> coding finance

About

Code

# Factor Based Analysis in Python

7/2/2019 — Written by DD

In the last post we performed several steps in downloading and analyzing the fund performance data. We used the Fama French’s 3 factor model to analyze Fidelity Contrafund Fund (FCNTX). In this post we will repeat the same steps without all the explanation. We will try to make things clear using the comments in our code. So lets begin by loading all the modules we will need to run our analysis.

```
# Pandas to read csv file and other things
import pandas as pd
# Datareader to download price data from Yahoo Finance
import pandas_datareader as web
# Statsmodels to run our multiple regression model
import statsmodels.api as smf
# To download the Fama French data from the web
import urllib.request
# To unzip the ZipFile
import zipfile
```

As we did in the last post our eventual goal is to automate the process. So we will build several functions and in the end combine all those into one function that automates the regression analysis for us in one line of code.

## Get Fama French Data

```
def get_fama_french():
    # Web url
    ff_url = 
"https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/F-
F_Research_Data_Factors_CSV.zip"

    # Download the file and save it
    # We will name it fama_french.zip file


    urllib.request.urlretrieve(ff_url,'fama_french.zip')
    zip_file = zipfile.ZipFile('fama_french.zip', 'r')

    # Next we extact the file data

    zip_file.extractall()

    # Make sure you close the file after extraction

    zip_file.close()

    # Now open the CSV file

    ff_factors = pd.read_csv('F-F_Research_Data_Factors.csv',
skiprows = 3, index_col = 0)
    # We want to find out the row with NULL value
    # We will skip these rows

    ff_row = ff_factors.isnull().any(1).nonzero()[0][0]

    # Read the csv file again with skipped rows
    ff_factors = pd.read_csv('F-F_Research_Data_Factors.csv',
skiprows = 3, nrows = ff_row, index_col = 0)

    # Format the date index
    ff_factors.index = pd.to_datetime(ff_factors.index, format=
'%Y%m')

    # Format dates to end of month
    ff_factors.index = ff_factors.index + pd.offsets.MonthEnd()

    # Convert from percent to decimal
    ff_factors = ff_factors.apply(lambda x: x/ 100)
    return ff_factors
```

Lets see if the data is downloaded correctly.

```
ff_data = get_fama_french()
print(ff_data.tail())
```

```
##             Mkt-RF     SMB     HML      RF
## 2019-01-31  0.0841  0.0302 -0.0060  0.0021
## 2019-02-28  0.0340  0.0202 -0.0284  0.0018
## 2019-03-31  0.0110 -0.0315 -0.0407  0.0019
## 2019-04-30  0.0396 -0.0170  0.0198  0.0021
## 2019-05-31 -0.0694 -0.0125 -0.0238  0.0021
```

This looks good. Our data has been downloaded correctly.

## Get the Mutual Fund Data

We want our mutual fund price data to align with the fama french data, so we need to get the last date of FF data.

```
# Last day of FF data
ff_last = ff_data.index[ff_data.shape[0] - 1].date()
# Build the get_price function
# We need 3 arguments, ticker, start and end date
def get_price_data(ticker, start, end):
    price = web.get_data_yahoo(ticker, start, end)
    price = price['Adj Close'] # keep only the Adj Price col
    return price
```

Lets check if this function works.

```
# Get Price data for Fidelity's fund
price_data = get_price_data("FCNTX", "1980-01-01", "2019-06-30")
# Make sure to only have data upto last date of Fama French data
price_data = price_data.loc[:ff_last]
print(price_data.tail())
```

```
## Date
## 2019-05-24    12.60
## 2019-05-28    12.59
## 2019-05-29    12.47
## 2019-05-30    12.53

## 2019-05-31    12.36
## Name: Adj Close, dtype: float64
```

As we can see the last date matches with the FF data. Next we need to build the returns calculation function.

