Group Project Representation

Group 10

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Table of Contents

Part 1 - Model Introduction

$$R_i - R_f = \alpha_i + \beta_i\left(R_{mkt} - R_f\right) + \beta_i^{SMB}(R_{\mathrm{B}} - R_S) + \beta_i^{HML}(R_H - R_L)$$

- Explained variable
  - Security premium
- Explanatory variables
  - Market premium
  - Scale difference between corporations
  - Liquidity difference between securities
- Goal
  - Figure out $\alpha_i, \beta_i, \beta_i^{SMB}, \beta_i^{HML}$ for each security with Multiple Linear Regression

Part 1 - Model Introduction

represented by

$R_{mkt} - R_f$ → S&P 500 return, 1-month treasury yield

$R_B - R_S$ → Market capitalization

$R_H - R_L$ → Average trading volume

Part 2 - Data Scraping and Processing

SlickCharts

Symbol of companies in S&P 500

YAHOO! FINANCE

Adjusted close price, volume

U.S. DEPARTMENT OF THE TREASURY

1-month treasury yield

Part 2 - Data Scraping and Processing

```
def SP500_cap_range_list():
    #web has already sorted by cap
    stocks_pool_url = 'https://www.slickcharts.com/sp500'
    stocks_pool_page = requests.get(stocks_pool_url)
    stocks_pool_DataFrame = pd.read_html(stocks_pool_page.text)[0]

    #construct the target list and list_for_test
    stocks_pool_DataFrame_test = stocks_pool_DataFrame[31:61]['Symbol']
    stocks_pool_DataFrame = stocks_pool_DataFrame[0:31]['Symbol']
    security_list_test = stocks_pool_DataFrame_test.tolist()
    security_list = stocks_pool_DataFrame.tolist()
    # security_list_test是回归方程等式右边， security_list是等式左边
    return security_list_test, security_list
```

- Extract the symbols of top 60 securities from S&P 500, equally divided into two lists
- Securities have been sorted by their market capitalization

Part 2 - Data Scraping and Processing

```
# Establish connection to Yahoo Finance
yahoo_finance_url = 'https://au.finance.yahoo.com/quote/%s/history?period1=%s&period2=%s&interval=1d&filter=history&frequency=1d'
# July 1, 2018 starting_date = '1530403200'
# June 30, 2019 ending_day = '1561939199'
timestamp = ['1530403200', '1536710400', '1543017600', '1549324800', '1555632000', '1561939199']
# timestamp = [2018.7.31, 2018.9.12, 2018.11.24, 2019.2.5, 2019.4.19, 2019.6.30]


def fetch_Yahoo_Finance(security, timestamp):
    All_DataFrame = None
    for starting in range(0, len(timestamp)-1):
        each_DataFrame = fetch_Yahoo_Finance_part(security, timestamp[starting], timestamp[starting+1])
        All_DataFrame = pd.concat([All_DataFrame, each_DataFrame], axis = 0)
    return All_DataFrame
```

- To get daily adjusted close price
- Yahoo Finance allows only 100 rows to be scraped one time, so we divide the time interval into 5 parts

Part 2 - Data Scraping and Processing

```
# Add security and daily stock return to the DataFrame
yahoo_hist_price_DataFrame['%s Ri'% security] = yahoo_hist_price_DataFrame['Adj. close**'].pct_change(periods=1)
```

- To calculate daily return of securities
- Use DataFrame.pct_change function on adjusted close price

Part 2 - Data Scraping and Processing

```
US_treasury_URL = 'https://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=yieldYear&year=%s'
year1 = '2018'
year2 = '2019'
US_treasury_hist_price_page1 = requests.get(US_treasury_URL % year1)
US_treasury_hist_price_page2 = requests.get(US_treasury_URL % year2)

#Read Table from the US Treasury webpage
US_treasury_hist_price_DataFrame1 = pd.read_html(US_treasury_hist_price_page1.text)[1]
US_treasury_hist_price_DataFrame2 = pd.read_html(US_treasury_hist_price_page2.text)[1]
```

- To get daily 1-month yield of US Treasury (risk-free rate)
- US Treasury website put the data of the same year in one page, so we scrape the data of 2018 and 2019 respectively

Part 2 - Data Scraping and Processing

```
def X1(Rm, Rf):
    X1_DataFrame = pd.merge(Rm, Rf, on = 'Date')
    X1_DataFrame['Rm_Rf'] = X1_DataFrame['^GSPC Rm'] - X1_DataFrame['1 mo']/100/365
    temp = X1_DataFrame.loc[:,['Date', 'Rm_Rf']]
    return temp
```

- To establish the first explanatory variable $R_{mkt} - R_f$, we use S&P 500 daily return rate minus 1-month US Treasury yield
- To unify the scales, 1-month US Treasury yield should be divided by 100 and 365

Part 2 - Data Scraping and Processing

```
def X2(security_list, timestamp):
    list1 = security_list[0:int(len(security_list)/2)]
    list2 = security_list[int(len(security_list)/2):int(len(security_list))]
    R1 = combine_Ri_DataFrame(list1, timestamp)
    R1.rename(columns = {'Ri Average':'R1 Ri Average'}, inplace = True)
    R2 = combine_Ri_DataFrame(list2, timestamp)
    R2.rename(columns = {'Ri Average':'R2 Ri Average'}, inplace = True)
    R = pd.merge(R1, R2, on = 'Date')
    R['Weight'] = R['R1 Ri Average'] - R['R2 Ri Average']
    temp = R.loc[:, ['Date', 'Weight']]
    return temp
```

