_error_test(gn_speckey=None):
    if isinstance(gn_speckey, int):
        print(f'{gn_speckey=} is an integer')
    else:
        raise TypeError(f'{gn_speckey=} is not an integer')
# Call function with integer
error_test(1)
# Call function with string
```

```
## gn_speckey=1 is an integer
```

```
try:
    ffi_error_test('abc')
except TypeError as error:
    print('Caught this error: ' + repr(error))
```

```
## Caught this error: TypeError("gn_speckey='abc' is not an integer")
```

3.2.1.2 Catch an Exception Noisily

Rather than only catching the last element of the exception, show the full trace. Notice in the example below, the second element of the loop does not work as function input. With exception catching, the loop continued despite the second element not working. The traceback shows more details at the end with full trace of the exception from the second element of the list as input for ffi_error_test().

```
import traceback
import numpy as np

ls_ob_inputs = [2, ['abc','efg'], 1, 123]

for ob_input in ls_ob_inputs:
    print(f'try input {ob_input=} with the error_test function:')
    try:
        ffi_error_test(ob_input)
    except TypeError as error:
        traceback.print_exc()
        print('Caught this error: ' + repr(error))
```

```
## try input ob_input=2 with the error_test function:
## gn_speckey=2 is an integer
## try input ob_input=['abc', 'efg'] with the error_test function:
## Caught this error: TypeError("gn_speckey=['abc', 'efg'] is not an integer")
## try input ob_input=1 with the error_test function:
## gn_speckey=1 is an integer
## try input ob_input=123 with the error_test function:
## gn_speckey=123 is an integer
##
## Traceback (most recent call last):
```

```
##   File "<string>", line 4, in <module>
##   File "<string>", line 5, in ffi_error_test
## TypeError: gn_speckey=['abc', 'efg'] is not an integer
```

see How to print the stack trace of an exception object in Python?

3.2.1.3 Handle Parameters When Conditions Not Satisfied

There is a function, that can estimate or simulate, under both functionalities, there is a common string parameter, that requires specifying estimation or simulation conditions. The common string parameter should be a simple string without special separators in the case of simulation, and should be four strings concatenated together with equal sign for estimation. Generate an exception if the function is called for estimation but the string parameter does not have the required structure.

- Python 3 TypeError
- Manually raising (throwing) an exception in Python

```
# ls_st_spec_key_dict = ['NG_S_D', 'NG_S_D=KAP_MO_NLD_M_SIMU=2=3']
# st_connector = '='
# ls_st_esti_simu = ['esti', 'simu']
# for st_spec_key_dict in ls_st_spec_key_dict:
#   for st_esti_simu in ls_st_esti_simu:
#     if st_esti_simu == 'simu':
#       if len(st_spec_key_dict.split(st_connector)) and
#         print('simulate with ' + st_spec_key_dict)

if estimate and not isinstance(spec_key_dict, str):
    pass
elif (estimate is False and isinstance(spec_key_dict, str)) or (estimate is False and isinstance(spe
    pass
else:
    st_error = 'speckey=' + speckey + ' and estimate=' + str(estimate)
    raise ValueError(st_error)
```

3.2.1.4 Proceed Despite Error

Sometimes, code should proceed despite error, to finish a loop for example:

```
# estimate at each initial random points
for it_esti_ctr in range(esti_param_vec_count):
    # Update the 3rd element of combo_type, which determines which draw index to use
    combo_type[3] = it_esti_ctr
    try:
        invoke_run_main.invoke_main(combo_type, **dc_invoke_main_args)
    except Exception:
        logging.critical(f'Finished this {it_esti_ctr=} of {range(esti_param_vec_count)=}')
```

Chapter 4

Statistics

4.1 Markov Process

4.1.1 Close Value Comparison

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import sys as sys
import numpy as np
import pprint
```

4.1.1.1 Generate a Markov Transition Matrix

With numerical probability approximations, discrete random variables often are not exact, given an AR(1) transition matrix, how to check whether the conditional probabilities conditional on each current state (row) sums up to one?

First define the AR(1) transition matrix, which is from a model of asset transitions. This is a 50 by 50 matrix.

```
mt_ar1_trans = np.array([[0.4334, 0.5183, 0.0454, 0.0027, 0.0002, 0., 0., 0., 0., 0., 0.,0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0.2624,
0.5967, 0.1245, 0.0145, 0.0016, 0.0002, 0., 0., 0., 0., 0.,0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0.1673, 0.5918, 0.2005,
0.0343, 0.0052, 0.0008, 0.0001, 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0.0312, 0.6497, 0.2774, 0.0371,
0.0041, 0.0005, 0.0001, 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0.0681, 0.6569, 0.2379, 0.0327,
0.0038, 0.0005, 0.0001, 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0.2201, 0.581 , 0.168 , 0.0264, 0.0038,
0.0006, 0.0001, 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0.,
0., 0., 0., 0.], [0., 0., 0., 0.0044, 0.4312, 0.4356, 0.1069, 0.0183, 0.003 ,
0.0005, 0.0001,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0.,
0., 0., 0.], [0., 0., 0., 0., 0.1408, 0.5552, 0.2403, 0.052 , 0.0095, 0.0017,
0.0003,0.0001, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0.,
```

```
0., 0.], [0., 0., 0., 0., 0.1666, 0.5902, 0.2019, 0.0349, 0.0053, 0.0008,
0.0001,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0.,
0.], [0., 0., 0., 0., 0., 0.2065, 0.5668, 0.1869, 0.0335, 0.0053, 0.0009,0.0001,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0.,
0., 0., 0., 0., 0.2414, 0.5453, 0.1748, 0.0321, 0.0053,0.0009, 0.0002, 0., 0.,
0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0.,
0.0172, 0.4156, 0.4019, 0.1281, 0.0293, 0.0062,0.0013, 0.0003, 0.0001, 0., 0.,
0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0.,
0.0033, 0.3222, 0.4594, 0.1655, 0.0391,0.0083, 0.0017, 0.0004, 0.0001, 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0.,
0.0111, 0.3744, 0.4215, 0.1471,0.0357, 0.0079, 0.0017, 0.0004, 0.0001, 0., 0.,
0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0.,
0.1555, 0.504 , 0.2526,0.0682, 0.0154, 0.0034, 0.0008, 0.0002, 0., 0., 0., 0.,
0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0.1987,
0.5249, 0.215 ,0.0493, 0.0097, 0.0019, 0.0004, 0.0001, 0., 0., 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0.0041,
0.3513, 0.4533,0.1499, 0.0331, 0.0066, 0.0013, 0.0003, 0.0001, 0., 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.0247,
0.441 ,0.3835, 0.1174, 0.0264, 0.0055, 0.0012, 0.0003, 0.0001, 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.1049,0.502 , 0.2887, 0.0809, 0.0183, 0.004 , 0.0009, 0.0002, 0.0001, 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.0003,0.1962, 0.4888, 0.2319, 0.0636, 0.0148, 0.0034, 0.0008, 0.0002, 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0.0087, 0.3362, 0.4294, 0.1674, 0.0444, 0.0105, 0.0025, 0.0006, 0.0001, 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0.0001, 0.154 , 0.4801, 0.2617, 0.0785, 0.0195, 0.0046, 0.0011, 0.0003,
0.0001, 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0.,0.0001, 0.1467, 0.48  , 0.2664, 0.0804, 0.02  , 0.0048, 0.0012, 0.0003,
0.0001, 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0.,0., 0.0149, 0.3543, 0.4103, 0.1614, 0.0444, 0.011 , 0.0027, 0.0007,
0.0002, 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0.,0., 0., 0.0438, 0.4122, 0.3611, 0.1334, 0.0369, 0.0094, 0.0024, 0.0006,
0.0002,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0., 0., 0., 0.1159, 0.452 , 0.2939, 0.1007, 0.0278, 0.0072, 0.0019,
0.0005,0.0001, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0.,0., 0., 0., 0.0012, 0.198 , 0.4463, 0.2436, 0.0804, 0.0224, 0.0059,
0.0016,0.0004, 0.0001, 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0.,
```

```
0., 0., 0.,0., 0., 0., 0., 0.0087, 0.2821, 0.4181, 0.2011, 0.0649, 0.0183, 0.005
,0.0014, 0.0004, 0.0001, 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0.,0., 0., 0., 0., 0., 0.0842, 0.4174, 0.3217, 0.1244, 0.0376,
0.0105,0.0029, 0.0008, 0.0002, 0.0001, 0., 0., 0., 0., 0., 0., 0.,0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.0828, 0.4166, 0.3229, 0.1251, 0.0379,
0.0106,0.0029, 0.0008, 0.0002, 0.0001, 0., 0., 0., 0., 0., 0., 0.,0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.0119, 0.2907, 0.4075, 0.1979, 0.0656,
0.019 ,0.0053, 0.0015, 0.0004, 0.0001, 0., 0., 0., 0., 0., 0., 0.,0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.0424, 0.3654, 0.3618, 0.1581,
0.0513,0.015 , 0.0043, 0.0012, 0.0004, 0.0001, 0., 0., 0., 0., 0., 0.,0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.0001, 0.1078, 0.4112, 0.3043,
0.1217,0.0388, 0.0114, 0.0033, 0.001 , 0.0003, 0.0001, 0., 0., 0., 0., 0.,0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.0015, 0.1714, 0.4155,
0.264 ,0.1015, 0.0323, 0.0097, 0.0029, 0.0009, 0.0003, 0.0001, 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0.0093,
0.2496, 0.4001,0.2214, 0.082 , 0.0261, 0.0079, 0.0024, 0.0007, 0.0002, 0.0001,
0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0.041 , 0.3374,0.3591, 0.1729, 0.0615, 0.0195, 0.006 , 0.0018, 0.0006, 0.0002,
0.0001, 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0.,
0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0.0295, 0.315 ,0.3712, 0.186 , 0.0673, 0.0215, 0.0066, 0.002 , 0.0006,
0.0002, 0.0001, 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0.,
0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.,
0., 0., 0., 0.0295, 0.315 ,0.3712, 0.186 , 0.0673, 0.0215, 0.0066, 0.002 ,
0.0006, 0.0002, 0.0001, 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0.,
0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0.,
0., 0., 0., 0.0036, 0.1927,0.4021, 0.2527, 0.1004, 0.0334, 0.0104,
0.0032, 0.001 , 0.0003, 0.0001, 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0.,
0., 0., 0., 0., 0., 0., 0.018 ,0.2749, 0.3813, 0.2082, 0.0792, 0.0262,
0.0083, 0.0026, 0.0008, 0.0003, 0.0001, 0.,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0.0539, 0.3414, 0.344 , 0.1684, 0.0621,
0.0205, 0.0065, 0.0021, 0.0007, 0.0002, 0.0001,0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0.0005, 0.1155, 0.3788, 0.2986,
0.1345, 0.0485, 0.016 , 0.0052, 0.0017, 0.0006, 0.0002,0.0001, 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0.0042, 0.1819, 0.385
, 0.259 , 0.1109, 0.0395, 0.0132, 0.0043, 0.0014, 0.0005,0.0002, 0.0001, 0., 0.,
0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0.0183,
0.2571, 0.3709, 0.2174, 0.089 , 0.0314, 0.0105, 0.0035, 0.0012,0.0004, 0.0001,
0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0.,
0.0041, 0.1763, 0.3796, 0.2619, 0.1149, 0.0419, 0.0142, 0.0047, 0.0016,0.0005,
0.0002, 0.0001, 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.], [0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,0., 0., 0.0028, 0.1598, 0.3781, 0.2712, 0.1209, 0.0444, 0.0151, 0.005 ,
0.0017,0.0006, 0.0002, 0.0001, 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0.,
```

```
0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0.,0., 0., 0., 0.0414, 0.3051, 0.3488, 0.1888, 0.0756, 0.0267, 0.009
, 0.003 ,0.001 , 0.0004, 0.0001, 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0.,
0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0.,0., 0., 0., 0.0002, 0.0784, 0.3412, 0.3202, 0.1621, 0.0637,
0.0225, 0.0077,0.0026, 0.0009, 0.0003, 0.0001, 0., 0., 0., 0., 0., 0., 0.,0.,
0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0.0018, 0.1352, 0.3634, 0.2843, 0.135
, 0.0521, 0.0184,0.0063, 0.0022, 0.0008, 0.0003, 0.0001, 0., 0., 0., 0., 0.,
0.,0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0.,0., 0., 0., 0., 0., 0.0111, 0.2115, 0.365 ,
0.2415, 0.1079, 0.0409,0.0145, 0.005 , 0.0017, 0.0006, 0.0002, 0.0001, 0., 0.,
0., 0., 0.,0., 0., 0., 0., 0., 0.]])
# print
print(f'{mt_ar1_trans.shape=}')
```

```
## mt_ar1_trans.shape=(50, 50)
```

What are the row sums, and how close is each to 1:

```
# Sum for each row across columns
ar_row_sums_ar1 = np.sum(mt_ar1_trans, axis=1)
fl_sum_to_match = 1
# print
print(f'{ar_row_sums_ar1=}')
```

```
## ar_row_sums_ar1=array([1.    , 0.9999, 1.    , 1.0001, 1.    , 1.    , 1.    , 0.9999,
##        0.9998, 1.    , 1.    , 1.    , 1.    , 0.9999, 1.0001, 1.    ,
##        1.    , 1.0001, 1.    , 1.    , 0.9998, 1.    , 1.    , 0.9999,
##        1.    , 1.    , 0.9999, 1.0001, 0.9998, 0.9999, 0.9999, 1.    ,
##        1.    , 1.0001, 0.9998, 1.0001, 1.    , 1.    , 0.9999, 0.9999,
##        0.9999, 1.0002, 1.0002, 0.9998, 1.    , 0.9999, 0.9999, 0.9999,
##        0.9999, 1.    ])
```

```
print(f'Total Avg Diff = {np.mean(abs(ar_row_sums_ar1-fl_sum_to_match))=}')
```

```
## Total Avg Diff = np.mean(abs(ar_row_sums_ar1-fl_sum_to_match))=6.800000000001916e-05
```

4.1.1.2 Check Row by Row Tolerance

Check for tolerance, *rtol* is the absolute base gap allowed, *atol* is multipled by fl_sum_to_match, below, we can check row by row, and see if any rows does not meet the approximately equality condition. If there is any one row that does not match this, then set condition for overall matrix checking to false.

```
fl_atol = 1e-08
print(f'with {fl_atol=}, many rows do not approximately equal to 1')
```

```
## with fl_atol=1e-08, many rows do not approximately equal to 1
```

```
print(f'{[np.allclose(fl_row_sum, fl_sum_to_match, rtol=0, atol=fl_atol) for fl_row_sum in ar_row_su
```

```
## [np.allclose(fl_row_sum, fl_sum_to_match, rtol=0, atol=fl_atol) for fl_row_sum in ar_row_sums_ar1
```

```
print(f'{np.allclose(ar_row_sums_ar1, fl_sum_to_match, rtol=0, atol=fl_atol)=}')
```

```
## np.allclose(ar_row_sums_ar1, fl_sum_to_match, rtol=0, atol=fl_atol)=False
```

```
fl_atol = 1e-03
print(f'With {fl_atol=}, all rows approximately equal to 1')
```

```
## With fl_atol=0.001, all rows approximately equal to 1
```

```
print(f'{[np.allclose(fl_row_sum, fl_sum_to_match, rtol=0, atol=fl_atol) for fl_row_sum in ar_row_su
```

```
## [np.allclose(fl_row_sum, fl_sum_to_match, rtol=0, atol=fl_atol) for fl_row_sum in ar_row_sums_ar1
```

```
print(f'{np.allclose(ar_row_sums_ar1, fl_sum_to_match, rtol=0, atol=fl_atol)=}')
```

```
## np.allclose(ar_row_sums_ar1, fl_sum_to_match, rtol=0, atol=fl_atol)=True
```

