Fama French Model

by Kashish Gupta (Masters in Quantitative Economics, UCLA)

This notebook contains estimation of fama-french factors for META stock

Data source: Yahoo Finance

In [86]:
```
import pandas as pd
import statsmodels.api as sm
import datetime as dt
import matplotlib.pyplot as plt
import numpy as np
from statsmodels.tsa.stattools import adfuller
from scipy import stats
from math import sqrt
```

In [2]:
```
pip install yfinance
```

```
Collecting yfinance
  Downloading yfinance-0.2.3-py2.py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 2.7 MB/s eta 0:00:01
Collecting pytz>=2022.5
  Downloading pytz-2022.7-py2.py3-none-any.whl (499 kB)
     |████████████████████████████████| 499 kB 5.2 MB/s eta 0:00:01
Collecting lxml>=4.9.1
  Downloading lxml-4.9.2-cp39-cp39-macosx_10_15_x86_64.whl (4.8 MB)
     |████████████████████████████████| 4.8 MB 5.6 MB/s eta 0:00:01
Collecting frozendict>=2.3.4
  Downloading frozendict-2.3.4-cp39-cp39-macosx_10_9_x86_64.whl (33 kB)
Collecting html5lib>=1.1
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
     |████████████████████████████████| 112 kB 10.9 MB/s eta 0:00:01
Requirement already satisfied: numpy>=1.16.5 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (1.21.5)
Requirement already satisfied: pandas>=1.3.0 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (1.4.2)
Requirement already satisfied: cryptography>=3.3.2 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (3.4.8)
Requirement already satisfied: beautifulsoup4>=4.11.1 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (4.11.1)
Requirement already satisfied: appdirs>=1.4.4 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (1.4.4)
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: requests>=2.26 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (2.27.1)
Requirement already satisfied: soupsieve>1.2 in ./opt/anaconda3/lib/python3.9/site-packages (from beautifulsoup4>=4.11.1->yfina
nce) (2.3.1)
Requirement already satisfied: cffi>=1.12 in ./opt/anaconda3/lib/python3.9/site-packages (from cryptography>=3.3.2->yfinance) (
1.15.0)
Requirement already satisfied: pycparser in ./opt/anaconda3/lib/python3.9/site-packages (from cffi>=1.12->cryptography>=3.3.2->
yfinance) (2.21)
Requirement already satisfied: webencodings in ./opt/anaconda3/lib/python3.9/site-packages (from html5lib>=1.1->yfinance) (0.5.
1)
Requirement already satisfied: six>=1.9 in ./opt/anaconda3/lib/python3.9/site-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in ./opt/anaconda3/lib/python3.9/site-packages (from pandas>=1.3.0->yfina
nce) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./opt/anaconda3/lib/python3.9/site-packages (from requests>=2.26->yfina
nce) (1.26.9)
Requirement already satisfied: charset-normalizer~=2.0.0 in ./opt/anaconda3/lib/python3.9/site-packages (from requests>=2.26->y
finance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in ./opt/anaconda3/lib/python3.9/site-packages (from requests>=2.26->yfinance) (3.3
)
Requirement already satisfied: certifi>=2017.4.17 in ./opt/anaconda3/lib/python3.9/site-packages (from requests>=2.26->yfinance
) (2021.10.8)
Installing collected packages: pytz, multitasking, lxml, html5lib, frozendict, yfinance
  Attempting uninstall: pytz
    Found existing installation: pytz 2021.3
    Uninstalling pytz-2021.3:
      Successfully uninstalled pytz-2021.3
  Attempting uninstall: lxml
    Found existing installation: lxml 4.8.0
    Uninstalling lxml-4.8.0:
      Successfully uninstalled lxml-4.8.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is th
e source of the following dependency conflicts.
conda-repo-cli 1.0.4 requires pathlib, which is not installed.
Successfully installed frozendict-2.3.4 html5lib-1.1 lxml-4.9.2 multitasking-0.0.11 pytz-2022.7 yfinance-0.2.3
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
pip install getFamaFrenchFactors
```

```
Collecting getFamaFrenchFactors
  Downloading getFamaFrenchFactors-0.0.5-py3-none-any.whl (4.6 kB)
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
Requirement already satisfied: pandas in ./opt/anaconda3/lib/python3.9/site-packages (from getFamaFrenchFactors) (1.4.2)
Requirement already satisfied: requests in ./opt/anaconda3/lib/python3.9/site-packages (from getFamaFrenchFactors) (2.27.1)
Requirement already satisfied: beautifulsoup4 in ./opt/anaconda3/lib/python3.9/site-packages (from bs4->getFamaFrenchFactors) (
4.11.1)
Requirement already satisfied: soupsieve>1.2 in ./opt/anaconda3/lib/python3.9/site-packages (from beautifulsoup4->bs4->getFamaF
renchFactors) (2.3.1)
Requirement already satisfied: python-dateutil>=2.8.1 in ./opt/anaconda3/lib/python3.9/site-packages (from pandas->getFamaFrenc
hFactors) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./opt/anaconda3/lib/python3.9/site-packages (from pandas->getFamaFrenchFactors)
(2022.7)
Requirement already satisfied: numpy>=1.18.5 in ./opt/anaconda3/lib/python3.9/site-packages (from pandas->getFamaFrenchFactors)
(1.21.5)
Requirement already satisfied: six>=1.5 in ./opt/anaconda3/lib/python3.9/site-packages (from python-dateutil>=2.8.1->pandas->ge
tFamaFrenchFactors) (1.16.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in ./opt/anaconda3/lib/python3.9/site-packages (from requests->getFama
FrenchFactors) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in ./opt/anaconda3/lib/python3.9/site-packages (from requests->getFamaFrenchFactors
) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in ./opt/anaconda3/lib/python3.9/site-packages (from requests->getFamaFrenchF
actors) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./opt/anaconda3/lib/python3.9/site-packages (from requests->getFamaFren
chFactors) (1.26.9)
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... done
  Created wheel for bs4: filename=bs4-0.0.1-py3-none-any.whl size=1272 sha256=7d661e9863fefd937a5d8630fffac6f456cd271026e6cecde
47f8495953c081a
  Stored in directory: /Users/kashishgupta/Library/Caches/pip/wheels/73/2b/cb/099980278a0c9a3e57ff1a89875ec07bfa0b6fcbebb9a8cad
3
Successfully built bs4
Installing collected packages: bs4, getFamaFrenchFactors
Successfully installed bs4-0.0.1 getFamaFrenchFactors-0.0.5
Note: you may need to restart the kernel to use updated packages.
```

