LEONARD DE VINCI GRADUATE SCHOOL OF ENGINEERING

MARKET RISK REPORT

Market Risk

Students :
Enzo DELGADO
Raphael DEMARE

Contents

1 Introduction and essential librairies

This project is the conclusion of the market risk course, allowing us to dive into the implementation of concepts seen in lectures and further explored in directed studies.

The code used for each question can be found in the "Appendix" section at the end of the report.

Python's `pandas` library plays a critical role in data preprocessin, enable efficient data manipulation and cleaning.
Once the data is prepared, the next step is to conduct our computation to obtain risk measures. Here, we leverage `numpy` for numerical computations and `scipy.stats` for statistical functions.
For visualization, `matplotlib.pyplot` is used extensively to create insightful charts and graphs.

2 Question A

From the time series of the daily prices of the stock Natixis between January 2015 and December 2016, provided with TD1, estimate a `historical VaR` on price returns at a one-day horizon for a given probability level (this probability is a parameter which must be changed easily). You must base your VaR on a `non-parametric distribution` (biweight Kernel, that is $K(u) = \frac{15}{16}(1-u^2)^2\mathbf{1}_{|u|\leq 1}$).

In this question, the 'datatd1td2.csv' file used is join in the mail. We firstly computed the returns using for each date the next formula: (Pj-Pi)/Pi.

We did it with the biweight Kernel density, which is a non-parametric method for estimating the probability density function (PDF) of a random variable from a sample of observations. It is a type of kernel density estimate, which involves weighting the observations in the sample using a kernel function and summing the weighted observations to estimate the density.

Value at Risk To find the Value at Risk (VaR) at a given confidence level (95% or 99%), we must identify the quantile of the estimated distribution corresponding to the chosen confidence level. This quantile is the point on the distribution curve where there is a (100 − confidence level)% probability that the risk factor will exceed this value.

Applying this method, we finally obtain a VaR of −3.88% at 95% confidence level. This indicates that, for a portfolio valued at €1000, there is a 5% probability of incurring a loss greater than €38.88.

VaR Exceedance Analysis To evaluate the effectiveness of our previously calculated non-parametric Value at Risk (VaR), we conducted a backtesting exercise on the data from a subsequent period, January 2017 to December 2018. This procedure is aimed at answering the following question: "Which proportion of price returns exceed the VaR

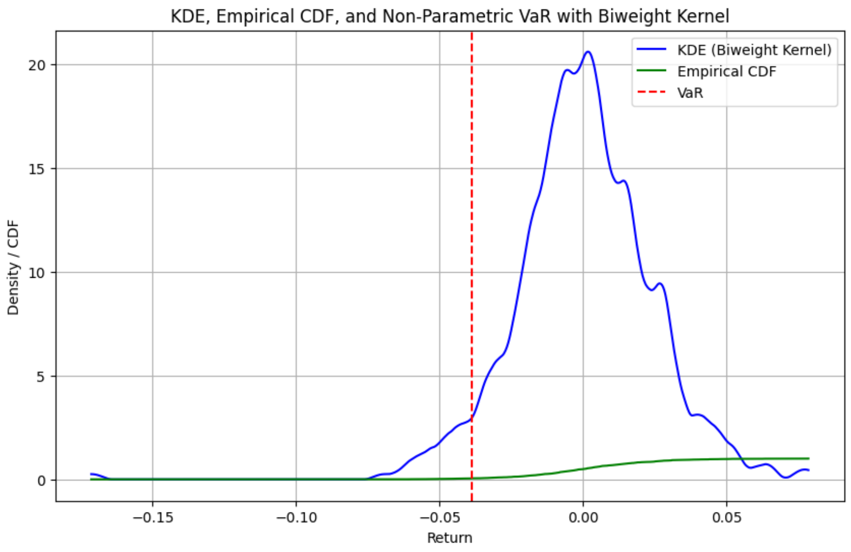

Figure 1: Kernel Density Estimation, Empirical CDF, and Non-Parametric VaR with Biweight Kernel

threshold defined previously, and does this validate the choice of our non-parametric VaR?"

1. We filtered the data to the specific time frame of interest.
2. We then computed the daily returns for this subset of data.
3. These returns were compared against the VaR threshold determined from our initial analysis.
4. Finally, we validated the VaR model by checking if the observed exceedance proportion was less than or equal to the expected exceedance rate, which is the complement of the confidence level (i.e., $1 - 0.95 = 0.05$ or 5%).

The results of this backtesting procedure yielded an exceedance proportion of approximately 1.57%, with the output $(0.01568627450980392, \text{True})$. Since this observed proportion is lower than the expected 5%, the validation check is True, confirming the reliability of our non-parametric VaR within the tested period. This indicates that the non-parametric VaR model we have chosen is conservative and effective for the level of confidence intended.

3 Question B

In this question, the ‘datatd1td2.csv‘ file used is join in the mail.

Expected Shortfall Given the Value at Risk (VaR) previously calculated as -0.0388 (or 3.88%), we proceed to compute the Expected Shortfall (ES). It is particularly useful in evaluating the tail risk of a distribution, as it provides the average loss in the worst 5% of cases, given a 95% confidence level.

To compute the ES, we first isolate all the returns that are less than the VaR (tail losses). These are the worst losses that exceed the VaR threshold. The ES is then calculated as the negative average of these tail losses.

Applying this formula to our dataset yields an ES of approximately 0.05336181815045533 (or 5.34%). This result suggests that, in the worst 5% of cases, the average loss is expected to be around 5.34%, which is higher than the VaR of 3.88%. This is consistent with the definition of ES. Indeed, ES provides a more comprehensive view of the risk by considering both the frequency and the severity of extreme losses, which are not captured by the VaR metric alone.

4 Question C

In this question, the 'datatd1td2.csv' file used is join in the mail.

Shape Parameter with Picklands We first compute the daily returns. These daily returns form the basis for our extreme value analysis. Extreme Value Theory (EVT) is particularly useful in assessing the risk of rare events by focusing on the tails of the distribution of returns. We divide our analysis in two way, one with absolut extreme sorted losses and extreme sorted gains.
For the positive tail representing extreme gains, and the negative tail representing extreme losses, we calculate the shape parameters using the Pickands estimator:

$$\text{Pickands Estimator} = \frac{1}{\log(2)} \log\left(\frac{X_{n-k+1:n} - X_{n-2k+1:n}}{X_{n-2k+1:n} - X_{n-4k+1:n}}\right)$$

The estimated shape parameters for the gains and losses are 0.7031 and -0.4822, respectively. Invoking the Fisher-Tippett Theorem, the positive shape parameter for gains implies a heavy-tailed distribution, reflecting the potential for unbounded extreme gains. Conversely, the negative shape parameter for losses suggests a distribution with a finite upper limit, indicating that losses, while potentially extreme, are not without bound; rather, they are contained within a certain threshold, beyond which the probability of occurrence is zero.

