ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Quantitative Risk Management - Problem Set 3
Group - G02

Students :
Antoine Bedanian
Nicolas de Lestable
Maxime Richiardi
Ivan Schoenenberger

Professors :
Semyon Malamud
Amir Khalilzadeh

Academic year 2019 – 2020

Question 1

We download the daily stock price from WRDS from March 10, 2011 to March 10, 2016 for Intel, Yahoo and Microsoft.

a) b) We compute the log return for the 3 stocks. Then assuming that the log return are normally distributed we compute the rolling covariance and mean with a 2 years window for the period March 11, 2013 to March 10, 2016. Using these rolling covariance and mean we generate for each day $M = 1000$ values for risk factor.

c) Then we use the variance-covariance method to compute the VaR_α for $\alpha = 0.95$ and $\alpha = 0.99$. We first determine the λ using the fact that $n * S_{Yhaoo} = m * S_{Microsoft} = 100 * S_{Intel}$ on March 11, 2013 and we compute the VaR_α for each day and we compare it to the real loss function. We obtain the graph bellow:

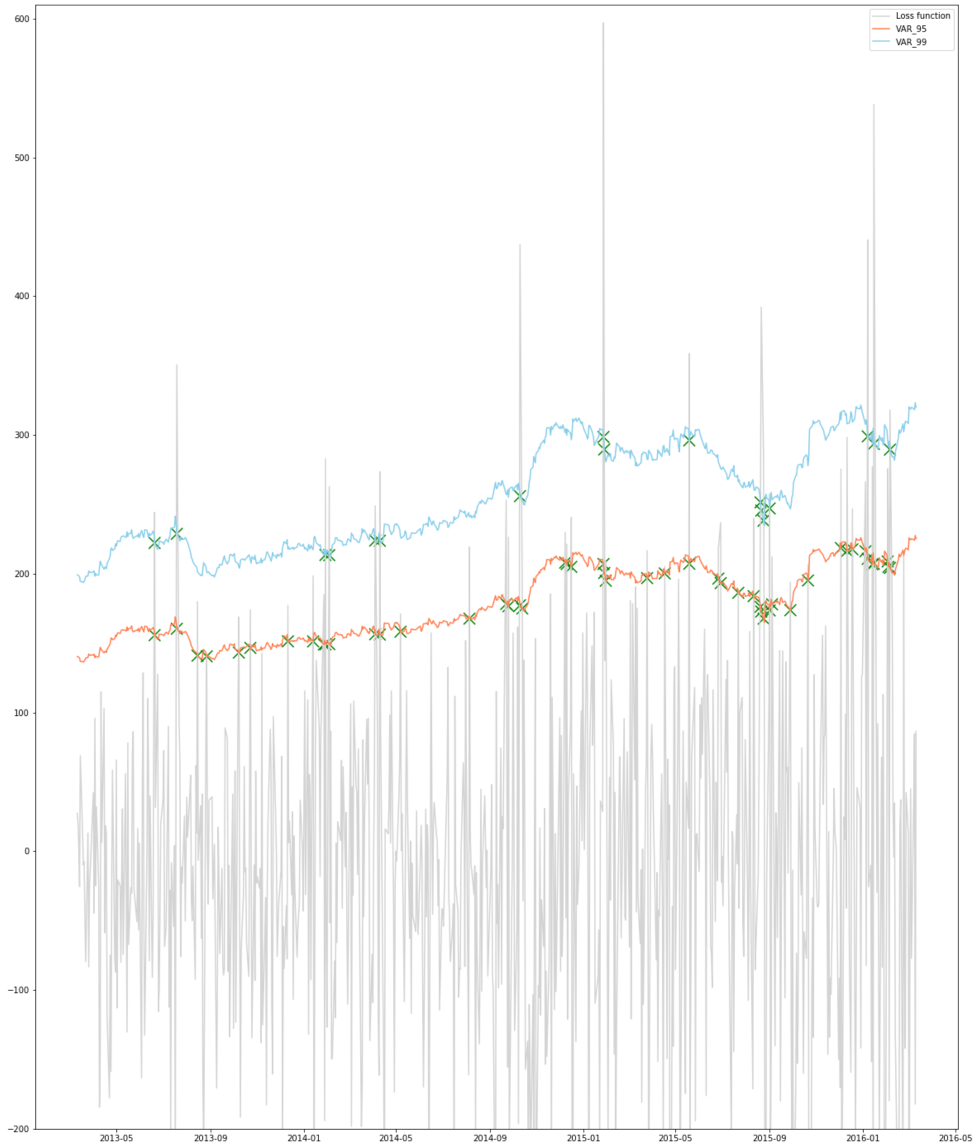


Figure 1: Daily VaR_α and losses from March 11, 2013 to March 10, 2016.