In [6]:

```python
import yfinance as yf
import getFamaFrenchFactors as gff
```

In [16]:

```python
# pick the asset of interest, here I have chosen stock of Meta for a period of 10 years

ticker = 'meta'                        # choose the stock of interest
start = dt.datetime(2013,1,31)        # start date
end = dt.datetime.now()               # takes date today automatically, 04th Jan 2023
```

In [17]:

```python
# download the relevant dataset from yahoo finance

stock_data = yf.download(ticker, start, end)
```

```
[*********************100%***********************]  1 of 1 completed
```

In [19]:

```python
# check for missing values
stock_data.isnull().sum()

# there are no null values
```

Out[19]:

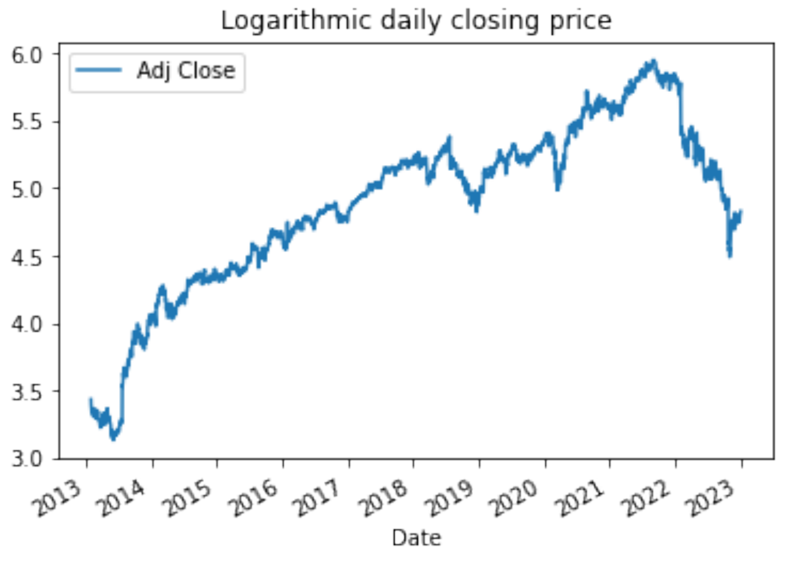
```
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

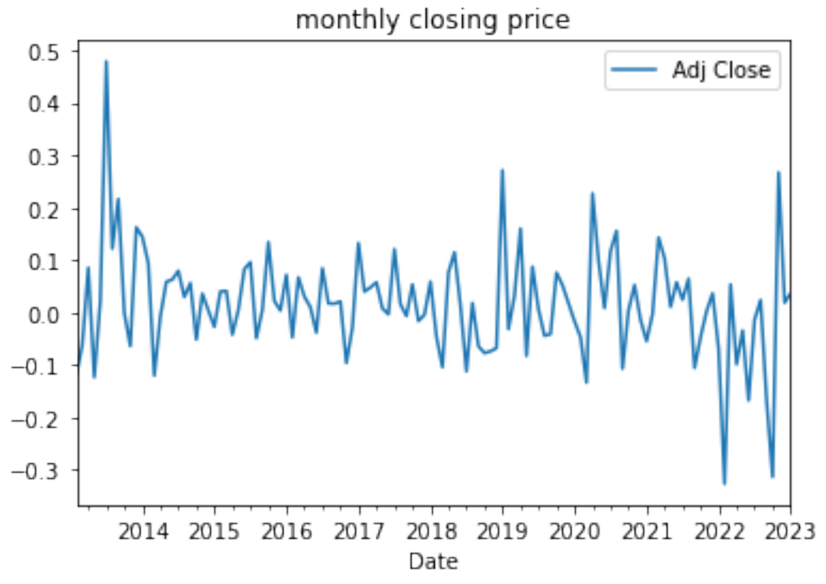
In [90]:

```python
# create chart for daily and monthly closing price

plt.figure()
np.log(stock_data['Adj Close']).plot()              # 'Adj Close' is the closing price of the stock
plt.legend(loc='best')
plt.title('Logarithmic daily closing price',fontsize=12)

# calculate for monthly closing price

mon = stock_data.resample('1M').last()
mon_rets = mon.pct_change().dropna()

#plots of montly rets:
plt.figure()
(mon_rets['Adj Close']).plot()
plt.legend(loc='best')
plt.title('monthly closing price',fontsize=12)
```

Out[90]:

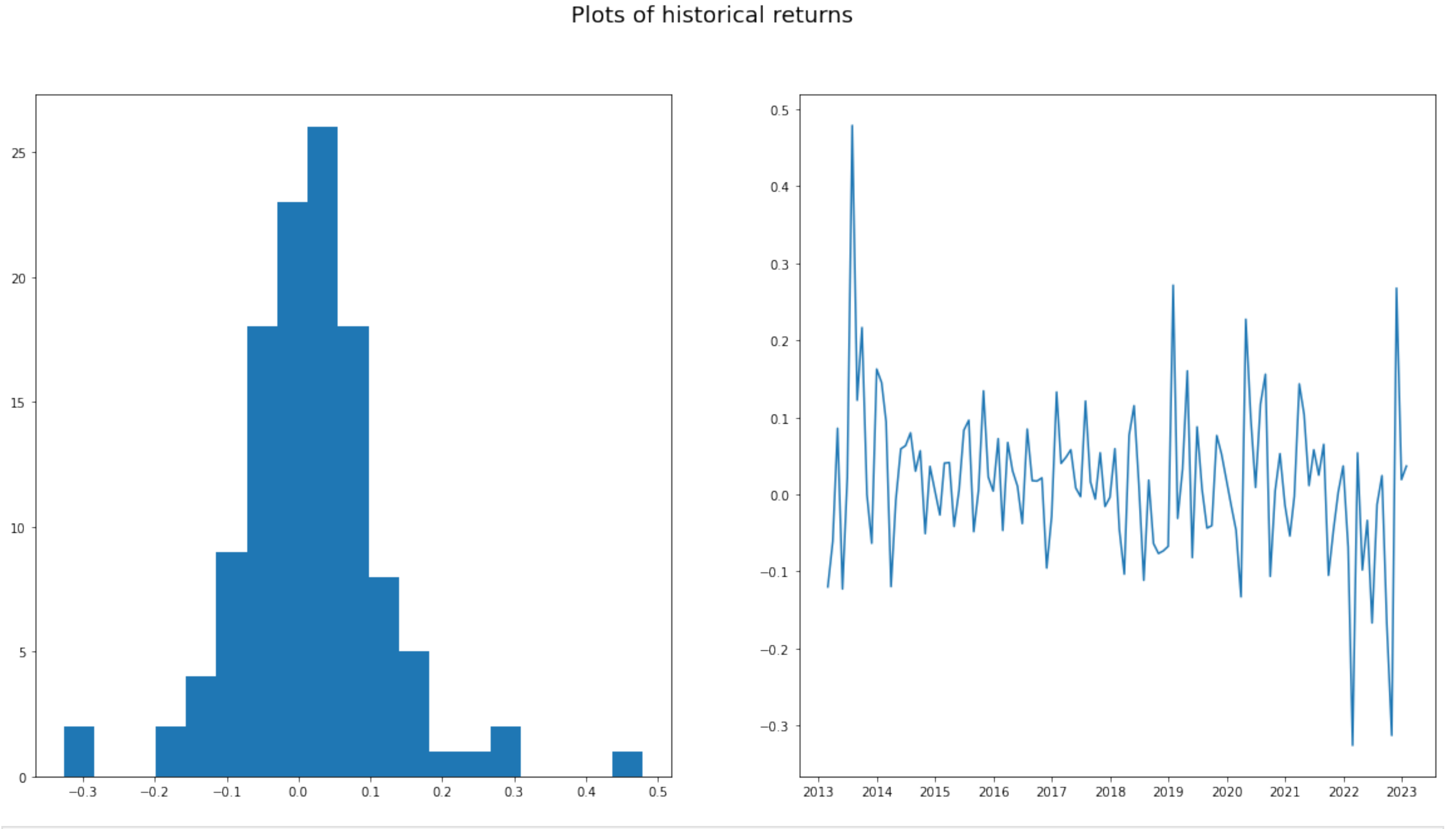
```
Text(0.5, 1.0, 'monthly closing price')
```





In [30]:
```
# visualize for daily asset returns

