Alessandro Loury – Endrit Kastrati – Sinan Güven – Grégory Porchet

Empirical Methods in Finance – Assignment 1

I. The Market Index:

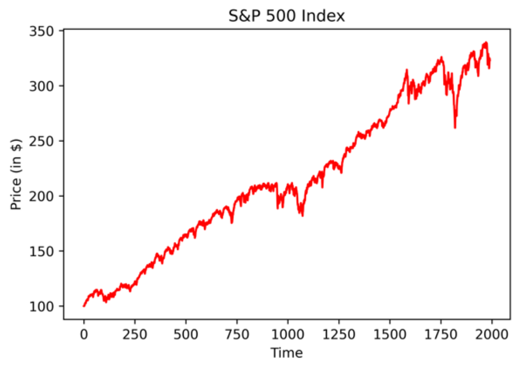
The S&P 500 (Standard and Poor) is a market index, the most well-known worldwide. It regroups 500 of the largest companies that are listed on the stock exchanges in the United States. The aim of this index is to measure the stock price performance of these companies. The S&P is a capitalization-weighted index, meaning that the components of the index are weighted according to the total market value of their outstanding shares. We can represent the value of the index at time t as follow with the weight attributed to firm i, its stock price and the number of shares outstanding:

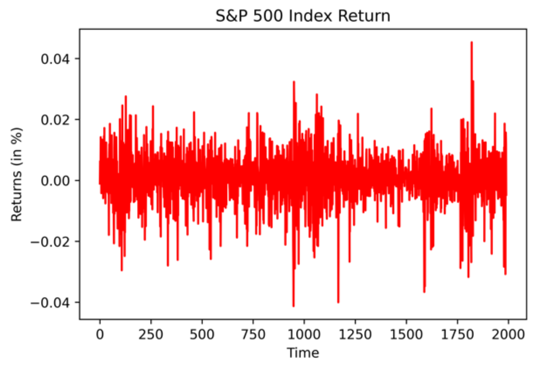
$$Index_t = \sum_{i=1}^{500} w_{i,t} * P_{i,t} \quad ; \quad w_{i,t} = \frac{Q_{i,t} * P_{i,t}}{\sum_{i=1}^{500} Q_{i,t} * P_{i,t}}$$

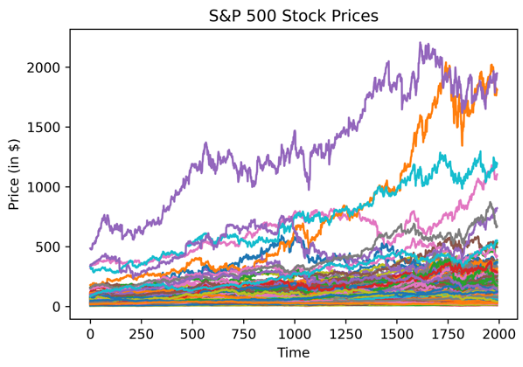
The conditions to enter this index are made by a selection committee. The main conditions are summarized below:

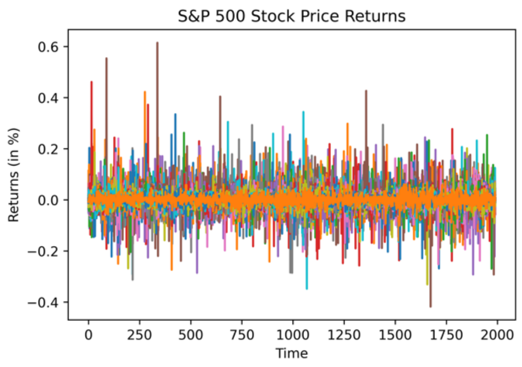
1)	*Market Cap ≥ $9.8 billion*
2)	*Ratio of annual dollar value traded to float-adjusted Market Cap ≥ 0.10*
3)	*Listed on an American stock exchange (NYSE, NASDAQ, …)*
4)	*Number of shares available for public trading ≥ 50%*
5)	*Sum of the earnings from the previous four quarters ≥ 0*
6)	*Earnings in the most recent quarter ≥ 0*

In this assignment, we downloaded a database including stock prices corresponding to 452 companies of the S&P 500 index. The daily evolution of the Index and the overall index daily return is given by the following graphs:









In the following, we computed the rolling 20-day (non-annualized) standard deviation of the returns for stock j :

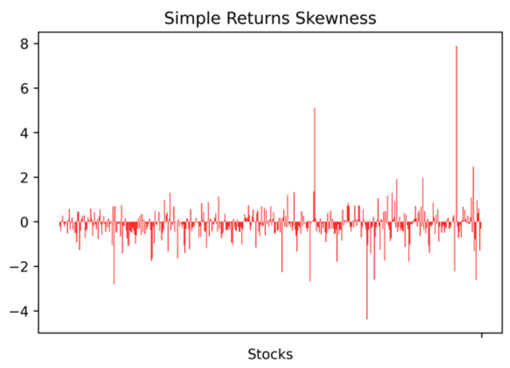
$$\hat{\sigma}_{j,t}^2 = \frac{1}{20}\sum_{i=0}^{20}\left(r_{j,t-i} - \hat{\mu}_j\right)^2 \quad ; \quad \tilde{r}_{j,t} = \frac{r_{j,t}}{\hat{\sigma}_{j,t}} \quad ; \quad \hat{\mu}_j = \frac{1}{T}\sum_{t=1}^{T} r_{j,t}$$