We obtain that the Loss function is above $VaR_{0.95}$ 49 times and above $VaR_{0.99}$ 17 times. For $\alpha = 0.95$, VaR_α was breached 49 times. The probability of 49 or more breaches in the course of 757 days should be: $1 - B(48, 757, 0.05) = 0.00203$ For $\alpha = 0.99$, VaR_α was breached 17 times. The probability of 17 or more breaches in the course of 757 days should be: $1 - B(16, 757, 0.01) = 0.04189$. The probability for $\alpha = 0.95$ is not so low but for $\alpha = 0.99$ it is unlikely to have more breaches above the $VaR_{0.99}$.

Question 2

1. The $VaR_{0.95}$ of the geometric distribution including 0 in the support is 4. In fact, the python geometric distribution does not allow 0 in support, we should also remove 1 to the number found to get the VaR required.
We can confirm the VaR by looking at its value for the range 0.9 to 0.99 on the figure below:

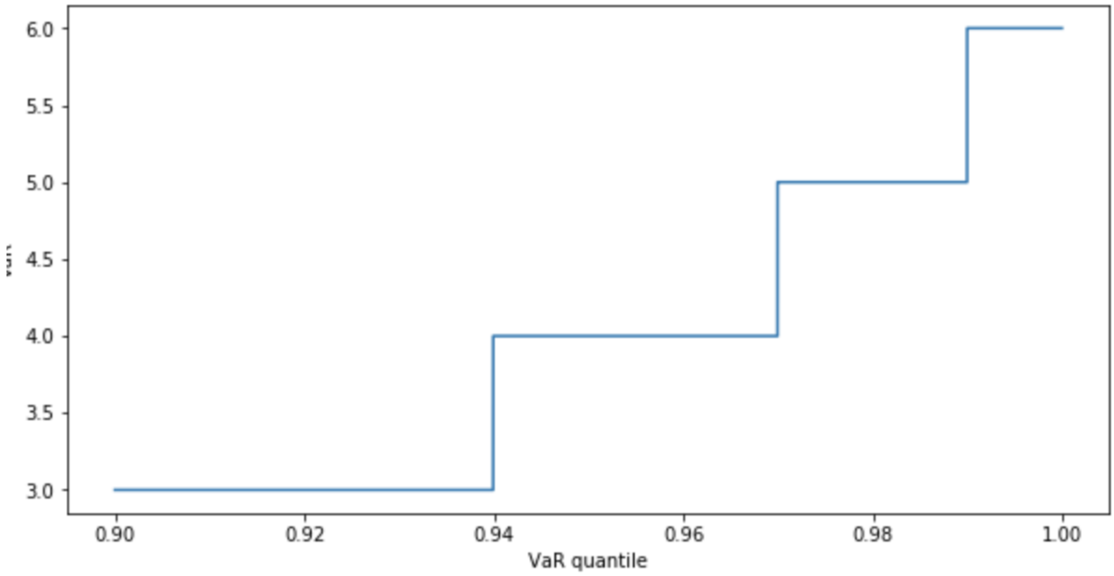


Figure 2: VaR of geometric distribution

2 If X and Y are 2 independent poisson distributions with parameters $\lambda_X = 1$ and $\lambda_Y = 2$, then from the demonstration above we have L = X + Y following a Poisson distribution with parameter $\lambda_L = 2 + 1 = 3$. It leads to the following VaR for all of the 3 distributions :