#get daily returns:
stock_returns = stock_data['Adj Close'].resample('M').last().pct_change().dropna()

#lets get plots of the daily returns:
plt.figure()
fig1, axs = plt.subplots(1, 2,figsize=(20, 10))
axs[0].hist(stock_returns,bins= 'fd')
axs[1].plot(stock_returns)
fig1.suptitle('Plots of historical returns',fontsize=18)
```

Out[30]:
```
Text(0.5, 0.98, 'Plots of historical returns')

<Figure size 432x288 with 0 Axes>
```



In [33]:
```
# do a stationarity check on daily returns
# we want stationary data to take forward the analysis

adf = adfuller(stock_returns)
print('ADF Statistic: %f' % adf[0])
print('p-value: %f' % adf[1])

# The p-value is significant which means we reject the null hypothesis and conclude that our data is stationary and there is no
```

```
ADF Statistic: -10.432680
p-value: 0.000000
```

In [92]:
```
# get fama-french estimates for 3 factors
# these estimates are for the stock market in general
# We have Mkt-RF: Market return - risk free rate, SMB: excess return on small cap stocks, HML: excess return on value stock
# we get monthly estimates

ff3_monthly = gff.famaFrench3Factor(frequency='m')
ff3_monthly.rename(columns={"date_ff_factors": 'Date'}, inplace=True)
ff3_monthly.set_index('Date', inplace=True)
ff3_monthly.head()
```

Out[92]:

Date	Mkt-RF	SMB	HML	RF
1926-07-31	0.0296	-0.0256	-0.0243	0.0022
1926-08-31	0.0264	-0.0117	0.0382	0.0025
1926-09-30	0.0036	-0.0140	0.0013	0.0023
1926-10-31	-0.0324	-0.0009	0.0070	0.0032
1926-11-30	0.0253	-0.0010	-0.0051	0.0031

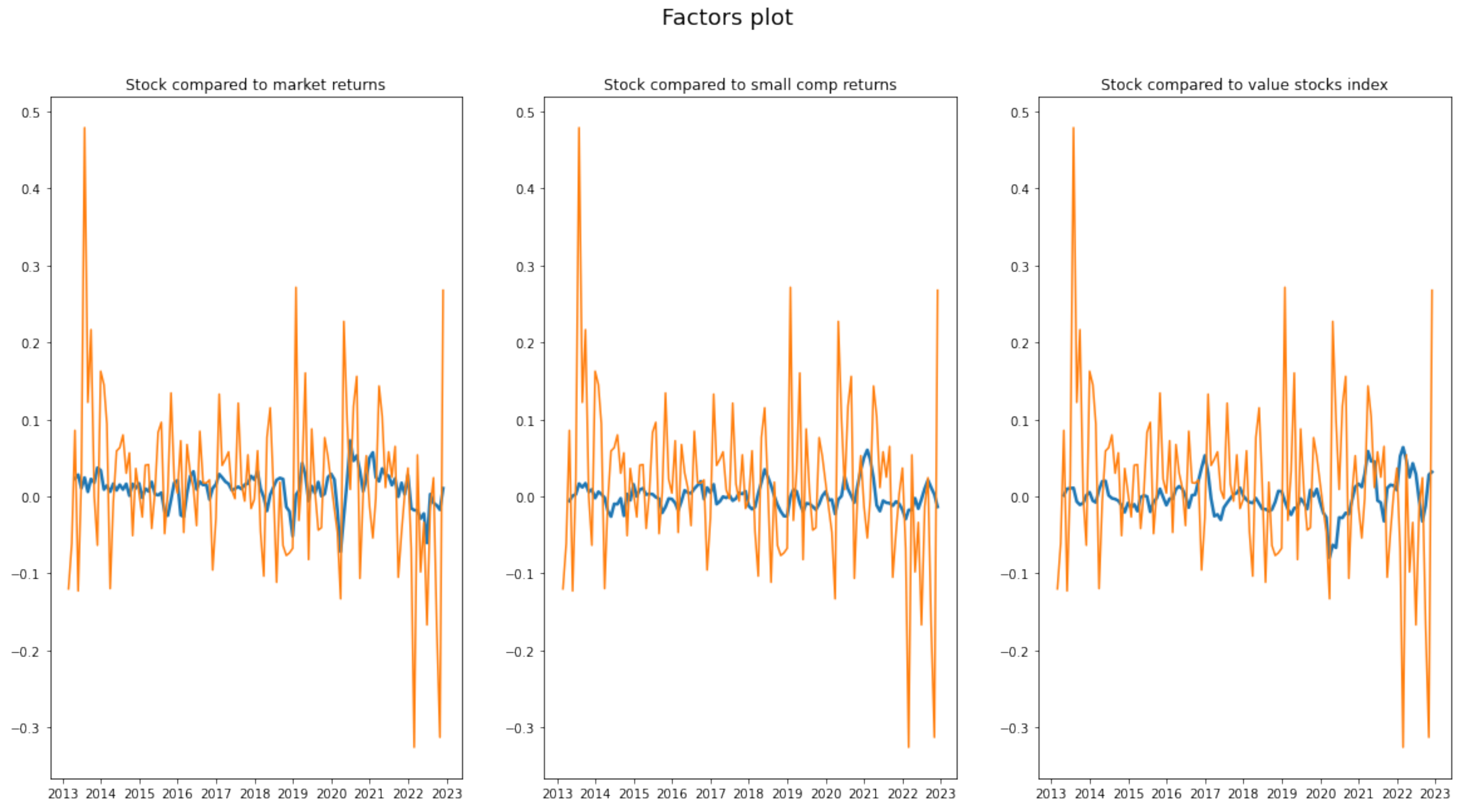
In [93]:
```
# combine fama-french estimates with return on meta stocks calculated above