EVT Var for various confidence level To calculate the Value at Risk (VaR) under the framework of Extreme Value Theory (EVT) for various confidence levels, we assume the returns are independent and identically distributed (iid).

The function was applied to both gains and losses at confidence levels of 90%, 95%, and 99%. The computed VaR values are:

- For Gains:
  - At 90% confidence level: 0.031226716623382784
  - At 95% confidence level: 0.031210095059210005
  - At 99% confidence level: 0.0311978341166503

- For Losses:
  - At 90% confidence level: $-0.245955076384582 42$
  - At 95% confidence level: -0.2544959672583297
  - At 99% confidence level: -0.26116242233901227

These results indicate that with an increase in the confidence level, the VaR for gains shows a slight decrease, reflecting minor changes in expected extreme gains. In contrast, the VaR for losses becomes more negative, suggesting a greater expected loss in more extreme scenarios.

5 Question D

In this question, the 'Dataset TD4.csv' file used is join in the mail.

Estimation of Almgren and Chriss model The data is processed to extract relevant information: the volume of transactions, transaction prices, and the sign of each transaction (indicating buying or selling pressure). Using this data, we calculate the returns as the percentage change between consecutive transaction prices.

Volatility Estimation The first parameter of interest in the Almgren and Chriss model is the volatility of the asset, denoted as σ. We estimate σ using the standard deviation of the returns, adjusted for the trading hours per day (24) and trading days per year (252) to obtain the annualize volatility. We obtain $\sigma = 10.7\%$.

γ Estimation The Almgren and Chriss model also requires the estimation of market impact parameters, specifically γ, this parameter is present in the permanent impact function of the model.

$$g\left(\frac{n_k}{\tau}\right) = \gamma\left(\frac{n_k}{\tau}\right)$$

where:

- $g\left(\frac{n_k}{\tau}\right)$ represents the permanent impact on price due to the trade.
- γ is the proportionality constant signifying the permanent market impact.
- $\frac{n_k}{\tau}$ is the rate of trading, with n_k being the size of the trade and τ representing the trading interval.

The model suggests that the permanent impact is directly proportional to the rate of trading. To estimate the parameter γ, we perform a linear regression analysis. In this regression, the dependent variable is the change in the asset's price, defined as $P(t+2) - P(t)$, and the independent variable is the rate of trading $\frac{n_k}{\tau}$. This regression helps us in quantifying the permanent market impact, represented by the slope of the regression line.

$$\text{Linear Regression:} \quad P(t+2) - P(t) = \gamma\left(\frac{n_k}{\tau}\right)$$

The regression analysis for estimating the market impact parameter γ has been completed. The estimated value of γ is 0.000502, with a mean squared error (MSE) of 0.00049

and a coefficient of determination (R^2) of 0.91. These results indicate a strong linear relationship between the signed volume and the price impact, as evidenced by the high R^2 value.
The regression plot, which visually depicts this relationship, is shown below:

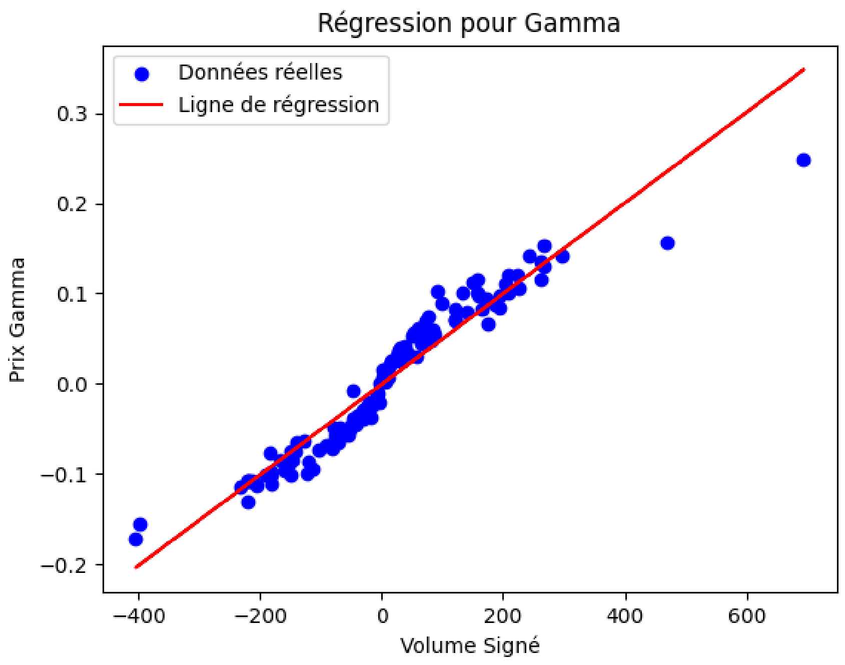

Figure 2: Linear Regression for Gamma Estimation

The positive slope of the regression line is consistent with the theoretical expectations of the Almgren and Chriss model, affirming the positive relationship between the trade size and the price change. This reinforces the validity of our model.

η Estimation

first approach : linear regression The estimation of η is performed using a linear regression model, with the formula:

$$P(t+2) - P(t+1) = h\left(\frac{n_k}{\tau}\right) = \phi \cdot \operatorname{sgn}(n_k) + \eta\left(\frac{n_k}{\tau}\right)$$

Here, ϕ represents the half spread, and $h\left(\frac{n_k}{\tau}\right)$ is the transient impact that is reflected in the price difference. The term $\phi \cdot \operatorname{sgn}(n_k)$ captures the component of the price impact that is independent of the size of the trade, while $\eta\left(\frac{n_k}{\tau}\right)$ captures the portion of the impact that scales with the rate of trading.

By the analysis of Figure 3, we understand that the data is clearly non-linear, so our model is not well specified.