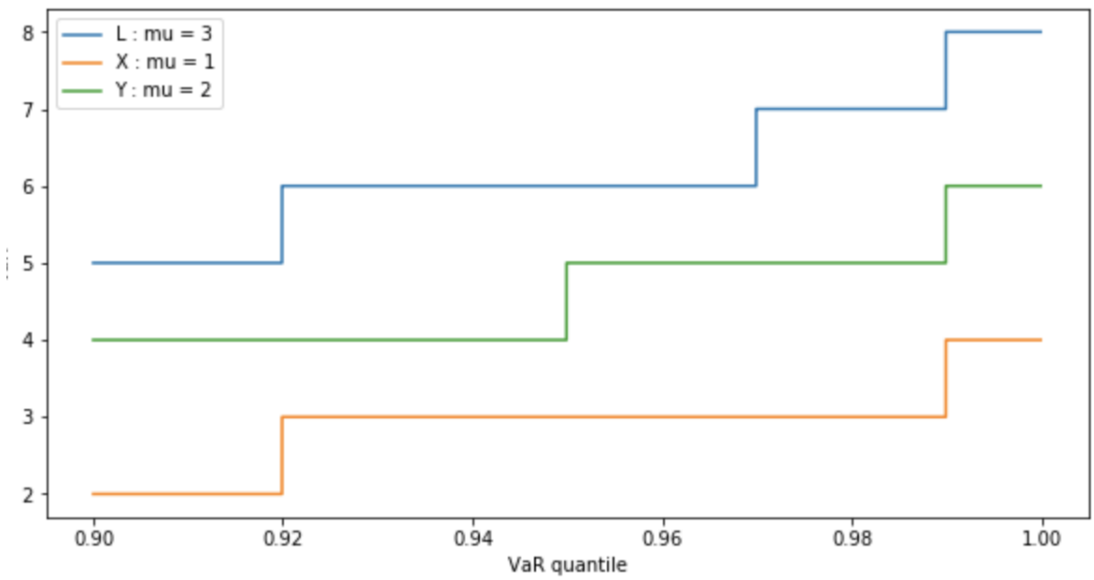


Figure 3: VaR of Poisson distributions with $\lambda = 1, 2, 3$

Indeed we now that the sum of two independent Poisson distributions are a Poisson distribution.

We have:

$$\begin{aligned}
P(L=k) &= \sum_{i=0}^{k} P(X+Y=k, X=i) \\
&= \sum_{i=0}^{k} P(Y=k-i, X=i) \\
&= \sum_{i=0}^{k} P(Y=k-i)P(X=i) \\
&= \sum_{i=0}^{k} e^{-\lambda_x} \frac{\lambda_x^{k-i}}{(k-i)!} e^{-\lambda_y} \frac{\lambda_y^i}{i!} \\
&= e^{-(\lambda_x+\lambda_y)} \frac{1}{k!} \sum_{i=0}^{k} \frac{k!}{i!(k-i)!} \lambda_x^{k-i} \lambda_y^i \\
&= \frac{(\lambda_x+\lambda_y)^k}{k!} \cdot e^{-(\lambda_x+\lambda_y)}
\end{aligned}$$

Therefore $L \sim \mathcal{P}(\lambda_x + \lambda_y)$

Question 3

Let's consider two random variables L_1 and L_2 which have four possible outcomes $i = 1, 2, 3, 4$ that occur with equal probability 1/4. We define the payoff of L_1 and L_2 like that:

$$L_1 = \begin{bmatrix} -1 \\ 0 \\ 1 \\ 2 \end{bmatrix} \quad L_2 = \begin{bmatrix} 0 \\ -1 \\ 1 \\ 2 \end{bmatrix} \quad L_1 + L_2 = \begin{bmatrix} -1 \\ -1 \\ 2 \\ 4 \end{bmatrix}$$

If we look at the value at risk of $\alpha = 0.75$ we have got:

$$VaR_{0.75}(L_1) = 0 \quad VaR_{0.75}(L_2) = 0 \quad VaR_{0.75}(L_1 + L_2) = 1$$