4.1.1.3 Check Joint Tolerance

For practical usages, we set joint condition. The difference between the sum of any row and 1 should be less than a threshold, additionally, the total average gap between row sums and 1 should be less than a threshold. Do this so that can have a more relaxed per-row tolerance requirement, and more stringent average requirement. Setting only a stringent per row requirement might be too restive, and reject transition matrixes that do not have problems.

```
# criteria
fl_atol_per_row = 1e-03
fl_atol_avg_row = 1e-04
# per-row check
bl_per_row_pass = np.allclose(ar_row_sums_ar1, fl_sum_to_match, rtol=0, atol=fl_atol_per_row)
# all-rows check
fl_row_gap_mean = np.mean(abs(ar_row_sums_ar1-fl_sum_to_match))
bl_avg_row_pass = np.allclose(fl_row_gap_mean+fl_sum_to_match, fl_sum_to_match, rtol=0, atol=fl_atol
# Joint
bl_ar1_sum_pass = bl_per_row_pass and bl_avg_row_pass
# Print
print(f'{bl_per_row_pass=}')
```

```
## bl_per_row_pass=True
```

```
print(f'{bl_avg_row_pass=} and {fl_row_gap_mean=}')
```

```
## bl_avg_row_pass=True and fl_row_gap_mean=6.800000000001916e-05
```

```
print(f'{bl_ar1_sum_pass=}')
```

```
## bl_ar1_sum_pass=True
```

4.1.1.4 Normalize Transition Matrix Row Sum

Having pass the checks, would like to have conditional probability sum up to 1, so "normalize". Division by broadcast, reshape the sum for each row as column, divide all columns for the same row by the same sum value. New sum of normalized ar(1) for each row is now equal to 1.

```
# current row sums
ar_row_sums_ar1 = np.sum(mt_ar1_trans, axis=1)
ar_row_sums_ar1 = np.reshape(ar_row_sums_ar1, [-1, 1])
# Update row values
mt_ar1_trans_norm = mt_ar1_trans/np.reshape(ar_row_sums_ar1, [-1, 1])
ar_row_sums_ar1_norm = np.sum(mt_ar1_trans_norm, axis=1)
ar_row_sums_ar1_norm = np.reshape(ar_row_sums_ar1_norm, [-1, 1])
# check sum
print(f'{np.column_stack([ar_row_sums_ar1, ar_row_sums_ar1_norm])=}')
```

```
## np.column_stack([ar_row_sums_ar1, ar_row_sums_ar1_norm])=array([[1.    , 1.    ],
##        [0.9999, 1.    ],
##        [1.    , 1.    ],
##        [1.0001, 1.    ],
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [0.9999, 1.    ],
##        [0.9998, 1.    ],
##        [1.    , 1.    ],
```

```
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [0.9999, 1.    ],
##        [1.0001, 1.    ],
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [1.0001, 1.    ],
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [0.9998, 1.    ],
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [0.9999, 1.    ],
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [0.9999, 1.    ],
##        [1.0001, 1.    ],
##        [0.9998, 1.    ],
##        [0.9999, 1.    ],
##        [0.9999, 1.    ],
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [1.0001, 1.    ],
##        [0.9998, 1.    ],
##        [1.0001, 1.    ],
##        [1.    , 1.    ],
##        [1.    , 1.    ],
##        [0.9999, 1.    ],
##        [0.9999, 1.    ],
##        [0.9999, 1.    ],
##        [1.0002, 1.    ],
##        [1.0002, 1.    ],
##        [0.9998, 1.    ],
##        [1.    , 1.    ],
##        [0.9999, 1.    ],
##        [0.9999, 1.    ],
##        [0.9999, 1.    ],
##        [0.9999, 1.    ],
##        [1.    , 1.    ]])
```

Chapter 5

Tables and Graphs

5.1 Matplotlib Base Plots

5.1.1 Line and Scatter Plots

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

5.1.1.1 Plot Random Walk and White Noise Jointly

Given x and y coordinates, plot out two lines. see matplotlib.pyplot.plot. Here we will plot out the extremes of AR(1), white noise (no persistence), and random walk (fully persistent shocks).

```
# Import Packages
import numpy as np
import matplotlib.pyplot as plt

# Generate X and Y
np.random.seed(123)
ar_fl_y1_rand = np.random.normal(0, 2, 100)
ar_fl_y2_rand = np.cumsum(np.random.normal(0, 1, 100))
ar_it_x_grid = np.arange(1,len(ar_fl_y1_rand)+1)

# Start Figure
fig, ax = plt.subplots()

# Graph
ax.plot(ar_it_x_grid, ar_fl_y1_rand,
                      color='blue', linestyle='dashed',
                      label='sd=2, 0 persistence')
```

```
## [<matplotlib.lines.Line2D object at 0x0000017170C90C40>]
```

```
ax.plot(ar_it_x_grid, ar_fl_y2_rand,
                      color='red', linestyle='solid',
                      label='sd=1, random walk')

# Labeling
```

```
## [<matplotlib.lines.Line2D object at 0x000001716DB51190>]
```

```
ax.legend(loc='upper left')
```

```
## <matplotlib.legend.Legend object at 0x0000017170F39640>
```

```
plt.ylabel('Random Standard Normal Draws')
```

```
## Text(0, 0.5, 'Random Standard Normal Draws')
```

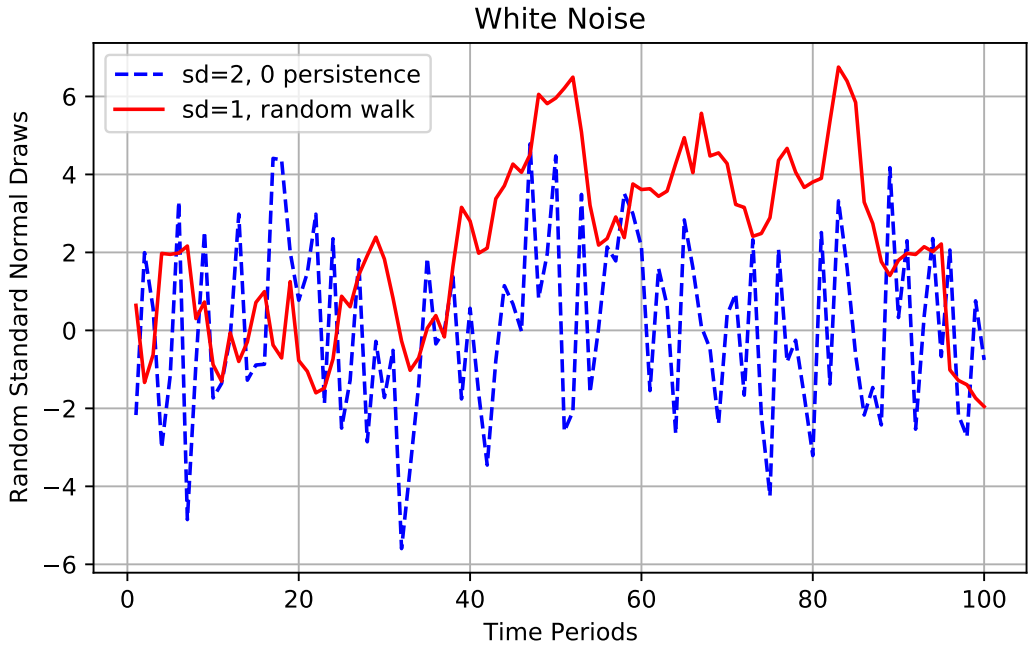
```
plt.xlabel('Time Periods')
```

```
## Text(0.5, 0, 'Time Periods')
```

```
plt.title('White Noise')
```

```
## Text(0.5, 1.0, 'White Noise')
```

```
plt.grid()
plt.show()
```



5.1.2 Text Plot

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

5.1.2.1 Plot Text

Plot Text as Image. Create text with different alignment and rotation.

```
# Import Packages
import matplotlib.pyplot as plt
import textwrap
import json

# Dict of String to String
dc_path = {'C:\\Users\\fan\\Documents\\Dropbox (UH-ECON)\\repos\\Tex4Econ\\'
           '_other\\equation\\cases.tex':
               'C:/Users/fan/Documents/cases.pdf',
           'C:\\Users\\fan\\Documents\\Dropbox (UH-ECON)\\repos\\Tex4Econ\\'
           '_other\\symbols\\fs_symbols.tex':
               'C:/Users/fan/Documents/fs_symbols.pdf'}
st_dc_path = textwrap.fill(json.dumps(dc_path), width = 70)
```

```
# Start Plot
fig, ax = plt.subplots()

# Text Plot
ax.text(0.5, 0.5, st_dc_path,
        horizontalalignment='center',
        verticalalignment='center',
        fontsize=14, color='black',
        transform=ax.transAxes)

# Labeling
```

```
## Text(0.5, 0.5, '{"C:\\\\Users\\\\fan\\\\Documents\\\\Dropbox (UH-\nECON)\\\\repos\\\\Tex4Econ\\\\
```

```
ax.set_axis_off()
plt.show()
```

{"C:\\Users\\fan\\Documents\\Dropbox (UH-ECON)\\repos\\Tex4Econ_other\\equation\\cases.tex": "C:/Users/fan/Documents/cases.pdf", "C:\\Users\\fan\\Documents\\Dropbox (UH-ECON)\\repos\\Tex4Econ_other\\symbols\\fs_symbols.tex": "C:/Users/fan/Documents/fs_symbols.pdf"}

Chapter 6

Amazon Web Services

6.1 AWS Setup

6.1.1 AWS Setup

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

6.1.1.1 Installation on Local Machine

First install anaconda, git, and associated programs.

1. Putty
2. access to .pem key
3. conda aws environment below

6.1.1.2 Conda AWS Environment

Can Start and Stop instances from Conda Prompt after this.

```
conda deactivate
conda list env
conda env remove -n wk_aws
conda create -n wk_aws -y
conda activate wk_aws

# Install External Tools
conda install -c anaconda pip -y

# Command line interface
conda install -c conda-forge awscli -y
# Programmatically send JSON instructions with boto3
conda install -c anaconda boto3 -y
```

6.1.1.3 AWS Account set-up

1. Sign-up for AWS web services account (can be the same as your Amazon shopping account)
2. Register for AWS Educate to get student or faculty voucher.

- The University of Houston is a part of AWS Educate, choose educator or student, should hear back within 24 hours with coupon code.
- UH students can get $100, faculty can get $200.

6.1.1.4 Start a AWS Instance and Link Local to Remote

Amazon has a lot of tutorials. Here is an outline.

1. Generate keypair on AWS, aws guide
   - this gives you a .pem file which you download and Amazon also remembers
   - local computers with the right .pem file can talk to your AWS instances
   - You might need to invoke the chmod command below to set permission:

```
chmod 400 "C:/Users/fan/Documents/Dropbox (UH-ECON)/Programming/AWS/fan_wang-key-pair-us_east_nv
```

2. *Launching Instance*: Go to your console, choose EC2, choose launch instance, select Amazon Linux Instance (review and launch)
3. *Instance security*: select VPC security group: I have for example: fan_wang_SG_us_east_nv_VPC (edit security group and submit)
   - Security group can allow any IP address to access your instance or just specific ones.
   - AWS has a tool here that just allows your current IP to access the EC2 instance
4. *Instance access key*: Select right keypair (your .pem key), fan_wang-key-pair-us_east_nv (prompted after submitting)
5. For SSH in, you can use Putty. aws guide
   - tell Putty your AWS instance DNS address and where your pem key is
   - Can use a Putty client to enter an EC2 instance
6. For SSH, can also do the process below:
   - open git bash (install putty before)

```
ssh-agent -s
eval $(ssh-agent -s)
```

   - Tell SSH where pem key is:

```
ssh-add "C:/Users/fan/Documents/Dropbox (UH-ECON)/Programming/AWS/fan_wang-key-pair-us_east_nv.p
```

   - You will find a public DNS address for your aws instance on the AWS user interface page

```
# ssh git bash command line
# for ubuntu machine
ssh ubuntu@ec2-54-197-6-153.compute-1.amazonaws.com
# for aws linux
ssh ec2-user@ec2-52-23-218-117.compute-1.amazonaws.com
# quit aws instance
# ctrl + D
```

   - if get: Permission denied (publickey), see:
     1. Trying to connect with the wrong key. Are you sure this instance is using this keypair?
     2. Trying to connect with the wrong username. ubuntu is the username for the ubuntu based AWS distribution, but on some others it's ec2-user (or admin on some Debians, according to Bogdan Kulbida's answer)(can also be root, fedora, see below)
     3. Trying to connect the wrong host. Is that the right host you are trying to log in to?
   - You can log in generally like this, note the instance gets new public DNS IP address every time you restart it:

```
LOCALPEM="C:/Users/fan/Documents/Dropbox (UH-ECON)/Programming/AWS/fan_wang-key-pair-us_east_nv
IPADD=34.207.250.160
REMOTEIP=ec2-user@$IPADD
ssh-keygen -R $IPADD
ssh -i "$LOCALPEM" $REMOTEIP
```

6.1.1.5 Use AWSCLI to Start and Stop an Instance

1. Install AWS CLI
2. Create individual IAM users
3. Follow instructions to Configure your awscli, and profit access key id and secrete access key when prompted.
   - do not copy and paste the Key ID and Access Key. They are example, type these in as answers given config prompt:

```
# aws configure
AWS Access Key ID [None]: XXXXIOSFODNN7EXAMPLE
```

```
AWS Secret Access Key [None]: wXalrXXtnXXXX/X7XXXXX/bXxXfiCXXXXXXXXXX
Default region name [None]: us-west-1
Default output format [None]: json
```

- this creates under a folder like this: C:/Users/fan/.aws, inside the folder these info will be stored in a configuration file.

```
# the credentials file
[default]
aws_access_key_id = XXXXIOSFODNN7EXAMPLE
aws_secret_access_key = wXalrXXtnXXXXX7XXXXXbXxXfiCXXXXXXXXXX
```

- then when you use aws cli, you will automatically be authenticated

4. Start an instance in console first (or directly in command line). Stop it. do not terminate. Now this instance will have a fixed instance ID. Its DNS IP address will change every time you restart it, but its instance ID is fixed. Instance ID is found easily in the EC2 Console.
   - Launch an instance

```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro --key-name MyK
```

   - Start an instance

```
aws ec2 start-instances --instance-ids i-XXXXXXXX
aws ec2 start-instances --instance-ids i-040c856530b2619bc
```

   - Stop an instance

```
aws ec2 stop-instances --instance-ids i-XXXXXXXX
aws ec2 stop-instances --instance-ids i-040c856530b2619bc
```

6.1.1.6 Set-up SSM on EC2 Instance

To execute commandlines etc remote on EC2, need to set up SSM: AWS Systems Manager Agent (SSM Agent)

SSM-agent is already installed in Amazon Linux.

Error Message regarding InvalidInstanceId. The following scenarios can result in this error message:

- Instance id is invalid (in the comments you have verified it isn't)
- Instance is in a different region (in the comments you have verified it isn't)
- Instance is not currently in the Running state
- Instance does not have the AWS SSM agent installed and running.

"You have to create and attach the policy AmazonSSMFullAccess to the machine (thats maybe more broad than you need) but that was why it wasn't working for me… You do that by clicking on (when selected on the ec2 instance) Action > Instance Settings > Attach/Replace IAM Role then create a role for ec2 that has that permission then attach, should take like 5-10 mins to pop up in SYSTEMS MANAGER SHARED RESOURCES - Managed Instances as mark mentions. – Glen Thompson Sep 20 '18 at 16:31"

```
# Start SSM Agent with
sudo systemctl start amazon-ssm-agent
```

6.1.2 AWS Boto3

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

6.1.2.1 Basics

Create local .aws folder under user for example that has credential information, this will be useful for AWS command line operations.

```
# IN C:\Users\fan\.aws
# config file
```

```
[default]
region = us-east-1
output = json
# credentials file
[default]
aws_access_key_id = XKIXXXGSXXXBZXX43XXX
aws_secret_access_key = xxTgp9r0f4XXXXXXX1XXlG1vTy07wydxXXXXXX11
```

Additionally, or alternatively, for boto3 operations, store in for example a yml file, so that appropriate value could be obtained.

```
- main_aws_id: 710673677961,
  aws_access_key_id: XKIXXXGSXXXBZXX43XXX
  aws_secret_access_key: xxTgp9r0f4XXXXXXX1XXlG1vTy07wydxXXXXXX11
  region: us-east-1
  main_ec2_instance_id: i-YYYxYYYYYYx2619xx
  main_ec2_linux_ami: ami-0xYYYYYxx95x71x9
  main_ec2_public_subnet: subnet-d9xxxxYY
  fargate_vpc_name: FanCluster
  fargate_vpc_id: vpc-xxx5xYYY
  fargate_public_subnet: subnet-e3dYYYxx
  fargate_security_group: sg-17xxxxYx
  fargate_task_executionRoleArn: ecsTaskExecutionRole
  batch_task_executionRoleArn: ecsExecutionRole
  fargate_route_table: rtb-5xxxYx25
  date_start: 20180701
```

6.1.2.2 Start Client Service

For the various AWS services, could use Boto3 to access and use programmatically. To use any particular service, first start the client for that service: boto3 client.

