Instructions:
Be verbose. Explain clearly your reasoning, methods, and results in your written work. Write clear code that is well documented. With 99% certainty, you cannot write too many code comments.

Written answers are worth 18 points. Code is worth 2 points. 10 points total.

1. When finished, respond to the question in Canvas as “done.” We will record your grade there.
2. In your code repository, create a folder called “Project01.”
3. In that folder, include
    a. a document (PDF) with your responses.
    b. All code
    c. A README file with instructions for us to run your code

Everything must be checked into your repository by 8am Saturday 2/8. A pull will be done at that time. Documents and code checked in after the instructors pull will not be graded.

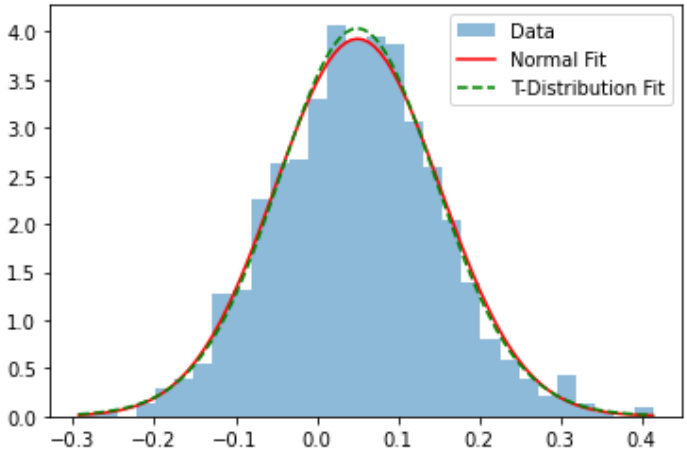
Data for problems can be found in CSV files with this document in the class repository.

Problem 1
Given the dataset in problem1.csv

A. Calculate the Mean, Variance, Skewness and Kurtosis of the data
   Mean: 0.0502
   Variance: 0.0103
   Skewness: 0.1206
   Kurtosis: 0.2301

B. Given a choice between a Normal Distribution and a T-Distribution, which one would you choose to model the data? Why?
   The data has slightly heavier tails (positive excess kurtosis = 0.23) compared to a normal distribution (kurtosis = 0). The T-distribution better accommodates this tail heaviness, making it more robust for modeling uncertainty, especially if the sample size is small. The near-zero skewness (0.12) aligns with both distributions, but the kurtosis tips the choice to T.

C. Fit both distributions and prove or disprove your choice in B using methods presented in class.
   The heavier tails (from kurtosis = 0.23) are not extreme enough to justify the added complexity.
   The data is close to normal, and the T-distribution’s extra parameter causes overfitting.



Problem 2

Given the data in problem2.csv

A. Calculate the pairwise covariance matrix of the data.

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?

After calculated Eigenvalues, I got

[6.78670573, 0.83443367, -0.31024286, 0.02797828, -0.13323183].

The matrix is **not positive semi-definite** because it has negative eigenvalues (-0.3102 and -0.1332). For a matrix to be positive semi-definite, all eigenvalues must be ≥ 0. The presence of negative eigenvalues violates this condition.

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

Higham method:

Nearest PSD Covariance Matrix:

	0	1	2	3	4
0	1.470484	1.332361	0.884378	1.627602	1.399556
1	1.332361	1.252078	0.619028	1.450604	1.214450
2	0.884378	0.619028	1.272425	1.076847	1.059658
3	1.627602	1.450604	1.076847	1.814469	1.577928
4	1.399556	1.214450	1.059658	1.577928	1.396186

Rebenato and Jackel:

	0	1	2	3	4
0	1.470484	1.327009	0.842583	1.624464	1.364833
1	1.327009	1.252078	0.555421	1.433109	1.165906
2	0.842583	0.555421	1.272425	1.052789	1.060424
3	1.624464	1.433109	1.052789	1.814469	1.544993
4	1.364833	1.165906	1.060424	1.544993	1.396186

The negative eigenvalues indicate the original matrix was not a valid covariance matrix. By applying Higham and Rebenato and Jackel, we obtain a nearest positive semidefinite matrix that corrects those negative eigenvalues with minimal change to

the original entries. This ensures we can use it confidently as a legitimate covariance matrix in our analyses.

D. Calculate the covariance matrix using only 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.

Both covariance matrices are attempting to estimate the same “highly correlated” structure (with true diagonals of 1 and off-diagonals of 0.99), but they differ because they use different data and/or constraints. The Higham method, which adjusts a covariance matrix to be positive semidefinite, yields generally larger values (some diagonals above 1 and large off-diagonal entries) due to the way it corrects and stabilizes the covariance estimates while preserving the near-unity correlation structure. In contrast, the matrix computed from only overlapping data (i.e., rows with no missing values) uses a smaller effective sample and naturally produces smaller numerical estimates (around 0.4–0.46 on the diagonal). Hence, the two matrices differ because the Higham method enforces positive semidefiniteness on a larger or more “complete” (possibly imputed) set of data, while the overlapping-data approach restricts itself strictly to rows without missing values—ultimately leading to smaller empirical variances and covariances.

Problem 3

Given the data in problem3.csv

A. Fit a multivariate normal to the data.

```
array([[0.0101622 , 0.00492354],
       [0.00492354, 0.02028441]])
```

B. Given that fit, what is the distribution of X2 given X1=0.6. Use the 2 methods described in class.

```
Method 1 - Conditional Mean of X2 |
X1=0.6: 0.3683249958609775
Method 1 - Conditional Variance of X2
| X1=0.6: 0.017898969645087522
```

```
Method 2 - Conditional Mean of X2 |
X1=0.6: 0.3683249958609775
Method 2 - Conditional Variance of X2
| X1=0.6: 0.017898969645087522
```

C. Given the properties of the Cholesky Root, create a simulation that proves your distribution of X2 | X1=0.6 is correct.

```
Simulated mean of X2 | X1=0.6:
0.36994915105621523
Simulated variance of X2 | X1=0.6:
0.018505753046915128
```

These results show agreement between the theoretical conditional distribution (using the standard formula for a bivariate normal) and the simulation via the Cholesky approach. The small difference between the two conditional means (and variances) is within what we'd expect from sampling variability.

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?

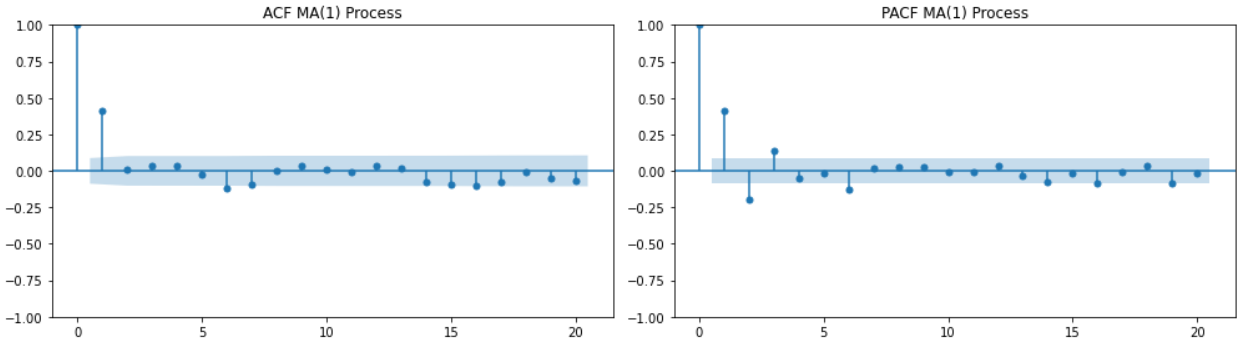
MA(1)

ACF:

- For an MA(1) process, the ACF is nonzero at lag 1 and zero for all lags greater than 1 (assuming a purely MA(1) with no AR parts). I see spike at lag 1, then values near zero (within the confidence intervals) afterward. The first autocorrelation will reflect the sign and magnitude of the MA(1) parameter

PACF:

- The partial autocorrelation function for an MA process does *not* have a strict cutoff. In an MA(1), the PACF *tapers off* slowly (can be oscillatory or gradually decaying) rather than cutting off after 1 lag. I see a more gradual decline or weaker pattern beyond lag 1, with no clear cutoff.



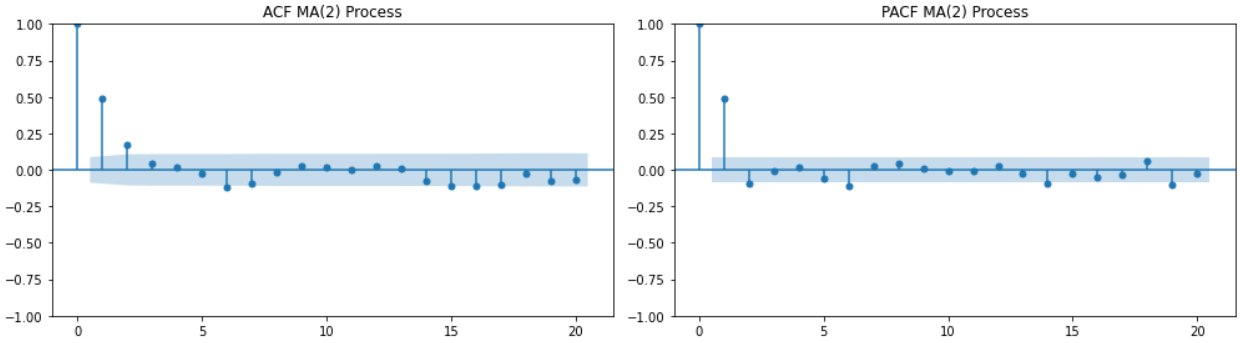
MA(2)

ACF:

- an MA(2) process, the ACF is nonzero at lags 1 and 2, then zero for lags > 2.
- A significant spikes at lags 1 and 2, and afterward the autocorrelations tend to hover around zero (within confidence bounds).

PACF:

- As with MA processes in general, the PACF does not have a strict cutoff. It can show *decaying* (and sometimes oscillatory) behavior beyond lag 2.
- A bit of a pattern in the first few lags, but there's no sharp cutoff as you would observe in an AR process.

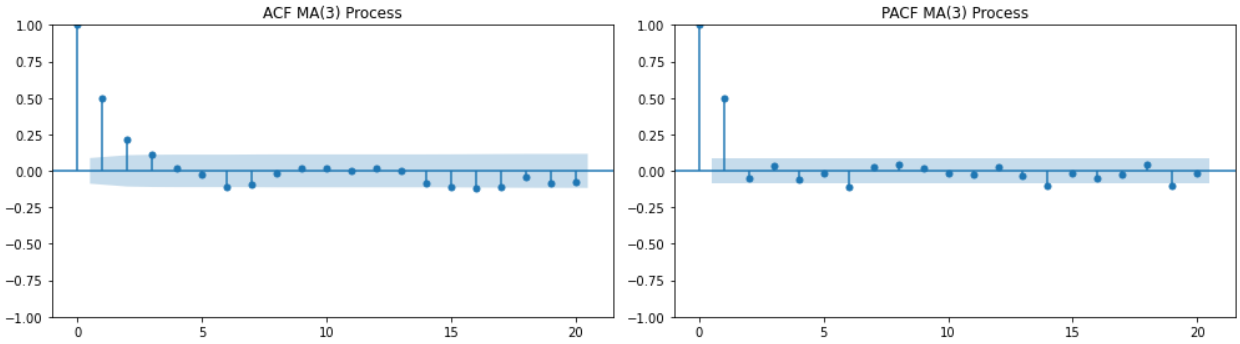


MA(3)

ACF:

- For an MA(3) process, the ACF is nonzero at lags 1, 2, and 3, then zero for lags > 3.
- A clear spikes at lags 1, 2, and 3. The values beyond lag 3 should be near zero (within the confidence interval).

- **PACF**:
  - Similar to other MA processes, the PACF has no strict cutoff. The influence of the three MA terms can cause more complex behavior in the partial autocorrelations.
  - The partial autocorrelations beyond lag 3 exhibit a gradual decay or weaker oscillations, but do not truncate.



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

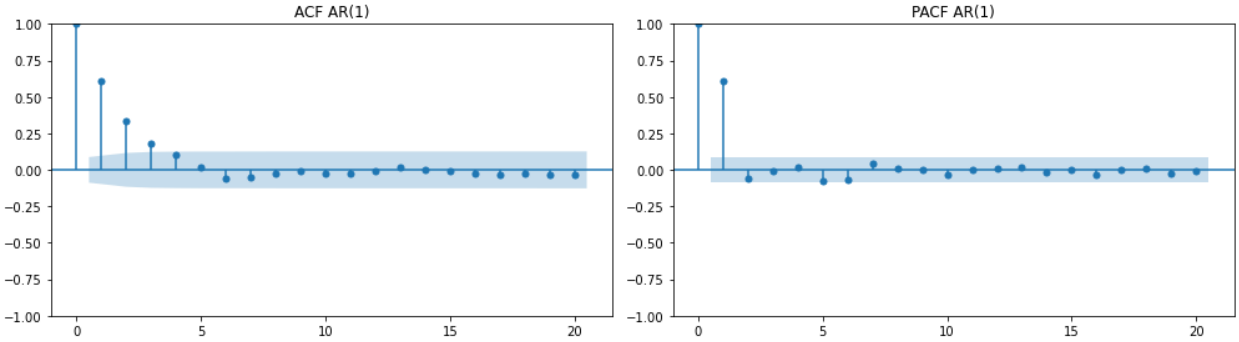
AR(1)

ACF:

- For a stationary AR(1), the ACF typically *decays exponentially* as lag increases.
- There is no “hard cutoff” in the ACF.

PACF:

- The **partial** autocorrelation function cuts off after **lag 1**. A significant spike at lag 1 in the PACF and the rest near zero.
- This is a hallmark of an AR(1).



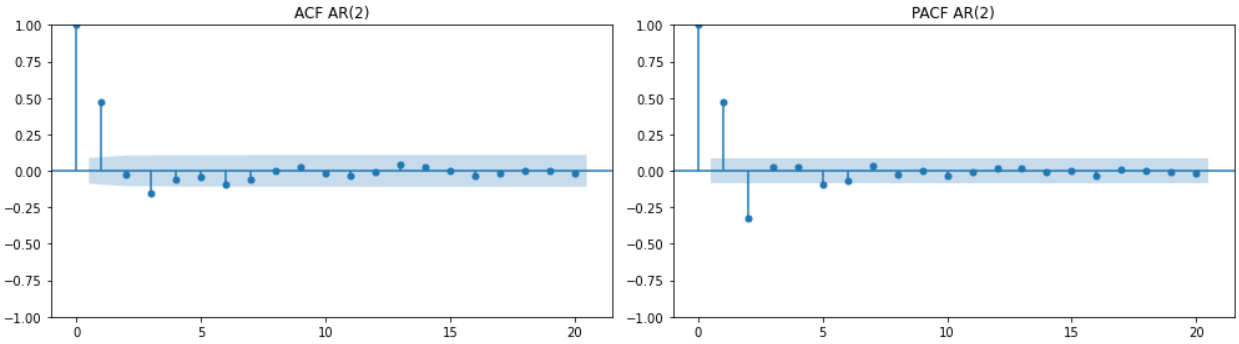
AR(2)

ACF:

- Still no hard cutoff, but the pattern can be more complex—often an exponential or damped oscillation, depending on the signs and magnitudes of phi_1 and phi_2.
- Typically, it *tapers* to zero if the process is stationary.

PACF:

- Now, the partial autocorrelation will typically have **significant spikes at lags 1 and 2** and then cut off for lags > 2.



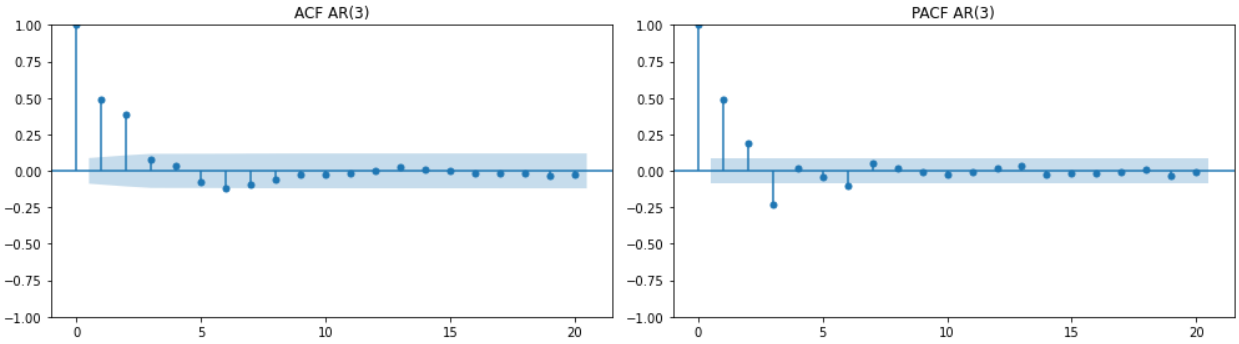
AR(3)

ACF:

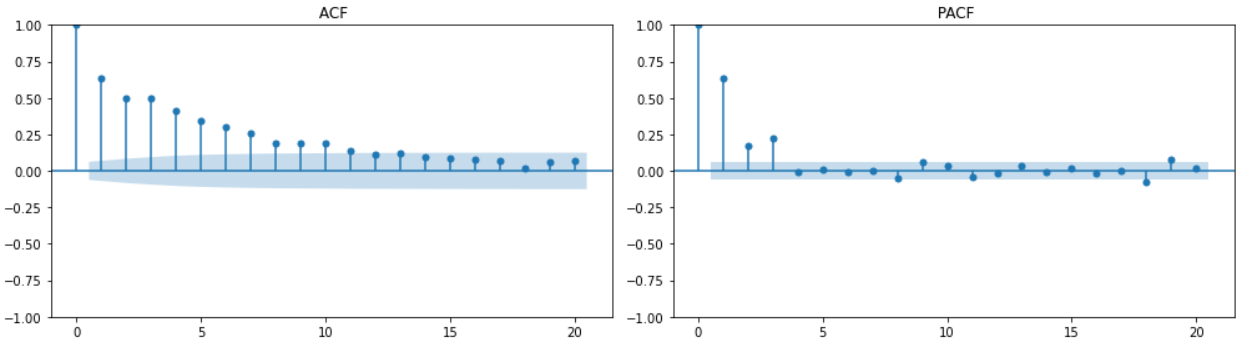
- Again, no hard cutoff. The shape can be even more varied (exponential decay, oscillatory, etc.) depending on phi_1, phi_2, phi_3.

PACF:

- The partial autocorrelation shows a cutoff at lag 3, meaning a significant partial autocorrelations at lags 1, 2, and 3, and then near zero afterward.



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



By looking at the PACF, AR(3) model is suggested because the PACF cuts off after lag 3, indicating an autoregressive process of order 3.

The absence of a sharp cutoff in the ACF suggests that an MA (Moving Average) model is less appropriate.

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?

```
In [64]: get_ar_aic_table(y)
Out[64]:
[(0, -1162.397518025169),
 (1, -1669.0892673454014),
 (2, -1696.0916854957047),
 (3, -1746.2817209024197),
 (4, -1744.3080737231003),
 (5, -1742.3619303876226),
 (6, -1740.3799125533487),
 (7, -1738.3833076063675),
 (8, -1739.0856403925172),
 (9, -1741.2976446170542),
 (10, -1741.0140814570725)]

In [65]: get_ma_aic_table(y)
Out[65]:
[(0, -1162.397518025169),
 (1, -1508.9270332411106),
 (2, -1559.2509318714744),
 (3, -1645.132968953476),
 (4, -1677.5846235717204),
 (5, -1703.2568220903954),
 (6, -1713.1689201145612),
 (7, -1729.159051822408),
 (8, -1727.5952281415014),
 (9, -1727.2818196509304),
 (10, -1734.1367557508256)]
```

Best AR model: AR(3) with AIC = -1746.28
Best MA model: MA(10) with AIC = -1734.14
AR(3) is preferred based on AIC (-1746.28 vs. -1734.14).

