Project 01

Name: Kefan Liang NetID: kl508

Problem 1

Given the dataset in problem1.csv

A. Calculate the Mean, Variance, Skewness and Kurtosis of the data

---

Using pandas packages to calculate this statistics:

```
Mean: 0.05019795790476916
Variance: 0.010322143931072109
Skewness: 0.1204447119194402
Kurtosis: 0.2229270674503816
```

B. Given a choice between a Normal Distribution and a T-Distribution, which one would you choose to model the data? Why?

If the data has strong symmetry, skewness is close to 0, and kurtosis is close to 3, it is more likely to be normal distribution. If the data is tail-heavy (kurtosis greater than 3) or has high skewness, the T distribution is better suited.

After these two comparation, we think **the data is more inclined to T distribution.**

C. Fit both distributions and prove or disprove your choice in B using methods presented in class.

Based on the initial assessment criteria—**symmetry, skewness close to 0, and kurtosis close to 3**—we initially suspected that a **T-distribution** might be a better fit due to potential heavy tails or high skewness. However, after evaluating both models using **AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion)**, the results suggest that the **Normal distribution provides a better fit** for the data.

- **AIC for Normal Distribution: -1731.59, AIC for T-Distribution: -1731.42**
- The lower AIC value for the **Normal distribution** suggests it fits the data slightly better in terms of likelihood while balancing model complexity.
- **BIC for Normal Distribution: -1721.77, BIC for T-Distribution: -1716.70**
- And **The Normal distribution has a lower BIC value**, indicating that when penalizing for model complexity, it is preferred over the **T-distribution**. Since BIC strongly discourages overfitting, this further reinforces that the Normal distribution is a better choice.

Even though the initial moment-based analysis suggested a T-distribution due to tail heaviness or skewness, **the AIC and BIC metrics objectively indicate that the Normal distribution fits the data better**. Therefore, based on **both likelihood-based model selection and penalization for complexity**, the **Normal distribution should be chosen** as the best fit for this dataset.

Problem 2

Given the data in problem2.csv

A. Calculate the pairwise covariance matrix of the data.

Simply use packages to calculate the pairwise covariance:

```
Pairwise Covariance Matrix
    x1        x2        x3        x4        x5
x1  1.470484  1.454214  0.877269  1.903226  1.444361
x2  1.454214  1.252078  0.539548  1.621918  1.237877
x3  0.877269  0.539548  1.272425  1.171959  1.091912
x4  1.903226  1.621918  1.171959  1.814469  1.589729
x5  1.444361  1.237877  1.091912  1.589729  1.396186
```

B. Is the Matrix at least positive semi-definite? Why?

Calculate the eigen values, and if all the eigen values larger than 0 which is `np.all(eigenvalues >= 0)`, the matrix is positive seme-definite.

```
Eigenvalues of the covariance matrix:
[ 6.78670573  0.83443367 -0.31024286  0.02797828 -0.13323183]
Is the covariance matrix positive semidefinite?: False
```

C. If not, find the nearest positive semi-definite matrix using Higham's method and the near-psd method of Rebenato and Jackel.

For Higham's method:

1. Ensure symmetry: If the matrix is asymmetric, take its symmetric part.
2. Calculate eigenvalue decomposition:$A = Q\Lambda Q^T$
3. Set negative eigenvalues to zero to obtain a modified eigenvalue matrix: A^+
4. Reconstruct the matrix: $A^+ = Q\Lambda^+ Q^T$
5. Reconstruct the matrix: Ensure that the diagonal is unchanged: maintain the property that the diagonal of the covariance matrix is 1.
6. Continue iteration until the matrix converges to the PSD.

```
Higham adjusted PSD covariance matrix:
[[1.47048437 1.33367702 0.89435261 1.62845754 1.40155153]
 [1.33367702 1.25207795 0.63522726 1.45526619 1.22052732]
 [0.89435261 0.63522726 1.272425   1.08023227 1.06208209]
 [1.62845754 1.45526619 1.08023227 1.81446921 1.57796266]
 [1.40155153 1.22052732 1.06208209 1.57796266 1.39618646]]
```

For near-psd method of Rebenato and Jackel:

1. Compute the singular value decomposition (SVD) of a matrix: $A = USV^T$
2. Set negative singular values to zero: $S^+ = max(s, 0)$
3. Reconstruct the matrix: $A+ = US^+V^T$
4. Adjust the diagonal elements to equal 1 (ensuring the properties of the covariance matrix)

```
Rebonato & Jackel adjusted PSD covariance matrix:
[[1.         1.45421424 0.87726904 1.90322645 1.44436105]
 [1.45421424 1.         0.53954816 1.62191837 1.23787697]
 [0.87726904 0.53954816 1.         1.17195897 1.091912  ]
 [1.90322645 1.62191837 1.17195897 1.         1.58972858]
 [1.44436105 1.23787697 1.091912   1.58972858 1.        ]]
```

D. Calculate the covariance matrix using only overlapping data.

Simply use `dropna()` to only calculate the overlapping data:

```
Covariance matrix based on overlapping data:
          x1        x2        x3        x4        x5
x1  0.418604  0.394054  0.424457  0.416382  0.434287
x2  0.394054  0.396786  0.409343  0.398401  0.422631
x3  0.424457  0.409343  0.441360  0.428441  0.448957
x4  0.416382  0.398401  0.428441  0.437274  0.440167
x5  0.434287  0.422631  0.448957  0.440167  0.466272
```

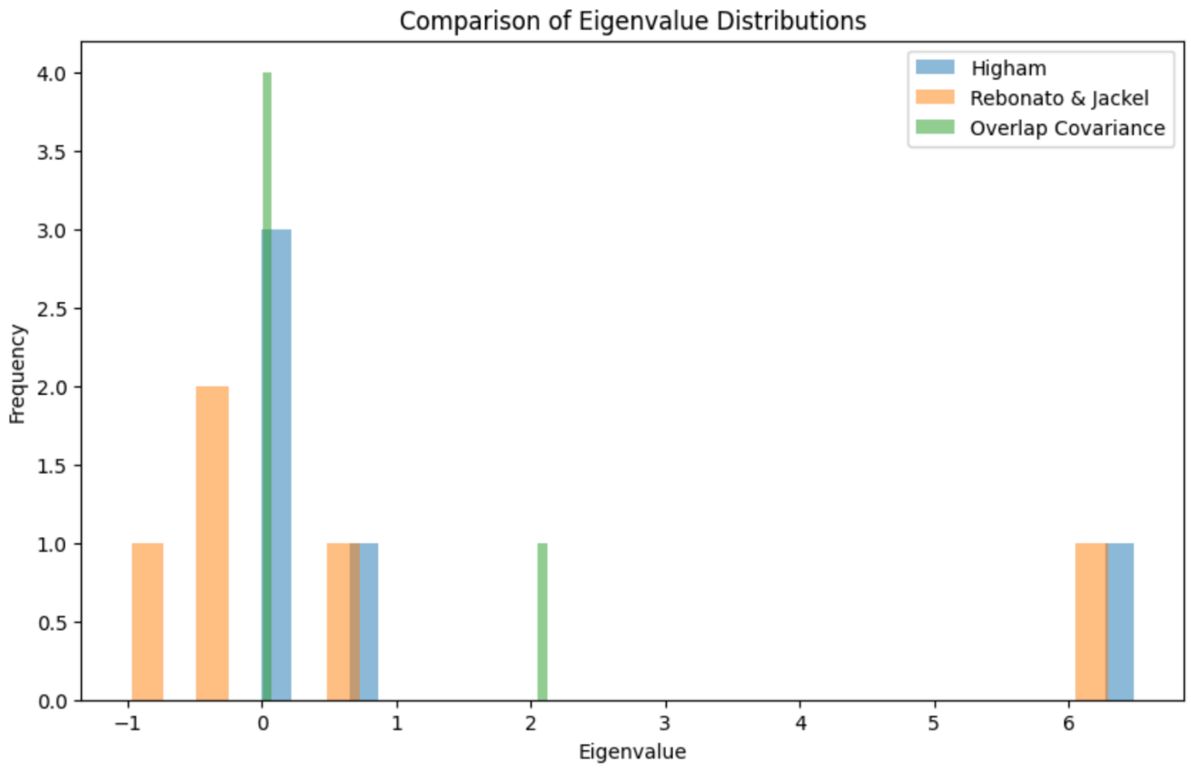
E. Compare the results of the covariance matrices in C and D. Explain the differences. Note: the generating process is a covariance matrix with 1 on the diagonals and 0.99 elsewhere.

1. **Firstly, we calculate Frobenius norm difference**

```
Higham vs Overlap Covariance Matrix: 4.484061725086761
Rebonato & Jackel vs Overlap Covariance Matrix: 4.434851539196421
Higham vs Rebonato & Jackel Covariance Matrix: 1.2070966654837034
```

- The difference between Higham and Overlap (4.484) is slightly larger than the difference between Rebonato & Jackel and Overlap (4.435): this indicates that the adjusted matrix of the Rebonato & Jackel method is closer to the covariance matrix calculated by Overlap. The possible reason is that the Rebonato & Jackel method corrects the PSD through SVD and is more inclined to maintain the numerical structure of the original matrix, while the Higham method is based on iterative optimization, which may lead to larger numerical changes.
- The difference between Higham's method and Rebonato & Jackel's method (1.207) is much smaller than their difference with Overlap: indicating that the matrices produced by these two PSD correction methods are relatively close, although they differ in correction strategies.

2. **Then we compute eigenvalue distribution**



- Higham method: The negative eigenvalues are close to 0 (the order of magnitude is 10−7), indicating that the Higham method has successfully corrected the negative eigenvalues to make them close to positive semidefinite. The maximum eigenvalue (6.4848) is slightly larger than the maximum eigenvalue (6.2960) of Rebonato & Jackel, indicating that the Higham method may cause the spectral radius of the matrix to increase and affect the numerical stability.
- Rebonato & Jackel method: There are still significant negative eigenvalues (-0.9718, -0.4859, -0.3913). This shows that the Rebonato & Jackel method fails to completely remove negative eigenvalues, even though it is closer in Frobenius error to the covariance matrix calculated by Overlap.
- Overlap method: All eigenvalues are positive, indicating that this matrix is positive definite. However, the maximum eigenvalue (2.1208) is much smaller than the maximum eigenvalue (6.4 and 6.3) of the Higham and Rebonato & Jackel methods, indicating that the covariance matrix calculated by the Overlap method has smaller variance explanation ability, which may be related to the way missing data is handled.

3. **For the condition number of the covariance matrix**

```
Condition Number (Higham): 8699459.77399493
Condition Number (Rebonato & Jackel): 16.088077147094193
Condition Number (Overlap): 509.9633017531094
```

- The condition number of Higham's method is extremely large (8699459.77): the condition number is the ratio of the largest and smallest eigenvalues of the matrix: $cond(A) = \frac{\lambda_{max}}{\lambda_{min}}$. Although Higham's method removes negative eigenvalues, it may overcorrect the PSD properties, resulting in a very small (but still positive) minimum eigenvalue, thereby greatly increasing the condition number. Such a high condition number means that the matrix value is unstable, which may lead to amplification of calculation errors.
- The Rebonato & Jackel method has a smaller condition number (16.08): This method does not strictly adjust the PSD, but only removes negative eigenvalues through SVD, but still retains a certain number of large negative eigenvalues. Since there are no extreme small eigenvalues, the condition number of the Rebonato & Jackel method is much lower than that of the Higham method, showing better numerical stability.
- The condition number calculated by Overlap is moderate (509.96): Although the Overlap method is calculated directly based on the data, its eigenvalues are smaller and the condition number is much smaller (more stable) than the Higham method. But compared to the Rebonato & Jackel method, it is still larger, possibly due to the uneven distribution of feature values due to missing data.

Problem 3

Given the data in problem3.csv

A. Fit a multivariate normal to the data.

Using **pandas** to compute the mean vector and covariance matrix:

$$\mu = \begin{bmatrix} 0.0460 \\ 0.0999 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 0.01016 & 0.00492 \\ 0.00492 & 0.02028 \end{bmatrix}$$

B. Given that fit, what is the distribution of X_2 given $X_1 = 0.6$. Use the 2 methods described in class.

For Multivariate Normal Distribution:

Given a bivariate normal distribution, the conditional mean and variance of X_2 given X_1 are:

$\mu_{X_2|X_1} = \mu_2 + \frac{\sigma_{12}}{\sigma_{11}}(X_1 - \mu_1)$ $\sigma^2_{X_2|X_1} = \sigma_{22} - \frac{\sigma^2_{12}}{\sigma_{11}}$

Substituting values from the covariance matrix, we get:

- **Conditional Mean**: **0.3683 Conditional Variance**: **0.0179**

For Ordinary Least Squares (OLS) Regression:

Since X_2 can be predicted from X_1 using a linear regression model:

$$X_2 = \beta_0 + \beta_1 X_1 + \epsilon$$

where $\beta_1 = \frac{\sigma_{12}}{\sigma_{11}}, \quad \beta_0 = \mu_2 - \beta_1 \mu_1$

Solving for X_2 when $X_1 = 0.6$, we obtain:

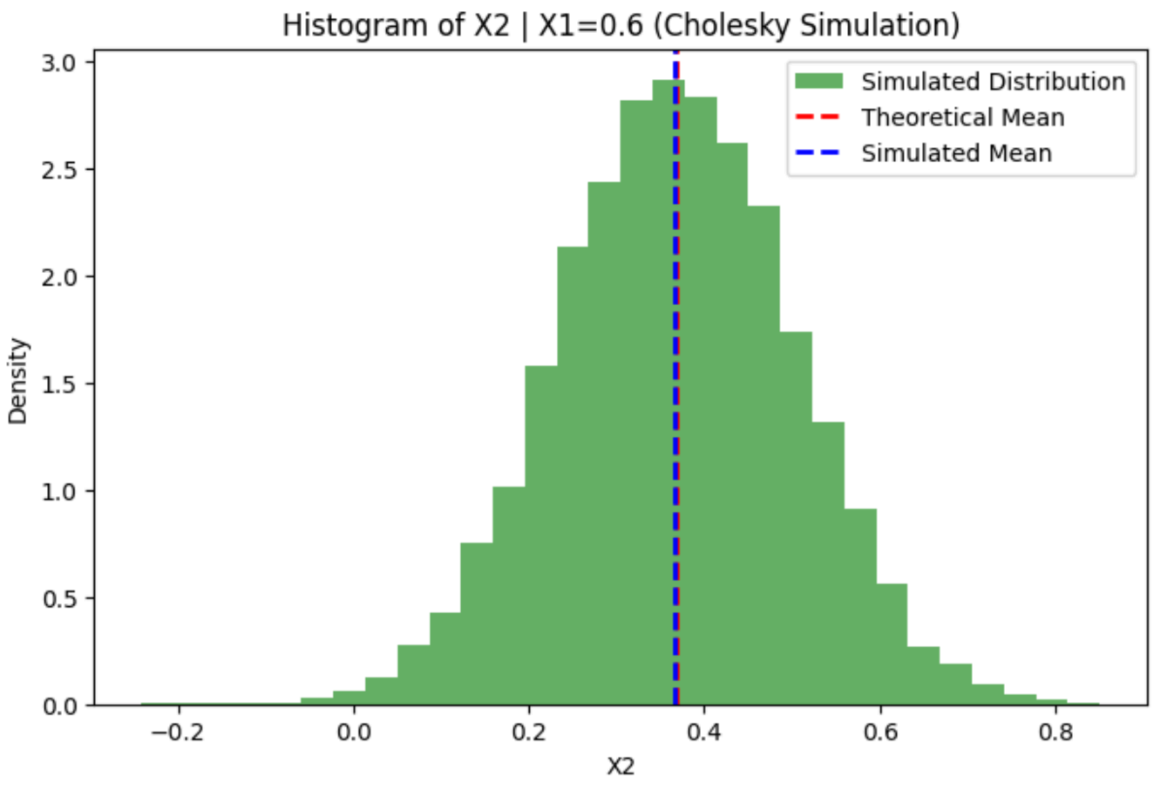
- **Conditional Mean**: **0.3683 Conditional Variance**: **0.0179**

Both methods yield the same results, confirming the correctness of our calculations.

###

C. Given the properties of the Cholesky Root, create a simulation that proves your distribution of $X_2 | X_1 = 0.6$ ***is correct.***

We generate **10,000** samples from a **bivariate normal distribution** using the Cholesky decomposition of Σ.



Fixing $X_1 = 0.6$, we compute the simulated values of X_2:

- **Simulated Conditional Mean**: **0.3672 Simulated Conditional Variance**: **0.0180**

The simulated mean and variance closely match our analytical results, confirming that the conditional distribution follows the expected theoretical values.

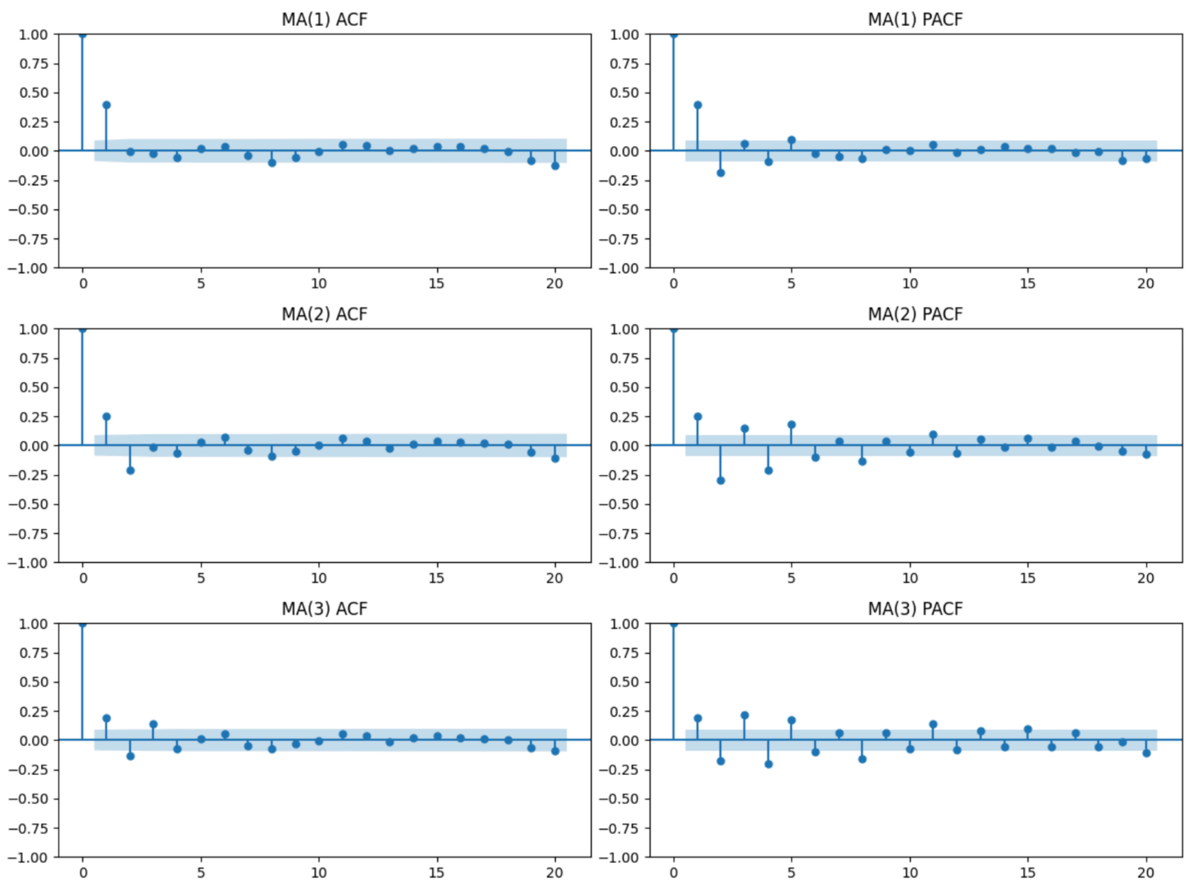
Using both theoretical formulas and regression analysis, we found that the conditional mean of X_2 given $X_1 = 0.6$ is **0.3683**, and the conditional variance is **0.0179**. The simulation results validate these findings, demonstrating the consistency of our approach.

Problem 4

Given the data in problem4.csv

A. Simulate an MA(1), MA(2), and MA(3) process and graph the ACF and PACF of each. What do you notice?

I smulated Moving Average (MA) processes with orders 1, 2, and 3, and plotted their Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF).

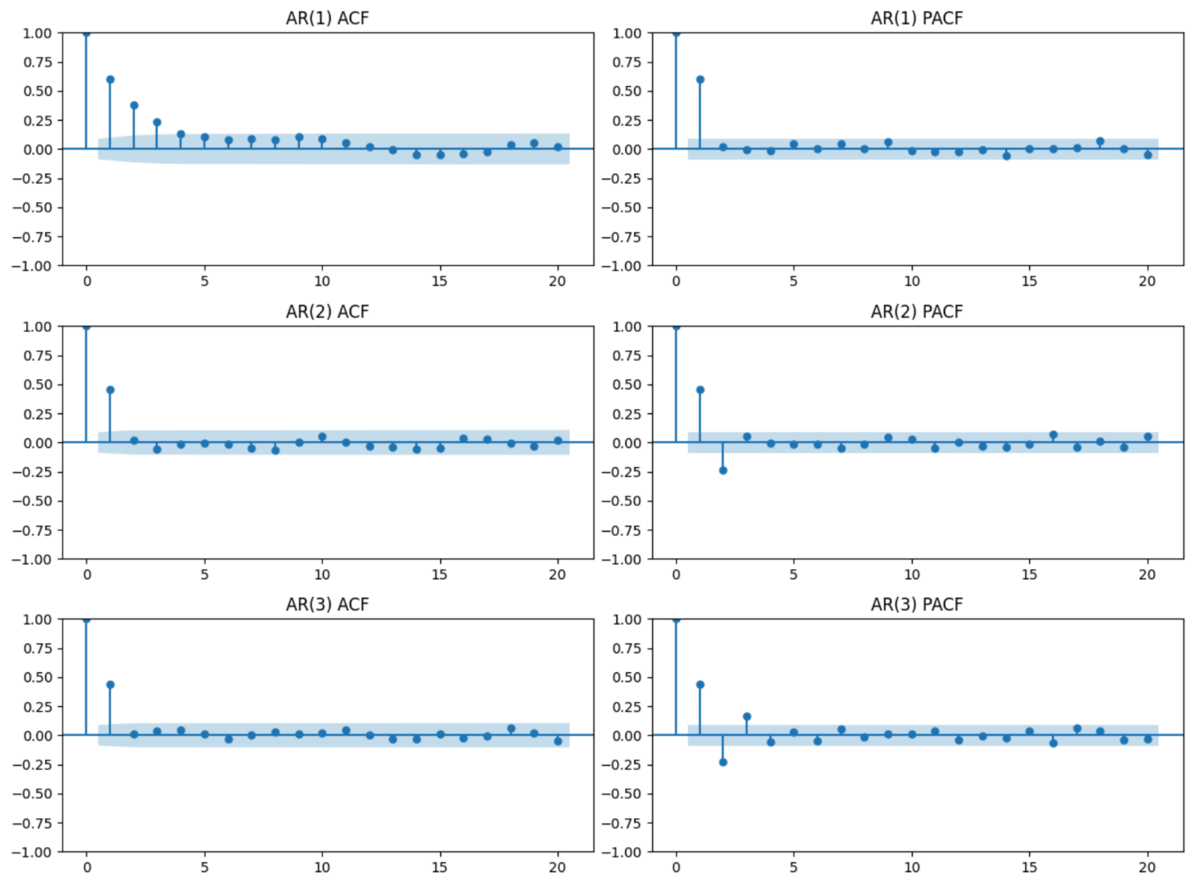


- **MA(1):** ACF shows a single significant lag, while PACF cuts off after lag 1.
- **MA(2):** ACF has two significant lags before cutting off, PACF still exhibits a sharp cutoff.
- **MA(3):** ACF extends up to lag 3 before dropping off, PACF shows three significant lags before cutting off.

These results are consistent with the theoretical behavior of MA processes, where ACF has a clear cutoff at order q, while PACF exhibits a gradual decay.

B. Simulate an AR(1), AR(2), and AR(3) process and graph the ACF and PACF of each. What do you notice?

I simulated Autoregressive (AR) processes with orders 1, 2, and 3, and plotted their ACF and PACF.



- **AR(1):** PACF shows a single sharp cutoff at lag 1, while ACF exhibits a gradual decay.
- **AR(2):** PACF has two significant lags before cutting off, while ACF decays more gradually.
- **AR(3):** PACF extends to lag 3 before cutting off, while ACF shows a prolonged decay.

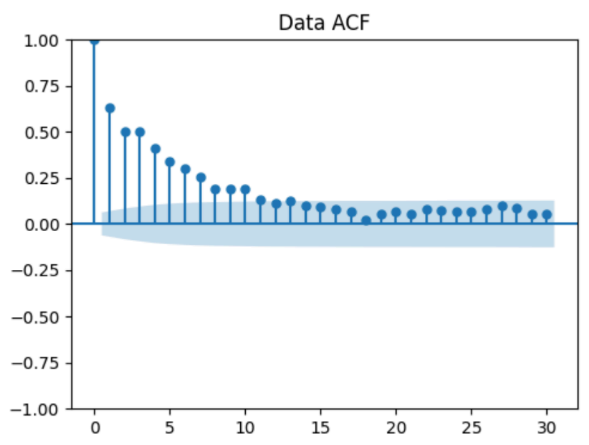
This aligns with the expected behavior of AR models, where PACF has a distinct cutoff at order p, while ACF decays slowly.

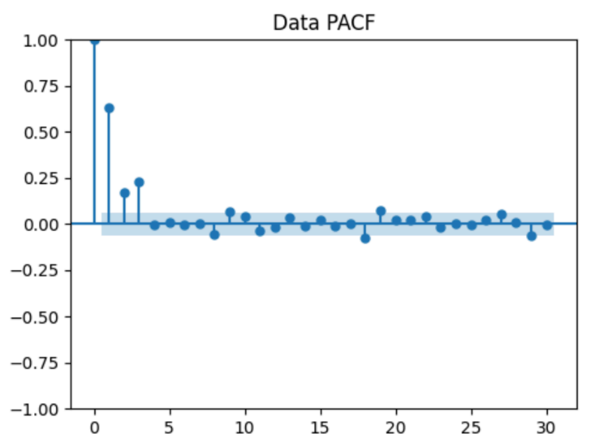
C. Examine the data in problem4.csv. What AR/MA process would you use to model the data? Why?

We analyzed the **ACF and PACF of the given dataset** to determine whether an AR or MA model would be a better fit.

We conducted an ADF test to check stationarity:

The **p-value** from the ADF test suggests the data is stationary.





If **ACF exhibits a sharp cutoff while PACF decays gradually**, an **MA(q) model** is preferred.

If **PACF exhibits a sharp cutoff while ACF decays gradually**, an **AR(p) model** is preferred.

Based on our ACF/PACF analysis, we identified an **AR(p) model** as the most appropriate choice.

D. Fit the model of your choice in C along with other AR/MA models. Compare the AICc of each. What is the best fit?

We fit AR(1), AR(2), AR(3) and MA(1), MA(2), MA(3) models, calculated the Corrected Akaike Information Criterion (AICc) for each model to balance goodness-of-fit and model complexity.

```
Model AICc values:
AR(1): -1669.0852593293716
AR(2): -1696.079649387366
AR(3): -1746.2576245169637
MA(1): -1508.9230252238815
MA(2): -1559.238895762558
MA(3): -1645.1088725670288

The best model is AR(3) with AICc = -1746.2576245169637
```

The AR(3) is the best model we fit.

Problem 5

Given the stock return data in DailyReturns.csv.

A. Create a routine for calculating an exponentially weighted covariance matrix. If you have a package that calculates it for you, verify it produces the expected results from the testdata folder.

Given the stock return data in `DailyReturns.csv`, we aim to calculate the EWMA.