Indeed because we know that:

$$VaR_\alpha(X) = -\sup_x \{x \in \mathbb{R} : P(X < x) \leq 1 - \alpha\}$$

Therefore with this definition it's easy to compute the values at risk. So we find an example where the VaR is not subadditive, since we have:

$$VaR_\alpha (L_1 + L_2) \geq VaR_\alpha (L_1) + VaR_\alpha (L_2)$$

Question 4

1. The current value of a bond of company i is 1000CHF. Therefore $V_{i,0} = 1,000$. If the company default the value at time Δ is 0, but if it doesn't default the value is 1,050CHF. Since I_i is the indicator of default, we can say that $V_{i,\Delta} = 1,050(1 - I_i)$. Therefore the loss on a bond of company i is:

$$\begin{aligned}
L_i &= -(V_{i,\Delta} - V_{i,0}) \\
&= 1,000 - 1,050(1 - I_i) \\
&= 1,050 I_i - 50
\end{aligned}$$

2. The probability distribution of L_i is:

$$\begin{cases} \mathbb{P}\left(L_i = -50\right) & = 0.98 \\ \mathbb{P}\left(L_i = 1,000\right) & = 0.02 \end{cases}$$

3a. Since V_a is just 100 units of a single bond we have $L_a = 100 \times L_i = 10500 I_i - 500$. Threfore we have:

$$\begin{cases} \mathbb{P}\left(L_a = -5,000\right) & = 0.98 \\ \mathbb{P}\left(L_a = 100,000\right) & = 0.02 \end{cases}$$

Now we can compute the value at risk for L_a:

$$\begin{cases} VaR_{0.95}(L_a) & = -5,000 \\ VaR_{0.99}(L_a) & = 100,000 \end{cases}$$

b. V_b is the sum of n=100 bonds, therefore we have:

$$\begin{aligned} V_b &= \sum_{i=1}^{1} 00 L_i \\ &= \sum_{i=1}^{1} 00\left(1,050 I_i - 50\right) \\ &= 1,050 \sum_{i=1}^{1} 00 I_i - 5,000 \\ &= 1,050 \times N - 5,000 \end{aligned}$$

Where N has a binomial distribution since it count the number of bounds which default, and since I_i are $n = 100$ independent Bernoulli variables, N is a binomial wich parameters $p = 0.02$ and $n = 100$. Therefore we have:

$$\begin{cases} VaR_{0.95}(L_b) & = 1,050 \times VaR_{0.95}(N) - 5,000 = 1,050 \times 5 - 5,000 = 250 \\ VaR_{0.99}(L_b) & = 1,050 \times VaR_{0.99}(N) - 5,000 = 1,050 \times 6 - 5,000 = 1,300 \end{cases}$$

Since:

$$VaR_{0.95}(N) = \text{Bino}^{-1}(0.95, 100, 0.02) = 5$$

$$VaR_{0.99}(N) = \text{Bino}^{-1}(0.99, 100, 0.02) = 6$$

October 14, 2019

```
In [1]: import numpy as np
        import pandas as pd
        import random as random
        import matplotlib.pyplot as plt
        import scipy.stats as ss
        from datetime import datetime
        import wrds
        random.seed(420)

In [2]: #mdp:
        #goqhuB-1hafqe-dojvix
        db = wrds.Connection(wrds_username = 'antb95')
```

```
Enter your WRDS username [bedanian]:antb95
Enter your password:ůůůůůůůů
WRDS recommends setting up a .pgpass file.
You can find more info here:
https://www.postgresql.org/docs/9.5/static/libpq-pgpass.html.
Loading library list...
Done
```

```
In [3]: msft = db.raw_sql("select prc, date from crsp.dsf where permco in (8048.0) and date >=
        intc = db.raw_sql("select prc, date from crsp.dsf where permco in (2367.0) and date >=
        yhoo = db.raw_sql("select prc, date from crsp.dsf where permco in (14521.0) and date >

In [4]: msft['date'] = pd.to_datetime(msft['date'], format='%Y-%m-%d')
        intc['date'] = pd.to_datetime(intc['date'], format='%Y-%m-%d')
        yhoo['date'] = pd.to_datetime(yhoo['date'], format='%Y-%m-%d')

In [5]: msft_r = np.log(msft['prc']) - np.log(msft['prc'].shift(1))
        intc_r = np.log(intc['prc']) - np.log(intc['prc'].shift(1))
        yhoo_r = np.log(yhoo['prc']) - np.log(yhoo['prc'].shift(1))

In [6]: df_stock = pd.DataFrame()
        df_stock['date'] = msft['date']
        df_stock['msft'] = msft['prc']
        df_stock['intc'] = intc['prc']
        df_stock['yhoo'] = yhoo['prc']
```

