Lecture 11 Fama-French 3 factor model

Douglas Chung

National Chengchi University
3 December 2021

1 Construction of Fama-French 3 factors

In this example, you will learn how to construct the Fama-French 3 factors from individual stocks on CRSP.

Size is defined as:

$$\text{ME}_{i,t} = |\text{PRC}_{i,t}| \times \text{SHROUT}_{i,t}$$

Book-to-market is defined as:

$$\text{BM}_{i,t} = \frac{\text{Book equity}_{i,t}}{\text{ME}_{i,t}}$$

Suggested structure of the code:

Import CRSP monthly returns.

Compute size $\text{ME}_{i,t}$ for each stock-month observation.

Identify NYSE size cutoffs (50-50 by median) for each month.

Sort stocks into small (S, bottom 50) and big stocks (B, top 50).

Combine CRSP monthly returns with COMPUSTAT book-to-market ratios.

Identify NYSE BM cutoffs (30-40-30 by percentiles) for each month.

Sort stocks into growth (G, bottom 30), neutral (N, middle 40), and value stocks (V, top 30).

Form 2 by 3 portfolios (ME $\times$ BM) through unconditional bivariate sorts.

Compute SMB returns over time with:

$$R_{\text{SMB}} = \frac{(R_{\text{SG}} + R_{\text{SN}} + R_{\text{SV}})}{3} - \frac{(R_{\text{BG}} + R_{\text{BN}} + R_{\text{BV}})}{3}$$

Compute VMG returns over time with:

$$R_{\text{WMG}} = \frac{(R_{\text{SV}} + R_{\text{BV}})}{2} - \frac{(R_{\text{SG}} + R_{\text{BG}})}{2}$$

Compare your SMB factor and HML factor with the ones constructed by Fama-French.

1.1 Import CRSP monthly returns

You have to read the "crsp_monthly_returns.csv" into Python. Please change the date column into datetime format in Python and select all common stocks with EXCHCD 10 and 11.

[1]:
```
import numpy as np
import pandas as pd

df = pd.read_csv("Stock_RET.csv")
df.date = pd.to_datetime(df.date, format='%Y%m%d', errors='ignore')+ pd.offsets.
 ↪MonthEnd(0)
df.EXCHCD = pd.to_numeric(df.EXCHCD, errors='coerce')
df.RET = pd.to_numeric(df.RET, errors='coerce')
df.ME = pd.to_numeric(df.ME, errors='coerce')
df = df.set_index(["date", "PERMNO"])
df.head()
```

[1]:

		EXCHCD	RET	ME
date	PERMNO			
2016-01-31	10001	2	0.116779	87.40160
2016-02-29	10001	2	-0.055288	82.56930
2016-03-31	10001	2	-0.006361	82.06748
2016-04-30	10001	2	-0.055698	76.73760
2016-05-31	10001	2	-0.021918	75.05568

1.2 Import book-to-market ratios from Compustat

[2]:
```
bm = pd.read_csv("Stock_BM.csv")
bm['date'] = pd.to_datetime(bm.yyyymm, format='%Y%m', errors='ignore') + pd.
 ↪offsets.MonthEnd(0)
bm = bm.drop('yyyymm', axis=1).set_index(["date", "PERMNO"])
bm.head()
```

[2]:

		BM
date	PERMNO	
2016-01-31	10001	0.925455
2016-02-29	10001	0.925455
2016-03-31	10001	0.925455
2016-04-30	10001	0.925455
2016-05-31	10001	0.925455

1.3 Combine Compustat data with CRSP

[3]:
```
df = df.join(bm, how='inner')
df = df.sort_index(level=1)
df.head()
```

[3]:
```
                   EXCHCD       RET        ME        BM
date       PERMNO
2016-01-31 10001        2  0.116779  87.40160  0.925455
2016-02-29 10001        2 -0.055288  82.56930  0.925455
2016-03-31 10001        2 -0.006361  82.06748  0.925455
2016-04-30 10001        2 -0.055698  76.73760  0.925455
2016-05-31 10001        2 -0.021918  75.05568  0.925455
```

1.4 Create a dataframe for NYSE stocks

[4]:
```
df_nyse = df[df.EXCHCD==1]
df = df.drop(['EXCHCD'], axis=1)
df_nyse = df_nyse.drop(['EXCHCD'], axis=1)
df_nyse.head()
```

[4]:
```
                        RET         ME        BM
date       PERMNO
2016-01-31 10051  -0.179939  476.07559  0.703169
2019-06-30 10051   0.026259  713.14600 -0.031398
2019-07-31 10051  -0.099217  642.39000 -0.031398
2019-08-31 10051   0.094493  704.28064 -0.031398
2019-09-30 10051   0.079449  760.23514 -0.031398
```

1.5 Extract market equities from NYSE

[5]:
```
me_nyse = df_nyse.unstack().xs('ME', axis=1)
me_nyse = me_nyse.loc[me_nyse.index.month==6]
me_nyse.head()
```