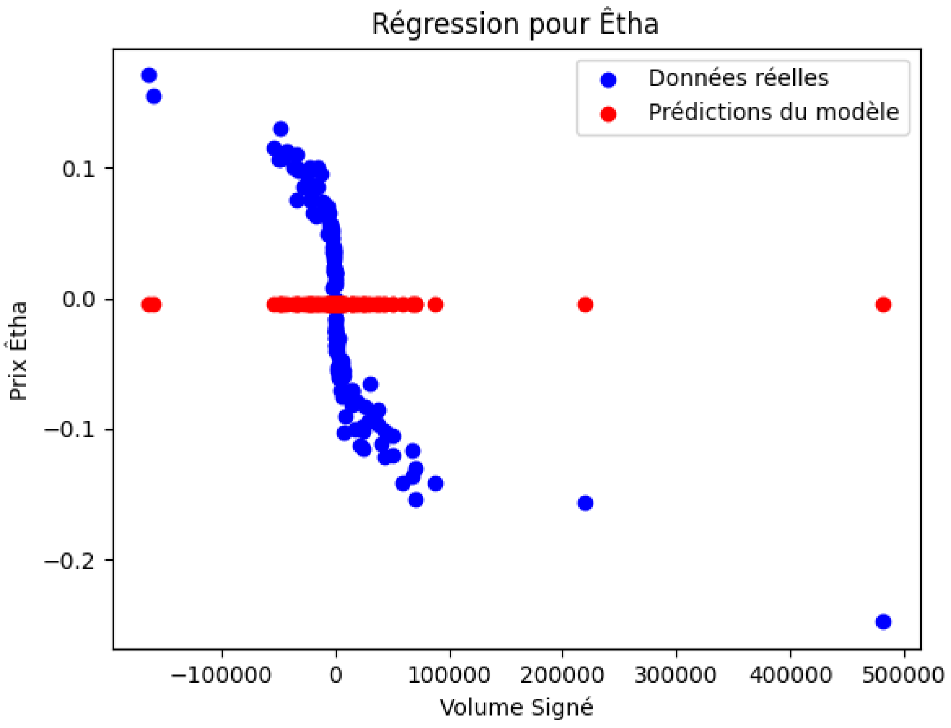

Figure 3: linear Regression for Eta Estimation

Second approach : multivariate regression So we gonna try a non linear approach by combining the quadratic cost of the first linear regression and the first equation this lead us to the following multivariate regression.

$$h\left(n_k^2 \cdot \mathrm{Sign}(n_k)\right) = \eta\left(n_k^2 \cdot \mathrm{Sign}(n_k)\right) = P(t+2) - P(t+1)$$

Where $h(n_k^2 \cdot \mathrm{Sign}(n_k))$ represents the price change due to the temporary market impact, and $n_k^2 \cdot \mathrm{Sign}(n_k)$ is the squared signed volume of the trade.

The regression analysis yields the following results for η and you can observe the plot in Figure 4:

- Estimated η: 5.966990046987537e-08
- Mean Squared Error (MSE): 0.00052813
- Coefficient of Determination (R^2): 0.913

Surprisingly, we find the same value for η with the two regression despite the fact that the regression metrics are extremely different.

Conclusion of parameter estimation The substantial difference in the R^2 values and the visual divergence in the regression plots raise questions about the model's specification. While the model for γ appears to be well-specified, capturing a significant proportion of the variance in price changes, the model for η suggests that additional factors may need to be considered to fully understand the temporary market impact. The non-linear relationship observed in the first regression may indicate the need for a more sophisticated model or the inclusion of additional explanatory variables to better capture the nuances of market behavior.

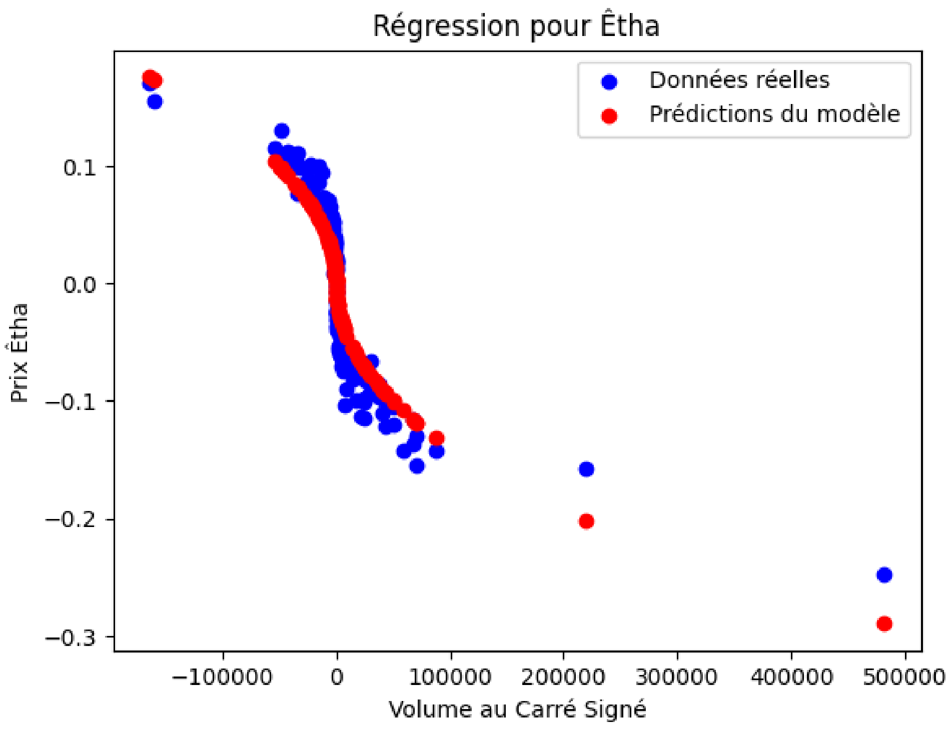

Figure 4: Multivariate Regression for Eta Estimation

Liquidation strategy despite misspecified η Despite the anomalies encountered in the specification of η, with Almgren and Chriss model. The model suggests that the optimal liquidation trajectory is influenced by the trade-off between liquidity risk and market risk, which can be modulated by the parameter λ, representing the risk aversion level.

We examine the liquidation trajectories for different values of λ spanning several orders of magnitude, with λ values of 1×10^{-5}, 1×10^{-4}, and 1×10^{-3}. These trajectories are calculated over a 24-hour period, with T representing the fractional part of the day.

