Data Structures and Cloud Services with Python

Fan Wang

2020-11-08

Contents

Preface

The work-in-progress pyfan repository contains:

1. Tutorials and examples for various research tasks: **bookdown site** and **bookdown pdf**.
2. A package for basic data, graph and research tasks: **readthedocs** and **pypi**.

Materials are gathered from various projects in which python code is used for research and paper-administrative tasks. Files are from **Fan**'s pyfan repository which has an associated package. The package functionalize various tasks tested out in the Rmd files. In addition, the pyecon repository and the associated package (readthedocs) contain functions and rmd files related explicitly to solving economic models.

From Fan's other repositories: For dynamic borrowing and savings problems, see Dynamic Asset Repository (Matlab); 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 Array

1.1.1 Strings

Go back to fan's Python Code Examples Repository (bookdown site).

```
import numpy as np
```

1.1.1.1 Print Strings with Numeric Values and Other Strings

After some code segment, print some outputs declaring the end of operation and print results.

```
dc_invoke_main_args = {'speckey': 'ng_s_t',
                       'numeric': 1.46,
                       'ge': False,
                       'multiprocess': False}

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

```
## speckey in dc_invoke_main_args is ng_s_t.
```

```
print(f'numeric in dc_invoke_main_args is {dc_invoke_main_args["numeric"]}.')
```

```
## numeric in dc_invoke_main_args is 1.46.
```

```
print(f'speckey in dc_invoke_main_args is {dc_invoke_main_args}.')
```

```
## speckey in dc_invoke_main_args is {'speckey': 'ng_s_t', 'numeric': 1.46, 'ge': False, 'multiproce
```

1.1.1.2 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']
```

1.1.1.3 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
```

1.1.1.4 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.
```

1.1.1.5 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
```

1.1.2 List

Go back to fan's Python Code Examples Repository (bookdown site).

```
import numpy as np
```

1.1.2.1 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_
```

1.1.2.2 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('='))
```

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

1.1.2.3 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']]
```

1.1.2.4 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]
```

1.1.2.5 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]
```

1.2 Dictionary

1.2.1 Dictionary

Go back to fan's Python Code Examples Repository (bookdown site).

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

1.2.1.1 Loop Through a Dictionary

Given a dictionary, loop through all of its elements

1.2.1.2 Create a List of Dictionaries

```
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
```

1.2.1.3 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

1.2.1.4 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}]
```

1.2.1.5 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'}
```

1.2.1.6 Select by Keys in Dictionary

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}]
```

1.2.1.7 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 Matrix

1.3.1 Generate Matrix from Arrays

Go back to fan's Python Code Examples Repository (bookdown site).

```
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).

```
import numpy as np
import pandas as pd
```

2.1.1.1 Single Arrays to Matrix

Given various arrays, generate a matrix

```
# 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   0   3   1   2   3
## r2   7   4   5   1   5
## r3   9   6   6   8   0
```

Chapter 3

Functions

3.1 Function Arguments

3.1.1 Function Arguments

Go back to fan's Python Code Examples Repository (bookdown site).

```
import pprint
```

3.1.1.1 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}
```

3.1.1.2 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}
```

Chapter 4

Tables and Graphs

4.1 Matplotlib Base Plots

4.1.1 Line and Scatter Plots

Go back to fan's Python Code Examples Repository (bookdown site).

4.1.1.1 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 0x00000200C70A2DF0>]
```

```
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 0x00000200C991D310>]
```

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