[5]:
```
PERMNO          10051        10104        10145        10147      10158  \
date
2016-06-30        NaN  168743.3389   88649.2168  53108.30239  125.78171
2017-06-30        NaN  207413.2355  101612.0244          NaN  211.17250
2018-06-30        NaN  175409.6893  107595.2665          NaN  354.79200
2019-06-30  713.14600  190041.6084  127056.4758          NaN  448.86729
2020-06-30  625.85208  169606.0541  101480.2023          NaN  819.78780

PERMNO           10220       10375        10516       10517       10606  ...  \
date                                                                     ...
2016-06-30  3713.39101  2163.79515  25201.39198  1592.49750  1622.89056  ...
2017-06-30  4828.54125  2729.42214  23518.61266  2748.67400  1761.95280  ...
2018-06-30  6212.05760  4142.38886  25627.40272  3056.49025  2181.08800  ...
2019-06-30  4957.31500  3413.42694  22854.73200  4156.04457  2580.34056  ...
2020-06-30  5393.77056         NaN  22164.29040  3067.76880  2226.60900  ...

PERMNO           93330       93372       93373       93374       93384  \
```

```
date
2016-06-30  2692.91304   578.97646  1147.62492  4407.10650  285.85974
2017-06-30  3446.77650   743.24390   531.86625  4939.58570  278.73907
2018-06-30  4388.77440  1089.64800   681.89460  5769.10740   80.47963
2019-06-30  5078.80295   907.01305   183.62799  6019.71630  358.78395
2020-06-30  4754.83140   802.73784    99.26686  5350.43642         NaN

PERMNO          93418       93420       93422       93423       93426
date
2016-06-30  248.67304  1685.40300  4226.75724  5405.28625  163.09326
2017-06-30        NaN  1911.33565  2429.09040  5401.26210  210.92160
2018-06-30        NaN  4115.48476  2914.30008  5852.25720  474.16635
2019-06-30        NaN  1829.14176  1721.06535  4185.93744  507.38744
2020-06-30        NaN         NaN   312.41478  1626.83727  308.33152

[5 rows x 1486 columns]
```

1.6 Find NYSE size cutoffs

[6]:
```
S_cutoff = pd.to_numeric(me_nyse.quantile(.5, axis=1, numeric_only=False))
S_cutoff.head()
```

[6]:
```
date
2016-06-30    2506.434960
2017-06-30    2886.386500
2018-06-30    3374.632515
2019-06-30    3176.082000
2020-06-30    2448.792270
Name: 0.5, dtype: float64
```

1.7 Extract market equities for all common stocks

[7]:
```
ME = df.unstack().xs('ME', axis=1)
ME = ME.loc[ME.index.month==6]
ME_port = pd.DataFrame(index=ME.index, columns=ME.columns)
ME.head()
```

[7]:
```
PERMNO          10001      10025        10026      10028       10032  \
date
2016-06-30   73.47888  411.47244  2220.568860    9.00017  1440.50400
2017-06-30  135.97100        NaN  2473.539217   44.93302  1765.45831
2018-06-30        NaN        NaN  2850.731590   19.38528  1926.77394
2019-06-30        NaN        NaN  3030.688500   35.00120  1721.15619
2020-06-30        NaN        NaN  2401.231440  164.23640  2059.36416

PERMNO         10044      10051        10104        10107    10116  ...  \
```

```
date                                                                  ...
2016-06-30  58.94105    0.00000  168743.3389   399535.360  21.8929  ...
2017-06-30  68.90184        NaN  207413.2355   531312.440      NaN  ...
2018-06-30  67.55320        NaN  175409.6893   757028.970      NaN  ...
2019-06-30  50.11440  713.14600  190041.6084  1023856.356      NaN  ...
2020-06-30  25.77625  625.85208  169606.0541  1540774.134      NaN  ...

PERMNO          93418       93420       93422       93423      93426  \
date
2016-06-30  248.67304  1685.40300  4226.75724  5405.28625  163.09326
2017-06-30        NaN  1911.33565  2429.09040  5401.26210  210.92160
2018-06-30        NaN  4115.48476  2914.30008  5852.25720  474.16635
2019-06-30        NaN  1829.14176  1721.06535  4185.93744  507.38744
2020-06-30        NaN   240.73800   312.41478  1626.83727  308.33152

PERMNO           93428        93429    93433     93434         93436
date
2016-06-30  1220.06808   5415.20670  32.5619  74.38596   31420.62420
2017-06-30  1330.15890  10307.81780      NaN  74.61700   60339.32776
2018-06-30         NaN  11666.97549      NaN  84.47075   58478.46391
2019-06-30         NaN  11576.50730      NaN  87.86976   40025.71007
2020-06-30         NaN  10234.68160      NaN  76.22724  200844.67120

[5 rows x 4245 columns]
```

1.8 Construct size sorted portfolios

[8]:
```
ME_port[ME.gt(S_cutoff, axis=0)] = 'Big'
ME_port[ME.le(S_cutoff, axis=0)] = 'Small'
ME_port.head()
```

[8]:
```
PERMNO      10001  10025  10026  10028  10032  10044  10051 10104 10107  \
date
2016-06-30  Small  Small  Small  Small  Small  Small  Small   Big   Big
2017-06-30  Small    NaN  Small  Small  Small  Small    NaN   Big   Big
2018-06-30    NaN    NaN  Small  Small  Small  Small    NaN   Big   Big
2019-06-30    NaN    NaN  Small  Small  Small  Small  Small   Big   Big
2020-06-30    NaN    NaN  Small  Small  Small  Small  Small   Big   Big

PERMNO      10116  ...  93418  93420  93422  93423  93426  93428 93429  93433  \
date               ...
2016-06-30  Small  ...  Small  Small    Big    Big  Small  Small   Big  Small
2017-06-30    NaN  ...    NaN  Small  Small    Big  Small  Small   Big    NaN
2018-06-30    NaN  ...    NaN    Big  Small    Big  Small    NaN   Big    NaN
2019-06-30    NaN  ...    NaN  Small  Small    Big  Small    NaN   Big    NaN
2020-06-30    NaN  ...    NaN  Small  Small  Small  Small    NaN   Big    NaN
```

```
PERMNO      93434 93436
date
2016-06-30  Small   Big
2017-06-30  Small   Big
2018-06-30  Small   Big
2019-06-30  Small   Big
2020-06-30  Small   Big

[5 rows x 4245 columns]
```