The resulting liquidation strategies illustrate how the choice of λ affects the liquidation speed. A higher λ corresponds to a more aggressive liquidation early on, as the trader is willing to incur a greater liquidity risk to reduce exposure to market risk. Conversely, a lower λ reflects a more gradual approach, minimizing liquidity risk at the expense of market risk exposition.

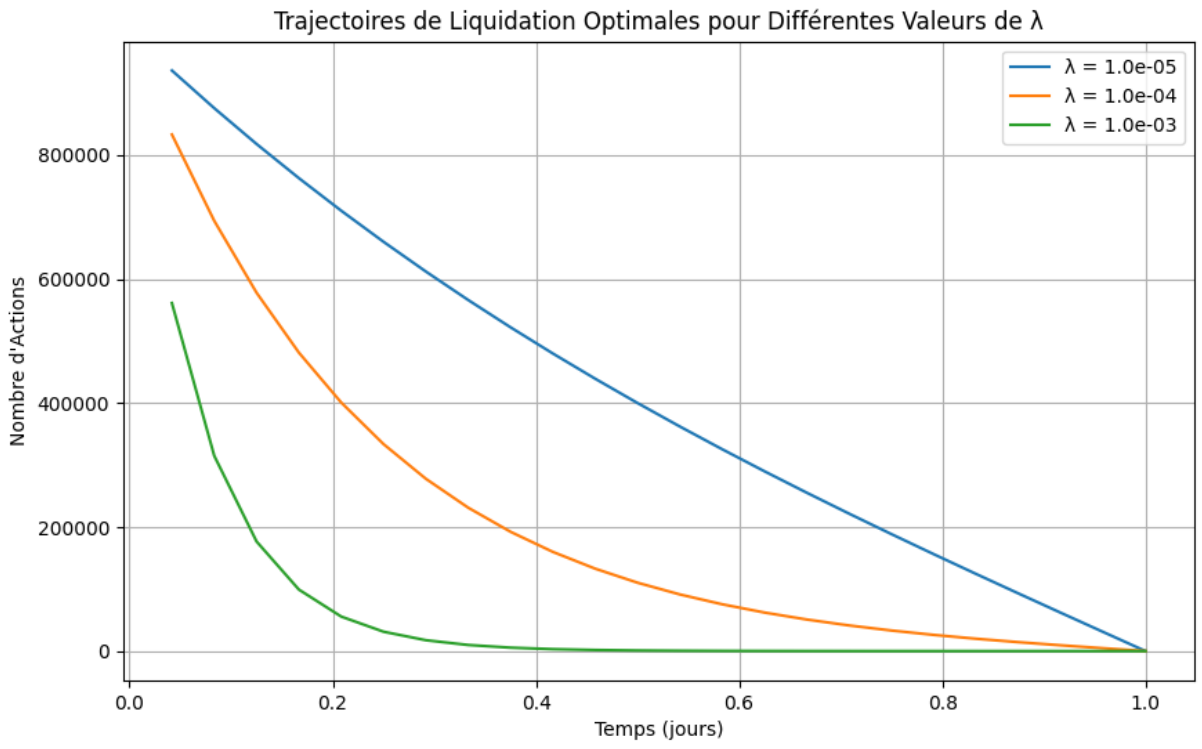

Figure 5: Liquidation strategies for various risk-aversion

6 Question E

In this question, the ‘Dataset-TD5-csv.csv‘ file used is join in the mail. And we work roughly on the last week of data.

Multiresolution Correlation Matrices To achieve this, we employ wavelet transforms, specifically the Haar transform. This approach allows us to decompose the time series data into different dimension components and analyze correlations at multiple levels of resolution.
We focus on one week of data (672 data points), this number is near $2^9 = 512$. So we will work with 513 data points to have 512 returns.

The process begins with the calculation of returns for each currency pair (GBPEUR, SEKEUR, and CADEUR) in our dataset. These returns are then subjected to the Haar wavelet transform. The Haar transform is simple indeed at each level of resolution, we calculate the average value of two consecutive points. This process reduces the total length of the series by a exponent two at each level of resolution.

For each level of decomposition, we construct a correlation matrix. This is done by applying the `multiresolution_correlation` function to pairs of transformed time series. The function computes the correlation coefficient between pairs of wavelet coefficients at a given level of resolution. These correlation matrices provide insights into how the relationships between the currency pairs evolve across different timescales.
As an illustration, the correlation matrix at the finest level (level 9) for our currency pairs is as follows:

$$\begin{bmatrix} 1.00 & 0.03 & 0.24 \\ 0.03 & 1.00 & -0.05 \\ 0.24 & -0.05 & 1.00 \end{bmatrix}$$

This matrix reveals the degree of linear relationship between each pair of currency returns at the specified resolution. A value close to 1 or -1 indicates a strong positive or negative correlation, respectively, whereas values near 0 imply a lack of linear relationship.

Hurst Exponent & Volatility Vector To obtain our Volatility Vector, we need to estimate the Hurst exponent for each currency pair using the method of absolute moments for $k = 2$. This method involves the following steps:

1. **Time Series Length**: We determine the total number of observations in the series, denoted as T.
2. **Second Moment Calculation**: The second moment, $M2$, is computed as the mean of squared absolute differences between consecutive values of the series.
3. **Adjusted Second Moment**: We calculate $M2'$, the mean of squared absolute differences for every second point in the series.
4. **Hurst Exponent Estimation**: The Hurst exponent, H, is estimated using $H = 0.5 \times \log_2(M2'/M2)$.

With the Hurst exponent calculated, we scale the volatilities of each currency pair to different timeframes up to a weekly scale. The initial step involves calculating the high-frequency volatility (15 min) for GBPEUR, SEKEUR, and CADEUR.

For each level j, representing different timescales-jump, we compute the scaled volatility using:

$$\text{Low-Frequency Volatility} = \text{High-Frequency Volatility} \times (2^j)^{\text{Hurst Exponent}}$$

For GBPEUR, the volatilities range from approximately 0.000356 at the shortest timescale to about 0.017102 at the weekly scale. In contrast, SEKEUR shows a lower volatility starting from 0.000183 and reaching up to 0.007030 over the same periods. CADEUR volatility begins at 0.000306 and extends to 0.014621.

Covariance Matrix Calculation and Portfolio Volatility Estimation To estimate the volatility of our portfolio containing the three FX rates (GBPEUR, SEKEUR, and CADEUR) at different timescales, we first construct covariance matrices for each level of resolution. This is achieved by combining the scaled volatilities with the correlation matrices obtained from the multiresolution analysis.

