5QQMN534: Algorithmic Finance

Week9: Statistical Analysis and OLS Regression: CAPM and Multifactor Models

Eryk Lewinson - Python for Finance Cookbook 2020: Chapter 4

Agenda

- Implementing CAPM in Python
- Implementing Fama French Three Factor Model
- Implementing Rolling Fama French Three Factor Model
- Implementing the Four and Five Factor Model

Introduction

- This chapter is devoted to estimating various factor models in Python. The idea behind these models is to explain the excess returns (over the risk-free rate) of a certain portfolio or asset using one or more factors (features).
- These risk factors can be considered a tool for understanding the cross-section of (expected) returns.

- In general, factor models can be used to identify interesting assets that can be added to the investment portfolio, which— in turn—should lead to better performing portfolios.

- By the end of this chapter, we will have constructed some of the most popular factor models. We will start with the simplest, yet very popular, one-factor model (the **Capital Asset Pricing Model**, or **CAPM**) and then explain how to estimate more advanced three-, four-, and five-factor models.

- We will also cover the interpretation of what these factors represent and give a high-level overview of how they are constructed. In this chapter, we cover the following codes:
    - Implementing the CAPM in Python
    - Implementing the Fama-French three-factor model in Python
    - Implementing the rolling three-factor model on a portfolio of assets
    - Implementing the four- and five-factor models in Python

Implementing CAPM in Python: Introduction

- We learn how to estimate the famous **Capital Asset Pricing Model** (**CAPM**) and obtain the beta coefficient.
- This model represents the relationship between the expected return on a risky asset and the market risk (also known as systematic or undiversifiable risk).
- CAPM can be considered a one-factor model, on top of which more complex factor models were built.

CAPM is represented by the following equation:

$$E(r_i) = r_f + \beta_i(E(r_m) - r_f)$$

- Here, *E(ri)* denotes the expected return on asset *i*, *rf* is the risk-free rate (such as a
- government bond), *E(rm)* is the expected return on the market, and is the beta coefficient.

Implementing CAPM in Python: Beta

Beta can be interpreted as the level of the asset return's sensitivity, as compared to the market in general. Some possible examples include:

- **beta <= -1:** The asset moves in the opposite direction as the benchmark and in a greater amount than the negative of the benchmark.

- **-1 < beta < 0**: The asset moves in the opposite direction to the benchmark.

- **beta = 0:** There is no correlation between the asset's price movement and the market benchmark.

- **0 < beta < 1:** The asset moves in the same direction as the market, but the amount is smaller. An example might be the stock of a company that is not very susceptible to day-to-day fluctuations.
- **beta = 1:** The asset and the market are moving in the same direction by the same amount.
- **beta > 1:** The asset moves in the same direction as the market, but the amount is greater. An example might be the stock of a company that is very susceptible to day-to-day market news.

Implementing CAPM in Python: Continued

CAPM can also be represented:

$$E(r_i) - r_f = \beta_i(E(r_m) - r_f)$$

- Here, the left-hand side of the equation can be interpreted as the risk premium, while the right-hand side contains the market premium.
- The same equation can be reshaped into:

$$\beta = \frac{cov(R_i, R_m)}{var(R_m)}$$

Here, $R_i = E(r_i) - r_f$ and $R_m = E(r_m) - r_f$.

- In this example, we consider the case of Amazon and assume that the S&P 500 index represents the market.
- We use 5 years (2014-2018) of monthly data to estimate the beta.
- *In current times, the risk-free rate is so low that, for simplicity's sake, we assume it is equal to zero.
- *Note at time of book publication this was true. However now, the risk free rate is considerably higher. For alignment the examples are kept in line with the code and book.

Implementing CAPM in Python: Code1

1. Import the libraries:

```
import pandas as pd
import yfinance as yf
import statsmodels.api as sm
```


2. Specify the risky asset and the time horizon:

```
RISKY_ASSET = 'AMZN'
MARKET_BENCHMARK = '^GSPC'
START_DATE = '2014-01-01'
END_DATE = '2018-12-31'
```


3. Download the necessary data from Yahoo Finance:

```
df = yf.download([RISKY_ASSET, MARKET_BENCHMARK],
                 start=START_DATE,
                 end=END_DATE,
                 adjusted=True,
                 progress=False)
```

```
df = yf.download([RISKY_ASSET, MARKET_BENCHMARK],
                 start=START_DATE,
                 end=END_DATE,
                 auto_adjust=True, # Auto Adjust will overwrite the Close price by Adj Close.
                 progress=False) # Progress will show a progress bar.

print(f'Downloaded {df.shape[0]} rows of data.')
help(yf.download)
```

Implementing CAPM in Python: Code2

X = df['Close'] changed due to prior code library parameter change / update

4. Resample to monthly data and calculate the simple returns:

```
X = df['Adj Close'].rename(columns={RISKY_ASSET: 'asset',
                                    MARKET_BENCHMARK: 'market'}) \
                   .resample('M') \
                   .last() \
                   .pct_change() \
                   .dropna()
```


5. Calculate beta using the covariance approach:

```
covariance = X.cov().iloc[0,1]
benchmark_variance = X.market.var()
beta = covariance / benchmark_variance
```

Implementing CAPM in Python: Code3

Null Hypothesis:	$H_0 : \beta_1 = 0$
Alternative Hypotheses:	$H_1 : \beta_1 \neq 0.$

- These results indicate that the beta (denoted as market here) is equal to 1.62, which means that Amazon's returns are 62% more volatile than the market (proxied by S&P 500).
- The value of the intercept is relatively small and statistically **insignificant** at the 5% significance level.
- The p-values for the coefficients indicate whether these relationships are statistically significant.
- P value is 0.056 > 0.05% therefore insignificant at 95%.

6. Prepare the input and estimate the CAPM as a linear regression:

```
y = X.pop('asset')
X = sm.add_constant(X)

capm_model = sm.OLS(y, X).fit()
print(capm_model.summary())
```