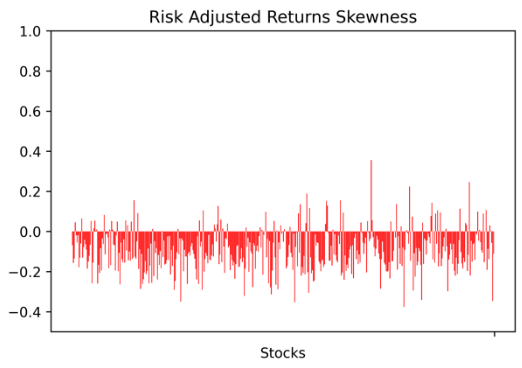
From this new volatilities we computed the risk-adjusted returns $\tilde{r}_{j,t}$ (see the code in appendix). Our goal in this report is to compare the statistical characteristics of the simple and risk-adjusted returns in order to run some tests on them.

II. Skewness:

At first, we have to remind what the Skewness means. It is a statistical measure that permits to analyze the asymmetry of the distribution (of returns in our case). A negative Skewness means that the curve of the distribution is right leaning, the left tail is longer so the main part of the distribution is concentrated on the right of the figure. For a positive Skewness, it is a left-leaning curve. In our case, a strong asymmetry would mean that extreme values (positive or negative) of returns are more likely to occur. The formula of the empirical Skewness for each asset i is given by:

$$\hat{S}_i = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{r_{i,t} - \hat{\mu}_i}{\hat{\sigma}_i}\right)^3$$



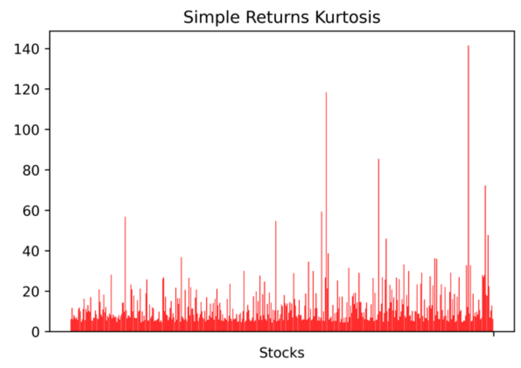


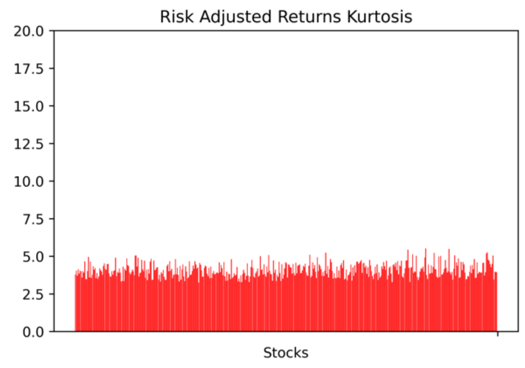
For a Normal Distribution, the Skewness is equal to zero. By writing some lines of codes, we found out that 287 stocks simple returns have a Skewness lower than 0 (≈ 63.49% of the assets of the sample) and 370 stocks risk-adjusted returns have a Skewness lower than 0 (≈ 81.85% of the sample). By looking at these graphs, globally the skewness for simple returns achieves more often higher extreme values than risk-adjusted returns. The Skewness for risk-adjusted returns are indeed closer to zero (mostly included in the $[-0.3, 0.2]$ interval), showing that the distribution asymmetry of these returns is close to a Normal one.

III. Kurtosis:

For this part, we computed the Kurtosis for both time series. It is another statistical measure that permits to analyze the tail-fatness of the distribution of the returns. For a Normal Law, the Kurtosis is equal to three. A kurtosis higher than this value means that the distribution will have fatter tails than the Normal distribution. The empirical Kurtosis is:

$$\hat{K}_i = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{r_{i,t} - \hat{\mu}_i}{\hat{\sigma}_i}\right)^4$$





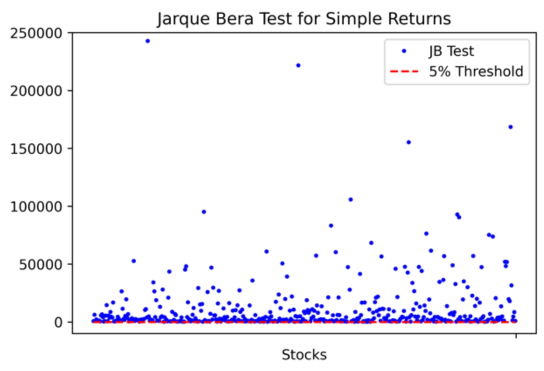
We found out that for each stocks' simple and risk-adjusted returns, the Kurtosis is always greater than 3 (100 % of the sample). As we can see from the graphs, the Kurtosis for simple returns is more likely to achieve higher values than the one for risk-adjusted returns. Given this large kurtosis, we can reveal the strong presence of fatter tails, which means more extreme positive and negative returns. Even if the kurtosis for these returns are all higher than 3, they are closer to this value than for the one for simple returns. This shows that the tail-fatness of the risk-adjusted returns is closer to a Normal one than simple returns. The values are mostly all inferior to 5 for risk-adjusted returns, while they are mostly around 20 for simple returns.