```
## <matplotlib.legend.Legend object at 0x00000200C904A4C0>
```

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

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

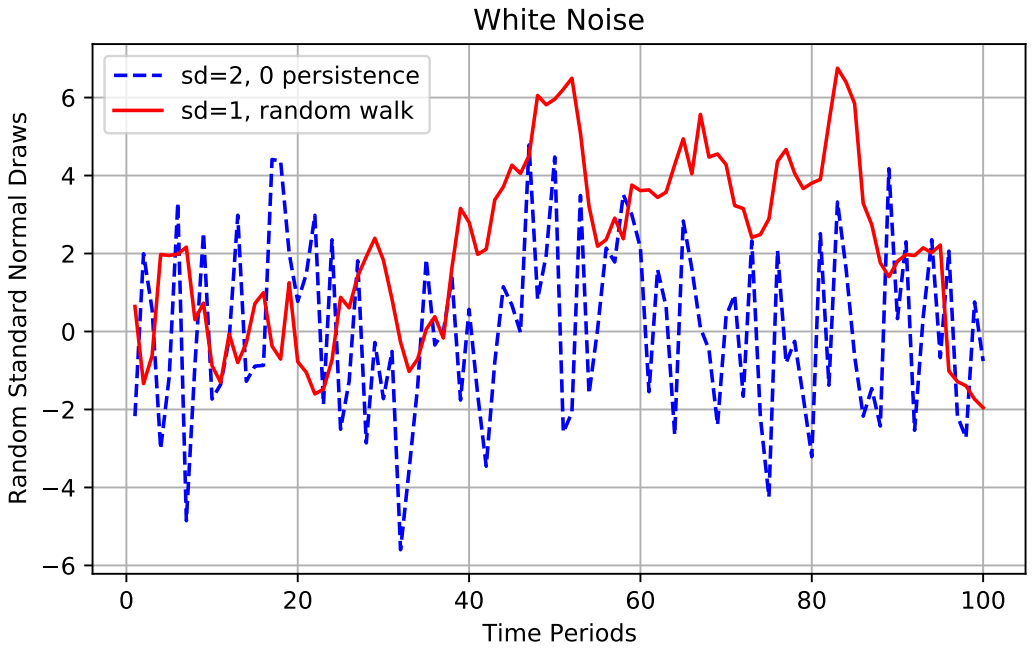
```
plt.xlabel('Time Periods')
```

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

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

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

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



4.1.2 Text Plot

Go back to fan's Python Code Examples Repository (bookdown site).

4.1.2.1 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()
```

Chapter 5

Amazon Web Services

5.1 AWS Setup

5.1.1 AWS Setup

Go back to fan's Python Code Examples Repository (bookdown site).

5.1.1.1 Installation on Local Machine

First install anaconda, git, and associated programs.

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

5.1.1.2 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
```

5.1.1.3 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.

5.1.1.4 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
```

5.1.1.5 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
```

5.1.1.6 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
```

5.1.2 AWS Boto3

Go back to fan's Python Code Examples Repository (bookdown site).

5.1.2.1 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
```

5.1.2.2 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': 0,
##                                                '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'}]}
```

5.2 S3

5.2.1 S3 Usages

Go back to fan's Python Code Examples Repository (bookdown site).

5.2.1.1 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/pyfan/vig/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)
```

5.3 Batch

5.3.1 AWS Batch Run

Go back to fan's Python Code Examples Repository (bookdown site).

5.3.1.1 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}*.

5.3.1.2 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': 'Mon, '
##                                               '09 '
##                                               'Nov '
##                                               '2020 '
##                                               '01:54:13 '
##                                               'GMT',
##                                       'x-amz-apigw-id': 'Vt4i2E2ioAMFejQ=',
##                                       'x-amzn-requestid': '61a71fb4-4742-42cc-b44e-23add2230877',
##                                       'x-amzn-trace-id': 'Root=1-5fa8a145-31067f21240087771f3d5ce
##                        'HTTPStatusCode': 200,
##                        'RequestId': '61a71fb4-4742-42cc-b44e-23add2230877',
##                        'RetryAttempts': 0},
##  'jobDefinitionArn': 'arn:aws:batch:us-east-1:710673677961:job-definition/pyfan-scatterline3-test
##  'jobDefinitionName': 'pyfan-scatterline3-test-rmd',
##  'revision': 16}
```

5.3.1.3 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
response = aws_batch.submit_job(
    jobName=st_jobName,
    jobQueue=st_jobQueue,
    arrayProperties={'size': array_size},
    jobDefinition=jobDefinitionName)

# Print response
```

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

```
## {'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
##                                       'content-length': '198',
##                                       'content-type': 'application/json',
##                                       'date': 'Mon, '
##                                               '09 '
##                                               'Nov '
##                                               '2020 '
##                                               '01:54:13 '
##                                               'GMT',
##                                       'x-amz-apigw-id': 'Vt4i7EkToAMFdGw=',
##                                       'x-amzn-requestid': '249c0c73-0404-4f2c-980b-23535d1a9826',
##                                       'x-amzn-trace-id': 'Root=1-5fa8a145-59074f2e2533c48464280e3
##                       'HTTPStatusCode': 200,
##                       'RequestId': '249c0c73-0404-4f2c-980b-23535d1a9826',
##                       'RetryAttempts': 0},
##  'jobArn': 'arn:aws:batch:us-east-1:710673677961:job/cb72a3d4-4209-4c11-86d0-26614012d093',
##  'jobId': 'cb72a3d4-4209-4c11-86d0-26614012d093',
##  'jobName': 'pyfan-scatterline3-test-rmd_20201108195413089514'}
```

Chapter 6

Docker Container

6.1 Docker Setup

6.1.1 Docker Setup

Go back to fan's Python Code Examples Repository (bookdown site).

6.1.1.1 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
```

