

FD GROUP WORK PROJECT 1 M5

Answers

Project Group	Student Group 8798
Number of Members	Groups of 3 Members
Tasks	<ol style="list-style-type: none">1. Data Quality2. Yield Curve Modeling3. Exploiting Correlation4. Empirical Analysis of ETFs

TASK 1: 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: 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 · ezekieli/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):

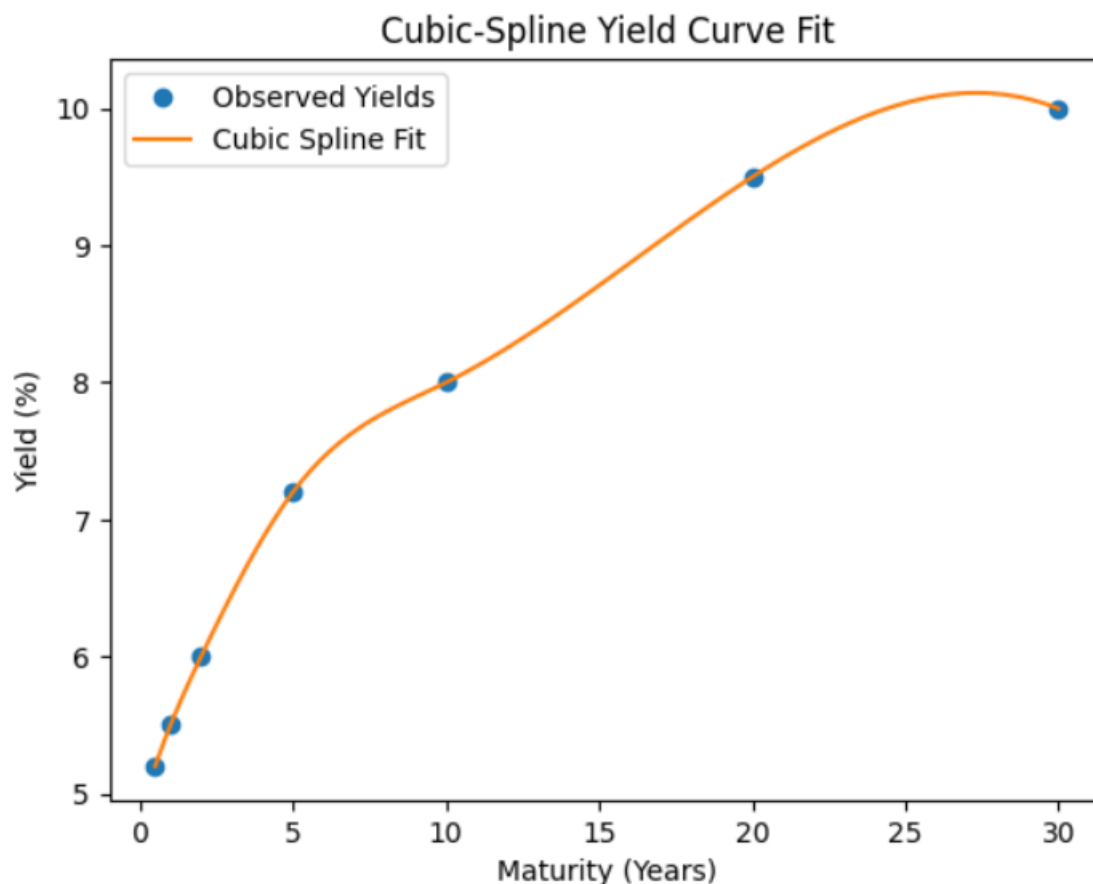
$$y(t) = a_0 + a_1t + a_2t^2 + a_3t^3$$

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 · ezekieli/WQU msc financial engineering](#)

Result from Google colab:



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 smooths 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: EXPLOITING CORRELATION

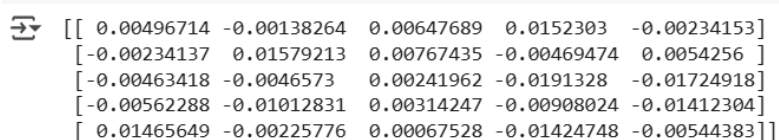
Understanding the role that correlation and principal components play.

