Untitled1

May 6, 2019

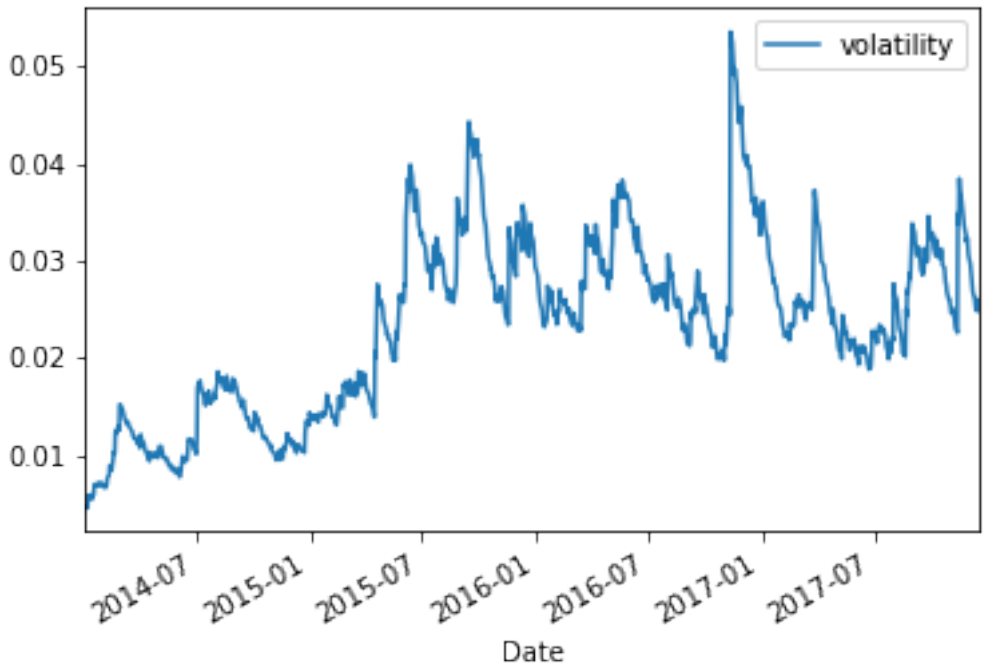
1 Risk HW4

Name: Huanyu Liu
Group members: Sejal Bharati, Huanyu Liu, Tongsu Peng

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt
from scipy.stats import norm
import sklearn.linear_model as lm
import seaborn as sns
```

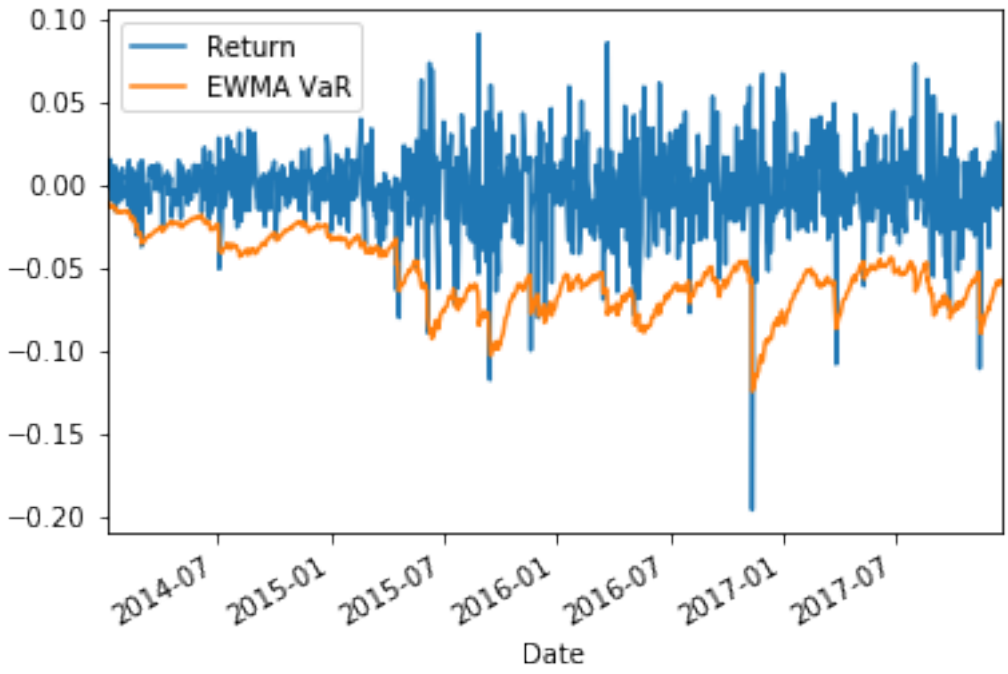
1.0.1 1.1

In [2]:
```
data = pd.read_csv('/Users/huanyu/Desktop/RiskManagement/hw4/hw3_returns2.csv')
lamb = 0.94
data['Date'] = pd.to_datetime(data['Date'])
# data.loc[data['Date'] == '2014-12-31','volatility'] = data.loc[data['Date'] < '2014-
volatility = list()
volatility.append(data.loc[0,'Return']**2)
for i in data.index[1:]:
    # print(i)
    volatility.append(volatility[i - 1] * lamb + (1 - lamb) * np.power(data.loc[i,'Retu
data['volatility'] = volatility
data['volatility'] = np.sqrt(data['volatility'])
data[['Date','volatility']].plot(x='Date')
plt.show()
data['EWMA VaR'] = data['volatility'] * norm.ppf(0.01)
print(data[['Date','volatility','EWMA VaR']])
exception1 = (data['Return'] < data['EWMA VaR']).value_counts()[True]
data[['Date','Return','EWMA VaR']].plot(x='Date')
plt.show()
```



	Date	volatility	EWMA VaR
0	2014-01-02	0.004572	-0.010635
1	2014-01-03	0.004673	-0.010871
2	2014-01-06	0.004544	-0.010571
3	2014-01-07	0.005810	-0.013515
4	2014-01-08	0.005636	-0.013111
5	2014-01-09	0.005465	-0.012714
6	2014-01-10	0.005770	-0.013422
7	2014-01-13	0.005656	-0.013157
8	2014-01-14	0.005495	-0.012782
9	2014-01-15	0.006030	-0.014028
10	2014-01-16	0.005928	-0.013789
11	2014-01-17	0.006954	-0.016178
12	2014-01-21	0.006995	-0.016272
13	2014-01-22	0.006824	-0.015875
14	2014-01-23	0.007042	-0.016383