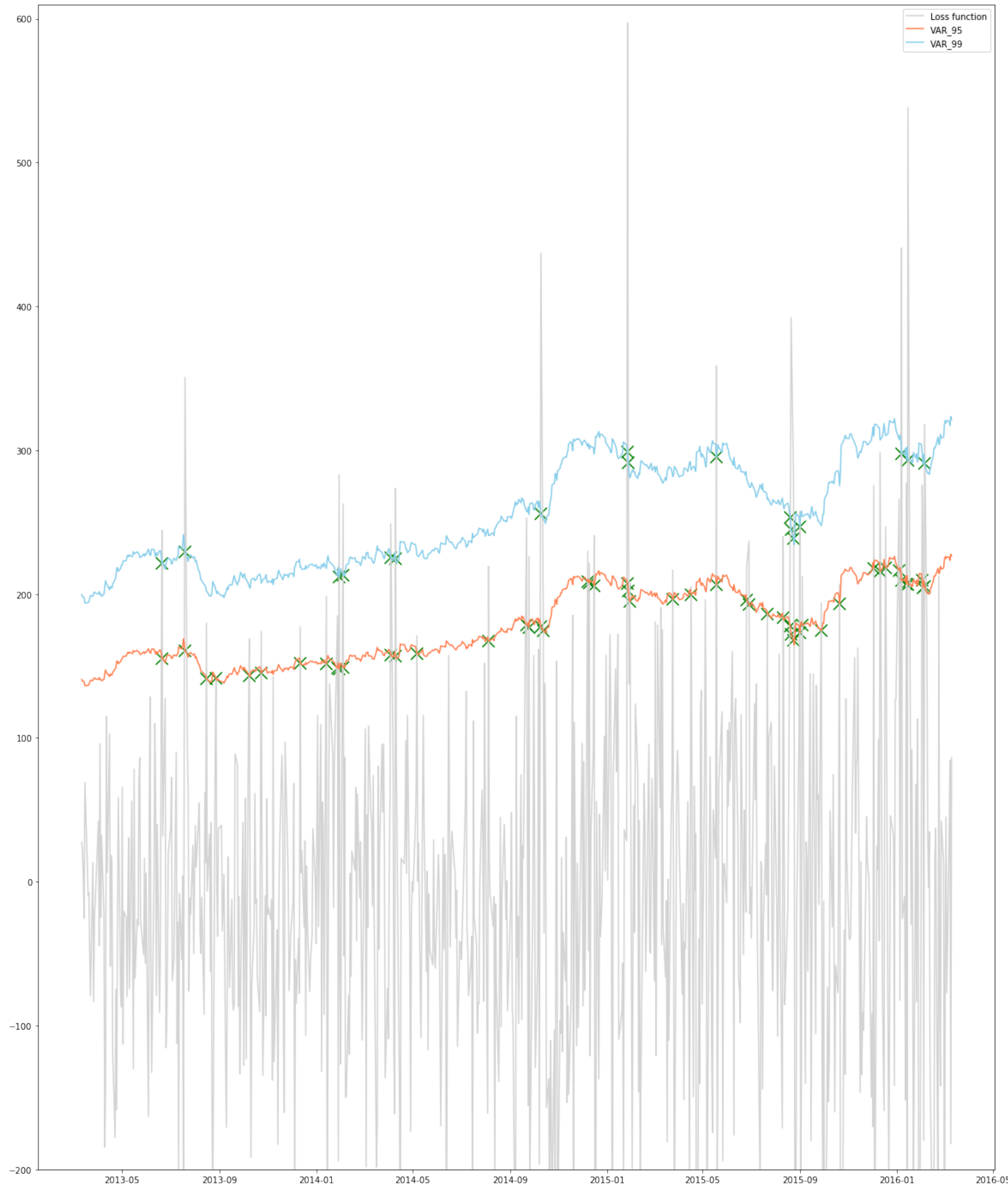
```
In [7]: df_return = pd.DataFrame()
        df_return['date'] = pd.to_datetime(msft['date'], format='%Y-%m-%d').copy()
        df_return['msft'] = msft_r
        df_return['intc'] = intc_r
        df_return['yhoo'] = yhoo_r
        df_return.dropna(inplace = True)

In [8]: cov_matrix = df_return.set_index('date').rolling(502).cov().dropna()
        mean = df_return.set_index('date').rolling(502).mean().dropna()

In [9]: #msft
        m = (intc[intc['date'] == '2013-03-11']['prc'].values[0]*100)/msft[msft['date'] == '20
        #yahoo
        n = (intc[intc['date'] == '2013-03-11']['prc'].values[0]*100)/yhoo[yhoo['date'] == '20
        #lambda
        lbda = np.array([m,100,n])

In [10]: M = 100000
         VAR_95 = []
         VAR_99 = []
         date = cov_matrix.index.get_level_values('date').drop_duplicates().values
         for i in date :
             temp_cov = cov_matrix.loc[i,:].values
             temp_mean = mean.loc[i,:].values
             rd_vec = np.random.multivariate_normal(temp_mean,temp_cov,M)
             #mean var method
             L = rd_vec * lbda * df_stock[df_stock['date'] == i][['msft','intc','yhoo']].value
             L = -1*np.sum(L, axis = 1)
             VAR_95 += [np.mean(L)+np.std(L)*ss.norm.ppf(0.95)]
             VAR_99 += [np.mean(L)+np.std(L)*ss.norm.ppf(0.99)]

In [11]: LOSS = []
         for i in date:
             temp = (np.exp(df_return[df_return['date'] == i][['msft','intc','yhoo']].values)-
             temp = -1*np.sum(temp, axis = 1)