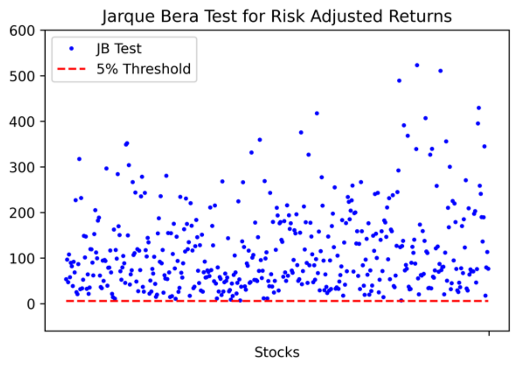
IV. Test for Normality:

One way to test for normality the data is the well-known Jarque-Bera Test. This is a type of test that permits to determine if the data distribution follows a Normal Distribution by using the standardized Skewness and Kurtosis. The T corresponds to the sample size. We run the test with a 5% threshold, meaning that the test is rejected if we have:

$$JB\ Test = T * \left[\frac{\hat{S}^2}{6} + \frac{(\hat{K}-3)^2}{24}\right] \geq \chi^2_{0.95}(2) = 5.991$$

Graphically, after running our code in order to compute the test, we obtained the following results:





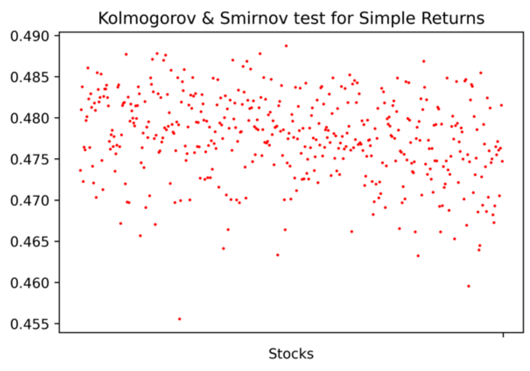
We found that the null hypothesis of normality is rejected for all stock returns (proportion of 100% for both time series). We conclude that using a Normal Distribution to fit the data is not a good way to model financial assets returns (simple or risk-adjusted) based on this method.

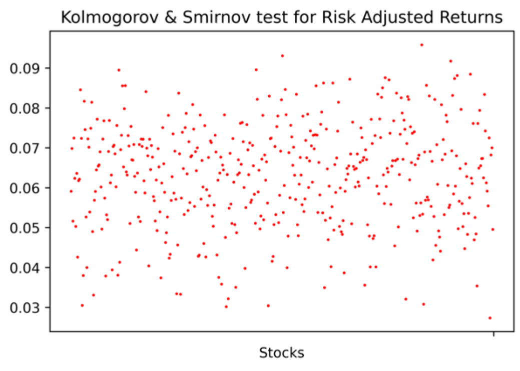
V. Kolmogorov and Smirnov Test:

We computed a Kolmogorov and Smirnov Test on Python for our time series. It is one of the simplest measures of the difference between two CDFs. We obtained the following results with the representation of the statistics of the test in the graphs. In order to compute the test, we have to follow different steps, explicated below. We have the assumed theoretical CDF $F^*(x\,;\theta)$ (here the normal distribution $\mathcal{N}(0,1)$) compared with the empirical CDF $G_T(x)$, where θ is the vector of parameters $[\mu, \sigma^2]$ in the case of the Normal distribution.

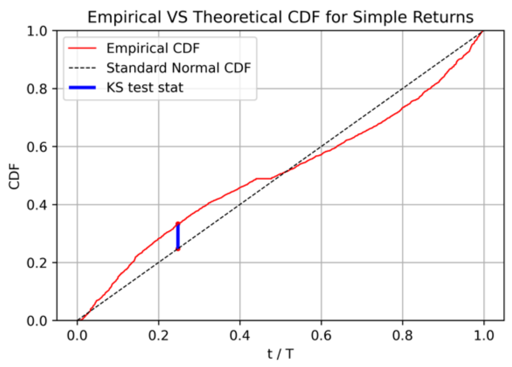
1)	*Sort data in increasing order with $r_1^* \leq \cdots \leq r_T^*$*
2)	*Denote the new sample $\{r_t^*\}_{t=1}^T$*
3)	*By construction, we obtained $G_T(r_t^*) = t/T$*
4)	*Evaluate the assumed CDF $F^*(r_t^*\,;\theta)$ for all values*
5)	*Finally, compute the following test statistic:*