15	2014-01-24	0.006937	-0.016138
16	2014-01-27	0.007173	-0.016686
17	2014-01-28	0.007041	-0.016381
18	2014-01-29	0.006928	-0.016116
19	2014-01-30	0.006922	-0.016104
20	2014-01-31	0.006720	-0.015634
21	2014-02-03	0.006813	-0.015849
22	2014-02-04	0.006616	-0.015391
23	2014-02-05	0.007071	-0.016451

```
24  2014-02-06    0.006914 -0.016084
25  2014-02-07    0.007618 -0.017721
26  2014-02-10    0.007936 -0.018462
27  2014-02-11    0.008898 -0.020699
28  2014-02-12    0.008641 -0.020101
29  2014-02-13    0.008707 -0.020254
..         ...         ...       ...
970 2017-11-07    0.024360 -0.056671
971 2017-11-08    0.024207 -0.056313
972 2017-11-09    0.023484 -0.054631
973 2017-11-10    0.023055 -0.053634
974 2017-11-13    0.022585 -0.052540
975 2017-11-14    0.034846 -0.081065
976 2017-11-15    0.033785 -0.078597
977 2017-11-16    0.038437 -0.089417
978 2017-11-17    0.037280 -0.086726
979 2017-11-20    0.036155 -0.084109
980 2017-11-21    0.035456 -0.082483
981 2017-11-22    0.034621 -0.080541
982 2017-11-24    0.033573 -0.078103
983 2017-11-27    0.032623 -0.075892
984 2017-11-28    0.031930 -0.074280
985 2017-11-29    0.032299 -0.075139
986 2017-11-30    0.031505 -0.073292
987 2017-12-01    0.030547 -0.071063
988 2017-12-04    0.029654 -0.068986
989 2017-12-05    0.028914 -0.067264
990 2017-12-06    0.028447 -0.066177
991 2017-12-07    0.027597 -0.064200
992 2017-12-08    0.026756 -0.062245
993 2017-12-11    0.026090 -0.060694
994 2017-12-12    0.025388 -0.059062
995 2017-12-13    0.024882 -0.057884
996 2017-12-14    0.025829 -0.060087
997 2017-12-15    0.026021 -0.060534
998 2017-12-18    0.025344 -0.058959
999 2017-12-19    0.024751 -0.057580

