Functional data analysis of stocks during COVID-19

Antonio De Patto, Danial Yntykbay, Jackie Islam

May 27, 2025

Abstract

The aim of this report is to analyse different commercial sectors during the Covid 19 period. In particular, we will consider the stocks of 64 companies from 8 different macro sectors, studying in particular which companies have been most affected by this period of crisis.

1 Data Presentation

The data set considered includes weekly stock closing prices from January 1, 2020, to December 31, 2022. This dataset provides a comprehensive overview of eight primary industrial sectors, namely automobile, fashion and clothing, food and beverage, healthcare, technology, logistics, oil and gas, and travel and tourism. For each sector, data was collected for eight representative stocks, resulting in a total of 64 stocks monitored over a period of 156 weeks. The dataset contains a total of 10,048 data points, capturing the weekly price movements of these stocks. This data is of significant value for conducting sector-wise trend analysis, enabling a comprehensive examination of the performance of each sector and comparative analysis across sectors. The main stocks considered were:

- **Automobile**: VW - Ferrari - Stellantis - Renault - Mercedes - BMW - Tesla - Toyota
- **Fashion and clothing**: Kering - Capri - Hermès - LVMH - Richemont - Adidas - Nike - Puma
- **Food and beverage**: Nestlè - Uniliver - Danone - Bonduelle - Pepsi - McDonalds - Kellogs - kraft Heinz
- **Healthcare**: Sanofi - Novartis - Bayer - AstraZeneca - UCB - Merck - Argenx - GSK
- **Tech**: Spotify - Netflix - Nvidia - Meta - Apple - IBM - Microsoft - Google
- **Logistic**: Zalando - UPS - Amazon - DHL - FedEx - Maersk - Walmart - SF express
- **Oil and Gas**: Shell - Eni - Enel - Engie - Orsted - Chevron - Repsol - total energies
- **Travel and tourism**: Trivago - Booking - Ryanair - Lyft - Trip.com - Tripadvisor - Hilton - Uber

1.1 Depth Analyses Before Transformation and Smoothing

Depth measures provide a framework for ranking functions from the center outward, offering insights into centrality, outliers, and variability in high-dimensional or functional datasets. Among these, Euclidean depth evaluates centrality based on the inverse of the Euclidean distance to a central location, and is sensitive to the global geometry of the data. A low Euclidean depth value, such as 0.00002917722 for the deepest point in a dataset, indicates that even the most central observation lies far from the overall center under this metric, due to high dispersion or the influence of outliers. In contrast, minimum volume depth considers the smallest volume that contains a given proportion of the data and is more robust to outliers, while Fraiman-Muniz depth integrates the rank of each function over the domain, offering a more significant view of centrality over time or space. The observation that the deepest point differs significantly from the column-wise medians in variables like UPS.Close, MSFT.Close, and GOOG.Close suggests a breakdown in the symmetry or unimodality assumptions often implicit in median-based summaries. This discrepancy signals that certain points may be pulling the distribution tails, reinforcing the need to examine potential outliers or atypical shapes that affect depth calculations and challenge the representativeness of conventional summary statistics. For this reason, in the next paragraph we will proceed with further transformations of our data, so as to make the data less sensitive to outliers and to

enhance the reliability of depth-based methods in capturing the true central structure of the dataset. By mitigating the impact of outliers, we aim to obtain more stable and interpretable depth rankings, which are crucial for downstream tasks such as clustering and classification. From the graph in Figure 2 we can see how the distance between the median point and the deepest and most central one is very high, indicating that the data is not symmetrically distributed around a clear central tendency. In functional data analysis, such a discrepancy often suggests the presence of skewness, heavy tails, or outliers that distort traditional measures of centrality like the pointwise (column-wise) median. Theoretically, when data follow a symmetric and unimodal distribution, the deepest point—defined according to a functional depth measure—should closely align with the pointwise median. However, a large distance between them implies that the notion of centrality captured by depth (which considers the entire function trajectory or global shape) diverges significantly from the marginal behavior summarized by the medians

1.1.1 Euclidean depth

```
> st[mdE,]
     ZAL.DE.Close        UPS.Close       AMZN.Close      DHL.DE.Close
         47.25000        207.17999        146.81750         42.44000
         FDX.Close       AMKBY.Close        WMT.Close  X002352.SZ.Close
        219.28000         15.27000         47.54333         53.69000
        SPOT.Close       NFLX.Close       NVDA.Close       META.Close
        148.91000        380.14999         25.93100        231.84000
        AAPL.Close        IBM.Close       MSFT.Close       GOOG.Close
        175.06000        128.89000        310.88000        141.06300
    VOW3.DE.Close    RACE.MI.Close   STLAM.MI.Close      RNO.PA.Close
        151.00000        177.89999         14.15600         22.99000
     MBG.DE.Close     BMW.DE.Close       TSLA.Close         TM.Close
         62.14000         75.32000        267.29666        165.92999
     KER.PA.Close       CPRI.Close      RMS.PA.Close      MC.PA.Close
        557.20001         50.16000       1113.50000        590.00000
     CFR.SW.Close     ADS.DE.Close      NKE.DE.Close      PUM.DE.Close
        104.00000        203.70000        108.44000         71.86000
     SAN.PA.Close    NOVN.SW.Close     BAYN.DE.Close       AZN.Close
         93.46938         75.47266         55.94000         61.37000
     UCB.BR.Close     MRK.DE.Close       ARGX.Close       GSK.L.Close
        100.50000        179.20000        286.47000       1581.55164
    NESN.SW.Close        UL.Close      BN.PA.Close      BON.PA.Close
        115.74000         44.39000         52.76000         17.18000
        PEP.Close       MCD.Close         K.Close       KHC.Close
        159.00000        232.57001         57.57747         37.82000
        SHEL.Close      ENI.MI.Close    ENEL.MI.Close    ENGI.PA.Close
         50.35000         12.92400          5.82600         11.53400
    ORSTED.CO.Close       CHV.F.Close      REP.MC.Close      TTE.PA.Close
        829.59998        144.72000         11.46800         45.75500
      SHEL.Close.1    ENI.MI.Close.1   ENEL.MI.Close.1   ENGI.PA.Close.1
         50.35000         12.92400          5.82600         11.53400
  ORSTED.CO.Close.1     CHV.F.Close.1     REP.MC.Close.1     TTE.PA.Close.1
        829.59998        144.72000         11.46800         45.75500
```

```
> apply(st,2,median)
     ZAL.DE.Close        UPS.Close       AMZN.Close      DHL.DE.Close
         66.55000        175.50500        155.11150         40.22500
         FDX.Close       AMKBY.Close        WMT.Close  X002352.SZ.Close
        229.71000         11.60500         46.21500         61.58500
        SPOT.Close       NFLX.Close       NVDA.Close       META.Close
        223.00000        480.54001         14.61325        249.07000
        AAPL.Close        IBM.Close       MSFT.Close       GOOG.Close
        135.38000        128.15134        245.14500        104.76375
    VOW3.DE.Close    RACE.MI.Close   STLAM.MI.Close      RNO.PA.Close
        151.89000        177.42500         13.50200         30.13000
     MBG.DE.Close     BMW.DE.Close       TSLA.Close         TM.Close
         57.85313         76.27000        225.39667        153.31499
     KER.PA.Close       CPRI.Close      RMS.PA.Close      MC.PA.Close
        564.75000         46.55000       1066.25000        601.10001
     CFR.SW.Close     ADS.DE.Close      NKE.DE.Close      PUM.DE.Close
         97.14000        256.25000        110.13000         77.19500
     SAN.PA.Close    NOVN.SW.Close     BAYN.DE.Close       AZN.Close
         86.79584         76.67636         53.35000         56.31500
     UCB.BR.Close     MRK.DE.Close       ARGX.Close       GSK.L.Close
         87.80000        155.32500        293.18500       1503.27991
    NESN.SW.Close        UL.Close      BN.PA.Close      BON.PA.Close
        109.47158         54.25500         56.19500         20.20000
        PEP.Close       MCD.Close         K.Close       KHC.Close
        148.05500        233.28500         60.92958         36.13000
        SHEL.Close      ENI.MI.Close    ENEL.MI.Close    ENGI.PA.Close
         41.60500         10.52400          7.22800         12.13100
    ORSTED.CO.Close       CHV.F.Close      REP.MC.Close      TTE.PA.Close
        842.39999         89.35000         10.63039         39.74750
      SHEL.Close.1    ENI.MI.Close.1   ENEL.MI.Close.1   ENGI.PA.Close.1
         41.60500         10.52400          7.22800         12.13100
  ORSTED.CO.Close.1     CHV.F.Close.1     REP.MC.Close.1     TTE.PA.Close.1
        842.39999         89.35000         10.63039         39.74750
```

Figure 1: Comparison of the deepest point vs. columns-wise medians

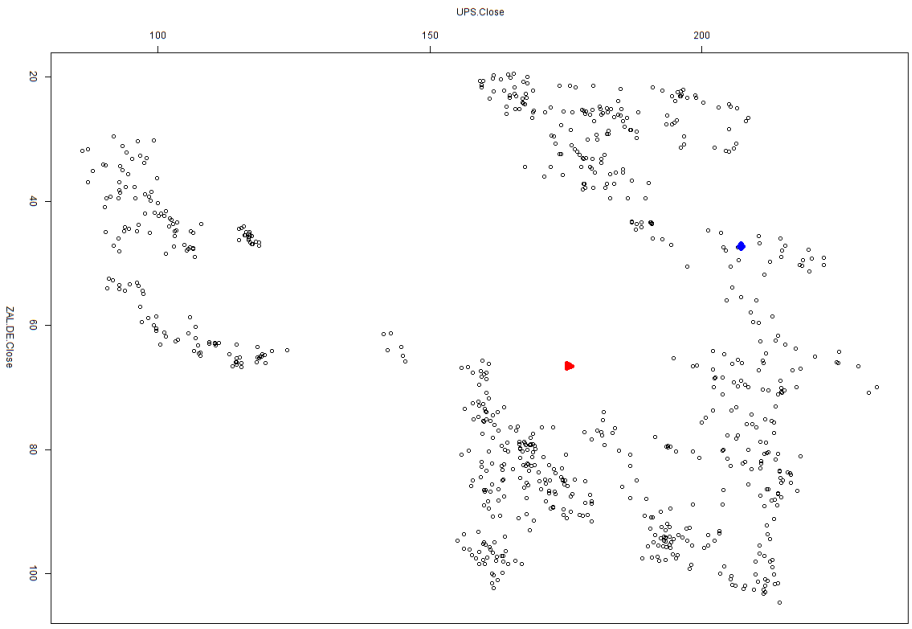


Figure 2: Comparison between the deepest point(in blue) and the median point(in red) in a scatter plot

1.1.2 Minimum Volume Depth (MBD) and Frainman-Muniz depth

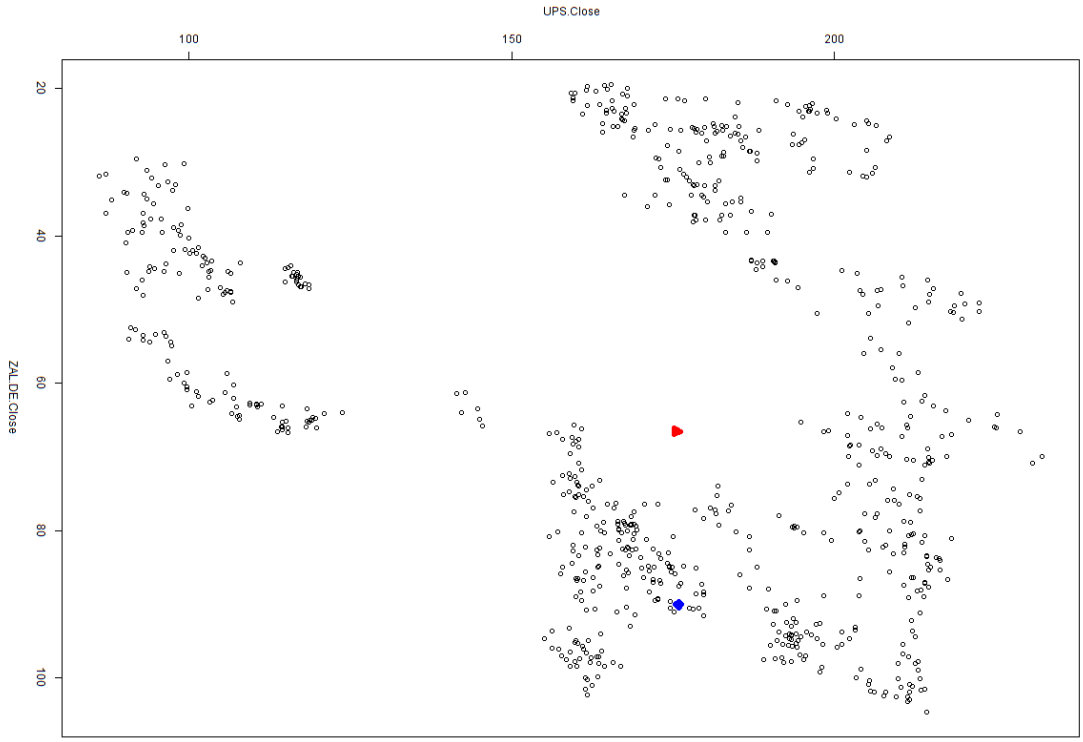


Figure 3: Comparison between the deepest point(in blue) and the median point(in red) in a scatter plot

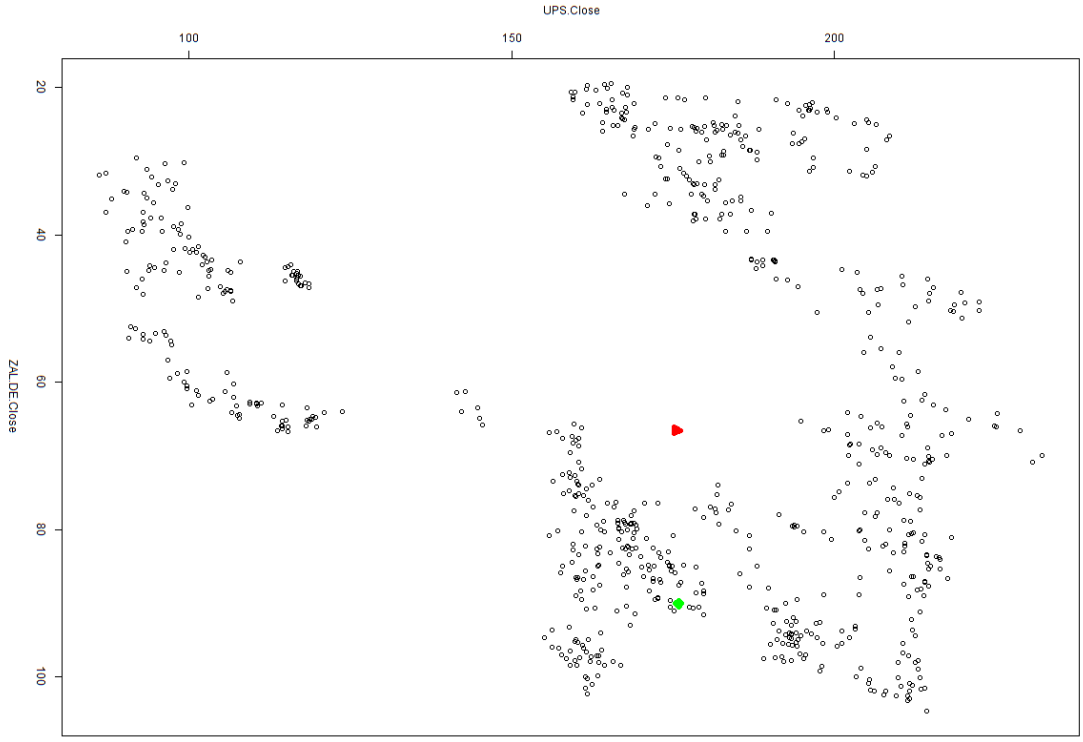


Figure 4: Comparison between the deepest point(in green) and the median point(in red) in a scatter plot

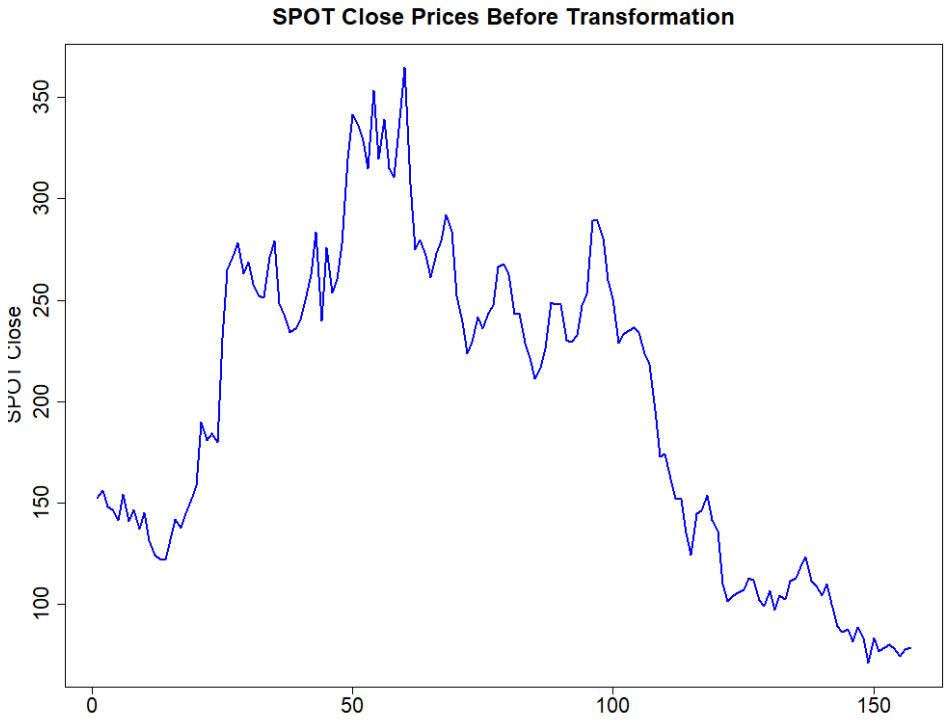
As seen previously for the Euclidean distance, the same problem occurs with the Minimum Volume Depth: the distance between the central point (i.e., the point with maximum depth) and the median point remains quite marked, as illustrated in Figure 3. This indicates that, even under a more robust depth measure like Minimum Volume Depth—which identifies centrality based on the smallest region containing a specified portion of the data—the global shape of the data is still influenced by irregularities, such as outliers or structural asymmetries. Similarly, the Fraiman-Muniz depth, which integrates the ranks of the functional data across the domain to provide a global assessment of centrality, shows a comparable degree of separation between the deepest point and the median. However, it is important to note that this distance is less pronounced than in the Euclidean case (Figure 2), highlighting the increased robustness of rank-based and volume-based depths compared to distance-based ones. Theoretically, this reflects the fact that Euclidean depth is highly sensitive to scale and outliers, while both Minimum Volume and Fraiman-Muniz depths are designed to resist such distortions, offering a more reliable estimate of functional centrality in complex or noisy datasets. This reinforces the need to carefully choose depth measures in accordance with the structure and quality of the data.

