1) Phạm Quỳnh Hương K194141724

2) Võ Thuỵ Uyên Nhi K194141736

BÀI TẬP NHÓM LẦN 2 - FAMA FRENCH

Ngày lựa chọn: ngày 7

Dữ liệu timeseries sử dụng tính RSMB; RHML và hồi quy mô hình từ 7/5/2007 đến 7/12/2020 (tổng cộng 164 tháng).

Clean Data:

1) Index name = "Year-Month".

2) Drop "ETF", "VT:FU…" and "#ERROR" columns.

3) Synchronize colunm names.

4) Drop "Code" row.

5) Vì các công ty đã Dead-Delist có BV được thể hiện sau ngày huỷ niêm yết là NaN. Tuy nhiên MV và Price vẫn được cập nhật theo giá ngày cuối cùng (lỗi hệ thống,…) nên nhóm sẽ đồng bộ dữ liệu của cả 3 sheets Price, MV, BV của các ngày sau khi công ty đã huỷ niêm yết bằng NaN.

6) Tương tự các BV của công ty niêm yết trên sàn có dữ liệu BV sớm hơn MV và Price (bị NaN vì công ty đó chưa có mở bán chứng khoán) nên nhóm sẽ tiến hành đồng bộ dữ liệu bằng NaN.

7) Drop các công ty niêm yết dưới 5 năm.

- **Change type of data sheets for fast:**

```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

# read data
df1 = pd.read_excel("/content/Data - FF.xlsx", sheet_name="Price")
df1.to_csv("Price.csv")
df2 = pd.read_excel("/content/Data - FF.xlsx", sheet_name="Bookvalue")
df2.to_csv("Bookvalue.csv")
df3 = pd.read_excel("/content/Data - FF.xlsx", sheet_name="Marketvalue")
df3.to_csv("Marketvalue.csv")
```

- **Load Price sheet:**

```python
# Price_day7
price = pd.read_csv('Price.csv', low_memory=False, index_col=1).iloc[:,1:]
# 449

# Drop 'ETF'
def ETF(df):
  l = []
  for j in df.columns:
    if "ETF" in j:
      l.append(j)
  df.drop(l, axis=1, inplace=True)
  return df.shape
ETF(price)
# (5689, 441)

# Drop 'Error' columns
for i in price.columns:
  if '#ERROR' in i:
    price.drop(i, axis=1, inplace=True)
# (5689, 438)

# Drop VT:FU...   # 14 kí tự
FU = []
for i in price.columns:
  if len(str(price[i][0])) >= 14:
    FU.append(i)
price.drop(FU, axis=1, inplace=True)
# (5689, 435)

# Get column name for price
price_col = []   # 435
x=0
for col in range(0, len(price.columns)):
  price_col.append(str(price.iloc[0,col])[:6])
  x+=1

# Drop 'Code' row and change index name
price.drop("Code", axis=0, inplace =True)
price.index.name = 'Date'
```

```python
# Deal with dead - delist companies   32 companies + Gia Lai Electricity
price.index = pd.to_datetime(price.index)
def Dead_Delist(df):
  for i in df.columns:
    if 'DEAD - DELIST' in i:
      delist_date = datetime.strptime(i[-8:], "%d/%m/%y")
      df.loc[(delist_date):,i] = np.nan
  return df.shape
Dead_Delist(price)
# (5688, 435)

# Change column names
price.columns = price_col

# Drop companies are established under 5 years
for i in price.columns:
  if price[i].count() < 1260:  # 252days/year x 5years (hàng1 còn tên viết tắt)
    price.drop(i, axis=1, inplace=True)


# Chọn mốc ngày 1/1/2007 - 31/12/2020
price = price.iloc[1826:5480]
# (3654, 337)

# Drop companies don't have BV
price.drop(['VT:C47', 'VT:SMB', 'VT:VCA', '67081W', '67083P'], axis=1, inplace=True)
# (3654, 332)

# Set day 7 of each month for Price
price.index = pd.to_datetime(price.index)
start = datetime(2007, 1, 1)
end = datetime(2020,12,31)
date =[]
timestamp = start
while timestamp < end:
    timestamp += timedelta(days = 1)
    if timestamp.date().strftime('%Y-%m-%d')[-2:] == '07':
        date.append(timestamp.date())
```

```python
# Drop Saturday and Sunday #nextfill
for i in range(len(date)):
    if date[i].strftime('%A') == "Saturday":
        date[i] -= timedelta(days=1)
    elif date[i].strftime('%A') == "Sunday":
        date[i] -= timedelta(days=2)
    else:
        pass
#print('List 7th day: \n', date)

# DataFrame with only the 7th day
price_month = price.loc[date]
price_month.index.names = ['Day 7 of each month']

#Change full date to date only month and year
price_month.index = pd.to_datetime(price_month.index)
price_month.index = price_month.index.strftime('%Y-%m')
print('Price only month-year: \n', price_month)    # 168 tháng, 332 cty