- To establish the second explanatory variable $R_B - R_S$, we
  - sort the 30 securities by market capitalization (done by SlickCharts)
  - calculate the average return of the first 15 securities and the last 15 securities respectively for every single day
  - calculate the difference between two average returns each day

Part 2 - Data Scraping and Processing

```
def X3(security_list, timestamp):
    list1 = security_list[0:int(len(security_list)/2)]
    list2 = security_list[int(len(security_list)/2):int(len(security_list))]
    R1 = combine_Ri_DataFrame(list1, timestamp)
    R1.rename(columns = {'Ri Average':'R1 Ri Average'}, inplace = True)
    R2 = combine_Ri_DataFrame(list2, timestamp)
    R2.rename(columns = {'Ri Average':'R2 Ri Average'}, inplace = True)
    R = pd.merge(R1, R2, on = 'Date')
    R['Volume'] = R['R1 Ri Average'] - R['R2 Ri Average']
    temp = R.loc[:, ['Date', 'Volume']]
    return temp
```

- To establish the third explanatory variable $R_H - R_L$, we
  - sort the 30 securities by up-to-date volume (done by function fetch_vol)
  - calculate the average return of the first 15 securities and the last 15 securities respectively for every single day
  - calculate the difference between two average returns each day

Part 2 - Data Scraping and Processing

```
def Y(security_list, timestamp, Rf):
    Ri = combine_Ri_DataFrame(security_list, timestamp)
    Y_DataFrame = pd.merge(Ri, Rf, on = 'Date')
    Y_DataFrame['Ri_Rf'] = Y_DataFrame['Ri Average'] - Y_DataFrame['1 mo']/100/365
    temp = Y_DataFrame.loc[:,['Date', 'Ri_Rf']]
    return temp
```

- To establish the explained variable $R_i - R_f$, we
  - simply do subtraction on the data we've gotten
  - notice that risk-free rate (1-month US Treasury yield) should be divided by 100 and 365

Part 3 – Regression and Result

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Ri_Rf   R-squared:                       0.945
Model:                            OLS   Adj. R-squared:                  0.945
Method:                 Least Squares   F-statistic:                     4172.
Date:                Tue, 20 Aug 2019   Prob (F-statistic):          5.15e-155
Time:                        18:44:05   Log-Likelihood:                 1121.1
No. Observations:                 245   AIC:                            -2238.
Df Residuals:                     243   BIC:                            -2231.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0002      0.000     -1.119      0.264      -0.000       0.000
Rm_Rf          1.0550      0.016     64.592      0.000       1.023       1.087
==============================================================================
Omnibus:                        0.146   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.930   Jarque-Bera (JB):                0.026
Skew:                          -0.002   Prob(JB):                        0.987
Kurtosis:                       3.050   Cond. No.                         102.
==============================================================================
```

- Firstly, we only consider the factor $R_{mkt} - R_f$
- Market premium has a positive influence on security return
- R^2 is 0.945, which means $R_{mkt} - R_f$ explains the change in security return very well

Part 3 - Regression and Result

OLS Regression Results

Dep. Variable:	Ri_Rf	R-squared:	0.946
Model:	OLS	Adj. R-squared:	0.946
Method:	Least Squares	F-statistic:	2119.
Date:	Tue, 20 Aug 2019	Prob (F-statistic):	4.44e-154
Time:	18:44:07	Log-Likelihood:	1123.3
No. Observations:	245	AIC:	-2241.
Df Residuals:	242	BIC:	-2230.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0002	0.000	-0.992	0.322	-0.000	0.000
Rm_Rf	1.0740	0.019	58.009	0.000	1.038	1.110
Weight	-0.0563	0.026	-2.127	0.034	-0.109	-0.004

Omnibus:	0.606	Durbin-Watson:	1.994
Prob(Omnibus):	0.739	Jarque-Bera (JB):	0.348
Skew:	0.040	Prob(JB):	0.840
Kurtosis:	3.166	Cond. No.	180.

- Secondly, we consider both $R_m - R_f$ and $R_B - R_S$
- $R_S - R_B$ has a negative parameter, which means larger company usually brings lower return
- R^2 only increased by 0.001

Part 3 - Regression and Result

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Ri_Rf   R-squared:                       0.948
Model:                            OLS   Adj. R-squared:                  0.947
Method:                 Least Squares   F-statistic:                     1451.
Date:                Tue, 20 Aug 2019   Prob (F-statistic):          6.49e-154
Time:                        18:44:09   Log-Likelihood:                 1127.0
No. Observations:                 245   AIC:                            -2246.
Df Residuals:                     241   BIC:                            -2232.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0001      0.000     -0.911      0.363      -0.000       0.000
Rm_Rf          1.0770      0.018     58.808      0.000       1.041       1.113
Weight        -0.0732      0.027     -2.722      0.007      -0.126      -0.020
Volume         0.0934      0.035      2.693      0.008       0.025       0.162
==============================================================================
Omnibus:                        0.668   Durbin-Watson:                   2.014
Prob(Omnibus):                  0.716   Jarque-Bera (JB):                0.393
Skew:                           0.013   Prob(JB):                        0.821
Kurtosis:                       3.195   Cond. No.                         230.
==============================================================================
```

- Finally, we consider $R_m - R_f$, $R_S - R_B$ and $R_H - R_L$
- $R_H - R_L$ has a positive parameter, so security with higher liquidity has higher return
- R^2 increased by 0.002
- It seems that $R_{mkt} - R_f$ can explain $R_i - R_f$ pretty well, adding new factors doesn’ t influence the result much

Thank you