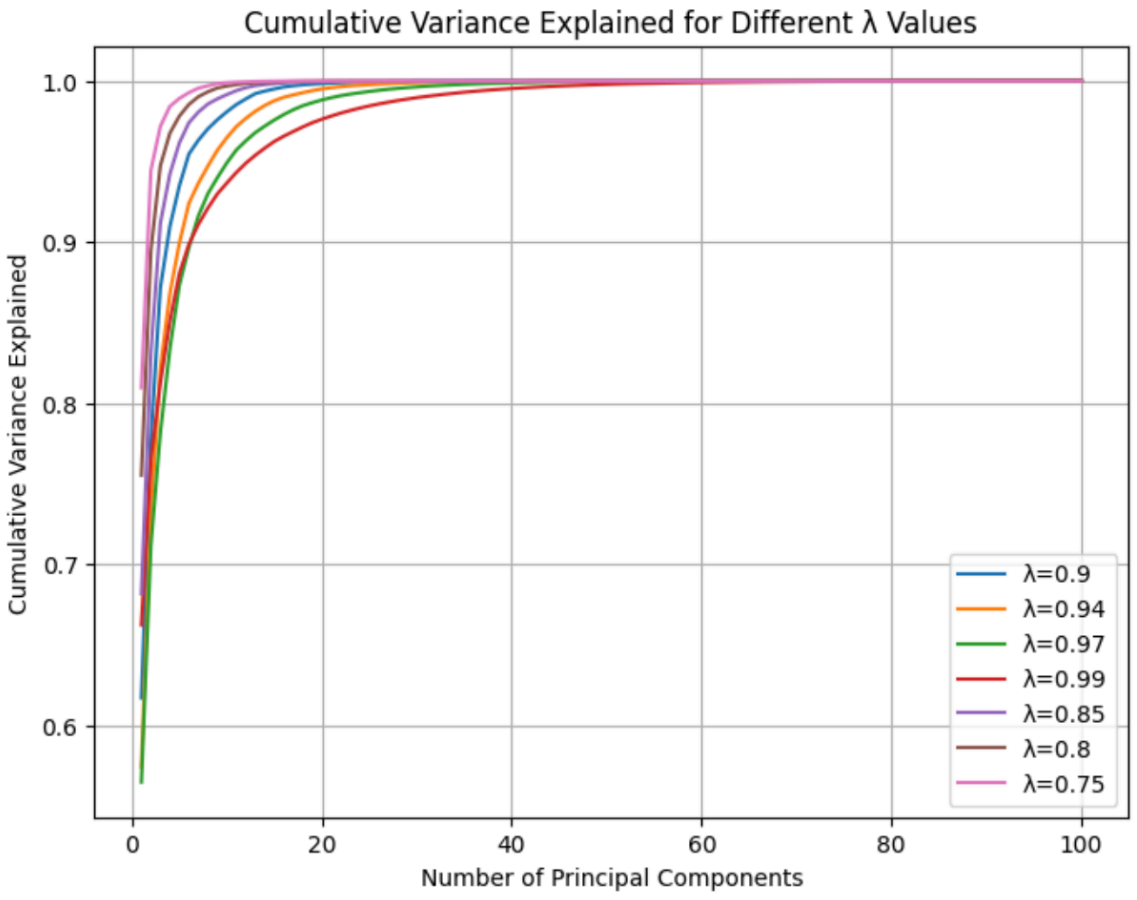
1. Read stock return data from `DailyReturns.csv`.
2. Compute deviations from the mean.
3. Apply exponential weighting to derive the EWMA covariance matrix.

4. Validate results using test data from the `testdata` folder.

We define a function called `ewma_cov_matrix()` that calculates the EWMA covariance matrix, and another function called `pandas_ewma_cov_matrix` which use panda's function `ewm()` and we using both functions to compute the same data and compare the final result.

B. Vary λ. Use PCA and plot the cumulative variance explained of λ in (0,1) by each eigenvalue for each λ chosen.

We choose λ range from 0.99 to 0.8 to plot the cumulative variance explained for the certain values.



C. What does this tell us about the values of λ and the effect it has on the covariance matrix?

To examine 's effect on the covariance matrix:

1. Compute EWMA covariance matrices for different values.
2. Perform **Principal Component Analysis (PCA)**.
3. Plot **cumulative variance explained** for each .

As we done it above, so for value λ:

Higherλ values (e.g., 0.99) → Older data retains more influence, leading to a **more stable covariance matrix** but potentially carrying outdated information.

Lower λ values (e.g., 0.75) → More emphasis is placed on recent data, making the covariance matrix **more responsive to market changes**, but also **more volatile**.

Problem 6

Implement a multivariate normal simulation using the Cholesky root of a covariance matrix. Implement a multivariate normal simulation using PCA with percent explained as an input. Using the covariance matrix found in problem6.csv