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
```

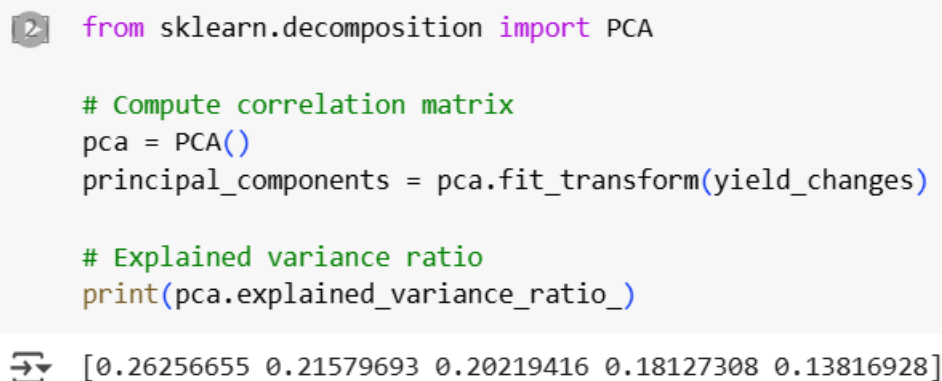


```
[[ 0.00496714 -0.00138264  0.00647689  0.0152303 -0.00234153]
 [-0.00234137  0.01579213  0.00767435 -0.00469474  0.0054256 ]
 [-0.00463418 -0.0046573  0.00241962 -0.0191328 -0.01724918]
 [-0.00562288 -0.01012831  0.00314247 -0.00908024 -0.01412304]
 [ 0.01465649 -0.00225776  0.00067528 -0.01424748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

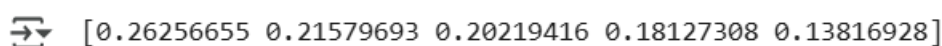
✓
4s



```
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%.

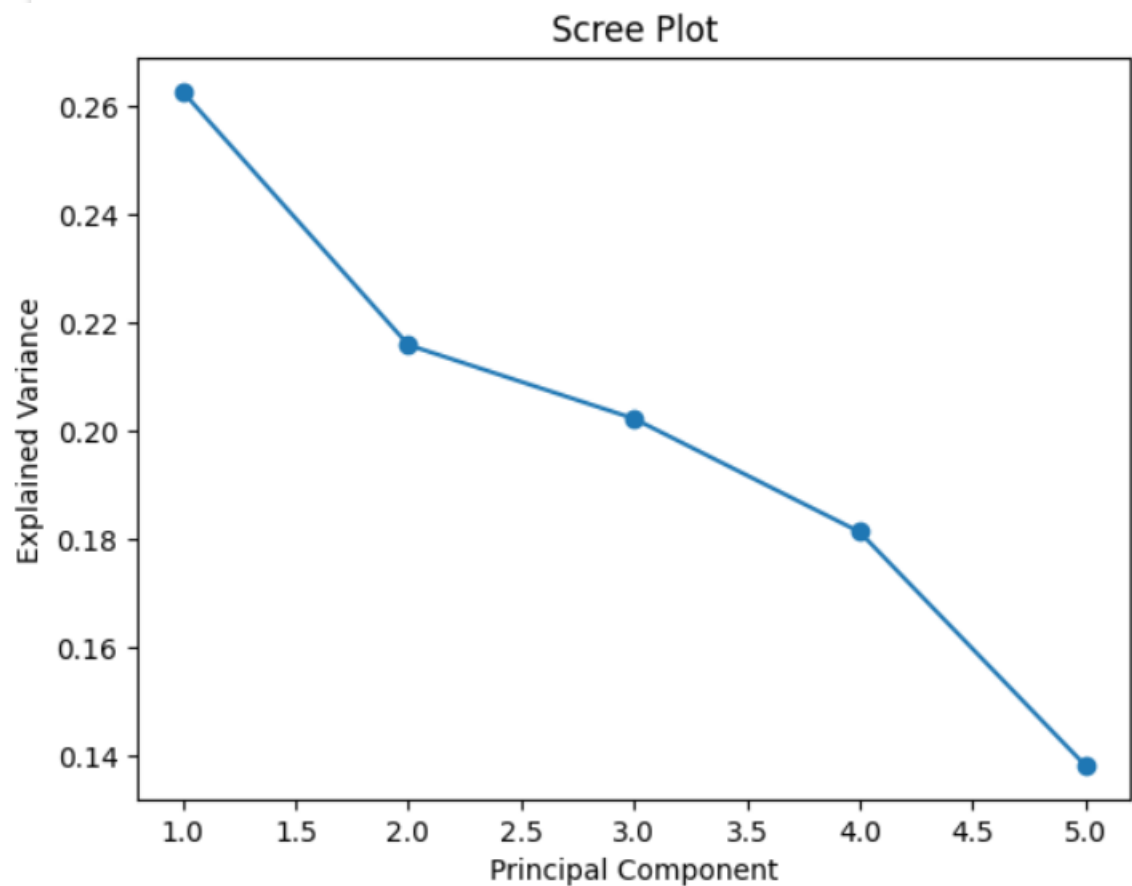
Remaining components: Account for very little variance.

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).