1.9 Extract book-to-market ratios from NYSE

[9]:
```
bm_nyse = df_nyse.unstack().xs('BM', axis=1)
bm_nyse = bm_nyse.loc[bm_nyse.index.month==6]
bm_nyse.head()
```

[9]:
```
PERMNO         10051     10104     10145     10147     10158     10220  \
date
2016-06-30       NaN  0.319560  0.236042  0.424589  1.646971  0.078547
2017-06-30       NaN  0.300814  0.224878       NaN  1.980440  0.038080
2018-06-30       NaN  0.277539  0.173783       NaN  1.425602  0.047463
2019-06-30 -0.031398  0.282542  0.203391       NaN  0.966414  0.062439
2020-06-30  0.009219  0.129745  0.159434       NaN  0.844302  0.068341

PERMNO         10375     10516     10517     10606  ...     93330     93372  \
date                                                ...
2016-06-30  0.846846  0.889146  1.029988  0.555812  ...  0.563636  0.541488
2017-06-30  0.646068  0.717326  0.769354  0.452688  ...  0.448359  0.370780
2018-06-30  0.680058  0.863985  0.691724  0.419240  ...  0.349775  0.346567
2019-06-30  0.728538  0.872806  0.703160  0.518956  ...  0.387541  0.406209
2020-06-30       NaN  0.790702  0.533931  0.369193  ...  0.341019  0.287443

PERMNO         93373     93374     93384     93418     93420     93422  \
date
2016-06-30  0.389621  0.740693  1.833255  0.708453  2.855260  2.291836
2017-06-30  0.732421  0.807990  0.606603       NaN  0.960353  0.981484
2018-06-30  0.803164  0.595677  0.423292       NaN  1.625491  1.871800
2019-06-30  1.881732  0.793348 -2.522636       NaN  2.291180  2.265606
2020-06-30  1.886844  0.715353       NaN       NaN       NaN  2.742920

PERMNO         93423     93426
date
2016-06-30  0.032601  1.255473
2017-06-30  0.002317  0.748481
2018-06-30 -0.071403  0.633590
2019-06-30 -0.099888  0.586222
```

```
2020-06-30 -0.123003  0.576182

[5 rows x 1486 columns]
```

1.10 Find NYSE book-to-market ratio cutoffs

[10]:
```
bm_nyse = bm_nyse.apply(pd.to_numeric)
L_cutoff = pd.to_numeric(bm_nyse.quantile(.3,axis=1,numeric_only=False))
H_cutoff = pd.to_numeric(bm_nyse.quantile(.7,axis=1,numeric_only=False))
H_cutoff.tail()
```

[10]:
```
date
2016-06-30    0.797621
2017-06-30    0.655417
2018-06-30    0.639257
2019-06-30    0.827990
2020-06-30    0.747450
Name: 0.7, dtype: float64
```

1.11 Extract book-to-market ratios for all common stocks

[11]:
```
BM = df.unstack().xs('BM', axis=1)
BM = BM.loc[BM.index.month==6]
BM_port = pd.DataFrame(index=BM.index, columns=BM.columns)
BM.head()
```

[11]:
```
PERMNO         10001     10025     10026     10028     10032     10044  \
date
2016-06-30  1.377216  0.282954  0.295408  0.953993  0.733165  0.317002
2017-06-30  0.789519       NaN  0.275043  0.384511  0.505577  0.309779
2018-06-30       NaN       NaN  0.262855  0.309202  0.502759  0.270789
2019-06-30       NaN       NaN  0.298914  0.679435  0.590952  0.385841
2020-06-30       NaN       NaN  0.257177  0.307973  0.387013  0.368183

PERMNO         10051     10104     10107     10116  ...     93418     93420  \
date                                                ...
2016-06-30  0.284643  0.319560  0.188588  0.267774  ...  0.708453  2.855260
2017-06-30       NaN  0.300814  0.152960       NaN  ...       NaN  0.960353
2018-06-30       NaN  0.277539  0.110646       NaN  ...       NaN  1.625491
2019-06-30 -0.031398  0.282542  0.106693       NaN  ...       NaN  2.291180
2020-06-30  0.009219  0.129745  0.085451       NaN  ...       NaN  3.727508

PERMNO         93422     93423     93426     93428     93429     93433  \
date
2016-06-30  2.291836  0.032601  1.255473  0.252853  0.049733 -0.837665
2017-06-30  0.981484  0.002317  0.748481  0.241305  0.052937       NaN
```

```
2018-06-30  1.871800 -0.071403  0.633590       NaN  0.256208       NaN
2019-06-30  2.265606 -0.099888  0.586222       NaN  0.335055       NaN
2020-06-30  2.742920 -0.123003  0.576182       NaN  0.282497       NaN

PERMNO         93434     93436
date
2016-06-30  0.837790  0.034522
2017-06-30  0.820741  0.137670
2018-06-30  0.645263  0.080625
2019-06-30  1.359421  0.085708
2020-06-30  1.430444  0.087374

[5 rows x 4245 columns]
```