## Get Returns data

```
def get_return_data(price_data, period = "M"):

    # Resample the data to monthly price
    price = price_data.resample(period).last()

    # Calculate the percent change
    ret_data = price.pct_change()[1:]

    # convert from series to DataFrame
    ret_data = pd.DataFrame(ret_data)

    # Rename the Column
    ret_data.columns = ['portfolio']
    return ret_data

ret_data = get_return_data(price_data, "M")
print(ret_data.tail())
```

```
##             portfolio
## Date
## 2019-01-31   0.094460
## 2019-02-28   0.023960
## 2019-03-31   0.022077
## 2019-04-30   0.048800
## 2019-05-31  -0.057208
```

Next we need to merge this data with the Fama French data

## Merge the portfolio return data with Fama French data

In this step we will merge the data. We also need to rename the columns to something more appropriate. We will also calculate the portfolio excess returns.

```
# Merging the data
all_data = pd.merge(pd.DataFrame(ret_data),ff_data, how =
'inner', left_index= True, right_index= True)
# Rename the columns
all_data.rename(columns={"Mkt-RF":"mkt_excess"}, inplace=True)
# Calculate the excess returns
all_data['port_excess'] = all_data['portfolio'] - all_data['RF']
print(all_data.tail())
```

```
##             portfolio  mkt_excess     ...          RF
port_excess
## 2019-01-31   0.094460      0.0841     ...       0.0021
0.092360
## 2019-02-28   0.023960      0.0340     ...       0.0018
0.022160
## 2019-03-31   0.022077      0.0110     ...       0.0019
0.020177
## 2019-04-30   0.048800      0.0396     ...       0.0021
0.046700
## 2019-05-31  -0.057208     -0.0694     ...       0.0021
-0.059308
##
## [5 rows x 6 columns]
```

## Run the multiple regression model

Finally our data is ready to run the regression model.

```
model = smf.formula.ols(formula = "port_excess ~ mkt_excess + SMB
+ HML", data = all_data).fit()
print(model.params)
```

```
## Intercept     0.001178
## mkt_excess    0.893443
## SMB           0.032794
## HML          -0.103232
## dtype: float64
```

Success!!!

We can see that the results match the results we got from our last post. We have successfully replicated the process in Python. Now you know how to calculate the alpha and beta of any portfolio returns against the Fama & French's 3 factors model.

Finally lets combine all these functions into one function that automates our analysis in the future.

```
def run_reg_model(ticker,start,end):
    # Get FF data
    ff_data = get_fama_french()
    ff_last = ff_data.index[ff_data.shape[0] - 1].date()
    #Get the fund price data
    price_data = get_price_data(ticker,start,end)
    price_data = price_data.loc[:ff_last]
    ret_data = get_return_data(price_data, "M")
    all_data = pd.merge(pd.DataFrame(ret_data),ff_data, how =
'inner', left_index= True, right_index= True)
    all_data.rename(columns={"Mkt-RF":"mkt_excess"},
inplace=True)
    all_data['port_excess'] = all_data['portfolio'] -
all_data['RF']
    # Run the model
    model = smf.formula.ols(formula = "port_excess ~ mkt_excess +
SMB + HML", data = all_data).fit()
    return model.params
```

Finally here comes the one line code to download and analyze a new fund. Lets test this on the same Goldman Sachs's Strategic Growth Fund (GGRAX) test this on the same Goldman Sachs's Strategic Growth Fund (GGRAX).

```
ggrax_model = run_reg_model("GGRAX", start = "1999-05-01", end =
"2019-06-30")
print(ggrax_model)
```

```
## Intercept    -0.000192
## mkt_excess    1.024486
## SMB          -0.191246
## HML          -0.173825
## dtype: float64
```

Great!!! It works on a different fund as well. So we have just build a powerful function that can download any public fund data and find the regression results against the Fama French 3 factors model.

(The slight difference in alpha compared to the last post is due to missing May 1999 returns which was included in the code we did in R.)

READ OTHER POSTS

**← Webscraping with R** **Factor Based Analysis →**