stock_returns.name = "monthly return"
fama_french = ff3_monthly.merge(stock_returns,on='Date')
fama_french = fama_french.dropna()
fama_french.head()
```

Out[93]:

Date	Mkt-RF	SMB	HML	RF	monthly return
2013-02-28	0.0129	-0.0028	0.0011	0.0	-0.120400
2013-03-31	0.0403	0.0081	-0.0019	0.0	-0.061284
2013-04-30	0.0155	-0.0236	0.0045	0.0	0.085614
2013-05-31	0.0280	0.0173	0.0263	0.0	-0.123154
2013-06-30	-0.0120	0.0133	0.0003	0.0	0.021766

In [56]:
```
# visualize the moving average of the fama-french models

stock = 'monthly return'
plt.figure()
fig3, axs = plt.subplots(1, 3,figsize=(20, 10))
axs[0].plot(fama_french['Mkt-RF'].rolling(3).mean(),linewidth=2.5)
axs[0].plot(fama_french[stock])
axs[0].set_title('Stock compared to market returns')
axs[1].plot(fama_french['SMB'].rolling(3).mean(),linewidth=2.5)
axs[1].plot(fama_french[stock])
axs[1].set_title('Stock compared to small comp returns')
axs[2].plot(fama_french['HML'].rolling(3).mean(),linewidth=2.5)
axs[2].plot(fama_french[stock])
axs[2].set_title('Stock compared to value stocks index')
fig3.suptitle('Factors plot',fontsize=18)

# the organge line depicts meta's returns
# the blue lines highlight fama-french factors
```

Out[56]:
```
Text(0.5, 0.98, 'Factors plot')

<Figure size 432x288 with 0 Axes>
```



In [94]:
```python
# caclulate pearson correlation coefficient

cor = fama_french.corr()

print(f'{stock} correlation with market index:',cor['Mkt-RF'][0])
print(f'{stock} correlation with small-company portfolio index:',cor['SMB'][0])
print(f'{stock} correlation with value stocks index:',cor['HML'][0])

# maximum correlation is with market index and least is with value stocks
```

```
monthly return correlation with market index: 1.0
monthly return correlation with small-company portfolio index: 0.2773450153084419
monthly return correlation with value stocks index: 0.04871147109950043
```

In [63]:
```python
# run a simple linear regression and estimate returns

X = fama_french[['Mkt-RF', 'SMB', 'HML']]
y = fama_french['monthly return'] - fama_french['RF']
X = sm.add_constant(X)
ff_model = sm.OLS(y, X).fit()
print(ff_model.summary())

intercept, b1, b2, b3 = ff_model.params

# the R-squared is: 0.242, which means that the Fama-French factors account for ~24% of the variability of the asset return. The
# the p-value of SMB is insigificant at 5% LOS and hence it might imply that SMB doesnt explain the variation in return for META
# The coefficient of Mkt-RF is positive and implies that as market premium increases so does return on META stocks, hence META s
# The coefficient of HML is negative which means there doesnt exist any value premium for META stock
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.242
Model:                            OLS   Adj. R-squared:                  0.222
Method:                 Least Squares   F-statistic:                     12.12
Date:                Wed, 04 Jan 2023   Prob (F-statistic):           6.06e-07
Time:                        12:06:30   Log-Likelihood:                 116.89
No. Observations:                 118   AIC:                            -225.8
Df Residuals:                     114   BIC:                            -214.7
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0038      0.009      0.440      0.661      -0.013       0.021
Mkt-RF         1.1394      0.200      5.691      0.000       0.743       1.536
SMB           -0.4793      0.342     -1.404      0.163      -1.156       0.197
HML           -0.5359      0.239     -2.244      0.027      -1.009      -0.063
==============================================================================
Omnibus:                       23.521   Durbin-Watson:                   1.897
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              148.194
Skew:                           0.246   Prob(JB):                     6.61e-33
Kurtosis:                       8.468   Cond. No.                         41.3
==============================================================================