             LOSS += [temp]

In [30]: up_95 = [0]*len(LOSS)
         up_99 = [0]*len(LOSS)
         val_95 = []
         val_99 = []
         for k in range(0,len(LOSS)):
             if LOSS[k] > VAR_95[k]:
                 up_95[k] = 1
                 val_95 += [[date[k],VAR_95[k]]]
             if LOSS[k] > VAR_99[k]:
                 up_99[k] = 1
                 val_99 += [[date[k],VAR_99[k]]]
         print('Number of days where the loss is above 95% of the Var :',np.sum(up_95))
```

```
        print('Number of days where the loss is above 99% of the Var :',np.sum(up_99))

        val_99 = np.array(val_99)
        val_95 = np.array(val_95)
```

```
Number of days where the loss is above 95% of the Var : 49
Number of days where the loss is above 99% of the Var : 17
```

```
In [13]: plt.figure(1, figsize = (20, 25))
         s = [200]*len(val_95)
         plt.scatter(val_95[:,0],val_95[:,1],color = 'g',marker = 'x', s=s)
         plt.scatter(val_99[:,0],val_99[:,1],color = 'g',marker = 'x', s=s)
         plt.plot(date,LOSS,color = 'lightgrey',label = 'Loss function')
         plt.plot(date,VAR_95,color = 'coral',label = 'VAR_95')
         plt.plot(date,VAR_99,color = 'skyblue',label = 'VAR_99')
         plt.ylim((-200, 610))
         plt.legend()
         plt.savefig('/Users/bedanian/Desktop/QRM/QRM/TD 3/Q1.png')
```

```
/Users/bedanian/anaconda3/lib/python3.7/site-packages/pandas/plotting/_converter.py:129: Future

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```




