Yilun Wang(yilun830@bu.edu) - ASSIGNMENT #3

convert notebook to html then print as PDF

In [126]:

```
!jupyter nbconvert --to html /content/Assignment_03_Yilun_Wang.ipynb
```

```
/bin/bash: -c: line 0: syntax error near unexpected token `('
/bin/bash: -c: line 0: `jupyter nbconvert --to html /content/Assignm
ent_03_Yilun_Wang_(yilun830_bu_edu).ipynb'
```

The main content of assignment 03

In [109]:

```
%matplotlib inline
```

In [110]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

In [111]:

```
data = pd.read_csv('https://raw.githubusercontent.com/ChasteloveCNN/ba765-sessio
n02/main/100-Stocks-Returns.csv')
```

In [112]:

```
# upload ff factors of year 2017-2021
ff_factors = pd.read_csv('https://raw.githubusercontent.com/ChasteloveCNN/ba765-
session02/main/FF-Factors-2017-2021.csv')
```

In [113]:

```
# upload tickers
tickers = pd.read_csv('https://raw.githubusercontent.com/ChasteloveCNN/ba765-ses
sion02/main/Assign3-100-Tickers-Final3.csv')
```

In [114]:

```
tickers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   TICKER  100 non-null    object
dtypes: object(1)
memory usage: 928.0+ bytes
```

In [115]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   PERMNO  6000 non-null   int64  
 1   date    6000 non-null   int64  
 2   TICKER  6000 non-null   object 
 3   RET     6000 non-null   float64
dtypes: float64(1), int64(2), object(1)
memory usage: 187.6+ KB
```

In [116]:

```
# print out the header
data.head()
```

Out[116]:

	PERMNO	date	TICKER	RET
0	10220	20170131	BWXT	0.045088
1	10220	20170228	BWXT	0.119306
2	10220	20170331	BWXT	0.026916
3	10220	20170428	BWXT	0.032983
4	10220	20170531	BWXT	-0.009355

In [117]:

```
# print out the tail
data.tail()
```

Out[117]:

	PERMNO	date	TICKER	RET
5995	90720	20210831	BLDR	0.197528
5996	90720	20210930	BLDR	-0.029086
5997	90720	20211029	BLDR	0.126208
5998	90720	20211130	BLDR	0.191694
5999	90720	20211231	BLDR	0.234303

In [118]:

```
# check ff factors info
ff_factors.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   dateff  60 non-null     int64  
 1   mktrf   60 non-null     float64
 2   smb     60 non-null     float64
 3   hml     60 non-null     float64
 4   rf      60 non-null     float64
dtypes: float64(4), int64(1)
memory usage: 2.5 KB
```

In [119]:

```
ff_factors.head(10)
```

Out[119]:

	dateff	mktrf	smb	hml	rf
0	20170131	0.0194	-0.0113	-0.0274	0.0004
1	20170228	0.0357	-0.0204	-0.0167	0.0004
2	20170331	0.0017	0.0113	-0.0333	0.0003
3	20170428	0.0109	0.0072	-0.0213	0.0005
4	20170531	0.0106	-0.0252	-0.0375	0.0006
5	20170630	0.0078	0.0223	0.0149	0.0006
6	20170731	0.0187	-0.0146	-0.0022	0.0007
7	20170831	0.0016	-0.0165	-0.0207	0.0009
8	20170929	0.0251	0.0445	0.0309	0.0009
9	20171031	0.0225	-0.0193	0.0022	0.0009

Rename date column to "date" to match WRDS data "date" column for each stock

In [120]:

```
# Rename date column to "date" to match WRDS data "date" column for each stock
ff_factors.rename(columns={'dateff':'date'}, inplace=True)
ff_factors.head()
```

Out[120]:

	date	mktrf	smb	hml	rf
0	20170131	0.0194	-0.0113	-0.0274	0.0004
1	20170228	0.0357	-0.0204	-0.0167	0.0004
2	20170331	0.0017	0.0113	-0.0333	0.0003
3	20170428	0.0109	0.0072	-0.0213	0.0005
4	20170531	0.0106	-0.0252	-0.0375	0.0006

Create loops for each stock to get new dataframe for each stock's monthly data

create a vacant dataframe for final outputs

In [121]:

```
# create a vacant dataframe for final outputs
final_outputs = pd.DataFrame(columns=['TICKER', 'R-squared', 'Adj. R-squared','c
onst','mktrf','smb','hml'])
final_outputs.head()
```

Out[121]:

TICKER	R-squared	Adj. R-squared	const	mktrf	smb	hml

create loop

In [122]:

```
for x in tickers['TICKER']:
  # create new df for x stock
  x_data = data[data["TICKER"] == x]

  # merge x_data & ff_factors for each x
  x_ff = pd.merge(x_data, ff_factors, on='date', how='outer')

  # Run OLS regression for each x (60 months) using FF 3-factor model
  y = x_ff["RET"] - x_ff["rf"]
  X = x_ff[['mktrf' , 'smb' , 'hml']]

  # Use statsmodels
  X = sm.add_constant(X) # adding a constant
  model = sm.OLS(y, X).fit()

  # create a new df for our necessary parameters
  # model.params has 4 numbers: alpha/mktrf/smb/hml
  # tickers = X'name, and we also have other two r-squareds.
  # As a result, we get a new df with 7 parameters as following:
  new=pd.DataFrame({'const': [model.params[0]],
                    'mktrf': [model.params[1]],
                    'smb': [model.params[2]],
                    'hml': [model.params[3]],
                    'TICKER': [x],
                    'R-squared': [model.rsquared],
                    'Adj. R-squared': [model.rsquared_adj]
                    })

  # Store the above items (TICKER, R-squared, Adj. R-squared, const, mktrf, smb,
hml) to a row in the final_outputs dataframe.
  final_outputs=final_outputs.append(new,ignore_index=True)
  # finish!
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:1
17: FutureWarning: In a future version of pandas all arguments of co
ncat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

In [123]:

```
# check our final results
final_outputs
```

Out[123]:

	TICKER	R-squared	Adj. R-squared	const	mktrf	smb	hml
0	B	0.444968	0.415234	-0.007043	1.005025	0.575441	0.467812
1	BA	0.373709	0.340158	0.000422	1.288971	0.320766	0.856764
2	BAC	0.830229	0.821134	0.004207	1.297279	-0.111081	0.948497
3	BAH	0.333477	0.297770	0.002267	0.764677	-0.241293	-0.486009
4	BAM	0.564023	0.540667	0.005898	1.192238	-0.025136	0.363742
...	...	...	...	...	...	...	...
95	CAKE	0.582309	0.559932	-0.008548	1.241215	1.147694	1.033762
96	CAL	0.491911	0.464692	-0.008616	2.010956	1.691033	1.102032
97	CALA	0.109994	0.062315	-0.032454	1.791782	-0.055778	-1.243154
98	CALM	0.077586	0.028171	-0.000832	-0.185303	0.610488	-0.343051
99	CAMP	0.604168	0.582962	-0.027340	2.130014	1.106197	0.494313

100 rows × 7 columns

Write the contents of the dataframe to a CSV file with the name "Assign3-Output.csv"

In [124]:

```
final_outputs.to_csv("Assign3-Output.csv")
```