We load AWS access key and secret acess key etc in from a yaml file to start boto3 client. We then start the client for AWS Batch. And then describe a compute environment.

```
import boto3
import yaml
import pprint

# Load YAML file
son_aws_yml = "C:/Users/fan/fanwangecon.github.io/_data/aws.yml"
fl_yaml = open(son_aws_yml)
ls_dict_yml = yaml.load(fl_yaml, Loader=yaml.BaseLoader)
# Get the first element of the yml list of dicts
aws_yml_dict_yml = ls_dict_yml[0]

# Use AWS Personal Access Keys etc to start boto3 client
aws_batch = boto3.client('batch',
  aws_access_key_id=aws_yml_dict_yml['aws_access_key_id'],
  aws_secret_access_key=aws_yml_dict_yml['aws_secret_access_key'],
  region_name=aws_yml_dict_yml['region'])

# Show a compute environment Delete some Personal Information
ob_response = aws_batch.describe_compute_environments(computeEnvironments=["SpotEnv2560"])
ob_response['ResponseMetadata'] = ''
ob_response['computeEnvironments'][0]['ecsClusterArn'] = ''
ob_response['computeEnvironments'][0]['serviceRole'] = ''
ob_response['computeEnvironments'][0]['computeResources']['instanceRole'] = ''
pprint.pprint(ob_response, width=1)
```

```
## {'ResponseMetadata': '',
##  'computeEnvironments': [{'computeEnvironmentArn': 'arn:aws:batch:us-east-1:710673677961:compute-
##                           'computeEnvironmentName': 'SpotEnv2560',
##                           'computeResources': {'desiredvCpus': 4,
##                                                'ec2KeyPair': 'fan_wang-key-pair-us_east_nv',
##                                                'instanceRole': '',
##                                                'instanceTypes': ['optimal'],
##                                                'maxvCpus': 2560,
##                                                'minvCpus': 0,
##                                                'securityGroupIds': ['sg-e6642399'],
##                                                'spotIamFleetRole': 'arn:aws:iam::710673677961:rol
##                                                'subnets': ['subnet-d9abbe82'],
##                                                'tags': {},
##                                                'type': 'SPOT'},
##                           'ecsClusterArn': '',
##                           'serviceRole': '',
##                           'state': 'ENABLED',
##                           'status': 'VALID',
##                           'statusReason': 'ComputeEnvironment '
##                                           'Healthy',
##                           'tags': {},
##                           'type': 'MANAGED'}]}
```

6.2 S3

6.2.1 S3 Usages

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

6.2.1.1 Upload Local File to S3

A program runs either locally or on a remote EC2 machine inside a docker container. Upon exit, data does not persist in the docker container and needs to be exported to be saved. The idea is to export program images, csv files, json files, etc to S3 when these are generated, if the program detects that it is been executed on an EC2 machine (in a container).

First, inside the program, detect platform status. For Docker Container on EC2, AWS Linux 2 has platform.release of something like *4.14.193-194.317.amzn2.x86_64*.

```
import platform as platform
print(platform.release())
# This assums using an EC2 instance where amzn is in platform name
```

```
## 10
```

```
if 'amzn' in platform.release():
    s3_status = True
else:
    s3_status = False
print(s3_status)
```

```
## False
```

Second, on s3, create a bucket, *fans3testbucket* for example (no underscore in name allowed). Before doing this, set up AWS Access Key ID and AWS Secrete Acccess KEy in */Users/fan/.aws* folder so that boto3 can access s3 from computer. Upon successful completion of the push, the file can be accessed at https://fans3testbucket.s3.amazonaws.com/_data/iris_s3.dta.

```
import boto3
s3 = boto3.client('s3')
```

```
spn_local_path_file_name = "C:/Users/fan/Py4Econ/aws/setup/_data/iris_s3.dta"
str_bucket_name = "fans3testbucket"
spn_remote_path_file_name = "_data/iris_s3.dta"
s3.upload_file(spn_local_path_file_name, str_bucket_name, spn_remote_path_file_name)
```

6.2.1.2 Download File from S3 to Local Machine

On a local computer, download a particular file from S3. Download back the file we just uploaded onto S3.

```
import boto3
s3 = boto3.client('s3')
spn_local_path_file_name = "C:/Users/fan/Py4Econ/aws/setup/_data/iris_s3_downloaded.dta"
str_bucket_name = "fans3testbucket"
spn_remote_path_file_name = "_data/iris_s3.dta"
s3.download_file(str_bucket_name, spn_remote_path_file_name, spn_local_path_file_name)
```

6.2.1.3 Download File from S3 to EC2 Machine

On a EC2 machine, *Amazon Linux 2 AMI*, for example. Install pip, and install boto3, then download file to the */data/* folder.

```
# ssh into EC2 linux 2 AMI
ssh -i "G:/repos/ThaiJMP/boto3aws/aws_ec2/pem/fan_wang-key-pair-us_east_nv.pem" ec2-user@3.81.101.14
# generate data folder
mkdir data
# install boto3
sudo yum install python-pip python3-wheel && Pip install boto3 --user
# try download file using boto3
# go into python
python
```

Now inside python, download the *iris_s3.dta* to the data folder under root in the EC2 Machine.

```
import boto3
s3 = boto3.client('s3')
spn_ec2_path_file_name = "/home/ec2-user/data/iris_s3_downloaded.dta"
str_bucket_name = "fans3testbucket"
spn_s3_path_file_name = "_data/iris_s3.dta"
s3.download_file(str_bucket_name, spn_s3_path_file_name, spn_ec2_path_file_name)
```

6.2.1.4 Download File from S3 to Active Docker Container

Working inside an active docker container.

First activate and enter into a docker container:

```
# inside EC2 AMI Linux 2, start dockers
sudo service docker start
sudo service docker status
# see docker images
docker images
# run docker container and enter inside
docker run -t -i fanconda /bin/bash
# make a data directory and a esti subdirectory
mkdir data
cd data
mkdir esti
# enter python
python
```

Inside docker, which has boto3 installed already. The Path is different than on EC2, the path root structure is shorter, but otherwise the same.

```
import boto3
s3 = boto3.client('s3')
spn_container_path_file_name = "/data/esti/iris_s3_downloaded.dta"
str_bucket_name = "fans3testbucket"
spn_s3_path_file_name = "_data/iris_s3.dta"
s3.download_file(str_bucket_name, spn_s3_path_file_name, spn_container_path_file_name)
```

6.2.1.5 Forward and Backward Slashes

Following up on the uploading example above, suppose that rather than using forward slash, backward slash is used, then AWS gets confused about folder. This will not appear under the _data folder, but will appear as a file with file name: *_data/iris_s3.dta* under the bucket.

Note that the *folder* for S3 is a GUI trick, but still, we want to use forward slash properly, so that all double backslash that might be generated by default path tools need to be converted to forward slashes

```
import os
# This generates a file directly under bucket _data\iris_s3:
spn_remote_path_file_name_backslash = "_data\\iris_s3_slashbackforward.dta"
s3.upload_file(spn_local_path_file_name, str_bucket_name, spn_remote_path_file_name_backslash)
# This allows the folder structure to be clickable:
spn_remote_path_file_name_forwardslash = spn_remote_path_file_name_backslash.replace(os.sep, '/')
s3.upload_file(spn_local_path_file_name, str_bucket_name, spn_remote_path_file_name_forwardslash)
# Print slashs
print(f'{spn_remote_path_file_name_backslash=}')
```

```
## spn_remote_path_file_name_backslash='_data\\iris_s3_slashbackforward.dta'
```

```
print(f'{spn_remote_path_file_name_forwardslash=}')
```

```
## spn_remote_path_file_name_forwardslash='_data/iris_s3_slashbackforward.dta'
```

6.2.1.6 Sync Local Drive with S3 of a Particular File Type

Boto3 does not offer directory upload/download for s3. Needs to rely on aws command line.

To sync local folder with all files from a particular AWS folder, exclude image files:

```
# CD into a directory
cd /d "G:\S3\fanconda202010\esti"
# Make a new directory making S3 Directory Name
mkdir e_20201025x_esr_medtst_list_tKap_mlt_ce1a2
# cd into the directory just made
cd /d "G:\S3\thaijmp202010\esti\e_20201025x_esr_medtst_list_tKap_mlt_ce1a2"
# copy all results from the s3 folder's subfolders including subfolders, excluding images
aws s3 cp ^
    s3://fanconda202010/esti/e_20201025x_esr_medtst_list_tKap_mlt_ce1a2/ . ^
    --recursive --exclude "*.png"
```

6.3 Batch

6.3.1 AWS Batch Run

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

6.3.1.1 Preparing a Docker Image and a Python Function for Batch Array Job

We want to set-up a function that can be used jointly with AWS Batch Array. With Batch Array, can run many simulations concurrently. All simulations might only differ in random seed for drawing shocks. This requires setting up the proper dockerfile as well as modifying the python function that we want to invoke slightly.

First, create and push a docker image, see this dockerfile. Following the AWS ECR instructions, this registers a docker image in AWS ECR with a URI: *XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fanconda*

The dockerfile has for CMD: *CMD ["python", "/pyfan/pyfan/graph/exa/scatterline3.py"]*. This runs the function scatterline3.

Second, the scatterline3 function checks if *AWS_BATCH_JOB_ARRAY_INDEX* is in the *os.environ*. *AWS_BATCH_JOB_ARRAY_INDEX*, if exists, is used as a random seed to generate data for the graph. When the function is run in a docker container via batch, the function saves the graph output to a bucket in AWS s3. The pushing the s3 is achieved by pyfan.aws.general.path.py.

In the batch job, when *arrayProperties = {'size': 10}*, this will generate *AWS_BATCH_JOB_ARRAY_INDEX* from 1 through 10 in 10 sub-tasks of a single batch task. These *AWS_BATCH_JOB_ARRAY_INDEX* could be used as different random seeds, and could be used as folder suffixes.

Here, the scatterline3 function generates a graph, that will be stored for testing purpose in pyfan_gph_scatter_line_rand folder of fans3testbucket bucket, the images saved has *seed_0.png*, *seed_1.png*, ..., *seed_10.png* as names when *arrayProperties = {'size': 10}*.

6.3.1.2 Register A Batch Job Definition

Given the docker image we created: *XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fanconda*, we can use this to register a batch job.

1. computing requirements: memory and cpu: *vCpus = 1* and *Memory=7168* for example
2. which container to pull from (ECR): List the image name: *XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fanconda* for example
3. job role ARN: *arn:aws:iam::XXXX7367XXXX:role/ecsExecutionRole* to allow for proper in and out from and to the container.

These can be registered programmatically by using boto3: Boto3 Batch Documentation

In the example below, will register a new job definition, this will add *pyfan-scatterline3-test-rmd* to job definition as an additional job definition.

Everytime, when the code below is re-run, a new batch revision number is generated. AWS allows per batch job to have potential hundreds of thousands of revisions.

```
import boto3
import yaml
import pprint

# Load YAML file with security info
srn_aws_yml = "C:/Users/fan/fanwangecon.github.io/_data/aws.yml"
fl_yaml = open(srn_aws_yml)
ls_dict_yml = yaml.load(fl_yaml, Loader=yaml.BaseLoader)
aws_yml_dict_yml = ls_dict_yml[0]

# Dictionary storing job definition related information
job_dict = {"jobDefinitionName": 'pyfan-scatterline3-test-rmd',
            "type": "container",
            "containerProperties": {
                "image": aws_yml_dict_yml['main_aws_id'] + ".dkr.ecr." +
                         aws_yml_dict_yml['region'] + ".amazonaws.com/fanconda",
                "vcpus": int(1),
                "memory": int(1024),
```

```
                "command": ["python",
                            "/pyfan/pyfan/graph/exa/scatterline3.py",
                            "-A", "fans3testbucket",
                            "-B", "111"],
                "jobRoleArn": "arn:aws:iam::" + aws_yml_dict_yml['main_aws_id'] +
                              ":role/" + aws_yml_dict_yml['batch_task_executionRoleArn']
            },
            "retryStrategy": {
                "attempts": 1
            }}

# Use AWS Personal Access Keys etc to start boto3 client
aws_batch = boto3.client('batch',
  aws_access_key_id=aws_yml_dict_yml['aws_access_key_id'],
  aws_secret_access_key=aws_yml_dict_yml['aws_secret_access_key'],
  region_name=aws_yml_dict_yml['region'])

# Register a job definition
response = aws_batch.register_job_definition(
        jobDefinitionName = job_dict['jobDefinitionName'],
        type = job_dict['type'],
        containerProperties = job_dict['containerProperties'],
        retryStrategy = job_dict['retryStrategy'])

# Print response
pprint.pprint(response, width=1)
```

```
## {'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
##                                       'content-length': '169',
##                                       'content-type': 'application/json',
##                                       'date': 'Tue, '
##                                               '29 '
##                                               'Dec '
##                                               '2020 '
##                                               '04:03:12 '
##                                               'GMT',
##                                       'x-amz-apigw-id': 'YS-UEF8xoAMFWVg=',
##                                       'x-amzn-requestid': '74ce59aa-3b09-4bee-a32c-a2b993935f40',
##                                       'x-amzn-trace-id': 'Root=1-5feaaa80-76c54f544651d37c14c0cbd
##                        'HTTPStatusCode': 200,
##                        'RequestId': '74ce59aa-3b09-4bee-a32c-a2b993935f40',
##                        'RetryAttempts': 0},
##  'jobDefinitionArn': 'arn:aws:batch:us-east-1:710673677961:job-definition/pyfan-scatterline3-test
##  'jobDefinitionName': 'pyfan-scatterline3-test-rmd',
##  'revision': 91}
```

6.3.1.3 Submit a Batch Array

Given the batch job definition that has been created. Create also Job Queues and related compute environments. Then we can run Batch Array. Upon submitting the batch array, you can monitor AWS EC2 instances, should notice potentially many instances of EC2 starting up. AWS is starting EC2 instances to complete the batch array jobs.

create a batch compute environment that uses spot price instances, which will be much cheaper than on demand costs. Will need to set proper AMI roles, *arn:aws:iam::XXXX7367XXXX:role/AmazonEC2SpotFleetRole* for *Spot fleet role*, and also proper securities.

When the *array_size* parameter is equal to 100, that starts 100 child processes, with 1 through 100 for *AWS_BATCH_JOB_ARRAY_INDEX*, which, could be used directly by the python function by taking

in the parameter from the os environment as shown earlier. For demonstration purposes, will only set *array_size=3* in the example below.