```

- **Result**:

```
Price only month-year:
                    VT:VCB    VT:VIC    VT:HPG  ... VT:TIC VT:VTF    67079K
Day 7 of each month                               ...
2007-01                NaN       NaN       NaN  ...    NaN    NaN  17575.75
2007-02                NaN       NaN       NaN  ...    NaN    NaN   21212.1
2007-03                NaN       NaN       NaN  ...    NaN    NaN  26666.64
2007-04                NaN       NaN       NaN  ...    NaN    NaN  21515.14
2007-05                NaN       NaN       NaN  ...    NaN    NaN  19696.95
...                    ...       ...       ...  ...    ...    ...       ...
2020-08              82900  77777.75  17925.92  ...    NaN    NaN       NaN
2020-09              83600  81333.31  18148.14  ...    NaN    NaN       NaN
2020-10              85000  82222.19  20592.59  ...    NaN    NaN       NaN
2020-11              85000  93422.19  22518.51  ...    NaN    NaN       NaN
2020-12              93100  94044.44  28296.29  ...    NaN    NaN       NaN

[168 rows x 332 columns]
```

- **Load Marketvalue sheet**:

```python
# Marketvalue_day7
mv = pd.read_csv('Marketvalue.csv', low_memory=False, index_col=1, header=1).iloc[:,1:]
mv.index.name = 'Date'

# Change column names
mv_col = []
for i in mv.columns:
  mv_col.append(str(i[:6]))
mv.columns = mv_col

# Synchronize price sheet and marketvalue sheet
mv = mv.loc[:,price.columns]
mv = mv.iloc[1826:5480,:]
# (3654, 340)

# Set day 7 of each month for Marketvalue
mv.index = pd.to_datetime(mv.index)
mv_month = pd.DataFrame(mv, index=date)
mv_month.index.names = ['Day 7 of each month']

#Change full date to date only month and year
mv_month.index = pd.to_datetime(price_month.index)
mv_month.index = mv_month.index.strftime('%Y-%m')

print('Marketvalue only month-year: \n', mv_month)
names = mv.columns
```

- **Result:**

```
Marketvalue only month-year:
                          VT:VCB       VT:VIC  ...     VT:VTF    67079K
Day 7 of each month                             ...
2007-01                      NaN          NaN  ...        NaN  619207.9
2007-02                      NaN          NaN  ...        NaN  747319.4
2007-03                      NaN          NaN  ...        NaN  939487.4
2007-04                      NaN          NaN  ...        NaN  757995.9
2007-05                      NaN          NaN  ...        NaN  693939.7
...                          ...          ...  ...        ...      ...
2020-08              307465700.0  295962600.0  ...  1379822.0  180980.3
2020-09              310061800.0  309492200.0  ...  1379822.0  180980.3
2020-10              315254300.0  312874800.0  ...  1379822.0  180980.3
2020-11              315254300.0  355493400.0  ...  1379822.0  180980.3
2020-12              345296400.0  357860900.0  ...  1379822.0  180980.3

[168 rows x 332 columns]
```

- **Load Bookvalue sheet:**

```python
# bv_day7
bv = pd.read_csv('Bookvalue.csv', low_memory=False, index_col=1, header=1).iloc[:,1:]
bv.index.name = 'Day 7 of each month'

# Change column names
bv_col = []
for i in bv.columns:
  bv_col.append(str(i[:6]))
bv.columns = bv_col

# Synchronize 3 sheets
bv = bv.loc[:,price.columns]
bv = bv.iloc[83:252,:]   # 31/1/2007-31/12/2020
# (3654, 332)

#Change full date to date only month and year
bv.index = pd.to_datetime(bv.index)
bv.index = bv.index.strftime('%Y-%m')
month_date = bv.index
bv = bv.iloc[:-1]
bv.index = month_date[1:]
print('Bookvalue only month-year: \n', bv)    # 168 tháng, 340 cty
```