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  asset   R-squared:                       0.359
Model:                            OLS   Adj. R-squared:                  0.347
Method:                 Least Squares   F-statistic:                     31.87
Date:                Thu, 02 Sep 2021   Prob (F-statistic):           5.44e-07
Time:                        12:57:43   Log-Likelihood:                 75.306
No. Observations:                  59   AIC:                            -146.6
Df Residuals:                      57   BIC:                            -142.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0178      0.009      1.950      0.056      -0.000      0.036
market         1.6292      0.289      5.645      0.000       1.051      2.207
==============================================================================
Omnibus:                        3.037   Durbin-Watson:                   1.864
Prob(Omnibus):                  0.219   Jarque-Bera (JB):                2.353
Skew:                           0.481   Prob(JB):                        0.308
Kurtosis:                       3.182   Cond. No.                         32.3
==============================================================================
```

Interpreting OLS Results Reminder

- Please see separate **Understanding OLS results.pdf**

- Understanding t tests and statical results helpful resources: https://towardsdatascience.com/the-statistical-analysis-t-test-explained-for-beginners-and-experts-fd0e358bbb62
- https://www.geeksforgeeks.org/interpreting-the-results-of-linear-regression-using-ols-summary/
- https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

Implementing CAPM in Python: Alpha

Definition:

$$\alpha_p = \overline{r_p} - \left[\overline{r_f} + \beta_p\left(\overline{r_m} - \overline{r_f}\right)\right]$$

where:

$\overline{r_f} + \beta_p\left(\overline{r_m} - \overline{r_f}\right)$:expected return of the portfolio, **using CAPM**, given portfolio's beta and average market return

α_p :portfolio's alpha

β_p :Portfolio β

Intuition: Average return on the portfolio over and above that predicted by CAPM

Implementing CAPM in Python: Explanation1

- First, we specified the assets (Amazon and S&P 500) we wanted to use and the time frame.
- In ***Step 3***, we downloaded the data from Yahoo Finance. Then, we only kept the last available price per month and calculated the monthly returns as the percentage change between the subsequent observations.
- In ***Step 5***, we calculated the beta as the ratio of the covariance between the risky asset and the benchmark to the benchmark's variance.
- In ***Step 6***, we separated the target (Amazon's stock returns) and the features (S&P 500 returns) using the `pop` method of a pandas DataFrame.
- Afterward, we added the constant to the features (effectively adding a column of ones) with the `add_constant` function.
- The idea behind adding the intercept to this regression is to investigate whether—after estimating the model—the intercept (in the case of the CAPM, also known as Jensen's alpha) is zero.

Implementing CAPM in Python: Explanation2

- If it is positive and significant, it means that– assuming the CAPM model is true—the asset or portfolio generates abnormally high risk-adjusted returns.
- There are two possible implications—either the market is inefficient or there are some other undiscovered risk factors that should be included in the model. This issue is known as the **joint hypothesis problem**.

We can also use the formula notation, which adds the constant automatically. To do so, we must `import statsmodels.formula.api as smf` and then run the slightly modified `capm_model = smf.ols(formula='asset ~ market', data=X).fit()`. The results of both approaches are the same.

Null Hypothesis: $H_0 : \beta_1 = 0.$

Alternative Hypotheses: $H_1 : \beta_1 \neq 0.$

- Lastly, we ran the OLS regression and printed the summary.
- Here, we could see that the coefficient by the market variable (that is, the CAPM beta) is equal to the beta that was calculated using the covariance between the asset and the market in *Step 5*.

Implementing CAPM in Python: Risk Free

- In the main example, we assumed there was no risk-free rate, which is a reasonable assumption to make nowadays. *see prior comment
- However, there might be cases when we would like to account for a non-zero risk-free rate.
- In this section, we present three possible approaches:

Implementing CAPM in Python: Approach1

Using data from prof. Kenneth French's website:

- The market premium (*rm- rf*) and the risk-free rate (approximated by the one-month Treasury bill) can be downloaded from Prof. Kenneth French's website (please refer to the *See also* section of this code for the link).
- Please bear in mind that the definition of the market benchmark used by prof. French is different than the S&P 500 index – a detailed description is available on his website.
- For a description of how to easily download the data, please refer to the *Implementing the Fama-French threefactor model in Python* code. This will be covered later.

Implementing CAPM in Python: Approach2

- The second option is to approximate the risk-free rate with, for example, the 13 Week (3-month) Treasury Bill (Yahoo finance ticker: ^IRX).
- Follow these steps to learn how to download the data and convert it into the appropriate risk-free rate.

1. Define the length of the period in days:

```
N_DAYS = 90
```

https://uk.finance.yahoo.com/quote/%5EIRX?p=%5EIRX&.tsrc=fin-srch

2. Download the data from Yahoo Finance:


```
df_rf = yf.download('^IRX', start=START_DATE, end=END_DATE)
```


3. Resample the data to monthly frequency (by taking the last value for each month):


```
rf = df_rf.resample('M').last().Close / 100
```

4. Calculate the risk-free return (expressed as daily values) and convert the values to monthly:

```
rf = ( 1 / (1 - rf * N_DAYS / 360) )**(1 / N_DAYS)
rf = (rf ** 30) - 1
```

$$r = \left(\frac{1}{1-\frac{dn}{360}}\right)^{1/n}$$

r is the daily risk-free return ratio
dn = N_DAYS
n = N_DAYS

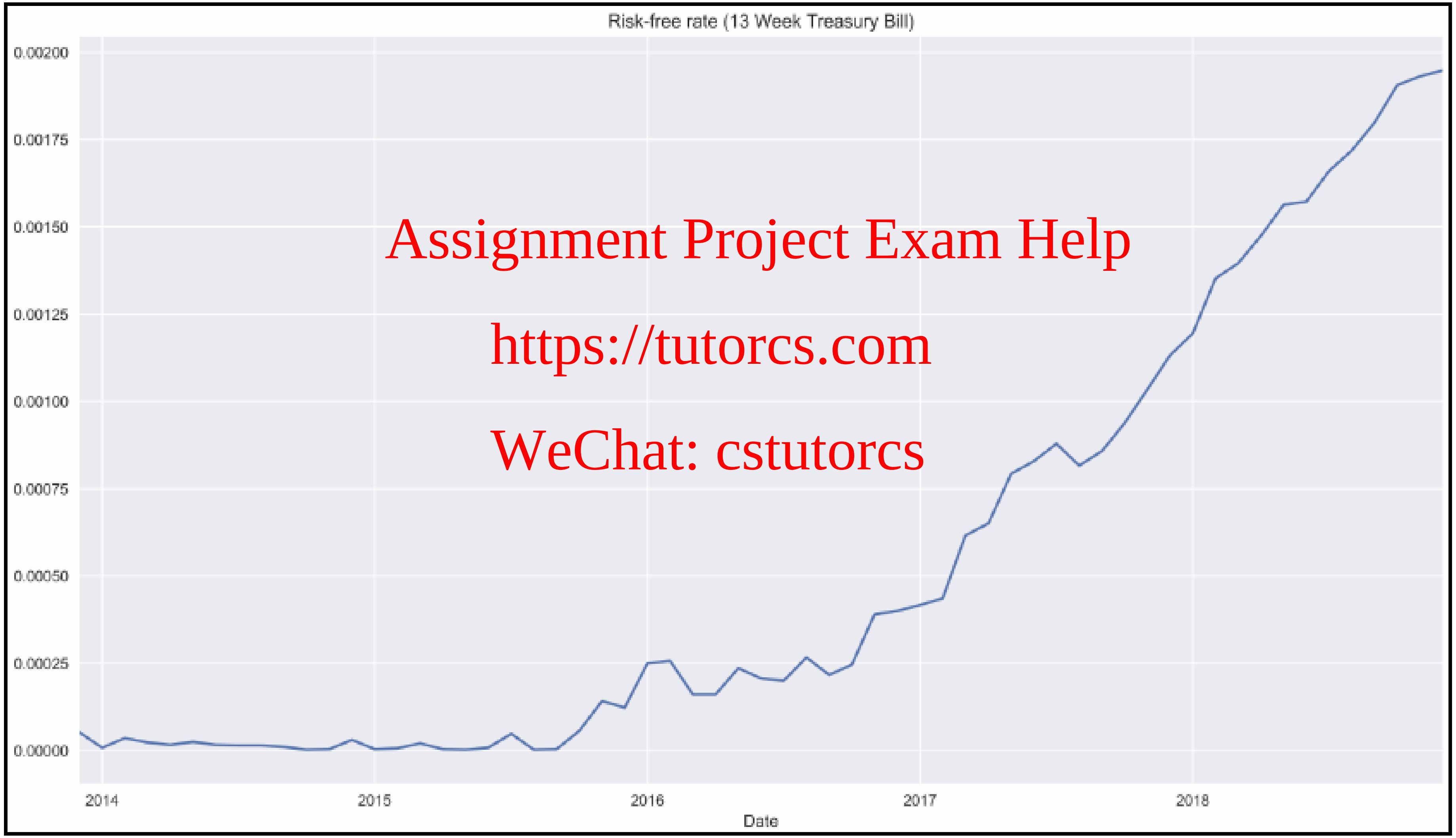
5. Plot the calculated risk-free rate:

```
rf.plot(title='Risk-free rate (13 Week Treasury Bill)')
```

The following plot shows the visualization of the risk-free rate over time:

https://quant.stackexchange.com/questions/33076/how-to-calculate-daily-risk-free-rate-using-13-week-treasury-bill

Implementing CAPM in Python: Approach2



Implementing CAPM in Python: Approach3

- The last approach is to approximate the risk-free rate using the 3-Month Treasury Bill (Secondary Market Rate), which can be downloaded from the **Federal Reserve Economic Data** (**FRED**) database.
- Follow these steps to learn how to download the data and convert it to a monthly risk-free rate:

1. Import the library:

```
import pandas_datareader.data as web
```

2. Download the data from the FRED database:

```
rf = web.DataReader('TB3MS', 'fred', start=START_DATE,
                    end=END_DATE)
```

3. Convert the obtained risk-free rate to monthly values:

```
rf = (1 + (rf / 100)) ** (1 / 12) - 1
```

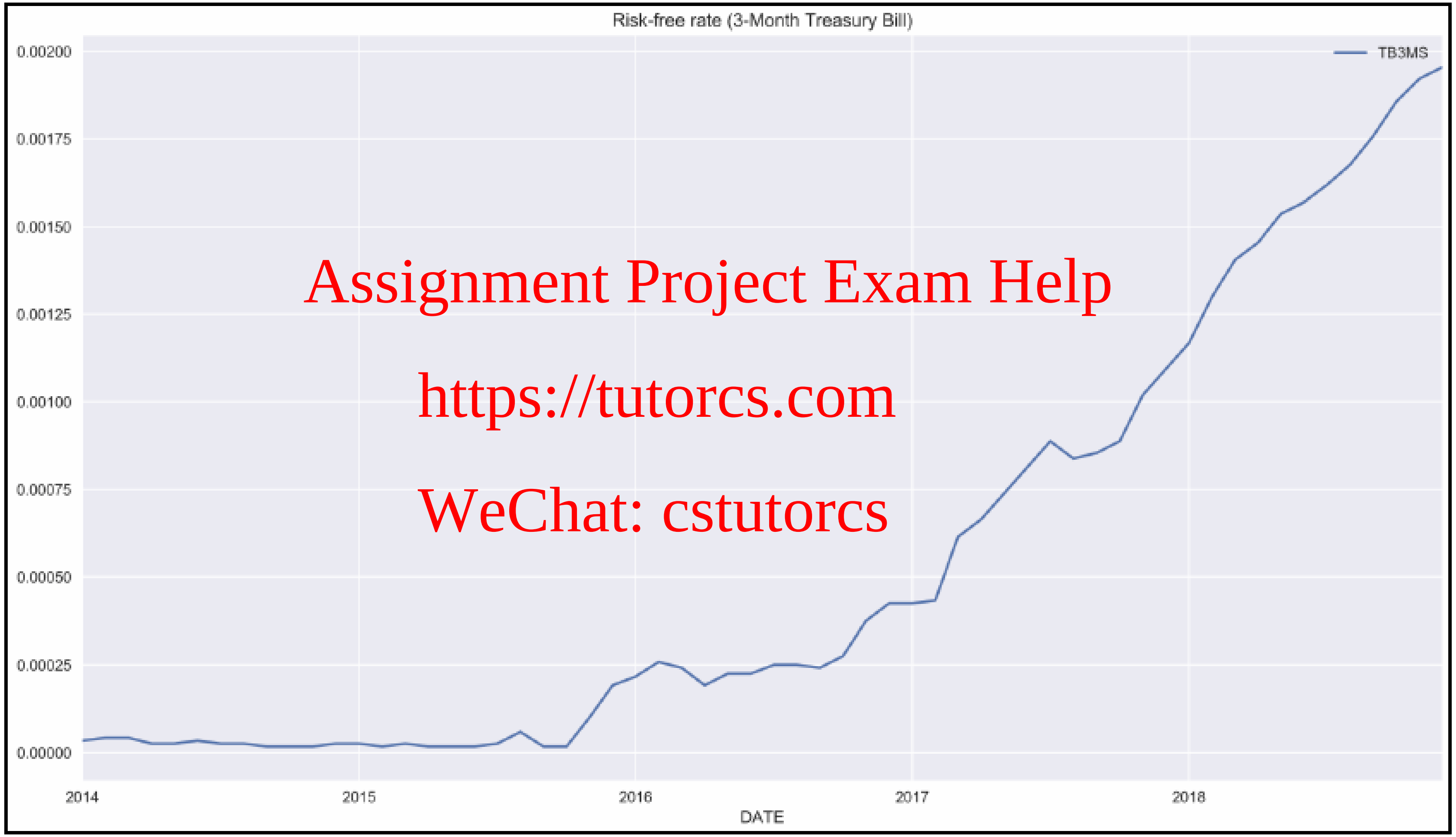
Monthly = (1 + daily)^(1/12) - 1

4. Plot the calculated risk-free rate:

```
rf.plot(title='Risk-free rate (3-Month Treasury Bill)')
```

https://fred.stlouisfed.org/series/TB3MS

Implementing CAPM in Python: Approach3



- We can conclude that the plots look very similar.

Implementing CAPM in Python: Resources

- Additional resources are available here:
- Sharpe, W. F. (1964). *Capital asset prices: A theory of market equilibrium under conditions of risk*. The journal of finance, 19(3), 425-442: https://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1964.tb02865.x
- Risk-free rate data on prof. Kenneth French's website: http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/F-F_Research_Data_Factors_CSV.zip
- Main Fama French data library: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

Implementing Fama French Three Factor Model

In their famous paper, Fama and French expanded the CAPM model by adding two additional factors explaining the excess returns of an asset or portfolio. The factors they considered are:

- **The market factor (MKT)**: It measures the excess return of the market, analogical to the one in the CAPM.
- **The size factor, SMB (Small Minus Big)**: It measures the excess return of stocks with a small market cap over those with a large market cap.
- **The value factor, HML (High Minus Low)**: It measures the excess return of value stocks over growth stocks. Value stocks have a high

The model can be represented as follows:

$$E(r_i) = r_f + \alpha + \beta_{mkt}(E(r_m) - r_f) + \beta_{smb}SMB + \beta_{hml}HML$$

Or in its simpler form:

$$E(r_i) - r_f = \alpha + \beta_{mkt}MKT + \beta_{smb}SMB + \beta_{hml}HML$$

- Here, $E(r_i)$ denotes the expected return on asset i,
- r_f is the risk-free rate (such as a government bond),
- α is the intercept.

Implementing Fama French Three Factor Model

- The reason for including the constant intercept is to make sure its value is equal to 0. This confirms that the three-factor model evaluates the relationship between the excess returns and the factors correctly.
- Due to the popularity of this approach, these factors became collectively known as the **Fama-French Factors**, or the **Three-Factor Model**. They have been widely accepted in both academia and the industry as stock market benchmarks and they are often used to evaluate investment performance.
- We estimate the three-factor model using 5 years (2014-2018) of monthly returns on Facebook.

Three Factor Model Description

Description of Fama/French Factors

Daily Returns: July 1, 1926 - January 31, 2022

Weekly Returns: July 2, 1926 - January 31, 2022

Monthly Returns: July 1926 - January 2022

Annual Returns: 1927 - 2021

Construction: The Fama/French factors are constructed using the 6 value-weight portfolios formed on size and book-to-market. (See the description of the 6 size/book-to-market portfolios.)

SMB (Small Minus Big) is the average return on the three small portfolios minus the average return on the three big portfolios,

SMB = 1/3 (Small Value + Small Neutral + Small Growth)
- 1/3 (Big Value + Big Neutral + Big Growth).

HML (High Minus Low) is the average return on the two value portfolios minus the average return on the two growth portfolios,

HML = 1/2 (Small Value + Big Value)
- 1/2 (Small Growth + Big Growth).

R_m-R_f, the excess return on the market, value-weight return of all CRSP firms incorporated in the US and listed on the NYSE, AMEX, or NASDAQ that have a CRSP share code of 10 or 11 at the beginning of month t, good shares and price data at the beginning of t, and good return data for t minus the one-month Treasury bill rate (from Ibbotson Associates).

See Fama and French, 1993, "Common Risk Factors in the Returns on Stocks and Bonds," *Journal of Financial Economics*, for a complete description of the factor returns.

- https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/f-f_factors.html

Three Factor Model: Code1

1. Import the libraries:

```
import pandas as pd
import yfinance as yf
import statsmodels.formula.api as smf
```

2. Download the necessary data from prof. French's website:

```
!wget
http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/F-F_Rese
arch_Data_Factors_CSV.zip

!unzip -a F-F_Research_Data_Factors_CSV.zip

!rm F-F_Research_Data_Factors_CSV.zip
```

3. Define the parameters:

```
RISKY_ASSET = 'FB'
START_DATE = '2013-12-31'
END_DATE = '2018-12-31'
```

Datasets are here: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

Step 2 Note:

- wget Requires a windows application installed so commenting out and this is not required.
- A method to download in a better way is covered later
- The file is supplied directly to you in a csv

U.S. Research Returns Data (Downloadable Files)

Changes in CRSP Data

Fama/French 3 Factors TXT CSV Details
Fama/French 3 Factors [Weekly] TXT CSV Details
Fama/French 3 Factors [Daily] TXT CSV Details

Fama/French 5 Factors (2x3) TXT CSV Details
Fama/French 5 Factors (2x3) [Daily] TXT CSV Details

datasets - List (297 elements)

Index	Type	Size	Value
0	str	25	F-F_Research_Data_Factors
1	str	32	F-F_Research_Data_Factors_weekly
2	str	31	F-F_Research_Data_Factors_daily
3	str	31	F-F_Research_Data_5_Factors_2x3
4	str	37	F-F_Research_Data_5_Factors_2x3_daily

```
#%% 1. Import the libraries:

from pandas_datareader.famafrench import get_available_datasets
import pandas_datareader.data as web

#%% 2. Print available datasets

datasets = get_available_datasets()
print(datasets[:5]) # first 5
```

Three Factor Model: Code2

1	This file was created by CMPT_ME_BEME_RETS using the 201911 CRSP database.				
2	The 1-mor	Inc.			
3					
4		Mkt-RF	SMB	HML	RF
5	192607	2.96	-2.3	-2.87	0.22

This is why skiprows =3. Inspect original data

4. Load the data from the source CSV file and keep only the monthly data:

```
factor_df = pd.read_csv('F-F_Research_Data_Factors.csv',
                        skiprows=3)

STR_TO_MATCH = ' Annual Factors: January-December '
indices = factor_df.iloc[:, 0] == STR_TO_MATCH
start_of_annual = factor_df[indices].index[0]

factor_df = factor_df[factor_df.index < start_of_annual]
```


5. Rename the columns of the DataFrame, set a datetime index, and filter by dates:

```
factor_df.columns = ['date', 'mkt', 'smb', 'hml', 'rf']
factor_df['date'] = pd.to_datetime(factor_df['date'],
                                   format='%Y%m') \
                      .dt.strftime("%Y-%m")
factor_df = factor_df.set_index('date')
factor_df = factor_df.loc[START_DATE:END_DATE]
```

Three Factor Model: Code3

6. Convert the values into numeric values and divide by 100:

```
factor_df = factor_df.apply(pd.to_numeric,
                             errors='coerce') \
                     .div(100)
```

errors : *{'ignore', 'raise', 'coerce'}, default 'raise'*

- If 'raise', then invalid parsing will raise an exception.
- If 'coerce', then invalid parsing will be set as NaN.
- If 'ignore', then invalid parsing will return the input.

The resulting data should be in the following form:

	mkt	smb	hml	rf
date				
2014-01	-0.0332	0.0085	-0.0209	0.0
2014-02	0.0465	0.0034	-0.0040	0.0
2014-03	0.0043	-0.0189	0.0509	0.0
2014-04	-0.0019	-0.0424	0.0114	0.0
2014-05	0.0206	-0.0186	-0.0027	0.0

https://pandas.pydata.org/docs/reference/api/pandas.to_numeric.html

7. Download the prices of the risky asset:

```
asset_df = yf.download(RISKY_ASSET,
                       start=START_DATE,
                       end=END_DATE,
                       adjusted=True)
```

Three Factor Model: Code4

8. Calculate the monthly returns on the risky asset:

```
y = asset_df['Adj Close'].resample('M') \
                          .last() \
                          .pct_change() \
                          .dropna()

y.index = y.index.strftime('%Y-%m')
y.name = 'rtn'
```


https://www.programiz.com/python-programming/datetime/strftime

9. Merge the datasets and calculate the excess returns:


```
ff_data = factor_df.join(y)
ff_data['excess_rtn'] = ff_data.rtn - ff_data.rf
```

10. Estimate the three-factor model:

```
ff_model = smf.ols(formula='excess_rtn ~ mkt + smb + hml',
                   data=ff_data).fit()
print(ff_model.summary())
```

Three Factor Model: Results

When interpreting the results of the three-factor model, we should pay attention to two issues:

1. Whether the intercept is positive and statistically significant
2. Which factors are statistically significant and if their direction matches past results (for example, from the literature) or our assumptions

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             excess_rtn   R-squared:                       0.217
Model:                            OLS   Adj. R-squared:                  0.175
Method:                 Least Squares   F-statistic:                     5.172
Date:                Mon, 02 Dec 2019   Prob (F-statistic):            0.00317
Time:                        20:39:06   Log-Likelihood:                 88.388
No. Observations:                  60   AIC:                            -168.8
Df Residuals:                      56   BIC:                            -160.4
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0105      0.008      1.374      0.175      -0.005       0.026
mkt            0.5138      0.237      2.166      0.035       0.039       0.989
smb           -0.1611      0.301     -0.535      0.595      -0.764       0.442
hml           -0.9599      0.312     -3.073      0.003      -1.586      -0.334
==============================================================================
Omnibus:                        0.008   Durbin-Watson:                   1.780
Prob(Omnibus):                  0.996   Jarque-Bera (JB):                0.151
Skew:                          -0.006   Prob(JB):                        0.927
Kurtosis:                       2.755   Cond. No.                         44.5
==============================================================================
```


- In our case, the intercept is positive, but **not** statistically significant at the 5% significance level.
- Of the risk factors, only the SMB factor is not significant. P value 0.595.
- However, a thorough literature study is required to formulate a hypothesis about the factors and their direction of influence.
- We can also look at the F-statistic that was presented in the regression summary, which tests the joint significance of the regression.
- The null hypothesis states that coefficients of all features (factors, in this case), except for the intercept, have values equal to 0.
- We can see that the corresponding p-value is 0.00317, which gives us reason to reject the null hypothesis at the 5% significance level.

Three Factor Model: Explanation

- In *Step 2*, we downloaded the data directly from prof. French's website.
- First, we downloaded the file using `wget` and then unzipped it using `unzip`. There are also ways to do this in Python only, but this seemed like a good place to introduce the possibility of mixing up bash script into the Notebooks. **This is not covered and commented out. File is provided directly to you for simplicity.**

- The link to the monthly data is always the same, and the file is updated every month. Implementation of above for frequently data downloading would have to be implemented. Easier method covered later.

- In *Steps 4* to *6*, we wrangled the raw data from the CSV file into a form that can be used for modelling. The file also contained annual factors located below the monthly ones, so we only kept the relevant rows (we also skipped the first three rows containing unnecessary information). We only kept the required dates, converted all the values into numeric ones, and then divided them by 100 (for example, 3.45 in the dataset is actually 3.45%).

Three Factor Model: Explanation

- In *Steps 7* and *8*, we downloaded the prices of Facebook's stock. We obtained the monthly returns by calculating the percentage change of the end-of-month prices. In *Step 6*, we also changed the formatting of the index to %Y-%m (for example, 2000-12) since the Fama- French factors contain dates in such a format.
- Then, we joined the two datasets in *Step 9*. Another noteworthy thing is that we must name the pandas Series by using y.name = 'asset'. This is a requirement for joining a DataFrame with a Series.
- Finally, in *Step 10*, we ran the regression using the formula notation – we do not need to manually add an intercept when doing so. One thing worth mentioning is that the coefficient by the mkt variable will not be equal to the CAPM's beta, as there are also other factors in the model and the factors' influence on the excess returns is distributed differently.

Three Factor Model: Datareader1

- We already showed how to download the factor-related data directly from prof. French's website.
- As an alternative, we can use the functionalities of pandas_datareader and avoid some manual pre-processing steps:

1. Import the libraries:

```
from pandas_datareader.famafrench import get_available_datasets
import pandas_datareader.data as web
```


2. Print the available datasets (only the first five):

```
get_available_datasets()[:5]
```

The preceding code results in the following output:

```
['F-F_Research_Data_Factors',
 'F-F_Research_Data_Factors_weekly',
 'F-F_Research_Data_Factors_daily',
 'F-F_Research_Data_5_Factors_2x3',
 'F-F_Research_Data_5_Factors_2x3_daily']
```

Three Factor Model: Datareader2

3. Download the selected dataset:

```
ff_dict = web.DataReader('F-F_Research_Data_Factors', 'famafrench',
                         start='2014-01-01')
```

The default behavior of `web.DataReader` downloads the last 5 years' worth of data. The resulting object is a dictionary and we can inspect its contents by running the following command:


```
ff_dict.keys()
```


This results in the following output:

```
dict_keys([0, 1, 'DESCR'])
```


4. Inspect the description of the dataset:

```
print(ff_dict['DESCR'])
```

The description is as follows:

```
F-F Research Data Factors
-------------------------
This file was created by CMPT_ME_BEME_RETS using the 201910 CRSP database. The 1-month TBill return is from Ibbotson
and Associates, Inc. Copyright 2019 Kenneth R. French
  0 : (70 rows x 4 cols)
  1 : Annual Factors: January-December (5 rows x 4 cols)
```

Three Factor Model: Datareader3

5. View the monthly dataset:

```
ff_dict[0].head()
```

The dataset looks like the one we manually downloaded from Prof. French's website:

	Mkt-RF	SMB	HML	RF
Date				
2014-01	-3.32	0.85	-2.09	0.0
2014-02	4.65	0.34	-0.40	0.0
2014-03	0.43	-1.89	5.09	0.0
2014-04	-0.19	-4.24	1.14	0.0
2014-05	2.06	-1.86	-0.27	0.0

The yearly values are stored under the key of 1 and can be accessed by using `ff_dict[1]`.

Three Factor Model: Resources

- For details on how all the factors were calculated, please refer to prof. French's website at: http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/f-f_factors.html
- Fama, E. F., and French, K. R. (1993). *Common risk factors in the returns on stocks and bonds*. Journal of financial economics, 33(1), 3-56: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.139.5892&rep=rep1&type=pdf

Rolling Fama French Three Factor Model: Introduction

- We learn how to estimate the three-factor model in a rolling fashion.
- What we mean by rolling is that we always consider an estimation window of a constant size (60 months, in this case) and roll it through the entire dataset, one period at a time.
- A potential reason for doing such an experiment is to test the stability of the results.
- In contrast to the previous codes, this time, we use portfolio returns instead of a single asset.
- To keep things simple, we assume that our allocation strategy is to have an equal share of the total portfolio's value in each of the following stocks: Amazon, Google, Apple, and Microsoft.
- For this experiment, we use stock prices from the years 2010-2018.