Outputs from the scatterline3 has a timestamp, so each time the code below is run, will generate several new images, with the same set of random seeds, but different date prefix. The output s3 folder is public.

```
import boto3
import yaml
import pprint

import datetime as datetime

# Using the "jobDefinitionName": 'pyfan-scatterline3-test-rmd' from registering
jobDefinitionName = 'pyfan-scatterline3-test-rmd'

# How many child batch processes to start
# child process differ in: AWS_BATCH_JOB_ARRAY_INDEX
array_size = 3

# job name
timestr = "{:%Y%m%d%H%M%S%f}".format(datetime.datetime.now())
timesufx = '_' + timestr
st_jobName = jobDefinitionName + timesufx

# job queue (needs to design own queue in batch)
st_jobQueue = 'Spot'

# start batch service
# Load YAML file with security info
srn_aws_yml = "C:/Users/fan/fanwangecon.github.io/_data/aws.yml"
fl_yaml = open(srn_aws_yml)
ls_dict_yml = yaml.load(fl_yaml, Loader=yaml.BaseLoader)
aws_yml_dict_yml = ls_dict_yml[0]
# Use AWS Personal Access Keys etc to start boto3 client
aws_batch = boto3.client('batch',
                         aws_access_key_id=aws_yml_dict_yml['aws_access_key_id'],
                         aws_secret_access_key=aws_yml_dict_yml['aws_secret_access_key'],
                         region_name=aws_yml_dict_yml['region'])

# aws batch submit job
dc_json_batch_response = aws_batch.submit_job(
    jobName=st_jobName,
    jobQueue=st_jobQueue,
    arrayProperties={'size': array_size},
    jobDefinition=jobDefinitionName)

# Print response
pprint.pprint(dc_json_batch_response, width=1)
```

```
## {'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
##                                       'content-length': '198',
##                                       'content-type': 'application/json',
##                                       'date': 'Tue, '
##                                               '29 '
##                                               'Dec '
##                                               '2020 '
##                                               '04:03:12 '
##                                               'GMT',
##                                       'x-amz-apigw-id': 'YS-UJH3ooAMFdmw=',
```

```
##                                          'x-amzn-requestid': 'd94f67cf-43f7-4ef0-af38-256c49f5868e',
##                                          'x-amzn-trace-id': 'Root=1-5feaaa80-11335e5955fc16131efefaa
##                       'HTTPStatusCode': 200,
##                       'RequestId': 'd94f67cf-43f7-4ef0-af38-256c49f5868e',
##                       'RetryAttempts': 0},
##  'jobArn': 'arn:aws:batch:us-east-1:710673677961:job/69181af6-e4f5-483a-84e8-784c1f2edc64',
##  'jobId': '69181af6-e4f5-483a-84e8-784c1f2edc64',
##  'jobName': 'pyfan-scatterline3-test-rmd_20201228220304759041'}
```

6.3.1.4 Track the Status of a Submitted Batch Array Until it Finished

To automate certain processes, often need to check and wait for job to complete. Can do this on web interface. Easier to do this via boto3 operations: describe_job and list_jobs

Given the batch array job we just generated above, first, parse out the job ID from the response from the batch array submission above. Then use list_jobs to check the length of JobSummaryList, and then use describe_jobs to check overall job completion status.

```
import time
# Get Job ID
st_batch_jobID = dc_json_batch_response['jobId']
# Print Job ID
print(f'{st_batch_jobID=}')
# While loop to check status
```

```
## st_batch_jobID='69181af6-e4f5-483a-84e8-784c1f2edc64'
```

```
bl_job_in_progress = True
it_wait_seconds = 0
while bl_job_in_progress and it_wait_seconds <= 600:
    # describe job
    dc_json_batch_describe_job_response = aws_batch.describe_jobs(jobs=[st_batch_jobID])
    # pprint.pprint(dc_json_batch_describe_job_response, width=1)
    it_array_size = dc_json_batch_describe_job_response['jobs'][0]['arrayProperties']['size']
    dc_status_summary = dc_json_batch_describe_job_response['jobs'][0]['arrayProperties']['statusSum
    if dc_status_summary:
        # check status
        it_completed = dc_status_summary['SUCCEEDED'] + dc_status_summary['FAILED']
        if it_completed < it_array_size:
            bl_job_in_progress = True
            # sleep three seconds
            time.sleep(10)
            it_wait_seconds = it_wait_seconds + 10
        else:
            bl_job_in_progress = False

        print(f'{it_wait_seconds=}, ArrayN={it_array_size},' \
              f'SUCCEEDED={dc_status_summary["SUCCEEDED"]}, FAILED={dc_status_summary["FAILED"]}, '
              f'RUNNING={dc_status_summary["RUNNING"]}, PENDING={dc_status_summary["PENDING"]}, ' \
              f'RUNNABLE={dc_status_summary["RUNNABLE"]}')
    else:
        #empty statussummary
        bl_job_in_progress = True
        time.sleep(10)
        it_wait_seconds = it_wait_seconds + 10
        print(f'{it_wait_seconds=}, ArrayN={it_array_size}')
```

```
## it_wait_seconds=10, ArrayN=3
## it_wait_seconds=20, ArrayN=3,SUCCEEDED=0, FAILED=0, RUNNING=0, PENDING=0, RUNNABLE=3
## it_wait_seconds=30, ArrayN=3,SUCCEEDED=0, FAILED=0, RUNNING=0, PENDING=0, RUNNABLE=3
## it_wait_seconds=40, ArrayN=3,SUCCEEDED=0, FAILED=0, RUNNING=0, PENDING=0, RUNNABLE=3
```

```
## it_wait_seconds=50, ArrayN=3,SUCCEEDED=2, FAILED=0, RUNNING=0, PENDING=0, RUNNABLE=1
## it_wait_seconds=50, ArrayN=3,SUCCEEDED=3, FAILED=0, RUNNING=0, PENDING=0, RUNNABLE=0
```

Chapter 7

Docker Container

7.1 Docker Setup

7.1.1 Docker Setup

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

7.1.1.1 Install Docker on AWS

Installation Instructions

1. Putty
2. access to .pem key
3. conda aws environment below

For Amazon Linux 2:

```
# SSH into EC2 Instance
ssh ec2-user@ec2-52-23-218-117.compute-1.amazonaws.com
# Update
sudo yum update -y
# install
sudo amazon-linux-extras install docker -y
# start service
sudo service docker start
sudo service docker status
# execute docker commands without sudo
sudo usermod -a -G docker ec2-user
# log out and in reboot does not change public address
sudo reboot
# if docker info does not work, docker start again
docker info
```

7.1.1.2 Create a Dockerfile and build it

Create a Dockerfile and build it. Building a dockerfile generates a docker image:

```
# docker folder
mkdir ~/docker
# cd into docker folder
cd ~/docker
# create a Dockerfile in the docker folder
# copy the Example Dockerfile below to the Dockerfile
vim Dockerfile
```

```
# in the docker directory build the docker file
docker build -t hello-world .
```

Example Dockerfile:

```
FROM ubuntu:12.04

# Install dependencies
RUN apt-get update -y
RUN apt-get install -y apache2

# Install apache and write hello world message
RUN echo "Hello World!" > /var/www/index.html

# Configure apache
RUN a2enmod rewrite
RUN chown -R www-data:www-data /var/www
ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2

EXPOSE 80

CMD ["/usr/sbin/apache2", "-D",  "FOREGROUND"]
```

7.1.1.3 Run, Enter and Exit

docker build generates a docker image, we start the docker container, run using the image created.

```
# list docker images available to run
docker images
```

```
These could be some images that are shown after running *docker images*:

REPOSITORY                                                      TAG                 IMAGE ID
XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fanconda       latest              5d1a0df0796e
XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fanconda2020   latest              2db5e859d70c
fanconda2020                                                    latest              2db5e859d70c
fanconda5                                                       latest              fa55672e7753
fanconda3                                                       latest              2083f1124465
```

Run the image and enter into it (use an image name) and run commands with programs, upon exit, container is stopped. Entering back into the container, the data was generated before now is no longer there, it is a new container based on the same image.

```
# will be inside now the conda image (base)
docker run -t -i fanconda /bin/bash
# can run programs inside here that have been loaded into the image
python /fanProg/invoke/run.py -A ng_s_t=kap_m0_nld_m_simu=2=3 -B b -C 20180814_beta -D esti_param.be
# review generated outputs inside docker, results are stored by the run.py program and associated fi
cd /data
ls
# To exit the currently running docker
exit
# show docker container exited
docker ps -a
```

Root directory in conda docker: > fanProg bin boot data dev etc home lib lib64 media mnt opt proc pyfan root run sbin srv sys tmp usr var

Run the image with program without entering into it.

```
docker run fancondajmp python /ThaiJMP/invoke/run.py -A ng_s_t=kap_m0_nld_m_simu=2=3 -B b -C 2018081
```

- Docker container will automatically stop after "docker run -d"

7.1.1.4 Status and Cleaning

7.1.1.4.1 Docker file and Git Repo To have docker file access a git repo without exposing git repo password. Generate a private token, and access as below. See stackoverflow-23391839.

```
RUN git clone https://b123451234dfc025a836927PRIVATETOKEND1239@github.com/FanWangEcon/ThaiJMP.git /T
```

7.1.1.4.2 Docker Status, Space and Clean First, start service:

```
# start docker
sudo service docker start
# see status
sudo service docker status
```

Second, list all docker related space usages, containers and images:

```
# check disk usage
docker system df
```

Third, clean containers

```
# see docker containers
docker container ls -a
# Remove all stopped docker containers
docker rm $(docker ps -a -q)
```

Fourth, clean images

```
# list all images
docker images
docker images --all
# Clean all images not referenced by a container
docker image prune
```

7.1.2 ECR Setup

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

7.1.2.1 Pull from Elastic Container Registry (ECR)

Given docker files already on ECR, in EC2, first, get password, then pull.

```
# log in
# copy the output from the line below and paste
aws ecr get-login --no-include-email
# this is copied from output of the command above
docker login -u AWS -p PASSWORDPASSWORDPASSWORDPASSWORD https://XXXX7367XXXX.dkr.ecr.us-east-1.amazo
# pull from docker
docker pull XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fancondaXX
```

7.1.2.2 Update Elastic Container Registry (ECR)

There is a local conda file, perhaps some project repo have been updated, need to update docker file on ECR (or create new ones). Controlling EC2 can be done manually, or via SSM.

1. Start a EC2 Instance
2. Create a docker folder on EC2 instance (on remote)
3. scp update dockerfile in EC2 docker folder (local to remote)

4. build on remote server docker container (on remote)
5. push from EC2 updated docker to ECR (on remote)

First, after creating/starting a EC2 instance, create a docker file and scp update:

```
# Then a sequences of SSM calls:
# on local machine:
ssh -i "C:/Users/fan/CondaPrj/boto3aws/aws_ec2/pem/fan_wang-key-pair-us_east_nv.pem" ec2-user@54.161

# if new instance, create a docker folder under main
# on remote machine
mkdir /home/ec2-user/docker

# ssm call to remove current dockerfile
# on remote machine
rm /home/ec2-user/docker/Dockerfile

# run local scp command to copy lateste Dockerfile to EC2, local scp generated by ec2managee
# on local machine
scp -o StrictHostKeyChecking=accept-new -i C:/Users/fan/CondaPrj/boto3aws/aws_ec2/pem/fan_wang-key-p
```

Second, start container service remotely, and build new container:

```
# start docker service on ec2
# on remote machine
sudo service docker start

# On remote machine
cd /home/ec2-user/docker
docker build -t fanconda6 --build-arg CACHE_DATE=2020-09-21-22-43-52 .
```

Third, push new container to ECR (tag, get token, login, push):

```
# Start Container Service
sudo service docker start

# CD into folder on remote
cd /home/ec2-user/docker

# tag docker
docker tag fancondaxxx XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fancondaxxx

# ECR Docker Log in
# ssm.get_authorization_token(registryIds=[boto3aws.aws_keys()['main_aws_id']])
# Decode authorization token
docker login -u AWS -p TOKENX6XXXxXXg1XXg3OX0= https://XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com

# ECR Docker Push to ECR
docker push XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fancondaxxx
```

7.1.2.3 PyFan Procedures

1. Start EC2 instance
2. Push to ECR
3. Get SSH link to EC2, SSH into EC2
4. *sudo service docker start* and *docker images* (or see pull earlier)
5. start docker image and enter to access via command line: *docker run -t -i fanconda /bin/bash*

7.1.2.4 More Example ECR Code and Outputs

7.1.2.4.1 Example Docker File for AWS Note that that private git repos are pulled in. Note also that AWS keys are set up to allow for various access to AWS services.

```
FROM continuumio/anaconda3

VOLUME /data

# Conda update
RUN conda update conda

# https://github.com/ContinuumIO/docker-images/issues/49#issuecomment-311556456
RUN apt-get update && \
    apt-get install libgl1-mesa-glx -y

# Install Conda additional packages that i use
RUN conda install -c conda-forge interpolation
RUN conda install -c conda-forge boto3

# see https://github.com/moby/moby/issues/22832, this allows for code below to run without --no-cach
ARG CACHE_DATE=2000-01-01

# Clone our private GitHub Repository: PyFan
RUN git clone https://b123451234dfc025a836927PRIVATETOKEND1239@github.com/FanWangEcon/pyfan.git /pyf

# Make port 80 available to the world outside this container
EXPOSE 80

# Install software
ENV PYTHONPATH /pyfan/

ENV AWS_BUCKET_NAME=BucketName
ENV AWS_ACCESS_KEY_ID=XKIXXXGSXXXBZXX43XXX
ENV AWS_SECRET_ACCESS_KEY=xxTgp9r0f4XXXXXXX1XXlG1vTy07wydxXXXXXX11

# Run
CMD ["python", "/pyfan/pyfan/graph/exa/scatterline3.py"]
```

```
# aws_keys stores keys
aws_keys_dict = aws_keys()
ssm = boto3.client('ssm',
                   aws_access_key_id=aws_keys_dict['aws_access_key_id'],
                   aws_secret_access_key=aws_keys_dict['aws_secret_access_key'],
                   region_name=aws_keys_dict['region'])
commands = 'rm /home/ec2-user/docker/Dockerfile'
resp = client.send_command(
        DocumentName="AWS-RunShellScript", # One of AWS' preconfigured documents
        Parameters={'commands': commands},
        InstanceIds=[instance_id])
```

7.1.2.4.2 Example SSM communication

7.1.2.4.3 Outputs from Docker Build outputs from docker build

```
json.py - jdump - 47 -  2020-09-22 16:04:32,459 - INFO list_command_invocation-cur_output
:[
    "Sending build context to Docker daemon  3.072kB\r\r",
```

```
"Step 1/16 : FROM continuumio/anaconda3",
" ---> 472a925c4385",
"Step 2/16 : VOLUME /data",
" ---> Using cache",
" ---> cf4e6a503f00",
"Step 3/16 : RUN conda update conda",
" ---> Using cache",
" ---> 542901f01365",
"Step 4/16 : RUN apt-get update && apt-get install libgl1-mesa-glx -y",
" ---> Using cache",
" ---> 6672960aa00c",
"Step 5/16 : RUN conda install -c conda-forge interpolation",
" ---> Using cache",
" ---> efd86a4259a4",
"Step 6/16 : RUN conda install -c conda-forge boto3",
" ---> Using cache",
" ---> bd0146dac9b3",
"Step 7/16 : ARG CACHE_DATE=2000-01-01",
" ---> Using cache",
" ---> dc40688e3720",
"Step 8/16 : RUN git clone https://XXXX@github.com/FanWangEcon/pyfan.git /pyfan/",
" ---> Running in 9c0c2a444540",
"\u001b[91mCloning into '/pyfan'...",
"\u001b[0mRemoving intermediate container 9c0c2a444540",
" ---> c80480cc51a1",
"Step 9/16 : RUN git clone https://XXXX@github.com/FanWangEcon/CondaProg.git /CondaProg/",
" ---> Running in 07d9f665b760",
"\u001b[91mCloning into '/CondaProg'...",
"\u001b[0mRemoving intermediate container 07d9f665b760",
" ---> a5ac6c6e1458",
"Step 10/16 : EXPOSE 80",
" ---> Running in 1a8ef516e236",
"Removing intermediate container 1a8ef516e236",
" ---> 13ab2965e892",
"Step 11/16 : ENV PYTHONPATH /pyfan/",
" ---> Running in 2d9e4b68164b",
"Removing intermediate container 2d9e4b68164b",
" ---> 0a74e69ce1c8",
"Step 12/16 : ENV PYTHONPATH $PYTHONPATH:/CondaProg/",
" ---> Running in ba59f1273f51",
"Removing intermediate container ba59f1273f51",
" ---> 11fd9d732e2e",
"Step 13/16 : ENV AWS_BUCKET_NAME=BucketName",
" ---> Running in e7a052d3eacf",
"Removing intermediate container e7a052d3eacf",
" ---> 5e294f562838",
"Step 14/16 : ENV AWS_ACCESS_KEY_ID=XXXXX5GSDZSXXXX43XXX",
" ---> Running in 60d810a8514f",
"Removing intermediate container 60d810a8514f",
" ---> 2fa1ac4e7d3b",
"Step 15/16 : ENV AWS_SECRET_ACCESS_KEY=XXXXXXXXXXXXX",
" ---> Running in 8b34126cee5d",
"Removing intermediate container 8b34126cee5d",
" ---> 93bd8b521d61",
"Step 16/16 : CMD [\"python\", \"/CondaPrj/invoke/invoke.py\"]",
" ---> Running in dd3ed44dcca7",
"Removing intermediate container dd3ed44dcca7",
```

```
    " ---> 506f92a794cd",
    "Successfully built 506f92a794cd",
    "Successfully tagged fanconda:latest",
    "Total reclaimed space: 0B",
    ""
]
```

```
json.py - jdump - 47 -  2020-09-22 16:05:32,986 - INFO list_command_invocation-cur_output
:[
    "The push refers to repository [XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fanconda]",
    "63cc929545c3: Preparing",
    "d849f5d67bbb: Preparing",
    "f9c77b2e4c5f: Preparing",
    "7ffd6385ae0e: Preparing",
    "2fc88e09d363: Preparing",
    "50e089036495: Preparing",
    "6637031dbcc2: Preparing",
    "68d0bdfd0715: Preparing",
    "d0f104dc0a1f: Preparing",
    "50e089036495: Waiting",
    "6637031dbcc2: Waiting",
    "68d0bdfd0715: Waiting",
    "d0f104dc0a1f: Waiting",
    "2fc88e09d363: Layer already exists",
    "7ffd6385ae0e: Layer already exists",
    "50e089036495: Layer already exists",
    "6637031dbcc2: Layer already exists",
    "68d0bdfd0715: Layer already exists",
    "d0f104dc0a1f: Layer already exists",
    "d849f5d67bbb: Pushed",
    "f9c77b2e4c5f: Pushed",
    "63cc929545c3: Pushed",
    "latest: digest: sha256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX size: 2226",
    ""
]
```