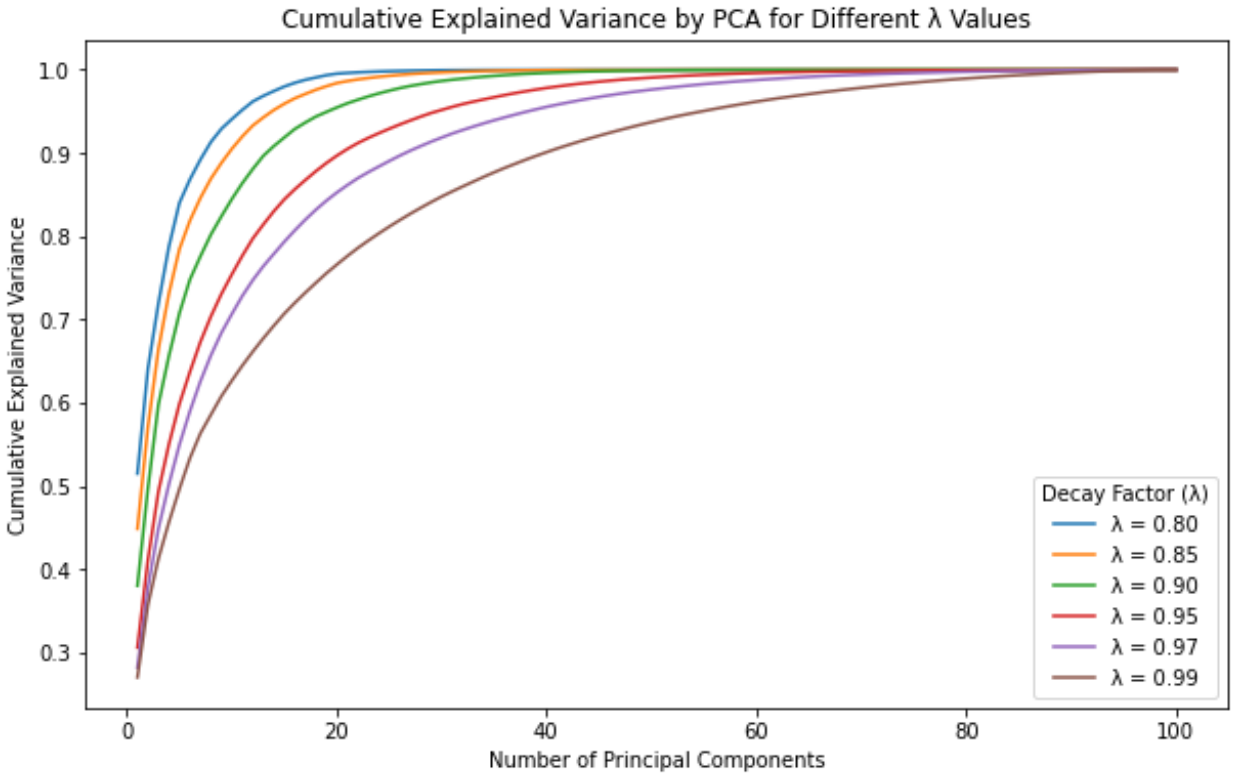
Problem 5

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.