Rolling Three Factor Model: Explanation

- In *Steps 2* and *3*, we downloaded data using pandas_datareader and yfinance. This is very similar to what we did in the *Implementing the Fama-French three-factor model in Python* code, so at this point, we will not go into too much detail about this.
- In *Step 4*, we calculated the portfolio returns as a weighted average of the portfolio constituents. This is possible as we are working with simple returns – for more details, please refer to the *Converting prices to returns* code in Chapter 1, *Financial Data and Preprocessing*. Bear in mind that this simple approach assumes that, at the end of each month, we have exactly the same asset allocation (as indicated by the weights). This can be achieved with portfolio rebalancing, that is, adjusting the allocation after a specified period of time to always match the intended weights' distribution.
- Afterward, we merged the two datasets in *Step 5*.
- In *Step 6*, we defined a function for estimating the *n*-factor model using a rolling window. The main idea is to loop over the DataFrame we prepared in previous steps and for each month, estimate the Fama-French model using the last 5 years' worth of data (60 months). By appropriately slicing the input DataFrame, we made sure that we only estimate the model from the 60th month onwards, to make sure we always have a full window of observations.
- Finally, we applied the defined function to the prepared DataFrame and plotted the results.

Rolling Fama French Three Factor Model: Code1

1. Import the libraries:

```
import pandas as pd
import yfinance as yf
import statsmodels.formula.api as smf
import pandas_datareader.data as web
```

2. Define the parameters:


```
ASSETS = ['AMZN', 'GOOG', 'AAPL', 'MSFT']
WEIGHTS = [0.25, 0.25, 0.25, 0.25]
START_DATE = '2009-12-31'
END_DATE = '2018-12-31'
```


3. Download the factor-related data:

```
df_three_factor = web.DataReader('F-F_Research_Data_Factors',
                                 'famafrench', start=START_DATE)[0]
df_three_factor = df_three_factor.div(100)
df_three_factor.index = df_three_factor.index.format()
```

Rolling Fama French Three Factor Model: Code2

4. Download the prices of risky assets from Yahoo Finance:

```
asset_df = yf.download(ASSETS,
                       start=START_DATE,
                       end=END_DATE,
                       adjusted=True,
                       progress=False)
```

5. Calculate the monthly returns on the risky assets:

```
asset_df = asset_df['Adj Close'].resample('M') \
                    .last() \
                    .pct_change() \
                    .dropna()
asset_df.index = asset_df.index.strftime('%Y-%m')
```

6. Calculate the portfolio returns:

```
asset_df['portfolio_returns'] = np.matmul(asset_df[ASSETS].values,
                                          WEIGHTS)
```

https://numpy.org/doc/stable/reference/generated/numpy.matmul.html

Rolling Fama French Three Factor Model: Code3

7. Merge the datasets:

```
ff_data = asset_df.join(df_three_factor).drop(ASSETS, axis=1)
ff_data.columns = ['portf_rtn', 'mkt', 'smb', 'hml', 'rf']
ff_data['portf_ex_rtn'] = ff_data.portf_rtn - ff_data.rf
```

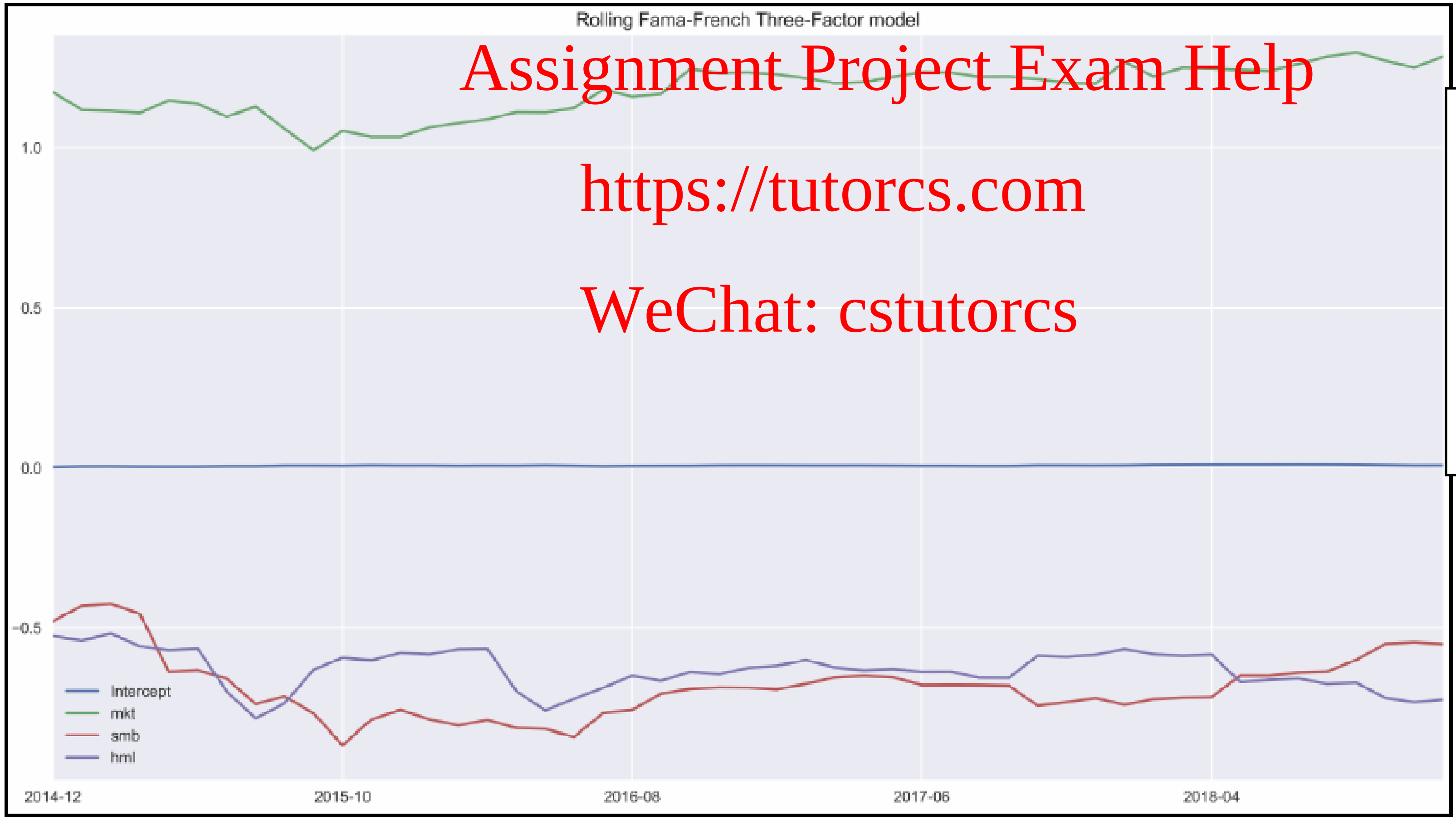
8. Define a function for the rolling *n*-factor model:


```
def rolling_factor_model(input_data, formula, window_size):
    coeffs = []

    for start_index in range(len(input_data) - window_size + 1):
        end_index = start_index + window_size

        ff_model = smf.ols(
            formula=formula,
            data=input_data[start_index:end_index]
        ).fit()
        coeffs.append(ff_model.params)

    coeffs_df = pd.DataFrame(
        coeffs,
        index=input_data.index[window_size - 1:]
    )

    return coeffs_df
```

Rolling Fama French Three Factor Model: Code4

9. Estimate the rolling three-factor model and plot the results:

```
MODEL_FORMULA = 'portf_ex_rtn ~ mkt + smb + hml'
results_df = rolling_factor_model(ff_data,
                                  MODEL_FORMULA,
                                  window_size=60)
results_df.plot(title = 'Rolling Fama-French Three-Factor model')
```



- The intercept is almost constant and very close to 0.
- There is some variability in the factors, but no sudden reversals or unexpected jumps.