1.2 Data Transformation

Since the stocks considered originate from different countries, currency fluctuations significantly influence the volatility of each stock. To mitigate the impact of such volatility and bring all variables to a comparable scale, a logarithmic transformation of the features is applied. This transformation reduces volatility, ensuring that the variation scale remains consistent across all variables. By compressing the data range, logarithmic transformation not only stabilizes variance but also diminishes the influence of outliers, making trends and patterns more discernible. This is particularly valuable when dealing with financial data, where extreme values can skew analysis and lead to misleading interpretations. Logarithmic return:

$$R = 100 * \log\left(\frac{P_t}{P_{t-1}}\right)$$

- R is the log return
- P_t is the stock price at time t
- P_{t-1} is the stock price at the previous time period



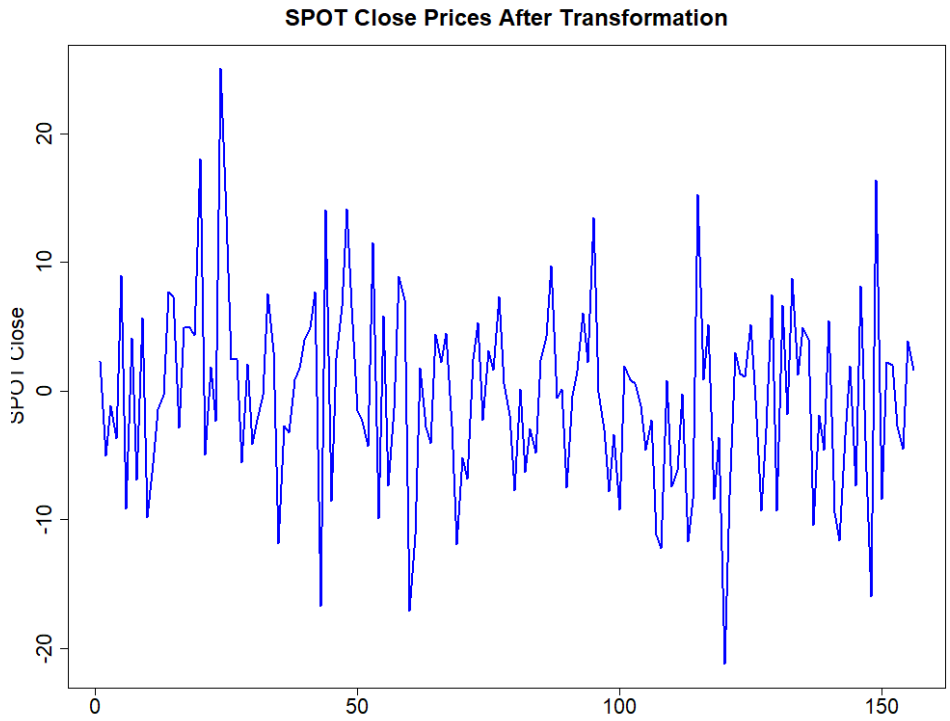


Figure 5: Example of Spotify Stock Before and After Transformations.

In Figure 5, we present the logarithmic transformation of Spotify's stock data. This transformation effectively reduces the volatility to a controlled range, allowing for a more standardized comparison with other stocks within the IT sector. The visual representation clearly illustrates how extreme fluctuations are compressed, emphasizing the relative stability and enabling a more direct analysis of underlying trends. This is particularly beneficial when evaluating sector-wide patterns, as the normalization of data ensures that the magnitude of price movements across different assets is brought to a comparable scale. By mitigating the effects of large price swings, we can more accurately assess the inherent risk and identify consistent patterns across multiple stocks. With this foundation established, we can now proceed with a more in-depth analysis, focusing on identifying key trends, anomalies, and underlying market behaviors that may not have been as apparent prior to the transformation.

2 Smoothing

Smoothing refers to the process of estimating a smooth underlying function from discrete and often noisy observations. Since functional data are typically sampled over a continuum (e.g., time, space), smoothing helps reconstruct the full trajectory or shape of a curve by reducing random variability while preserving essential features such as trends, peaks, and curvature. Effective smoothing is crucial for subsequent tasks like differentiation, integration, and functional principal component analysis.

Overview of Smoothing Methods

- **B-splines:** A basis function approach using piecewise polynomials connected at specific points (knots). B-splines allow local control of smoothness and are widely used in FDA for their flexibility and computational efficiency.
- **Local regression kernel:** Also known as local polynomial regression, this method fits a low-degree polynomial to data points in a neighborhood of each target point, weighted by a kernel function. It adapts well to local structure and varying smoothness in the data.
- **Nadaraya-Watson kernel:** A special case of kernel regression where the estimate at a given point is a weighted average of nearby observations. The weights are determined by a kernel function centered at the target point.
- **Normal kernel:** A Gaussian-shaped kernel that assigns smoothly decreasing weights to observations as their distance from the target point increases. It is commonly used due to its infinite support and smooth behavior.
- **Triweight kernel:** A compact, smooth kernel with a bell shape that assigns zero weight to points outside a certain range. It produces very smooth estimates while emphasizing observations close to the target.
- **Uniform kernel:** A simple kernel that gives equal weight to all observations within a fixed bandwidth and zero to others. It is easy to implement but can lead to less smooth estimates compared to more sophisticated kernels.

2.1 B-spline Method

The application of the B-spline smoothing method with penalization produced a smooth estimate of the underlying functional data, balancing fidelity to the observed data and the desired smoothness of the fitted curve. The optimal smoothing parameter, selected via generalized cross-validation (GCV), was found to be approximately $\lambda \sim 371$, with an associated number of basis functions around 7. This relatively low number of basis functions suggests that the data exhibit a globally smooth structure that does not require a highly flexible basis for accurate representation.

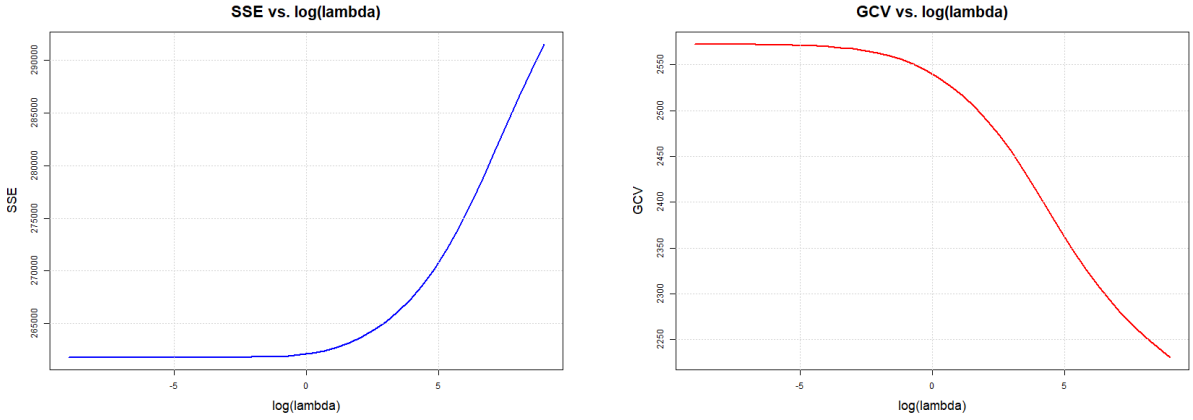


Figure 6: Comparison of SSE and GCV with respect to lambda.

The minimized GCV score of approximately 2230 indicates an effective trade-off between bias and variance, reinforcing the suitability of the chosen smoothing level. Figure 6 provides a visual comparison

of SSE and GCV across a range of λ values, showing how both metrics evolve as the smoothness penalty increases. This helps to illustrate the typical U-shaped behavior of GCV, with a clear minimum that guides the optimal selection of λ. Overall, the B-spline approach with penalized smoothing effectively reconstructs the functional structure while avoiding overfitting, making it a robust tool for functional data analysis.

2.2 Kernel Smoothing Results

Kernel smoothing was applied to the functional data using a range of bandwidths between 3 and 70, with the optimal value selected through generalized cross-validation (GCV). This approach allows for a data-driven choice of the smoothing parameter, ensuring that the final estimate balances the trade-off between oversmoothing (which can obscure important features) and undersmoothing (which can preserve too much noise). The bandwidth in kernel smoothing plays a role analogous to the smoothing parameter in spline-based methods: smaller bandwidths emphasize local structure and result in more variable, less smooth estimates, while larger bandwidths yield smoother curves by averaging over a broader neighborhood. Exploring a broad sequence of bandwidth values provides insight into the sensitivity of the smoother to local variations in the data and helps identify the most appropriate level of smoothness for the functional representation. The GCV criterion offers an objective way to select the optimal bandwidth, aiming to minimize prediction error. This process is particularly important in functional data analysis, where the underlying curves must be estimated accurately to support subsequent analyses such as functional principal component analysis, clustering, or regression. The chosen bandwidth thus directly influences the quality and interpretability of the smoothed functional data.

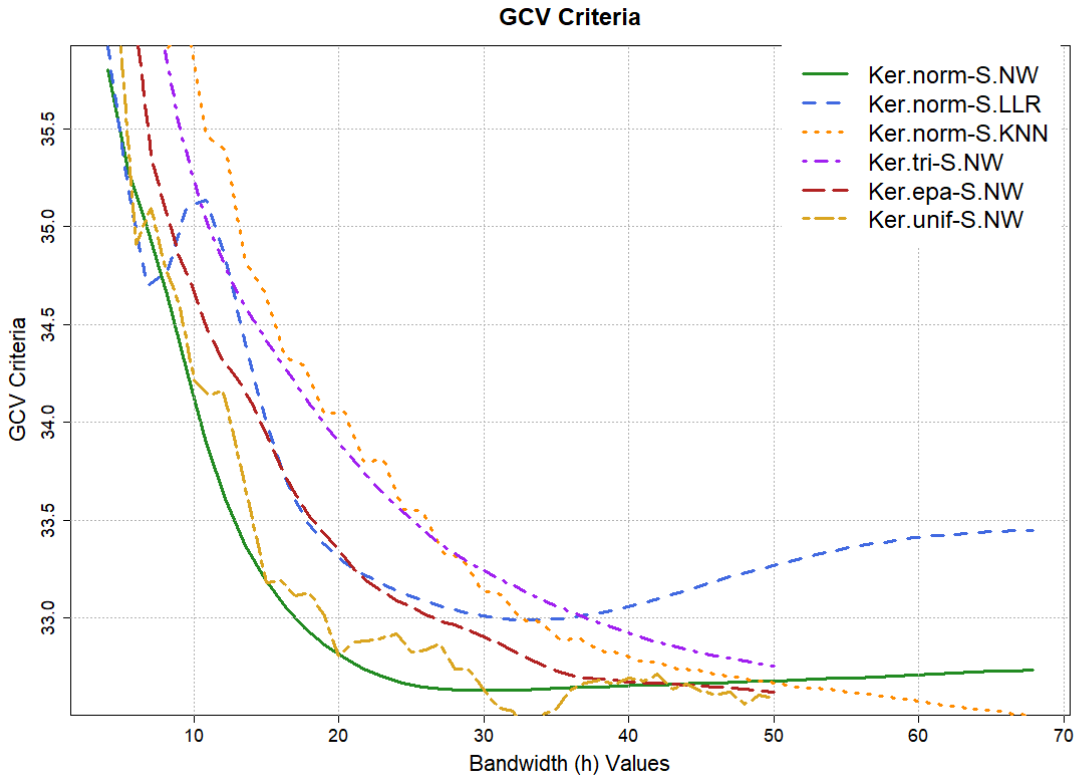


Figure 7: GCV criteria respect to bandwidth

2.3 Comparing Optimal GCV and SSE

To evaluate the performance of the various smoothing techniques, we compared the optimal values of Generalized Cross-Validation (GCV) and Sum of Squared Errors (SSE) obtained across methods. GCV serves as an internal model selection criterion that balances goodness-of-fit with model complexity, while SSE measures the raw discrepancy between the observed data and the fitted values. The comparison reveals that each method exhibits distinct trade-offs between smoothness and fidelity to the data. B-

spline smoothing, with an optimal GCV of approximately 2230 and an SSE around 307923, provided a globally smooth fit that effectively captured the overall trend while maintaining a relatively low model complexity. In contrast, kernel-based methods, which rely heavily on bandwidth selection, demonstrated more localized adaptation to data features. The range of tested bandwidths (from 3 to 70) showed how sensitive these methods are to the smoothing parameter, with GCV again guiding the optimal choice. Although kernel methods may yield slightly lower SSE in some cases due to their flexibility, they are also more prone to overfitting when the bandwidth is too small. Overall, the B-spline method offered a robust compromise between interpretability and accuracy, while kernel smoothing provided finer local adjustment at the potential cost of increased variance. This comparison underscores the importance of context-specific smoothing choices in functional data analysis, depending on whether the priority is global structure or local detail.

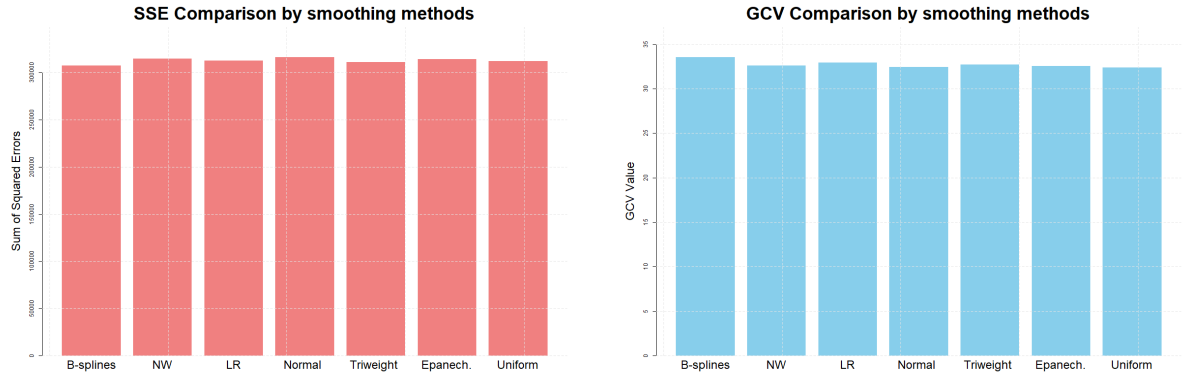


Figure 8: Comparison of optimal GCV and SSE among smoothing methods.

2.4 B-splines vs. Normal kernel and Triweight kernel

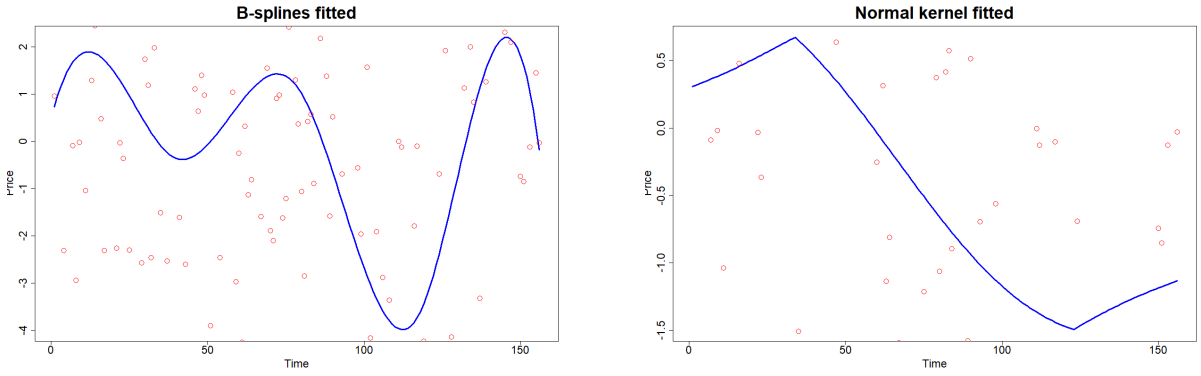


Figure 9: Comparison of fitted and actual values for B-splines and Normal kernel.

As illustrated in Figure 8, the B-spline method achieves the lowest Sum of Squared Errors (SSE), indicating that it provides the best fit to the observed data among the methods considered. SSE measures the total squared discrepancy between the fitted curve and the data points, and a lower value generally reflects a closer adherence to the observations. However, SSE alone does not account for model complexity and may favor models that overfit the data. To address this limitation, we also consider the Generalized Cross-Validation (GCV) criterion, which balances fit quality with model smoothness by penalizing excessive complexity. According to GCV, the best-performing model is the Normal Kernel smoother, which attains the lowest GCV score among the tested methods.

To better understand these differences, we examine Figures 9 and 10, which compare the behavior of the B-spline, Normal Kernel, and Triweight Kernel methods. These plots show the smoothed functions produced by each method, allowing for visual evaluation of their flexibility and generalization. While the Normal Kernel achieves the lowest GCV, its smoother may introduce slight local variations due to its sensitivity to nearby data points. The Triweight Kernel, which has compact support and a sharper

weighting function, also performs well but may be more sensitive to the choice of bandwidth. In contrast, the B-spline method produces a smooth, stable curve that avoids both overfitting and underfitting, maintaining a good balance between flexibility and regularity. Although its GCV is not the lowest, the B-spline smoother demonstrates robust performance, making it a compelling choice for functional data analysis where interpretability and global structure are important.

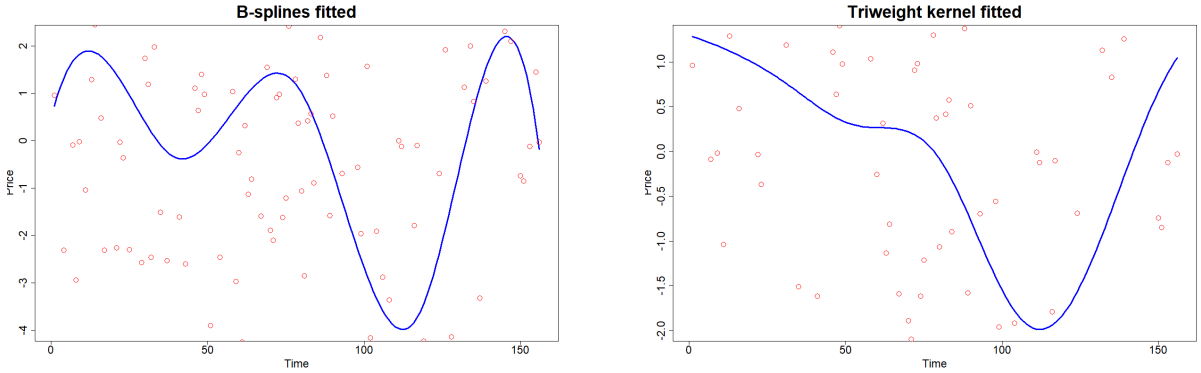


Figure 10: Comparison of optimal GCV and SSE among smoothing methods.

