Arbitrage Pricing Theory (APT)

June 26, 2021

1 Import Packages

[78]:
```
!pip install pandas_datareader
```

```
Requirement already satisfied: pandas_datareader in
c:\users\rluck\anaconda3\lib\site-packages (0.9.0)
Requirement already satisfied: requests>=2.19.0 in
c:\users\rluck\anaconda3\lib\site-packages (from pandas_datareader) (2.25.1)
Requirement already satisfied: lxml in c:\users\rluck\anaconda3\lib\site-
packages (from pandas_datareader) (4.6.3)
Requirement already satisfied: pandas>=0.23 in
c:\users\rluck\anaconda3\lib\site-packages (from pandas_datareader) (1.2.4)
Requirement already satisfied: numpy>=1.16.5 in
c:\users\rluck\anaconda3\lib\site-packages (from
pandas>=0.23->pandas_datareader) (1.20.1)
Requirement already satisfied: pytz>=2017.3 in
c:\users\rluck\anaconda3\lib\site-packages (from
pandas>=0.23->pandas_datareader) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
c:\users\rluck\anaconda3\lib\site-packages (from
pandas>=0.23->pandas_datareader) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\rluck\anaconda3\lib\site-
packages (from python-dateutil>=2.7.3->pandas>=0.23->pandas_datareader) (1.15.0)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\rluck\anaconda3\lib\site-packages (from
requests>=2.19.0->pandas_datareader) (2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\rluck\anaconda3\lib\site-packages (from
requests>=2.19.0->pandas_datareader) (1.26.4)
Requirement already satisfied: chardet<5,>=3.0.2 in
c:\users\rluck\anaconda3\lib\site-packages (from
requests>=2.19.0->pandas_datareader) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in
c:\users\rluck\anaconda3\lib\site-packages (from
requests>=2.19.0->pandas_datareader) (2.10)
```

[44]:
```python
import pandas as pd
import pandas_datareader as data
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

2 Reading data from yahoo finance

[80]:
```python
#S&P500 =sp
sp= data.DataReader("^GSPC",
                       start='2016-1-1',
                       end='2021-5-25',
                       data_source='yahoo')
#Stock (Nike)= st
st= data.DataReader("NKE",
                       start='2016-1-1',
                       end='2021-5-25',
                       data_source='yahoo')
#Wilshire 5000 index
wls=data.DataReader("^W5000",
                       start='2016-1-1',
                       end='2021-5-25',
                       data_source='yahoo')
#Russell 1000 value index
rlv=data.DataReader("^RLV",
                       start='2016-1-1',
                       end='2021-5-25',
                       data_source='yahoo')
#Risk-free rate (Rf)
rf=data.DataReader("^IRX",
                       start='2016-1-1',
                       end='2021-5-25',
                       data_source='yahoo')
rlv
```


[80]:
```
                   High          Low         Open        Close  Volume  \
Date
2015-12-31   972.630005   964.460022   969.619995   964.609985       0
2016-01-04   963.090027   941.010010   963.090027   952.119995       0
2016-01-05   956.000000   947.729980   952.159973   954.630005       0
2016-01-06   953.419983   934.460022   953.419983   939.280029       0
2016-01-07   938.659973   915.200012   938.659973   917.770020       0
…                   …            …            …            …       …
2021-05-19  1564.410034  1547.459961  1564.410034  1547.459961       0
2021-05-20  1556.939941  1556.079956  1556.079956  1556.750000       0
```

```
2021-05-21  1571.930054  1565.180054  1565.180054  1571.930054       0
2021-05-24  1577.939941  1574.689941  1574.689941  1577.939941       0
2021-05-25  1582.310059  1579.160034  1579.160034  1582.310059       0

              Adj Close
Date
2015-12-31   964.609985
2016-01-04   952.119995
2016-01-05   954.630005
2016-01-06   939.280029
2016-01-07   917.770020
…                   …
2021-05-19  1547.459961
2021-05-20  1556.750000
2021-05-21  1571.930054
2021-05-24  1577.939941
2021-05-25  1582.310059

[1359 rows x 6 columns]
```


3 Computing Annualised Returns

$R = 365 * ln(p_t/p_{t-1})$

[46]:
```
#Stock returns
R =365*np.log(st['Adj Close']/st['Adj Close'].shift(1)).dropna()
#Market Index returns: S&P500
M =365*np.log(sp['Adj Close']/sp['Adj Close'].shift(1)).dropna()
#Size index: Wilshire 5000 index
S =365*np.log(wls['Adj Close']/wls['Adj Close'].shift(1)).dropna()
#Value index: Russell 1000 value index
V =365*np.log(rlv['Adj Close']/rlv['Adj Close'].shift(1)).dropna()
#Risk-free rate returns
Rf =(rf['Adj Close']/100).dropna()
```


[47]:
```
#Determining the mean returns of NIKE, S&P500, Wilshire 5000 index, Russell␣
↪1000 value index
name= ['r_n','r_m','r_s','r_v','r_f']
mean=[R.mean(),M.mean(), S.mean(),V.mean(),Rf.mean()]
ret= (name,mean)
ret
```

[47]:
```
(['r_n', 'r_m', 'r_s', 'r_v', 'r_f'],
 [0.2213318208464223,
  0.19281434539869813,
  0.19519557381753774,
  0.13302268067813539,
```