[1000 rows x 3 columns]
```



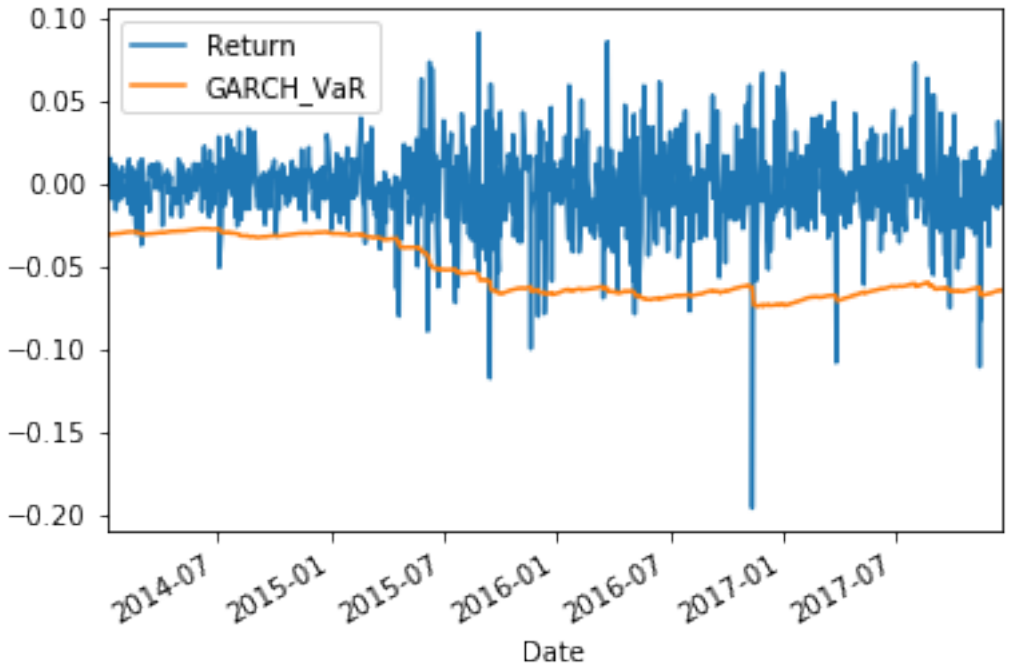
Exceptions: {{exception1}}

1.0.2 1.2

```
In [3]: def garch11(omega, alpha, beta, returns):
            length = len(returns)
            sigma_2 = np.zeros(length)
            sigma_2[0] = omega/(1 - alpha - beta)
            for i in range(1,length):
                sigma_2[i] = omega + alpha * returns[i - 1]**2 + beta * sigma_2[i - 1]
            return sigma_2
        def garch_log_like(parameters, returns):
            if parameters[1] + parameters[2] >= 1:
                return 0
            elif parameters[0] < 0 or parameters[1] < 0 or parameters[2] < 0:
                return 0
            omega, alhpa, beta = parameters
            sigma_2 = garch11(omega,alhpa,beta,returns)
            log = -np.sum(-np.log(sigma_2) - returns**2/sigma_2)
            return log
        parameters0 = [0.1,0.1,0.1]
        parameters = opt.fmin(garch_log_like,parameters0,args=(data['Return'],),maxiter=10000)
        data['GARCH_vol'] = np.sqrt(garch11(parameters[0],parameters[1],parameters[2],data['Re
        data['GARCH_VaR'] = data['GARCH_vol'] * norm.ppf(0.01)
        exception2 = (data['Return'] < data['GARCH_VaR']).value_counts()[True]
```

```
        print(data[['Date','GARCH_vol','GARCH_VaR']])
        data[['Date','Return','GARCH_VaR']].plot(x='Date')
        plt.show()