1.12 Construct BM sorted portfolios

[12]:
```
BM_port[BM.gt(H_cutoff, axis=0)] = 'Value'
BM_port[(BM.le(H_cutoff, axis=0)) & (BM.ge(L_cutoff, axis=0))] = 'Neutral'
BM_port[BM.lt(L_cutoff, axis=0)] = 'Growth'
BM_port.head()
```

[12]:
```
PERMNO      10001    10025    10026    10028    10032    10044    10051  \
date
2016-06-30  Value   Growth   Growth    Value  Neutral  Neutral  Growth
2017-06-30  Value      NaN   Growth  Neutral  Neutral  Neutral     NaN
2018-06-30    NaN      NaN  Neutral  Neutral  Neutral  Neutral     NaN
2019-06-30    NaN      NaN   Growth  Neutral  Neutral  Neutral  Growth
2020-06-30    NaN      NaN   Growth  Neutral  Neutral  Neutral  Growth

PERMNO        10104    10107    10116  ...    93418  93420  93422    93423  \
date                                   ...
2016-06-30  Neutral   Growth   Growth  ...  Neutral  Value  Value   Growth
2017-06-30  Neutral   Growth      NaN  ...      NaN  Value  Value   Growth
2018-06-30  Neutral   Growth      NaN  ...      NaN  Value  Value   Growth
2019-06-30   Growth   Growth      NaN  ...      NaN  Value  Value   Growth
2020-06-30   Growth   Growth      NaN  ...      NaN  Value  Value   Growth

PERMNO        93426    93428    93429    93433  93434    93436
date
2016-06-30    Value   Growth   Growth   Growth  Value   Growth
2017-06-30    Value   Growth   Growth      NaN  Value   Growth
2018-06-30  Neutral      NaN  Neutral      NaN  Value   Growth
2019-06-30  Neutral      NaN  Neutral      NaN  Value   Growth
2020-06-30  Neutral      NaN  Neutral      NaN  Value   Growth

[5 rows x 4245 columns]
```

1.13 Extract returns for all common stocks

[13]:
```
RET = df.unstack().xs('RET', axis=1)
RET = RET.apply(pd.to_numeric)
RET.tail()
```

[13]:
```
PERMNO      10001  10025     10026     10028     10032     10044     10051  \
date
2020-08-31    NaN    NaN  0.104118 -0.082742  0.023960 -0.018072  0.131730
2020-09-30    NaN    NaN -0.036668  0.105670 -0.071513 -0.177914 -0.199393
2020-10-31    NaN    NaN  0.039727 -0.058275 -0.015432  0.000000  0.104298
2020-11-30    NaN    NaN  0.072435  0.143564  0.074346  0.593284  0.298798
2020-12-31    NaN    NaN  0.072598  0.125541  0.046848 -0.051522 -0.030851

PERMNO         10104     10107  10116  ...  93418     93420     93422  \
date                                   ...
2020-08-31  0.031921  0.102580    NaN  ...    NaN -0.132187 -0.115646
2020-09-30  0.043341 -0.067397    NaN  ...    NaN -0.495859 -0.305538
2020-10-31 -0.056114 -0.037370    NaN  ...    NaN       NaN -0.003102
2020-11-30  0.028694  0.060058    NaN  ...    NaN       NaN  0.788889
2020-12-31  0.120755  0.039006    NaN  ...    NaN       NaN  0.484472

PERMNO         93423     93426  93428     93429  93433     93434     93436
date
2020-08-31  0.249569 -0.021218    NaN  0.051425    NaN  0.008584  0.741452
2020-09-30 -0.065808  0.016459    NaN -0.044122    NaN  0.055319 -0.139087
2020-10-31  0.065025 -0.056477    NaN -0.073513    NaN -0.080645 -0.095499
2020-11-30  0.421369  0.224362    NaN  0.128552    NaN  0.144737  0.462736
2020-12-31  0.109665  0.076239    NaN  0.019711    NaN  0.122605  0.243252

[5 rows x 4245 columns]
```

1.14 Construct 2 by 3 ME-BM portfolios

[14]:
```
TMP = ME_port + BM_port
ME_BM_port = pd.DataFrame(index=RET.index)
ME_BM_port = ME_BM_port.join(TMP)
ME_BM_port = ME_BM_port.ffill(axis=0,limit=11).shift(1)
ME_BM_port.tail()
```

[14]:
```
           10001 10025        10026         10028         10032         10044  \
date
2020-08-31   NaN   NaN  SmallGrowth  SmallNeutral  SmallNeutral  SmallNeutral
2020-09-30   NaN   NaN  SmallGrowth  SmallNeutral  SmallNeutral  SmallNeutral
2020-10-31   NaN   NaN  SmallGrowth  SmallNeutral  SmallNeutral  SmallNeutral
2020-11-30   NaN   NaN  SmallGrowth  SmallNeutral  SmallNeutral  SmallNeutral
2020-12-31   NaN   NaN  SmallGrowth  SmallNeutral  SmallNeutral  SmallNeutral
```

```
                10051      10104      10107 10116  ... 93418       93420  \
date                                                ...
2020-08-31  SmallGrowth  BigGrowth  BigGrowth   NaN  ...   NaN  SmallValue
2020-09-30  SmallGrowth  BigGrowth  BigGrowth   NaN  ...   NaN  SmallValue
2020-10-31  SmallGrowth  BigGrowth  BigGrowth   NaN  ...   NaN  SmallValue
2020-11-30  SmallGrowth  BigGrowth  BigGrowth   NaN  ...   NaN  SmallValue
2020-12-31  SmallGrowth  BigGrowth  BigGrowth   NaN  ...   NaN  SmallValue

                93422        93423         93426 93428       93429 93433  \
date
2020-08-31  SmallValue  SmallGrowth  SmallNeutral   NaN  BigNeutral   NaN
2020-09-30  SmallValue  SmallGrowth  SmallNeutral   NaN  BigNeutral   NaN
2020-10-31  SmallValue  SmallGrowth  SmallNeutral   NaN  BigNeutral   NaN
2020-11-30  SmallValue  SmallGrowth  SmallNeutral   NaN  BigNeutral   NaN
2020-12-31  SmallValue  SmallGrowth  SmallNeutral   NaN  BigNeutral   NaN

                93434      93436
date
2020-08-31  SmallValue  BigGrowth
2020-09-30  SmallValue  BigGrowth
2020-10-31  SmallValue  BigGrowth
2020-11-30  SmallValue  BigGrowth
2020-12-31  SmallValue  BigGrowth

[5 rows x 4245 columns]
```

1.15 Compute returns for 2 by 3 ME-BM portfolios

[15]:
```
ME_lag = df.unstack().xs('ME', axis=1).shift(1)

unique_port = ['SmallGrowth', 'SmallNeutral', 'SmallValue', 'BigGrowth',␣
 ↪'BigNeutral', 'BigValue']
RET_port = pd.DataFrame(index=RET.index, columns=unique_port)
N_firm = pd.DataFrame(index=RET.index, columns=unique_port)

for p in unique_port:
    TMP_RET = RET[ME_BM_port==p].apply(pd.to_numeric)
    TMP_ME = ME_lag[ME_BM_port==p].apply(pd.to_numeric)
    TMP_PROD = TMP_RET*TMP_ME
    RET_port[p] = TMP_PROD.sum(axis=1)/TMP_ME.sum(axis=1)
    N_firm[p] = TMP_RET.count(axis=1)
RET_port = RET_port.dropna()*100
RET_port.tail()
```

[15]:
```
            SmallGrowth  SmallNeutral  SmallValue  BigGrowth  BigNeutral  \
date
2020-08-31     6.233505      4.766238    6.298374   9.534417    3.457365
2020-09-30    -1.913589     -4.292057   -5.627923  -4.058011   -3.003876
2020-10-31     0.820848      2.704116    5.279059  -3.454844   -0.370999
2020-11-30    23.829603     17.713952   20.053387  10.705744   13.794444
2020-12-31    12.861766      8.273868    8.021720   4.621791    4.229645

             BigValue
date
2020-08-31   4.058304
2020-09-30  -4.335978
2020-10-31   0.522830
2020-11-30  16.566307
2020-12-31   4.867752
```

1.16 Construct the SMB factor

[16]:
```
SMB = (RET_port.SmallValue + RET_port.SmallNeutral + RET_port.SmallGrowth)/3 -␣
 ↪(RET_port.BigValue + RET_port.BigNeutral + RET_port.BigGrowth)/3
df_SMB = pd.DataFrame(SMB, columns=['SMBrep'])
df_SMB.head()
```

[16]:
```
              SMBrep
date
2016-07-31  3.113683
2016-08-31  1.145219
2016-09-30  1.950472
2016-10-31 -3.995589
2016-11-30  6.370793
```

1.17 Construct the HML factor

[17]:
```
HML = (RET_port.SmallValue + RET_port.BigValue)/2 - (RET_port.SmallGrowth +␣
 ↪RET_port.BigGrowth)/2
df_HML = pd.DataFrame(HML, columns=['HMLrep'])
df_HML.head()
```

[17]:
```
              HMLrep
date
2016-07-31 -1.905421
2016-08-31  1.873755
2016-09-30 -0.872897
2016-10-31  3.915732
2016-11-30  6.932787
```

[18]:
```
df_REP = pd.concat([df_SMB, df_HML], axis=1)
df_REP.head()
```

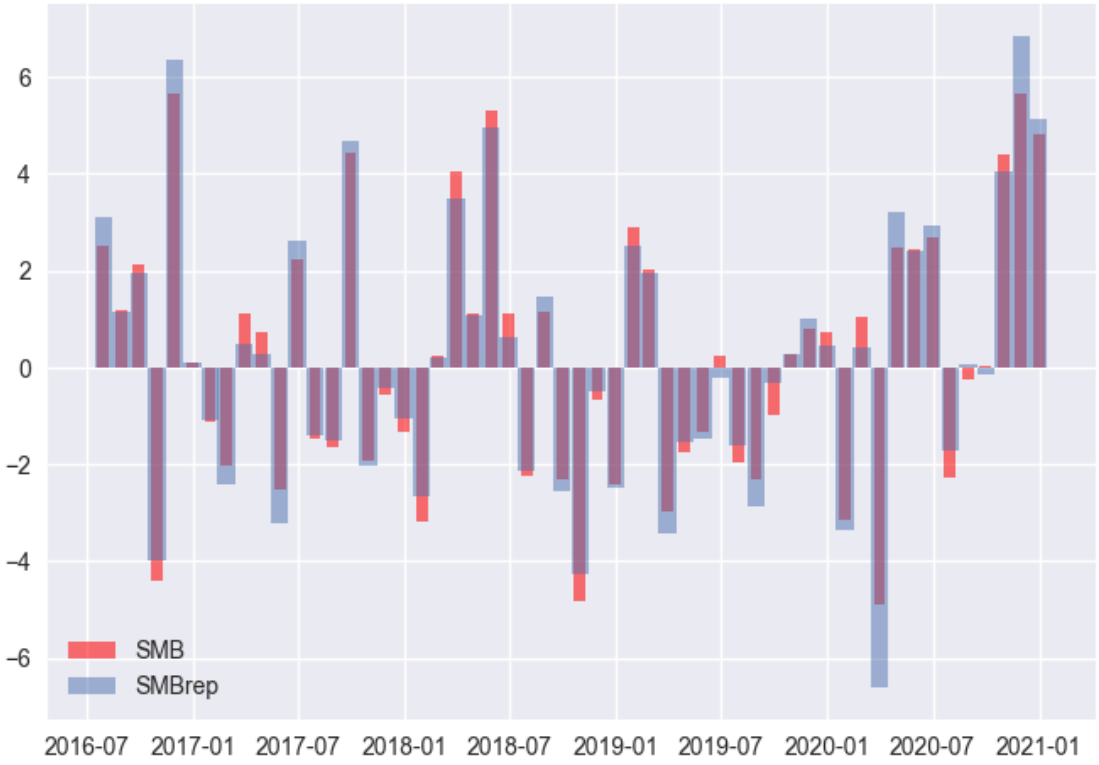
[18]:

date	SMBrep	HMLrep
2016-07-31	3.113683	-1.905421
2016-08-31	1.145219	1.873755
2016-09-30	1.950472	-0.872897
2016-10-31	-3.995589	3.915732
2016-11-30	6.370793	6.932787

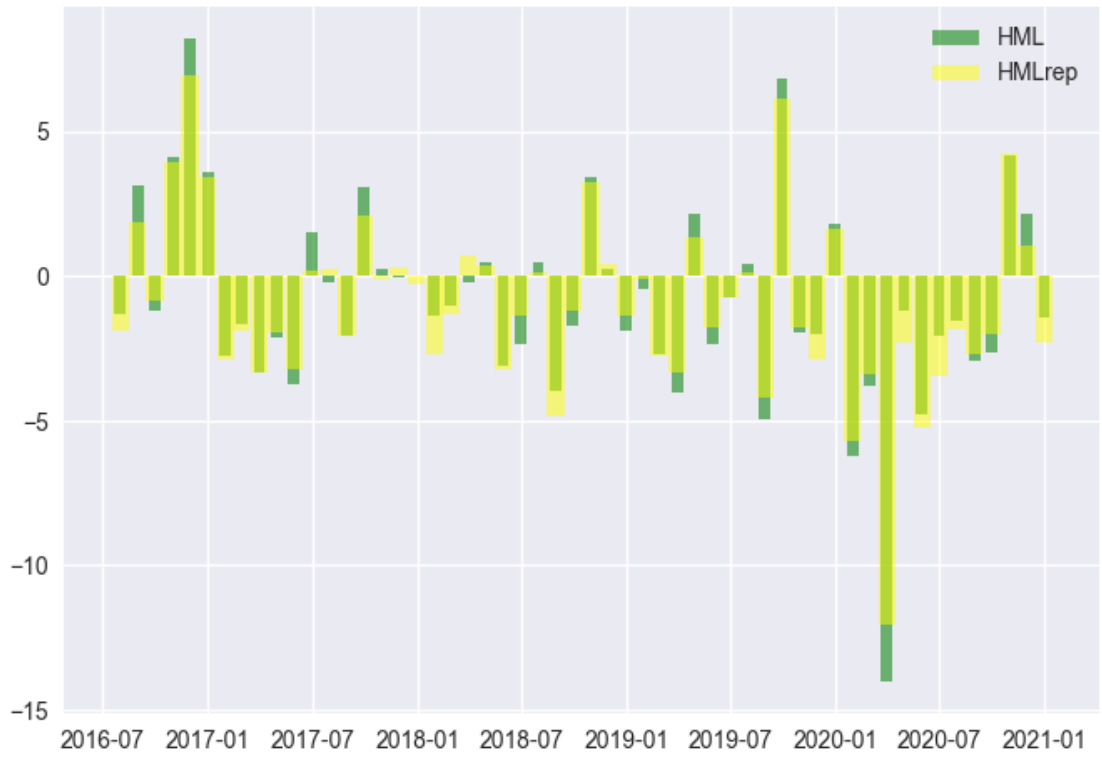
1.18 Compare our replications with Fama-French

[19]:
```
import pandas_datareader as pdr
factor = pdr.get_data_famafrench('F-F_Research_Data_Factors', start=df_REP.index.
 ↪astype(str)[0], end=df_REP.index.astype(str)[-1])
FF = factor[0]
FF.index = df_REP.index
```