- **Result:**

```
Bookvalue only month-year:
                         VT:VCB     VT:VIC     VT:HPG  ...  VT:TIC  VT:VTF    67079K
Day 7 of each month                                    ...
2007-01              11335.121    138.628     96.477  ...     NaN     NaN  8728.184
2007-02              13578.439    429.674    916.486  ...     NaN     NaN  9510.048
2007-03              13578.439    429.674    916.486  ...     NaN     NaN  9510.048
2007-04              13578.439    429.674    916.486  ...     NaN     NaN  9510.048
2007-05              13578.439    429.674    916.486  ...     NaN     NaN  9510.048
...                        ...        ...        ...  ...     ...     ...       ...
2020-08              25347.280  21611.872  13206.336  ...     NaN     NaN -8641.599
2020-09              25347.280  21611.872  13206.336  ...     NaN     NaN -8641.599
2020-10              25347.280  21611.872  13206.336  ...     NaN     NaN -8641.599
2020-11              25347.280  21611.872  13206.336  ...     NaN     NaN -8641.599
2020-12              25347.280  21611.872  13206.336  ...     NaN     NaN -8641.599

[168 rows x 332 columns]
```

- **Return Rate of return dataframe from Price sheet:**

```python
# Số liệu của BV chính xác, các cty đã huỷ niêm yết sẽ bị NaN
# Số liệu của Price, Marketvalue canh theo giá thị trường ngày cuối cùng giao dịch mà ffill nên k chính xác.

# Synchronize 'NaN' in 3 sheets
import math

price_month = price_month.astype('float')
mv_month = mv_month.astype('float')
bv = bv.astype('float')

for i in range(len(bv.columns)):
  for j in range(len(bv.index)):
    if math.isnan(price_month.iloc[j][i]) == True:
      bv.iloc[j][i] = np.nan
      mv_month.iloc[j][i] = np.nan

for i in range(len(bv.columns)):
  for j in range(len(bv.index)):
    if math.isnan(bv.iloc[j][i]) == True:
      price_month.iloc[j][i] = np.nan
      mv_month.iloc[j][i] = np.nan

print(price_month)
print(bv)
print(mv_month)
```

- **Return B/M dataframe by month:**

```
                      VT:VCB    VT:VIC    VT:HPG  ...  VT:TIC  VT:VTF    67079K
Day 7 of each month                                ...
2007-01                  NaN       NaN       NaN  ...     NaN     NaN  17575.75
2007-02                  NaN       NaN       NaN  ...     NaN     NaN  21212.10
2007-03                  NaN       NaN       NaN  ...     NaN     NaN  26666.64
2007-04                  NaN       NaN       NaN  ...     NaN     NaN  21515.14
2007-05                  NaN       NaN       NaN  ...     NaN     NaN  19696.95
...                      ...       ...       ...  ...     ...     ...       ...
2020-08              82900.0  77777.75  17925.92  ...     NaN     NaN       NaN
2020-09              83600.0  81333.31  18148.14  ...     NaN     NaN       NaN
2020-10              85000.0  82222.19  20592.59  ...     NaN     NaN       NaN
2020-11              85000.0  93422.19  22518.51  ...     NaN     NaN       NaN
2020-12              93100.0  94044.44  28296.29  ...     NaN     NaN       NaN

[168 rows x 332 columns]
                      VT:VCB     VT:VIC     VT:HPG  ...  VT:TIC  VT:VTF    67079K
Day 7 of each month                                  ...
2007-01                  NaN        NaN        NaN  ...     NaN     NaN  8728.184
2007-02                  NaN        NaN        NaN  ...     NaN     NaN  9510.048
2007-03                  NaN        NaN        NaN  ...     NaN     NaN  9510.048
2007-04                  NaN        NaN        NaN  ...     NaN     NaN  9510.048
2007-05                  NaN        NaN        NaN  ...     NaN     NaN  9510.048
...                      ...        ...        ...  ...     ...     ...       ...
2020-08             25347.28  21611.872  13206.336  ...     NaN     NaN       NaN
2020-09             25347.28  21611.872  13206.336  ...     NaN     NaN       NaN
2020-10             25347.28  21611.872  13206.336  ...     NaN     NaN       NaN
2020-11             25347.28  21611.872  13206.336  ...     NaN     NaN       NaN
2020-12             25347.28  21611.872  13206.336  ...     NaN     NaN       NaN

[168 rows x 332 columns]
                       VT:VCB       VT:VIC  ...  VT:VTF    67079K
Day 7 of each month                         ...
2007-01                   NaN          NaN  ...     NaN  619207.9
2007-02                   NaN          NaN  ...     NaN  747319.4
2007-03                   NaN          NaN  ...     NaN  939487.4
2007-04                   NaN          NaN  ...     NaN  757995.9
2007-05                   NaN          NaN  ...     NaN  693939.7
...                       ...          ...  ...     ...       ...
2020-08             307465700.0  295962600.0  ...     NaN       NaN
2020-09             310061800.0  309492200.0  ...     NaN       NaN
2020-10             315254300.0  312874800.0  ...     NaN       NaN
2020-11             315254300.0  355493400.0  ...     NaN       NaN
2020-12             345296400.0  357860900.0  ...     NaN       NaN

[168 rows x 332 columns]
```