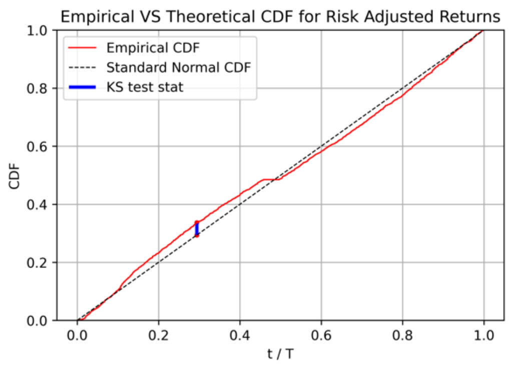
$$KS = \max_{t=1,\dots,T} |G_T(x) - F^*(x;\theta)| = \max_{t=1,\dots,T} \left|F^*(r_t) - \frac{t}{T}\right|$$





After running our code, the Kolmogorov Smirnov is rejected for the 452 stocks concerning the simple returns (100% of the sample). The test is rejected for 448 stocks concerning the risk-adjusted returns (99.12% of the sample). This means that the empirical CDF for all stocks' simple returns does not match the theoretical standardized Normal distribution. Concerning the risk-adjusted returns, it is not totally the case since we can observe few stocks for which their CDF are very close enough to the Normal one to not reject the test.
However, as we can see from the graphs, the values of the simple returns' statistics are all strictly higher than the risk-adjusted returns' values. The later are very close to the value of rejection (around $1.36/\sqrt{T} \approx 0.03$) meaning that the difference between the empirical CDF of risk-adjusted returns and the Normal one is not very big. As an example, we have the representation of the difference between the CDF of a stock return and the CDF of a Normal distribution:





VI. Ljung-Box Test:

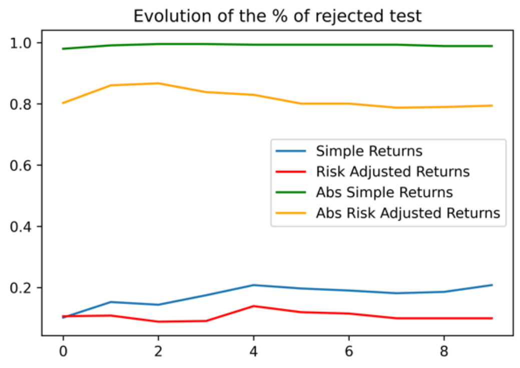
For this question, we computed the Ljung Box Test on Python. We wanted to test the null hypothesis that the first $p = 10$ first serial correlations are equal to zero. We have the hypothesis with the corresponding statistic under H_0 :

$$H_0 : \rho_1 = \cdots = \rho_p = 0 \quad ; \quad H_1 : \rho_j \neq 0 \; \exists j \in [1, \dots, p]$$

$$Q'_p = T(T+2)\sum_{j=1}^{p} \frac{1}{T-j}\hat{\rho}_j^2 \underbrace{\sim}_{H_0} \chi^2(p) \quad ; \quad \hat{\rho}_j = \frac{\sum_{t=j+1}^{T}(r_t - \hat{\mu})(r_{t-j} - \hat{\mu})}{\sum_{t=1}^{T}(r_t - \hat{\mu})^2} \; for \; 0 < j < T-1$$

With T corresponding to the sample size, $\hat{\rho}_j$ the sample autocorrelation at lag j and p the number of lags. The H_0 hypothesis is rejected if we have $Q'_p \geq \chi^2_{1-\alpha}(p)$. We obtain the proportion of rejected below (Abs = Absolute Returns):

Lag	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Simple	9.29%	14.38%	13.49%	16.37%	19.69%	18.8%	18.14%	17.26%	17.7%	19.9%
Risk-Adj	9.07%	9.51%	7.522%	7.74%	12.38%	10.39%	9.96%	8.4%	8.4%	8.4%
Abs Simp	98%	99%	99.6%	99.6%	99.34%	99.34%	99.34%	99.34%	98.89%	98.89%
Abs Risk-A	79.87%	85.84%	86.5%	83.63%	82.74%	79.86%	79.86%	78.54%	78.54%	78.54%



As we can see, the proportion of stocks for which the simple returns are autocorrelated is quite higher than the proportion of stocks for which the risk-adjusted returns are autocorrelated (almost the double for most of the lags). We can therefore say that the risk-adjusted returns are more uncorrelated from each other through time compared to simple returns. The reason behind the fact simple returns across time are more correlated to each other is surely

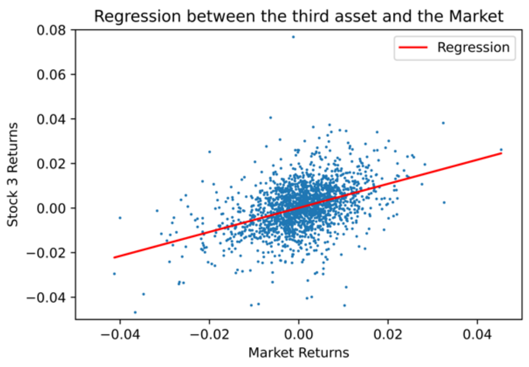
because the risk the stock returns carry are omitted while risk-adjusted returns allow us to represent the returns by taking their risk into account. It is better to get rid of the autocorrelation since the latter can be misleading for the model and can affect its forecasting accuracy.
We can see the same phenomenon for absolute returns: simple returns are more autocorrelated then risk-adjusted ones. Overall, the absolute returns are way more correlated to each other than their non-absolute counterparts because there are only positive returns.