7.1.2.4.4 Output from Docker Push

7.1.2.4.5 AWS ECR Instructions Under repositories listed under ECR, click on *View push command*, which shows:

```
# Retrieve an authentication token and authenticate your Docker client to your registry.
# Use the AWS CLI:

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin XXXX736

# Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the
# Build your Docker image using the following command. For information on building a Docker file fro
docker build -t fanconda .

# After the build completes, tag your image so you can push the image to this repository:
docker tag fanconda:latest XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fanconda:latest
```

7.1.2.5 Command Line Sequence Gathered

Gathered sequence of command line operations:

```
ssh -i "G:/repos/CondaPrj/boto3aws/aws_ec2/pem/fan_wang-key-pair-us_east_nv.pem" ec2-user@34.229.39.

docker run -t -i fanconda /bin/bash

scp -o StrictHostKeyChecking=accept-new -i G:/repos/CondaPrj/boto3aws/aws_ec2/pem/fan_wang-key-pair-

sudo service docker start

cd /home/ec2-user/docker
docker build -t fanconda --build-arg CACHE_DATE=2020-12-22-10-58-57 .
docker system prune --force

cd /home/ec2-user/docker
docker tag fanconda 710673677961.dkr.ecr.us-east-1.amazonaws.com/fanconda

aws ecr get-login --no-include-email
docker login -u AWS -p XXXXX= https://710673677961.dkr.ecr.us-east-1.amazonaws.com

docker push 710673677961.dkr.ecr.us-east-1.amazonaws.com/fanconda
```

Chapter 8

Get Data

8.1 Environmental Data

8.1.1 ECMWF ERA5 Data

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

8.1.1.1 Basic Conda Setup

1. Download Anaconda for Python 3. For more involved conda instructions see here
2. Open up *anaconda prompt* with admin rights (right click choose as admin).

```
# Inside anaconda prompt
where python
where anaconda
# C:/ProgramData/Anaconda3/Scripts/anaconda.exe
# C:/ProgramData/Anaconda3/python.exe
```

3. Add to Path
4. Install cdsapi and eccodes

```
conda config --add channels conda-forge
conda install -c anaconda pandas
conda install -c conda-forge eccodes -y
conda install -c conda-forge cfgrib -y
conda install -c anaconda xarray
```

8.1.1.2 Account Registration

1. Register for an account
2. Agree to Licence
3. Go to your CDS user page copy the url and key: Get url and key
   - this has UID, 4XXXX, and API KEY, 4XXXfXXX-XXXf-4XXX-9XXX-7XXXebXXfdXX
   - together they should look like: 4XXXX:4XXXfXXX-XXXf-4XXX-9XXX-7XXXebXXfdXX
4. Open up an editor (notepad++ for example), create an empty file, paste the url and your UID:APIKEY into the file as below. Save file as: *C:/Users/fan/.cdsapirc*. Under user root, as *.cdsapirc* file. Note *.cdsapirc* is the file name, you are saving that under the directory *C:/Users/fan/*.

```
url: https://cds.climate.copernicus.eu/api/v2
key: 4XXXX:4XXXfXXX-XXXf-4XXX-9XXX-7XXXebXXfdXX
```

8.1.1.3 Run API Request via Jupyter Notebook

1. open up Jupyter Notebook (this opens up a browser page)
   - cd "C:/Users/fan/Downloads"
   - jupyter notebook
2. create a new *python3* file somewhere you like
3. name the file *cdstest* (saved as ipynb file)
4. paste the code below inside the *ipynb* file you opened (modify *spt_root*):

```
import cdsapi
import urllib.request
# download folder
spt_root = "C:/Users/fan/downloads/_data/"
spn_dl_test_grib = spt_root + "test_china_temp.grib"
# request
c = cdsapi.Client()
res = c.retrieve("reanalysis-era5-pressure-levels",
  {
    'product_type': 'reanalysis',
    'variable': 'temperature',
    'pressure_level': '1000',
    'year': '2008',
    'month': '01',
    'day': '01',
    'time': '12:00',
    'format': 'netcdf',
    'area'           : [53.31, 73, 4.15, 135],
    'grid'           : [1.0, 1.0],
    "format": "grib"
  },
  spn_dl_test_grib
)
# show results
print('print results')
print(res)
print(type(res))
```

5. click run

8.1.1.4 Run API request via Ipython

1. In Anaconda Prompt: *ipython*
2. Open a file in notepad++ or elsewhere, copy the code above over and edit the spt_root to reflect your directories
3. Select the entire code in the notepad++ page, and copy all lines
4. Now inside ipython, type percentage and paste: %paste
5. This should run the file above and save the grib file in the folder you specified with the name you specified.

8.1.1.5 Convert CRIB data to CSV

1. inside conda prompt cd into the folder where you downloaded the grib file
2. *grib_ls* shows what is in the grib file
3. *grib_get_data* translates grib to csv

```
cd "C:/Users/fan/downloads/_data/"
grib_ls test_china_temp.grib
grib_get_data test_china_temp.grib > test_china_temp_raw.csv
```

8.1.1.6 More Advanced Download Setup and Instructions

8.1.1.6.1 Conda Enviornment and Installation In conda, set up a conda environment for downloading ECMWF data using the ECMWF API. (Conda Set-up)

```
# Set up
conda deactivate
conda list env
conda env remove -n wk_ecmwf
conda create -n wk_ecmwf -y
conda activate wk_ecmwf

# Add conda-forge to channel in env
conda config --env --add channels conda-forge
conda config --get channels
conda config --get channels --env

# Install
conda install cdsapi -y
conda install -c anaconda pandas
conda install -c conda-forge eccodes -y
conda install -c conda-forge cfgrib -y
conda install -c anaconda xarray
```

This creates the conda env that we are using here for python.

8.1.1.6.2 Config File .cdsapirc Open up the *cdsapirc*, create new if does note exist. Below, open up the file and save the text. See Python Reading and Writing to File Examples.

First, get the text for the config file:

```
stf_cds_cdsapirc = """\
url: https://cds.climate.copernicus.eu/api/v2
key: 4XXXX:4XXXfXXX-XXXf-4XXX-9XXX-7XXXebXXfdXX\
"""
print(stf_cds_cdsapirc)
```

Second save text to file:

```
# Relative file name
spt_file_cds = "C:/Users/fan/"
snm_file_cds = ".cdsapirc"
spn_file_cds = spt_file_cds + snm_file_cds
# Open new file
fl_cdsapirc_contents = open(spn_file_cds, 'w')
# Write to File
fl_cdsapirc_contents.write(stf_cds_cdsapirc)
# Close
fl_cdsapirc_contents.close()
```

```
# Open the config file to check
code "C:/Users/fan/.cdsapirc"
```

8.1.1.7 Generate API Requests

Go to the sites below, choose download data, pick what is needed, and then select *Show API request* at the bottom of page:

ERA5 pressure levels from 1979 to present

- ERA5 hourly pressure
- ERA5 monthly pressure

ERA5 single levels from 1979 to present

- ERA5 hourly pressure
- ERA5 monthly pressure

8.1.1.7.1 API Request China Temp Test

API function is here.

Select based on China's area, some testing data and download grib file. The data is from 2008, Jan 1st, at 12 noon?

Open up Jupyter notebook: *jupyter notebook*

```
# import module in conda env wk_ecmwf
import cdsapi
import urllib.request

# download folder
spt_root = "C:/Users/fan/py4econ/vig/getdata/envir/"
spn_dl_test_grib = spt_root + "_data/test/test_china_temp.grib"

# request
c = cdsapi.Client()
res = c.retrieve("reanalysis-era5-pressure-levels",
  {
    'product_type': 'reanalysis',
    'variable': 'temperature',
    'pressure_level': '1000',
    'year': '2008',
    'month': '01',
    'day': '01',
    'time': '12:00',
    'format': 'netcdf',
    'area'          : [53.31, 73, 4.15, 135],
    'grid'          : [1.0, 1.0],
    "format": "grib"
  },
  spn_dl_test_grib
)
# show results
print('print results')
print(res)
print(type(res))

# download
# response = urllib.request.urlopen('http://www.example.com/')
# html = response.read()
```

2020-06-17 23:51:35,107 INFO Welcome to the CDS 2020-06-17 23:51:35,107 INFO Sending request to https://cds.climate.copernicus.eu/api/v2/resources/reanalysis-era5-pressure-levels 2020-06-17 23:51:36,441 INFO Request is queued 2020-06-17 23:51:39,183 INFO Request is running 2020-06-17 23:51:45,059 INFO Request is completed 2020-06-17 23:51:45,060 INFO Downloading > http://136.156.133.25/cache-compute-0008/cache/data2/adaptor.mars.internal-1592455900.8655114-11162-11-68e1ea23-8985-4926-95e6-9f181cc7792> 7.grib to C:/Users/fan/pyfan/vig/getdata/envir/_data/test/test_china_temp.grib (6.3K) 2020-06-17 23:51:45,441 INFO Download rate 16.6K/s print results Result(content_length=6480,content_type=application/x-grib,location=http://136.156.133.25/cache-compute-0008/cache/data2/adaptor.mars.inte>

rnal-1592455900.8655114-11162-11-68e1ea23-8985-4926-95e6-9f181cc77927.grib) <class 'cdsapi.api.Result'>

Convert grib to raw csv, open up command line:

```
cd "C:/Users/fan/pyfan/vig/getdata/envir/_data/test/"
grib_ls test_china_temp.grib
grib_get_data test_china_temp.grib > test_china_temp_raw.csv
```

Format the CSV file (is not comma separated)

```
spt_root = "C:/Users/fan/pyfan/vig/getdata/envir/_data/test/"
spn_csv_raw = spt_root + "test_china_temp_raw.csv"
spn_csv_edi = spt_root + "test_china_temp.csv"

with open(spn_csv_raw, 'r') as f_in, open(spn_csv_edi, 'w') as f_out:
    f_out.write(next(f_in))
    [f_out.write(','.join(line.split()) + '\n') for line in f_in]
```

Show CSV results:

```
# Path and Read
spt_root = "C:/Users/fan/pyfan/vig/getdata/envir/"
spn_dl_test_csv = paste0(spt_root, "_data/test/test_china_temp.csv")
china_weather_data <- read.csv(spn_dl_test_csv)

# Top 50 rows
kable(head(china_weather_data, 50),
      caption="Chinese Long and Lat, Temperature Pressure, 2008 Jan 1st at 12 noon?") %>%
  kable_styling_fc()
```

Chinese Long and Lat, Temperature Pressure, 2008 Jan 1st at 12 noon?

time	latitude	longitude	u10	v10	d2m	t2m	msl	sp
2008-01-01 02:00:00	23.25	113	-2.6031342	-4.829605	265.4314	284.8363	102918.8	102616.0
2008-01-01 02:00:00	23.25	114	-2.7173920	-3.808121	262.7693	284.2719	102862.1	101628.0
2008-01-01 02:00:00	22.25	113	-2.6246185	-6.311050	266.9294	284.2602	102796.6	102059.0
2008-01-01 02:00:00	22.25	114	-2.6285248	-6.152847	267.4978	285.3168	102710.1	102201.0
2008-01-01 12:00:00	23.25	113	-1.1495056	-2.728592	265.8091	286.1729	102588.9	102290.7
2008-01-01 12:00:00	23.25	114	-1.4454040	-2.477615	265.3033	285.0987	102591.6	101374.7
2008-01-01 12:00:00	22.25	113	-0.6924744	-4.270584	268.0396	286.5753	102482.9	101757.7
2008-01-01 12:00:00	22.25	114	-1.9668884	-4.906326	266.7486	288.0030	102440.1	101940.7

"ERA5 is a comprehensive reanalysis, from 1979 (soon to be backdated to 1950) to near real time, which assimilates as many observations as possible in the upper air and near surface. The ERA5 atmospheric model is coupled with a land surface model and a wave model."

1. Register for an account
2. Agree to Licence

8.1.1.8 Learning

8.1.1.8.1 Terminologies Links:

- status of the CDS queue.

Terminologies:

- single level parameters

8.1.1.8.2 Single Level Parameters ERA5 Variables?

1. Table 1: surface and single level parameters: invariants

2. Table 9: pressure level parameters: instantaneous

- Temperature

ER5 Data Download Instructions.

8.1.1.9 UTCI, NC Format Data, Download, Unzip, Convert to combined CSV

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: API request by itself
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

```
################################################
# ------------ 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')


# 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'  : [53.31, 73, 4.15, 135],
        '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)

################################################
# ------------ 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 9

System and Support

9.1 Command Line

9.1.1 Python Command Line

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

9.1.1.1 Run Command Line from Inside Python

Use subprocess, where is python:

```
import subprocess
cmd_popen = subprocess.Popen(["where", "python"],
                             stdin=subprocess.PIPE,
                             stdout=subprocess.PIPE,
                             stderr=subprocess.PIPE)
output, err = cmd_popen.communicate()
print(output.decode('utf-8'))
```

```
## G:\ProgramData\Anaconda3\envs\wk_pyfan\python.exe
## G:\ProgramData\Anaconda3\python.exe
## C:\Users\fan\AppData\Local\Microsoft\WindowsApps\python.exe
```

Command line file redirection symbol,

```
# The > command line sends current console output to file.txt
# cd "C:\users\fan"
# ls > ls_files.txt
# rm ls_files.txt
import os
import subprocess

# ls in current location
cmd_ls = subprocess.Popen(["ls"], stdin=subprocess.PIPE, stdout=subprocess.PIPE, stderr=subprocess.P
stf_out_cmd_ls, err = cmd_ls.communicate()
stf_out_cmd_ls = stf_out_cmd_ls.decode('utf-8')
print(stf_out_cmd_ls)
```

```
srt_file_tex = "_file/"
sna_file_tex = "test_ls_pyfanvig_stdout"
srn_file_tex = srt_file_tex + sna_file_tex + ".txt"
fl_tex_contents = open(srn_file_tex, 'w')
fl_tex_contents.write(stf_out_cmd_ls)
```

```
## 0
```

```
fl_tex_contents.close()
```

9.1.1.2 Execute Command Line Python Functions

- run python from command line
- run python function with parameters from command line

Here run python from command line inside python itself.

```
# Run:
from py.fan.util.rmd.mattexmd import fp_mlxtex2md
fp_mlxtex2md(spt_root='C:/Users/fan/Math4Econ/matrix_application/', ls_srt_subfolders=None, st_rglob
```

```
# Run:
python -c "from pyfan.util.rmd.mattexmd import fp_mlxtex2md; fp_mlxtex2md(spt_root='C:/Users/fan/Mat
```

9.1.2 Run Matlab Functions

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

9.1.2.1 Generate A template Matlab Script

Generate an example matlab script file with parameter x.