6.1.1.2 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"]
```

6.1.1.3 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"

6.1.1.4 Status and Cleaning

6.1.1.4.1 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
```

6.1.1.4.2 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
```

6.1.2 ECR Setup

Go back to fan's Python Code Examples Repository (bookdown site).

6.1.2.1 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
```

6.1.2.2 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/ThaiJMP/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/ThaiJMP/boto3aws/aws_ec2/pem/fan_wang-key-pa
```

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
```

6.1.2.3 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*

6.1.2.4 More Example ECR Code and Outputs

6.1.2.4.1 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])
```

6.1.2.4.2 Example SSM communication

6.1.2.4.3 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\", \"/ThaiJMP/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",
    ""
]
```

6.1.2.4.4 Output from Docker Push

6.1.2.4.5 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
```

Chapter 7

Get Data

7.1 Environmental Data

7.1.1 ECMWF ERA5 Data

Go back to fan's Python Code Examples Repository (bookdown site).

7.1.1.1 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
```

7.1.1.2 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
```

7.1.1.3 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

7.1.1.4 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.

7.1.1.5 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
```

7.1.1.6 More Advanced Download Setup and Instructions

7.1.1.6.1 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.

7.1.1.6.2 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"
```

7.1.1.7 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

7.1.1.7.1 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/pyfan/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

7.1.1.8 Learning

7.1.1.8.1 Terminologies Links:

- status of the CDS queue.

Terminologies:

- single level parameters

7.1.1.8.2 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.

7.1.1.9 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))")
```

```r
# 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 8

System and Support

8.1 Command Line

8.1.1 Python Command Line

Go back to fan's Python Code Examples Repository (bookdown site).

8.1.1.1 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'))
```

```
## C:\ProgramData\Anaconda3\envs\wk_pyfan\python.exe
## C:\ProgramData\Anaconda3\python.exe
## C:\Users\fan\.windows-build-tools\python27\python.exe
## C:\Users\fan\AppData\Local\Microsoft\WindowsApps\python.exe
## C:\Program Files\Inkscape\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)
```

```
## Data Structures and Cloud Services with Python.Rmd
## Data-Structures-and-Cloud-Services-with-Python.Rmd
## Data-Structures-and-Cloud-Services-with-Python_files
## LICENSE
## README.md
```

```
## README_appendix.md
## README_end.md
## README_pre.md
## README_toc.md
## _bookdown.yml
## _bookdown_files
## _config.yml
## _file
## _folder
## _log
## _m
## _output.yml
## _output_kniti_html.yaml
## _output_kniti_pdf.yaml
## _templates
## book.bib
## bookdown
## build
## dist
## doc
## hdga.html
## htmlpdfr
## img
## index.Rmd
## packages.bib
## poetry.lock
## preamble.tex
## preamble_book.tex
## pyfan
## pyfan.egg-info
## pyproject.toml
## setup.py
## style.css
## tests
## title.Rmd
## vig
```

```
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)
```

```
## 565
```

```
fl_tex_contents.close()
```

8.1.1.2 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
```

8.1.2 Run Matlab Functions

Go back to fan's Python Code Examples Repository (bookdown site).

8.1.2.1 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()
```

8.1.2.2 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'))
```

```
## C:\Program Files\MATLAB\R2020b\bin\matlab.exe
## C:\Program Files\MATLAB\R2019b\bin\matlab.exe
```

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

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

```
## C:\Users\fan\pyfan
```

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

```
## C:\Users\fan\pyfan\_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())
```

```
## C:\Users\fan\pyfan\_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
##
```

8.2 File In and Out

8.2.1 Read and Write and Convert

Go back to fan's Python Code Examples Repository (bookdown site).

- python create a text file
- python write file from paragraphs

8.2.1.1 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()
```

8.2.1.2 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()
```

8.2.1.3 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/pyfan/vig/support/inout")

# Convert tex to pdf
subprocess.call(['C:/texlive/2019/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
```

```
## ---
##
## \maketitle
## 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.
```