Optimization terminated successfully.
         Current function value: -6479.645680
         Iterations: 212
         Function evaluations: 390
          Date  GARCH_vol  GARCH_VaR
0   2014-01-02   0.013314  -0.030974
1   2014-01-03   0.013270  -0.030870
2   2014-01-06   0.013230  -0.030776
3   2014-01-07   0.013180  -0.030660
4   2014-01-08   0.013199  -0.030704
5   2014-01-09   0.013148  -0.030588
6   2014-01-10   0.013098  -0.030471
7   2014-01-13   0.013074  -0.030414
8   2014-01-14   0.013027  -0.030306
9   2014-01-15   0.012978  -0.030192
10  2014-01-16   0.012968  -0.030168
11  2014-01-17   0.012923  -0.030064
12  2014-01-21   0.012949  -0.030124
13  2014-01-22   0.012917  -0.030049
14  2014-01-23   0.012870  -0.029941
15  2014-01-24   0.012850  -0.029894
16  2014-01-27   0.012809  -0.029797
17  2014-01-28   0.012791  -0.029755
18  2014-01-29   0.012748  -0.029656
19  2014-01-30   0.012706  -0.029559
20  2014-01-31   0.012672  -0.029479
21  2014-02-03   0.012624  -0.029368
22  2014-02-04   0.012596  -0.029303
23  2014-02-05   0.012549  -0.029193
24  2014-02-06   0.012546  -0.029186
25  2014-02-07   0.012502  -0.029084
26  2014-02-10   0.012521  -0.029128
27  2014-02-11   0.012516  -0.029116
28  2014-02-12   0.012569  -0.029241
29  2014-02-13   0.012523  -0.029132
..         ...        ...        ...
970 2017-11-07   0.027229  -0.063345
971 2017-11-08   0.027137  -0.063129
972 2017-11-09   0.027099  -0.063042
973 2017-11-10   0.026997  -0.062805
974 2017-11-13   0.026926  -0.062638
975 2017-11-14   0.026848  -0.062457
976 2017-11-15   0.028434  -0.066147
977 2017-11-16   0.028326  -0.065895
```

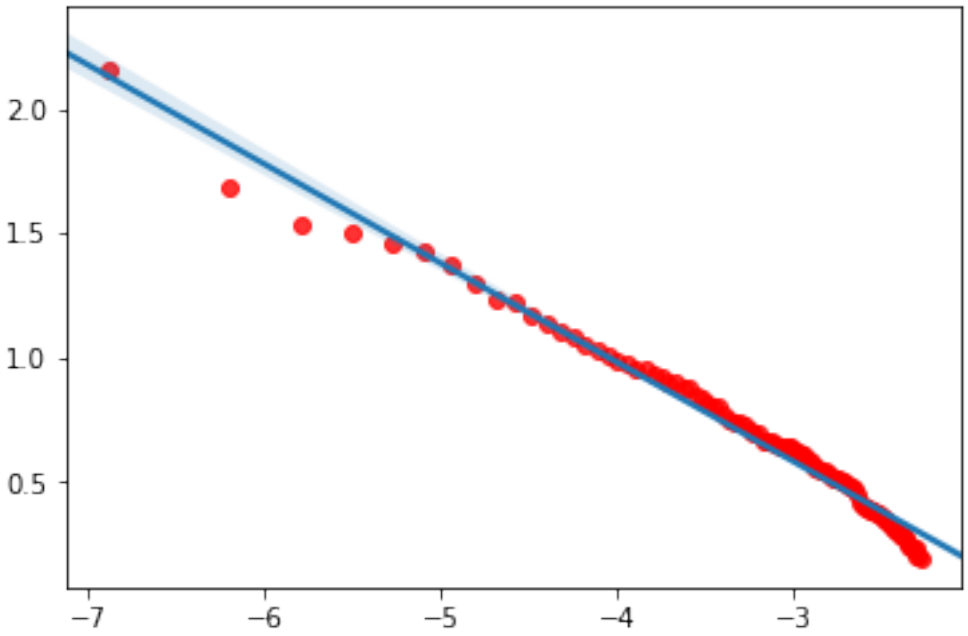
```
978 2017-11-17   0.029112  -0.067725
979 2017-11-20   0.029003  -0.067472
980 2017-11-21   0.028895  -0.067219
981 2017-11-22   0.028847  -0.067108
982 2017-11-24   0.028774  -0.066939
983 2017-11-27   0.028666  -0.066686
984 2017-11-28   0.028567  -0.066456
985 2017-11-29   0.028501  -0.066302
986 2017-11-30   0.028581  -0.066489
987 2017-12-01   0.028498  -0.066297
988 2017-12-04   0.028390  -0.066045
989 2017-12-05   0.028287  -0.065805
990 2017-12-06   0.028200  -0.065604
991 2017-12-07   0.028145  -0.065476
992 2017-12-08   0.028040  -0.065231
993 2017-12-11   0.027933  -0.064983
994 2017-12-12   0.027844  -0.064776
995 2017-12-13   0.027749  -0.064554
996 2017-12-14   0.027674  -0.064379
997 2017-12-15   0.027763  -0.064587
998 2017-12-18   0.027772  -0.064607
999 2017-12-19   0.027680  -0.064392

[1000 rows x 3 columns]
```



Exceptions: {{exception2}}

1.0.3 1.3

```
In [4]: data['monthly_std'] = data['Return'].rolling(22).std().shift(1)
        data['normalized_gain'] = (data['Return'] - data['Return'].rolling(22).mean().shift(1)
        normalized_gain = data.loc[22:,'normalized_gain']
        normalized_gain = sorted(normalized_gain)
        sns.regplot([np.log(x / len(normalized_gain)) for x in range(1,101)],np.log(np.abs(nor
        plt.show()
```



It's linear in the tail, so the left tail of these normalized returns follows a power law.