	SPY	AAPL	NVDA	...	PLD	LRCX	EQIX
SPY	0.000072	0.000054	0.000124	...	0.000059	0.000127	0.000053
AAPL	0.000054	0.000139	0.000041	...	0.000060	0.000084	0.000038
NVDA	0.000124	0.000041	0.000663	...	0.000014	0.000322	0.000050
MSFT	0.000080	0.000084	0.000133	...	0.000061	0.000152	0.000054
AMZN	0.000112	0.000081	0.000196	...	0.000057	0.000185	0.000071
...	...	...	...	...	...	...	...
KKR	0.000135	0.000041	0.000220	...	0.000065	0.000193	0.000098
MU	0.000148	0.000055	0.000304	...	0.000156	0.000707	0.000085
PLD	0.000059	0.000060	0.000014	...	0.000250	0.000084	0.000088
LRCX	0.000127	0.000084	0.000322	...	0.000084	0.000738	0.000073
EQIX	0.000053	0.000038	0.000050	...	0.000088	0.000073	0.000153

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



C. What does this tell us about the values of λ and the effect it has on the covariance matrix?
Smaller λ (e.g., 0.80):
More emphasis on recent returns.
The first few principal components explain a larger fraction of the variance quickly (steep early slope).
This suggests a more "concentrated" covariance structure dominated by the most recent shocks/correlations.

Larger λ (e.g., 0.99):
Greater weighting of older observations.
The variance is spread more evenly across more principal components (the curve is more gradually rising).
Need more components to reach, say, 90% of variance explained, indicating a more "balanced" correlation structure shaped by both recent and older data.

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.
The data was not least positive semi-definite, so I used Higham's method from Problem 2.

Below is head of my 10,000 simulations using the Cholesky Root method:

```
In [78]: df_cholesky
Out[78]:
           0         1         2    ...       497       498       499
0     0.115349  0.027411 -0.035906  ...  0.066805  0.041961 -0.007720
1     0.038340 -0.072320  0.092948  ... -0.031124 -0.011756 -0.047689
2     0.016110  0.010443  0.191807  ... -0.048807  0.022908  0.017099
3     0.063678 -0.043339 -0.050048  ... -0.013193 -0.054124 -0.006741
4     0.004707 -0.013644  0.010012  ... -0.039538 -0.013448 -0.051726
...        ...       ...       ...  ...       ...       ...       ...
9995  0.108646  0.025423 -0.124792  ... -0.035919  0.030548 -0.016422
9996 -0.006163  0.040683 -0.024801  ... -0.027036 -0.026567 -0.009605
9997 -0.027701  0.009133  0.016751  ...  0.033538  0.073390  0.056418
9998  0.024363  0.043775  0.030760  ...  0.019219  0.010847 -0.019896
9999 -0.066892  0.031035  0.033692  ... -0.038950 -0.016853  0.010248
```