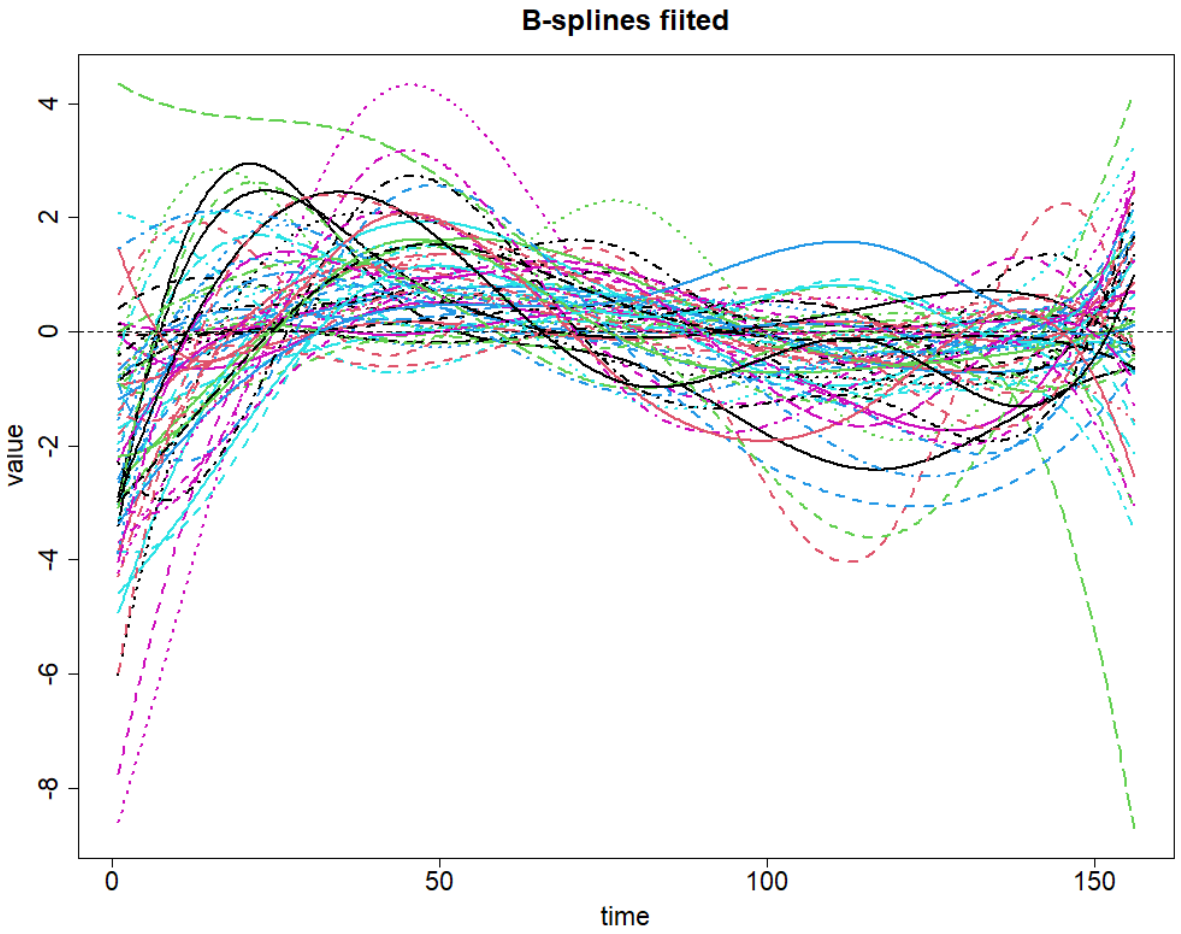


Figure 11: B-splines fitted for all stocks

3 Exploratory Data Analysis (EDA)

3.1 Functional Derivatives

To uncover the dynamic behavior in the time series data, we computed the first and second derivatives of the smoothed functions of the B spline.

- **First Derivative**: Represents the rate of change in the logarithmic returns of the stocks over time, highlighting periods of acceleration or deceleration. Each line represents one stock's derivative curve across 156 weeks (pandemic period). The high variation at the beginning (Weeks 0-20) and the end (Weeks 140-156) corresponds to the early pandemic shock and post-pandemic recovery phases. The mid-period (Weeks 50-120) has derivatives mainly centered around zero, indicating more stable or consolidating market conditions. Some stocks exhibit positive (e.g Tech and healthcare stocks) or negative spikes (e.g travel and automobile stocks), indicating bursts of buying or selling pressure or news-driven movements. Divergent trajectories suggest outlier behavior or sector-specific dynamics.

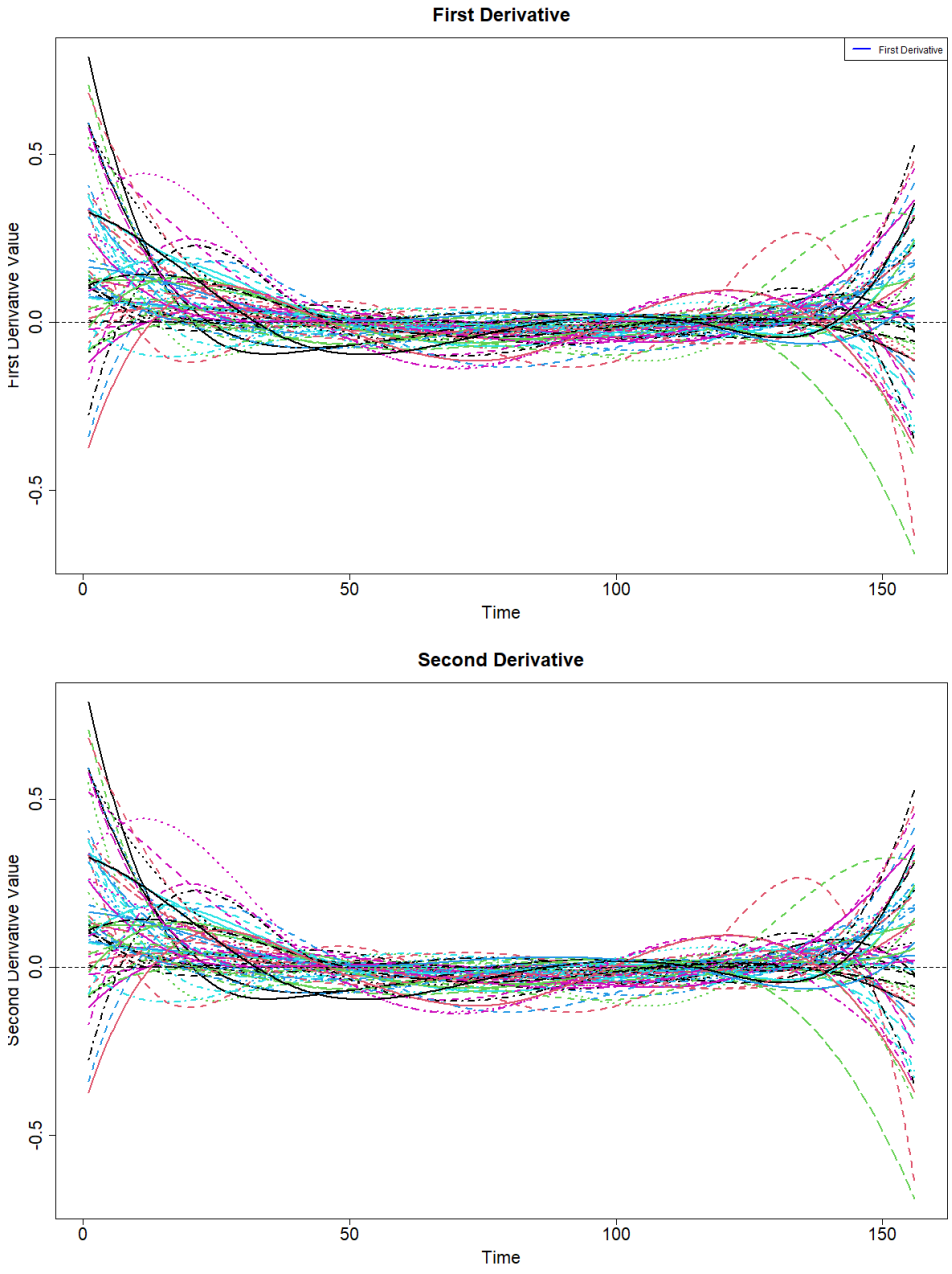


Figure 12: 1st and 2nd derivatives of B-splines smoothed

- **Second Derivative**: It Captures the curvature of the functional data, providing insight into inflection points and shifts in behavior. This plot illustrates the acceleration or deceleration in stock returns, identifying turning points and trend shifts. Strong positive and negative curvature at the start and end highlighting abrupt changes linked to policy shifts, stimulus announcements or vaccine developments. The mid-period is relatively flat, indicating gradual adaptation or market saturation. Sharp inflections in certain stocks suggest company-specific or sector-driven shocks.

  Both derivatives revealed structural differences in volatility and market sensitivity between companies.

- **Outlier Detection**: Outlier detection in functional data involves identifying curves or observations that deviate significantly from the "typical" behavior represented by the mean curve.

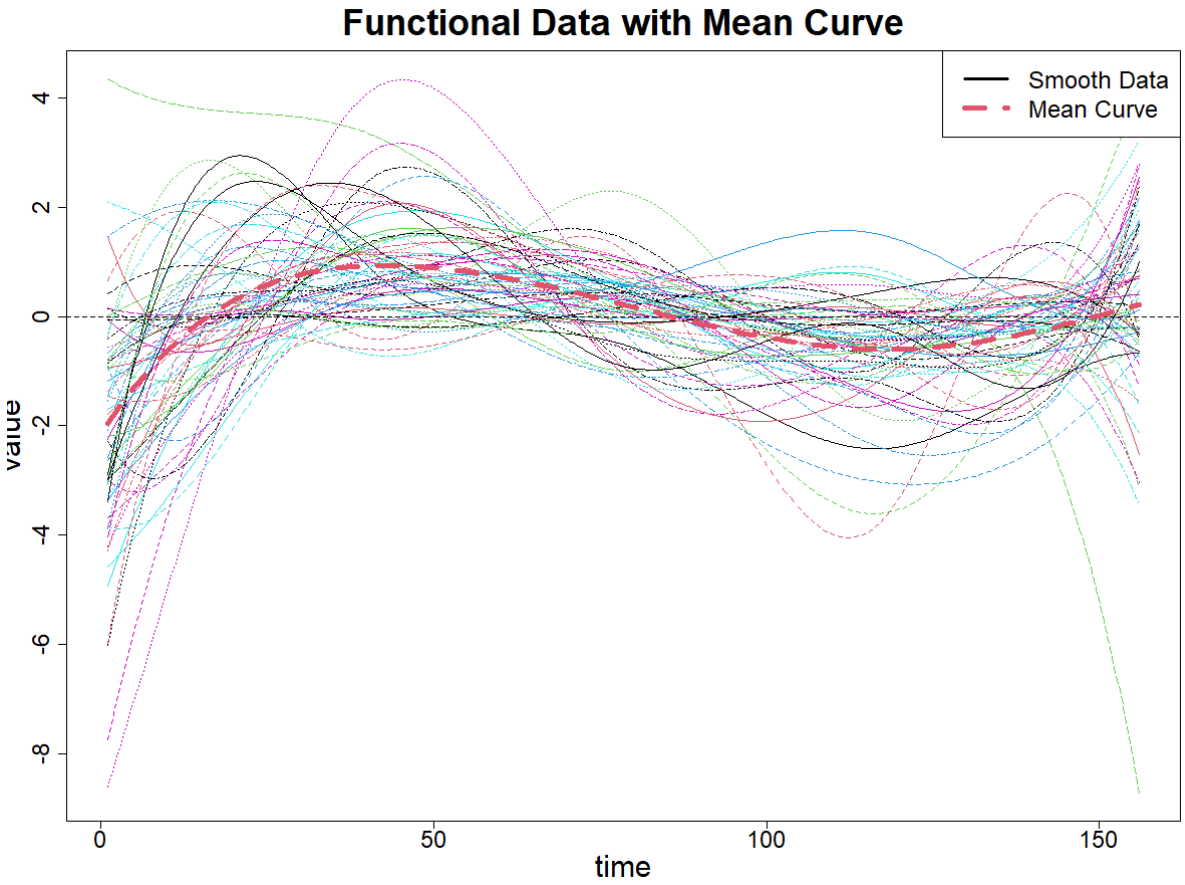


Figure 13: Mean Curve

In the Figure the gray curves show significant variability in stock returns over time, capturing heterogeneous behaviors across sectors and individual stocks. Some curves show pronounced peaks and troughs, reflecting periods of high volatility likely linked to pandemic-related economic events. The overlaid mean curve summarizes the central tendency of the entire functional dataset, effectively representing the average stock return trajectory. Its smoother and more moderate pattern compared to individual curves indicates typical market behavior averaged over diverse stocks. The mean function reveals key temporal features such as market disruptions and recovery phases during the pandemic, evidenced by fluctuations around specific weeks. This curve helps identify common periods of market stress and rebound. The underlying B-spline smoothing framework allows for computation of functional standard deviation bands. The wide spread of individual curves around the mean suggests heterogeneous impacts of COVID-19 across sectors and companies. The choice of penalized B-splines with optimized basis dimension and smoothing parameter (lambda) ensures a balance between fitting the data well and avoiding overfitting. The mean curve further confirms this, providing a stable summary of the functional data.

3.2 Centrality and Depth Analysis

We evaluated the centrality of curves using multiple depth measures:

- **Euclidean Depth**: It shows Sensitivity to outliers and scale, showing divergence from median.

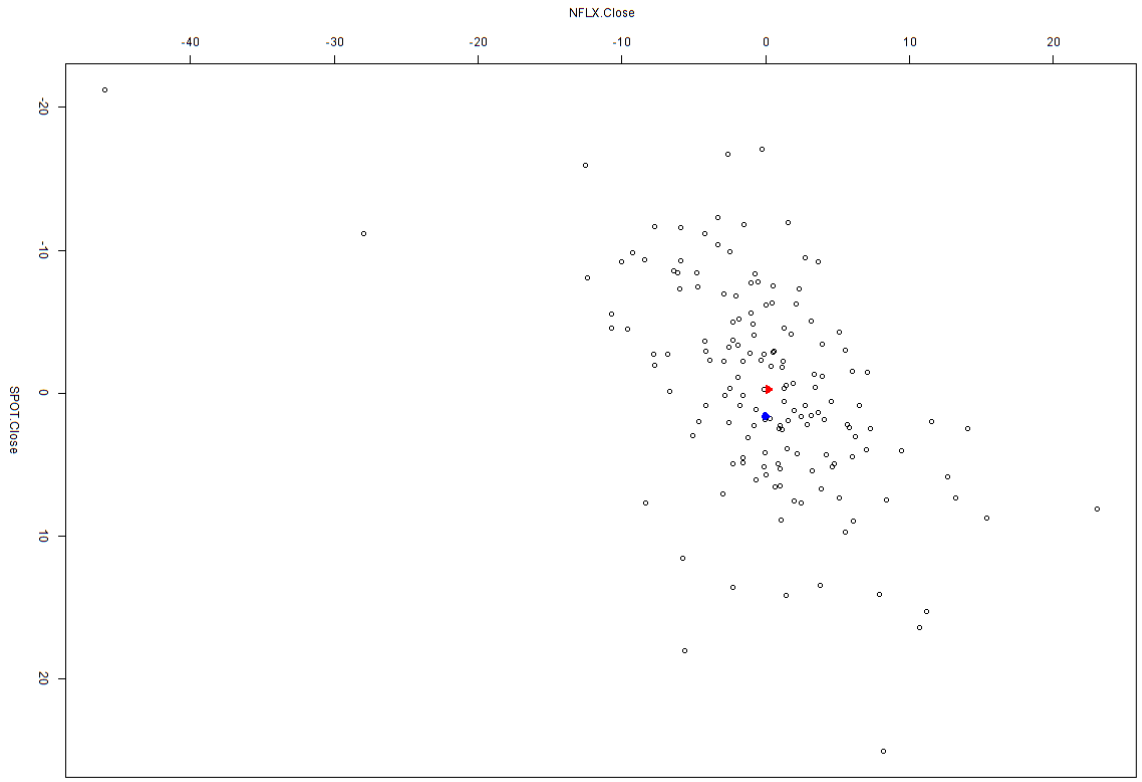


Figure 14: Comparison between the deepest point(in blue) and the median point(in red) in a scatter plot

Here, the blue Point (Deepest) represents the most central observation in terms of Euclidean depth. In this dataset, however, the value is very low (0.00002917722), suggesting high dispersion in the data. In the plot the red Point (median) represents the coordinate wise median across all stock dimensions. The visible separation between the deepest point (blue) and the median point (red) implies the presence of outliers or skewed distributions. The dataset is not symmetric because the deepest point deviates from the traditional center. Stocks such as UPS.Close, MSFT.Close, or GOOG.Close significantly influence this spread. As we know that, Euclidean depth is sensitive to global geometry, so even small deviations in a few variables can reduce the depth value. Thus, the deepest point may not visually appear "central" in every dimension. The unreliability of Euclidean depth in the presence of outliers suggests the need for robust depth measures like MBD (Modified Band Depth) or Fraiman-Muniz depth for further analysis.

- **Friman-Muniz and Modified Band Depth (MBD)**: To obtain a more robust notion of centrality, we employed rank-based depth measures: Friman-Muniz depth and Modified Band Depth (MBD). Unlike Euclidean depth, these methods consider the global structure of the data and are less affected by outliers or asymmetries. Both plot display scatter plots of the stock data projected on the `SPOT.Close` and `NFLX.Close` axes. In both cases, the most central observation as determined by the depth measure (blue point) is very close to the coordinate-wise median (red point), suggesting that the data are relatively symmetric and centrally clustered in this projection.

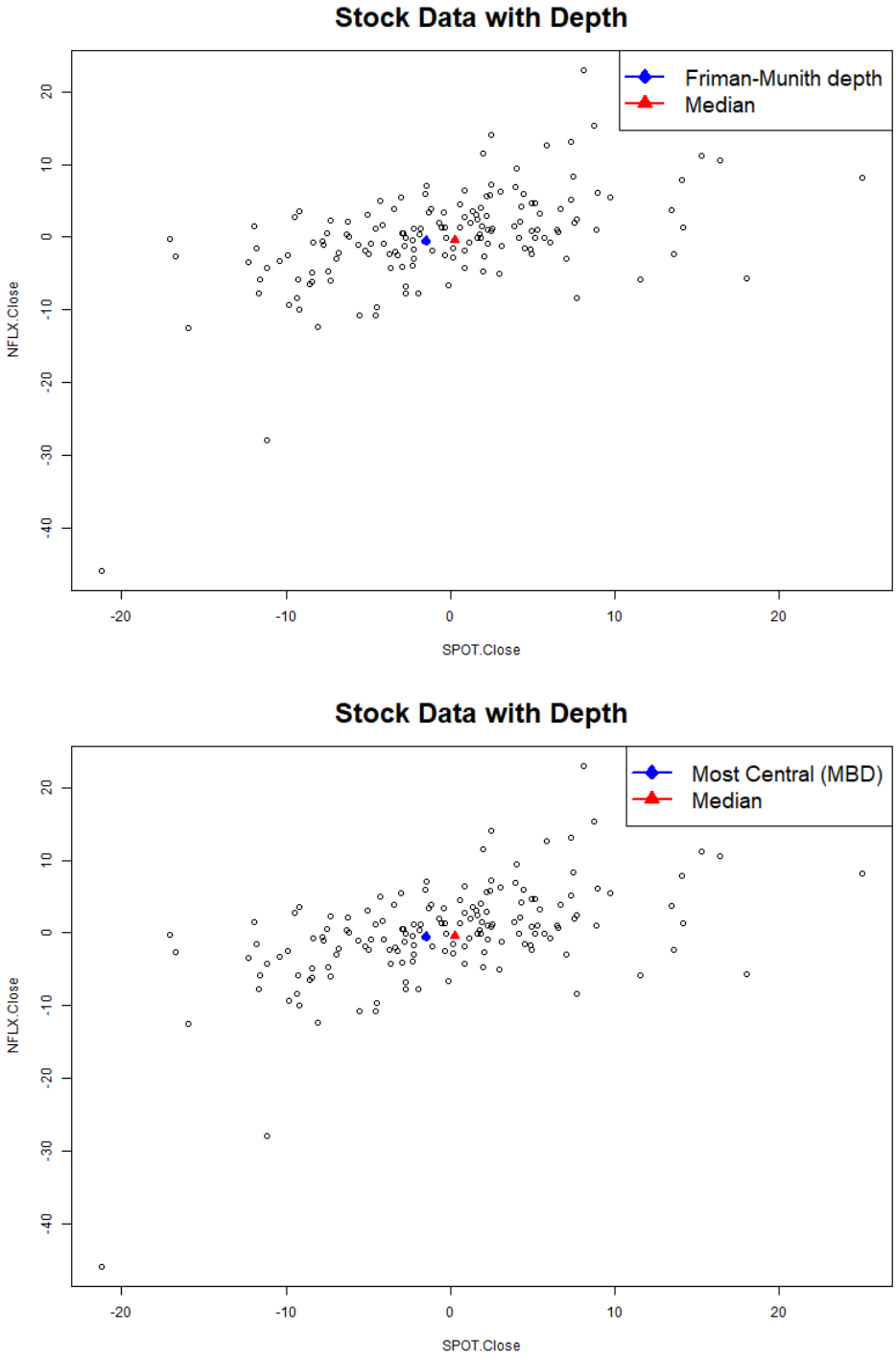


Figure 15: Friman-Muniz depth and MBD

These results confirm that both MBD and Friman-Muniz depth provide reliable estimates of centrality and indicate that the dataset, while complex, maintains a relatively coherent central structure. Their robustness makes them preferable to Euclidean depth in financial data with potential noise or outliers. The FM depth captures centrality by aggregating the rank of each observation across the domain, making it less sensitive to outliers and capable of identifying global central structures in functional data. In the plot, the blue point (FM depth) and red point (median) lie relatively close together. This indicates a more symmetric and balanced distribution compared to the Euclidean depth plot. The proximity suggests that rank-based centrality (FM) aligns well with coordinate-wise centrality, validating the robustness of this method for central tendency detection in moderately noisy datasets. MBD is also a rank-based method but focuses on how often a func-

tion lies within bands formed by other pairs of functions. It offers a more resilient measure against local noise and isolated fluctuations. Again, the blue (MBD central point) is very close to the red (median). This closeness confirms the central symmetry and lack of extreme outliers in this 2D projection. The fact that both FM and MBD identify nearly identical central points shows strong internal consistency in the central structure of the data and supports the reliability of the results derived from either method.

