MFE 409; HW4

Aliaksei Kanstantsinau

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

In [10]:
```
df = pd.read_csv('hw4_returns.csv')
df.head()
```

Out[10]:

	Date	Return
0	1/2/2014	0.004572
1	1/3/2014	0.006045
2	1/6/2014	-0.001432
3	1/7/2014	0.015461
4	1/8/2014	0.000763

Choosing VaR Technique

Historical Method

Download the excel file which contains the time series of gains for a strategy from 1/2/2014 to 12/19/2017

(a) For each day in 2015-2017, compute historical VaR and exponential weighted 1-day 99%-VaR (with λ = 0.995).

- Compute historical VaR:

In [11]:
```
# convert to date, calculate mu
df['Date'] = pd.to_datetime(df['Date'])
```

In [21]:
```
# set lambda
lmbd = .995
# get historical var
df['hist_VaR'] = df['Return'].rolling(window=250).quantile(.01)
# calculate exponentially weighted returns
exp_weights = (1 - lmbd) * (lmbd ** np.arange(250)[::-1])
# get exponential var
df['exp_VaR'] = df['Return'].rolling(window=250).apply(lambda x: np.quantile
```

In [20]:
```
df.head(-5)
```

Out[20]:

	Date	Return	hist_VaR	exp_VaR
0	2014-01-02	0.004572	NaN	NaN
1	2014-01-03	0.006045	NaN	NaN
2	2014-01-06	-0.001432	NaN	NaN
3	2014-01-07	0.015461	NaN	NaN
4	2014-01-08	0.000763	NaN	NaN
...	...	...	...	...
990	2017-12-06	0.019731	-0.078661	-0.00016
991	2017-12-07	0.003931	-0.078661	-0.00016
992	2017-12-08	-0.000508	-0.078661	-0.00016
993	2017-12-11	-0.011339	-0.078661	-0.00016
994	2017-12-12	0.008876	-0.078661	-0.00016

995 rows × 4 columns

b) Backtest the measures for VaR you obtained in question 1. How many exceptions did the two measures produce? What do you conclude?

In [24]:
```
df['hist_exception'] = np.where(df['Return'] <= df['hist_VaR'], 1, 0)
df['exp_exception'] = np.where(df['Return'] <= df['exp_VaR'], 1, 0)
df.head()
```

Out[24]:

	Date	Return	hist_VaR	exp_VaR	hist_exception	exp_exception
0	2014-01-02	0.004572	NaN	NaN	0	0
1	2014-01-03	0.006045	NaN	NaN	0	0
2	2014-01-06	-0.001432	NaN	NaN	0	0
3	2014-01-07	0.015461	NaN	NaN	0	0
4	2014-01-08	0.000763	NaN	NaN	0	0

In [25]:
```
print(f'Historical Exceptions: ', df['hist_exception'].sum())
print(f'Exponential Exceptions: ', df['exp_exception'].sum())
```

```
Historical Exceptions:  14
Exponential Exceptions:  391
```

- Based on the results above we can conclude that historical var is much better.

(c) For each day in the sample, compute the 95% confidence intervals of the historical VaR and the exponential weighted VaR you obtained in question 1, using both parametric (for the historical VaR) and bootstrap methods (for the two measures). For the parametric method, assume the gains are normally distributed.

In [26]:
```
from scipy.stats import norm
```

In [37]:
```
dev = df['Return'].std()
z_score = norm.ppf(.975)
std = dev/np.sqrt(len(df['Return']))
error = z_score * std
ewma_dev = df['Return'].ewm(alpha=(1 - lmbd)).std()
ewma_error = z_score * ewma_dev
```

In [39]:
```
hist_var_parametric = (df['hist_VaR'] - error, df['hist_VaR'] + error)
exp_var_parametric = (df['exp_VaR'] - ewma_error, df['exp_VaR'] + ewma_error
print(f'Historical parametric: ', hist_var_parametric)
print(f'Exponential parametric: ', exp_var_parametric)
```

```
Historical parametric:  (0           NaN
1            NaN
2            NaN
3            NaN
4            NaN
         ...
995   -0.080223
996   -0.080223
997   -0.080223
998   -0.080223
999   -0.080223
Name: hist_VaR, Length: 1000, dtype: float64, 0           NaN
1            NaN
2            NaN
3            NaN
4            NaN
         ...
995   -0.077099
996   -0.077099
997   -0.077099
998   -0.077099
999   -0.077099
Name: hist_VaR, Length: 1000, dtype: float64)
Exponential parametric:  (0           NaN
1            NaN
2            NaN
3            NaN
4            NaN
         ...
995   -0.054211
996   -0.054373
997   -0.054418
998   -0.054310
999   -0.054189
Length: 1000, dtype: float64, 0           NaN
1            NaN
2            NaN
3            NaN
4            NaN
         ...
995    0.053891
996    0.054053
997    0.054098
998    0.053990
999    0.053869
Length: 1000, dtype: float64)
```