```
✓ 0s  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()
```



3b. Exploiting Correlation

Find Python codes written in Google Colab here:

[WQU_msc_financial_engineering/Financial-Data-Group-Work-Project1-Module5/Task3b_Exploiting_Correlation.ipynb](#) at master · [ezekielibe/WQU_msc_financial_engineering](#)

TASK 4: EMPIRICAL ANALYSIS OF ETFS

a. Finding the 30 largest holdings

S/N	Real Estate Select Sector SPDR Fund (XLRE)
1	American Tower Corporation (AMT) - 9.99%
2	Prologis, Inc. (PLD) - 8.69%
3	Welltower Inc. (WELL) - 8.68%
4	Equinix, Inc. (EQIX) - 7.52%
5	Realty Income Corporation (O) - 4.78%
6	Digital Realty Trust, Inc. (DLR) - 4.50%
7	Simon Property Group, Inc. (SPG) - 4.50%
8	Public Storage (PSA) - 4.38%
9	Crown Castle Inc. (CCI) - 4.30%
10	CBRE Group, Inc. (CBRE) - 3.52%
11	VICI Properties Inc. (VICI) - 3.24%
12	CoStar Group, Inc. (CSGP) - 3.22%
13	Ventas, Inc. (VTR) - 2.89%
14	Extra Space Storage Inc. (EXR) - 2.80%
15	AvalonBay Communities, Inc. (AVB) - 2.78%
16	Iron Mountain Incorporated (IRM) - 2.36%
17	SBA Communications Corporation (SBAC) - 2.34%
18	Equity Residential (EQR) - 2.21%
19	Weyerhaeuser Company (WY) - 1.88%
20	Mid-America Apartment Communities, Inc. (MAA) - 1.83%
21	Invitation Homes Inc. (INVH) - 1.82%
22	Essex Property Trust, Inc. (ESS) - 1.71%
23	Kimco Realty Corporation (KIM) - 1.33%
24	Healthpeak Properties, Inc. (DOC) - 1.31%
25	Alexandria Real Estate Equities, Inc. (ARE) - 1.24%
26	UDR, Inc. (UDR) - 1.20%
27	Camden Property Trust (CPT) - 1.18%
28	Regency Centers Corporation (REG) - 1.11%
29	Host Hotels & Resorts, Inc. (HST) - 0.92%
30	BXP, Inc. (BXP) - 0.86%

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. Compute the daily returns.

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

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

d. Compute the covariance matrix.

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	\
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. Compute the PCA.

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. Compute the SVD.

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

6. Perform Singular Value Decomposition (SVD)

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.