Comparisons showed significant asymmetry and highlighted potential outliers.

3.3 Covariance and Contour Visualization

We calculated the bivariate covariance function $v(s,t)$ to evaluate the comovement:

- **3D perspective plot**: Illustrated variance strength over time. The 3D surface plot provided depicts the bivariate covariance function (variance surface) of the smoothed stock return curves over time (weeks).

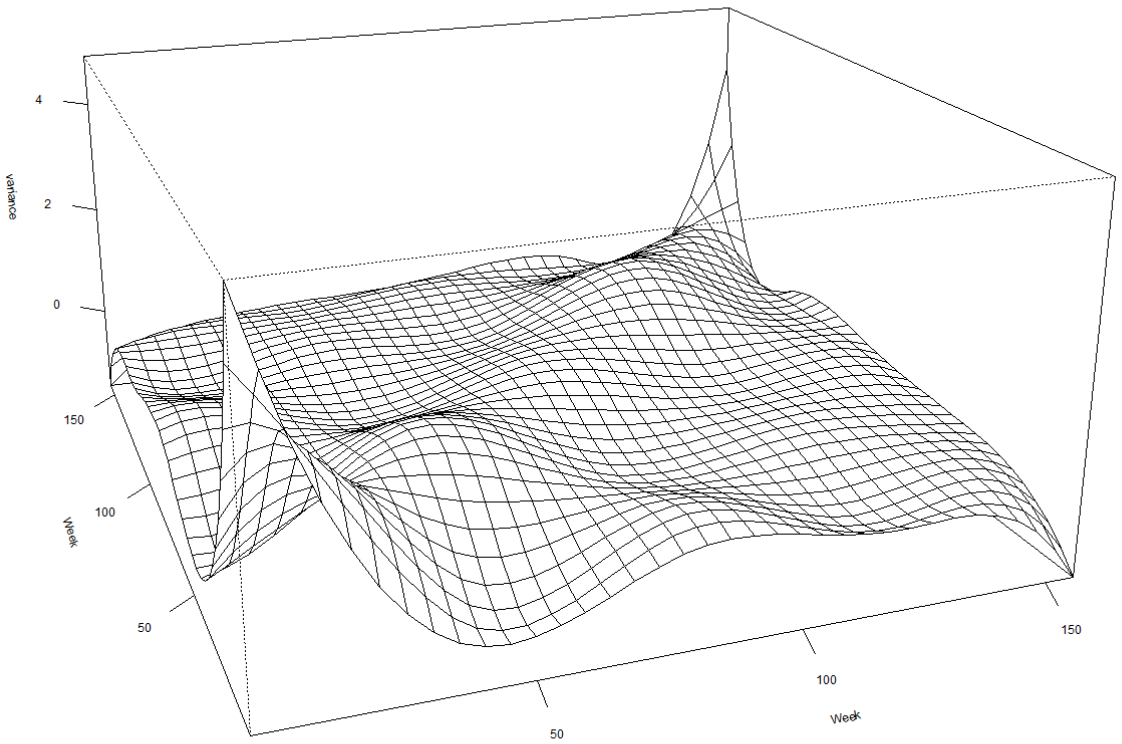


Figure 16: 3D visualization of the variance function for smoothed stock returns

In The above plot, both the X and Y axes represent time (weeks), covering the entire study period. The Z axis represents the variance (or covariance) of the stock returns between two time points. Variance surface shows how variance evolves and interacts across different weeks. Peaks indicate time periods with higher variability or volatility in stock returns. Valleys correspond to periods of lower variance, suggesting more stable or consistent behavior among stock returns during those weeks. As the variance surface is not flat, indicating that volatility changes significantly over time. Sharp peaks near some early weeks suggested that the initial phases of the COVID-19 period had

higher market uncertainty and larger fluctuations. The smoother sections later on indicate periods where stock returns behaved more consistently or had lower volatility. The interaction between weeks shows cross-temporal covariance, meaning that variance at one time point is related to variance at other times, reflecting persistence or memory in volatility. This 3D variance plot offers insight into temporal heteroscedasticity (changing variance over time), crucial for risk modeling and financial forecasting during turbulent periods like the pandemic. Identifying time frames with high variance can help investors and analysts focus on critical periods with heightened risk. The covariance structure can be used in functional principal component analysis (FPCA) and functional clustering to understand dominant modes of variability.

- **Contour plot**: This is a Simplified interpretation of localized volatility.

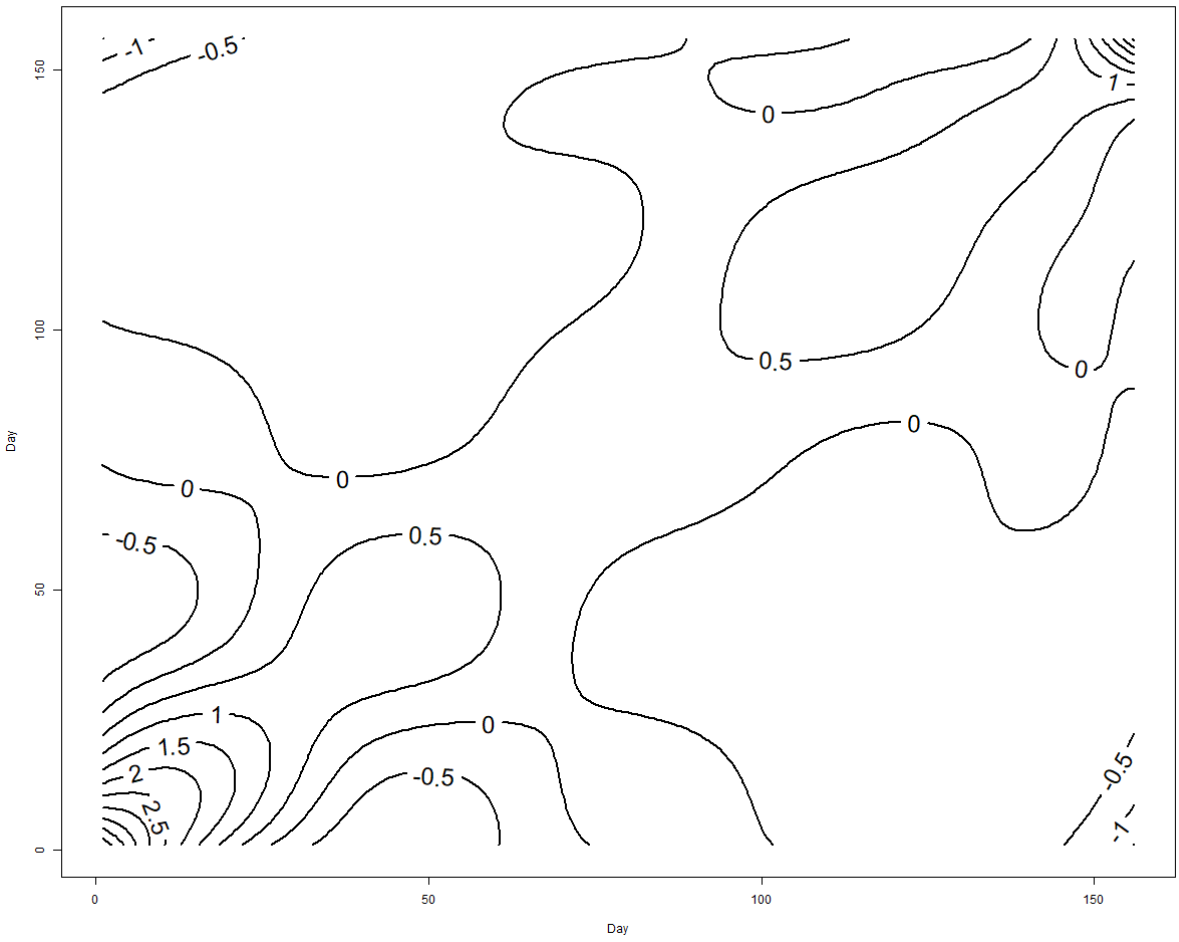


Figure 17: Contour plot

In the plot both X and Y axes represent time (days or weeks). The contour lines represent levels of covariance between stock returns at different pairs of time points. Lines with higher values (positive contours) indicate time points where the variance or covariance is stronger. Negative contours show time regions with negative covariance, suggesting inverse relationships between returns at those time points. The plot shows how variance changes smoothly over time, and where the covariance between returns is strong or weak. The highest concentration of contours is near the origin (early weeks), indicating high variance and covariance early in the period, likely corresponding to the onset of COVID-19 market impacts. The shape and spacing of contour lines suggest non-stationary covariance, with variance/covariance evolving in complex ways over the time window. The existence of both positive and negative covariance regions points to dynamic relationships among returns at different times. The contour plot complements the 3D variance surface by giving a clearer visual on where variance and covariance concentrate or weaken over time. It helps identify periods of

heightened risk and instability.

3.4 Boxplot of Functional Data

By identifying symmetry and the spread of functional values, the boxplot presents the distribution of the smoothed functional stock return values evaluated at multiple time points across the studied period (156 weeks). Each box corresponds to the distribution of stock return values at a particular evaluation point on the time grid used for the functional data object.

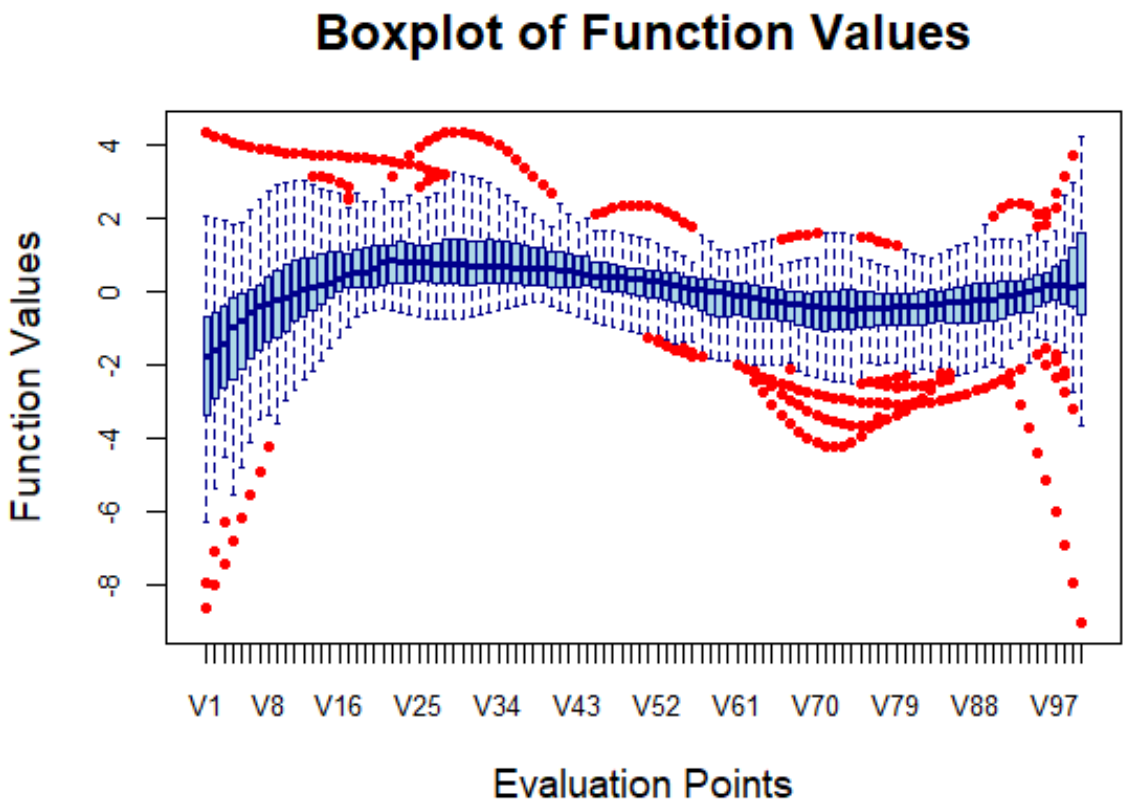


Figure 18: Box plot Of Function Values

The central line inside each box indicates the median value of returns at each time point, while the box edges show the interquartile range (IQR). The boxplot clearly depicts the temporal evolution of median returns and their variability. Red points beyond the whiskers represent outliers at each time point, signaling unusual or extreme returns that deviate significantly from the bulk of observations. Their presence highlights periods of increased volatility, which may correspond to significant market events during the COVID-19 pandemic. Temporal Volatility Pattern: Observing the shape of the boxes over the time domain allows one to identify phases of heightened or reduced volatility. Wider boxes and more outliers at certain weeks indicate times of elevated risk and market uncertainty. The boxplot provides a comprehensive snapshot of the dynamic distribution of returns, supplementing pointwise statistical summaries like mean and median with detailed distributional characteristics, critical for understanding sectoral responses to the COVID-19 shock. This visual tool complements the smoothing and depth analyses by showcasing the range and extremity of returns captured by the functional data framework, facilitating robust identification of volatility clusters and outliers in stock price trajectories

4 Functional Principal Component Analysis(FPCA)

4.1 Functional Principal Component Variations

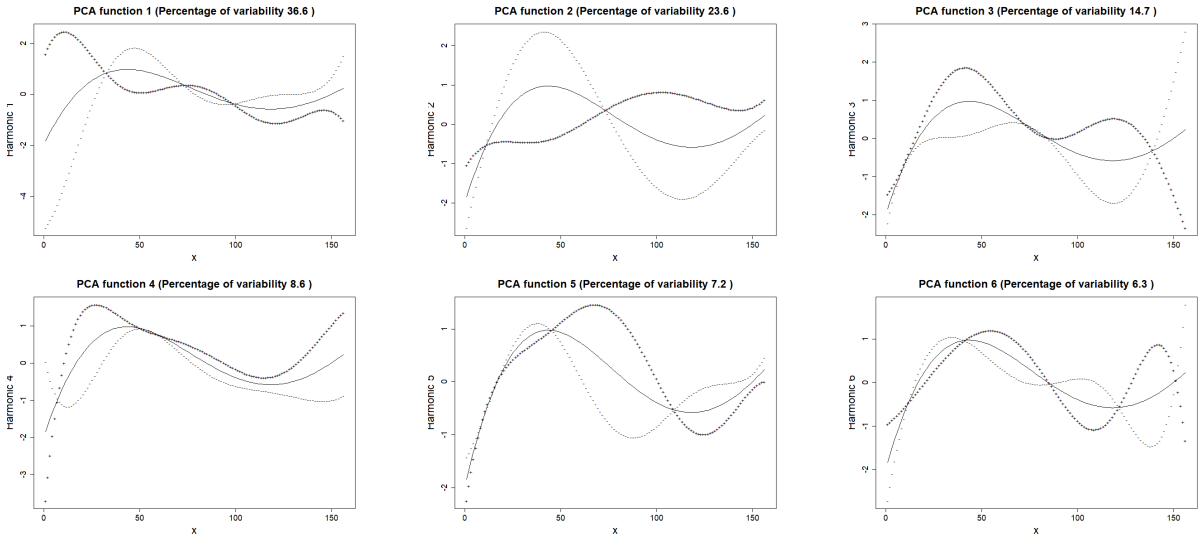


Figure 19: Functional Principal Component Analysis: Plots of the first six principal component functions (PC1 to PC6) showing the major modes of functional variation. Each subfigure shows the mean function (solid line), variability bounds (±2 standard deviations, dotted lines), and functional scores (plus signs). The percentages indicate the variability explained by each component.

In the plot we can see that, the FPCA cumilatively explains 82% variability in the stock price. Where, Principal Component Function 1 (PC1) explaining 36.6% of variability. The x-axis represents the functional domain (e.g., time), while the y-axis shows the amplitude of the principal component function, indicating how the mean function is modified along this component. The solid line is the mean function of the dataset. The dotted lines represent ±2 standard deviations around the mean score, showing typical variation. Plus signs (+) mark individual functional scores projecting observations onto PC1. PC1 captures the largest source of variation, often reflecting broad-scale trends affecting all functions, such as market-wide shifts in stock returns. Principal Component Function 2 (PC2) explaining 23.6% of variability. Similar to PC1, the x-axis is the functional domain and the y-axis the amplitude of PC2. This component captures secondary variation patterns orthogonal to PC1, such as phase shifts or localized oscillations in the data. The mean function (solid) and ±2 standard deviation bounds (dotted) show typical variation ranges. Plus signs (+) denote individual scores on PC2, highlighting the spread of data along this mode. Principal Component Function 3 (PC3) explaining 14.7% of variability. PC3 represents more subtle or localized variations, possibly corresponding to short-term fluctuations or sector-specific effects. The shape shows complex oscillations, with the solid line as the mean function and dotted lines indicating ±2 standard deviations. Individual scores are marked with plus signs (+), illustrating how observations vary along this component. Principal Component Function 4 (PC4) explaining 8.6% of variability. PC4 captures finer-scale variations affecting a subset of functions or specific intervals within the domain. The plot includes the mean function (solid), variability bounds (±2 standard deviations, dotted), and individual scores (+). Though it explains less variance, PC4 adds important nuance for capturing subtle functional differences. Principal Component Function 5 (PC5) explaining 7.2% of variability. This component reveals small-scale or localized variations, including sharp peaks or troughs that characterize unique or rare patterns. The mean function is shown with a solid line, with ±2 standard deviation bounds as dotted lines, and individual functional scores marked by plus signs (+). Principal Component Function 6 (PC6) explaining 6.3% of variability. PC6 captures the smallest retained mode of variation, highlighting subtle or idiosyncratic features in the data. The solid line indicates the mean function, dotted lines show ±2 standard deviations, and plus signs (+) mark individual scores. Together with prior components, PC6 contributes to a near-complete functional representation.