VII. The Market Model Implementation:

For this part, we were asked to run the following regression, with r_t as the dependent variable and r_t^{MP} as the return of the market portfolio corresponding to the Index. This kind of regression model corresponds to the CAPM model, which gives an estimation of the relation between the market and the corresponding asset/stock.

$$r_t = \alpha + \beta * r_t^{MP} + \sigma\varepsilon_t \Leftrightarrow \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} + r_t^{MP} * \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} + \sigma \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix} \ ; \ \varepsilon \sim \mathcal{N}(0,1)\ iid$$

As an example, for the regression with the third stock, we obtain the following estimation with $\hat{\beta}_3 = 0.5398$:



So, we run the regression for the 452 stocks returns of the sample in order to obtain the estimation of the different betas. We obtained that 231 stock returns have a beta higher than 1 (≈ 51% of the sample), meaning that these stocks are more volatile than the market.

VIII. Investing Conclusions:

After studying the different characteristics of stocks returns of the S&P 500, using a normal distribution is globally not a good way to model assets returns. We observed considerable differences between simple returns and risk-adjusted returns concerning the different statistical tests that we have used along in our analysis. In both cases, we reject the normality assumption (Jarque-Bera test, Kolmogorov-Smirnov test) for the majority of the stocks. However, by using the 20-days rolling volatility and therefore accounting for the degree of risk, we saw that the difference between the Normal distribution and the empirical distribution of risk-adjusted returns is less important. In consequence, it is much more suitable to use risk-adjusted returns for financial analysis in order to perform investing decisions. By doing so, we would be able to use the Normal Distribution to model these returns and to perform forecasts.

Furthermore, apart from the comparison between simple and risk-adjusted returns, we learn important conclusions from the regression of the market model using simple returns. We saw that more than the half of the index has stocks with a beta higher than one, implying that they are more volatile than the market. These stocks would be a perfect investment for risk seeking investors. Stocks with a beta lower than one, and so less volatile than the market, would be perfect for risk averse investors.
An investor should be aware of these particularities and these different metrics to take optimal decisions. Consistent analyses must be performed in time to see the evolutions of these metrics. Different time windows should also be taken into consideration to get an overview of micro and macro trends depending on multiple factors (political and legal framework, investment preferences, technology, …).