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

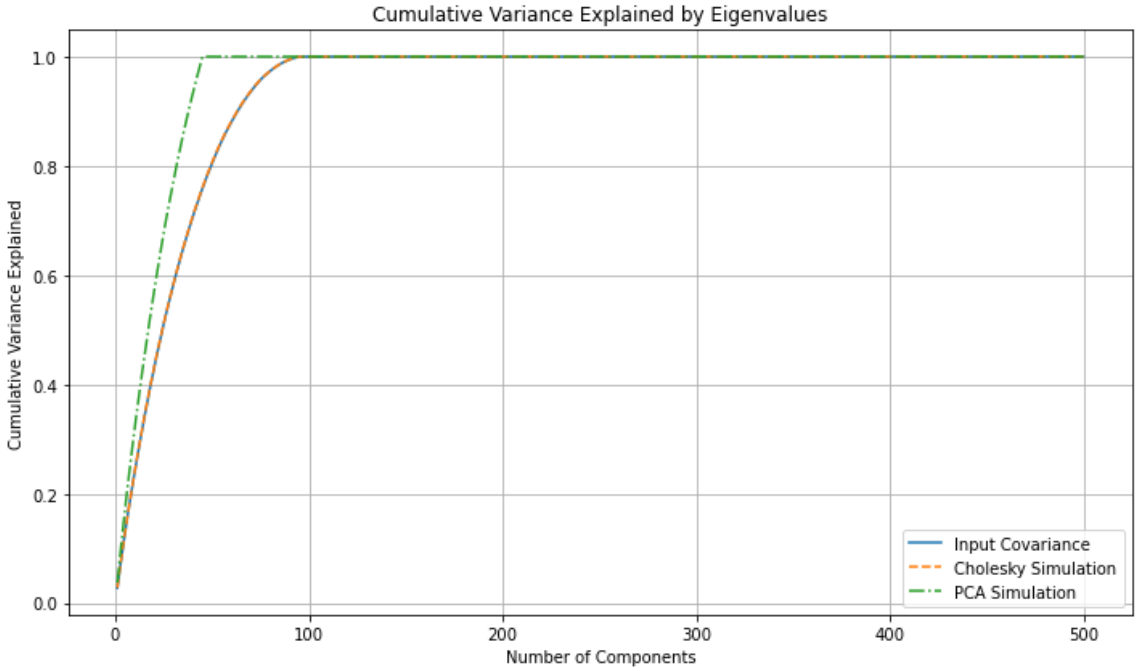
```
In [81]: df_pca
Out[81]:
           0         1         2    ...       497       498       499
0    -0.058977 -0.024816 -0.061999  ... -0.034337 -0.076362 -0.017062
1    -0.016817 -0.049182  0.092177  ...  0.065319 -0.015101 -0.013192
2     0.044460 -0.007828  0.063570  ... -0.001349 -0.016906  0.009171
3    -0.029720 -0.055090 -0.023395  ...  0.012653  0.013226  0.036388
4     0.017867 -0.079807 -0.000827  ... -0.024445 -0.110716 -0.005377
...        ...       ...       ...  ...       ...       ...       ...
9995  0.000381  0.036772 -0.088455  ... -0.009040 -0.053882  0.019209
9996 -0.101447 -0.007997 -0.002595  ...  0.040044 -0.032837  0.004236
9997 -0.049137 -0.028911 -0.050730  ...  0.002695 -0.020654 -0.028367
9998 -0.035361  0.036877 -0.052651  ...  0.053506 -0.020037  0.022217
9999 -0.081205 -0.029714  0.026700  ...  0.010955  0.017057 -0.002599
```

C. Take the covariance of each simulation. Compare the Frobenius norm of these matrices to the original covariance matrix. What do you notice?
Frobenius norm of the original covariance matrix: 0.2603
Frobenius norm difference (Cholesky simulation): 0.0218
Frobenius norm difference (PCA simulation): 0.0831

Cholesky approach is an exact decomposition of the covariance matrix (assuming positive definiteness). When simulating from a multivariate normal using the Cholesky factor, we are effectively preserving the full correlation structure exactly in theory (up to numerical precision and sampling variation).
As a result, we would expect any discrepancy from the original covariance to be quite small.

On the other hand, PCA approach simulation often involves truncation of the smaller principal components (or at least downweighting them).
By focusing on the principal directions of maximum variance, PCA can lose some of the finer structure in the covariance.
Consequently, the reconstructed or “simulated” covariance from PCA tends to be an approximation that can show larger deviations from the original matrix.

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



From the plot, the Cholesky‐simulated covariance matches the original’s eigenvalue spectrum more closely (reflected by its smaller Frobenius‐norm difference). In contrast, the PCA‐based simulation concentrates variance into fewer components (the green curve rises more steeply) and ends up farther from the original overall (larger Frobenius‐norm difference).

E. Compare the time it took to run both simulations.
Cholesky method took 0.15450 seconds.
PCA method took 0.01850 seconds.
PCA method runs faster.

F. Discuss the tradeoffs between the two methods.

Cholesky Factorization

Accuracy: Preserves the exact covariance structure (in theory), yielding very small deviations from the original matrix.

Complexity: Computationally more expensive for larger dimensions.

Use Case: Ideal when you need a highly accurate simulation of the covariance structure and can afford the computational cost.

PCA-Based Simulation

Speed: Runs faster, especially for high-dimensional data, because it can truncate or downweight smaller principal components.

Approximation: By focusing on the largest principal components, it loses some fine-grained covariance details, leading to larger deviations from the original matrix.

Use Case: Suitable when computational speed or dimensionality reduction is a priority and a small approximation error is acceptable.