Four Factor Model

We implement two extensions of the Fama-French three-factor model.

- **Carhart's Four-Factor model:** The underlying assumption of this extension is that, within a short period of time, a winner stock will remain a winner, while a loser will remain a loser. An example of a criterion for classifying winners and losers could be the last 12-month cumulative total returns. After identifying the two groups, we long the winners and short the losers within a certain holding period.

- The **momentum factor** (WML; Winners Minus Losers) measures the excess returns of the winner stocks over the loser stocks in the past 12 months (please refer to the *See also* section of this code for references on the calculations of the momentum factor).

The four-factor model can be expressed:

$$E(r_i) - r_f = \alpha + \beta_{mkt} MKT + \beta_{smb} SMB + \beta_{hml} HML + \beta_{wml} WML$$

Five Factor Model

- **Fama-French's Five-Factor model:** Fama and French expanded their three-factor model by adding two factors:
- **Robust Minus Weak** (**RMW**) measures the excess returns of companies with high profit margins (robust profitability) over those with lower profits (weak profitability).
- **Conservative Minus Aggressive** (**CMA**) measures the excess returns of firms with low investment policies (conservative) over those investing more (aggressive).

The five-factor model can be expressed as follows:

$$E(r_i) - r_f = \alpha + \beta_{mkt} MKT + \beta_{smb} SMB + \beta_{hml} HML + \beta_{rmw} RMW + \beta_{cma} CMA$$

Like in all factor models, if the exposure to the risk factors captures all possible variations in expected returns, the intercept (α) for all the assets/portfolios should be equal to zero.

Four & Five Factor Model Description

Description of Fama/French 5 Factors (2x3)

Monthly Returns: July 1963 - January 2022

Annual Returns: 1964 - 2021

SMB (Small Minus Big) is the average return on the nine small stock portfolios minus the average return on the nine big stock portfolios,

$SMB_{(B/M)}$ = *1/3 (Small Value + Small Neutral + Small Growth)*
- 1/3 (Big Value + Big Neutral + Big Growth).

$SMB_{(OP)}$ = *1/3 (Small Robust + Small Neutral + Small Weak)*
- 1/3 (Big Robust + Big Neutral + Big Weak).

$SMB_{(INV)}$ = *1/3 (Small Conservative + Small Neutral + Small Aggressive)*
- 1/3 (Big Conservative + Big Neutral + Big Aggressive).

SMB = $1/3\ (SMB_{(B/M)} + SMB_{(OP)} + SMB_{(INV)})$.

HML (High Minus Low) is the average return on the two value portfolios minus the average return on the two growth portfolios,

HML = *1/2 (Small Value + Big Value)*
- 1/2 (Small Growth + Big Growth).

RMW (Robust Minus Weak) is the average return on the two robust operating profitability portfolios minus the average return on the two weak operating profitability portfolios,

RMW = *1/2 (Small Robust + Big Robust)*
- 1/2 (Small Weak + Big Weak).

CMA (Conservative Minus Aggressive) is the average return on the two conservative investment portfolios minus the average return on the two aggressive investment portfolios,

CMA = *1/2 (Small Conservative + Big Conservative)*
- 1/2 (Small Aggressive + Big Aggressive).

Rm-Rf, the excess return on the market, value-weight return of all CRSP firms incorporated in the US and listed on the NYSE, AMEX, or NASDAQ that have a CRSP share code of 10 or 11 at the beginning of month t, good shares and price data at the beginning of t, and good return data for t minus the one-month Treasury bill rate (from Ibbotson Associates).

See Fama and French, 1993, "Common Risk Factors in the Returns on Stocks and Bonds," *Journal of Financial Economics*, and Fama and French, 2014, "A Five-Factor Asset Pricing Model" for a complete description of the factor returns.

- https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/f-f_5_factors_2x3.html#:~:text=The%20Fama%2FFrench%205%20factors,formed%20on%20size%20and%20investment.

Four and Five Factor Model: Code1

- In this code, we explain monthly returns on Amazon from 2014-2018 with the four- and five-factor models.

1. Import the libraries:

```
import pandas as pd
import yfinance as yf
import statsmodels.formula.api as smf
import pandas_datareader.data as web
```


2. Specify the risky asset and the time horizon:

```
RISKY_ASSET = 'AMZN'
START_DATE = '2013-12-31'
END_DATE = '2018-12-31'
```

Four and Five Factor Model: Code2

3. Download the risk factors from prof. French's website:

```
# three factors
df_three_factor = web.DataReader('F-F_Research_Data_Factors',
                                 'famafrench', start=START_DATE)[0]
df_three_factor.index = df_three_factor.index.format()

# momentum factor
df_mom = web.DataReader('F-F_Momentum_Factor', 'famafrench',
                        start=START_DATE)[0]
df_mom.index = df_mom.index.format()

# five factors
df_five_factor = web.DataReader('F-F_Research_Data_5_Factors_2x3',
                                'famafrench',
                                start=START_DATE)[0]
df_five_factor.index = df_five_factor.index.format()
```

datasets - List (297 elements)

Index	Type	Size	Value
0	str	25	F-F_Research_Data_Factors
1	str	32	F-F_Research_Data_Factors_weekly
2	str	31	F-F_Research_Data_Factors_daily
3	str	31	F-F_Research_Data_5_Factors_2x3
4	str	37	F-F_Research_Data_5_Factors_2x3_daily

datasets - List (297 elements)

Index	Type	Size	Value
69	str	32	6_Portfolios_ME_CFP_2x3_Wout_Div
70	str	22	6_Portfolios_ME_DP_2x3
71	str	31	6_Portfolios_ME_DP_2x3_Wout_Div
72	str	19	F-F_Momentum_Factor
73	str	25	F-F_Momentum_Factor_daily
74	str	26	6_Portfolios_ME_Prior_12_2

4. Download the data of the risky asset from Yahoo Finance:

```
asset_df = yf.download(RISKY_ASSET,
                       start=START_DATE,
                       end=END_DATE,
                       adjusted=True,
                       progress=False)
```

datasets - List (297 elements)

Index	Type	Size	Value
0	str	25	F-F_Research_Data_Factors
1	str	32	F-F_Research_Data_Factors_weekly
2	str	31	F-F_Research_Data_Factors_daily
3	str	31	F-F_Research_Data_5_Factors_2x3
4	str	37	F-F_Research_Data_5_Factors_2x3_daily

Four and Five Factor Model: Code3

5. Calculate the monthly returns:

```
y = asset_df['Adj Close'].resample('M') \
                         .last() \
                         .pct_change() \
                         .dropna()

y.index = y.index.strftime('%Y-%m')
y.name = 'return'
```