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

In [64]:
```
# calculate the average of all the factors

rf = fama_french['RF'].mean()
market_premium = ff3_monthly['Mkt-RF'].mean()
size_premium = ff3_monthly['SMB'].mean()
value_premium = ff3_monthly['HML'].mean()
```

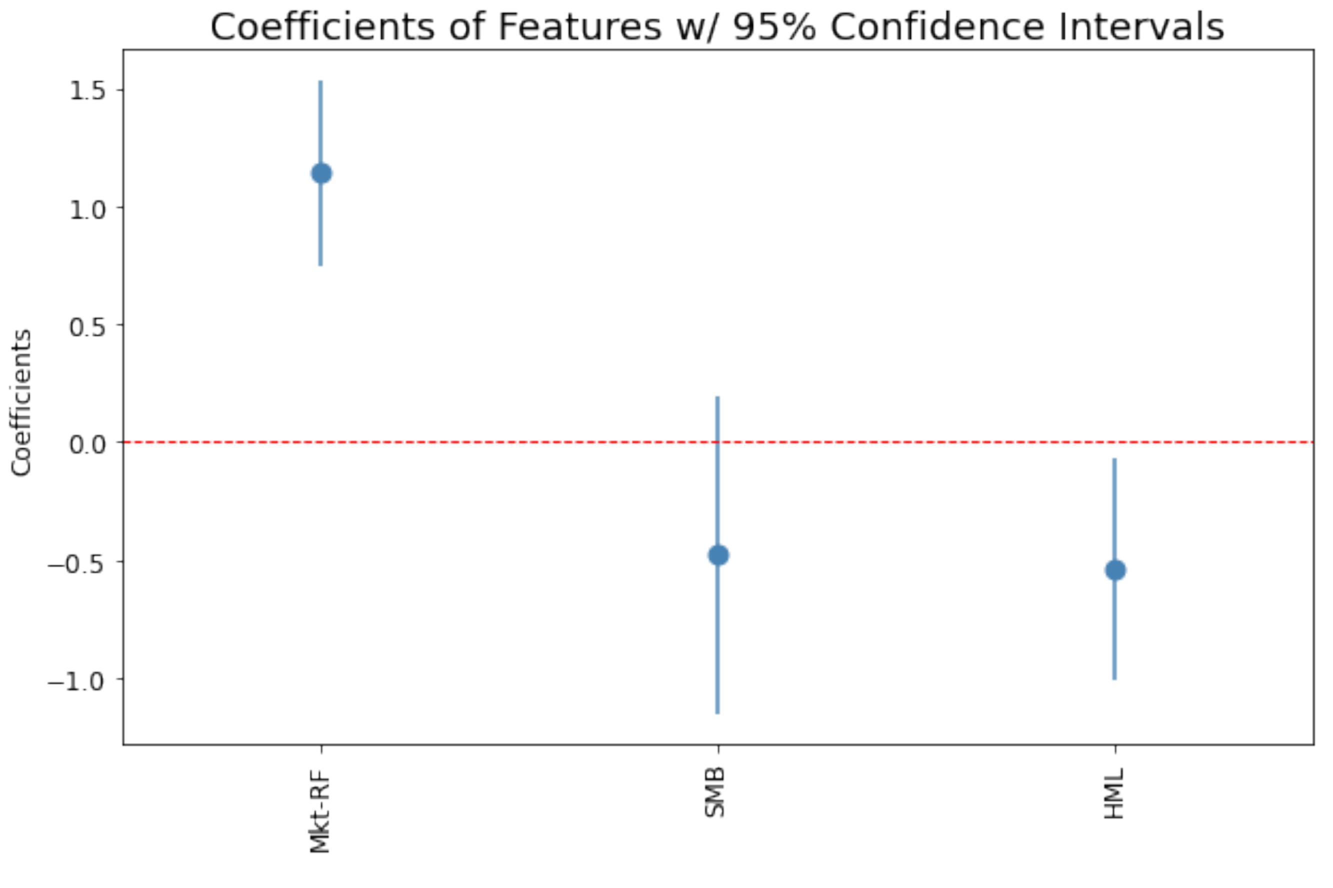
In [66]:
```
# calculate the monthly and annual expected returns

expected_monthly_return = rf + b1 * market_premium + b2 * size_premium + b3 * value_premium
expected_yearly_return = expected_monthly_return * 12

print(f'The expected monthly return for {stock} is:',expected_monthly_return)
print(f'The expected anual return for {stock} is:',((1 + expected_monthly_return) ** 12) - 1)   # using compounding
```

```
The expected monthly return for monthly return is: 0.005389941500103766
The expected anual return for monthly return is: 0.06663156523805869
```