```
In [29]: res_99 = 1 - ss.binom.cdf(np.sum(up_99) - 1, len(date), 0.01)
         res_95 = 1 - ss.binom.cdf(np.sum(up_95) - 1, len(date), 0.05)
         print('The probability of 17 or more breaches in the course of 3 years is ','{0:.5f}'
         print('The probability of 49 or more breaches in the course of 3 years is ','{0:.5f}'
```

```
The probability of 17 or more breaches in the course of 3 years is  0.00203
The probability of 49 or more breaches in the course of 3 years is  0.04189
```

PS3_2

October 14, 2019

```
In [1]: import numpy as np
        import scipy.stats as stats
        import math
        import matplotlib.pyplot as plt
        import pandas as pd
```

0.1 Question 2.1

```
In [2]: #We include -1 in the formula below because the geometric function of python does not 
        print('The VaR 0.95 of a geometric distribution with p = 0.5 is : ' + str(stats.geom.pp
```

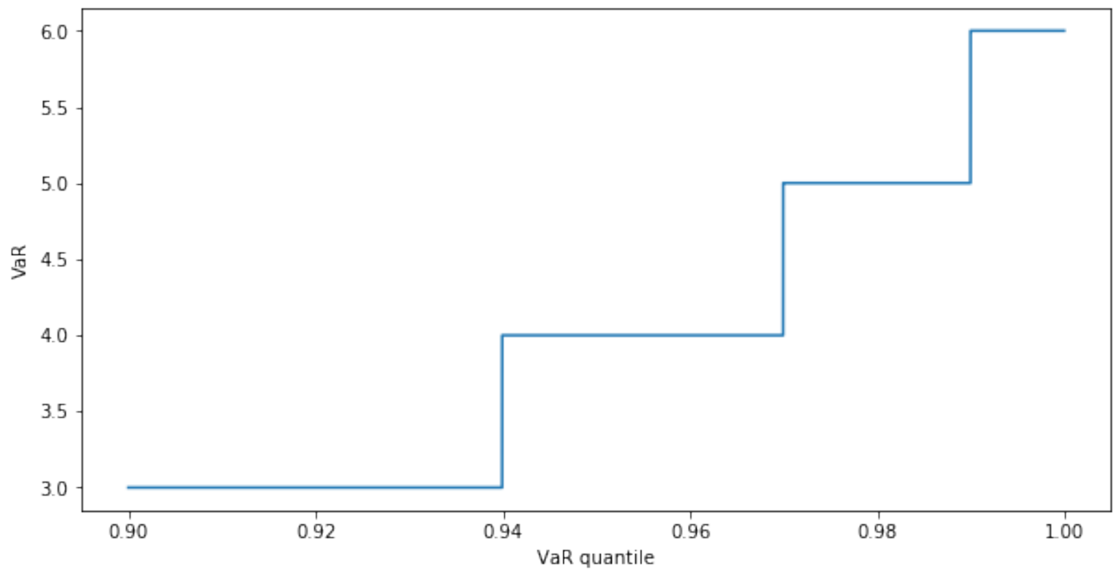
```
The VaR 0.95 of a geometric distribution with p = 0.5 is : 4.0
```

```
In [3]: alpha = np.arange(0.9,1,0.01)
        alpha1 = alpha + 0.0099
        alpha2 = alpha1 + 0.00001
        alpha = np.append(alpha,alpha1)
        alpha = np.append(alpha,alpha2)
        alpha = np.sort(alpha)
        alpha = alpha[:-1]
```

```
In [4]: df = pd.DataFrame(index = alpha , columns = ['VaR'])
        for i in range(len(alpha)):
            if (i+1) % 3 != 0:
                df.VaR[i] = stats.geom.ppf(np.trunc(alpha[i]*100)/100,0.5) - 1
            else:
                df.VaR[i] = stats.geom.ppf(round(alpha[i],2),0.5) - 1
```