For each resolution level j ranging from 0 to 9, corresponding to timescales from 15 minutes to roughly one week, we perform the following steps:

1. We retrieve the correlation matrix at the current scale from the list of correlation matrices obtained using the wavelet-based method.

2. A covariance matrix is then constructed for each scale. Each element of the covariance matrix is calculated as the product of the correlation coefficient between two currency pairs and the scaled volatilities of these pairs. Mathematically, the covariance matrix element $\text{Cov}_{i,k}$ is given by :
   $\text{Cov}_{i,k} = \text{Correlation}_{i,k} \times \text{Volatility}_i \times \text{Volatility}_k$.

Using these covariance matrices, we calculate the portfolio's volatility at each timescale. Assuming equal weights for each currency pair in our portfolio, represented by vector $W = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]$, we compute the portfolio variance as the dot product of $W{\cdot}T$ with the covariance matrix and W again. The portfolio volatility at each scale is then the square root of the portfolio variance.

Upon completing the volatility estimation at various scales, we further extend our analysis to calculate the annualized volatility of the portfolio.
The annualized portfolio volatility is derived from the weekly scale volatility, under the assumption that there are 52 weeks in a year. For our portfolio, consisting of equal weights in GBPEUR, SEKEUR, and CADEUR, the annualized volatility is found to be approximately 6.23%.
This level of annualized volatility is satisfactory for our portfolio composition, indicating a moderate level of risk. It reflects the combined effect of the individual volatilities of the currency pairs and their correlations.

Overlapping volatility Now the idea is to compute the volatility of the portfolio directly. So, first we have to study a way to approach volatility calculation.

- We opt for a technique based on **overlapping returns**, which entails calculating returns over a sliding window of data. This approach is favored for its ability to effectively capture shorter-term volatility patterns.
- To ensure the credibility of this approach, we rigorously validate that the chosen horizon (H) for the sliding window does not substantially exceed the total number of observations (T) in our dataset. This critical verification is accomplished by calculating delta, representing the ratio of the window size to the number of data points, and ensuring it remains comfortably **below the 68% threshold**.
- Our methodology specifies **a bi-daily (two-day) horizon** for the assessment of volatility. This decision strikes an optimal balance, allowing us to account for short-term market fluctuations while maintaining an adequate number of data points to uphold statistical reliability

After this estimation, we need to verify its validity. To do so, we must test various aspects:

- To validate our approach and assess the normality of returns, we employ **the Jarque-Bera test**.This statistical test verifies the alignment of our returns distribution with the expected $\sqrt{T}(\mathcal{M}(H)-H)$ convergence to a $\mathcal{N}(0, \frac{2}{3}H(H-1)(2H-1))$ distribution.
- For hypothesis testing, we set a significance level (alpha) at 0.05. A p-value below 0.05 in the Jarque-Bera test allows us to reject the assumption of normality in returns, while a p-value above 0.05 indicates a lack of sufficient evidence to make such a rejection.

In conclusion, our chosen methodology, which incorporates the use of overlapping returns, a bi-daily horizon, and the Jarque-Bera test, yields consistent results. Furthermore, the validity of our approach is supported by the fact that the calculated delta is less than 68%, indicating that our chosen horizon is suitable for the dataset. Additionally, the acceptance of the null hypothesis under the Jarque-Bera test provides further confidence in the robustness of our analysis, indicating that the distributional properties of the portfolio's returns align with our chosen statistical assumptions.