In [69]:
```
# ploting the coefficients and the confidence intervals


#create dataframe of results summary
factors = ['Mkt-RF', 'SMB', 'HML']

coef_df = pd.DataFrame(ff_model.summary().tables[1].data)
coef_df.columns = coef_df.iloc[0]

coef_df = coef_df.drop(0) #drop the extra row with column labels
coef_df = coef_df.set_index(coef_df.columns[0]) #set index to variable names

#change datatype from object to float
coef_df = coef_df.astype(float)

#rename erros column
coef_df.rename(columns = {'std err':'std_err'},inplace = True)

# Drop the constant for plotting purposes
coef_df.drop(['const'],inplace=True)

#add factors to the dataframe to ease plotting
coef_df['factors'] = factors
coef_df['er_interval'] = coef_df['std_err']*1.959

#Plot coefficients
fig, ax = plt.subplots(figsize=(10, 6))

coef_df.plot(x='factors', y='coef', kind='bar',
             ax=ax, color='none', fontsize=12,
             ecolor='steelblue',capsize=0,
             yerr='er_interval', legend=False)

plt.title('Coefficients of Features w/ 95% Confidence Intervals',fontsize=18)
ax.set_ylabel('Coefficients',fontsize=12)
ax.set_xlabel('',fontsize=12)

ax.scatter(x=pd.np.arange(coef_df.shape[0]),
           marker='o', s=80,
           y=coef_df['coef'], color='steelblue')

ax.axhline(y=0, linestyle='--', color='red', linewidth=1) #line to define zero on the y-axis


# observation: the coefficient of SMB is crossing the red line which signify that the second factor might not have any impact or
# We might want to exclude this second factor to generate better and significant results
```

```
/var/folders/1d/l6tryydj5ll_81bf6qw66y1m0000gn/T/ipykernel_12877/2675185018.py:35: FutureWarning: The pandas.np module is depre
cated and will be removed from pandas in a future version. Import numpy directly instead.
  ax.scatter(x=pd.np.arange(coef_df.shape[0]),
```

Out[69]:
```
<matplotlib.lines.Line2D at 0x7fc9895c61f0>
```



In [100…

```
# Since SMB is insigficant, we exclude and rerun the model

X1 = fama_french[['Mkt-RF', 'HML']]
y1 = fama_french['monthly return'] - fama_french['RF']
X1 = sm.add_constant(X1)
ff_model1 = sm.OLS(y1, X1).fit()
print(ff_model1.summary())

# clearly the Rsquared have improved, although not drastically
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.229
Model:                            OLS   Adj. R-squared:                  0.215
Method:                 Least Squares   F-statistic:                     17.05
Date:                Wed, 04 Jan 2023   Prob (F-statistic):           3.27e-07
Time:                        14:52:07   Log-Likelihood:                 115.88
No. Observations:                 118   AIC:                            -225.8
Df Residuals:                     115   BIC:                            -217.4
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0049      0.009      0.560      0.577      -0.012       0.022
Mkt-RF         1.0616      0.193      5.495      0.000       0.679       1.444
HML           -0.5379      0.240     -2.243      0.027      -1.013      -0.063
==============================================================================
Omnibus:                       21.834   Durbin-Watson:                   1.930
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              127.291
Skew:                           0.197   Prob(JB):                     2.29e-28
Kurtosis:                       8.073   Cond. No.                         28.4
==============================================================================

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

bootstrapping

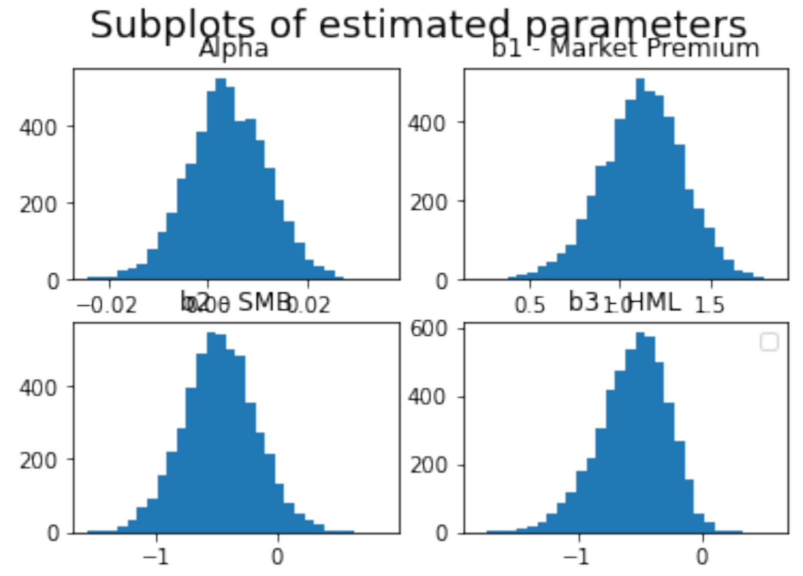
In [79]:

```
# using bootstrapping to get distribution for model parameters
# we will get confidence intervals for asset returns

#get returns - risk free rate
fama_french['returns'] = fama_french['monthly return'] - fama_french['RF']

#define variables to be filled with the simulations
boot_betas = []
mod_beta = 0
boot_return = 0
boot_returns = []

#loop 5000 times
for i in range(5000):

    sample = fama_french.sample(n=len(fama_french), replace=True)
    sample_means = sample.mean() #used to create the manual regression

    boot_model = sm.OLS(sample['returns'], sm.add_constant(sample[factors])) #here we have remove the risk free
    boot_result = boot_model.fit()

    mod_beta = boot_result.params
    boot_betas.append(mod_beta)

    boot_return = mod_beta[1]*sample_means['Mkt-RF'] + mod_beta[2]*sample_means['SMB'] + mod_beta[3]*sample_means['HML']

    boot_returns.append(boot_return)

#normality test:
r_test = stats.normaltest(boot_returns)

boot_betas = pd.DataFrame(boot_betas) #convert results to a dataframe

#visualize results distributions:
plt.figure()
fig4, axs = plt.subplots(2, 2)
axs[0,0].hist(boot_betas['const'],bins=30)
axs[0,0].set_title('Alpha')
axs[0,1].hist(boot_betas['Mkt-RF'],bins=30)
axs[0,1].set_title('b1 - Market Premium')
axs[1,0].hist(boot_betas['SMB'],bins=30)
axs[1,0].set_title('b2 - SMB')
axs[1,1].hist(boot_betas['HML'],bins=30)
axs[1,1].set_title('b3 - HML')
fig4.suptitle('Subplots of estimated parameters',fontsize=18)
plt.legend()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend(
) is called with no argument.
```

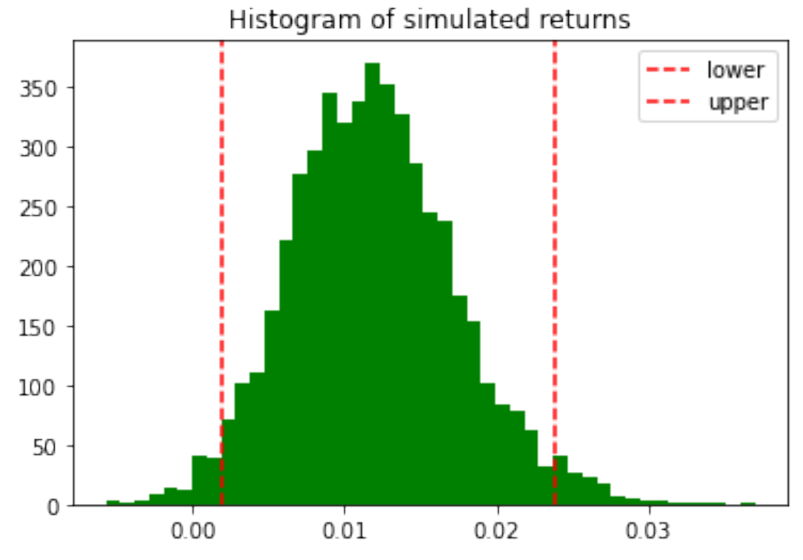
Out[79]:

```
<matplotlib.legend.Legend at 0x7fc9ade20cd0>

<Figure size 432x288 with 0 Axes>
```



In [80]:
```
# simulated returns for Meta stocks

plt.figure()
plt.hist(boot_returns, alpha = 1,bins=45,color = 'green')
plt.axvline(np.percentile(boot_returns,2.5),color = 'red',linestyle = '--')
plt.axvline(np.percentile(boot_returns,97.5),color = 'red',linestyle = '--')
plt.title('Histogram of simulated returns')
plt.legend(['lower','upper'])
```

Out[80]:
```
<matplotlib.legend.Legend at 0x7fc9ade36310>
```



In [99]:
```
# final results

print('Number of times when asset returns are less than 0: ',sum(np.array(boot_returns) <= 0 / len(boot_returns)))
print(f'The average of expected monthly bootstrap return for Meta is:',(np.array(boot_returns).mean()*100),'%')

print("The WORST CASE estimation for Meta stock montly returns is: ",(np.percentile(boot_returns,2.5)*100),'%')
print("The BEST CASE estimation for Meta stock montly returns is: ",(np.percentile(boot_returns,97.5)*100),'%')

# Using the bootstrap method we have obtained a slightly less-biased estimate for the returns as well as an upper and lower bour
```

```
Number of times when asset returns are less than 0:  45
The average of expected monthly bootstrap return for Meta is: 1.1939001600593446 %
The WORST CASE estimation for Meta stock montly returns is:  0.19071959881273465 %
The BEST CASE estimation for Meta stock montly returns is:  2.383363344904974 %
```