[20]:
```
import matplotlib.pyplot as plt
plt.style.use('seaborn')
plt.bar(FF.index, FF.SMB, width=20, alpha=0.55, color='red')
plt.bar(df_REP.index, df_REP.SMBrep, width=30, alpha=0.5)
plt.legend(['SMB','SMBrep'])
plt.show()
```



[21]:
```
plt.bar(FF.index, FF.HML, width=20, alpha=0.55, color='green')
plt.bar(df_REP.index, df_REP.HMLrep, width=30, alpha=0.5, color='yellow')
plt.legend(['HML','HMLrep'])
plt.show()
```



[22]:
```
FF.join(df_REP).drop(['Mkt-RF','RF'],axis=1).corr()
```

[22]:

	SMB	HML	SMBrep	HMLrep
SMB	1.000000	0.280485	0.986374	0.222325
HML	0.280485	1.000000	0.369870	0.982383
SMBrep	0.986374	0.369870	1.000000	0.296754
HMLrep	0.222325	0.982383	0.296754	1.000000

[23]:
```
mebm6 = pdr.get_data_famafrench('6_Portfolios_2x3', start=df_REP.index.
 ↪astype(str)[0], end=df_REP.index.astype(str)[-1])
print(mebm6['DESCR'])
```

```
6 Portfolios 2x3
----------------

This file was created by CMPT_ME_BEME_OP_INV_RETS using the 202107 CRSP
database. It contains value- and equal-weighted returns for portfolios formed on
ME and BEME. The portfolios are constructed at the end of June. BEME is book
value at the last fiscal year end of the prior calendar year divided by ME at
the end of December of the prior year. Annual returns are from January to
December. Missing data are indicated by -99.99 or -999. The break points include
utilities and include financials. The portfolios include utilities and include
financials. Copyright 2021 Kenneth R. French
```

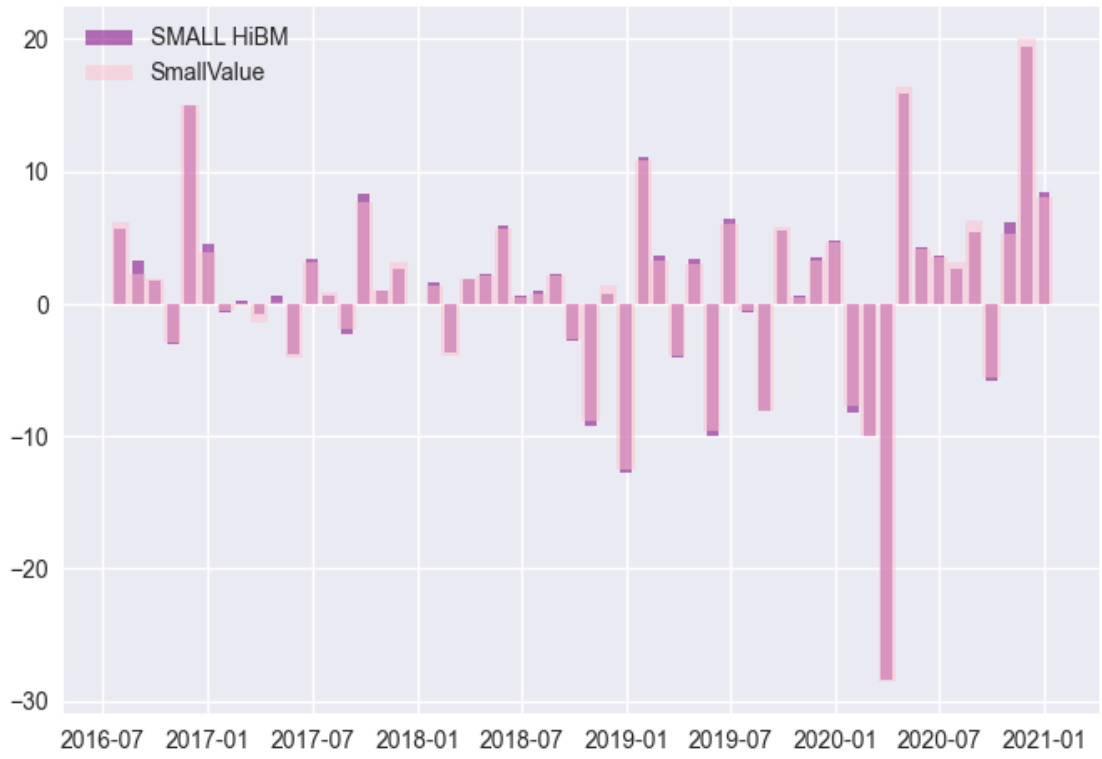
```
  0 : Average Value Weighted Returns -- Monthly (54 rows x 6 cols)
  1 : Average Equal Weighted Returns -- Monthly (54 rows x 6 cols)
  2 : Average Value Weighted Returns -- Annual (5 rows x 6 cols)
  3 : Average Equal Weighted Returns -- Annual (5 rows x 6 cols)
  4 : Number of Firms in Portfolios (54 rows x 6 cols)
  5 : Average Market Cap (54 rows x 6 cols)
  6 : For portfolios formed in June of year t   Value Weight Average of BE/ME
Calculated for June of t to June of t+1 as:    Sum[ME(Mth) * BE(Fiscal Year t-1)
/ ME(Dec t-1)] / Sum[ME(Mth)]   Where Mth is a month from June of t to June of
t+1   and BE(Fiscal Year t-1) is adjusted for net stock issuance to Dec t-1 (54
rows x 6 cols)
  7 : For portfolios formed in June of year t   Value Weight Average of
BE_FYt-1/ME_June t Calculated for June of t to June of t+1 as:    Sum[ME(Mth) *
BE(Fiscal Year t-1) / ME(Jun t)] / Sum[ME(Mth)]   Where Mth is a month from June
of t to June of t+1   and BE(Fiscal Year t-1) is adjusted for net stock issuance
to Jun t (54 rows x 6 cols)
  8 : For portfolios formed in June of year t   Value Weight Average of OP
Calculated as:    Sum[ME(Mth) * OP(fiscal year t-1) / BE(fiscal year t-1)] /
Sum[ME(Mth)]    Where Mth is a month from June of t to June of t+1 (54 rows x 6
cols)
  9 : For portfolios formed in June of year t   Value Weight Average of
investment (rate of growth of assets) Calculated as:    Sum[ME(Mth) *
Log(ASSET(t-1) / ASSET(t-2) / Sum[ME(Mth)]    Where Mth is a month from June of
t to June of t+1 (54 rows x 6 cols)
```