7 Appendix

```
import pandas as pd
import numpy as np
import math as m
import matplotlib.pyplot as plt
from scipy.stats import norm

def load_txt_file(filename='Nat.txt'):
    try:
        data = pd.read_csv(filename, sep="\t", names=['date', 'prix
           '], dayfirst=True)
        data['prix'] = data['prix'].astype(str).str.replace(',', '.
           ').astype(float)
        return data
    except Exception as e:
        print(f"Erreur␣lors␣de␣la␣lecture␣du␣fichier:␣{e}")
        return None

def save_to_csv(df, filename='dataTd1Td2.csv'):
    try:
        df.to_csv(filename, index=False)
        print(f"Les␣donn es␣ont␣  t  ␣sauvegard es␣dans␣le␣
           fichier␣{filename}.")
    except Exception as e:
        print(f"Erreur␣lors␣de␣la␣sauvegarde:␣{e}")

data = load_txt_file()
if data is not None:
    save_to_csv(data)

Natixis_Data = pd.read_csv("dataTd1Td2.csv")

returns = []

for i in range(1, len(Natixis_Data)):
    close_price_today = Natixis_Data['prix'][i]
    close_price_yesterday = Natixis_Data['prix'][i - 1]
    daily_return = (close_price_today - close_price_yesterday) /
       close_price_yesterday
    returns.append(daily_return)

data['Daily_Return'] = [None] + returns

print(returns)

data['date'] = pd.to_datetime(data['date'], format='%d/%m/%Y')
data_k = data[(data['date'] >= '2015-01-01') & (data['date'] <= '
   2016-12-31')].dropna()
```

```
data_k.head(), data_k.tail()

returns_kernel = data_k['Daily_Return'].values

def biweight_kernel(u):
    abs_u = np.abs(u)
    mask = abs_u <= 1
    return (15/16) * (1 - u**2)**2 * mask

def kernel_density_estimate_biweight(x, data, h):
    n = len(data)
    u = (x - data) / h
    kde = np.sum(biweight_kernel(u)) / (n * h)
    return kde

n = len(returns_kernel)
h = 1.06 * np.std(returns_kernel) * (n ** (-1/5))

x_range = np.linspace(min(returns_kernel), max(returns_kernel), num
   =1000)

kde_values_biweight = [kernel_density_estimate_biweight(x,
   returns_kernel, h) for x in x_range]

def empirical_cdf(x, data):
    return np.sum(data <= x) / len(data)

cdf_values = [empirical_cdf(x, returns_kernel) for x in x_range]

probability_level = 0.95
non_parametric_var_biweight = None
for i, cdf in enumerate(cdf_values):
    if cdf >= (1 - probability_level):
        non_parametric_var_biweight = x_range[i]
        break

non_parametric_var_biweight_message = f"Non-Parametric␣VaR␣(
   Biweight␣Kernel)␣at␣{probability_level␣*␣100}%␣confidence␣level:
   ␣{non_parametric_var_biweight:.4f}"
print(non_parametric_var_biweight_message)

plt.figure(figsize=(10, 6))
plt.plot(x_range, kde_values_biweight, label='density␣(Biweight␣
   Kernel)', color='blue')
plt.plot(x_range, cdf_values, label='Empirical␣CDF', color='green')
plt.axvline(x=non_parametric_var_biweight, color='red', linestyle='
   --', label='VaR')
plt.xlabel('Return')
plt.ylabel('Density␣/␣CDF')
plt.title('RCDF,␣and␣Non-Parametric␣VaR␣with␣Biweight␣Kernel')
plt.grid(True)
plt.legend()
plt.show()

data['date'] = pd.to_datetime(data['date'], format='%d/%m/%Y')
data_2017 = data[(data['date'] >= '2017-01-01') & (data['date'] <=
    '2018-12-31')]

data_2017['Daily_Return'] = data_2017['prix'].pct_change()

data_filtered = data_2017.dropna()

var_threshold = -0.0388  # This is the VaR estimated previously
exceedances = data_2017['Daily_Return'] < var_threshold  # VaR
    exceedances are when returns are below the threshold
exceedance_proportion = exceedances.sum() / len(data_2017)

confidence_level = 0.95
expected_proportion = 1 - confidence_level
validation = exceedance_proportion <= expected_proportion

exceedance_proportion, validation

"""# Question 2"""

var = -0.0388
tail_losses = [loss for loss in returns_kernel if loss < var]

expected_shortfall = - (sum(tail_losses) / len(tail_losses))

print(f"The␣Expected␣Shortfall␣at␣95%␣confidence␣level␣is:␣{
    expected_shortfall}")

"""# Question 3"""

data = pd.read_csv("dataTd1Td2.csv")

returns = []

for i in range(1, len(Natixis_Data)):
    close_price_today = Natixis_Data['prix'][i]
    close_price_yesterday = Natixis_Data['prix'][i - 1]
    daily_return = (close_price_today - close_price_yesterday) /
        close_price_yesterday
    returns.append(daily_return)

data['Daily_Return'] = [None] + returns

print(returns)
```

```
gains = data['Daily_Return'][data['Daily_Return'] > 0]
sort_gains = np.sort(gains)

losses = data['Daily_Return'][data['Daily_Return'] < 0]
sort_losses = np.sort(abs(losses))

def pickands_estimator(extreme_values):
    n = len(extreme_values)
    k = int(np.log(n))
    return (1 / np.log(2)) * np.log((extreme_values[n-k+1] -
       extreme_values[n - 2*k + 1]) / (extreme_values[n - 2*k +1] -
        extreme_values[n - 4*k +1]))

shape_param_gains_pickands = pickands_estimator(sort_gains)
shape_param_losses_pickands = pickands_estimator(sort_losses)

shape_param_gains_pickands, shape_param_losses_pickands

def calculate_var(extreme_values, k, xi_p, confidence_level):
    n = len(extreme_values)
    p = 1 - confidence_level
    l = int(np.log(n))

    var_p = ((k / (n * (1 - p))) ** xi_p - 1) / (1 - 2 ** (-xi_p))
       * (extreme_values[n-l+1] - extreme_values[n-2*l+1]) +
       extreme_values[n-l+1]
    return var_p


confidence_levels = [0.9, 0.95, 0.99]
vars_confidence_levels_gains = {cl: calculate_var(sort_gains, 1,
   shape_param_gains_pickands, cl) for cl in confidence_levels}
vars_confidence_levels_losses = {cl: calculate_var(sort_losses, 1,
   shape_param_losses_pickands, cl) for cl in confidence_levels}

print('gains␣var',vars_confidence_levels_gains)
print('losses␣var',vars_confidence_levels_losses)

"""# Question 4"""

csv_data = pd.read_csv("Dataset␣TD4.csv", delimiter=";")


volume = []
transaction_price = []
transaction_sign = []


for i in range(len(csv_data) - 2):
    if pd.notna(csv_data['volume␣of␣the␣transaction␣(if␣known)'].
       iloc[i]):
