**GROUP WORK PROJECT # ___** MScFE 600: FINANCIAL DATA

**GROUP NUMBER:** 11303

| FULL LEGAL NAME | LOCATION (COUNTRY) | EMAIL ADDRESS | MARK X FOR ANY NON-CONTRIBUTING MEMBER |
|---|---|---|---|
| Thulani Doctor Mathonsi | South Africa | MathonsiThulani008@gmail.com | |
| Amna Ishfaq Ishfaq | N/A | N/A | X |
| Emmanuel Oluwasegun Ismaila | Nigeria | ismailaemmanueloluwasegun@gmail.com | |

**Statement of integrity:** By typing the names of all group members in the text boxes below, you confirm that the assignment submitted is original work produced by the group (excluding any non-contributing members identified with an "X" above).

| | |
|---|---|
| **Team member 1** | **Thulani Doctor Mathonsi** |
| **Team member 2** | **Emmanuel Oluwasegun Ismaila** |
| **Team member 3** | |

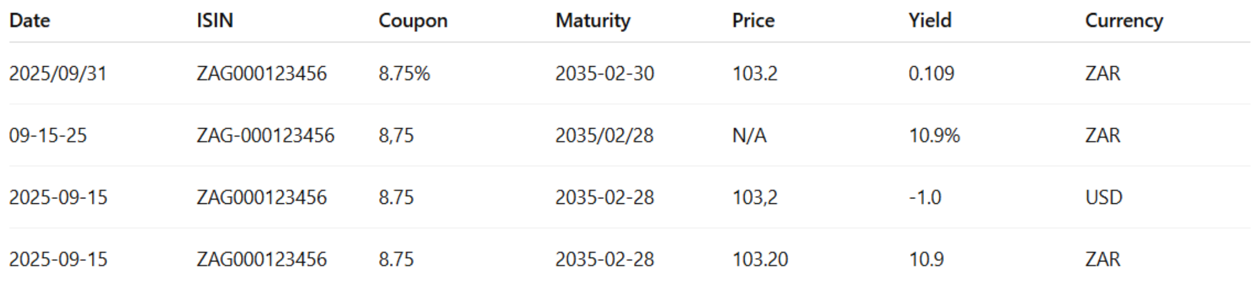
Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed.
**Note:** You may be required to provide proof of your outreach to non-contributing members upon request.

We tried to reach out to each using Discussion Group Forum on M5.

**GROUP WORK PROJECT # ___**
**Group Number:** 11303

# Task 1: Data Quality

### a.) Example of poor quality structured data



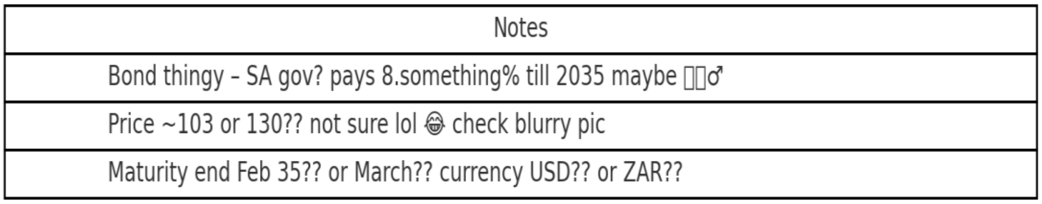
| Date | ISIN | Coupon | Maturity | Price | Yield | Currency |
|---|---|---|---|---|---|---|
| 2025/09/31 | ZAG000123456 | 8.75% | 2035-02-30 | 103.2 | 0.109 | ZAR |
| 09-15-25 | ZAG-000123456 | 8,75 | 2035/02/28 | N/A | 10.9% | ZAR |
| 2025-09-15 | ZAG000123456 | 8.75 | 2035-02-28 | 103,2 | -1.0 | USD |
| 2025-09-15 | ZAG000123456 | 8.75 | 2035-02-28 | 103.20 | 10.9 | ZAR |

***Figure 1.1***: Shows subset data example of a poor quality structured data.

### b.) Why this is poor quality (3–4 sentences)

Because dates like "2025/09/31" and "2035-02-30" are nonexistent and date formats differ throughout rows, this data is invalid and inconsistent. Currency is inconsistent (USD vs. ZAR) for the same instrument, and accuracy is questionable (e.g., "Yield = −1.0" for a normal-coupon ZAR bond without context). "N/A" in a pricing field and the absence of percent symbols or defined units compromise completeness. Because the same ISIN is replicated with contradicting values, uniqueness fails, undermining reconciliation and downstream analytics.

### c.) Example of poor-quality unstructured data



| Notes |
|---|
| Bond thingy – SA gov? pays 8.something% till 2035 maybe □□♂ |
| Price ~103 or 130?? not sure lol 😂 check blurry pic |
| Maturity end Feb 35?? or March?? currency USD?? or ZAR?? |

***Figure 1.2***: Shows subset data example of a poor quality unstructured data.

### d.) Why this unstructured data is poor (3–4 sentences)

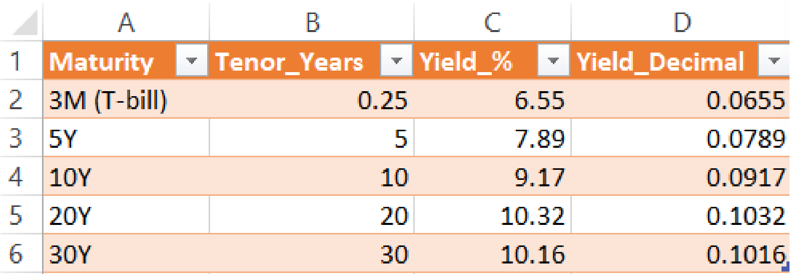
Because it uses imprecise and vague terms like "8.something%" and "~103 or 130??" that don't provide a dependable numerical figure, this unstructured data is poor. The style is erratic, with emojis mixed in with text and ambiguous date references such as "March?? or February 35??." Additionally, it is incomplete because important information like the currency, exact maturity, and ISIN is either absent or unclear. Lastly, it is inappropriate for automated or organized financial analysis due to its conversational, casual character.

# Task 2: Yield Curve Modeling

### a. Country & securities selected

We decided to use South African government bonds as the underlying securities for this investigation. From short-term Treasury bills (as short as six months) to long-term government bonds (with maturities as long as thirty years), South Africa offers a broad variety of fixed-income products with varying maturities.

Observed South African sovereign yields over a range of maturities make up the dataset. Because it offers a comprehensive term structure ranging from short-term instruments (which capture near-term interest rate expectations) to long-term bonds (which reflect inflation and growth expectations over decades), it is suitable for fitting both the Nelson-Siegel model and the Cubic-Spline model.



| | A | B | C | D |
|---|---|---|---|---|
| 1 | Maturity | Tenor_Years | Yield_% | Yield_Decimal |
| 2 | 3M (T-bill) | 0.25 | 6.55 | 0.0655 |
| 3 | 5Y | 5 | 7.89 | 0.0789 |
| 4 | 10Y | 10 | 9.17 | 0.0917 |
| 5 | 20Y | 20 | 10.32 | 0.1032 |
| 6 | 30Y | 30 | 10.16 | 0.1016 |

***Figure 2.1***: Shows South African yield curve data.

### b. Be sure to pick maturities ranging from short-term to long-term (e.g. 6 month maturity to 20 or 30 year maturities).

The following maturities were chosen to depict the entire term structure of interest rates from the South African government securities dataset:

A short-term instrument that reflects the present monetary policy stance and short-term market expectations is the 3-month T-bill.

Five years is a medium-term maturity that reflects market expectations for inflation and economic growth in the upcoming years.

Ten Years: This long-term benchmark, which takes sovereign risk and ongoing inflation forecasts into account, is frequently used for pricing.

A particularly long-term bond that offers information on structural economic conditions is the 20-year bond.

The ultra-long maturity of 30 years captures long-term growth and budgetary sustainability assumptions.

Robust estimate of the Nelson-Siegel and Cubic-Spline models is made possible by this range, which guarantees that the short-term and long-term dynamics of the South African yield curve are captured.

### c. Definition of the Nelson–Siegel Model

One popular parametric model for estimating the term structure of interest rates is the Nelson–Siegel model (Nelson & Siegel, 1987). It allows for an economical interpretation in terms of level, slope, and curvature parameters, is economical, and is adaptable enough to capture various yield curve shapes.

The **forward rate curve** $f(\tau)$, with time to maturity $\tau$, is specified as:

$$f(\tau) = \beta_0 + \beta_1 \frac{1-e^{-\tau/\lambda}}{\tau/\lambda} + \beta_2 (\frac{1-e^{-\frac{\tau}{\lambda}}}{\frac{\tau}{\lambda}} - e^{-\tau/\lambda})$$

where:

- $\beta_0$ represents the **long-term level** of interest rates,
- $\beta_1$ captures the **slope** of the yield curve,
- $\beta_2$ controls the **curvature** (hump/trough),
- $\lambda > 0$ **shape parameter**, determining the exponential decay rate and the maturity at which the curvature peaks

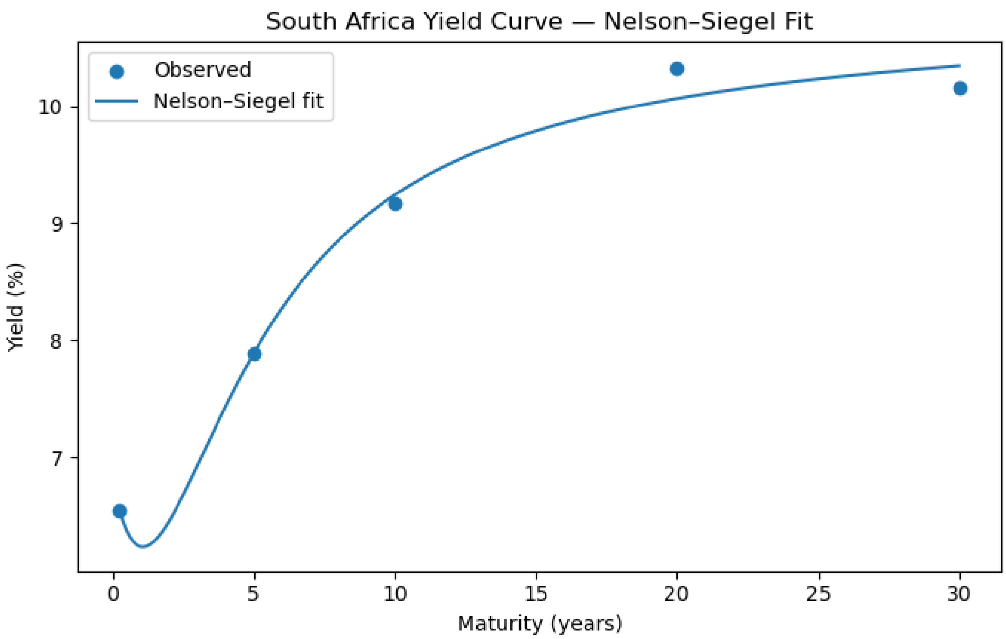
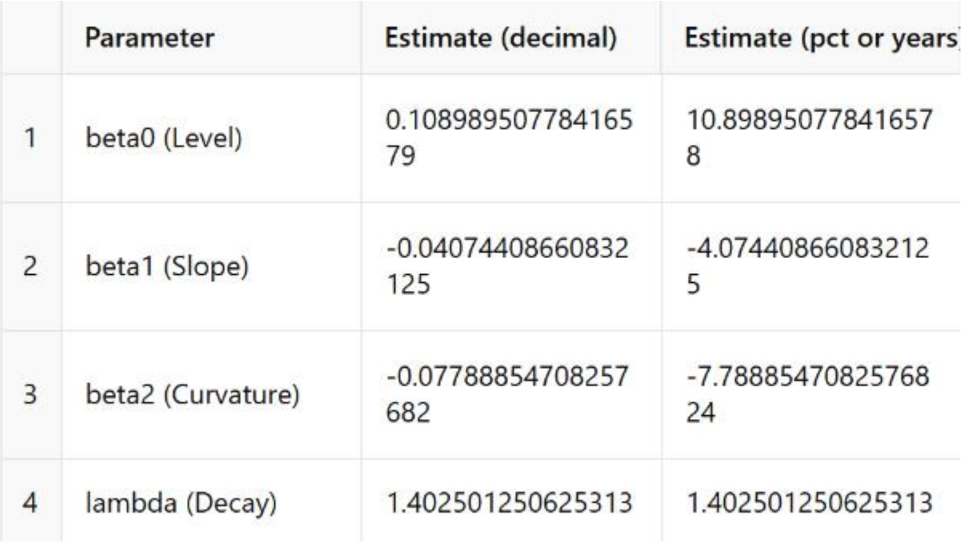


***Figure 2.2*:** Shows South African yield curve Nelson Siegel Fit graph



| | Parameter | Estimate (decimal) | Estimate (pct or years |
|---|---|---|---|
| 1 | beta0 (Level) | 0.10898950778416579 | 10.898950778416578 |
| 2 | beta1 (Slope) | -0.04074408660832125 | -4.074408660832125 |
| 3 | beta2 (Curvature) | -0.07788854708257682 | -7.788854708257682 4 |
| 4 | lambda (Decay) | 1.402501250625313 | 1.402501250625313 |

***Figure 2.2*:** Shows the Nelson Siegel Fitted Model parameters.

### d. Definition of the Cubic-Spline Model

A piecewise cubic polynomial known as a cubic spline enables flexible modeling of smooth, non-linear interactions between a result and a continuous predictor variable. The range of the variable is divided into knot-defined intervals, and a cubic polynomial is used to represent the connection within each interval. To guarantee that the resulting curve is smooth at the knots and continuous, restrictions are applied. Additionally, before the first knot and after the last knot, a restricted cubic spline becomes linear, stabilizing the model in the distribution's tails. (Wu, Gooley, & Gauthier, 2020).

### Equation

The general regression form of the cubic spline is given as:

$$g(y) = \beta_0 + \beta_1 x + \sum_{i=2}^{k-1} \beta_i C_i(x)$$

Where:

- $g(y)$ is a link function (e.g., logit in logistic regression, or log–log survivor function in Cox regression),
- $x$ is the continuous variable,
- $C_i(x)$ are the cubic spline basis functions defined by knots,
- $\beta_0$,$\beta_1$, … … , $\beta_i$ are coefficients estimated from the data,
- $k$ is the number of knots used

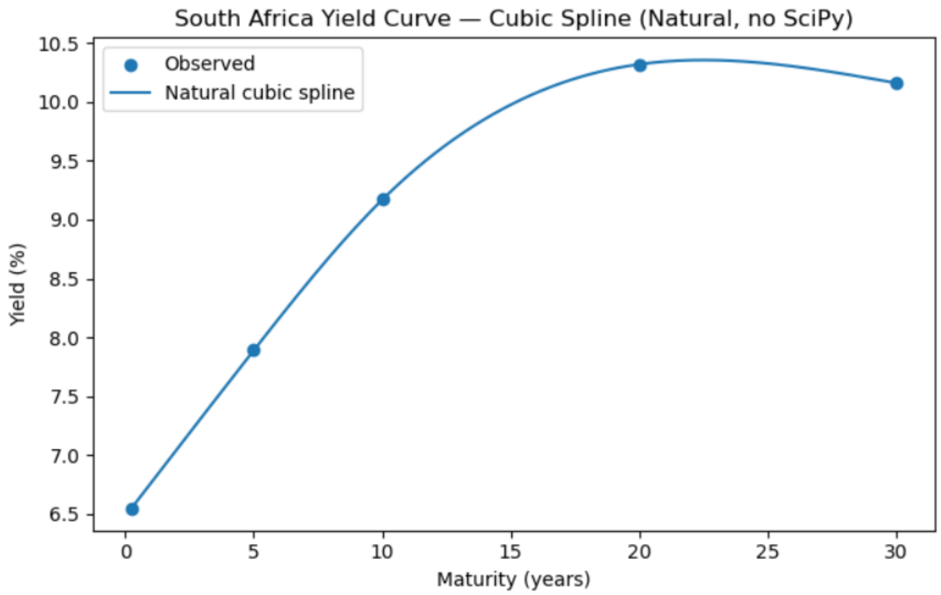


***Figure 2.3***: Shows the Cubic Spline Fitted Model graph.

### e. Compare Nelson–Siegel vs. Cubic-Spline

| **Aspect** | **Nelson–Siegel (NS)** | **Cubic-Spline (Natural, interpolating)** |
|---|---|---|
| In-sample error | **RMSE ≈ 14.3 bp**, **MAE ≈ 10.3 bp** on our 5 tenors | **RMSE = 0 bp** by construction (passes exactly through all points) |
| Smoothness | Smooth, single-hump structure; no wiggles | Smooth between knots; with only 5 knots it's well-behaved |
| Sensitivity to noise | **Low–moderate** (parametric structure dampens idiosyncratic noise) | **High** for an *interpolating* spline (fits noise exactly); a *smoothing* spline (s>0) can mitigate this |
| Stability with sparse points | With 5 points and 4 params, NS is **identifiable but sensitive**, especially λ\lambdaλ & curvature | Determinate, but shape between knots is **entirely driven** by the two neighbors (can be fragile) |
| Extrapolation beyond data | **Disciplined**: converges to β0\beta_0β0 (level) with monotone tail behavior | **Not recommended**: natural spline is defined on the knot interval; extrapolation tends to be unreliable |

In summary, fit

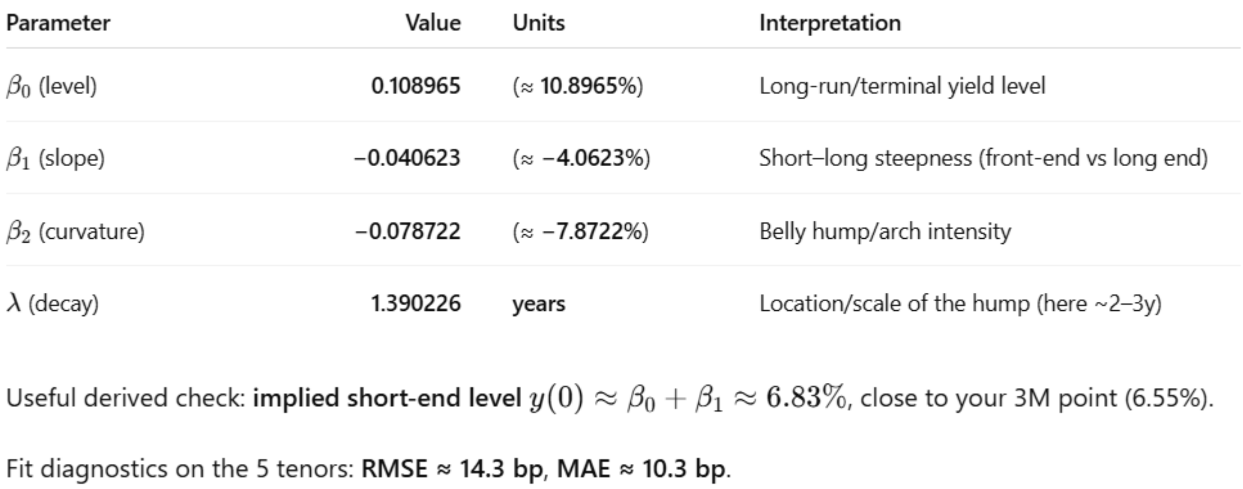
The interpolating spline (zero error) is the best option if you require a precise curve-of-the-day for pricing at the observed maturities.

NS is better if you want a strong, smoothed curve that acts logically off the ends and won't overfit five noisy points.

### f. Parameter levels (explicit estimates)

**Nelson–Siegel (NS)**

**Model:** $y(\tau) = \beta_0 + \beta_1 \frac{1-e^{-\tau/\lambda}}{\tau/\lambda} + \beta_2 (\frac{1-e^{-\frac{\tau}{\lambda}}}{\frac{\tau}{\lambda}} - e^{-\tau/\lambda})$



| Parameter | Value | Units | Interpretation |
|---|---:|---|---|
| $\beta_0$ (level) | **0.108965** | (≈ **10.8965%**) | Long-run/terminal yield level |
| $\beta_1$ (slope) | **−0.040623** | (≈ **−4.0623%**) | Short–long steepness (front-end vs long end) |
| $\beta_2$ (curvature) | **−0.078722** | (≈ **−7.8722%**) | Belly hump/arch intensity |
| $\lambda$ (decay) | **1.390226** | **years** | Location/scale of the hump (here ~2–3y) |

Useful derived check: **implied short-end level** $y(0) \approx \beta_0 + \beta_1 \approx 6.83\%$, close to your 3M point (6.55%).

Fit diagnostics on the 5 tenors: **RMSE ≈ 14.3 bp**, **MAE ≈ 10.3 bp**.

### Cubic Spline (natural, interpolating)

Specification (parameters of the setup):

- **Type:** Natural cubic spline (C² continuous).
- **Knots (fixed) :**{0.25,5,10,20,30} years.
- **Boundary conditions: Natural** → second derivative s″s"s″ at **0.25y** and **30y** set to **0**.
- **Smoothing factor: s=0s = 0s=0** (pure interpolation).

### g. Is Nelson–Siegel "smoothing" unethical?

No, it is not intrinsically unethical to use Nelson-Siegel (NS).

It is a common, model-based method for estimating a continuous yield curve from quotes for discrete bonds. Intent, openness, and the way the outcomes are used and presented determine whether something is morally right or wrong.

Reasons for the universal acceptance of NS smoothing

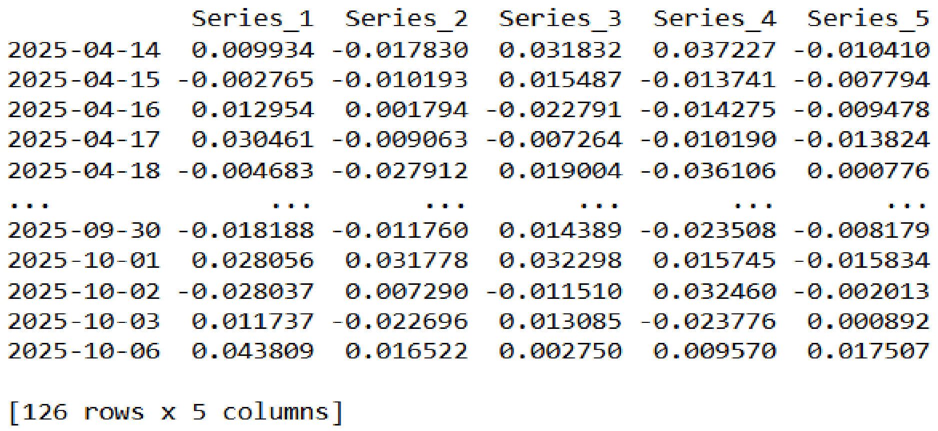
Justifiable goal: In order to price and hedge products between quoted maturities, markets need a continuous curve. A well-known and compact functional form (level–slope–curvature) is offered by NS.

Model, not data manipulation: NS condenses the raw quotes into factors without changing them. The original data is still present and ought to be included in the model fit.

Reproducible and auditable: Anyone can replicate the curve if you provide the data, estimation date, and estimated parameters ($\beta_0$, $\beta_1$, $\beta_2$, $\lambda$) together with fit diagnostics.

# Task 3: Exploiting Correlation

**a. Generate 5 uncorrelated Gaussian random variables**

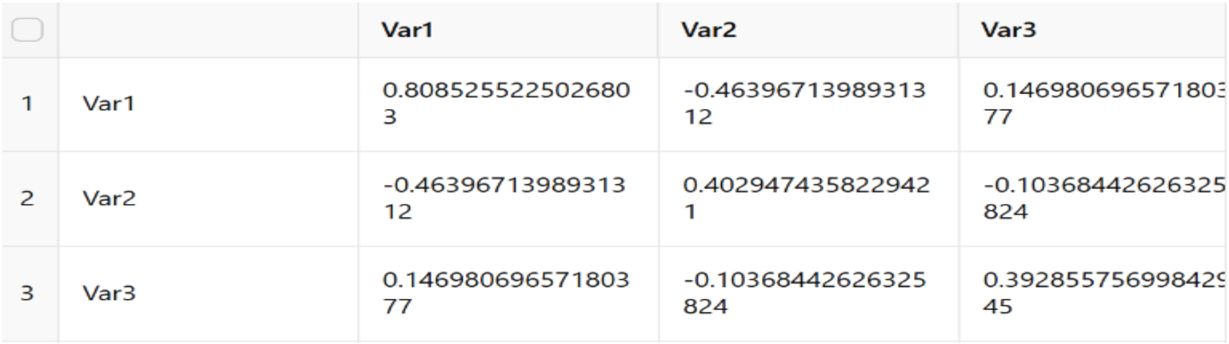


```
            Series_1  Series_2  Series_3  Series_4  Series_5
2025-04-14  0.009934 -0.017830  0.031832  0.037227 -0.010410
2025-04-15 -0.002765 -0.010193  0.015487 -0.013741 -0.007794
2025-04-16  0.012954  0.001794 -0.022791 -0.014275 -0.009478
2025-04-17  0.030461 -0.009063 -0.007264 -0.010190 -0.013824
2025-04-18 -0.004683 -0.027912  0.019004 -0.036106  0.000776
...              ...       ...       ...       ...       ...
2025-09-30 -0.018188 -0.011760  0.014389 -0.023508 -0.008179
2025-10-01  0.028056  0.031778  0.032298  0.015745 -0.015834
2025-10-02 -0.028037  0.007290 -0.011510  0.032460 -0.002013
2025-10-03  0.011737 -0.022696  0.013085 -0.023776  0.000892
2025-10-06  0.043809  0.016522  0.002750  0.009570  0.017507

[126 rows x 5 columns]
```

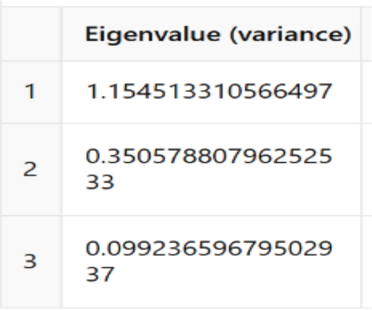
***Figure 3.1*:** Shows Uncorrelated Gaussian Series.

**b. Compute PCA using the covariance matrix**



| □ | | Var1 | Var2 | Var3 |
|---|---|---|---|---|
| 1 | Var1 | 0.8085255225026803 | -0.4639671398931312 | 0.14698069657180377 |
| 2 | Var2 | -0.4639671398931312 | 0.4029474358229421 | -0.10368442626325824 |
| 3 | Var3 | 0.14698069657180377 | -0.10368442626325824 | 0.39285575699842945 |

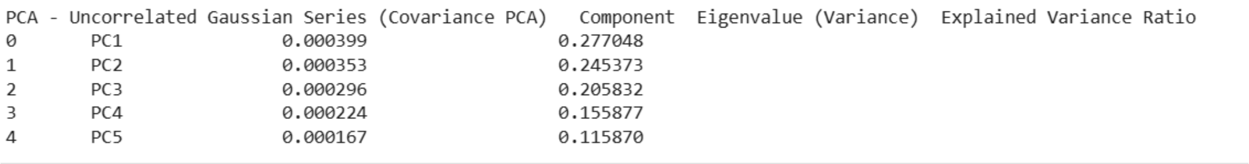
***Figure 3.2*:** Shows the Covariance Matrix from centered data.



| | Eigenvalue (variance) |
|---|---|
| 1 | 1.154513310566497 |
| 2 | 0.35057880796252533 |
| 3 | 0.09923659679502937 |

***Figure 3.3*:** Shows the Eigenvalue Variance

c. **Print explained variances and percentages for components**



```
PCA - Uncorrelated Gaussian Series (Covariance PCA)   Component  Eigenvalue (Variance)  Explained Variance Ratio
0       PC1                  0.000399                    0.277048
1       PC2                  0.000353                    0.245373
2       PC3                  0.000296                    0.205832
3       PC4                  0.000224                    0.155877
4       PC5                  0.000167                    0.115870
```

***Figure 3.4*:** Shows the Expected Variance Ratios.

d. **Produce a screeplot of variance explained by each component**

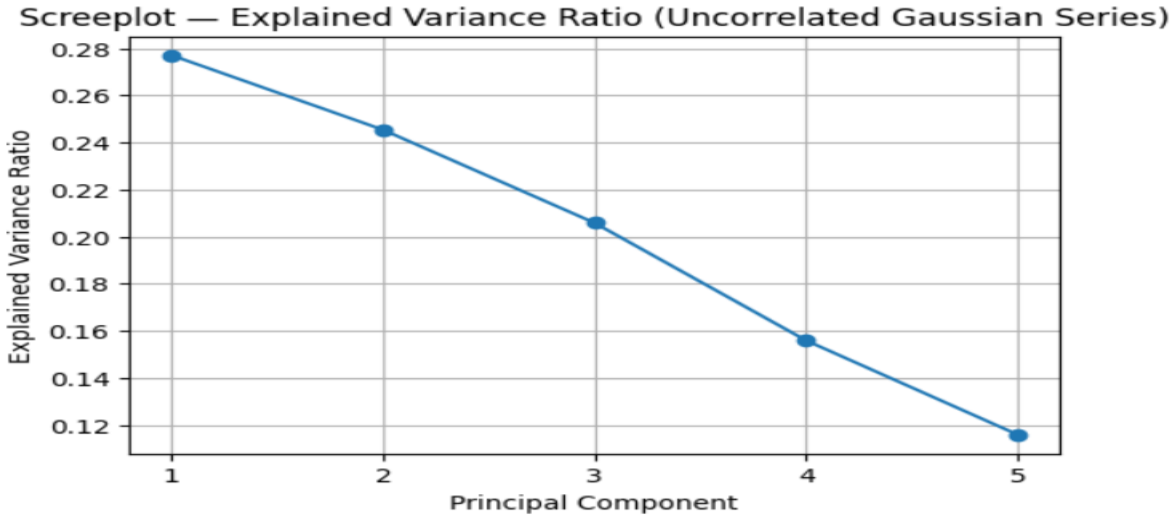


***Figure 3.5*:** Shows the Screeplot of Variance Ratio.

e. **Collect the daily closing yields for 5 government securities**

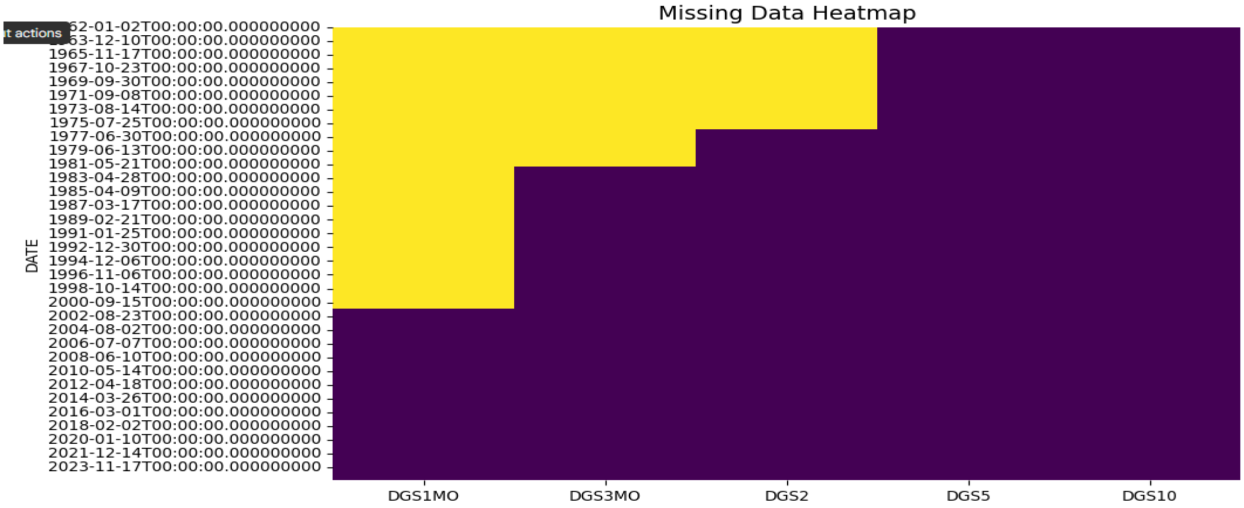


***Figure 3.6*:** Shows the Missing Data Heatmap.

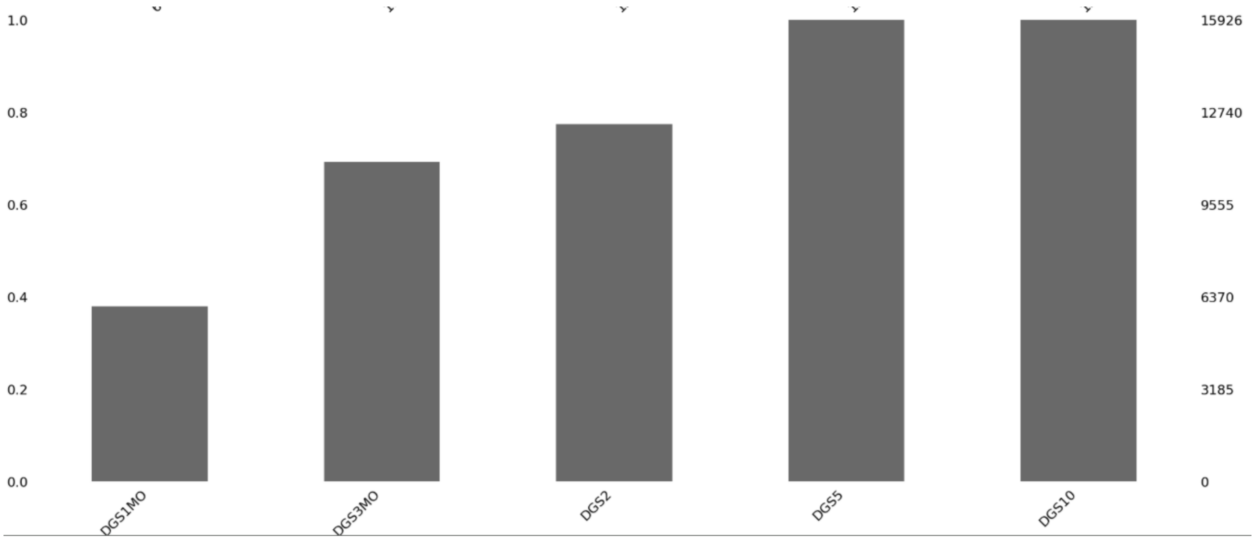


***Figure 3.7*****:** Shows the Bar graph for the Missing Data Heatmap.

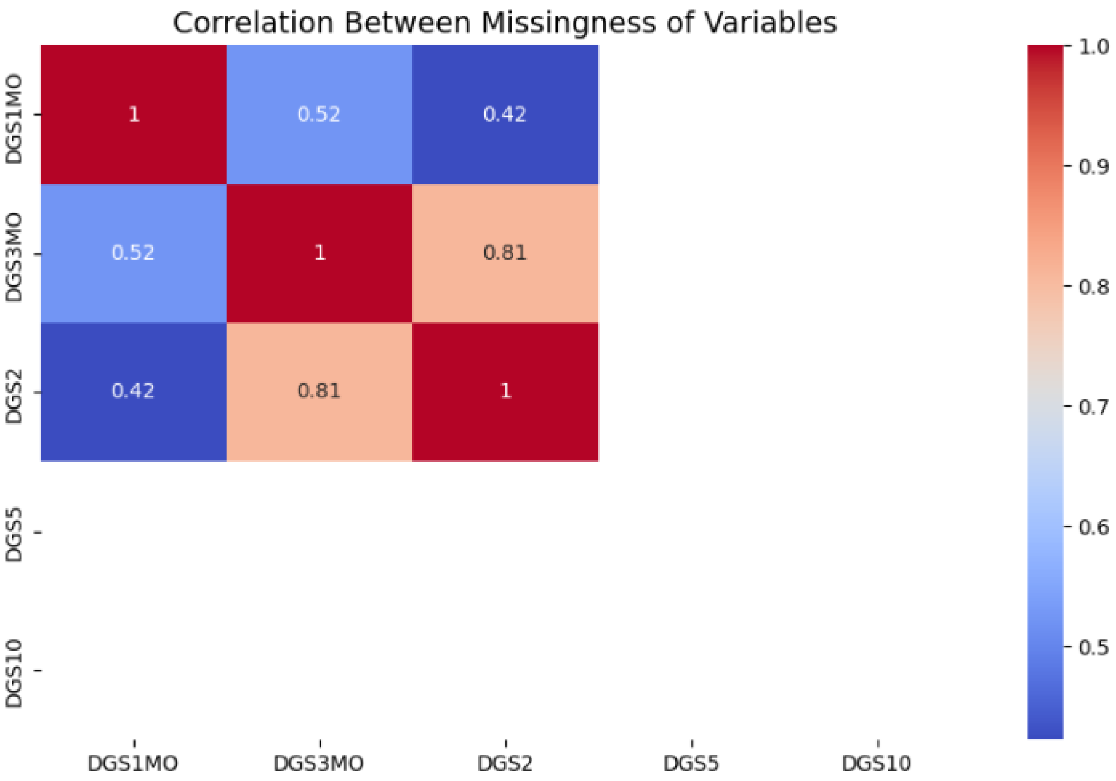
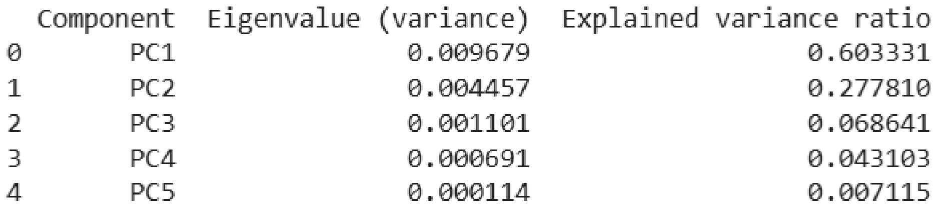


***Figure 3.8*****:** Shows the Correlation Between Missingness of variables.

**h. Print explained variance breakdown**



```
  Component  Eigenvalue (variance)  Explained variance ratio
0       PC1               0.009679                  0.603331
1       PC2               0.004457                  0.277810
2       PC3               0.001101                  0.068641
3       PC4               0.000691                  0.043103
4       PC5               0.000114                  0.007115
```

***Figure 3.9*:** Shows the explainable variance breakdown.

**i. Screeplot for each variance component**

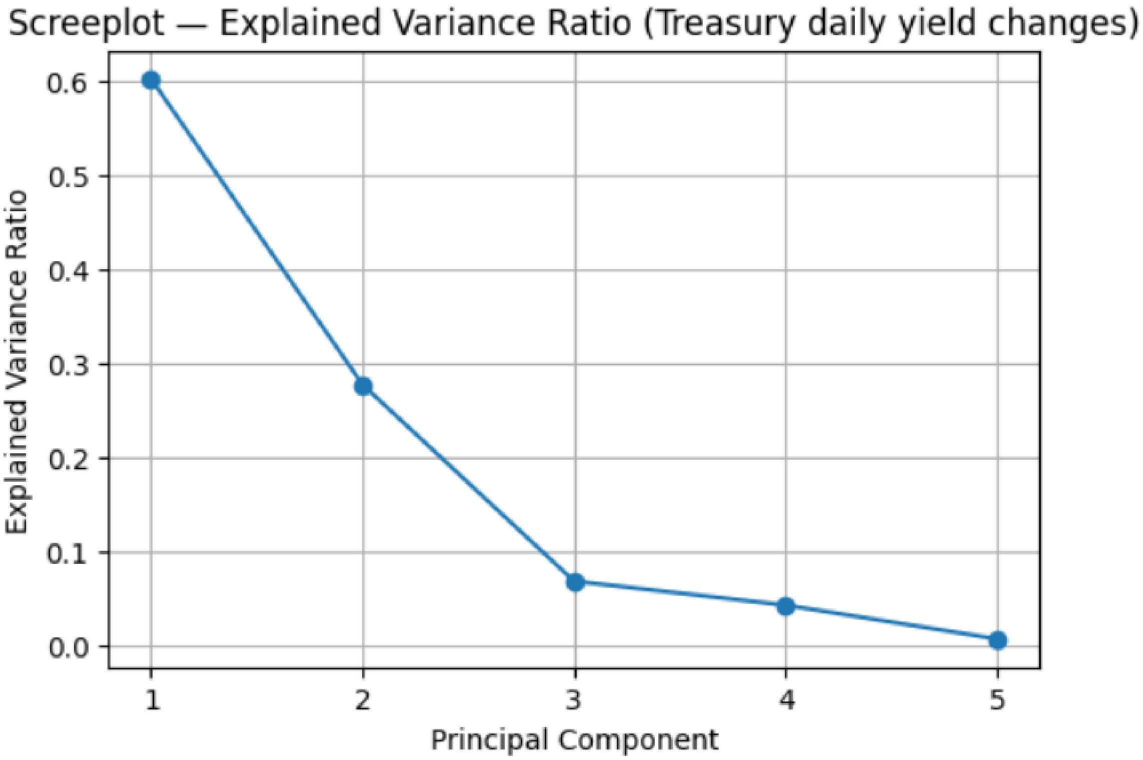
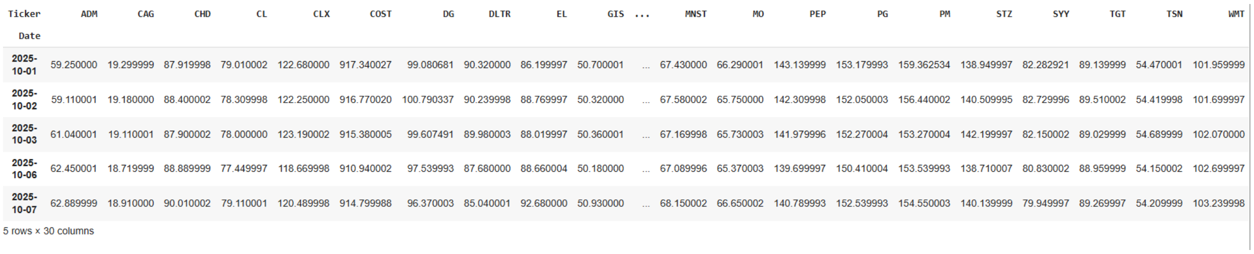


***Figure 3.10*:** Shows the Screeplot explained variance ratio.

# Task 4: Empirical Analysis of ETFs

### b. Get at least 6 months of data (~ 120 data points)

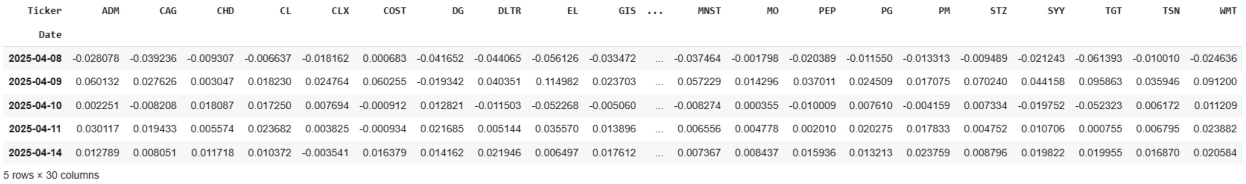


| Ticker | ADM | CAG | CHD | CL | CLX | COST | DG | DLTR | EL | GIS | ... | MNST | MO | PEP | PG | PM | STZ | SYY | TGT | TSN | WMT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Date | | | | | | | | | | | | | | | | | | | | | |
| 2025-10-01 | 59.250000 | 19.299999 | 87.919998 | 79.010002 | 122.680000 | 917.340027 | 99.080681 | 90.320000 | 86.199997 | 50.700001 | ... | 67.430000 | 66.290001 | 143.139999 | 153.179993 | 159.362534 | 138.949997 | 82.282921 | 89.139999 | 54.470001 | 101.959999 |
| 2025-10-02 | 59.110001 | 19.180000 | 88.400002 | 78.309998 | 122.250000 | 916.770020 | 100.790337 | 90.239998 | 88.769997 | 50.320000 | ... | 67.580002 | 65.750000 | 142.309998 | 152.050003 | 156.440002 | 140.509995 | 82.729996 | 89.510002 | 54.419998 | 101.699997 |
| 2025-10-03 | 61.040001 | 19.110001 | 87.900002 | 78.000000 | 123.190002 | 915.380005 | 99.607491 | 89.980003 | 88.019997 | 50.360001 | ... | 67.169998 | 65.730003 | 141.979996 | 152.270004 | 153.270004 | 142.199997 | 82.150002 | 89.029999 | 54.689999 | 102.070000 |
| 2025-10-06 | 62.450001 | 18.719999 | 88.889999 | 77.449997 | 118.669998 | 910.940002 | 97.539993 | 87.680000 | 88.660004 | 50.180000 | ... | 67.089996 | 65.370003 | 139.699997 | 150.410004 | 153.539993 | 138.710007 | 80.830002 | 88.959999 | 54.150002 | 102.699997 |
| 2025-10-07 | 62.889999 | 18.910000 | 90.010002 | 79.110001 | 120.489998 | 914.799988 | 96.370003 | 85.040001 | 92.680000 | 50.930000 | ... | 68.150002 | 66.650002 | 140.789993 | 152.539993 | 154.550003 | 140.139999 | 79.949997 | 89.269997 | 54.209999 | 103.239998 |

5 rows × 30 columns

***Figure 4.1*:** Shows the Tickers data points.

### c. Get at least 6 months of data (~ 120 data points)



| Ticker | ADM | CAG | CHD | CL | CLX | COST | DG | DLTR | EL | GIS | ... | MNST | MO | PEP | PG | PM | STZ | SYY | TGT | TSN | WMT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Date | | | | | | | | | | | | | | | | | | | | | |
| 2025-04-08 | -0.028078 | -0.039236 | -0.009307 | -0.006637 | -0.018162 | 0.000683 | -0.041652 | -0.044065 | -0.056126 | -0.033472 | ... | -0.037464 | -0.001798 | -0.020389 | -0.011550 | -0.013313 | -0.009489 | -0.021243 | -0.061393 | -0.010010 | -0.024636 |
| 2025-04-09 | 0.060132 | 0.027626 | 0.003047 | 0.018230 | 0.024764 | 0.060255 | -0.019342 | 0.040351 | 0.114982 | 0.023703 | ... | 0.057229 | 0.014296 | 0.037011 | 0.024509 | 0.017075 | 0.070240 | 0.044158 | 0.095863 | 0.035946 | 0.091200 |
| 2025-04-10 | 0.002251 | -0.008208 | 0.018087 | 0.017250 | 0.007694 | -0.000912 | 0.012821 | -0.011503 | -0.052268 | -0.005060 | ... | -0.008274 | 0.000355 | -0.010009 | 0.007610 | -0.004159 | 0.007334 | -0.019752 | -0.052323 | 0.006172 | 0.011209 |
| 2025-04-11 | 0.030117 | 0.019433 | 0.005574 | 0.023682 | 0.003825 | -0.000934 | 0.021685 | 0.005144 | 0.035570 | 0.013896 | ... | 0.006556 | 0.004778 | 0.002010 | 0.020275 | 0.017833 | 0.004752 | 0.010706 | 0.000755 | 0.006795 | 0.023882 |
| 2025-04-14 | 0.012789 | 0.008051 | 0.011718 | 0.010372 | -0.003541 | 0.016379 | 0.014162 | 0.021946 | 0.006497 | 0.017612 | ... | 0.007367 | 0.008437 | 0.015936 | 0.013213 | 0.023759 | 0.008796 | 0.019822 | 0.019955 | 0.016870 | 0.020584 |

5 rows × 30 columns

***Figure 4.2*:** Shows the Tickers log daily returns.

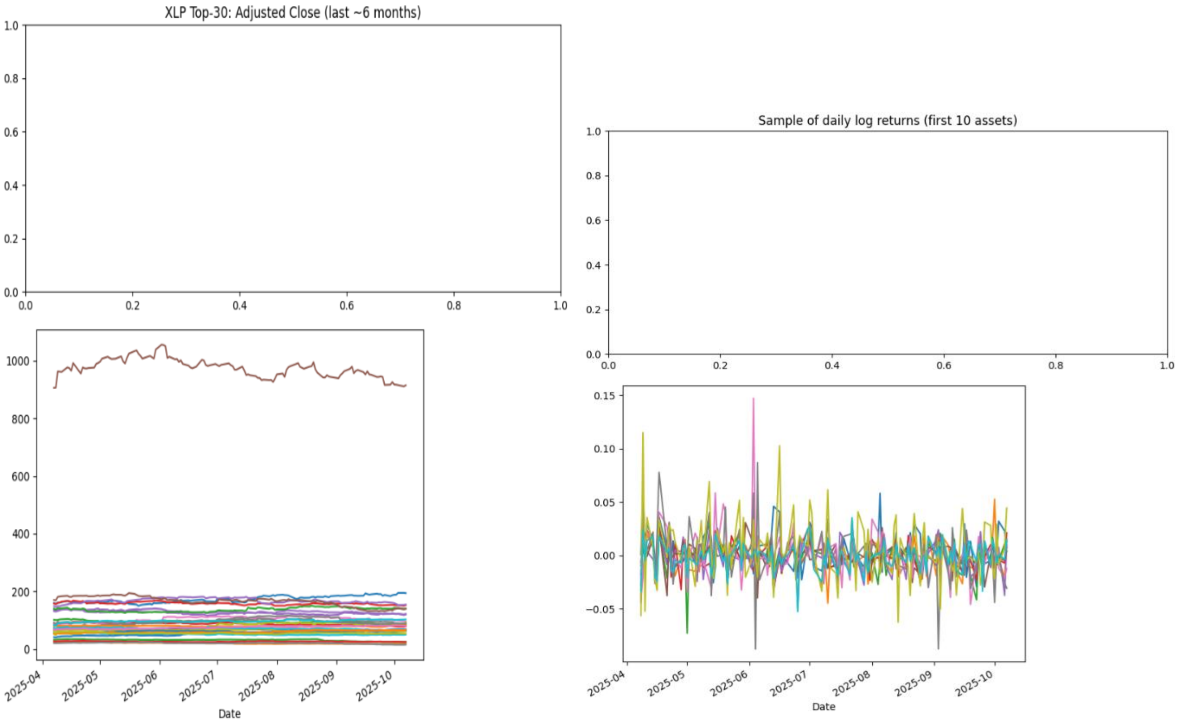
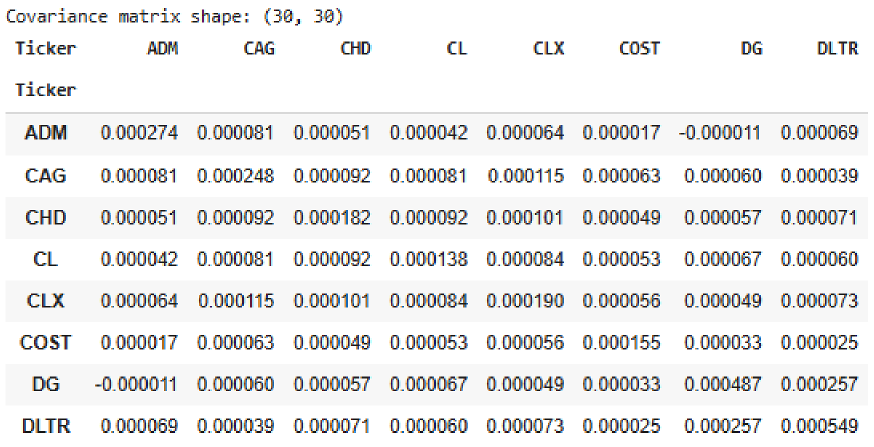


***Figure 4.3*:** Shows the  Plots sample of price series and returns.

### d. Compute the covariance matrix



```
Covariance matrix shape: (30, 30)
```

| Ticker | ADM | CAG | CHD | CL | CLX | COST | DG | DLTR |
|---|---|---|---|---|---|---|---|---|
| **Ticker** | | | | | | | | |
| **ADM** | 0.000274 | 0.000081 | 0.000051 | 0.000042 | 0.000064 | 0.000017 | -0.000011 | 0.000069 |
| **CAG** | 0.000081 | 0.000248 | 0.000092 | 0.000081 | 0.000115 | 0.000063 | 0.000060 | 0.000039 |
| **CHD** | 0.000051 | 0.000092 | 0.000182 | 0.000092 | 0.000101 | 0.000049 | 0.000057 | 0.000071 |
| **CL** | 0.000042 | 0.000081 | 0.000092 | 0.000138 | 0.000084 | 0.000053 | 0.000067 | 0.000060 |
| **CLX** | 0.000064 | 0.000115 | 0.000101 | 0.000084 | 0.000190 | 0.000056 | 0.000049 | 0.000073 |
| **COST** | 0.000017 | 0.000063 | 0.000049 | 0.000053 | 0.000056 | 0.000155 | 0.000033 | 0.000025 |
| **DG** | -0.000011 | 0.000060 | 0.000057 | 0.000067 | 0.000049 | 0.000033 | 0.000487 | 0.000257 |
| **DLTR** | 0.000069 | 0.000039 | 0.000071 | 0.000060 | 0.000073 | 0.000025 | 0.000257 | 0.000549 |

***Figure 4.4***: Shows the covariance matrix shape.

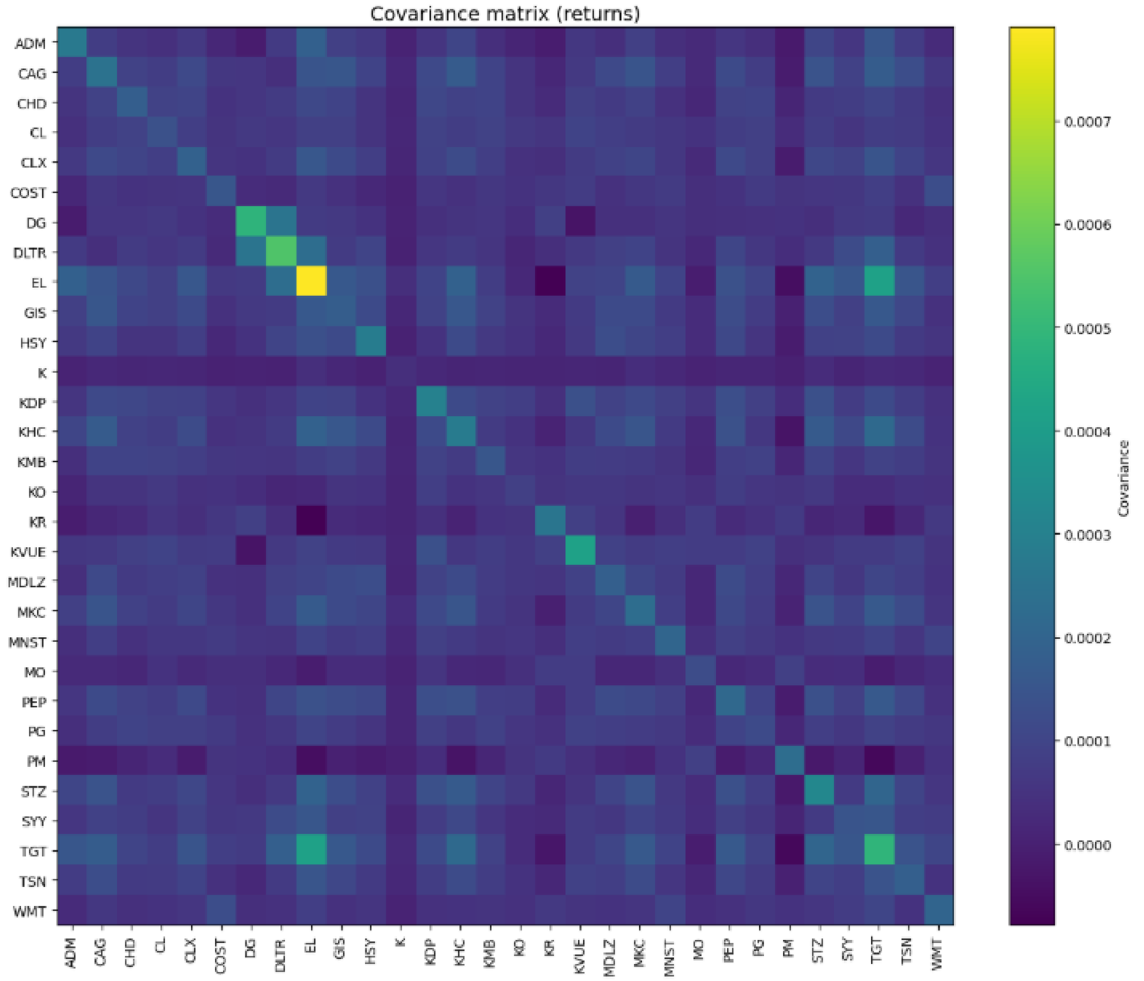


***Figure 4.5***: Shows the covariance matrix returns.

### e. Compute the covariance matrix

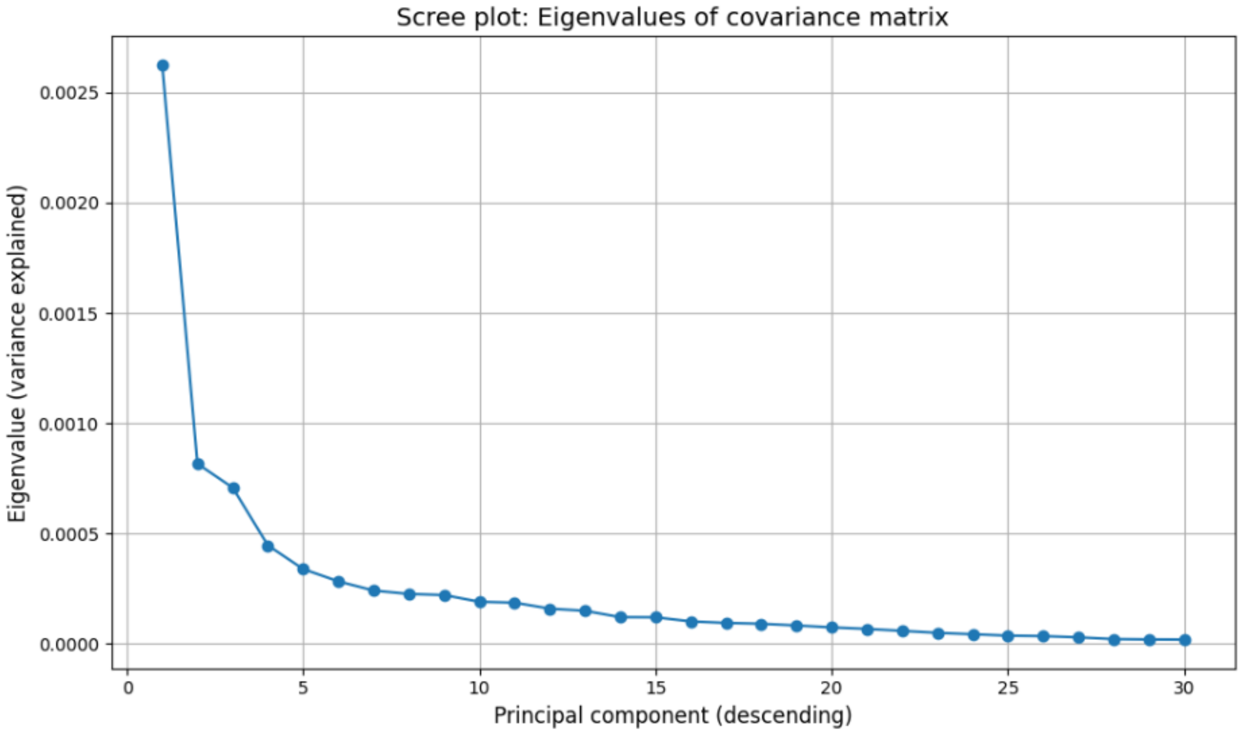


***Figure 4.6*:** Shows the screeplot eigenvalues of covariance matrix.

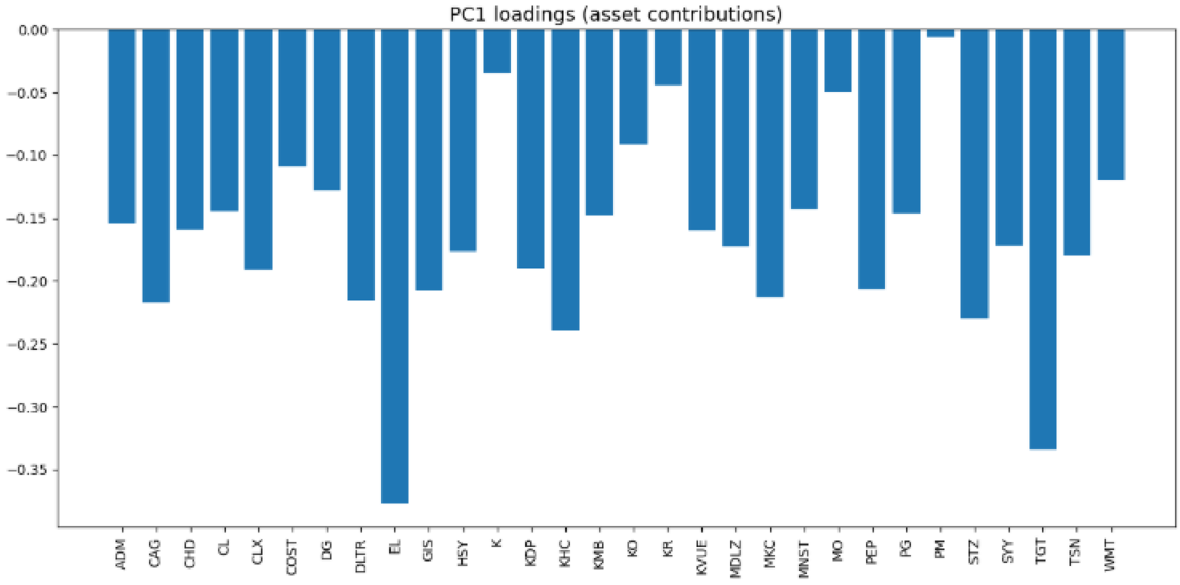


***Figure 4.7*:** Shows the PC1 loadings.

**f. Compute the SVD.**

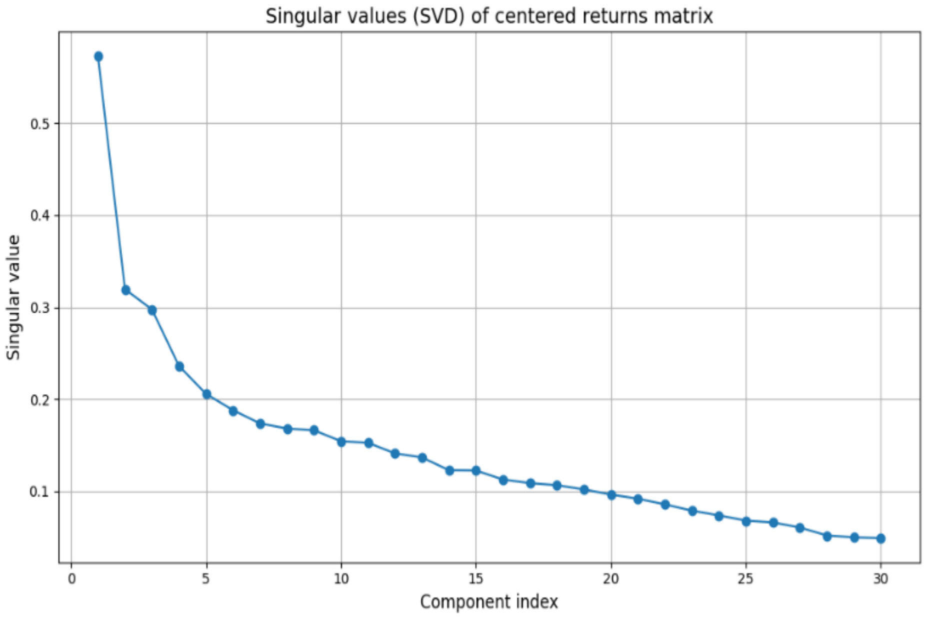
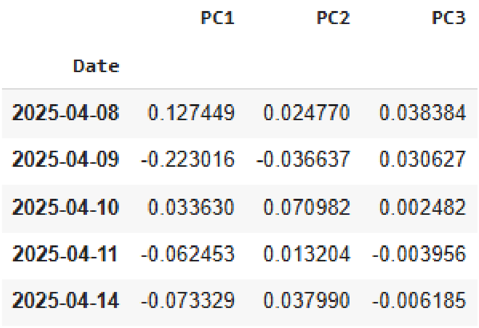


***Figure 4.8*****:** Shows the plot singular values of centered returns matrix.



| | PC1 | PC2 | PC3 |
|---|---|---|---|
| **Date** | | | |
| **2025-04-08** | 0.127449 | 0.024770 | 0.038384 |
| **2025-04-09** | -0.223016 | -0.036637 | 0.030627 |
| **2025-04-10** | 0.033630 | 0.070982 | 0.002482 |
| **2025-04-11** | -0.062453 | 0.013204 | -0.003956 |
| **2025-04-14** | -0.073329 | 0.037990 | -0.006185 |

***Figure 4.9*****:** Shows the PC scores time series.

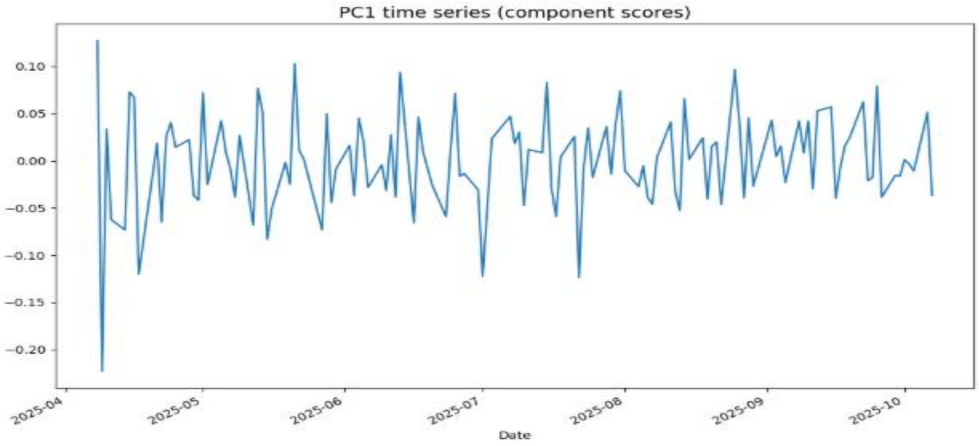


***Figure 4.10*****:** Shows the PC1 time series scores.

Empirical PCA and SVD Analysis of XLP (Consumer Staples)

We chose the Consumer Staples Sector SPDR (XLP) as our sample ETF because its largest constituents are major consumer firms like Walmart (WMT), Costco (COST), Procter & Gamble (PG), Coca-Cola (KO), and Philip Morris (PM) ("XLP Holdings List - Consumer Staples Select Sector SPDR Fund"). We acquired ~6 months of daily prices for XLP's 30 largest constituents and calculated log-daily returns (each day's percentage change). We used returns instead of raw prices because it is scale-invariant (all of an asset's returns are in analogous percentage units and are likely to be stationary, whereas prices have unit roots)("Why Do We Usually Model Returns and Not Prices?", "Why Log Returns"). Practically, using log-returns makes returns accumulated over multiple periods of time (e.g., weeks, quarters), additive and more normally-distributed for risk modeling purposes ("Why Log Returns"). Normalization also gives a meaningful interpretation to the covariance matrix by revealing how each pair of assets' percentage returns co-move.

Having gotten our returns matrix, we then calculated the sample covariance matrix of all 30 assets' returns. This covariance matrix is key to portfolio analysis because each off-diagonal element reveals a pair of stocks' co-movement (positive covariance implies they tend to both rise and fall, negative implies opposite motion). The diagonal elements are each asset's returns variance (each asset's volatility). Therefore, the covariance matrix summarizes the portfolio's risk and correlation structure("Why Do We Usually Model Returns and Not Prices?"). Stocks, for instance, which are in analogous sub-industries, tend to have higher covariation.

Principal Component Analysis (PCA)

We used PCA by eigendecomposing the covariance matrix. PCA finds new orthogonal directions (principal components) which capture maximal variance of the data ("Principal Component

Analysis”, “Principal Component Analysis (PCA): Explained Step-by-Step”). Mathematically, the principal components are the eigenvectors of the covariance matrix. The first eigenvector (principal component) is the direction in 30-dimensional return-space explaining the largest total variance; the second eigenvector is orthogonal to the first and explains the next largest variance, etc (“Principal Component Analysis”, “Principal Component Analysis (PCA): Explained Step-by-Step”). That is, each eigenvector describes a linear combination (factor) of the 30 stocks, and its related eigenvalue describes how much variance that factor explains. Large eigenvalues correspond to strong factors. Intuitively, in financial terms, we find here that first few principal components frequently correspond to broad market/sector factors, and later components correspond to smaller variation sources (idiosyncratic moves). For instance, it's a known empirical fact that first principal component of stock returns tends to have all coefficients of same sign, essentially corresponding to the "market factor"/equity risk premium. Indeed, our calculated first principal component has nearly constant positive loadings on nearly all constituents of XLP, indicating a common market-level effect (e.g. overall consumer spending patterns) (“Why Is the First Principal Component a Proxy for the Market Portfolio, and What Other Proxies Exist?”). Successive components have mixed signs and capture more industry-specific patterns.

The eigenvalues of the covariance matrix actually quantify how many of variance each principal component explains (“Principal Component Analysis (PCA): Explained Step-by-Step”). With our data, largest eigenvalue may explain, say, 30–40% total variance, and successive eigenvalue explains progressively less. This is often illustrated by a scree plot. Intuitively, for portfolio risk, largest eigenvectors (with largest eigenvalues) correspond to dominant risk factors. In other words, a very large first eigenvalue suggests a lot of co-movement being caused by a single common factor. We saw our first 3–4 eigenvalues of our XLP data absorb a lot of variance, indicating a few latent factors are enough to characterize most of the variability of returns. The eigenvectors themselves (the "loadings" or weights for each stock) explain how each stock participates in each factor. For example, if we find the first eigenvector to have about equal positive weights, it means all stocks are moving together by means of that principal direction (“Why Is the First Principal Component a Proxy for the Market Portfolio, and What Other Proxies Exist?”, “Principal Component Analysis (PCA): Explained Step-by-Step”). Low-ranked eigenvectors (small eigenvalues) are related to niche patterns (e.g. one might capture the behavior of a specific stock or sub-sector) and don't contribute to overall variance much (“Principal Component Analysis (PCA): Explained Step-by-Step”).

Singular Value Decomposition (SVD)

We also performed Singular Value Decomposition on the centered returns matrix X (rows=time, columns=assets). SVD decomposes X into $X = U \Sigma V^T$ (Wikipedia). Here U (T×T) holds left singular vectors (an orthonormal basis for time-domain patterns), V (30×30) holds right singular vectors (asset loadings), and Σ is a diagonal matrix of non-negative singular values(Wikipedia,

Wikipedia). Every singular value $\sigma_i$ (Σ entry) is equivalent to the square root of the ith eigenvalue of the covariance matrix (“Task 4 | Principal Component Analysis | Eigenvalues and Eigenvectors”, “Singular Value Decomposition”). Indeed, one discovers that applying PCA to the covariance matrix or SVD to data produces the identical set of orthogonal components for centered data. In SVD, the first right singular vector coincides with the first principal component of PCA, and its singular value $\sigma_1$ is the square root of the first eigenvalue (so $\sigma_1^2 = \lambda_1$, variance explained). The benefit of SVD is directly working on (potentially rectangular) data matrix and being numerically stable ("PCA, Eigen Decomposition and SVD", “Singular Value Decomposition”). Here, in our context, the right singular vectors in V are merely principal component directions across assets (identical to PCA eigenvectors), and left singular vectors in U encode each component's time-series "scores." Diagonal singular values in Σ express how significantly each component impacts data's variance (“Singular Value Decomposition”). Large $\sigma_1$, for example, means first component (market factor) dominates, whereas tiny $\sigma_n$ signifies nth component adds negligibly. We saw, practically, singular values plummeted swiftly, verifying that a handful of components retain majority of structure.

Comparisons between PCA and SVD and Interpretation

Both PCA and SVD aim to reduce dimensionality by finding the most informative directions. In PCA, one explicitly computes the covariance matrix and performs eigen-decomposition. SVD bypasses that by directly decomposing the data matrix. PCA has a clear statistical interpretation (variance explained by each component) whereas SVD is more a general algebraic factorization ("PCA, Eigen Decomposition and SVD", “Singular Value Decomposition”). In practice, when data is mean-centered, the two methods yield equivalent information. Conceptually, PCA is tied to the covariance structure of returns, while SVD is a more general tool that can be applied even when data is not square ("PCA, Eigen Decomposition and SVD", “Singular Value Decomposition”).

For our ETF data, we interpret the results as follows: the eigenvectors (or right-singular vectors) show the principal patterns of co-movement among the 30 stocks. The first eigenvector has all-positive weights, confirming it captures the overall consumer-sector movement (“Why Is the First Principal Component a Proxy for the Market Portfolio, and What Other Proxies Exist?”). Each subsequent eigenvector has weights that highlight contrasts (for example, one eigenvector might load positively on consumer staples versus negatively on tobacco, isolating different industry swings). The eigenvalues (and squared singular values) tell us how much variance each pattern explains (“Principal Component Analysis (PCA): Explained Step-by-Step”, “Singular Value Decomposition”). In our case, the first eigenvalue is largest (strong market factor), the next few are moderately large (sub-sector factors), and the rest are small. The singular values (√eigenvalues) likewise quantify these strengths. Thus, PCA/SVD reveal that the 30-dimensional return data is largely driven by a few latent factors: one broad market factor and a handful of sector-specific factors. This insight is useful for portfolio risk management, as it shows that diversifying across all 30 stocks primarily addresses one or two main sources of risk rather than 30 independent ones.

In summary, computing daily returns and analyzing their covariance via PCA/SVD uncovers the underlying factor structure. Returns are the natural input because they standardize different-priced assets and make variances interpretable (quant.stachex, “Why Log Returns”). PCA identifies uncorrelated principal portfolios (eigenvectors) and quantifies their importance (eigenvalues) (“Principal Component Analysis (PCA): Explained Step-by-Step”, “Singular Value Decomposition”). SVD provides an equivalent decomposition on the raw returns matrix. Together, these techniques help us understand and reduce the dimensionality of the return data, highlighting that most variance comes from a few key factors in the consumer staples sector (“Why Is the First Principal Component a Proxy for the Market Portfolio, and What Other Proxies Exist?”, “Principal Component Analysis (PCA): Explained Step-by-Step”).

# References

Nelson, C. R., & Siegel, A. F. (1987). *Parsimonious modeling of yield curves. Journal of Business*, 60(4), 473–489. https://doi.org/10.1086/296409

Wu, Y., Gooley, T. A., & Gauthier, J. (2020). *Cubic splines for modeling continuous variables in regression analyses. Statistics in Medicine*, 39(30), 4322–4340. https://doi.org/10.1002/sim.8759

PCA, Eigen Decomposition and SVD. Michigan Technological University, https://pages.mtu.edu/~shanem/psy5220/daily/Day04/PCA.html

Principal Component Analysis. Wikipedia: The Free Encyclopedia, Wikimedia Foundation, https://en.wikipedia.org/wiki/Principal_component_analysis

Principal Component Analysis (PCA): Explained Step-by-Step. Built In, https://builtin.com/data-science/step-step-explanation-principal-component-analysis

Singular Value Decomposition. Wikipedia: The Free Encyclopedia, Wikimedia Foundation, https://en.wikipedia.org/wiki/Singular_value_decomposition

Task 4 | Principal Component Analysis | Eigenvalues and Eigenvectors. Scribd, https://www.scribd.com/document/924321982/Task-4

Why Do We Usually Model Returns and Not Prices?" Quantitative Finance Stack Exchange, https://quant.stackexchange.com/questions/16481/why-do-we-usually-model-returns-and-not-prices

Why Is the First Principal Component a Proxy for the Market Portfolio, and What Other Proxies Exist? Quantitative Finance Stack Exchange, https://quant.stackexchange.com/questions/2679/why-is-the-first-principal-component-a-proxy-for-the-market-portfolio-and-what

Why Log Returns. Quantivity, 21 Feb. 2011, https://quantivity.wordpress.com/2011/02/21/why-log-returns/

XLP ETF Sector Allocation. MarketXLS, MarketXLS Limited, https://marketxls.com/etfs/xlp/sectors

XLP Holdings List - Consumer Staples Select Sector SPDR Fund. Stock Analysis, https://stockanalysis.com/etf/xlp/holdings/