A. Simulate 10,000 draws using the Cholesky Root method.

Using the formula: $X_s = L@Z + \mu$

B. Simulate 10,000 draws using PCA with 75% variance

Using the formula: $X_s = B \times (diag(P_{stdev})) \times Z + \mu$

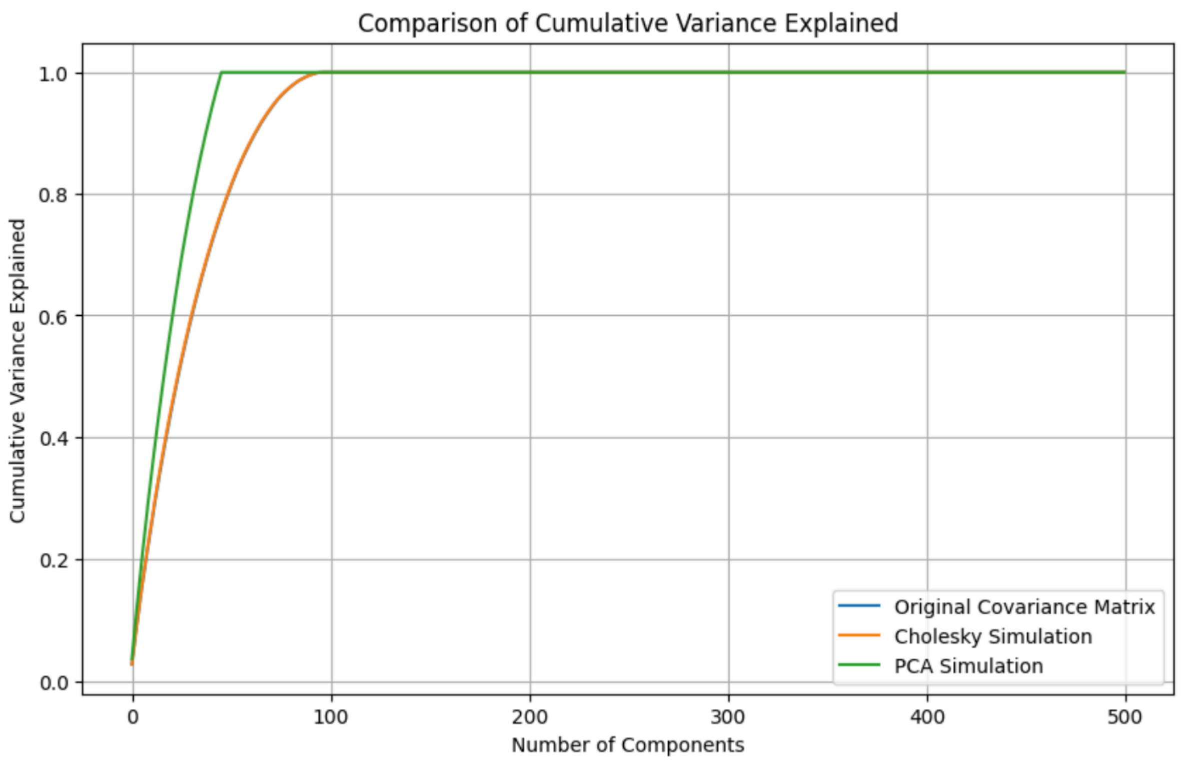
C. Take the covariance of each simulation. Compare the Frobenius norm of these matrices to the original covariance matrix. What do you notice?

The Frobenius norm of the difference between the original covariance matrix and the Cholesky simulated covariance matrix is **0.0207**, while the Frobenius norm of the difference between the original covariance matrix and the PCA simulated covariance matrix is **0.3267**.

The Cholesky method produces a covariance matrix that is much closer to the original covariance matrix compared to the PCA method. This is expected because the Cholesky method directly uses the covariance matrix for simulation, while the PCA method approximates the covariance matrix by retaining only a portion of the variance (75% in this case).

D. Compare the cumulative variance explained by each eigenvalue of the 2 simulated covariance matrices along with the input matrix. What do you notice?

The cumulative variance explained by the eigenvalues of the original covariance matrix, the Cholesky simulated covariance matrix, and the PCA simulated covariance matrix was plotted and compared.



- The cumulative variance explained by the Cholesky simulated covariance matrix closely follows that of the original covariance matrix, indicating that the Cholesky method preserves the variance structure of the original data.
- The PCA simulated covariance matrix, however, reaches 100% cumulative variance much faster, as it only retains 75% of the variance. This results in a less accurate representation of the original covariance structure compared to the Cholesky method.

E. Compare the time it took to run both simulations.

- **Cholesky Simulation Execution Time: 1.4394 seconds**
- **PCA Simulation Execution Time: 0.7967 seconds**

The PCA simulation was faster than the Cholesky simulation. This is because the PCA method reduces the dimensionality of the data by retaining only the principal components that explain 75% of the variance, which reduces the computational complexity compared to the Cholesky method.

F. Discuss the tradeoffs between the two methods.

- **Cholesky Method:**

  Pros:

  - Produces a covariance matrix that is very close to the original, preserving the full variance structure.
  - More accurate for simulations where the exact covariance structure is important.

  Cons:

  - Computationally more expensive, especially for large matrices.
  - Requires the covariance matrix to be positive definite.
- **PCA Method:**

Pros:

- Faster computation due to dimensionality reduction.
- Useful when only a portion of the variance is needed, reducing noise and focusing on the most important components.

Cons:

- Less accurate in preserving the original covariance structure, especially when a significant portion of the variance is discarded.
- May not be suitable for applications requiring the full covariance structure.