        volume.append(csv_data['volume␣of␣the␣transaction␣(if␣known
           )'].iloc[i])
        transaction_price.extend([csv_data['Price␣(before␣
           transaction)'].iloc[i], csv_data['Price␣(before␣
           transaction)'].iloc[i + 1]])
        transaction_sign.append(csv_data['Sign␣of␣the␣transaction'
           ].iloc[i])



returns = [(transaction_price[i + 1] - transaction_price[i]) /
   transaction_price[i] for i in range(len(transaction_price) - 1)]
returns = np.array(returns)

sigma = np.std(returns) * np.sqrt(24) * np.sqrt(252)  # 24 pour les
    heures par jour, 252 pour les jours de trading par an
sigma

total_volume = np.sum(volume)

time_interval = 1/24

price_difference = np.array([transaction_price[i+1] -
   transaction_price[i] for i in range(0, len(transaction_price) -
   1, 2)])
signed_volume = np.array([volume[i] * transaction_sign[i] for i in
   range(len(volume))])
volume_squared_signed = np.array([transaction_sign[i] * volume[i]
   ** 2 for i in range(len(price_difference))])

modified_price_gamma = price_difference
modified_price_ tha = -price_difference
X_gamma_mod = signed_volume.reshape(-1,1)
X_ tha_mod = np.column_stack((signed_volume, volume_squared_signed
   ))

total_volume, modified_price_gamma[:5], modified_price_ tha[:5],
   X_gamma_mod[:5], X_ tha_mod[:5]

import numpy as np
import matplotlib.pyplot as plt

def linear_regression_np(X, y):
    X_mean = np.mean(X)
    y_mean = np.mean(y)

    slope = np.sum((X - X_mean) * (y - y_mean)) / np.sum((X -
       X_mean)**2)
    intercept = y_mean - slope * X_mean

    return slope, intercept

X_gamma_array = np.array(X_gamma_mod[:, 0])
prix_gamma_array = np.array(modified_price_gamma)

slope_gamma, intercept_gamma = linear_regression_np(X_gamma_array,
   prix_gamma_array)

y_pred_gamma = slope_gamma * X_gamma_array + intercept_gamma
mse_gamma = np.mean((prix_gamma_array - y_pred_gamma)**2)
r2_gamma = 1 - sum((prix_gamma_array - y_pred_gamma)**2) / sum((
   prix_gamma_array - np.mean(prix_gamma_array))**2)

print(f"\nEstimation␣gamma␣:␣{slope_gamma}")
print(f"MSE␣:␣{mse_gamma}␣;␣R^2␣:␣{r2_gamma}")

plt.scatter(X_gamma_array, prix_gamma_array, color="blue", label="
   Donn es␣r elles")
plt.plot(X_gamma_array, y_pred_gamma, color="red", label="Ligne␣de␣
   r gression")
plt.xlabel("Volume␣Sign ")
plt.ylabel("Prix␣Gamma")
plt.title("R gression␣pour␣Gamma")
plt.legend()
plt.show()

import numpy as np
import matplotlib.pyplot as plt

def linear_regression_multivariate(X, y):
    X_mean = np.mean(X, axis=0)
    y_mean = np.mean(y)

    # Calcul de la pente (slope) et de l'ordonn e   l'origine (
       intercept) pour une r gression multivari e
    X_cov = np.dot(X.T, X) - np.sum(X, axis=0) * X_mean
    X_cov_inv = np.linalg.inv(X_cov)
    X_X_mean_y_mean = np.dot(X.T, y) - np.sum(X, axis=0) * y_mean
    coefficients = np.dot(X_cov_inv, X_X_mean_y_mean)

    intercept = y_mean - np.dot(coefficients.T, X_mean)

    return coefficients, intercept

X_ tha_array = np.column_stack((X_gamma_mod, volume_squared_signed
   ))
prix_ tha_array = np.array(modified_price_ tha)
```

```
coefficients_ tha, intercept_ tha =
   linear_regression_multivariate(X_ tha_array, prix_ tha_array)

y_pred_ tha = np.dot(X_ tha_array, coefficients_ tha) +
   intercept_ tha
mse_ tha = np.mean((prix_ tha_array - y_pred_ tha)**2)
r2_ tha = 1 - sum((prix_ tha_array - y_pred_ tha)**2) / sum((
   prix_ tha_array - np.mean(prix_ tha_array))**2)

 tha  = coefficients_ tha[1] * tau # Etha est le second
   coefficient
print(f"\nEstimation␣ tha ␣:␣{ tha }")
print(f"MSE␣:␣{mse_ tha}␣;␣R^2␣:␣{r2_ tha}")

plt.scatter(volume_squared_signed, prix_ tha_array, color="blue",
   label="Donn es␣r elles")
plt.scatter(volume_squared_signed, y_pred_ tha, color="red", label
   ="Pr dictions␣du␣mod le")
plt.xlabel("Volume␣au␣Carr ␣Sign ")
plt.ylabel("Prix␣ tha ")
plt.title("R gression␣pour␣ tha ")
plt.legend()
plt.show()

X = 1000000
T = [(1/24) * i for i in range(1, 25)]

Lambda_values = [1*10**(-5), 1*10**(-4), 1*10**(-3)]


# Calcul de K pour chaque valeur de lambda et trac  des
   strat gies de liquidation
plt.figure(figsize=(10, 6))
for Lambda in Lambda_values:
    K = np.sqrt(Lambda * (sigma**2) /  tha )
    strat = [np.sinh(K * (1 - t)) * X / np.sinh(K * 1) for t in T]
    plt.plot(T, strat, label=f'  ␣=␣{Lambda:.1e}')

plt.title('Trajectoires␣de␣Liquidation␣Optimales␣pour␣Diff rentes␣
   Valeurs␣de␣  ')
plt.xlabel('Temps␣(jours)')
plt.ylabel('Nombre␣d\'Actions')
plt.legend()
plt.grid(True)
plt.show()

"""# Question 5"""

import pandas as pd
import numpy as np

Currencies = pd.read_csv("Dataset-TD5-csv.csv")

# Split du dataset et calcul des moyennes
GBPEUR = Currencies.iloc[:,:3]
SEKEUR = Currencies.iloc[:,3:6]
CADEUR = Currencies.iloc[:,6:]


GBPEUR['GBPEUR␣AVG'] = GBPEUR.iloc[:, [1, 2]].mean(axis=1)
SEKEUR['SEKEUR␣AVG'] = SEKEUR.iloc[:, [1, 2]].mean(axis=1)
CADEUR['CADEUR␣AVG'] = CADEUR.iloc[:, [1, 2]].mean(axis=1)

# On verif que nous disposons des 512 derniers points de donn es (
    apr s avoir supprim  les valeurs manquantes) proches de 672,
    ce qui  quivaut     7 * 96.
GBPEUR = GBPEUR.iloc[-513:]
SEKEUR = SEKEUR.iloc[-513:]
CADEUR = CADEUR.iloc[-513:]

GBPEUR_returns = GBPEUR['GBPEUR␣AVG'].pct_change().dropna()
SEKEUR_returns = SEKEUR['SEKEUR␣AVG'].pct_change().dropna()
CADEUR_returns = CADEUR['CADEUR␣AVG'].pct_change().dropna()

def haar_transform(data):
    n = len(data)
    output = np.zeros(n)
    while n > 1:
        n = n // 2
        for i in range(n):
            output[i] = (data[2 * i] + data[2 * i + 1]) / 2
            output[n + i] = (data[2 * i] - data[2 * i + 1]) / 2
        data[:n] = output[:n]
    return output

def multiresolution_correlation(haar1, haar2, level):
    length = 2 ** level
    truncated_haar1 = haar1[:length]
    truncated_haar2 = haar2[:length]
    return np.corrcoef(truncated_haar1, truncated_haar2)[0, 1]