```
In [5]: plt.figure(figsize = (10,5))
        plt.plot(df)
        plt.xlabel('VaR quantile')
        plt.ylabel('VaR')
```

```
Out[5]: Text(0, 0.5, 'VaR')
```



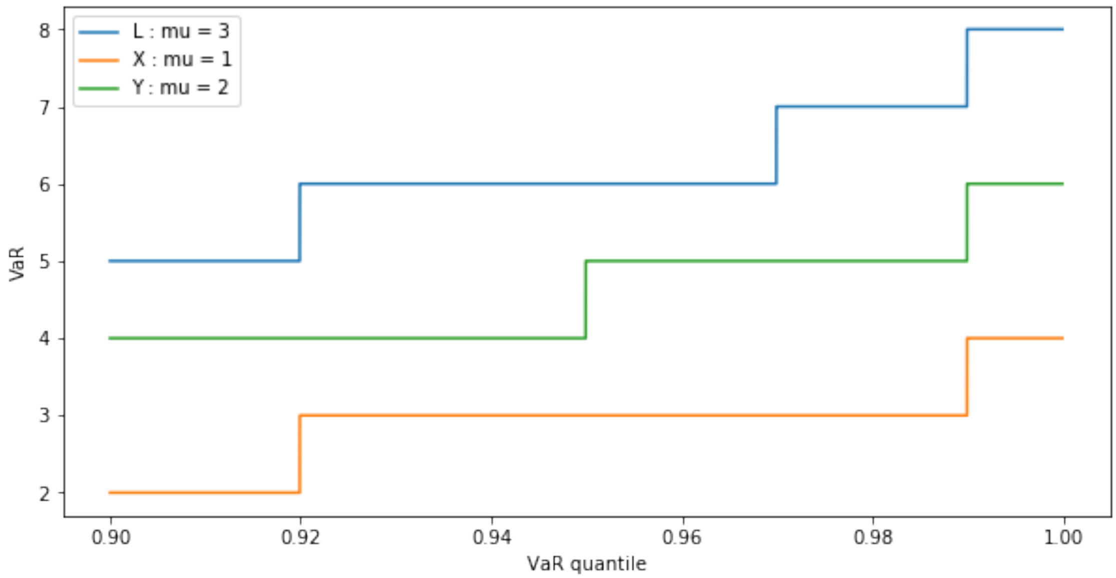
0.2 Question 2.2

In [6]:
```
def Poisson(mu,k):
    return np.exp(-mu)*np.power(mu, k)/ math.factorial(k)
```

In [7]:
```
df2 = pd.DataFrame(index = alpha,columns = ['X' , 'Y' , 'L'])
for i in range(len(alpha)):
    if (i+1) % 3 != 0:
        df2.at[alpha[i],'X'] = stats.poisson.ppf(np.trunc(alpha[i]*100)/100,1)
        df2.at[alpha[i],'Y'] = stats.poisson.ppf(np.trunc(alpha[i]*100)/100,2)
        df2.at[alpha[i],'L'] = stats.poisson.ppf(np.trunc(alpha[i]*100)/100,3)
    else:
        df2.at[alpha[i],'X'] = stats.poisson.ppf(round(alpha[i],2),1)
        df2.at[alpha[i],'Y'] = stats.poisson.ppf(round(alpha[i],2),2)
        df2.at[alpha[i],'L'] = stats.poisson.ppf(round(alpha[i],2),3)
```

In [8]:
```
plt.figure(figsize = (10,5))
plt.plot(alpha,df2.L,label = 'L : mu = 3')
plt.plot(df2.X,label = 'X : mu = 1')
plt.plot(df2.Y , label = 'Y : mu = 2')
plt.xlabel('VaR quantile')
plt.ylabel('VaR')
plt.legend()
```

Out[8]:
```
<matplotlib.legend.Legend at 0x124704d10>
```


L : mu = 3
X : mu = 1
Y : mu = 2
VaR
8
7
6
5
4
3
2
0.90
0.92
0.94
0.96
0.98
1.00
VaR quantile

```
In [ ]:
```