```
In [5]: def pareto_pdf(xi, beta, y):
            return 1 / beta * np.power(1 + np.multiply(y,xi / beta),-1/xi - 1)

        u = abs(data.loc[22:,'normalized_gain'].quantile(0.05))
        def pareto_log_like(parameters,y,u):
            return -np.sum(np.log(pareto_pdf(parameters[0],parameters[1],y-u)))

        parameters_pareto0 = [0.2,0.8]
        parameters_pareto = opt.fmin(pareto_log_like,parameters_pareto0,args=(np.abs(normalize
        VaR = u + parameters_pareto[1] / parameters_pareto[0] * (np.power(20 * 0.01,-parameter
        VaR *= data.loc[999,'monthly_std']
        print(VaR)
```

```
Optimization terminated successfully.
         Current function value: 44.164183
```

```
         Iterations: 26
         Function evaluations: 50
0.08031002682921835
```

1.4

There are more exceptions for EWMA and GARCH model. However, the trend of the EWMA VaR is similar to the volatility of returns. Thus, you could add capital to invest the stock though the exceptions are not really small. In the opposite, if the VaRs are high and the number of exceptions is small, you will not invest in the stock because of the high VaRs.

2.1

The broad trading strategy of LTCM was to take advantage of small differences in prices among closely related securities. They aim for "relative-value", or "convergence-arbitrage", trades. The key is that the securities would ultimately converge to the same value. This was used in a variety of markets, such as long swap-government

spreads, long mortgage-backed securities versus short government, long high-yielding versus short low- yielding European bonds, Japanese convertible bond arbitrage, equity pairs, and so on. LTCM also had other

non-arbitrage strategies, such as short positions in equity options, bets on takeover stocks, emerging market debt, etc. Their trades are usually profitable, unless there is default or market disruption.

2.2

They needed a lot of leverage because their trading strategy only generate tiny profits. By using leverage, they can produce much more attractive returns. To control risk, they set their leverage to a target volatility of an unlevered position in US equities. They ended up being four times the asset size of the next largest hedge fund.

2.3

1. LTCM was a highly leveraged hedge fund. The 1997, the Debt to Equity Ratio of the fund was 28:1 which was very high.
2. In June 1998 the Firm lost 16% of its capital due to downturn in MBS market, which further increased the leverage ratio to 31:1.
3. On August 17, 1998, Russia de facto defaulted on its Treasury Bonds, which led to global reassessment of Sovereign risks. Since, LTCM's major book was mostly Interest rates swaps and equity volatility, LTCM suffered huge losses in August due to increase in credit spreads, liquidity spreads and risk premia.
4. By end of August the fund had lost 52% of its Dec 1997 value and the leverage ratio increased from 31 to 55:1.
5. In September, Lenders feared that they would have to liquidate their Repo collaterals, because LTCM would not be able to fulfil its Margin calls, because their

capital was substantially reduced, and the firm's value had reduced too mere $400 millions.

6. This led to the demise of LTCM, at which point the Federal Reserve had to Bailout LTCM to avoid large rippling losses in the entire financial market.

2.4

1. LTCM targeted the firm's volatility to an unlevered position in US Equities. As per volatility calculation, the daily dollar volatility of LTCM was close to $45 millions, as stated by them in May 1998.

2. However, LTCM made some very aggressive assumptions about their volatility calculations:
   They assumed that Volatility is constant when in fact in turbulent times it can easily double
   Also, they only focused on Volatility (second) moment which is under the assumption that distribution of PnL is symmetric on both sides. However, credit risks often have higher downside than upside.
   Also, its distribution in tails cannot be assumed as normal, as we can see from past realized returns that most financial series have fatter tails than normal.

3. When the risk models of the firm had started to blow up, they made another mistake of selling off their most liquid investments, because of less profitability, thereby exposing themselves to liquidity risk.

4. LTCM gave more weightage to recent times, (which were periods of low volatility) and ignored major previous crashes such as 1987 crash or sovereign defaults.

2.5

Firstly, historical VaR cannot be used as the sole measure of risk, especially for firms like LTCM which have specialized trading strategies. When measuring the volatility, it is not sufficient to extrapolate from market data when your assets are highly correlated. Furthermore, liquidity concerns must be taken into account if access to emergency capital is not guaranteed. For this purpose, the author suggests the use of stress testing to predict the firm's actual needed equity capital, not by looking at past data, instead of considering possible scenarios for the portfolio in question. We need to take into account that returns are not symmetric and there can be significant tail risk in times when volatility spikes. Secondly, to reduce correlation between assets it is safer to diversify the portfolio so catastrophic events do no end up bankrupting the firm.