```
# Example Matlab Function
stf_m_contents = """\
a = x + 1
b = 10*x\
"""
# Print
print(stf_m_contents)
# Open new file
```

```
## a = x + 1
## b = 10*x
```

```
fl_m_contents = open("_m/fs_test.m", 'w')
# Write to File
fl_m_contents.write(stf_m_contents)
# print
```

```
## 18
```

```
fl_m_contents.close()
```

9.1.2.2 Run the Matlab Function from Commandline

- run matlab function from command line
- Retrieving the output of subprocess.call
- https://www.mathworks.com/help/matlab/ref/matlabwindows.html

First, check where matlab is installed:

```
import subprocess
cmd_popen = subprocess.Popen(["where", "matlab"],
                             stdin=subprocess.PIPE,
                             stdout=subprocess.PIPE,
                             stderr=subprocess.PIPE)
output, err = cmd_popen.communicate()
print(output.decode('utf-8'))
```

```
## G:\ProgramData\MATLAB\R2020b\bin\matlab.exe
```

Second, run the matlab file, first definet he parameter x:

```
import os
# print and set directory
print(os.getcwd())
```

```
## G:\repos\Py4Econ
```

```
os.chdir('_m')
print(os.getcwd())
# run matlab script saved prior
# running command line: matlab -batch "fs_test; exit"
```

```
## G:\repos\Py4Econ\_m
```

```
cmd_popen = subprocess.Popen(["matlab", "-batch", "\"x=1; fs_test; exit\""],
                             stdin=subprocess.PIPE,
                             stdout=subprocess.PIPE,
                             stderr=subprocess.PIPE)
output, err = cmd_popen.communicate()
print(output.decode('utf-8'))
```

```
##
## a =
##
##      2
##
##
## b =
##
##     10
##
```

Third, run the function again, but with x=3:

```
os.chdir('_m')
print(os.getcwd())
```

```
## G:\repos\Py4Econ\_m
```

```
print(subprocess.Popen(["matlab", "-batch", "\"x=5; fs_test; exit\""],
                       stdin=subprocess.PIPE,
                       stdout=subprocess.PIPE,
                       stderr=subprocess.PIPE).communicate()[0].decode('utf-8'))
```

```
##
## a =
##
##      6
##
##
## b =
##
##     50
##
```

9.2 File In and Out

9.2.1 Check, Read, Write and Convert Files

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

9.2.1.1 Check if where a Program is.

Suppose we want to generate a commandline call from within python and want to start it in bash. Calling something like "C:/Program Files/Git/git-bash.exe -c COMMANDS". However, depending on the computer that is been used, the git-bash command might be in different spots. How do we find the right path to the *git-bash* file. Accomplish this by using *shutil.which*, which can find the path to different commands, including the *git* command. Given the path that we found, the *git-bash.exe* file is stored in the *Git* folder, two levels up. So we will use pathlib to get to the correct path location. To be safe, go up one level, and then two levels to search for *git-bash.exe*.

First, find the path to the git exe command:

```
# Imports
import os
import shutil
# cmd
st_cmd = 'git'
# Using shutil.which() method to find local path to the *git* command
spn_path_to_git = shutil.which(st_cmd)
# Print result
print(f'{spn_path_to_git=}')
```

```
## spn_path_to_git='G:\\ProgramData\\Git\\cmd\\git.EXE'
```

Second, find the parent and grandparent folders:

```
from pathlib import Path
# Get the parent folder 2 levels up
srt_path_git_parent_folder = Path(spn_path_to_git).parents[0]
srt_path_git_grandparent_folder = Path(spn_path_to_git).parents[1]
# Print
print(f'{srt_path_git_parent_folder=}')
```

```
## srt_path_git_parent_folder=WindowsPath('G:/ProgramData/Git/cmd')
```

```
print(f'{srt_path_git_grandparent_folder=}')
# Search for for the git-bash.exe file in parent and then in the grandparent folder.
```

```
## srt_path_git_grandparent_folder=WindowsPath('G:/ProgramData/Git')
```

Third, search inside parent folder first, and then grand until find the path to *git-bash.exe*. Will put all three steps code together:

```
# required packages
import shutil
from pathlib import Path
# find path to git
st_cmd = 'git'
spn_path_to_git = shutil.which(st_cmd)
# find path to git-bash.exe
spn_path_to_gitbash = ''
for it_up_iter in [0,1]:
    # up-tier folder
    srt_path_git_up_folder = Path(spn_path_to_git).parents[it_up_iter]
    # search
    # get file names in folders (not recursively)
    ls_spn_found_git_bash = [spn_file for spt_srh in [srt_path_git_up_folder]
                             for spn_file in Path(spt_srh).glob('git-bash.exe')]
    # if found, length of ls of founds files must be 1
    if len(ls_spn_found_git_bash) == 1:
        spn_path_to_gitbash = ls_spn_found_git_bash[0]
        break
```

```
if spn_path_to_gitbash == '':
    raise NameError(f'failed to find git-bash, {spn_path_to_git=}')
else:
    print(f'Found git-bash: {spn_path_to_gitbash} by searching around {spn_path_to_git=}')
```

```
## Found git-bash: G:\ProgramData\Git\git-bash.exe by searching around spn_path_to_git='G:\\ProgramD
```

9.2.1.2 Generate a tex file

Will a bare-bone tex file with some texts inside, save inside the *_file* subfolder.

First, create the text text string, note the the linebreaks utomatically generate linebreaks, note that slash need double slash:

```
# Create the Tex Text
# Note that trible quotes begin first and end last lines
stf_tex_contents = """\\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
\\end{document}"""
# Print
print(stf_tex_contents)
```

```
## \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://fanw
## \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
## \end{document}
```

Second, write the contents of the file to a new tex file stored inside the *_file* subfolder of the directory:

```
# Relative file name
srt_file_tex = "_file/"
sna_file_tex = "test_fan"
srn_file_tex = srt_file_tex + sna_file_tex + ".tex"
# Open new file
fl_tex_contents = open(srn_file_tex, 'w')
# Write to File
fl_tex_contents.write(stf_tex_contents)
# print
```

```
## 617
```

```
fl_tex_contents.close()
```

9.2.1.3 Replace Strings in a tex file

Replace a set of strings in the file just generated by a set of alternative strings.

```
# Open file Get text
fl_tex_contents = open(srn_file_tex)
stf_tex_contents = fl_tex_contents.read()
print(srn_file_tex)

# define new and old
```

```
## _file/test_fan.tex
```

```
ls_st_old = ['information', 'Latex']
ls_st_new = ['INFOREPLACE', 'LATEX']

# zip and loop and replace
for old, new in zip(ls_st_old, ls_st_new):
  stf_tex_contents = stf_tex_contents.replace(old, new)
print(stf_tex_contents)

# write to file with replacements
```

```
## \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 INFOREPLACE \href{https://fanw
## \maketitle
## Ipsum INFOREPLACE 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 INFOREPLACE is taken from a village head survey.\footnote{Generally students went
## \end{document}
```

```
sna_file_edited_tex = "test_fan_edited"
srn_file_edited_tex = srt_file_tex + sna_file_edited_tex + ".tex"
fl_tex_ed_contents = open(srn_file_edited_tex, 'w')
fl_tex_ed_contents.write(stf_tex_contents)
```

```
## 617
```

```
fl_tex_ed_contents.close()
```

9.2.1.4 Convert Tex File to Pandoc and Compile Latex

Compile tex file to pdf and clean up the extraneous pdf outputs. See ff_pdf_gen_clean.

```
import subprocess
import os

# Change to local directory so path in tex respected.
os.chdir("C:/Users/fan/py4econ/support/inout")

# Convert tex to pdf
subprocess.call(['C:/texlive/2020/bin/win32/xelatex.exe', '-output-directory',
                 srt_file_tex, srn_file_edited_tex], shell=False)
# Clean pdf extraneous output
```

```
## 0
```

```
ls_st_remove_suffix = ['aux','log','out','bbl','blg']
for st_suffix in ls_st_remove_suffix:
    srn_cur_file = srt_file_tex + sna_file_edited_tex + "." + st_suffix
    if (os.path.isfile(srn_cur_file)):
```

```
        os.remove(srt_file_tex + sna_file_edited_tex + "." + st_suffix)
```

Use pandoc to convert tex file

```
import subprocess

# md file name
srn_file_md = srt_file_tex + "test_fan_edited.md"
# Convert tex to md
subprocess.call(['pandoc', '-s', srn_file_tex, '-o', srn_file_md])
# Open md file
```

```
## 0
```

```
fl_md_contents = open(srn_file_md)
print(fl_md_contents.read())
```

```
## ---
## author:
## - "[Fan Wang](http://fanwangecon.github.io/) [^1]"
## title: A Latex Testing File
## ---
##
## Ipsum information dolor sit amet, consectetur adipiscing elit. Integer
## Latex placerat nunc orci.
##
## #### [Data](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3140132)
##
## Village closure information is taken from a village head survey.[^2]
##
## [^1]: See information
##     [Tex4Econ](https://fanwangecon.github.io/Tex4Econ/) for more.
##
## [^2]: Generally students went to schools.
```

9.2.1.5 Search for Files with Suffix in Several Folders

- python search all files in folders with suffix

Search for files in several directories that have a particular suffix. Then decompose directory into sub-components.

Search file inside several folders (not recursively in subfolders):

```
from pathlib import Path

# directories to search in
ls_spt_srh = ["C:/Users/fan/R4Econ/amto/",
              "C:/Users/fan/R4Econ/function/"]

# get file names in folders (not recursively)
ls_spn_found = [spn_file for spt_srh in ls_spt_srh
                         for spn_file in Path(spt_srh).glob('*.Rmd')]
for spn_found in ls_spn_found:
  print(spn_found)
```

```
## C:\Users\fan\R4Econ\amto\main.Rmd
## C:\Users\fan\R4Econ\function\main.Rmd
```

Search file recursively in all subfolders of folders:

```
from pathlib import Path

# directories to search in
ls_spt_srh = ["C:/Users/fan/R4Econ/amto/array/",
              "C:/Users/fan/R4Econ/amto/list"]

# get file names recursively in all subfolders
ls_spn_found = [spn_file for spt_srh in ls_spt_srh
                         for spn_file in Path(spt_srh).rglob('*.R')]
for spn_found in ls_spn_found:
  drive, path_and_file = os.path.splitdrive(spn_found)
  path_no_suffix = os.path.splitext(spn_found)[0]
  path_no_file, file = os.path.split(spn_found)
  file_no_suffix = Path(spn_found).stem
  print('file:', file, '\ndrive:', drive,
        '\nfile no suffix:', file_no_suffix,
        '\nfull path:', spn_found,
        '\npt no fle:', path_no_file,
        '\npt no suf:', path_no_suffix, '\n')
```

```
## file: fs_ary_basics.R
## drive: C:
## file no suffix: fs_ary_basics
## full path: C:\Users\fan\R4Econ\amto\array\htmlpdfr\fs_ary_basics.R
## pt no fle: C:\Users\fan\R4Econ\amto\array\htmlpdfr
## pt no suf: C:\Users\fan\R4Econ\amto\array\htmlpdfr\fs_ary_basics
##
## file: fs_ary_generate.R
## drive: C:
## file no suffix: fs_ary_generate
## full path: C:\Users\fan\R4Econ\amto\array\htmlpdfr\fs_ary_generate.R
## pt no fle: C:\Users\fan\R4Econ\amto\array\htmlpdfr
## pt no suf: C:\Users\fan\R4Econ\amto\array\htmlpdfr\fs_ary_generate
##
## file: fs_ary_mesh.R
## drive: C:
## file no suffix: fs_ary_mesh
## full path: C:\Users\fan\R4Econ\amto\array\htmlpdfr\fs_ary_mesh.R
## pt no fle: C:\Users\fan\R4Econ\amto\array\htmlpdfr
## pt no suf: C:\Users\fan\R4Econ\amto\array\htmlpdfr\fs_ary_mesh
##
## file: fs_ary_string.R
## drive: C:
## file no suffix: fs_ary_string
## full path: C:\Users\fan\R4Econ\amto\array\htmlpdfr\fs_ary_string.R
## pt no fle: C:\Users\fan\R4Econ\amto\array\htmlpdfr
## pt no suf: C:\Users\fan\R4Econ\amto\array\htmlpdfr\fs_ary_string
##
## file: fs_listr.R
## drive: C:
## file no suffix: fs_listr
## full path: C:\Users\fan\R4Econ\amto\list\htmlpdfr\fs_listr.R
## pt no fle: C:\Users\fan\R4Econ\amto\list\htmlpdfr
## pt no suf: C:\Users\fan\R4Econ\amto\list\htmlpdfr\fs_listr
##
## file: fs_lst_basics.R
## drive: C:
## file no suffix: fs_lst_basics
```

```
## full path: C:\Users\fan\R4Econ\amto\list\htmlpdfr\fs_lst_basics.R
## pt no fle: C:\Users\fan\R4Econ\amto\list\htmlpdfr
## pt no suf: C:\Users\fan\R4Econ\amto\list\htmlpdfr\fs_lst_basics
```

9.2.2 Folder Operations

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

9.2.2.1 Create an Absolute Folder with Path join

Create a platform free full absolute path to a particular folder

```
import os
import pathlib

# suffix
st_suffix = "_mlt_region_ne"
srt_folder = "testfolder" + st_suffix + '_other_stuff'
# path join with os.sep
srt_path = os.path.join(os.sep, "users", "fan", "pyfan", "vig", "support", "inout", "_folder", "test
# Path Name
spn_path = os.path.abspath(srt_path)
# Create the folder
pathlib.Path(spn_path).mkdir(parents=True, exist_ok=True)
# Print
print(f'{srt_folder=}')
```

```
## srt_folder='testfolder_mlt_region_ne_other_stuff'
```

```
print(f'{srt_path=}')
```

```
## srt_path='\\users\\fan\\pyfan\\vig\\support\\inout\\_folder\\testfolder_mlt_region_ne\\subfolder'
```

```
print(f'{spn_path=}')
```

```
## spn_path='G:\\users\\fan\\pyfan\\vig\\support\\inout\\_folder\\testfolder_mlt_region_ne\\subfolde
```

Slash converion, convert the system slash to forward-slash all:

```
srt_folder_slashconverted = spn_path.replace(os.sep, '/')
print(f'{srt_folder_slashconverted=}')
```

```
## srt_folder_slashconverted='G:/users/fan/pyfan/vig/support/inout/_folder/testfolder_mlt_region_ne/
```

see: constructing absolute path with os.path.join().

9.2.2.2 Get the Last Directory in a Path without Some Suffix

Suppose there is a directory with 'abc_suffix_other/subfolder' as the name, generate a new folder that has 'abc' as the folder name without '_suffix'. Generate this folder in the same root folder that the abc_suffix folder resides in.

```
# Absolute path just created:
print(f'{spn_path=}')
# the suffix used
```

```
## spn_path='G:\\users\\fan\\pyfan\\vig\\support\\inout\\_folder\\testfolder_mlt_region_ne\\subfolde
```

```
print(f'{st_suffix=}')
# get path without what comes after suffix
```

```
## st_suffix='_mlt_region_ne'
```

```
spn_path_no_suffix = spn_path[:spn_path.index(st_suffix)]
# Create the folder
pathlib.Path(spn_path_no_suffix).mkdir(parents=True, exist_ok=True)
# Get the new folder name create
spt_root_main, srt_new_subfolder = os.path.split(spn_path_no_suffix)
# Add Slash to new subfolder
spn_path_no_suffix = spn_path_no_suffix + os.sep
# Print
print(f'{spn_path_no_suffix=}')
```

```
## spn_path_no_suffix='G:\\users\\fan\\pyfan\\vig\\support\\inout\\_folder\\testfolder\\'
```

```
print(f'{spt_root_main=}')
```

```
## spt_root_main='G:\\users\\fan\\pyfan\\vig\\support\\inout\\_folder'
```

```
print(f'{srt_new_subfolder=}')
```

```
## srt_new_subfolder='testfolder'
```

9.2.2.3 New Folder and Files

1. create a folder and subfolder
2. create two files in the new folder

```
import pathlib

# folder root
srt_folder = "_folder/"

# new folder
srt_subfolder = srt_folder + "fa/"
# new subfolder
srt_subfolder = srt_subfolder + "faa/"
# generate folders recursively
pathlib.Path(srt_subfolder).mkdir(parents=True, exist_ok=True)

# Open new file
fl_tex_contents_aa = open(srt_subfolder + "file_a.txt", 'w')
# Write to File
fl_tex_contents_aa.write('contents of file a')
```

```
## 18
```

```
fl_tex_contents_aa.close()

# Open another new file and save
fl_tex_contents_ab = open(srt_subfolder + "file_b.txt", 'w')
# Write to File
fl_tex_contents_ab.write('contents of file b')
```

```
## 18
```

```
fl_tex_contents_ab.close()
```

Generate more folders without files:

```
# generate folders recursively
pathlib.Path("_folder/fb/fba/").mkdir(parents=True, exist_ok=True)
# generate folders recursively
pathlib.Path("_folder/fc/").mkdir(parents=True, exist_ok=True)
# generate folders recursively
pathlib.Path("_folder/fd/").mkdir(parents=True, exist_ok=True)
```

9.2.2.4 Copy a File from One Folder to Another

Move the two files from *_folder/fa/faa* to *_folder/faa* as well as to *_folder/fb/faa. *Use* shutil.copy2* so that more metadata is copied over. But *copyfile* is faster.

- How do I copy a file in Python?

Moving one file:

```
import shutil
# Faster method
shutil.copyfile('_folder/fa/faa/file_a.txt', '_folder/fb/file_a.txt')
# More metadat copied, and don't need to specify name
```

```
## '_folder/fb/file_a.txt'
```

```
shutil.copy2('_folder/fa/faa/file_a.txt', '_folder/fb/fba')
```

```
## '_folder/fb/fba\\file_a.txt'
```

9.2.2.5 Copy Folder to Multiple Destimations

Move Entire Folder, How do I copy an entire directory of files into an existing directory using Python?:

```
from distutils.dir_util import copy_tree

# Move contents from fa/faa/ to fc/faa
srt_currroot = '_folder/fa/'
srt_folder = 'faa/'
srt_newroot = '_folder/fc/'

# Full source and destination
srt_sourc = srt_currroot + srt_folder
srt_desct = srt_newroot + srt_folder

# Check/Create new Directory
pathlib.Path(srt_desct).mkdir(parents=True, exist_ok=True)

# Move
copy_tree(srt_sourc, srt_desct)
```

```
## ['_folder/fc/faa/file_a.txt', '_folder/fc/faa/file_b.txt']
```

Move contents to multiple destinations:

```
from distutils.dir_util import copy_tree
# Check/Create new Directory
pathlib.Path('_folder/fd/faa/fa_images').mkdir(parents=True, exist_ok=True)
pathlib.Path('_folder/fd/faa/fb_images').mkdir(parents=True, exist_ok=True)
pathlib.Path('_folder/fd/faa/fc_images').mkdir(parents=True, exist_ok=True)
pathlib.Path('_folder/fd/faa/fz_img').mkdir(parents=True, exist_ok=True)
pathlib.Path('_folder/fd/faa/fz_other').mkdir(parents=True, exist_ok=True)

# Move
copy_tree('_folder/fa/faa/', '_folder/fd/faa/fa_images')
```

```
## ['_folder/fd/faa/fa_images\\file_a.txt', '_folder/fd/faa/fa_images\\file_b.txt']
```

```
copy_tree('_folder/fa/faa/', '_folder/fd/faa/fb_images')
```

```
## ['_folder/fd/faa/fb_images\\file_a.txt', '_folder/fd/faa/fb_images\\file_b.txt']
```

```
copy_tree('_folder/fa/faa/', '_folder/fd/faa/fc_images')
```

```
## ['_folder/fd/faa/fc_images\\file_a.txt', '_folder/fd/faa/fc_images\\file_b.txt']
```

```
copy_tree('_folder/fa/faa/', '_folder/fd/faa/fz_img')
```

```
## ['_folder/fd/faa/fz_img\\file_a.txt', '_folder/fd/faa/fz_img\\file_b.txt']
```

```
copy_tree('_folder/fa/faa/', '_folder/fd/faa/fz_other')
# Empty Folder
```

```
## ['_folder/fd/faa/fz_other\\file_a.txt', '_folder/fd/faa/fz_other\\file_b.txt']
```

```
pathlib.Path('_folder/fd/faa/fd_images').mkdir(parents=True, exist_ok=True)
pathlib.Path('_folder/fd/faa/fe_images').mkdir(parents=True, exist_ok=True)
```

9.2.2.6 Search for Files in Folder

Find the total number of files in a folder.

```
from pathlib import Path

# the number of files in folder found with search critiera
st_fle_search = '*.txt'
ls_spn = [Path(spn).stem for spn in Path('_folder/fd/faa/fa_images').rglob(st_fle_search)]
print(ls_spn)

# count files in a non-empty folder
```

```
## ['file_a', 'file_b']
```

```
srn = '_folder/fd/faa/fa_images'
[spn for spn in Path(srn).rglob(st_fle_search)]
```

```
## [WindowsPath('_folder/fd/faa/fa_images/file_a.txt'), WindowsPath('_folder/fd/faa/fa_images/file_b
```

```
bl_folder_is_empty = len([spn for spn in Path(srn).rglob(st_fle_search)])>0
print(bl_folder_is_empty)

# count files in an empty folder
```

```
## True
```

```
srn = '_folder/fd/faa/fd_images'
[spn for spn in Path(srn).rglob(st_fle_search)]
```

```
## []
```

```
bl_folder_is_empty = len([spn for spn in Path(srn).rglob(st_fle_search)])>0
print(bl_folder_is_empty)
```

```
## False
```

9.2.2.7 Search for Folder Names

- python search for folders containing strings

Search for folders with certain search word in folder name, and only keep if folder actually has files.

```
import os

# get all folder names in folder
ls_spt = os.listdir('_folder/fd/faa/')
print(ls_spt)
```

```
# Select only subfolder names containing _images
```

```
## ['fa_images', 'fb_images', 'fc_images', 'fd_images', 'fe_images', 'fz_img', 'fz_other', '_img']
```

```
srt = '_folder/fd/faa/'
st_search = '_images'
ls_srt_found = [srt + spt
                for spt in os.listdir(srt)
                if st_search in spt]
print(ls_srt_found)
```

```
## ['_folder/fd/faa/fa_images', '_folder/fd/faa/fb_images', '_folder/fd/faa/fc_images', '_folder/fd/
```

9.2.2.8 Find Non-empty Folders by Name

Search:

1. Get subfolders in folder with string in name
2. Only collect if there are files in the subfolder

```
import pathlib

# Select only subfolder names containing _images
srt = '_folder/fd/faa/'
# the folder names must contain _images
st_srt_srh = '_images'
# there must be files in the folder with this string
st_fle_srh = '*.txt'

# All folders that have String
ls_srt_found = [srt + spt
                for spt in os.listdir(srt)
                if st_srt_srh in spt]
print(ls_srt_found)

# All folders that have String and that are nonempty
```

```
## ['_folder/fd/faa/fa_images', '_folder/fd/faa/fb_images', '_folder/fd/faa/fc_images', '_folder/fd/
```

```
ls_srt_found = [srt + spt
                for spt in os.listdir(srt)
                if ((st_srt_srh in spt)
                    and
                    (len([spn for spn
                          in Path(srt + spt).rglob(st_fle_srh)])>0)) ]
print(ls_srt_found)
```

```
## ['_folder/fd/faa/fa_images', '_folder/fd/faa/fb_images', '_folder/fd/faa/fc_images']
```

9.2.2.9 Found Folders to new Folder

1. Search for subfolders by folder name string in a folder
2. Select nonempty subfolders
3. Move nonsempty subfolders to one new folder
4. Move this single combination folder

The results here are implemented as function in the pyfan package: fp_agg_move_subfiles.

```
import pathlib
import os
import shutil
from distutils.dir_util import copy_tree
```

```
# 1 Define Parameters

# Select only subfolder names containing _images
srt = '_folder/fd/faa/'
# the folder names must contain _images
st_srt_srh = '_images'
# there must be files in the folder with this string
st_fle_srh = '*.txt'

# new aggregating folder name
srt_agg = '_img'

# folders to move aggregation files towards
ls_srt_dest = ['_folder/fd/faa/', '_folder/']

# delete source
bl_delete_source = False

# 2 Gather Folders
ls_ls_srt_found = [[srt + spt, spt]
                   for spt in os.listdir(srt)
                   if ((st_srt_srh in spt)
                       and
                       (len([spn for spn
                             in Path(srt + spt).rglob(st_fle_srh)])>0)) ]
print(ls_ls_srt_found)

# 3 Loop over destination folders, loop over source folders
```

```
## [['_folder/fd/faa/fa_images', 'fa_images'], ['_folder/fd/faa/fb_images', 'fb_images'], ['_folder/
```

```
for srt in ls_srt_dest:

  # Move each folder over
  for ls_srt_found in ls_ls_srt_found:

    # Paths
    srt_source = ls_srt_found[0]
    srt_dest = os.path.join(srt, srt_agg, ls_srt_found[1])

    # dest folders
    pathlib.Path(srt_dest).mkdir(parents=True, exist_ok=True)

    # move
    copy_tree(ls_srt_found[0], srt_dest)

# 4. Delete Sources
```

```
## ['_folder/fd/faa/_img\\fa_images\\file_a.txt', '_folder/fd/faa/_img\\fa_images\\file_b.txt']
## ['_folder/fd/faa/_img\\fb_images\\file_a.txt', '_folder/fd/faa/_img\\fb_images\\file_b.txt']
## ['_folder/fd/faa/_img\\fc_images\\file_a.txt', '_folder/fd/faa/_img\\fc_images\\file_b.txt']
## ['_folder/_img\\fa_images\\file_a.txt', '_folder/_img\\fa_images\\file_b.txt']
## ['_folder/_img\\fb_images\\file_a.txt', '_folder/_img\\fb_images\\file_b.txt']
## ['_folder/_img\\fc_images\\file_a.txt', '_folder/_img\\fc_images\\file_b.txt']
```

```
if bl_delete_source:
  for ls_srt_found in ls_ls_srt_found:
    shutil.rmtree(ls_srt_found[0])
```

9.2.3 Parse Yaml

> Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

Use the PyYAML to parse yaml.

9.2.3.1 Write and Create a Simple YAML file

First, Yaml as a string variable:

```
# Create the Tex Text
# Note that trible quotes begin first and end last lines
stf_tex_contents = """\
- file: matrix_matlab
  title: "One Variable Graphs and Tables"
  description: |
    Frequency table, bar chart and histogram.
    R function and lapply to generate graphs/tables for different variables.
  core:
  - package: r
    code: |
      c('word1','word2')
      function()
      for (ctr in c(1,2)) {}
  - package: dplyr
    code: |
      group_by()
  date: 2020-05-02
  output:
    pdf_document:
      pandoc_args: '../_output_kniti_pdf.yaml'
      includes:
        in_header: '../preamble.tex'
  urlcolor: blue
- file: matrix_algebra_rules
  title: "Opening a Dataset"
  titleshort: "Opening a Dataset"
  description: |
    Opening a Dataset.
  core:
  - package: r
    code: |
      setwd()
  - package: readr
    code: |
      write_csv()
  date: 2020-05-02
  date_start: 2018-12-01
- file: matrix_two
  title: "Third file"
  titleshort: "Third file"
  description: |
    Third file description."""
# Print
print(stf_tex_contents)
```

```
## - file: matrix_matlab
##   title: "One Variable Graphs and Tables"
##   description: |
```

```
##     Frequency table, bar chart and histogram.
##     R function and lapply to generate graphs/tables for different variables.
##   core:
##   - package: r
##     code: |
##       c('word1','word2')
##       function()
##       for (ctr in c(1,2)) {}
##   - package: dplyr
##     code: |
##       group_by()
##   date: 2020-05-02
##   output:
##     pdf_document:
##       pandoc_args: '../_output_kniti_pdf.yaml'
##       includes:
##         in_header: '../preamble.tex'
##   urlcolor: blue
## - file: matrix_algebra_rules
##   title: "Opening a Dataset"
##   titleshort: "Opening a Dataset"
##   description: |
##     Opening a Dataset.
##   core:
##   - package: r
##     code: |
##       setwd()
##   - package: readr
##     code: |
##       write_csv()
##   date: 2020-05-02
##   date_start: 2018-12-01
## - file: matrix_two
##   title: "Third file"
##   titleshort: "Third file"
##   description: |
##     Third file description.
```

Second, write the contents of the file to a new tex file stored inside the *_file* subfolder of the directory:

```
# Relative file name
srt_file_tex = "_file/"
sna_file_tex = "test_yml_fan"
srn_file_tex = srt_file_tex + sna_file_tex + ".yml"
# Open new file
fl_tex_contents = open(srn_file_tex, 'w')
# Write to File
fl_tex_contents.write(stf_tex_contents)
# print
```

```
## 908
```

```
fl_tex_contents.close()
```

9.2.3.2 Select Subset of Values by Key

Load Yaml file created prior, the output is a list of dictionaries:

```
import yaml
import pprint
```

```
# Open yaml file
fl_yaml = open(srn_file_tex)
# load yaml
ls_dict_yml = yaml.load(fl_yaml, Loader=yaml.BaseLoader)
# type
type(ls_dict_yml)
```

```
## <class 'list'>
```

```
type(ls_dict_yml[0])
# display
```

```
## <class 'dict'>
```

```
pprint.pprint(ls_dict_yml, width=1)
```

```
## [{'core': [{'code': "c('word1','word2')\n"
##                     'function()\n'
##                     'for '
##                     '(ctr '
##                     'in '
##                     'c(1,2)) '
##                     '{}\n',
##             'package': 'r'},
##            {'code': 'group_by()\n',
##             'package': 'dplyr'}],
##   'date': '2020-05-02',
##   'description': 'Frequency '
##                  'table, '
##                  'bar '
##                  'chart '
##                  'and '
##                  'histogram.\n'
##                  'R '
##                  'function '
##                  'and '
##                  'lapply '
##                  'to '
##                  'generate '
##                  'graphs/tables '
##                  'for '
##                  'different '
##                  'variables.\n',
##   'file': 'matrix_matlab',
##   'output': {'pdf_document': {'includes': {'in_header': '../preamble.tex'},
##                               'pandoc_args': '../_output_kniti_pdf.yaml'}},
##   'title': 'One '
##            'Variable '
##            'Graphs '
##            'and '
##            'Tables',
##   'urlcolor': 'blue'},
##  {'core': [{'code': 'setwd()\n',
##             'package': 'r'},
##            {'code': 'write_csv()\n',
##             'package': 'readr'}],
##   'date': '2020-05-02',
##   'date_start': '2018-12-01',
##   'description': 'Opening '
```

```
##                  'a '
##                  'Dataset.\n',
##   'file': 'matrix_algebra_rules',
##   'title': 'Opening '
##            'a '
##            'Dataset',
##   'titleshort': 'Opening '
##                 'a '
##                 'Dataset'},
##  {'description': 'Third '
##                  'file '
##                  'description.',
##   'file': 'matrix_two',
##   'title': 'Third '
##            'file',
##   'titleshort': 'Third '
##                 'file'}]
```

Select yaml information by *file* name which is a key shared by components of the list:

```
ls_str_file_ids = ['matrix_two']
ls_dict_selected = [dict_yml for dict_yml in ls_dict_yml if dict_yml['file'] in ls_str_file_ids]
pprint.pprint(ls_dc_selected, width=1)
```

```
## [{'date': datetime.date(2020, 5, 2),
##   'description': 'Frequency '
##                  'table, '
##                  'bar '
##                  'chart '
##                  'and '
##                  'histogram',
##   'file': 'mat_matlab',
##   'title': 'One '
##            'Variable '
##            'Graphs '
##            'and '
##            'Tables',
##   'val': 1}]
```

9.2.3.3 Dump List of Dictionary as YAML

- py yaml dump pipe

Given a list of dictionaries, dump values to yaml. Note that dumped output does not use pipe for long sentences, but use single quote and space line, which works with the rmdparrse.py function without problem.

```
ls_dict_selected = [dict_yml for dict_yml in ls_dict_yml
                    if dict_yml['file'] in ['matrix_two','matrix_matlab']]
print(yaml.dump(ls_dict_selected))
```

```
## - core:
##   - code: 'c(''word1'',''word2'')
##
##       function()
##
##       for (ctr in c(1,2)) {}
##
##       '
##     package: r
##   - code: 'group_by()
```

```
##
##       '
##     package: dplyr
##   date: '2020-05-02'
##   description: 'Frequency table, bar chart and histogram.
##
##     R function and lapply to generate graphs/tables for different variables.
##
##     '
##   file: matrix_matlab
##   output:
##     pdf_document:
##       includes:
##         in_header: ../preamble.tex
##       pandoc_args: ../_output_kniti_pdf.yaml
##   title: One Variable Graphs and Tables
##   urlcolor: blue
## - description: Third file description.
##   file: matrix_two
##   title: Third file
##   titleshort: Third file
```

9.3 Install Python

9.3.1 Core Installations

> Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

Use the PyYAML to parse yaml.

9.3.1.1 Git Bash

1. Download and install git

9.3.1.2 Conda Install

1. Download Anaconda for Python 3. For more involved conda instructions see here
2. Get where you installed conda: open up *anaconda prompt* with admin rights (press windows button, and search for anaconda prompt, right click on the resulting terminal icon, choose as admin, a terminal opens up).

```
where python
where anaconda

# C:/ProgramData/Anaconda3/Scripts/anaconda.exe
# C:/ProgramData/Anaconda3/python.exe
```

3. Add to Path: open up windows *Path* and copy the paths found above inside.

9.3.1.2.1 Add To Path Details To Add Anaconda to Path, In Windows

1. Search for: Environment Variables
2. Edit Environment Variables
3. Add new to Path (lower half):
   - C:/ProgramData/Anaconda3/Scripts/
   - C:/ProgramData/Anaconda3/
4. Now open up regular windows command Prompt, Type in: conda –version
5. Close and Open up Git Bash: conda –version

Alternatively, in windows, directly search for Path, and add the python and anaconda exe paths to paths.

9.4 Documentation

9.4.1 Numpy Doc Documentation Guide

Go back to fan's Python Code Examples Repository (bookdown site) or the pyfan Package (API).

- sphinxcontrib-napoleon examples.
- numpydoc examples.
- Documenting Python APIs with docstrings
- Numpy Doc Example
-

9.4.1.1 Parameters

Check types:

```
print(type(111))
print(type('111'))
import logging
print(type(logging.WARNING))
```

Style 1:

```
Parameters
----------
n : int
    The upper limit of the range to generate, from 0 to `n` - 1.
param1 : int
    The first parameter.
param1 : str
    Description of `param1`.
msg : str
    Human readable string describing the exception.
param1 : int
    The first parameter.
param2 : str
    The second parameter.
param3 : str, optional
    The second parameter.
param5: dict
    A dictionary
param6: bool
    boolean
arr1 : ndarray
    2D array containing data with `float` type.
arr2 : ndarray
    1D mask array(containing data with boolean type).
```

Style 2, this will add a link to the types in python doc:

```
Parameters
----------
param2 : :obj:`str`, optional
    The second parameter.
code : :obj:`int`, optional
```

```
    Numeric error code.
param3 : :obj:`int`, optional
    Description of `param3`.
param4 : :obj:`list` of :obj:`str`
    Description of `param2`. Multiple
    lines are supported.
```

For args and kwargs:

```
Parameters
----------
*args
    Variable length argument list.
**kwargs
    Arbitrary keyword arguments.
```

9.4.1.2 Returns

```
Returns
-------
numpy.array of shape (1, it_draws)
    A vector of sorted or unsorted random grid points, or equi-quantile
    points.

Returns
-------
:obj:`tuple` of :obj:`bool`
```

```
Returns
-------
None
```

9.4.1.3 Function Calls

To refer to functions in the same .py file, just need to use: *:func:`log_format`* to refer to function name. For function in different .py files, might need its full path

```
**kwargs
  Arguments for functions that is called, including :func:`log_format`
```

9.4.1.4 Examples

Array outputs.

```
Examples
--------
>>> fl_mu = 0
>>> fl_sd = 1
>>> it_draws = 5
>>> it_seed = 123
>>> fl_lower_sd = -1
>>> fl_higher_sd = 0.8
>>> it_draw_type = 0
>>> ar_draw_random_normal(fl_mu, fl_sd, it_draws,
...                       it_seed, it_draw_type,
...                       fl_lower_sd, fl_higher_sd)
[-1.          0.8         0.2829785 - 1. - 0.57860025]
>>> it_draw_type = 1
>>> ar_draw_random_normal(fl_mu, fl_sd, it_draws,
...                       it_seed, it_draw_type,
```

```
...                       fl_lower_sd, fl_higher_sd)
[-1. - 0.47883617 - 0.06672597  0.3338994   0.8]
>>> it_draw_type = 2
>>> ar_draw_random_normal(fl_mu, fl_sd, it_draws,
...                       it_seed, it_draw_type,
...                       fl_lower_sd, fl_higher_sd)
[-1. - 1. - 0.57860025  0.2829785   0.8]
```

String outputs.

```
Examples
--------
>>> log_vig_start(spt_root = proj_sys_sup.main_directory(),
...               main_folder_name='logvig', sub_folder_name='parameters',
...               subsub_folder_name='combo_type',
...               file_name='fs_gen_combo_type',
...               it_time_format=8, log_level=logging.INFO)
C:\\Users\\fan\\logvig\\parameters\\combo_type\\fs_gen_combo_type_20201030.log.py
```

Appendix A

Index and Code Links

A.1 Data Structures links

A.1.1 Section 1.1 Numbers, Strings, Lists and Tuples links

1. Basic Number Numeric Manipulations: **rmd** | **r** | **pdf** | **html**
   - Loop over a list of numbers where the first and second digits have different interpretations.
   - **py**: *int(np.floor(it_num/10)) + it_num%10*
   - **numpy**: *floor*
2. Define and Unpack Tuple: **rmd** | **r** | **pdf** | **html**
   - Define/deal multiple variables on the same line
   - Define tuple in python with and without parenthesis, unpack tuple, get subset of elements.
   - Access tuple element and fail to mutate tuple element.
   - **py**: *isinstance(tp_abc, tuple)*
3. List Manipulations and Defaults: **rmd** | **r** | **pdf** | **html**
   - Conditional statements based on list length and element value.
   - Provide default for element of a list when list does not have that element.
   - **py**: *lambda + join + append() + if len(X) >= 3 and X[2] is not None + if elif else*
4. Python String Manipulation Examples: **rmd** | **r** | **pdf** | **html**
   - Count unique elements of a string array, generate frequency list.
   - Search for substring, replace string, wrap string.
   - Display and format numeric string with fstring.
   - Change the decimal rounding given a list of estimates and standard error string arrays.
   - **py**: *zip() + upper() + join() + round() + float() + split() + replace() + ascii_lowercase() + set()*
   - **textwrap**: *fill(st, width = 20)*
   - **fstring**: *f + f'{fl_esti_rounded:.{it_round_decimal}f}'*
   - **random**: *choice*

A.1.2 Section 1.2 Dictionary links

1. Python Dictionary Examples and Usages: **rmd** | **r** | **pdf** | **html**
   - Generate a dictionary, loop through a dictionary.
   - List comprehension with dictionary.
   - **py**: *dc = {'key': "name", 'val': 1}*
   - **copy**: *deepcopy*

A.1.3 Section 1.3 Numpy Arrays links

1. Numpy Combine Arrays to Matrix: **rmd** | **r** | **pdf** | **html**
   - Arrays to matrix.
   - **numpy**: *column_stack() + random.choice() + reshape()*

A.2 Pandas links

A.2.1 Section 2.1 Panda Basics links

1. Pandas Generate Dataframes with Random Numeric and String Data: **rmd** | **r** | **pdf** | **html**
   - Generate a dataframe from arrays.
   - Generate a dataframe with random integers as well as random string variables.
   - **np**: *random.randint() + reshape() + column_stack()*
   - **pandas**: *DataFrame()*
2. Python Pandas Conditional Selection of Selectiotn Rows and Columns: **rmd** | **r** | **pdf** | **html**
   - Select subset of rows or columns based on cell value conditions.
   - **pandas**: *pd.DataFrame() + replace(['Zvcss', 'Dugei'], 'Zqovt') + df.loc[df['c5'] == 'Zqovt']*
3. Dataframe Export as CSV with Automatic File Path and Name: **rmd** | **r** | **pdf** | **html**
   - Export a pandas dataframe to csv, store automatically in user home download folder.
   - File name contains the variable name, use fstring to get variable name as file string.
   - **pandas**: *df2export.to_csv(spn_csv_path, sep=",")*
   - **pathlib**: *home() + joinpath() + mkdir(parents=True, exist_ok=True)*
   - **fstring**: *f'{mt_abc=}'.split('=')[0]*
   - **time**: *strftime("%Y%m%d-%H%M%S")*

A.3 Functions links

A.3.1 Section 3.1 Function Arguments and Returns links

1. Python Function Data Type Handling: **rmd** | **r** | **pdf** | **html**
   - Check if parameter is string or integer, conditional execution and exception handling.
   - Check if parameter is string or an integer between some values.
   - **py**: *type + isinstance(abc, str) + isinstance(abc, int) + raise + try except*
2. Tuple and Dictionary as Arguments with args and kwargs: **rmd** | **r** | **pdf** | **html**
   - Update default parameters with dictionary that replaces and appends additional key-value pairs using kwargs.
   - Pass a dictionary for named arguments to a function.
   - Python function None as default for mutable list argument.
   - **python**: *dict3 = {dict1, dict2} + dict1.update(dict2) + func(par1='val1', kwargs)*
3. Command Line Argument Parsing Positional and Optional Arguments: **rmd** | **r** | **pdf** | **html**
   - Parse parameters entered via command line to call a python script.
   - Optional and positional arguments of different data types (int, str, etc.).
   - Default values, allowed list of values.
   - **argparse**: *parser.add_argument() + parser.parse_args()*
4. Function value returns: **rmd** | **r** | **pdf** | **html**
   - Return one or multiple values from function.
   - **python**: *return a, b, c*

A.3.2 Section 3.2 Exceptions links

1. Python Raise, Try and Catch Exceptions: **rmd** | **r** | **pdf** | **html**
   - Raise an Exception in a python function, try and catch and print to string.
   - Trace full exception stack.
   - **python**: *raise + try except + ValueError + TypeError*
   - **traceback**: *print_exc()*

A.4 Statistics links

A.4.1 Section 4.1 Markov Process links

1. Markov Transition Conditional Probability Check Sum to 1: **rmd** | **r** | **pdf** | **html**
   - Generate a sample 50 by 50 markov transition matrix.
   - Check row sums for approximate equality to 1.

- **numpy**: *allclose + reshape + sum*

A.5 Tables and Graphs links

A.5.1 Section 5.1 Matplotlib Base Plots links

1. Mabplotlib Scatter and Line Plots: **rmd** | **r** | **pdf** | **html**
   - Plot several arrays of data, grid, figure title, and line and point patterns and colors.
   - Plot out random walk and white noise first-order autoregressive processes.
   - **matplotlib**: *subplots() + ax.plot() + ax.legend() + ylabel() + xlabel() + title() + grid() + show()*
   - **numpy**: *random.normal() + random.seed() + cumsum() + arange()*
2. Mabplotlib Text Plots: **rmd** | **r** | **pdf** | **html**
   - Print text as figure.
   - **matplotlib**: *ax.text()*
   - **textwrap**: *fill()*
   - **json**: *dump()*

A.6 Amazon Web Services links

A.6.1 Section 6.1 AWS Setup links

1. AWS Account Set-up and Start Instance: **rmd** | **r** | **pdf** | **html**
   - Generate keypair on AWS, launch instance, launch security, ssh access, and AWSCLI.
   - **ssh**: *ssh-agent + ssh-keygen + ssh ec2-user@ec2-52-23-218-117.compute-1.amazonaws.com*
   - **aws**: *aws ec2 start-instances + aws ec2 stop-instances + systemctl start amazon-ssm-agent*
2. Boto3 Client Service Communications: **rmd** | **r** | **pdf** | **html**
   - Start AWS services, send requests etc via boto3.
   - **boto3**: *boto3.client(service, aws_access_key_id, aws_secret_access_key, region_name)*

A.6.2 Section 6.2 S3 links

1. AWS S3 Uploading, Downloading and Syncing, Locally, EC2 and in Docker Container: **rmd** | **r** | **pdf** | **html**
   - From EC2 or local computer upload files to S3 folders.
   - Download sync folders with exclusions between local and S3 folders.
   - Download file from S3 to local computer, an EC2 Linux computer, or into a Docker Container.
   - **py**: *platform.release()*
   - **boto3**: *boto3.client('s3') + s3.upload_file() + s3.download_file()*
   - **os**: *sep*

A.6.3 Section 6.3 Batch links

1. AWS Batch, Batch Array: **rmd** | **r** | **pdf** | **html**
   - Set up python function that uses AWS_BATCH_JOB_ARRAY_INDEX.
   - Register batch task and submit batch array tasks using ECR image, and save results to S3.
   - Batch Array status check until success.
   - **yaml**: *load()*
   - **boto3**: *client() + register_job_definition(jobDefinitionName, type, containerProperties, retryStrategy) + aws_batch.submit_job(jobName, jobQueue, arrayProperties={'size':10}, jobDefinition) + aws_batch.describe_jobs()*

A.7 Docker Container links

A.7.1 Section 7.1 Docker Setup links

1. Docker Container Set-Up and Run on AWS: **rmd** | **r** | **pdf** | **html**
   - Install Docker on AWS and build Docker image.

- Start docker container and run programs inside Docker.
- **aws**: *ssh + yum update -y + amazon-linux-extras install docker -y*
- **docker**: *service docker start + service docker status + docker build + docker images + docker image prune + docker run -t -i fanconda /bin/bash + python /fanProg/invoke/run.py + docker ps -a + docker system df + docker container ls -a*

2. AWS Docker Elastic Container Registery (ECR) Update and Push: **rmd** | **r** | **pdf** | **html**
   - Update and push to Elastic Container Registry (ECR) with newly built Docker image.
   - Pull from Elastic Container Registry docker image.
   - **scp**: *scp -o StrictHostKeyChecking=accept-new -i*
   - **aws**: *aws ecr get-login*
   - **docker**: *docker login + docker build + docker tag + docker push + docker pull*

A.8 Get Data links

A.8.1 Section 8.1 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.9 System and Support links

A.9.1 Section 9.1 Command Line links

1. Execute Python from Command Line and Run Command Line in Python: **rmd** | **r** | **pdf** | **html**
   - Run python functions from command line.
2. Run Matlab Command Line Operations: **rmd** | **r** | **pdf** | **html**
   - Generate a matlab script and run the script with parameters.
   - **subprocess**: *cmd = Popen(ls_str, stdin=PIPE, stdout=PIPE, stderr=PIPE) + cmd.communicate()*
   - **decode**: *decode('utf-8')*
   - **os**: *chdir() + getcdw()*

A.9.2 Section 9.2 File In and Out links

1. Searching for Programs, Reading and Writing to File Examples: **rmd** | **r** | **pdf** | **html**
   - Check the path to a particular installed program.
   - Get the parent folder of the current file.
   - Reading from file and replace strings in file.
   - Convert text file to latex using pandoc and clean.
   - **py**: *open() + write() + replace() + [c for b in [[1,2],[2,3]] for c in b]*
   - **subprocess**: *call()*
   - **pathlib**: *Path().rglob() + Path().stem + Path(spn).parents[1]*
   - **os**: *remove() + listdir() + path.isfile() + path.splitdrive() + path.splitext() + path.split()*
   - **shutil**: *which()*
2. Python Directory and Folder Operations: **rmd** | **r** | **pdf** | **html**
   - Join folder names to form absolute path.
   - Folder path slash conversion from system os.sep to forward slash.
   - Generate new folders and files, with existing folder substrings.
   - Generate subfolder recursively.
   - **py**: *open(srt, 'w') + write() + close()*
   - **os**: *os.sep + os.listdir() + os.path.abspath() + os.path.abspath(os.path.join(os.sep, 'users', 'fan')) + os.path.join('/', 'c:', 'fa', 'fb') + spn_path.replace(os.sep, '/')*
   - **pathlib**: *Path(srt).mkdir(parents=True, exist_ok=True) + [Path(spn).stem for spn in Path(srt).rglob(st)]*

- **shutil**: *shutil.copyfile('/fa/fl.txt', '/fb/fl.txt') + shutil.copy2('/fa/fl.txt', '/fb') + shutil.rmtree('/fb')*
- **distutils**: *dir_util.copy_tree('/fa', '/fb')*

3. Python Yaml File Parsing: **rmd** | **r** | **pdf** | **html**
   - Parse and read yaml files.
   - **yaml**: *load(fl_yaml, Loader=yaml.BaseLoader) + dump()*
   - **pprint**: *pprint.pprint(ls_dict_yml, width=1)*

A.9.3 Section 9.3 Install Python links

1. Basic Conda Setup Instructions: **rmd** | **r** | **pdf** | **html**
   - Conda and git installations
   - **bash**: *where*

A.9.4 Section 9.4 Documentation links

1. Python Documentation Numpy Doc: **rmd** | **r** | **pdf** | **html**
   - Numpy documentation examples.

Bibliography

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.18.