IX. Appendix:

All the code that we used for this assignment is given below:

```
@author : The Quants Managers => Alessandro, Sinan, Endrit & Grégory
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sci
import statsmodels.api as sm
from scipy.stats import skew
from scipy.stats import kurtosis
from scipy.stats import norm
import random


#Downloading the data


filename = "Data_assignment.xlsx"

xls = pd.ExcelFile(filename)

S_P_500 = pd.read_excel(xls, 'Data_export')

S_P_500_2 = S_P_500.iloc[:, 0:452]

Index = S_P_500.iloc[:, 452]

Index_formated = np.array(Index)

Prices = np.array(S_P_500_2)

num_lines = np.size(Prices, 0)

Returns = np.divide(Prices[1:(num_lines), :], Prices[0:(num_lines - 1)]) - 1

Index_Returns = np.divide(Index_formated[1:(num_lines)], Index_formated[0:(num_lines - 1)]) - 1

Expectation_Index = np.mean(Index_Returns)

labels = list(S_P_500_2)


plt.figure(dpi=600)
plt.plot(Prices)
plt.xlabel('Time')
plt.ylabel('Price (in $)')
plt.title('S&P 500 Stock Prices')

plt.figure(dpi=600)
plt.plot(Index_formated, 'r')
plt.xlabel('Time')
plt.ylabel('Price (in $)')
plt.title('S&P 500 Index')

plt.figure(dpi=600)
plt.plot(Returns)
plt.xlabel('Time')
plt.ylabel('Returns (in %)')
plt.title('S&P 500 Stock Price Returns')

plt.figure(dpi=600)
plt.plot(Index_Returns, 'r')
plt.xlabel('Time')
plt.ylabel('Returns (in %)')
plt.title('S&P 500 Index Return')


#Computation of Rolling Volatilities


MU = np.mean(Returns, 0)
MU = np.array(MU)
```

```
def Rolling_Vol(y):
    Rol_Var_y = (1/20) * ((Returns[y - 0, :] - MU)**2 + (Returns[y - 1, :] - MU)**2
                          + (Returns[y - 2, :] - MU)**2 + (Returns[y - 3, :] - MU)**2
                          + (Returns[y - 4, :] - MU)**2 + (Returns[y - 5, :] - MU)**2
                          + (Returns[y - 6, :] - MU)**2 + (Returns[y - 7, :] - MU)**2
                          + (Returns[y - 8, :] - MU)**2 + (Returns[y - 9, :] - MU)**2
                          + (Returns[y - 10, :] - MU)**2 + (Returns[y - 11, :] - MU)**2
                          + (Returns[y - 12, :] - MU)**2 + (Returns[y - 13, :] - MU)**2
                          + (Returns[y - 14, :] - MU)**2 + (Returns[y - 15, :] - MU)**2
                          + (Returns[y - 16, :] - MU)**2 + (Returns[y - 17, :] - MU)**2
                          + (Returns[y - 18, :] - MU)**2 + (Returns[y - 19, :] - MU)**2
                          + (Returns[y - 20, :] - MU)**2)
    return Rol_Var_y


ZZ_Top = Rolling_Vol(20)

m = []

for y in range(20, num_lines - 1):
    LED = Rolling_Vol(y)
    print(LED)
    m.append(LED)

M = np.array(m)

M_std_dev = np.power(M, 0.5)



#Computation of Risk Adjusted Returns



Risk_adjusted_Returns = np.divide(Returns[20:, :], M_std_dev)
plt.plot(Risk_adjusted_Returns)
plt.title('S&P 500 Stock Risk Adjusted Returns')

MU_Risk = np.mean(Risk_adjusted_Returns, 0)



#Computation of Volatility



plt.figure(dpi=600)
plt.bar(labels, MU)
plt.xticks('')

Squared_Returns = np.power(Returns, 2)
Variance_Returns = np.mean(Squared_Returns, 0) - np.power(np.mean(Returns, 0), 2)

Volatility = np.power(Variance_Returns, 0.5)
Annualized_Volatility = Volatility * np.power(52, 0.5)

plt.figure(dpi=600)
plt.bar(labels, Annualized_Volatility)
plt.xticks(size = 0)
plt.xticks('')


Squared_Risk_Returns = np.power(Risk_adjusted_Returns, 2)
Variance_Risk_Returns = np.mean(Squared_Risk_Returns, 0) - np.power(np.mean(Risk_adjusted_Returns, 0), 2)
Volatility_Risk_Returns = np.power(Variance_Risk_Returns, 0.5)



#Computation of Skewness



Skewness = skew(Returns, 0)

plt.figure(dpi = 600)
plt.bar(labels, Skewness, color = 'red')
plt.xlabel('Stocks')
plt.xticks(rotation = 90, size = 0)
plt.xticks('')
plt.title('Simple Returns Skewness')

Skewness_Lower = Skewness[Skewness < 0]
len(Skewness_Lower)


Skewness_Risk = skew(Risk_adjusted_Returns, 0)

Skewness_Risk_Lower = Skewness_Risk[Skewness_Risk < 0]
len(Skewness_Risk_Lower)
```

```
plt.figure(dpi = 600)
plt.bar(labels, Skewness_Risk, color = 'red')
plt.xlabel('Stocks')
plt.xticks(rotation = 90, size = 0)
plt.xticks('')
plt.ylim([-0.5, 1])
plt.title('Risk Adjusted Returns Skewness')



#Computation of the Kurtosis



Kurtosis = kurtosis(Returns, 0, fisher = False)

plt.figure(dpi = 600)
plt.bar(labels, Kurtosis, color = 'red')
plt.xlabel('Stocks')
plt.xticks(rotation = 90, size = 0)
plt.xticks('')
plt.title('Simple Returns Kurtosis')