8.2.1.4 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
```

8.2.2 Folder Operations

Go back to fan's Python Code Examples Repository (bookdown site).

8.2.2.1 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)
```

8.2.2.2 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'
```

8.2.2.3 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)
```

8.2.2.4 Search for Files in Folder

Find the total number of files in a folder.

```
import pathlib
# 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
```

8.2.2.5 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/
```

8.2.2.6 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']
```

8.2.2.7 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])
```

8.2.3 Parse Yaml

Go back to fan's Python Code Examples Repository (bookdown site).

Use the PyYAML to parse yaml.

8.2.3.1 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()
```

8.2.3.2 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}]
```

8.2.3.3 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
```

8.3 Install Python

8.3.1 Core Installations

Go back to fan's Python Code Examples Repository (bookdown site).

Use the PyYAML to parse yaml.

8.3.1.1 Git Bash

1. Download and install git

8.3.1.2 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.

8.3.1.2.1 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.

8.4 Documentation

8.4.1 Numpy Doc Documentation Guide

Go back to fan's Python Code Examples Repository (bookdown site).

- sphinxcontrib-napoleon examples.
- numpydoc examples.
- Documenting Python APIs with docstrings
-

8.4.1.1 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
```

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.
```

8.4.1.2 Returns

```
Returns
-------
numpy.array of shape (1, it_draws)
    A vector of sorted or unsorted random grid points, or equi-quantile
    points.
```

```
Returns
-------
None
```

8.4.1.3 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 Array links

1. Python String Manipulation Examples: **rmd** | **r** | **pdf** | **html**
   - Search for substring, replace string, wrap string.
   - **py**: *zip() + upper() + fstring*
   - **textwrap**: *fill(st, width = 20)*
   - **fstring**: *f*
2. 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*

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 Matrix 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 Matrix to Dataframes: **rmd** | **r** | **pdf** | **html**
   - Arrays to matrix.
   - **pandas**: *pd.DataFrame()*

A.3 Functions links

A.3.1 Section 3.1 Function Arguments links

1. 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 paris using kwargs.
- Pass a dictionary for named arguments to a function.
- **python**: *dict3 = {dict1, dict2} + dict1.update(dict2) + func(par1='val1', kwargs)*

A.4 Tables and Graphs links

A.4.1 Section 4.1 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.5 Amazon Web Services links

A.5.1 Section 5.1 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.5.2 Section 5.2 S3 links

1. AWS S3 Uploading Downloading and Syncing: **rmd** | **r** | **pdf** | **html**
   - From EC2 or local computer upload files to S3 folders.
   - Download sync folders with exclusions between local and S3 folders.
   - **py**: *platform.release()*
   - **boto3**: *boto3.client('s3') + s3.upload_file*

A.5.3 Section 5.3 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.
   - **yaml**: *load()*
   - **boto3**: *client() + register_job_definition(jobDefinitionName, type, containerProperties, retryStrategy) + aws_batch.submit_job(jobName, jobQueue, arrayProperties={'size':10}, jobDefinition)*

A.6 Docker Container links

A.6.1 Section 6.1 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.7 Get Data links

A.7.1 Section 7.1 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.8 System and Support links

A.8.1 Section 8.1 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.8.2 Section 8.2 File In and Out links

1. Python Reading and Writing to File Examples: **rmd** | **r** | **pdf** | **html**
   - Reading from file and replace strings in file.
   - Convert text file to latex using pandoc and clean.
   - Search for files in several folders with file substring.
   - Get path root, file name, file stem, etc from path.
   - **py**: *open() + write() + replace() + [c for b in [[1,2],[2,3]] for c in b]*
   - **subprocess**: *call()*
   - **pathlib**: *Path().rglob() + Path().stem*
   - **os**: *remove() + listdir() + path.isfile() + path.splitdrive() + os.path.splitext() + os.path.split()*
2. Python Directory and Folder Operations: **rmd** | **r** | **pdf** | **html**
   - Generate new folders and files.
   - Generate subfolder recursively.
   - Copying and moving files across folders.
   - Aggregate subfolders into a folder and move.
   - **py**: *open(srt, 'w') + write() + close()*
   - **os**: *os.listdir() + os.path.join('/', 'c:', 'fa', 'fb')*
   - **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.8.3 Section 8.3 Install Python links

1. Basic Conda Setup Instructions: **rmd** | **r** | **pdf** | **html**
   - Conda and git installations
   - **bash**: *where*

A.8.4 Section 8.4 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.