6. Merge the datasets for the four-factor model:


```
# join all datasets on the index
four_factor_data = df_three_factor.join(df_mom).join(y)

# rename columns
four_factor_data.columns = ['mkt', 'smb', 'hml', 'rf', 'mom',
                            'rtn']

# divide everything (except returns) by 100
four_factor_data.loc[:, four_factor_data.columns != 'rtn'] /= 100

# convert index to datetime
four_factor_data.index = pd.to_datetime(four_factor_data.index,
                                        format='%Y-%m')

# select period of interest
four_factor_data = four_factor_data.loc[START_DATE:END_DATE]

# calculate excess returns
four_factor_data['excess_rtn'] = four_factor_data.rtn -
four_factor_data.rf
```

Four and Five Factor Model: Code4

7. Merge the datasets for the five-factor model:

```
# join all datasets on the index
five_factor_data = df_five_factor.join(y)

# rename columns
five_factor_data.columns = ['mkt', 'smb', 'hml', 'rmw', 'cma',
                            'rf', 'rtn']

# divide everything (except returns) by 100
five_factor_data.loc[:, five_factor_data.columns != 'rtn'] /= 100

# convert index to datetime
five_factor_data.index = pd.to_datetime(five_factor_data.index,
                                        format='%Y-%m')

# select period of interest
five_factor_data = five_factor_data.loc[START_DATE:END_DATE]

# calculate excess returns
five_factor_data['excess_rtn'] = five_factor_data.rtn -
five_factor_data.rf
```

Four and Five Factor Model: Code5

8. Estimate the four-factor model:

```
four_factor_model = smf.ols(formula='excess_rtn ~ mkt + smb + hml +
                            mom', data=four_factor_data).fit()

print(four_factor_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             excess_rtn   R-squared:                       0.550
Model:                            OLS   Adj. R-squared:                  0.517
Method:                 Least Squares   F-statistic:                     16.78
Date:                Mon, 02 Dec 2019   Prob (F-statistic):           4.80e-09
Time:                        20:40:43   Log-Likelihood:                 86.248
No. Observations:                  60   AIC:                            -162.5
Df Residuals:                      55   BIC:                            -152.0
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0094      0.008      1.165      0.249      -0.007      0.025
mkt            1.7202      0.256      6.712      0.000       1.207      2.234
smb           -0.5547      0.315     -1.762      0.084      -1.186      0.076
hml           -1.0756      0.391     -2.748      0.008      -1.860     -0.291
mom            0.3251      0.294      1.104      0.274      -0.265      0.915
==============================================================================
Omnibus:                        5.028   Durbin-Watson:                   1.700
Prob(Omnibus):                  0.081   Jarque-Bera (JB):                4.598
Skew:                           0.678   Prob(JB):                        0.100
Kurtosis:                       3.034   Cond. No.                         57.6
==============================================================================
```


*Note results may be slightly different due to auto_adjusted stock price as per yfinance data and therefore returns and excess returns

Four and Five Factor Model: Code6

9. Estimate the five-factor model:

```
five_factor_model = smf.ols(
    formula='excess_rtn ~ mkt + smb + hml + rmw + cma',
    data=five_factor_data
).fit()

print(five_factor_model.summary())
```



```
                            OLS Regression Results
==============================================================================
Dep. Variable:             excess_rtn   R-squared:                       0.595
Model:                            OLS   Adj. R-squared:                  0.557
Method:                 Least Squares   F-statistic:                     15.85
Date:                Mon, 02 Dec 2019   Prob (F-statistic):           1.38e-09
Time:                        20:40:44   Log-Likelihood:                 89.415
No. Observations:                  60   AIC:                            -166.8
Df Residuals:                      54   BIC:                            -154.3
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0101      0.008      1.308      0.197      -0.005       0.025
mkt            1.5508      0.246      6.303      0.000       1.058       2.044
smb           -0.7826      0.342     -2.288      0.026      -1.468      -0.097
hml           -0.5938      0.423     -1.404      0.166      -1.442       0.254
rmw           -0.7025      0.571     -1.231      0.224      -1.847       0.442
cma           -1.4384      0.695     -2.071      0.043      -2.831      -0.046
==============================================================================
Omnibus:                        1.893   Durbin-Watson:                   1.734
Prob(Omnibus):                  0.388   Jarque-Bera (JB):                1.496
Skew:                           0.387   Prob(JB):                        0.473
Kurtosis:                       3.015   Cond. No.                         105.
==============================================================================
```


- According to the five-factor model, Amazon's excess returns are negatively exposed to most of the factors (all but the market factor). See coef's.
- Here, we present an example of the interpretation of the coefficients: an increase by 1 percentage point in the market factor results in an increase of 0.015 p.p.
- In other words, for a 1% return by the market factor, we can expect our portfolio (Amazon's stock) to return 1.5508 * 1% in excess of the risk-free rate.
- Similarly to the three-factor model, if the five-factor model fully explains the excess stock returns, the estimated intercept should be statistically indistinguishable from zero (which is the case for the considered problem).

Four and Five Factor Model: Explanation

- In *Step 2*, we defined the parameters (the ticker of the considered stock and timeframes) for later use.
- In *Step 3*, we downloaded the necessary datasets using pandas_datareader, which provides us with a convenient way of downloading the risk-factors-related data without manually downloading the CSV files. For more information on this process, please refer to the *There's more* section in the *Implementing the Fama-French three-factor model in Python*. One thing to note here is that we applied the format method to the index, as we had to remove PeriodIndex in order to join multiple datasets later.
- In *Steps 4* and *5*, we downloaded Amazon's stock prices and calculated the monthly returns using the previously explained methodology.
- In *Steps 6* and *7*, we joined all the datasets, renamed the columns, selected the period of interest, and calculated the excess returns. When using the join method without specifying what we want to join on (the on argument), the default is the index of the DataFrames. This way, we prepared all the necessary inputs for the four- and five-factor models. We also had to divide all the data we downloaded from prof. French's website by 100 to arrive at the correct scale.

Four and Five Factor Model: Explanation

- The SMB factor in the five-factor dataset is calculated differently compared to how it is in the three-factor dataset. For more details, please refer to the link in the *See also* section of this code
- In *Step 8* and *Step* 9, we estimated the models using the functional form of OLS regression from the statsmodels library. The functional form automatically adds the intercept to the regression equation.

See Also

- For details on the calculation of the factors, please refer to the following links:

- Momentum factor: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/det_mom_factor.html

- Five-factor model: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/f-f_5_factors_2x3.html

For papers introducing the four- and five-factor models, please refer to the following links:

- Carhart, M. M. (1997). *On persistence in mutual fund performance.* The Journal of Finance, 52(1), 57-82: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1997.tb03808.x
- Fama, E. F., & French, K. R. (2015). *A five-factor asset pricing model.* Journal of financial economics, 116(1), 1-22: https://tevgeniou.github.io/EquityRiskFactors/bibliography/FiveFactor.pdf