[24]:
```
mebm6[0].head()
```

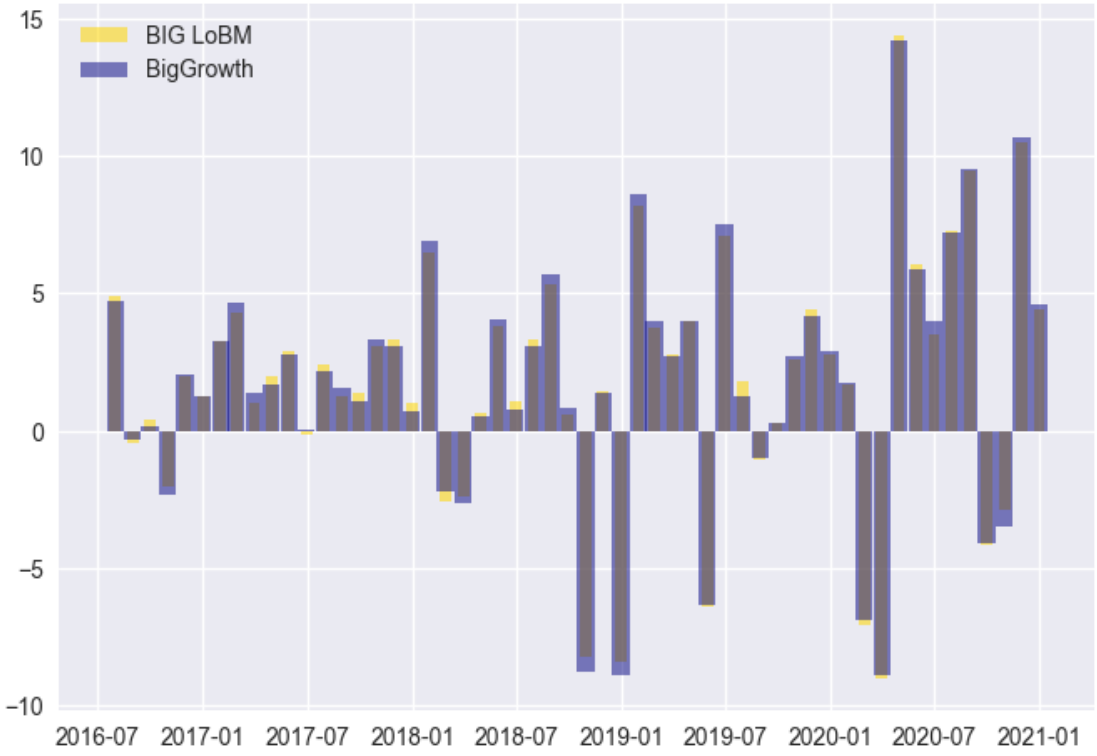
[24]:

Date	SMALL LoBM	ME1 BM2	SMALL HiBM	BIG LoBM	ME2 BM2	BIG HiBM
2016-07	7.0391	5.6291	5.6866	4.9023	2.2617	3.6699
2016-08	1.3009	2.7039	3.2145	-0.4301	0.2545	3.8763
2016-09	3.2742	1.2982	1.7466	0.4419	-0.0827	-0.4423
2016-10	-8.1473	-5.1820	-3.0105	-2.0245	-2.1588	1.0863
2016-11	9.1400	12.4935	14.9845	1.9783	5.0788	12.5522

[25]:
```
plt.bar(RET_port.index, mebm6[0]['SMALL HiBM'], width=20, alpha=0.55,␣
↪color='purple')
plt.bar(RET_port.index, RET_port.SmallValue, width=30, alpha=0.5, color='pink')
plt.legend(['SMALL HiBM','SmallValue'])
plt.show()
```



[26]:
```
plt.bar(RET_port.index, mebm6[0]['BIG LoBM'], width=20, alpha=0.55, color='gold')
plt.bar(RET_port.index, RET_port.BigGrowth, width=30, alpha=0.5, color='navy')
plt.legend(['BIG LoBM','BigGrowth'])
plt.show()
```



References

[1] Eugene Fama and Kenneth French (1993) "Common risk factors in the returns on stocks and bonds", *Journal of Financial Economics*, 33(1): 3–56.