- **Return list of RSMB and list of RHML:**

```
[19]  1  # Price_Rate of return (theo tháng, mốc là ngày 7 mỗi tháng)
      2  rate = price_month.pct_change(periods=1)
      3  rate = rate.iloc[1:,:]
      4  # 7/1/2007-7/12/2020
      5  # [168 rows x 332 columns]
      6
      7  mv_month = mv_month.iloc[1:,:]
      8  bv = bv.iloc[1:,:]
      9  # --> Đổi ra tháng 2 hết
     10  print(rate.to_string())
     11  print(rate.shape)
```

```
      1  # Tạo dataframe chứa B/M theo ngày 7 mỗi tháng của các cty
      2
      3  BM = (bv*1000)/mv_month
      4  # (167, 332)
      5  print(BM.to_string())
```

```
      1  # Split companies into 2 dataframes by day 7 of each month
      2  import statistics
      3  import math
      4
      5  r_smb = []
      6  r_hml = []
      7  ri = []
      8  # bv = bv.iloc[2:]
      9  # mv_month = mv_month.iloc[2:]
     10  # rate = rate.iloc[2:]
     11  # BM = BM.iloc[2:]
     12  for i in mv_month.index:
     13    # Mean of each row:
     14    value = []
     15    for j in mv_month.columns:
     16      if math.isnan(mv_month.loc[i,j]) == False:
     17        value.append(mv_month.loc[i,j])
     18    meann = statistics.mean(value)
     19    # Split big-small:
     20    big = []
     21    small = []
     22    for k in mv_month.columns:   # tháng
     23      if math.isnan(mv_month.loc[i,k]) == False and mv_month.loc[i,k] < meann:
     24        small.append(k)
     25      elif math.isnan(mv_month.loc[i,k]) == False and mv_month.loc[i,k] >= meann:
     26        big.append(k)
     27    # Create dataframe for big-small:
     28    df_big = pd.DataFrame(BM.loc[i,big].T, columns=['BM'], index=big)
     29    df_small = pd.DataFrame(BM.loc[i,small].T, columns=['BM'], index=small)
     30    rate_big = rate.loc[i,big].T
     31    rate_small = rate.loc[i,small].T
     32    rate_big.columns = ['Rate of return']
     33    rate_small.columns = ['Rate of return']
     34    df_big = pd.concat([df_big, rate_big], axis=1)
     35    df_small = pd.concat([df_small, rate_small], axis=1)
```

```
    # Sort by BM
    df_big = df_big.sort_values(by=['BM'])
    df_small = df_small.sort_values(by=['BM'])
    # 30% BL - 40% BN - 30% BH & 30% SL - 40% SN - 30% SH
    count_big = len(big)   # Số cty lớn
    count_small = len(small)    # Số cty nhỏ
    rbl = df_big.iloc[:int(0.3*count_big),1].mean()
    rbn = df_big.iloc[int(0.3*count_big):int(0.7*count_big),1].mean()
    rbh = df_big.iloc[int(0.7*count_big):,1].mean()
    rsl = df_small.iloc[:int(0.3*count_big),1].mean()
    rsn = df_small.iloc[int(0.3*count_big):int(0.7*count_big),1].mean()
    rsh = df_small.iloc[int(0.7*count_big):,1].mean()
    rsmb = (1/3)*(rsh+rsn+rsl) - (1/3)*(rbh+rbn+rbl)
    rhml = (1/2)*(rbh+rsh) - (1/2)*(rbl+rsl)
    r_smb.append(rsmb)
    r_hml.append(rhml)
    # For model
    m = (rbl+rbn+rbh+rsl+rsn+rsh)/6
    ri.append(m)

r_hml = r_hml[3:]
r_smb = r_smb[3:]
ri = ri[3:]
```