4.2 Varimax Rotations

Functional Principal Component Analysis initially produces orthogonal components ranked by explained variance but is sometimes difficult to interpret due to complex loadings overlapping in the domain. The VARIMAX rotation addresses this by maximizing the variance of squared loads within each component, effectively simplifying the structure. Concentrate component loadings on fewer regions of the functional domain, making each rotated component more interpretable as it highlights distinct modes of variation localized in time or space. This often redistributes the explained variance among the components, but maintains the overall total variance close to that of the unrotated solution. The result is a set of rotated principal components that correspond more clearly to meaningful real-world phenomena, such as different phases of a market cycle, specific sector impacts, or reaction periods to external events (e.g., COVID-19 lockdowns). This improved interpretability supports downstream analyses, such as functional clustering, anomaly detection, or regression, by providing clearer factors linked to data variability.

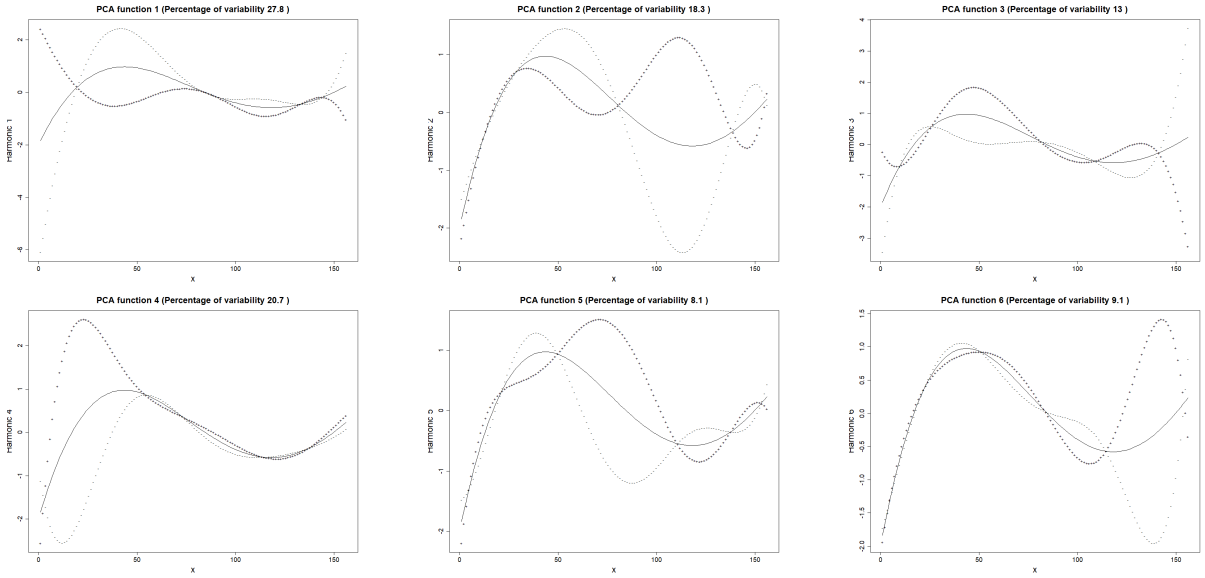


Figure 20: VARIMAX Rotation of FPCA 1 to FPCA 6

Here, the varimax rotation of PCA cumulatively explains the 79.8% variability. PCA1 captures 27.8% variability, PCA2 captures 18.3% variability, PCA3 captures 13.0% variability PCA 4 captured 20.7% and showed strong variability from the mean amongstocks till the 70th week(could explain starting phase). PCA5 captures 9.1% variability and PCA6 captures 27.8% variability. PCA 1 may capture the overall market variability across different phases in response to specific events possibly COVID-19 related. Volatility in stock prices in beginning, mid and end of pandemic. PCA functions captured the most volatility in different time periods.

5 Functional Clustering

The goal of cluster analysis in the context of FDA is to identify groups of similar functions that exhibit comparable patterns over the domain of interest. Selecting the optimal number of clusters is a crucial step in functional clustering. One common criterion for determining the number of clusters is the Bayesian Information Criterion (BIC). The BIC is defined as:

$$BIC = -2 \cdot \ln(L) + k \cdot \ln(n)$$

where:

- L: Likelihood of the model
- K: Number of parameters
- n: Number of observations

In our analysis, we selected the AkjBk model with clusters based on the highest BIC score of -198106.2. Models with more than 5 clusters resulted in empty clusters, indicating potential overfitting or lack of sufficient data structure to support additional clusters.

```
>> K = 2
Error in .fstep(fd, T, lambda) : One cluster is almost empty!
>> K = 3
AkjBk   :        bic = -350051.4
>> K = 4
AkjBk   :        bic = -318623.1
>> K = 5
AkjBk   :        bic = -198106.2
>> K = 6
Error in .fstep(fd, T, lambda) : One cluster is almost empty!
>> K = 7
Error in .fstep(fd, T, lambda) : One cluster is almost empty!
>> K = 8
Error in .fstep(fd, T, lambda) : One cluster is almost empty!
>> K = 9
Error in .fstep(fd, T, lambda) : One cluster is almost empty!
>> K = 10
Error in .fstep(fd, T, lambda) : One cluster is almost empty!
The best model is AkjBk with K = 5 ( bic = -198106.2 )
```

	K	model	bic	aic	icl	nbprm	ll
1	2	AkjBk	NA	NA	NA	NA	NA
2	3	AkjBk	-350051.4	-350021.2	-350051.4	28	-349993.2
3	4	AkjBk	-318623.1	-318573.5	-318623.1	46	-318527.5
4	5	AkjBk	-198106.2	-198033.9	-198106.2	67	-197966.9
5	6	AkjBk	NA	NA	NA	NA	NA
6	7	AkjBk	NA	NA	NA	NA	NA
7	8	AkjBk	NA	NA	NA	NA	NA
8	9	AkjBk	NA	NA	NA	NA	NA
9	10	AkjBk	NA	NA	NA	NA	NA

Figure 21: The choice of the best number of clusters to consider

After identifying the optimal number of clusters, we proceed to visualize the clustering results. This involves comparing the original smoothed data functions against the clustered data to assess how well the clusters represent the underlying data structure.

The AkjBk model is a flexible and powerful approach for clustering functional data within the FunFEM (Functional EM) framework. In this context, each functional observation is first projected into a lower-dimensional discriminative subspace that best captures the between-group variability. The AkjBk model assumes that each cluster k has its own mean function (denoted by the "A" component) and its own covariance structure (denoted by the "Bk" component), making it well-suited for capturing heterogeneous variances across groups. This flexibility allows the model to adapt to varying shapes and amplitudes among clusters. In our analysis, the AkjBk model with K=5 clusters was selected due to its optimal BIC score, suggesting it best captures the inherent structure in the smoothed functional data without overfitting.

The centroids shown in Figure 23 on the right represent the mean function of each cluster, providing a concise summary of the cluster's behavior over the observed domain. In the context of Functional Data Analysis, centroids are obtained by averaging the smoothed functions within each cluster, often using basis expansion coefficients. This yields representative trajectories that capture the central tendency of the data in each group.

By comparing the first 20 individual curves to their respective cluster means, we gain insights into the internal variability and coherence of the clusters. A high degree of similarity between the curves and their corresponding centroids indicates strong homogeneity within clusters. Conversely, large deviations might signal within-cluster heterogeneity or potential misclassifications.

Analyzing the distinctness between centroids also provides a sense of how well-separated the clusters are. Clearly separated centroids reflect good discriminative power in the clustering process. This centroid-based comparison is especially valuable in functional settings, where the visual interpretation of curves and their trends over time plays a crucial role in understanding the clustering structure.

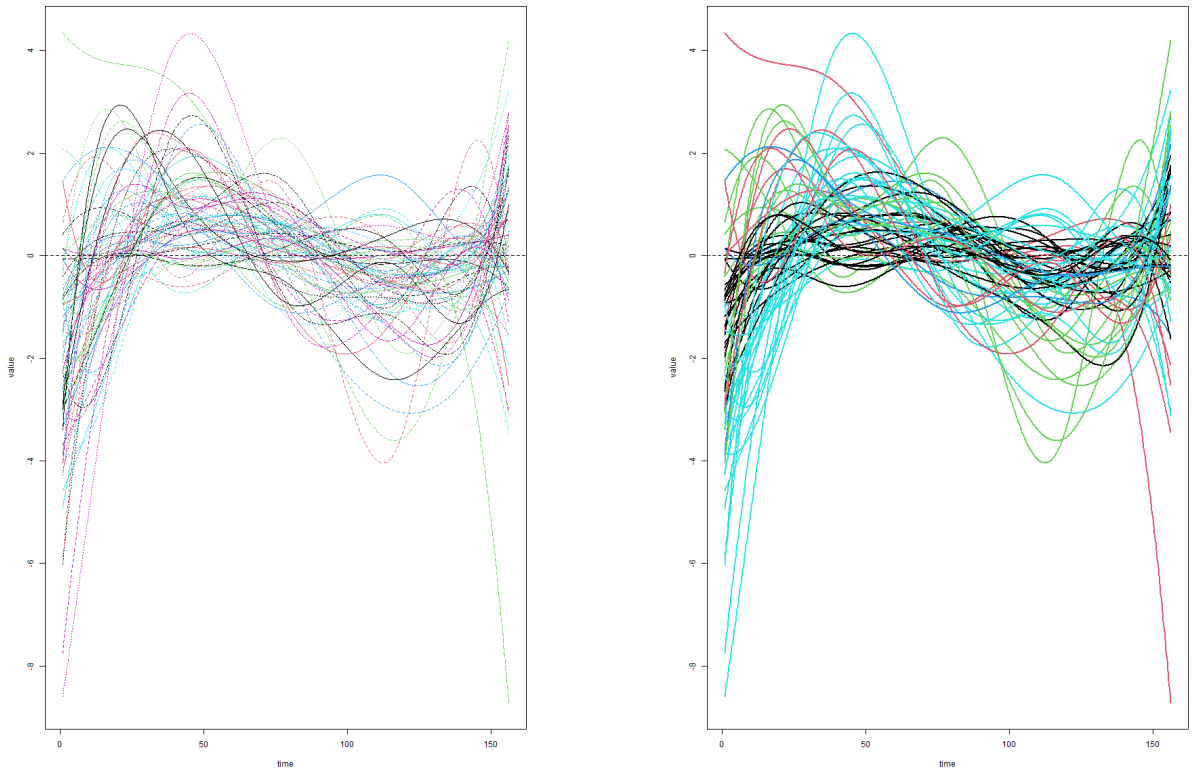


Figure 22: Comparison between original smoothed data and clusters

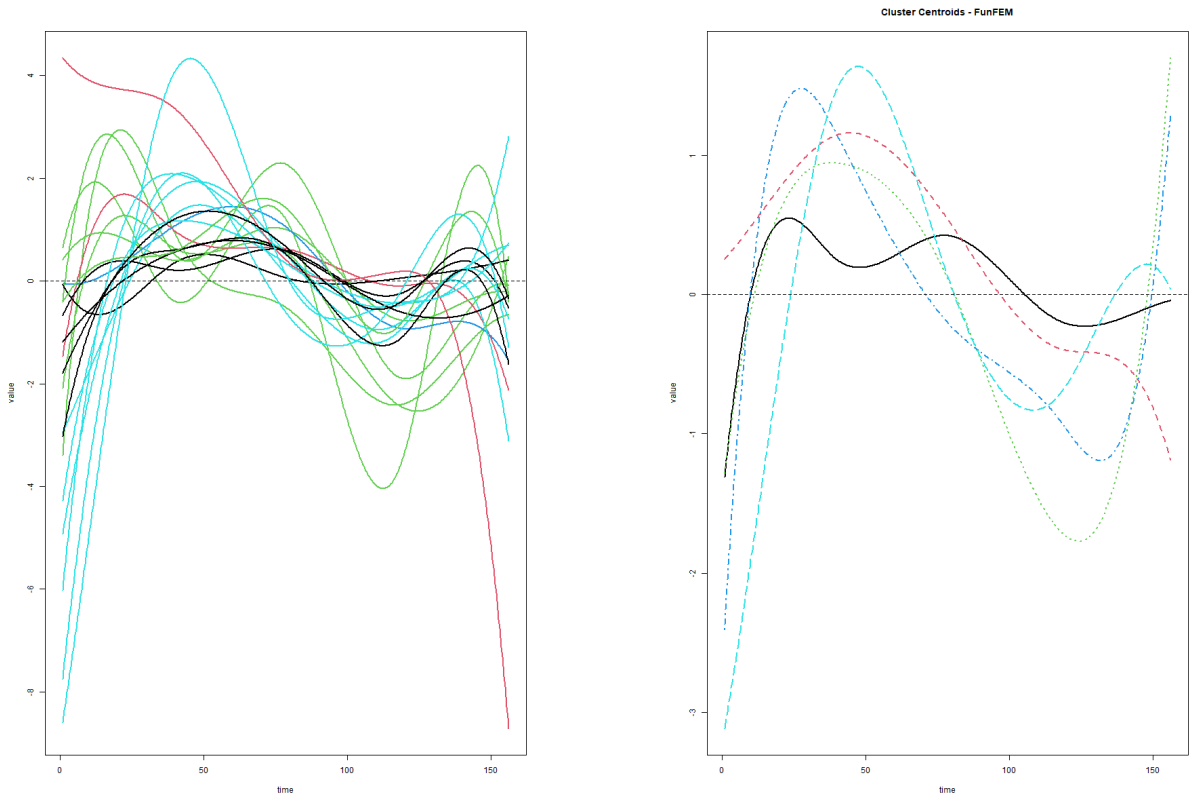


Figure 23: A more in-depth analyses comparing the first 20 curves(on the left) and the mean of each cluster(on the right)

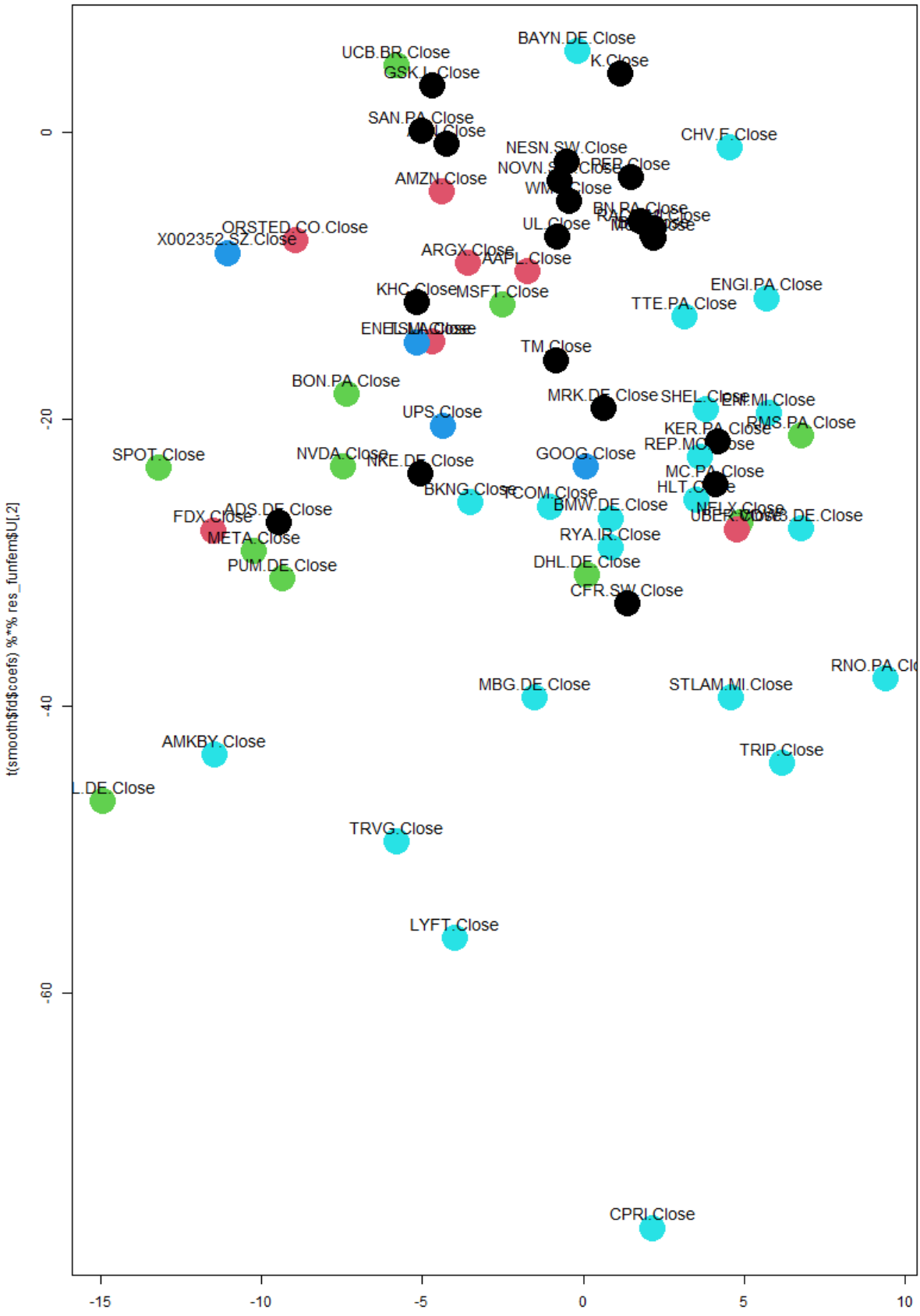


Figure 24: Discriminative Space Plot

From the graph in Figure 22 we can see the transition from the smoothed functions on the left to

those divided into clusters on the right. The division of the clusters is not clear and well-defined as the functions tend to be one above the other, as can be seen by looking at how the original functions are represented. However, each cluster is able to capture the general behavior of the functions. For example, the blue cluster contains a series of functions that have a different trend and performance than the black or green cluster. Among the less crowded clusters and with more variable functions inside we find the blue and red ones, in which the distinction with the other clusters is not so clear and distinct. To have a clearer idea of each cluster we can look at Figure 23, in which we find on the left the first 20 functions of the smoothed dataset and on the right we find the centrodes, which highlight the general behavior of each cluster.

A discriminative space plot in Figure 24 is then utilized to project the smoothed data functions onto a lower-dimensional space where the clusters can be more clearly visualized. This helps in visually assessing the separability of clusters and identifying potential outliers. Stocks that share similar temporal behaviors (e.g., volatility patterns, trends) are close to each other in this 2D space, while dissimilar ones are farther apart.

The light blue cluster (e.g., TRIP, LYFT, CPRI) includes stocks positioned far from the origin, indicating unique patterns likely due to macroeconomic cycles or structural market differences. The green cluster (e.g., SPOT, META, NVDA, PUM.DE) mainly contains tech-focused or digital companies with dynamic and volatile behavior. The black cluster (e.g., KO, UL, WMT, RDSA) includes consumer staples and energy giants, showing more stable and balanced functional forms. The red cluster (e.g., AMZN, MSFT, ARKG, ORSTED) reflects potentially aggressive growth or transition dynamics, while the blue cluster (e.g., GOOG, BKNG, BMW, UPS) groups diverse firms with mixed behaviors like industrial-tech or logistics companies.

Notably, outliers like CPRI and LYFT sit on the margins of the plot, reflecting the most unique functional behaviors. From this graph it is possible to better notice how the separability of the clusters remains unclear and distant but in general it can be said that companies belonging to similar sectors, such as technology or emerging companies, tend to group together.

5.1 Hierarchical Clustering (HCLUST)

Hierarchical clustering (HCLUST) is a bottom-up clustering method that recursively merges data points or clusters based on a chosen linkage criterion. We utilized the complete linkage method to form 5 clusters.

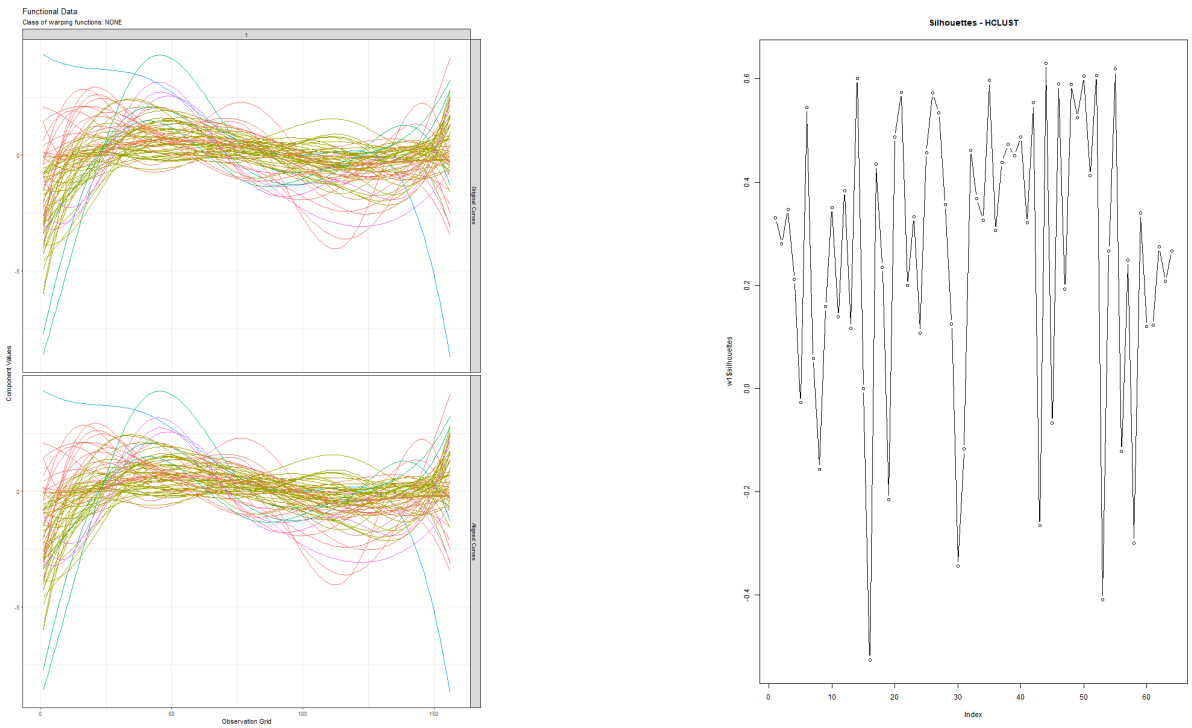


Figure 25: Hclust with 5 clusters and complete linkage method

To evaluate the clustering quality, we used the silhouette plot, defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where:

- s(i): Silhouette score for point
- a(i): Average distance from point to other points in the same cluster
- b(i): Minimum average distance from point to points in other clusters

The silhouette plot serves as a powerful diagnostic tool to evaluate the quality and coherence of the hierarchical clustering (HCLUST) results. Each silhouette coefficient $s(i)$ reflects how well a data point fits within its assigned cluster compared to neighboring clusters. Values close to +1 indicate that the data point is appropriately grouped, demonstrating strong intra-cluster similarity and distinct separation from other clusters. Coefficients near 0 suggest that the point lies on the boundary between two clusters, making its assignment ambiguous. Negative values, particularly those below zero, often signal potential misclassification, as the point appears to be closer to a different cluster than the one it was assigned to.

In our analysis, we observed significant variability in the silhouette coefficients across data points, reflecting uneven cluster cohesion. Some instances—such as those around indices 17 and 58—exhibited negative silhouette values, indicating likely misclassifications. A substantial number of points displayed coefficients close to zero, implying that these data points are on the fringes of their respective clusters and may require further review. On the other hand, several data points, particularly within the range of indices 45 to 55, demonstrated high positive silhouette values, signaling well-defined and robust cluster structures.

Overall, the silhouette plot reveals a mixed clustering outcome: while certain clusters exhibit clear internal consistency and strong separation from others, several ambiguous or poorly grouped observations suggest that further investigation or refinement of the clustering parameters might be necessary to enhance cluster validity.

5.2 K-means Clustering Analysis

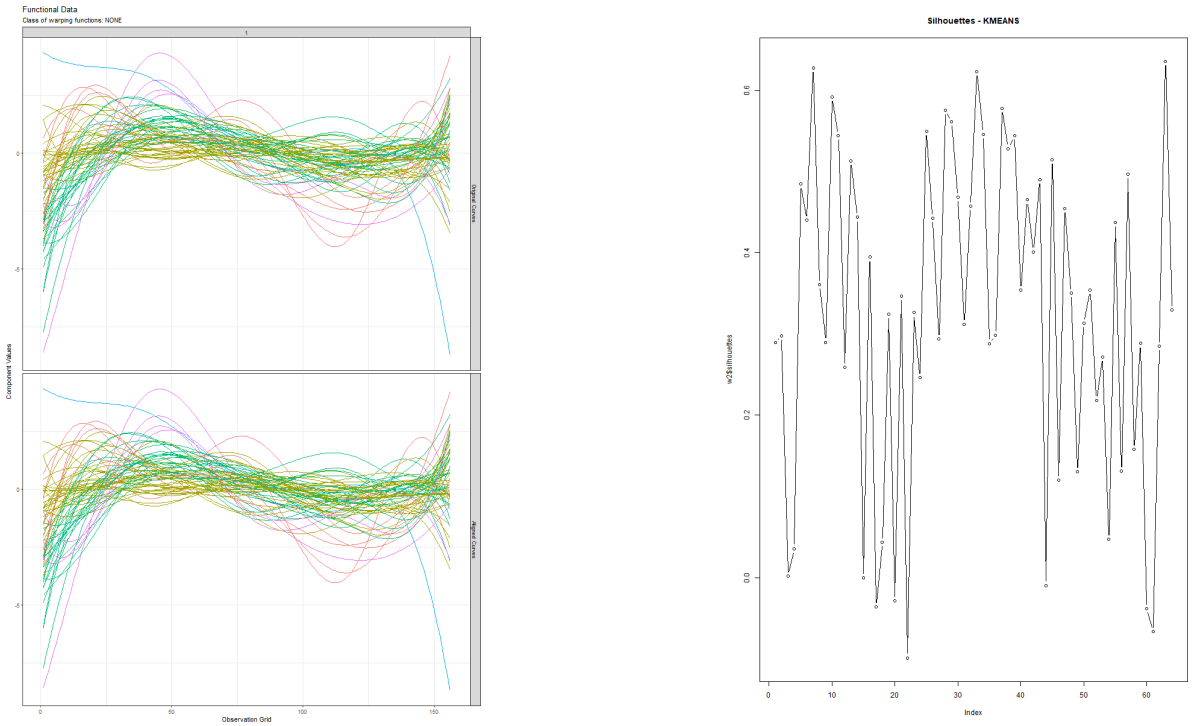


Figure 26: K-means clustering with 5 clusters and mean used for centroid computation

K-means clustering partitions the data into clusters by minimizing the within-cluster sum of squares (WCSS), defined as:

$$WCSS = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2$$

Where μ_k represents the centroid of cluster k.

is the centroid of that cluster. This criterion ensures that each data point is as close as possible to the center of its assigned cluster, promoting high intra-cluster similarity.

In practice, the K-means algorithm operates iteratively by initializing cluster centroids, assigning each point to the nearest centroid, and then updating the centroids based on the new cluster memberships. This process repeats until convergence is achieved, typically when assignments no longer change or the WCSS reaches a local minimum. The resulting clusters tend to form spherical or convex shapes in the feature space.

In our analysis, K-means effectively grouped the functional data into well-separated clusters, reflecting similar temporal or structural patterns among the elements. However, it is important to note that K-means assumes that clusters are of similar size and density, which may not always hold in real-world functional data, potentially affecting the interpretability of the results. The best separated clusters are the dark green and yellow ones, while it can be noted that the light blue cluster is formed only by one function which is the Tesla stock, indicating that this function is an outlier. The graph in general is similar to the one previously obtained with Hierarchical clustering and provides us with a good separation of the functions.

5.3 DBSCAN Clustering Analysis

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering method that identifies clusters based on the density of points. It is particularly effective in detecting outliers. In our analysis, DBSCAN identified Tesla as a significant outlier, indicating that its functional behavior deviated substantially from the other stocks.

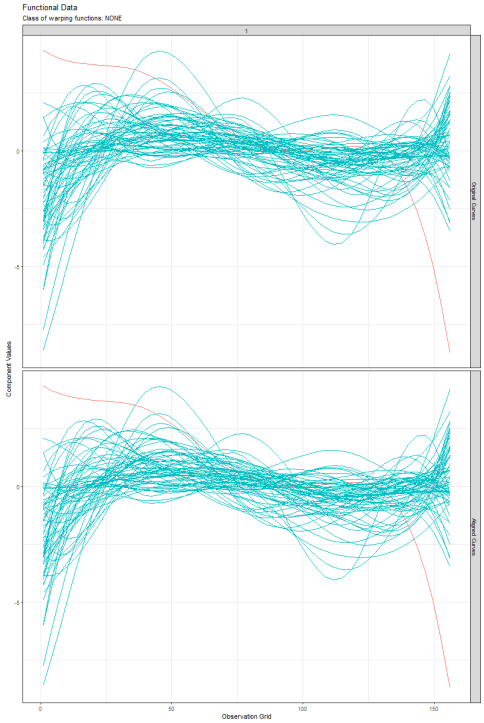


Figure 27: Results from DBSCAN clustering methods. Red function is Tesla

Using this method, different from the previous ones because it is based on density and not on the creation of centroids, we can notice that only two clusters are recognized, one of which is formed only by the Tesla stock, which we can identify as an outlier compared to the general trend of the other functions.

In this case we can say that DBSCAN does not work optimally due to the high density of the functions and the overlap between them, thus making the final result not very useful for our analyses.

5.4 Cluster Interpretations

```
> dt_w[dt_w$FunFEM == 1,]
       StockName FunFEM Hclust Kmeans Dbscan
6      IBM.Close      1      2      2      1
10 RACE.MI.Close      1      2      2      1
16       TM.Close      1      1      2      1
17  KER.PA.Close      1      2      3      1
20   MC.PA.Close      1      2      3      1
21  CFR.SW.Close      1      2      3      1
22  ADS.DE.Close      1      2      3      1
23  NKE.DE.Close      1      2      2      1
25  SAN.PA.Close      1      2      2      1
26 NOVN.SW.Close      1      2      2      1
28     AZN.Close      1      2      2      1
30  MRK.DE.Close      1      1      2      1
32   GSK.L.Close      1      2      2      1
33 NESN.SW.Close      1      2      2      1
34      UL.Close      1      2      2      1
35   BN.PA.Close      1      2      2      1
37     PEP.Close      1      2      2      1
38     MCD.Close      1      2      2      1
39       K.Close      1      2      2      1
40     KHC.Close      1      2      2      1
63     WMT.Close      1      2      2      1
```

Figure 28: Stocks collected in cluster 1 according to the FunFEM method

```
> dt_w[dt_w$FunFEM == 2,]
         StockName FunFEM Hclust Kmeans Dbscan
5       AAPL.Close      2      1      2      1
15      TSLA.Close      2      4      4      0
31      ARGX.Close      2      1      2      1
45 ORSTED.CO.Close      2      1      2      1
56      UBER.Close      2      1      2      1
59      AMZN.Close      2      1      2      1
61       FDX.Close      2      2      3      1
> dt_w[dt_w$FunFEM == 3,]
      StockName FunFEM Hclust Kmeans Dbscan
1    SPOT.Close      3      1      1      1
2    NFLX.Close      3      1      1      1
3    NVDA.Close      3      1      1      1
4    META.Close      3      1      1      1
7    MSFT.Close      3      1      2      1
19 RMS.PA.Close      3      1      2      1
24 PUM.DE.Close      3      2      2      1
29 UCB.BR.Close      3      1      2      1
36 BON.PA.Close      3      2      2      1
57 ZAL.DE.Close      3      1      1      1
60 DHL.DE.Close      3      2      3      1
> dt_w[dt_w$FunFEM == 4,]
          StockName FunFEM Hclust Kmeans Dbscan
8        GOOG.Close      4      1      2      1
43    ENEL.MI.Close      4      1      2      1
58        UPS.Close      4      1      2      1
64 X002352.SZ.Close      4      1      2      1
```

Figure 29: Stocks collected in cluster 2-3-4 according to the FunFEM method

```
> dt_w[dt_w$FunFEM == 5,]
        StockName FunFEM Hclust Kmeans Dbscan
9   VOW3.DE.Close      5      2      3      1
11 STLAM.MI.Close      5      2      3      1
12   RNO.PA.Close      5      3      3      1
13   MBG.DE.Close      5      2      3      1
14   BMW.DE.Close      5      2      3      1
18     CPRI.Close      5      3      5      1
27  BAYN.DE.Close      5      2      2      1
41     SHEL.Close      5      2      3      1
42   ENI.MI.Close      5      2      3      1
44  ENGI.PA.Close      5      2      3      1
46    CHV.F.Close      5      2      3      1
47   REP.MC.Close      5      2      3      1
48   TTE.PA.Close      5      2      3      1
49     TRVG.Close      5      5      5      1
50     BKNG.Close      5      2      3      1
51   RYA.IR.Close      5      2      3      1
52     LYFT.Close      5      5      5      1
53     TCOM.Close      5      3      3      1
54     TRIP.Close      5      5      5      1
55      HLT.Close      5      2      3      1
62    AMKBY.Close      5      2      3      1
```

Figure 30: Stocks collected in cluster 5 according to the FunFEM method

The clustering results reveal meaningful patterns across the functional data, reflecting different behavioral profiles among the grouped stocks.

- **Cluster 1** includes stable, mature companies such as IBM, WMT, and MCD. These firms are characterized by consistent, low-volatility trends across time, likely due to their well-established market positions and diversified operations. Their functional profiles suggest resilience to market fluctuations and a tendency to follow steady, predictable patterns.
- **Cluster 2** groups high-growth, tech-driven firms like AAPL, TSLA, and AMZN. These companies exhibit more dynamic and volatile trajectories, often associated with innovation cycles, rapid expansion, and heightened sensitivity to market sentiment. The functional data indicate strong upward momentum with occasional sharp fluctuations, a typical feature of growth-oriented technology stocks.
- **Cluster 3** consists of digital media and streaming-related companies such as SPOT, NVDA, and META. These stocks share common functional signatures marked by momentum bursts, potentially linked to sector-specific trends such as online content consumption, digital advertising, and AI-related developments.
- **Cluster 4** appears to be a smaller, mixed group containing firms like GOOG and UPS. These companies display intermediate characteristics, neither as stable as Cluster 1 nor as volatile as Cluster 2. This cluster may represent businesses undergoing strategic transitions or operating in diverse markets that balance stability and innovation.
- **Cluster 5** encompasses firms from automotive, energy, and travel sectors, including BMW, ENI, and TRIP. The functional forms of these stocks reveal cyclical behavior and macroeconomic sensitivity, likely influenced by factors such as commodity prices, geopolitical developments, and global mobility trends.

Overall, the clustering framework effectively delineates coherent groups with shared temporal dynamics, offering insightful segmentation across industries. The consistency of these results across different clustering methods reinforces the robustness of the functional patterns captured, while DBSCAN further contributes by flagging potential outliers that deviate from the dominant group structures.

6 Hypothesis Testing

6.1 Pointwise ANOVA Analysis

This plot displays the p-values of the point-wise ANOVA tests performed at each time point (week) between industries. It tests whether the mean returns of different industries differ significantly at each time point.

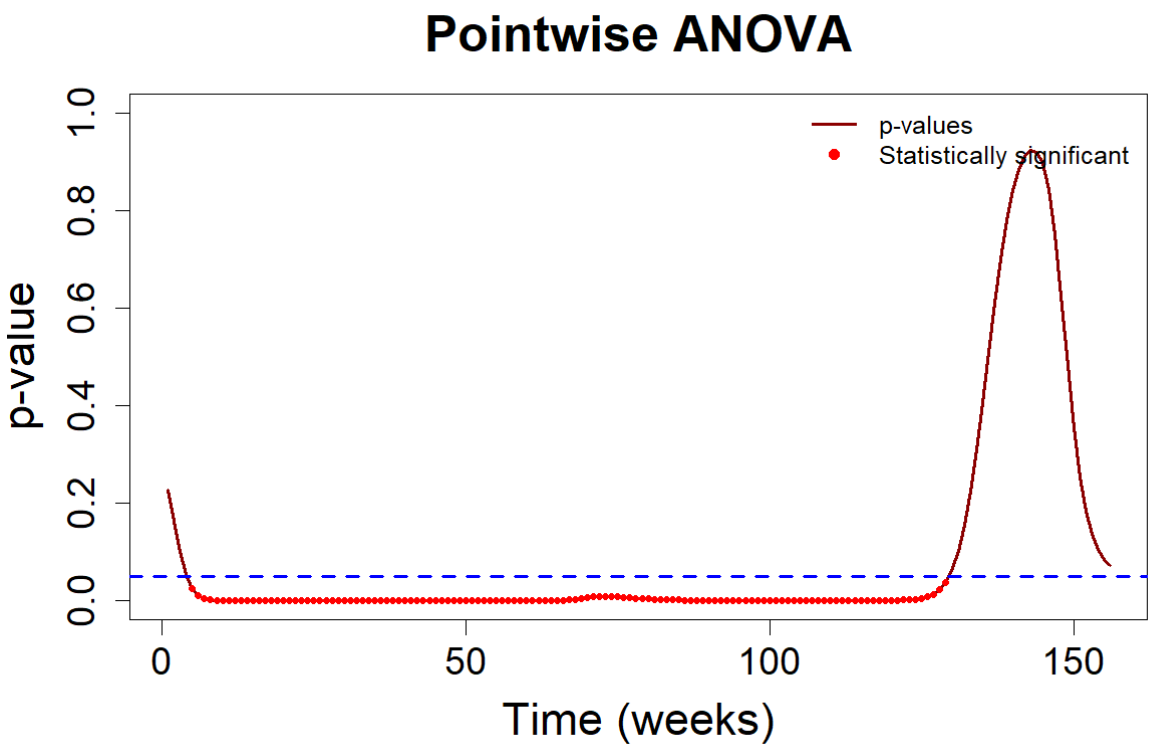


Figure 31: Pointwise ANOVA p-values over 156 weeks. P-values below the dashed blue line (typically 0.05) indicate statistically significant differences in stock returns across industries at those time points. Notable dips in early and late periods suggest strong industry-specific impacts during pandemic onset and recovery phases.

Almost the entire timeline shows p-values below 0.05 (blue dashed line), indicating statistically significant differences in industry stock returns over time. In early weeks (0-20), there is a sharp dip in the p-values and a strong divergence in industry performance at the start of the pandemic. In later weeks (130–150), another sharp rise in p-values returns below significance immediately after. This could correspond to major economic reopening or stabilization events.

6.2 Industry Mean Stock Price Trajectories

This plot presents the smoothed mean stock price trajectories for each industry for 156 weeks. All industries began with a sharp decline (around week 0), typical of the onset of the pandemic. Recovery and Divergence.

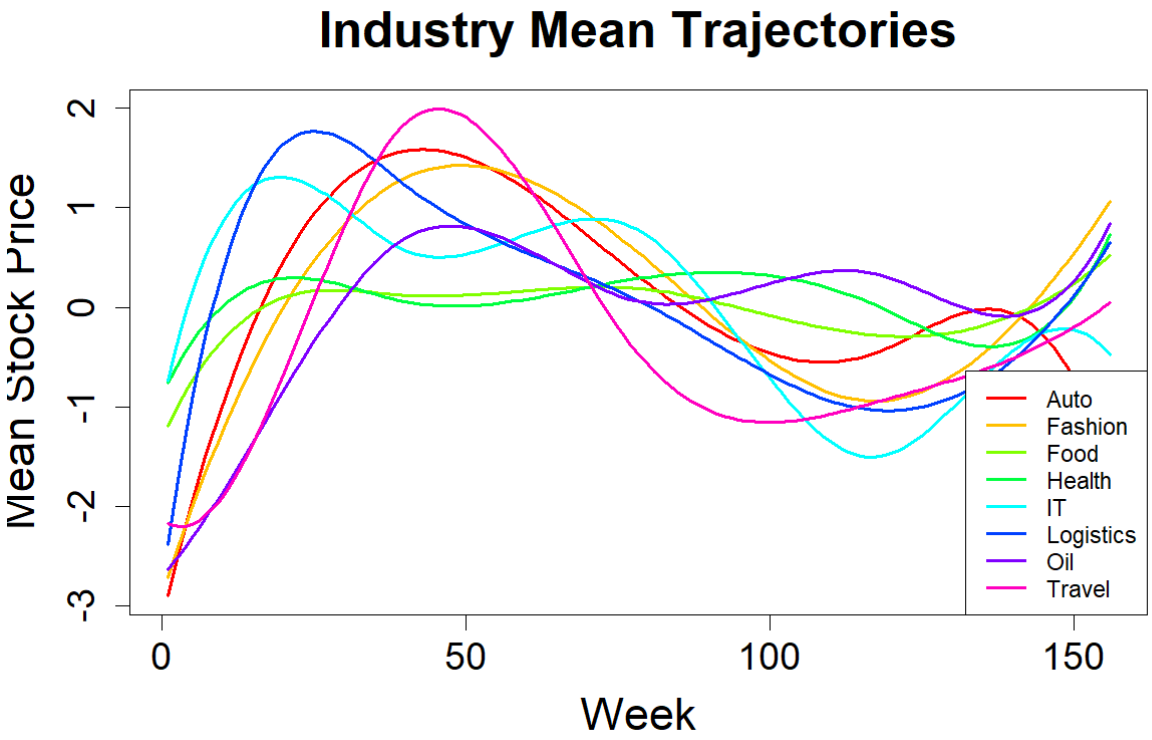


Figure 32: Tech (blue) and Travel (magenta) show strong post-drop rebound, though travel's curve later dips. Food (green) and Health (cyan) show more stable and moderate patterns—likely due to their essential nature. Fashion (orange) and Auto (red) reveal slower or weaker recoveries. Most industries exhibit 2-3 turning points, suggesting phase-wise economic response during the pandemic. There's a clear differentiation in stock performance across industries.Industries did not follow a uniform trajectory through COVID-19.

6.3 Post Hoc Test

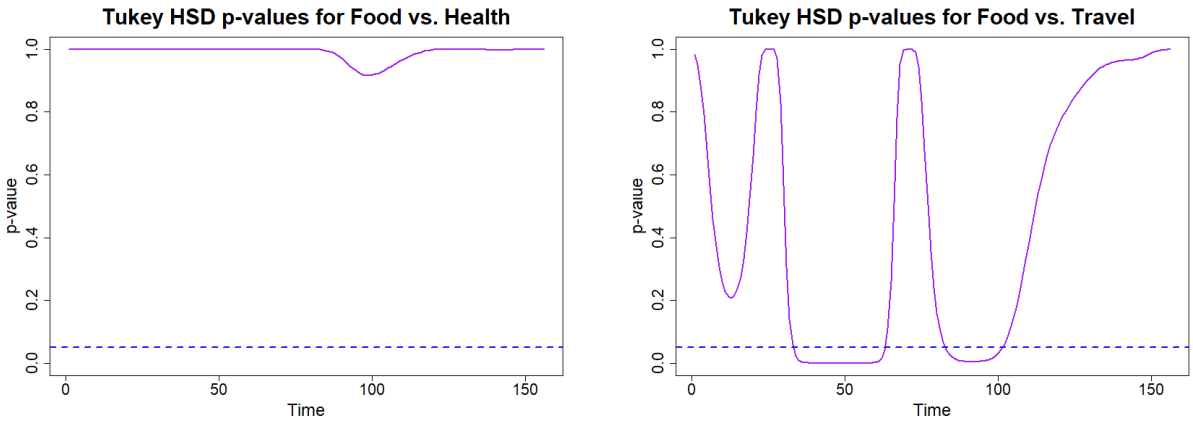


Figure 33: Tukey p-values

- **Food vs Health:** P-values mostly above 0.05 except for a brief dip around week 100. Food and Health stocks have very similar behavior with no strong statistical difference for most of the timeline, showing they were similarly affected or resilient.
- **Food vs. Travel:** P-values dip below 0.05 early and mid-period, with a trend toward significance

again near the end. Food and Travel industries show distinct stock behavior in early and mid-pandemic phases, with some divergence reappearing late in the period.

- **Logistic vs Oil:** Early pandemic phase shows significant difference in returns, possibly due to immediate disruptions impacting logistics and oil differently. The mid-late significant period may reflect oil market volatility or recovery dynamics differing from logistics. The mid-period similarity might suggest a temporary convergence in sector behaviors, perhaps due to shared macroeconomic conditions or policies. Early pandemic phase shows significant difference in returns, possibly due to immediate disruptions impacting logistics and oil differently.
- **Logistic vs Travel:** Logistics and Travel industries exhibit significant differences at multiple points, especially early and mid-pandemic. This aligns with their differing roles: Travel was hit very hard early on, while logistics faced fluctuating demands due to supply chain disruptions. Late phase divergence suggests they may have recovered or responded differently post-pandemic.

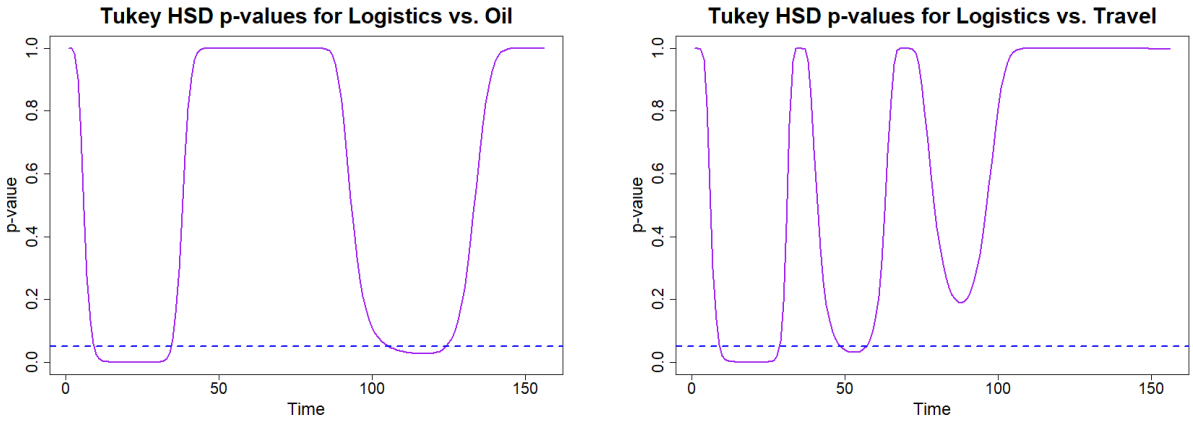


Figure 34: Tukey p-values

- **It vs Travel:** Several significant dips (below 0.05) at weeks 0–20, 50–70, and 90–110. IT and Travel stock returns diverged significantly during early, mid, and late pandemic phases, reflecting the differing economic sensitivities of these sectors.
- **It vs Oil:** Multiple dips below 0.05 across the timeline, especially weeks 20–70 and 90–110. IT and Oil stocks frequently show significantly different returns, highlighting varying sector-specific impacts of the pandemic and market conditions.

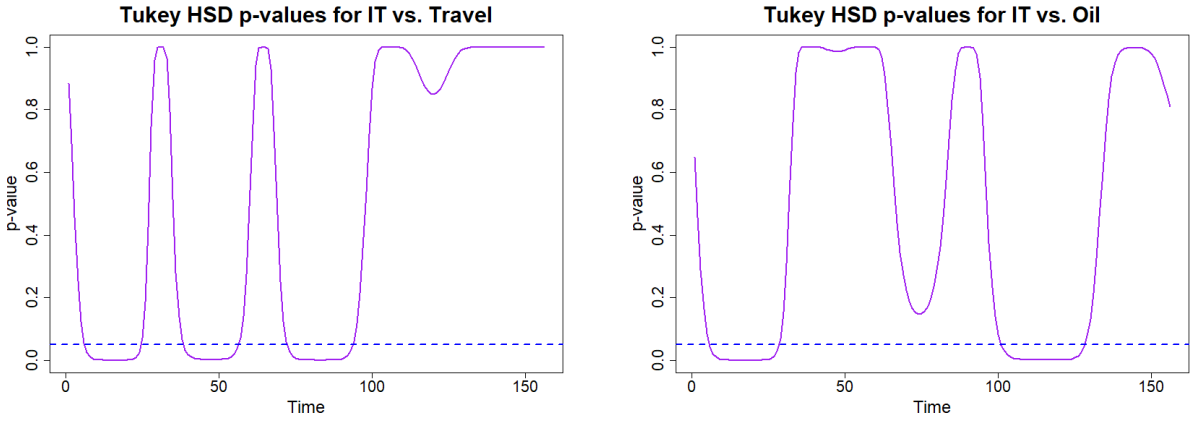


Figure 35: Tukey p-values

- **Auto vs Fashion:** P-values mostly above 0.05, except near the very end where they drop sharply. The difference in returns between Auto and Fashion industries is generally not significant for most

of the time period, but becomes significant close to the end (week 150), indicating a late divergence in their stock behavior.

- **Auto vs Food:** P-values drop below 0.05 between approximately week 20 to 70, then rise and drop again near week 150. Significant differences in stock returns appear mainly in the early to mid-pandemic phase, indicating these industries behaved differently during critical COVID periods. Toward the end, differences become less pronounced but then re-emerge late.

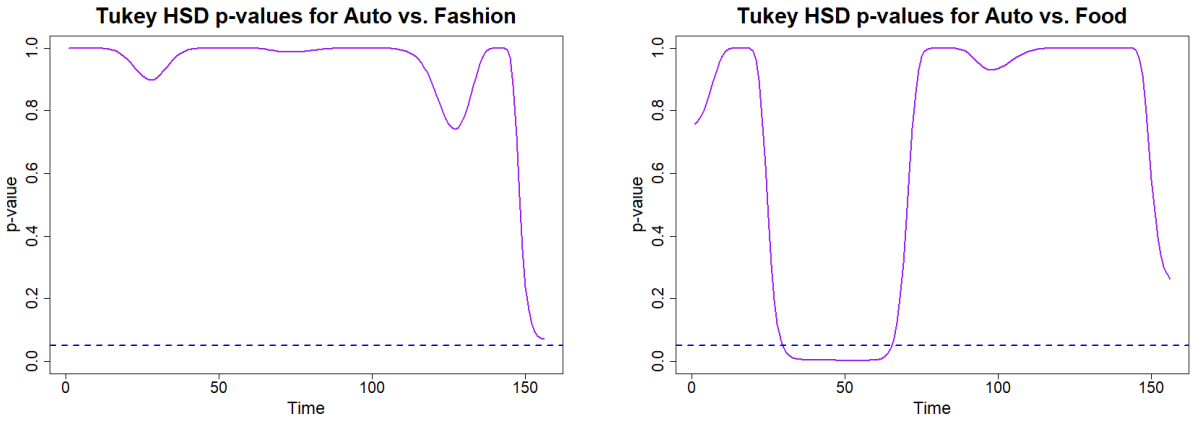


Figure 36: Tukey p-values

- **Fashion vs It:** P-values frequently dip below 0.05, especially early on and mid-period, with a few spikes above. Fashion and IT stocks show significant differences in returns at various times, suggesting their recovery or impact timelines differed through the pandemic.

- **Fashion vs Logistics:** P-values drop below 0.05 twice and early weeks and mid-period (week 20–60), with high values in between and toward the end. Returns of Fashion and Logistics stocks are significantly different at the start and mid-period, indicating distinct market dynamics during these pandemic phases.

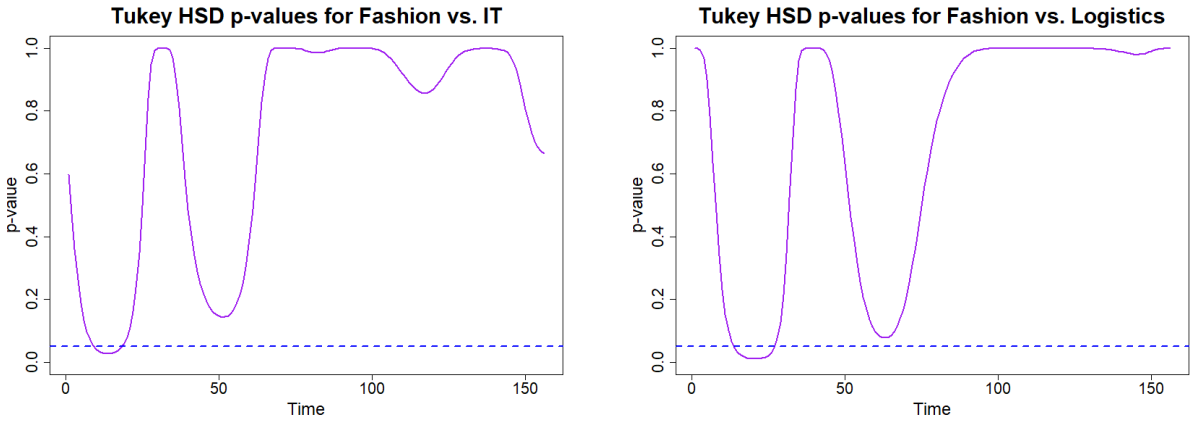


Figure 37: Tukey p-values

- **Food vs It:** Several dips below 0.05 around weeks 0–20, 50–70, and 90–110. Food and IT stocks significantly differ in returns multiple times during the pandemic, indicating asynchronous responses or recoveries.

- **Fashion vs Travel:** P-values drop below 0.05 mainly between week 60 and 100. Fashion and Travel stocks behaved similarly early and late in the period, but showed significant differences mid-pandemic, possibly reflecting sector-specific shocks or recoveries.

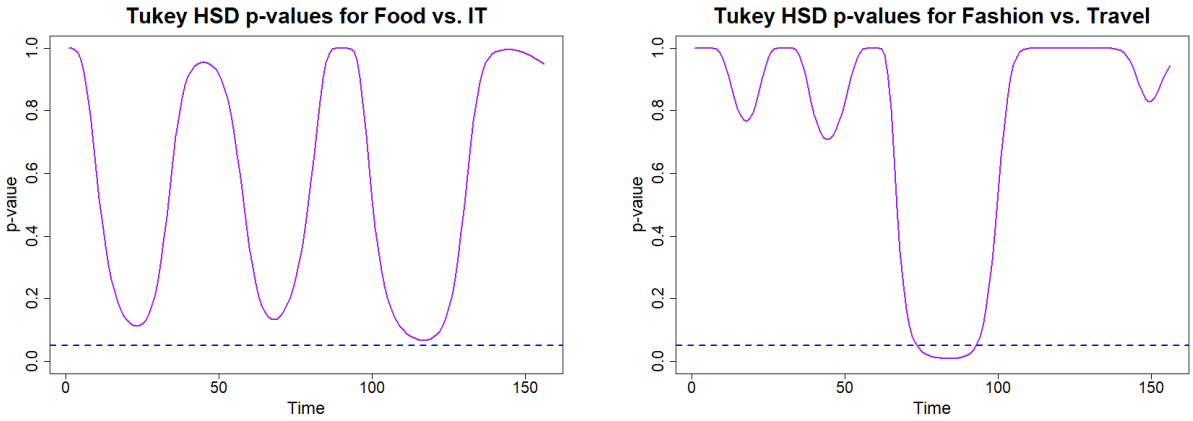


Figure 38: Tukey p-values

7 Functional Autoregressive Process

7.1 FAR Data Format

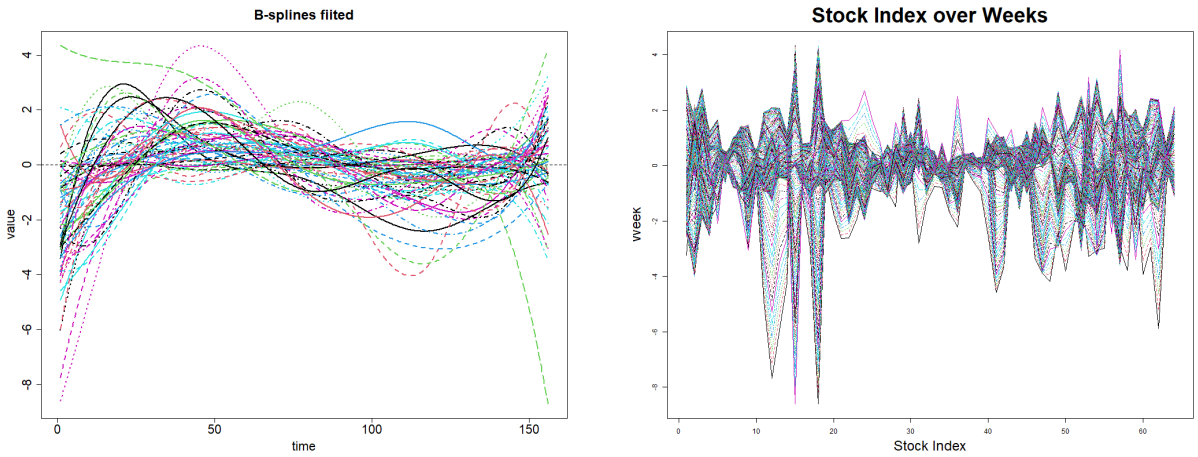


Figure 39: Change of Data Format

For continuing with the Functional Autoregressive process, we had to change the data format, converting it to time series data. For example, our data consists of 64 stocks smoothed over 156 weeks. Now we consider 156 weeks as observations over 64 stocks.

7.2 Parameters FAR(1)

Parameters	
Train set	146 weeks
Test set	10 weeks
Optimal K-NN by GCV	4
Past returns lag	1

Table 1: Parameters

For FAR(1) we split our data into the first 146 weeks for the training set and the last 10 weeks as the test set. By cross-validation, K-NN is chosen 4 and with order 1. Order 1 is chosen without cross-validation.