Kurtosis_Higher = Kurtosis[Kurtosis > 3]
len(Kurtosis_Higher)


Kurtosis_Risk = kurtosis(Risk_adjusted_Returns, 0, fisher=False)

Kurtosis_Risk_Higher = Kurtosis_Risk[Kurtosis_Risk > 3]
len(Kurtosis_Risk_Higher)

plt.figure(dpi = 600)
plt.bar(labels, Kurtosis_Risk, color = 'red')
plt.xlabel('Stocks')
plt.xticks(rotation = 90, size = 0)
plt.xticks('')
plt.ylim([0, 20])
plt.title('Risk Adjusted Returns Kurtosis')



#Computation of the Jarque Bera Test



test_JB = np.size(Returns, 0) * (np.power(Skewness, 2)/6 + np.power(Kurtosis - 3, 2)/24)

plt.figure(dpi = 600)
plt.plot(labels, test_JB, 'bo', labels, test_JB * 0 + 5.991, 'r--', markersize = 2)
plt.xlabel('Stocks')
plt.xticks(rotation = 90, size  = 0)
plt.legend(["JB Test", "5% Threshold"])
plt.xticks('')
plt.ylim([-10000, 250000])
plt.title('Jarque Bera Test for Simple Returns')

Rejected = test_JB[test_JB > 5.991]
len(Rejected)



test_JB_Risk = np.size(Risk_adjusted_Returns, 0) * (np.power(Skewness_Risk, 2)/6 + np.power(Kurtosis_Risk - 3, 
2)/24)

plt.figure(dpi = 600)
plt.plot(labels, test_JB_Risk, 'bo', labels, test_JB_Risk * 0 + 5.991, 'r--', markersize = 2)
plt.xlabel('Stocks')
plt.legend(["JB Test", "5% Threshold"])
plt.xticks('')
plt.ylim([-60, 600])
plt.title('Jarque Bera Test for Risk Adjusted Returns')

Rejected_Risk = test_JB_Risk[test_JB_Risk > 5.991]
len(Rejected_Risk)



#Compute the Kolmogorov and Smirnov Test



from scipy import stats
stats.kstest(Returns[:, 0], 'norm')


test = [] ;
```

```
for i in range(0, 452) :

    temp = stats.kstest(Returns[:, i], 'norm');
    temp = temp.statistic
    test += [temp] ;

plt.figure(dpi=600)
plt.plot(labels, test, 'bo', color = 'red', markersize = 1)
plt.xlabel('Stocks')
plt.xticks('')
plt.title('Kolmogorov & Smirnov test for Simple Returns')

test_array = np.array(test)

Rejected_KS = test_array[test_array > (1.36 / np.sqrt(1990))]
len(Rejected_KS)


test2 = [] ;

for i in range(0, 452) :

    temp2 = stats.kstest(Risk_adjusted_Returns[:, i], 'norm');
    temp2 = temp2.statistic
    test2 += [temp2] ;

plt.figure(dpi=600)
plt.plot(labels, test2, 'bo', color = 'red', markersize = 1)
plt.xlabel('Stocks')
plt.xticks('')
plt.title('Kolmogorov & Smirnov test for Risk Adjusted Returns')

test_2_array = np.array(test2)

Rejected_KS_Risk = test_2_array[test_2_array > (1.36 / np.sqrt(1970))]
len(Rejected_KS_Risk)



#Graphs of the Distance between the Empirical CDF and the Theoretical CDF



ecdfs = np.arange(1990, dtype=float)/1990
GS = []
x = random.randint(0,452)
KS = Returns[:,x]
maxi = 0
index = 0
SOR = np.sort(KS)
for t in range(len(SOR)) :
    KS[t] = norm.cdf(SOR[t], MU[x],(Variance_Returns[x]**0.5))
    GS.append(abs(norm.cdf(SOR[t], MU[x],(Variance_Returns[x]**0.5)) - (t/len(SOR))))
    if GS[t] > maxi :
        index = t
        maxi = GS[t]
    else :
        continue
plt.figure(dpi=600)
plt.plot(ecdfs,KS,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs,ecdfs,color='k',lw=0.8,linestyle='dashed',label="Standard Normal CDF")
plt.scatter(index/1990,ecdfs[index],s=7,color='r')
plt.scatter(index/1990,KS[index],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index/1990], ecdfs[index], KS[index], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF for Simple Returns')


ecdfs = np.arange(1970, dtype=float)/1970
GS = []
x = random.randint(0,452)
KS = Risk_adjusted_Returns[:,x]
maxi = 0
index = 0
SOR = np.sort(KS)
for t in range(len(SOR)) :
    KS[t] = norm.cdf(SOR[t], MU_Risk[x],(Variance_Risk_Returns[x]**0.5))
    GS.append(abs(norm.cdf(SOR[t], MU_Risk[x],(Variance_Risk_Returns[x]**0.5)) - (t/len(SOR))))
    if GS[t] > maxi :
        index = t
        maxi = GS[t]
    else :
        continue
plt.figure(dpi=600)
plt.plot(ecdfs,KS,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs,ecdfs,color='k',lw=0.8,linestyle='dashed',label="Standard Normal CDF")
plt.scatter(index/1970,ecdfs[index],s=7,color='r')
plt.scatter(index/1970,KS[index],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index/1970], ecdfs[index], KS[index], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF for Risk Adjusted Returns')



#Compute the Ljung Box Test



def LjungBoxQ(x,lag):
    T=len(x);
    Formatedx=pd.Series(x);
    output=[];
    critical_values=[];
    for i_lag in range(1,lag+1):
        temp=0;
        for i in range(1,i_lag+1):
            temp+=1/(T-i+1)*np.power(Formatedx.autocorr(i),2)