- **Build model**

```
# Create dataframe for X, y:

# VNINDEX
vnindex = pd.read_csv('/content/capm-data.csv', index_col=0, parse_dates=True, header=0).iloc[1741:5330,0]  # 2/4/2007-31/12/2020
# Series with only the 7th day
vnindex_day7 = vnindex.loc[date[3:]]
vnindex_day7.index.names = ['Year-Month']

# Change full date to date only month and year
vnindex_day7.index = pd.to_datetime(vnindex_day7.index)   # Day 7 of each month
vnindex_day7.index = vnindex_day7.index.strftime('%Y-%m')

# Rate of return by VNINDEX
vnindex_day7.astype('float')
rate_vnindex = vnindex_day7.pct_change()
rate_vnindex.dropna(axis=0, inplace=True)    # 7/5/2007-7/12/2020
rate_vnindex = pd.DataFrame(rate_vnindex, columns=['VNINDEX'])
print('VNINDEX rate only month-year: \n', rate_vnindex)   #

# BOND YIELD
bond = pd.read_excel('/content/Bond Yiled.xlsx', index_col=0, parse_dates=True, header=None, skiprows=6).iloc[66:231]
bond.index.names = ['Year-Month']
bond.columns = ['BOND']

# Change full date to date only month and year
bond.index = pd.to_datetime(bond.index)
bond.index = bond.index.strftime('%Y-%m')

bond = bond.fillna(8)
bond = bond/12

bond.astype('float')
rate_bond = bond.pct_change()
rate_bond.dropna(axis=0, inplace=True)    # 7/5/2007-7/12/2020
#rate_bond = rate_bond.iloc[3:]
print('BOND YIELD rate only month-year: \n', rate_bond)    #  Length: 167
```

```
# Create 'VNINDEX-rf' feature:
r_mkt = []
for i in range(164):
  r_mkt.append(float(rate_vnindex.iloc[i,0]) - float(rate_bond.iloc[i,0]))

# Create dataframe for X:
data_X = pd.DataFrame(np.array(r_smb).T, index=rate_vnindex.index, columns=['RSMB'])
r_hml = pd.DataFrame(np.array(r_hml), index=rate_vnindex.index, columns=['RHML'])
r_mkt = pd.DataFrame(np.array(r_mkt), index=rate_vnindex.index, columns=['RMKT'])
data_X = pd.concat([data_X,r_hml], axis=1)
data_X = pd.concat([data_X,r_mkt], axis=1)

# Create dataframe for y:
y = []
for i in range(164):
  y.append(ri[i] - rate_bond.iloc[i,0])
y = pd.DataFrame(y, index=rate_vnindex.index, columns=['y'])
data = pd.concat([data_X, y], axis=1)
print(data)   # [164 rows x 4 columns]
```

- **Result**:

```
                RSMB      RHML      RMKT         y
Year-Month
2007-05     0.037516 -0.059250 -0.048650 -0.124361
2007-06     0.091363  0.072864  0.083845  0.105265
2007-07    -0.019416  0.142391 -0.102588 -0.091213
2007-08     0.102667 -0.008374 -0.130083 -0.071545
2007-09     0.069605 -0.011427  0.035718  0.077717
...              ...       ...       ...       ...
2020-08    -0.003954  0.025378  0.005922  0.010936
2020-09     0.011915 -0.007674  0.100855  0.135130
2020-10    -0.007493 -0.003981  0.097041  0.137805
2020-11    -0.064430 -0.034654  0.079631  0.059001
2020-12    -0.028847  0.024607  0.085329  0.076357

[164 rows x 4 columns]
```

- **Run model**:

```
# Import library
import pandas as pd
import statsmodels.formula.api as smf

# y = ri - rf = price_month - rf
# X = smb, hml, rm - rf = VNINDEX - rf

model = 'y ~ RMKT + RSMB + RHML'
ff_model = smf.ols(model, data=data).fit()
print(ff_model.summary())
```

- **Result**:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.972
Model:                            OLS   Adj. R-squared:                  0.972
Method:                 Least Squares   F-statistic:                     1862.
Date:                Mon, 01 Nov 2021   Prob (F-statistic):          3.86e-124
Time:                        06:52:41   Log-Likelihood:                 378.83
No. Observations:                 164   AIC:                            -749.7
Df Residuals:                     160   BIC:                            -737.3
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0086      0.002      4.432      0.000       0.005       0.012
RMKT           1.0446      0.014     74.456      0.000       1.017       1.072
RSMB           0.4822      0.046     10.502      0.000       0.391       0.573
RHML           0.1475      0.046      3.236      0.001       0.057       0.238
==============================================================================
Omnibus:                       78.333   Durbin-Watson:                   1.541
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              665.243
Skew:                           1.497   Prob(JB):                    3.50e-145
Kurtosis:                      12.401   Cond. No.                         25.0
==============================================================================
```