```
0.010222813645885903])
```

[49]:
```
# Determining the volatilites of NIKE stock, S&P500 index, Wilshire 5000 index␣
↪and Russell 1000 value index
name= ['s_n','s_m','s_s','s_v','s_f']
std=[R.var()**0.5,M.var()**0.5, S.var()**0.5,V.var()**0.5,Rf.var()**0.5]
std= (name,std)
std
```

[49]:
```
(['s_n', 's_m', 's_s', 's_v', 's_f'],
 [6.442245133103492,
  4.3773149805433516,
  4.44603301379896,
  4.457652010346641,
  0.008358817599314233])
```

4 Merging the columns into in one worksheet

[50]:
```
dt_M =pd.merge(M,Rf, on='Date', how='left').dropna()
dt =pd.merge(dt_M,R, on='Date', how='left').dropna()
dt_1= pd.merge(dt,S, on ='Date', how='left').dropna()
dta= pd.merge(dt_1,V, on='Date', how='left').dropna()
```


5 Renaming the Row Header

[51]:
```
dta_cols=['M','Rf','St','S','V']
dta.columns =dta_cols
dta
```

[51]:
```
                   M       Rf        St         S         V
Date
2016-01-04 -5.629045  0.00155 -5.768505 -5.673750 -4.756967
2016-01-05  0.733725  0.00205  5.067068  0.674867  0.960959
2016-01-06 -4.818789  0.00205 -5.245099 -5.043475 -5.916716
2016-01-07 -8.754823  0.00190 -9.867127 -9.041746 -8.455889
2016-01-08 -3.977601  0.00190 -6.026039 -4.063790 -4.517930
…                …        …         …         …         …
2021-05-19 -1.075929  0.00005 -7.068576 -1.279263 -4.202521
2021-05-20  3.832292  0.00003  0.850007  4.053303  2.184694
2021-05-21 -0.286229  0.00003 -1.674484 -0.180701  3.541917
2021-05-24  3.599812  0.00003  3.831742  3.592793  1.392827
2021-05-25 -0.776554  0.00010  0.707204 -1.099065  1.009473

[1340 rows x 5 columns]
```

6 OLS Regression to determine beta under APT (3-factor Model)

[69]:
```
#Factor Risk Premium
dta['Rp']= dta['M']-dta['Rf']
dta['Rs'] = dta['S']-dta['M']
dta['Rv']= dta['V']-dta['M']
#X & y Variables defined
X = dta [['Rp','Rs','Rv']]
X = sm.add_constant(X)
y= dta.St-dta.Rf
#OLS model
model = sm.OLS(y,X).fit()
predictions =model.predict(X)
Q = model.summary()
print(Q)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.433
Model:                            OLS   Adj. R-squared:                  0.431
Method:                 Least Squares   F-statistic:                     339.5
Date:                Sat, 26 Jun 2021   Prob (F-statistic):          8.08e-164
Time:                        23:08:31   Log-Likelihood:                -4016.8
No. Observations:                1340   AIC:                             8042.
Df Residuals:                    1336   BIC:                             8062.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0715      0.133      0.538      0.590      -0.189       0.332
Rp             0.9620      0.031     31.458      0.000       0.902       1.022
Rs             0.5828      0.298      1.953      0.051      -0.003       1.168
Rv             0.1596      0.086      1.858      0.063      -0.009       0.328
==============================================================================
Omnibus:                      456.086   Durbin-Watson:                   2.053
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8215.373
Skew:                           1.104   Prob(JB):                         0.00
Kurtosis:                      14.928   Cond. No.                         9.90
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```


[74]:
```
#Determining the risk-free rate and factor risk premiums of NIKE, S&P500,
↪Wilshire 5000 index and Russell 1000 value index based on average.
```

```
f_m = M.mean()-Rf.mean()
f_s = S.mean()-M.mean()
f_v = V.mean()-M.mean()
r_f= Rf.mean()
```

[76]:
```
#Determining Expected Returns from APT given factor risk premiums
ER = r_f + model.params['Rp']*f_m+model.params['Rs']*f_s+model.params['Rv']*f_v
ER
```

[76]: 0.1777202609792306

[77]:
```
#Determining Alpha (or excess returns)
Alpha = R.mean()-ER
Alpha
```

[77]: 0.0436115598671917

[ ]:
```

```