def correlation_matrix_at_level(level, haar_transforms):
    matrix_size = len(haar_transforms)
    correlation_matrix = np.zeros((matrix_size, matrix_size))
    for i in range(matrix_size):
        for j in range(i, matrix_size):
            if i == j:
                correlation_matrix[i, j] = 1
            else:
                correlation = multiresolution_correlation(
                   haar_transforms[i], haar_transforms[j], level)
                correlation_matrix[i, j] = correlation_matrix[j, i]
                    = correlation
    return correlation_matrix

gbpeur_haar = haar_transform(GBPEUR_returns.values)
sekeur_haar = haar_transform(SEKEUR_returns.values)
cadeur_haar = haar_transform(CADEUR_returns.values)

num_levels = 9

haar_transforms = [gbpeur_haar, sekeur_haar, cadeur_haar]
correlation_matrices = [correlation_matrix_at_level(level,
   haar_transforms) for level in range(num_levels)]

correlation_matrix_j9 = correlation_matrices[-1]
correlation_matrix_j9

import math as m

def Hurst(serie):
    T = len(serie)
    M2 = np.mean((abs(serie[1:] - serie[:-1]))**2)
    M2_prime = np.mean(abs((serie[2:] - serie[:-2]))**2)
    H_est = 0.5 * m.log2(M2_prime / M2)
    return H_est

hurst_gbpeur = Hurst(GBPEUR['GBPEUR␣AVG'].values)
hurst_sekeur = Hurst(SEKEUR['SEKEUR␣AVG'].values)
hurst_cadeur = Hurst(CADEUR['CADEUR␣AVG'].values)
#hurst_gbpeur = Hurst(GBPEUR_returns.values)
#hurst_sekeur = Hurst(SEKEUR_returns.values)
#hurst_cadeur = Hurst(CADEUR_returns.values)

hurst_gbpeur, hurst_sekeur, hurst_cadeur

vol_gbpeur_hf = np.std(GBPEUR_returns)
vol_sekeur_hf = np.std(SEKEUR_returns)
vol_cadeur_hf = np.std(CADEUR_returns)

volatility_gbpeur = []
volatility_sekeur = []
volatility_cadeur = []

num_levels = 9
for j in range(num_levels):
    periods = 2 ** j
    volatility_gbpeur.append(vol_gbpeur_hf * (periods **
       hurst_gbpeur))
    volatility_sekeur.append(vol_sekeur_hf * (periods **
       hurst_sekeur))
    volatility_cadeur.append(vol_cadeur_hf * (periods **
       hurst_cadeur))

volatility_gbpeur, volatility_sekeur, volatility_cadeur

num_levels = 9
correlation_matrices = [correlation_matrix_at_level(level, [
   gbpeur_haar, sekeur_haar, cadeur_haar]) for level in range(
   num_levels)]

volatility_gbpeur = [vol_gbpeur_hf * (2**j)**hurst_gbpeur for j in
   range(num_levels)]
volatility_sekeur = [vol_sekeur_hf * (2**j)**hurst_sekeur for j in
   range(num_levels)]
volatility_cadeur = [vol_cadeur_hf * (2**j)**hurst_cadeur for j in
   range(num_levels)]

covariance_matrices = []

for j in range(num_levels):
    correlation_matrix = correlation_matrices[j]

    covariance_matrix = np.zeros((3, 3))

    volatilities = [volatility_gbpeur[j], volatility_sekeur[j],
       volatility_cadeur[j]]
    for i in range(3):
        for k in range(3):
            covariance_matrix[i, k] = correlation_matrix[i, k] *
               volatilities[i] * volatilities[k]

    covariance_matrices.append(covariance_matrix)

# Supposons que covariance_matrices est la liste de vos matrices de
    covariance
covariance_matrices = [np.nan_to_num(matrix) for matrix in
   covariance_matrices]

covariance_matrix = covariance_matrices[0]
print(covariance_matrix)

W = np.array([1/3, 1/3, 1/3])

portfolio_volatility = []

for covariance_matrix in covariance_matrices:
    portfolio_variance = np.dot(W.T, np.dot(covariance_matrix, W))
    portfolio_volatility.append(np.sqrt(portfolio_variance))
```

```
portfolio_volatility
#portfolio_volatility_j9 = portfolio_volatility[8]
#portfolio_annual = np.sqrt(52) * portfolio_volatility_j9 ##juste
   pour v rifier
#portfolio_annual

import scipy.stats as stats

Currencies = pd.read_csv("Dataset-TD5-csv.csv")

Currencies = Currencies.iloc[-513:]


GBPEUR = Currencies.iloc[:,:3]
SEKEUR = Currencies.iloc[:,3:6]
CADEUR = Currencies.iloc[:,6:]

GBPEUR['GBPEUR␣AVG'] = GBPEUR.iloc[:, [1, 2]].mean(axis=1)
SEKEUR['SEKEUR␣AVG'] = SEKEUR.iloc[:, [1, 2]].mean(axis=1)
CADEUR['CADEUR␣AVG'] = CADEUR.iloc[:, [1, 2]].mean(axis=1)
Currencies['Portfolio␣AVG'] = (GBPEUR['GBPEUR␣AVG'] + SEKEUR['
   SEKEUR␣AVG'] + CADEUR['CADEUR␣AVG'] )/3

# On definit le nombre de lines par ech lle de temps
lines_per_scale = {
    '15min': 1,
    '1hour': 4,
    '1day': 96,
    # on peut en ajouter plus si on veut
}

# On d finit la window_size pour du bi-journalier
window_size = 2 * lines_per_scale['1day']

# overlapping
overlapping_vols = []
for i in range(0, len(Currencies) - window_size + 1):
    window_returns = Currencies['Portfolio␣AVG'][i:i + window_size
       ].pct_change().dropna()
    vol = np.std(window_returns)
    overlapping_vols.append(vol)


annualized_overlap = np.mean(overlapping_vols) * np.sqrt(252 /
   window_size)
scaling_factor = 100
annualized_overlap_scaled = annualized_overlap * scaling_factor
```

```python
print("Bi-Journalier␣-␣Volatility:")
print(f"Bi-Journalier␣-␣Overlapping␣Volatility:␣{
   annualized_overlap_scaled}")

# Calcule du delta (horizon / nombre de donn es)
delta = window_size / len(Currencies)
print(f"Delta␣(Horizon␣/␣Nombre␣de␣donn es):␣{delta}") # Pour que
   l'estimation est du sens delta < 68% ici c'est bien le cas

from scipy.stats import jarque_bera
# Le but est de verifier la propri t  de Lo et Mackinlay. On va
   le faire   l'aide d'un test de Jarque et Berra

# Calcul de la variance de overlapping_vols
var_overlapping_vols = np.var(overlapping_vols)

# Calcul de M(H)
M_H = var_overlapping_vols / (annualized_overlap ** 2)

# Nombre d'observations
T = len(overlapping_vols)

# Valeur de H
H = window_size

# Calcul de la statistique de test
test_statistic, p_value = jarque_bera(np.sqrt(T) * (M_H - H))

alpha = 0.05

if p_value < alpha:
    print("Le␣test␣rejette␣l'hypoth se␣de␣normalit .")
else:
    print("Le␣test␣n'a␣pas␣suffisamment␣de␣preuves␣pour␣rejeter␣l'
       hypoth se␣de␣normalit .")
```