        output+=[temp*T*(T+2)];
        critical_values+=[sci.chi2.isf(.05,i_lag)];
    return pd.DataFrame([output, critical_values])

result_Jung = [];
for i in range(0,452):
    temp = np.transpose(LjungBoxQ(Returns[:,i],10));
    if i == 451:
        result_Jung += [temp.iloc[:,0]];
        result_Jung += [temp.iloc[:,1]];
    else:
        result_Jung+=[temp.iloc[:,0]];
titles=list(S_P_500_2);
titles.append('Critical Value');
result_Jung=pd.DataFrame(np.transpose(result_Jung), list(range(0,10)),titles)


proportion_Ljung=[]
for i in range (10):
    Ljung_correlated=[]
    for j in range (452):
        if result_Jung.iloc[i,j]>result_Jung.iloc[i,452]:
            Ljung_correlated.append(result_Jung.iloc[i,j])
    proportion_Ljung.append(np.size(Ljung_correlated)/452)




result_Jung_Risk = [];
for i in range(0,452):
    temp_Risk = np.transpose(LjungBoxQ(Risk_adjusted_Returns[:,i],10));
    if i == 451:
        result_Jung_Risk += [temp_Risk.iloc[:,0]];
        result_Jung_Risk += [temp_Risk.iloc[:,1]];
    else:
        result_Jung_Risk += [temp_Risk.iloc[:,0]];
titles=list(S_P_500_2);
titles.append('Critical Value');
result_Jung_Risk = pd.DataFrame(np.transpose(result_Jung_Risk), list(range(0,10)),titles)


proportion_Ljung_Risk = []
for i in range (10):
    Ljung_correlated_Risk = []
    for j in range (452):
        if result_Jung_Risk.iloc[i,j] > result_Jung_Risk.iloc[i,452]:
            Ljung_correlated_Risk.append(result_Jung_Risk.iloc[i,j])
    proportion_Ljung_Risk.append(np.size(Ljung_correlated_Risk)/452)




Absolute_Returns = abs(Returns)
plt.plot(Absolute_Returns)

Result_Jung_Abs = [];
for i in range(0,452):
    temp_Abs = np.transpose(LjungBoxQ(Absolute_Returns[:,i],10));
    if i == 451:
        Result_Jung_Abs += [temp_Abs.iloc[:,0]];
        Result_Jung_Abs += [temp_Abs.iloc[:,1]];
    else:
        Result_Jung_Abs += [temp_Abs.iloc[:,0]];
titles=list(S_P_500_2);
titles.append('Critical Value');
```

```
Result_Jung_Abs = pd.DataFrame(np.transpose(Result_Jung_Abs), list(range(0,10)),titles)


proportion_Ljung_Abs = []
for i in range (10):
    Ljung_correlated_Abs = []
    for j in range (452):
        if Result_Jung_Abs.iloc[i,j] > Result_Jung_Abs.iloc[i,452]:
            Ljung_correlated_Abs.append(Result_Jung_Abs.iloc[i,j])
    proportion_Ljung_Abs.append(np.size(Ljung_correlated_Abs)/452)




Absolute_Risk_Returns = abs(Risk_adjusted_Returns)
plt.plot(Absolute_Risk_Returns)

Result_Jung_Risk_Abs = [];
for i in range(0,452):
    temp_Risk_Abs = np.transpose(LjungBoxQ(Absolute_Risk_Returns[:,i],10));
    if i == 451:
        Result_Jung_Risk_Abs += [temp_Risk_Abs.iloc[:,0]];
        Result_Jung_Risk_Abs += [temp_Risk_Abs.iloc[:,1]];
    else:
        Result_Jung_Risk_Abs += [temp_Risk_Abs.iloc[:,0]];
titles=list(S_P_500_2);
titles.append('Critical Value');
Result_Jung_Risk_Abs = pd.DataFrame(np.transpose(Result_Jung_Risk_Abs), list(range(0,10)),titles)


proportion_Ljung_Risk_Abs = []
for i in range (10):
    Ljung_correlated_Risk_Abs = []
    for j in range (452):
        if Result_Jung_Risk_Abs.iloc[i,j] > Result_Jung_Risk_Abs.iloc[i,452]:
            Ljung_correlated_Risk_Abs.append(Result_Jung_Risk_Abs.iloc[i,j])
    proportion_Ljung_Risk_Abs.append(np.size(Ljung_correlated_Risk_Abs)/452)


plt.figure(dpi=600)
plt.plot(proportion_Ljung)
plt.plot(proportion_Ljung_Risk, color = 'red')
plt.plot(proportion_Ljung_Abs, color = 'green')
plt.plot(proportion_Ljung_Risk_Abs, color = 'orange')
plt.legend(['Simple Returns', 'Risk Adjusted Returns', 'Abs Simple Returns', 'Abs Risk Adjusted Returns'])
plt.title('Evolution of the % of rejected test')



#Regression between the Market Portfolio and the Stock Returns



X1 = sm.add_constant(Index_Returns)

def OLS(y) :
    Model_CAPM = sm.OLS(y, X1)
    results = Model_CAPM.fit()
    return results.params


print(OLS(Returns[:, 2]))

Parameters = np.apply_along_axis(OLS, 0, Returns)

Beta = Parameters[1, :]
Beta_Higher = Beta[Beta > 1]
len(Beta_Higher)

Model_CAPM_2 = sm.OLS(Returns[:, 2], X1)
p = Model_CAPM_2.fit().params

print(Model_CAPM_2.fit().summary())

plt.figure(dpi=1000)
plt.scatter(X1[:, 1], Returns[:, 2], s = 1)
plt.plot(X1, p[0] + p[1] * X1, color = 'red')
plt.axis([-0.05, 0.05, -0.05, 0.08])
plt.xlabel('Market Returns')
plt.ylabel('Stock 3 Returns')
plt.title('Regression between the third asset and the Market')
plt.legend(["Regression"])
```