In [33]:

```python
def bootstrap(series, quantile, n_bootstraps=1000):
    bootstrapped_vals = []
    for _ in range(n_bootstraps):
        sample = series.sample(n=len(series), replace=True)
        bootstrapped_vals.append(sample.quantile(quantile))
    lower_bound = np.percentile(bootstrapped_vals, 2.5)
    upper_bound = np.percentile(bootstrapped_vals, 97.5)
    return lower_bound, upper_bound
```

```
hist_var_bootstrap = bootstrap(df['Return'], 0.01)
exp_var_bootstrap = bootstrap(df['exp_VaR'], 0.01)
print(f'Historical bootstrap: ', hist_var_bootstrap)
print(f'Exponential bootstrap: ', exp_var_bootstrap)
```

```
Historical bootstrap:  (-0.089048805, -0.0625473249)
Exponential bootstrap:  (-0.00017253111655546723, -0.00017253111655546723)
```

Model-building approach

(a) Compute volatility using the EWMA with λ = 0.94. Compute the corresponding measure of VaR

In [40]:
```
lmbd = .94
df['ewma'] = np.sqrt(.064*(df['Return'].shift(-1)**2 + lmbd*df['Return'].shi
```

(b) Use maximum likelihood estimation to estimate a GARCH model for volatility. Compute the corresponding measure of VaR.

In [42]:
```
from arch import arch_model
```

In [43]:
```
garch_model = arch_model(df['Return'], p=1, q=1)
garch_fit = garch_model.fit(update_freq=10)
print(garch_fit.summary())
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: -2338.2613557943137
            Iterations: 2
            Function evaluations: 26
            Gradient evaluations: 2
                     Constant Mean - GARCH Model Results
============================================================================
==
Dep. Variable:                  Return   R-squared:                      0.0
00
Mean Model:              Constant Mean   Adj. R-squared:                 0.0
00
Vol Model:                       GARCH   Log-Likelihood:               2338.
26
Distribution:                   Normal   AIC:                         -4668.
52
Method:             Maximum Likelihood   BIC:                         -4648.
89
                                         No. Observations:                10
00
Date:                 Mon, Apr 29 2024   Df Residuals:                     9
99
Time:                         17:14:22   Df Model:
1
                                   Mean Model
============================================================================
===
                 coef    std err          t      P>|t|       95.0% Conf. I
nt.
----------------------------------------------------------------------------
---
mu         -1.7125e-03  7.128e-04     -2.402  1.629e-02 [-3.110e-03,-3.154e-
04]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      1.2695e-05  4.058e-12  3.129e+06      0.000 [1.269e-05,1.270e-05]
alpha[1]       0.1000  3.572e-02      2.800  5.115e-03   [2.999e-02,  0.170]
beta[1]        0.8800  3.326e-02     26.460 2.820e-154     [  0.815,  0.945]
============================================================================

Covariance estimator: robust
```

```
/Users/a.kanstantsinau/anaconda3/lib/python3.11/site-packages/arch/univariat
e/base.py:311: DataScaleWarning: y is poorly scaled, which may affect conver
gence of the optimizer when
estimating the model parameters. The scale of y is 0.0006348. Parameter
estimation work better when this value is between 1 and 1000. The recommende
d
rescaling is 100 * y.

This warning can be disabled by either rescaling y before initializing the
model or by setting rescale=False.

  warnings.warn(
```

In [46]:
```
df['garch'] = np.sqrt(.000012695 + .1*df['Return'].shift(-1)**2 + .88*df['Re
df.head(-5)
```

Out[46]:

	Date	Return	hist_VaR	exp_VaR	hist_exception	exp_exception	ewma
0	2014-01-02	0.004572	NaN	NaN	0	0	0.003997
1	2014-01-03	0.006045	NaN	NaN	0	0	0.003814
2	2014-01-06	-0.001432	NaN	NaN	0	0	0.003917
3	2014-01-07	0.015461	NaN	NaN	0	0	0.002228
4	2014-01-08	0.000763	NaN	NaN	0	0	0.002427
...	...	...	...	...	...	...	...
990	2017-12-06	0.019731	-0.078661	-0.00016	0	0	0.002877
991	2017-12-07	0.003931	-0.078661	-0.00016	0	0	0.003494
992	2017-12-08	-0.000508	-0.078661	-0.00016	0	1	0.005043
993	2017-12-11	-0.011339	-0.078661	-0.00016	0	1	0.009928
994	2017-12-12	0.008876	-0.078661	-0.00016	0	0	0.012109

995 rows × 8 columns

(c) Compare the results from the two approaches.

- Seems that garch is more convervative.

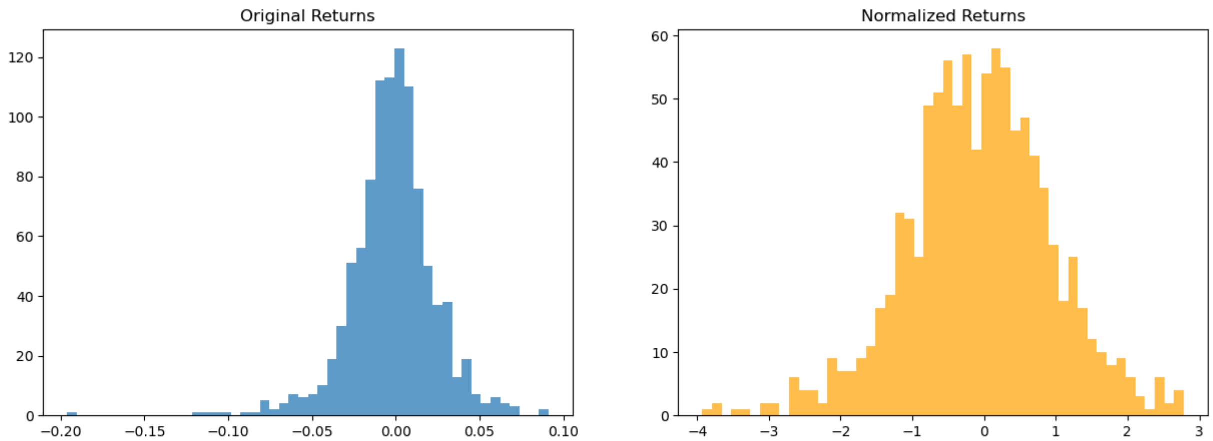
A mixed approach

(a) For each day in the sample, compute the volatility of the portfolio in the previous month. Normalize gains with estimated volatility. Compare the distribution of the normalized gain with the original ones.

In [48]:
```
import matplotlib.pyplot as plt
```

In [50]:
```
df['Vol'] = df['Return'].rolling(window=21).std()
df['N_Return'] = df['Return'] / df['Vol']
```

```
plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
plt.hist(df['Return'], bins=50, alpha=0.7)
plt.title('Original Returns')
plt.subplot(1, 2, 2)
plt.hist(df['N_Return'], bins=50, alpha=0.7, color='orange')
plt.title('Normalized Returns')
plt.show()
```



(b) Develop an approach to measure VaR which takes advantage of your response to the previous question. Implement it and compare its exceptions with the previous approaches. Optional: You can use the approach of extreme value theory.

- Lets utilize standardised returns to calculate VaR

In [53]:
```
cl = 0.05
normalized_var = np.percentile(df['N_Return'], (1-cl) * 100)
current_volatility = df['Vol'].iloc[-1]
denormalized_var = normalized_var * current_volatility

df['norm_exception'] = np.where(df['Return'] <= denormalized_var, 1, 0)
norm_exceptions = df['norm_exception'].sum()

print(f'Normalized Exceptions: ', norm_exceptions)
```

```
Normalized Exceptions:  0
```

Conclusion.

Combining your answers to the previous questions, write a proposal to the head of trading to measure the risk of this trade in real time, justifying your choices.

- Traditionally, our approach relied heavily on VaR, which does not fully take into account rapidly changing market volatility. The proposal is the following, we should utilize normalized returns in our VaR calculations, that will help us achieve more sophisticated defense system.

In [ ]: