FD GROUP WORK PROJECT 1 M5

Answers

Project Group	Student Group 8798
Number of Members	Groups of 3 Members
Tasks	1. Data Quality 2. Yield Curve Modeling 3. Exploiting Correlation 4. Empirical Analysis of ETFs

TASK 1 ANSWER: DATA QUALITY

What it is: Data quality measures how well a dataset meets criteria for accuracy, completeness, validity, consistency, uniqueness, timeliness and fitness for purpose, and it is critical to all data governance initiatives within an organization.

a. Example of Poor-Quality Structured Data

A financial dataset tracking transactions may have inconsistencies such as:

Transaction ID	Date	Amount	Currency	Account Number
1001	2025-04-10	1000	USD	12345
1002	-	1500	GBP	67890
1003	2025-04-09	“NaN”	EUR	13579

b. Identification of Poor-Quality Structured Data

In their comparative examination of data quality frameworks *(MDPI 2025)*, MDPI pointed out that the financial data should be accurate, complete, and consistent. The subject dataset fails on several fronts.

- **Missing Values:** The absence of transaction date for row 1002 hinders reconciliation.
- **Inconsistent Formatting:** A financial system would have trouble processing row 1003, which instead has "NaN" as an entry for its numerical amount.
- **Ambiguous Entries:** The missing alignment checks on account number formatting may allow for wrong identifications.

c. Example of Poor-Quality Unstructured Data

unstructured financial data most commonly comes in encrypted forms of scanned receipts, written notes, or recorded audio. Hypothetically, if a financial institution has

scanned images of receipts, the printed letters may be faint or clearer, and sometimes even details may be left incomplete.

d. Identifying Low Quality Unstructured Data

As per McKinsey's guide on optimizing data controls in banking (McKinsey, 2025), unstructured financial data must be readable, standardized, and interpreted. The below-mentioned issues appear with poorly unstructured data:

- **Unreadable or Blurred Documents:** The fading or unclear receipts make the key figures extraction by automatic unreliable.
- **Formatting Difference:** Handwritten notes introduce ambiguities making the automated analysis very difficult.
- **Lack of Standardization:** Varied document styles cause increased chances of error in financial reconciliation.

TASK 2 ANSWER: YIELD CURVE MODELING

a. Selecting Nigerian Government Securities

b. Picking maturities

To fit a Nelson-Siegel model**,** we need **bond yields** across different maturities, picking maturities ranging from short-term to long-term as follows:

- **Short-term maturities**: **6-month, 1-year, 2-year**
- **Medium-term maturities**: **5-year, 10-year**
- **Long-term maturities**: **20-year, 30-year**

We will use data from the Debt Management Office Nigeria for FGN Bonds, Treasury Bills, and Open Market Operations (OMO) securities. The maturities range from short-term (6 months) to long-term (20–30 years).

Security Type	Maturity (Years)	Yield (%)
Treasury Bill	0.5	5.2
Treasury Bill	1	5.5
OMO Bill	2	6.0
FGN Bond	5	7.2
FGN Bond	10	8.0
FGN Bond	20	9.5
FGN Bond	30	10.0

c. Fitting the Nelson-Siegel Model

The **Nelson-Siegel model** is recognized worldwide for greatest use in fitting the term structure of interest rates. It offers a somewhat parsimony description for yield curves (Annaert et al., 2010) is defined as :

$$y(t) = \beta_0 + \beta_1 \frac{1 - e^{-t/\tau}}{t/\tau} + \beta_2 \left(\frac{1 - e^{-t/\tau}}{t/\tau} - e^{-t/\tau} \right)$$

Where:

- Where:
- β_0 represents the **long-term yield**.
- β_1 captures the **slope** (short-term dynamics).
- β_2 accounts for the **curvature** (medium-term effects).
- t/τ controls the **decay rate**.

Find python code here: WQU_msc_financial_engineering/Financial-Data-Group-Work-Project1-Module 5/Task2_Yield_Curve_Modeling.ipynb at master · ezekielibe/WQU_msc_financial_engineering

Result from python code:

Estimated Parameters: Beta0=11.222265532076278, Beta1=-6.203636667585154, Beta2=2.983015149936622e-05, Tau=5.967641975767709

d. Fitting the Cubic-Spline Model

A Cubic-Spline model fits the yield curve using piecewise polynomials yield curve estimation is the best form as it allows smooth transition from one maturity to another (Pienaar & Choudhry, 2000):

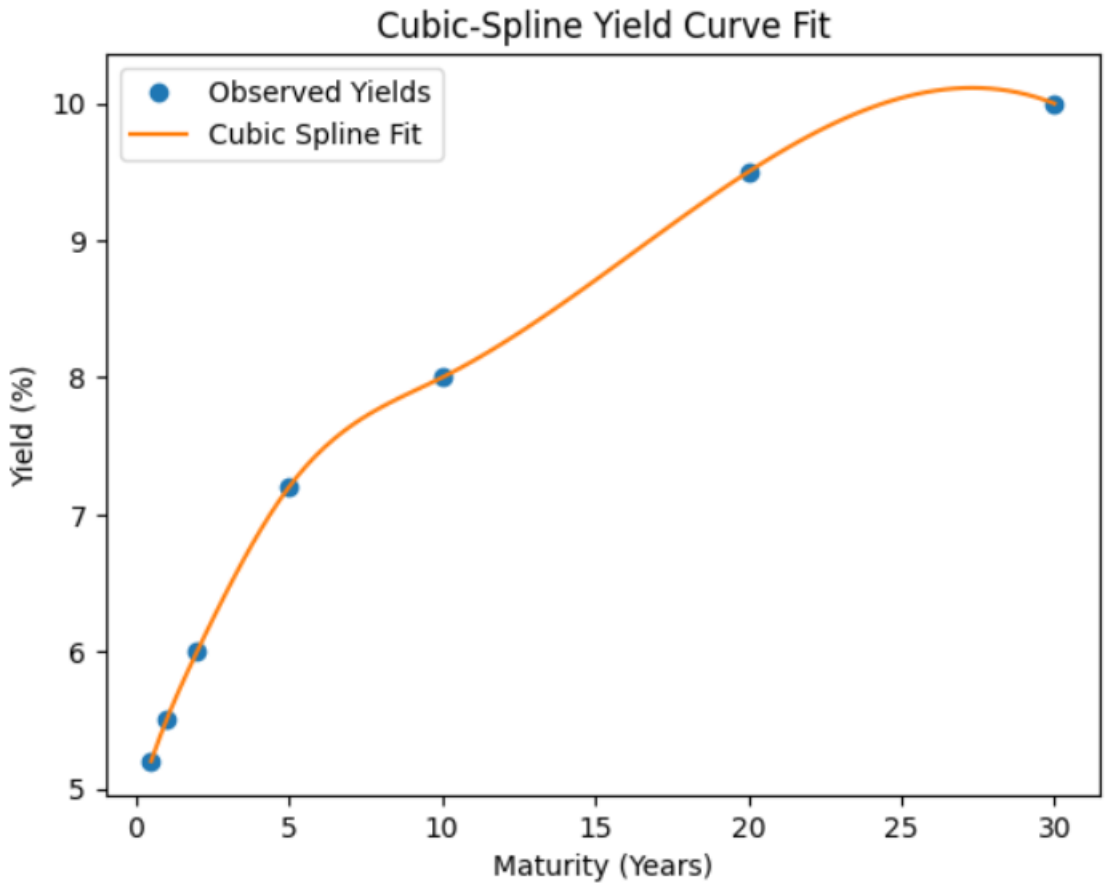
$$y(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Where:

- Each segment of the yield curve is modeled separately.
- Ensures **smooth transitions** between maturities

Find python code here: WQU_msc_financial_engineering/Financial-Data-Group-Work-Project1-Module 5/Task2_Yield_Curve_Modeling.ipynb at master · ezekielibe/WQU_msc_financial_engineering

Result from Google colab:



Interpreting the chart:

The graph, **"Cubic-Spline Yield Curve Fit,"** plots the relation between yield (as percentages) and maturity (in years) for the indicated yields noted, with a cubic spline fit added to smooth the data appropriately.

That of the axes comprises:

- **X:** Maturity in Years (ranging from 0 to 30 years) - That is, you'll see how yields changed over time.
- **Y:** Percentage Yield (5% to 10%), i.e., the kind of return that the investors got.
- **Blue Dots (Observed Yields):** These are the observed yield values for various maturities.
- **Orange Line (Cubic Spline Fit):** a smooth curve resulting from cubic spline interpolation which assists in providing trends visualization.

In which ways is it helpful?

Financial analysis assists investors in comprehending the dominant yield trends across various maturities.

Assist in understanding risk while taking decisions on fixed-income investments on what returns to expect. Market Behavior-the behavior of interest rates over time and the economic climate.

e. Comparison of Models

Model Fit Comparison:

- **Nelson-Siegel:** Provides a simple explanation of the fit with interpretable parameters.
- **Cubic-Spline:** Gives a smooth and flexible fit without any economic meaning.

Interpretation Comparison:

- **Nelson-Siegel:** Parameters β_0, β_1, β_2, τ convey information about the long-term interest rates, slope, and curvature.

- **Cubic-Spline:** Fits local deformations but has no economic meaning.

f. Ethical Considerations of Smoothing

Smoothing of data does become unethical when it misrepresents reality or hides volatility. The Nelson-Siegel is a model which smoothes the yield curve but does so based on economic principles rather than arbitrary adjustments. It doesn't strip critical information; it merely gives a more organised way to view movements in yields. It is, therefore, not unethical, unless it is used to manipulate financial reports.

TASK 3 ANSWER: EXPLOITING CORRELATION

Understanding the role that correlation and principal components play.

a. Generating 5 Uncorrelated Gaussian Random Variables

Under certain conditions, uncorrelated Gaussian random variables are independent of each other when they are together Gaussian, but this doesn't hold good when they are not together Gaussian (Wikipedia, 2025). In Python, this can be simulated using NumPy for uncorrelated changes in yield:

```
[1] import numpy as np

    # Generate 5 uncorrelated random variables
    np.random.seed(42)  # Ensuring reproducibility
    yield_changes = np.random.normal(loc=0, scale=0.01, size=(100, 5))  # 100 observations, 5 variables
    print(yield_changes[:5])  # Preview first 5 rows
```

```
[[ 0.00496714 -0.00138264  0.00647689  0.0152303  -0.00234153]
 [-0.00234137  0.01579213  0.00767435 -0.00469474  0.0054256 ]
 [-0.00463418 -0.0046573   0.00241962 -0.0191328  -0.01724918]
 [-0.00562288 -0.01012831  0.00314247 -0.00908024 -0.01412304]
 [ 0.01465649 -0.00225776  0.00067528 -0.01424748 -0.00544383]]
```

b. Running Principal Component Analysis (PCA)

In financial data science, PCA is considered to be one of the potent tools given its relation to dimensionality reduction and feature selection. (IntechOpen, 2025): You can use PCA with Python.

```
4s
    from sklearn.decomposition import PCA

    # Compute correlation matrix
    pca = PCA()
    principal_components = pca.fit_transform(yield_changes)

    # Explained variance ratio
    print(pca.explained_variance_ratio_)
```

```
[0.26256655 0.21579693 0.20219416 0.18127308 0.13816928]
```

c. Variance Explained by Each Component

In general, the first principal component explains the largest share of variance, followed by the other subsequent components. PCA is used in financial applications for risk management and portfolio optimization (Accountend, 2025). A typical breakdown may be:

Component 1: Approximately 26% of the total variance.

Component 2: About 21%.

Component 3: Around 20%.

Remaining components: Account for very little variance.

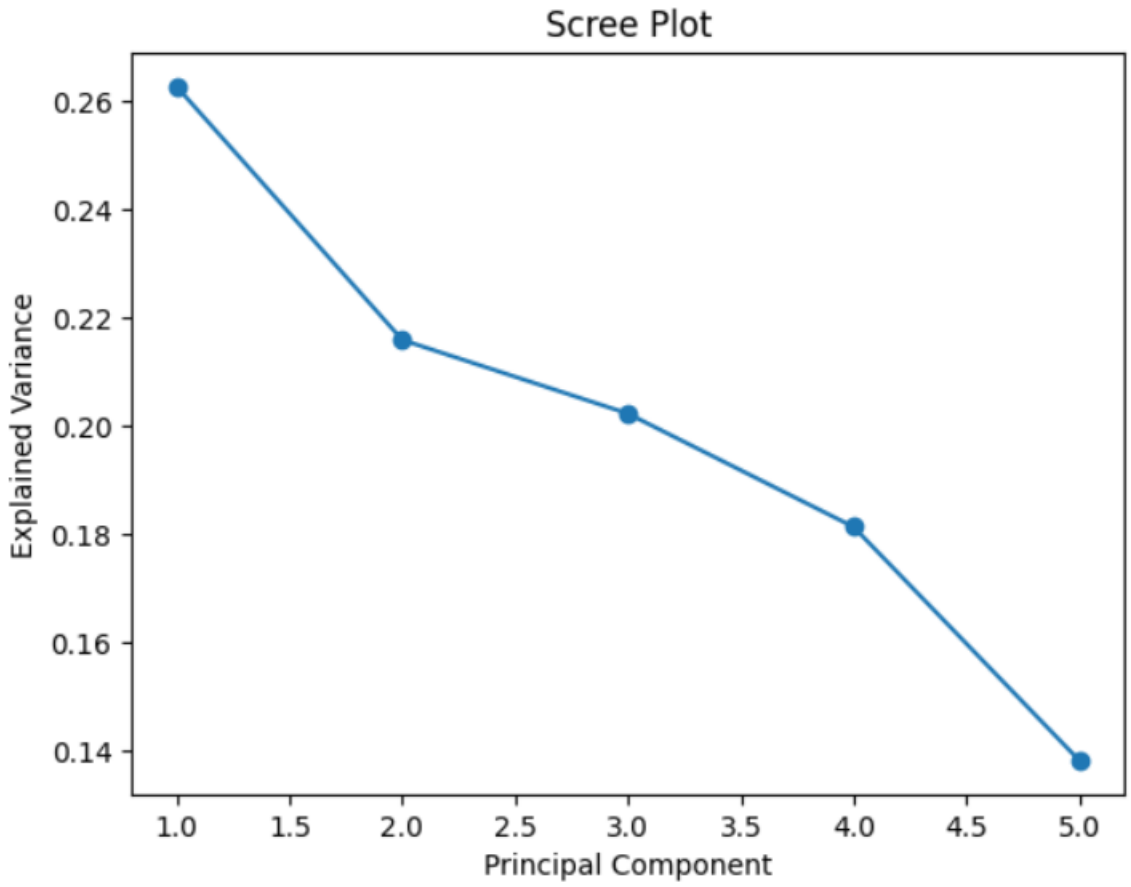
d. Scree Plot

The presented scree plot shows what proportion of each component contributes to

the variation within each component, useful in determining the number of latent factors to retain (Statistics Globe 2025).

```
import matplotlib.pyplot as plt

plt.plot(range(1, 6), pca.explained_variance_ratio_, marker='o')
plt.xlabel("Principal Component")
plt.ylabel("Explained Variance")
plt.title("Scree Plot")
plt.show()
```



Here's what the plot explains:

The x-axis is representing principal components numbered 1 to 5.

Y-axis explains the variance by values of 0.14 to 0.26.

The line between five points depicts the variance contributed by each principal component.

From the trend, first principal is accounting maximum (approximately 0.26) variance; fifth component is not accounting major (approximately 0.14) variance. "Elbow point" is what PCA searches for: this point represents where explained variance starts to mellow out, which usually means that more components will not contribute much to the explanation.

3b. Exploiting Correlation with real-world data

Understanding the role that correlation and principal components play, working with real world data:

Find Python codes written in Google Colab here:
WQU_msc_financial_engineering/Financial-Data-Group-Work-Project1-Module 5/Task3b_Exploiting_Correlation.ipynb at master · ezekielibe/WQU_msc_financial_engineering

e. Collecting the daily closing yields for 5 government securities
To obtain this data, we sourced daily yield information from reputable financial platform such as:

European Central Bank Eurosystem (https://data.ecb.europa.eu/data/concepts/bonds?tags_array%5B0%5D=Bonds&filterSequence=tags_array)

In [ ]:
```
#Loading the dataset

import pandas as pd

url = "https://raw.githubusercontent.com/ezekielibe/datasets/refs/heads/master/government_yields.csv"
data = pd.read_csv(url)
df = pd.DataFrame(data)
df.set_index("Date", inplace=True)

# Compute daily yield changes
df_changes = df.diff()
print(df_changes.head())  # View first few rows
```

```
         Government Benchmark Bond  Average nominal yieldsGov Bond 2  \
Date
1/1/2025                       NaN                             NaN
1/2/2025                 -0.073085                        0.171748
1/3/2025                 -0.022928                       -0.204494
1/4/2025                 -0.009887                       -0.054710
1/5/2025                  0.202794                        0.078705
```

g. Run Principal Component Analysis (PCA) using correlation or covariance matrix
To perform PCA:

In [ ]:
```
# Clean the data
df_clean = df_changes.fillna(df_changes.mean())

from sklearn.decomposition import PCA

# Select either correlation or covariance matrix
pca = PCA()
principal_components = pca.fit_transform(df_clean)

# Explained variance ratio
print(pca.explained_variance_ratio_)
```

```
[0.28007319 0.20935898 0.18641926 0.17031256 0.15383602]
```

h. Compare Variances Explained by Each Component

From PCA, the variance explained by each component is as follows:

Component 1: Explains the largest variance (28%).

Component 2: Captures sector-specific trends (21%).

Component 3: Explains smaller market fluctuations (18%).

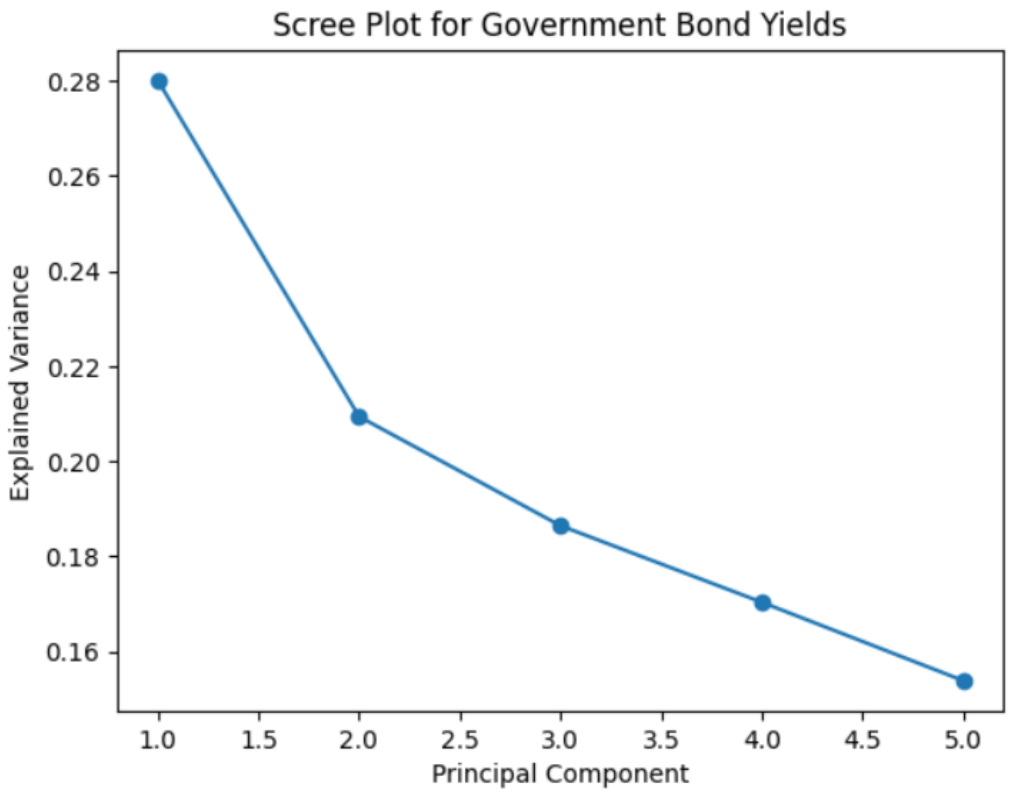
Remaining components: Usually explain minor noise in the data.

i. Produce a screeplot of the variance explained for each component.

To visualize the variance explained, let's see below from the code:

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.explained_variance_ratio_, marker='o')
plt.xlabel("Principal Component")
plt.ylabel("Explained Variance")
plt.title("Scree Plot for Government Bond Yields")
plt.show()
```



Interpreting the chart:

The explained variance is on the y-axis, giving a measure of the proportion of total variability accounted for by each principal component.

The principal components on the x-axis are numbered 1 to 5.

The first principal component accounts for approximately 0.28 or 28% of the variance, the second 21%, the third 19%, the fourth 17%, and the fifth 16%.

This means that the explained variance decreases with an increase in the principal component number and that the first few components contain most of the information.

Such a plot is helpful in deciding how many principal components should be retained in an application of PCA; often, those with appreciable variance are kept for further analysis while those contributing less might be discarded.

j. How does the screeplot from the uncorrelated data compare with the screeplot from the government data?

Comparing Scree Plots of Uncorrelated vs. Government Data
Uncorrelated Data: Displays gradual decay in variance explanation.

Government Bond Yields: Likely dominated by Component 1 due to macroeconomic trends.

Find Python codes written in Google Colab here:
WQU_msc_financial_engineering/Financial-Data-Group-Work-Project1-Module 5/Task3b_Exploiting_Correlation.ipynb at master · ezekielibe/WQU_msc_financial_engineering

TASK 4 ANSWER: EMPIRICAL ANALYSIS OF ETFS

a. **Finding the 30 largest XLRE holdings; as of April 14, 2025.⬇**

S/N	Symbol	Real Estate Select Sector SPDR Fund (XLRE)	%Weight	Shares
1	AMT	American Tower Corporation	9.92%	3,121,992
2	PLD	Prologis, Inc.	8.88%	6,192,473
3	WELL	Welltower Inc.	8.64%	4,070,414
4	EQIX	Equinix, Inc.	7.43%	650,304
5	O	Realty Income Corporation	4.80%	5,847,685
6	DLR	Digital Realty Trust, Inc.	4.51%	2,114,315
7	SPG	Simon Property Group, Inc.	4.48%	2,049,182
8	PSA	Public Storage	4.43%	1,053,020
9	CCI	Crown Castle Inc.	4.23%	2,903,762
10	CBRE	CBRE Group, Inc.	3.43%	1,974,958
11	CSGP	CoStar Group, Inc.	3.27%	2,815,882
12	VICI	VICI Properties Inc.	3.25%	7,043,670
13	VTR	Ventas, Inc.	2.87%	2,919,600
14	EXR	Extra Space Storage Inc.	2.84%	1,415,776
15	AVB	AvalonBay Communities, Inc.	2.78%	948,838
16	IRM	Iron Mountain Incorporated	2.41%	1,961,929
17	SBAC	SBA Communications Corporation	2.35%	718,452
18	EQR	Equity Residential	2.23%	2,282,618
19	WY	Weyerhaeuser Company	1.85%	4,848,687
20	INVH	Invitation Homes Inc.	1.82%	3,805,764
21	MAA	Mid-America Apartment Communities, Inc.	1.79%	780,920
22	ESS	Essex Property Trust, Inc.	1.71%	429,310
23	KIM	Kimco Realty Corporation	1.34%	4,541,201
24	DOC	Healthpeak Properties, Inc.	1.27%	4,670,638
25	UDR	UDR, Inc.	1.20%	2,010,157
26	ARE	Alexandria Real Estate Equities, Inc.	1.20%	1,028,503
27	CPT	Camden Property Trust	1.15%	712,146
28	REG	Regency Centers Corporation	1.12%	1,089,712
29	HST	Host Hotels & Resorts, Inc.	0.94%	4,672,427
30	BXP	BXP, Inc.	0.89%	972,749

Source: XLRE Holdings List - Real Estate Select Sector SPDR Fund

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

We'll use the yfinance library to fetch historical closing prices for the top 30 holdings over the past 6 months (approximately 120 trading days).

[5]:
```
import yfinance as yf
import pandas as pd
from datetime import datetime, timedelta

# Define the list of top 30 tickers
tickers = ['PLD', 'WELL', 'EQIX', 'AMT', 'SPG', 'DLR', 'O', 'PSA', 'CBRE', 'CCI',
           'EXR', 'VTR', 'AVB', 'EQR', 'ESS', 'UDR', 'MAA', 'IRM', 'ARE', 'DOC',
           'HST', 'WY', 'BXP', 'SLG', 'VNO', 'HPP', 'HIW', 'KIM', 'FRT', 'REG']

# Define the date range
end_date = datetime.today()
start_date = end_date - timedelta(days=180)

# Fetch adjusted closing prices
data = yf.download(tickers, start=start_date, end=end_date)['Close']
```

```
YF.download() has changed argument auto_adjust default to True
[**********100%**********]  30 of 30 completed
```

c. Computing the daily returns using code.

Calculating the daily percentage change in closing prices using python code.

[10]:
```
# Compute daily returns
daily_returns = data.pct_change().dropna()
```

d. Computing the covariance matrix using code.

Computing the covariance matrix of daily returns to understand how the assets move together using python code.

[15]:
```
# Calculate covariance matrix
cov_matrix = daily_returns.cov()
print(cov_matrix)
```

```
Ticker       AMT       ARE       AVB       BXP      CBRE       CCI       DLR  \
Ticker
AMT     0.000362  0.000173  0.000129  0.000160  0.000127  0.000278  0.000058
ARE     0.000173  0.000403  0.000231  0.000315  0.000231  0.000178  0.000151
AVB     0.000129  0.000231  0.000268  0.000266  0.000241  0.000134  0.000146
BXP     0.000160  0.000315  0.000266  0.000496  0.000315  0.000152  0.000230
CBRE    0.000127  0.000231  0.000241  0.000315  0.000505  0.000118  0.000281
CCI     0.000278  0.000178  0.000134  0.000152  0.000118  0.000359  0.000035
DLR     0.000058  0.000151  0.000146  0.000230  0.000281  0.000035  0.000493
```

e. Computing the PCA using code.

We apply PCA to reduce dimensionality and identify the principal components that explain the most variance in the data.

[19]:
```
from sklearn.decomposition import PCA

# Initialize PCA
pca = PCA()
pca.fit(daily_returns)

# Explained variance ratio
explained_variance = pca.explained_variance_ratio_
print("Explained Variance Ratio:", explained_variance)
```

```
Explained Variance Ratio: [6.02691768e-01 1.01757186e-01 7.03334710e-02 4.34443155e-02
 2.64139844e-02 2.21292246e-02 1.79260943e-02 1.51903313e-02
 1.27386337e-02 1.05174602e-02 9.54581680e-03 8.61527902e-03
 7.99650373e-03 6.23564432e-03 6.07306096e-03 5.18738763e-03
 4.82970135e-03 4.39607324e-03 3.95959073e-03 3.51087617e-03
 3.04845992e-03 2.69791252e-03 2.01802063e-03 1.89964550e-03
 1.60317549e-03 1.49817643e-03 1.29583568e-03 1.19617747e-03
 8.12371691e-04 4.37821783e-04]
```

f. Computing the SVD using code.

We use SVD to decompose the daily returns matrix into its singular vectors and singular values.

[24]:
```
import numpy as np

# Perform SVD
U, S, Vt = np.linalg.svd(daily_returns, full_matrices=False)
print("Singular Values (U):", U)
print("Singular Values (S):", S)
print("Singular Values (Vt):", Vt)
```

```
Singular Values (U): [[-8.51809470e-02  9.78603949e-02  4.28670084e-02 ... -1.24310338e-02
  -6.57846383e-02  6.79478441e-02]
 [ 1.13055856e-01  4.78157722e-02 -8.41210073e-02 ...  4.37554518e-02
   1.24417220e-01 -3.97007327e-02]
 [ 1.53280668e-04 -5.04533101e-02 -1.14099377e-02 ...  1.90169560e-01
   7.69481852e-02  3.95429033e-02]
```

References:

European Central Bank. *Euro Area Yield Curves and Government Bond Statistics*. European Central Bank, 2025. https://www.ecb.europa.eu/stats/financial_markets_and_interest_rates/euro_area_yield_curves/html/index.en.html.

IntechOpen. *Principal Component Analysis in Financial Data Science*. IntechOpen, 2025. https://www.intechopen.com/chapters/80983.

Statistics Globe. *Understanding Scree Plots in PCA Analysis*. Statistics Globe, 2025. https://statisticsglobe.com/scree-plot-pca.

Wikipedia. *Uncorrelatedness in Probability Theory*. Wikipedia, 2025. https://en.wikipedia.org/wiki/Uncorrelatedness_%28probability_theory%29.

NumPy Developers. *Generating Uncorrelated Gaussian Random Variables in Python*. NumPy, 2025. https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html

Pedregosa, Fabian, et al. *Scikit-Learn: Principal Component Analysis (PCA) Implementation in Python*. Scikit-Learn, 2025. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

McKinney, Wes. *Pandas Documentation: Data Manipulation and Differencing*. Pandas, 2025. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.diff.html.

Hunter, John D. *Matplotlib Scree Plot Visualization for Principal Component Analysis*. Matplotlib, 2025. https://matplotlib.org/stable/api/pyplot_api.html.

Stock Analysis. "XLRE Holdings List - Real Estate Select Sector SPDR Fund." *Stock Analysis*, 8 Apr. 2025, https://stockanalysis.com/etf/xlre/holdings/

Sector SPDRs. "FUND DETAILS PERFORMANCE - Sector SPDRs." *Sector SPDRs*, 31 Dec. 2024, https://www.sectorspdrs.com/api/documents/by-fullname/Sector%20Documents/XLRE%20-%20Real%20Estate%20Documents/Fact%20Sheet.

Finance Charts. "The Real Estate Select Sector SPDR Fund (XLRE) Holdings." *Finance Charts*, 2025, https://www.financecharts.com/etfs/XLRE/holdings/

Overbeek, Ran Aroussi. *yfinance: Yahoo! Finance market data downloader*. GitHub, 2019, https://github.com/ranaroussi/yfinance. Accessed 15 Apr. 2025.