7.3 FAR(1) Fitted and Residuals

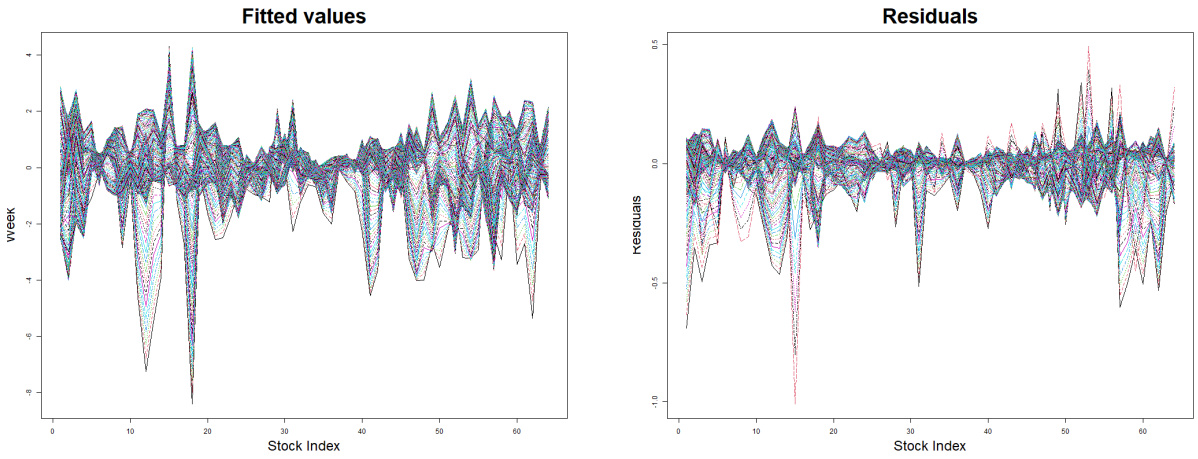


Figure 40: Fitted and Residuals

Here above, we can see the estimated model FAR(1) with fitted and residual values. The main point was to check if we can observe any patterns in residuals. For starting stock index, we notice some pattern, as well as ending stock indexes.

7.4 Forecasting on Test set

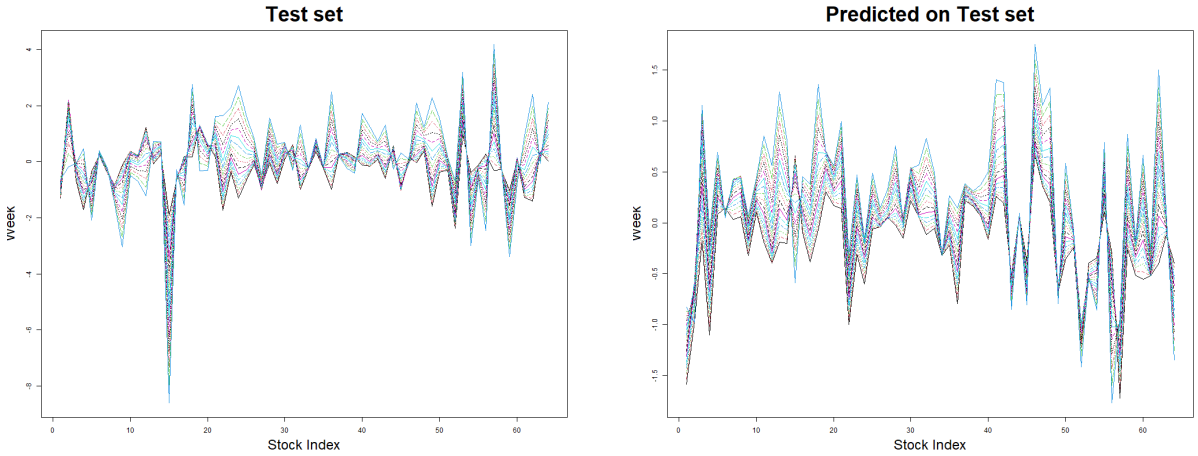


Figure 41: Prediction on Test set

As we had test set on last 10 weeks, we predicted with last 10 weeks. What we can notice is that predictions were less accurate.

7.5 FPCA + FAR(3)

PCA Function	Percentage Contribution
PCA Function 1	57%
PCA Function 2	26%
PCA Function 3	6%
Total	**89%**

Table 2: PCA Function Contributions to FAR

Predicting with Function Principal Components with FAR(3) order 3, Contributions to model and variations with FPCA 1 is about 57% and FPCA 2 is 26%, Total we captured 89%

7.6 Fitted and Residuals

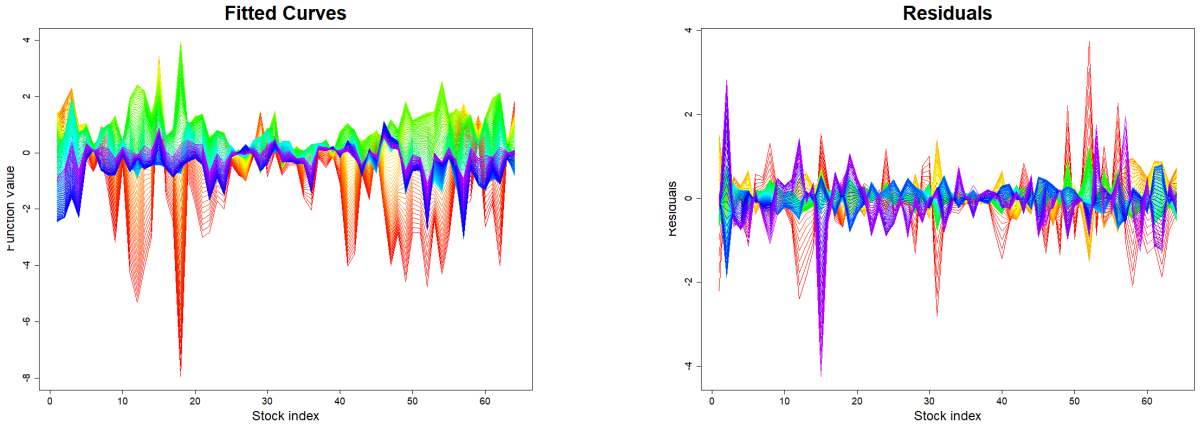


Figure 42: Fitted and Residuals

Here above, we can see the estimated model FPCA + FAR(3) with fitted and residual values. The main point was to check if we can observe any patterns in residuals. Residuals showed fewer patterns than FAR(1)

7.7 Forecasting on Test set

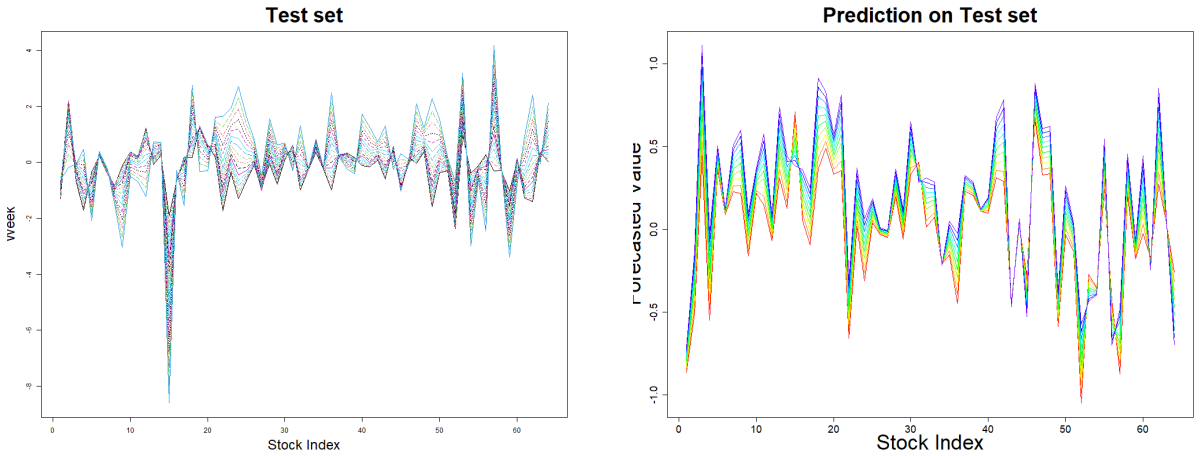


Figure 43: Prediction on Test set

FPCA + FAR(3) showed a better prediction for the stock index between 1 and 20. But later stocks showed less accurate predictions. The table above shows the mean squared error in train and test set.

Model	MSE
FAR(1)	0.05
FPCA + FAR(3)	0.16

Table 3: MSE Results on Train set

Model	MSE
FAR(1)	1.38
FPCA + FAR(3)	1.82

Table 4: MSE Results on Test set

FAR (1) showed an overfit on the graph and MSE on the train set with 0.05, while FPCA + FAR(3) showed MSE 0.16. However, the FAR(1) model showed better accuracy results than FPCA + FAR(3) on test sets 1.38 and 1.82, respectively.

8 General Conclusions

In analyzing the data, we found that there were no evident clusters corresponding to the eight sectors under consideration, suggesting a lack of clear separation based on industry affiliation. The resulting clusters were often noisy and included companies from diverse product sectors, indicating overlapping behavior patterns rather than distinct groupings. Nevertheless, some sectors exhibited stronger internal cohesion than others. Notably, companies within the technology sector consistently grouped together, reflecting their similar responses and trajectories. Likewise, firms in the automotive and energy sectors tended to form more coherent clusters, pointing to sector-specific dynamics that may have influenced their behavior during the period.

To gain further insights, we applied Principal Component Analysis (PCA) and examined the first derivative of the principal components over time. This revealed significant variation, particularly at the onset and the conclusion of the pandemic period. The first constant derivative observed after the start of COVID-19 and again toward the end of the crisis period indicates how the pandemic influenced the price movement of stocks—highlighting these timeframes as critical drivers of change across multiple sectors.

Interestingly, when testing our initial hypotheses, only the Health and Food industries displayed consistent patterns of behavior, reinforcing the idea that these sectors may have responded in a more uniform and predictable manner during the crisis. From a modeling standpoint, both the Functional Autoregressive model of order one (FAR(1)) and the Functional Principal Component Analysis (FPCA) approach demonstrated relatively weak predictive accuracy. However, the FAR(1) model achieved a lower mean squared error, suggesting a relatively better fit for this dataset.

9 Code

```
library(quantmod)
library(tidyquant)
library(tidyverse)
library(dplyr)
library(zoo)
library(httr)


stock_list <- list(it_stocks, automobile_stocks, fashion_stocks, healthcare_stocks,
                   food_stoks, oil_stocks, travel_stocks, logistics_stocks)

# Merge all datasets by 'date'

start_date <- as.Date("2020-01-01")
end_date <- as.Date("2022-12-31")
all_stocks[3,]
# Generate all Fridays within the date range
all_dates <- seq(from = start_date, to = end_date, by = "day")

# Filter to only Fridays
fridays <- all_dates[weekdays(all_dates) == "Friday"]

all_stocks_xts <- xts(all_stocks[,-1], order.by = all_stocks$date)

# Create an empty xts object with Fridays
friday_xts <- xts(, order.by = fridays)

# Merge and fill missing values with the last observation
weekly_stocks <- merge(all_stocks_xts, friday_xts, all = TRUE)

# Fill forward and backward to handle all NAs
weekly_stocks <- na.locf(weekly_stocks, fromLast = FALSE)
                    # Forward fill (Thursday, Wednesday, etc.)
weekly_stocks <- na.locf(weekly_stocks, fromLast = TRUE)
                    # Backward fill for leading NAs

# Keep only the rows aligned to Fridays
weekly_stocks <- weekly_stocks[fridays]

# Verify the data
head(weekly_stocks)
nrow(weekly_stocks)
sum(is.na(weekly_stocks))
weekly_stocks_df <- data.frame(date = index(weekly_stocks), coredata(weekly_stocks))

getwd()
# Save the dataset
write.csv(weekly_stocks_df, "weekly_stock_prices_cleaned.csv", row.names = FALSE)

st <- weekly_stocks_df
# Check for any remaining missing values
sum(is.na(weekly_stocks_df))

# Check the result
head(weekly_stocks_df)
# Check the weekly datas
```

```
# Check the merged dataset
head(all_stocks)
dim(all_stocks)
all_stocks[,2]
colnames(all_stocks)
# your file_path

path <- getwd()
path
setwd(file.path(getwd(), "data_stocks"))

# read CSV
it <- read.csv("it_stocks.csv")
automobile <- read.csv("automobile_stocks.csv")
fashion<- read.csv("fashion_stocks.csv")
healthcare<- read.csv("healthcare_stocks.csv")
food <- read.csv("food_stoks.csv")
oil<- read.csv("oil_stocks.csv")
travel <- read.csv("travel_stocks.csv")
logistics <- read.csv("logistics_stocks.csv")
st[,1]

# merge all stocks
st <- cbind.data.frame(logistics,it,automobile,fashion,healthcare,food,oil,travel)
# save the final file
st[,1]
spotify
write.csv(st, file = "final_data.csv", row.names = FALSE)

library(quantmod)
library(tidyquant)
library(tidyverse)
library(dplyr)
library(zoo)
library(httr)
library(DepthProc)
library(MultiRNG)

# Read the file
path <- getwd()
setwd(file.path(getwd(), "data_stocks"))
st <- read.csv("weekly_stock_prices_cleaned.csv")
head(st)

sum(is.na(st))
#Data preprocessing
st <- st[, -1]
log_returns <- apply(st, 2, function(x) c(diff(log(x))))
st <- 100*log_returns
st <- data.frame(st)
dim(log_returns)


# Depth analysis before smoothing

st <- as.matrix(st)
sum(is.na(st))
```

```
# Euclidean depth for not transformed data

dE <- depthEuclid(st, st)
mdE <- which.max(dE)
dE[mdE] # euclidean depth of the deepest point in the dataset

# a very small depth value (like this one) suggests that the dataset
# is highly spread out or has strong outliers,
meaning even the "deepest" point is not very central.

st[mdE,] # retrieves the most central (deepest) row
            from the dataset based on Euclidean Depth

apply(st,2,median)

plot(st)
points(st[mdE,1], st[mdE,2], pch=23, col="blue", bg="blue", lwd=2)
points(median(st[,1]), median(st[,2]), pch=24, col="red", bg="red", lwd=2)
# red one is median point while the blue one is the deepest point,
affected by outliers
# if the blue (deepest) and red (median) points are close,
the dataset is relatively symmetric and balanced.
# if they are far apart, the dataset might have outliers or
skewed distribution (the Euclidean Depth is influenced by the overall spread of the data)

# Local depth
dL <- depthLocal(st, st, depth_params1 = list(method = "LP"))
dL
mdL <- which.max(dL)
dL[mdL]
st[mdL,]

depthContour(st[,1:2], depth_params = list(method = "Local",
                depth_params1 = list(method = "LP")))

# MBD, Frainman-Muniz
dMBD <- fncDepth(st, method = "MBD")
dFM <- fncDepth(st, method = "FM")
mdMBD <- which.max(dMBD)
mdFM <- which.max(dFM)

dMBD[mdMBD]
st[mdMBD,]

dFM[mdFM]
st[mdFM,]
plot(st)
points(st[mdMBD,5], st[mdMBD,6], pch=23, col="blue", bg="blue", lwd=2)
points(st[mdFM,5], st[mdFM,6], pch=23, col="green", bg="green", lwd=2)
points(median(st[,5]), median(st[,6]), pch=24, col="red", bg="red", lwd=2)

fncDepthMedian(st, method = "MBD")
fncDepthMedian(st, method = "FM")

##################################### B-splines ######################################

# B-splines with penalty
library(fda)
```

```
library(fda.usc)

dim(st)
st <- as.matrix(st)
day <- c(1:156)

nrow(st)
# Create a grid for lambda and number of basis
l <- c(0 ,2^seq(-9, 9, len = 40))
nb <- seq(7, 40, by = 2)
time_points <- 1:156
# Create functional ojects with argumen values
fdata_obj <- fdata(t(st), argvals = time_points)

# Smooth with B-splines
out0 <- optim.basis(fdata_obj, lambda = l, numbasis = nb, type.basis = "bspline")
sum((fdata_obj$data - out0$fdata.est)^2)
basis <- create.bspline.basis(c(1,156),nbasis= out0$numbasis.opt, norder = 4)

out0$fdata.est # the smoothed functional data
out0$numbasis.opt # the optimal number of basis functions

# Calculate SSE
SSE <-sum((fdata_obj - out0$fdata.est )^2)

gcv = rep(0,40)
df = rep(0,40)
sse = rep(0,40)

# Iterate with different lambda for graph
lambda_seq = c(0 ,2^seq(-9, 9, len = 40))
for(i in 1:40){
  lambda=lambda_seq[i]
  tD2fdPar = fdPar(basis,Lfdobj=int2Lfd(2),lambda=lambda)

  smooth = smooth.basis(day, st, tD2fdPar)

  gcv[i] = sum(smooth$gcv)
  df[i] = smooth$df
  sse[i] = smooth$SSE
}

# Plot df, SSE and GCV
par(mfrow = c(3,1))
plot(0:39,df[1:40],type='l',xlab='log lambda',ylab='df',cex.lab=1.5)
plot(0:39,sse[1:40],type='l',xlab='log lambda',ylab='sse',cex.lab=1.5)
plot(0:39,gcv[1:40],type='l',xlab='log lambda',ylab='gcv',cex.lab=1.5)
dev.off()

plot(seq(-9, 9, len = 40), sse[1:40], type = 'l',
     xlab = 'log(lambda)', ylab = 'SSE',
     cex.lab = 1.5, lwd = 2, col = 'blue',
     main = 'SSE vs. log(lambda)', cex.main = 1.8)


plot(seq(-9, 9, len = 40), gcv[1:40], type = 'l',
     xlab = 'log(lambda)', ylab = 'GCV',
     cex.lab = 1.5, lwd = 2, col = 'red',
```

```
     main = 'GCV vs. log(lambda)', cex.main = 1.8)
grid()


# Find optimal lambda
optimal_lambda_index = which.min(gcv)
optimal_lambda = lambda_seq[optimal_lambda_index]
optimal_df = df[optimal_lambda_index]
optimal_sse = sse[optimal_lambda_index]
basis <- create.bspline.basis(c(1,156),nbasis= 7, norder = 4)

tD3fdPar = fdPar(basis,Lfdobj=int2Lfd(2),lambda=optimal_lambda)
smooth <- smooth.basis(day,st,tD3fdPar)
smooth$SSE
plot(smooth$fd)
par()
dev.off()
fitted<- eval.fd(1:156, smooth$fd)
plot(st[,3])
lines(fitted[,3])

plot(out0$fdataobj)
names(out0$fdataobj)
dim(t(out0$fdataobj$data))
dim(t(st))
SSE <-sum((st - t(out0$fdataobj$data))^2)
st[1,]


# EDA and outliers detection for b-spline
plot(smooth$fd)
smooth.fd = smooth$fd
dev.off()
plot(smooth.fd)
D1 <- deriv.fd(smooth.fd, deriv = 1)
D2 <- deriv.fd(smooth.fd, deriv = 2)
plot(D1)
plot(D2)
png("first_derivative.png", width = 1200, height = 800)

plot(D1, , lwd = 2, main = "First Derivative",
     xlab = "Time", ylab = "First Derivative Value",cex = 2,cex.lab = 1.8, # Axis labels
     cex.axis = 1.8, # Axis tick labels
     cex.main = 2,   # Main title
     cex.sub = 2)

legend("topright", legend = "First Derivative", col = "blue", lwd = 2)

dev.off()  # Close the device to save the plot

# 2. Save Second Derivative Plot
png("second_derivative.png", width = 1200, height = 800)

plot(D2, lwd = 2, main = "Second Derivative",
     xlab = "Time", ylab = "Second Derivative Value",cex = 2,cex.lab = 1.8, # Axis labels
     cex.axis = 1.8, # Axis tick labels
     cex.main = 2,   # Main title
     cex.sub = 2)
```

```
dev.off()  # Close the device to save the plot
b_spline_mean = mean.fd(smooth.fd)
b_spline_sd = std.fd(smooth.fd)
plot(smooth.fd)
b_spline_mean = mean.fd(smooth.fd)
lines(b_spline_mean, lwd=3, lty=2, col=2)
b_spline_mean <- mean.fd(smooth.fd)

# Open a graphics device to save the plot (e.g., PNG)
png("functional_data_plot_large_text.png", width = 1200, height = 900)

# Plot the smooth.fd object with larger text sizes
plot(smooth.fd, main = "Functional Data with Mean Curve",
     cex.main = 3,      # Title text size
     cex.lab = 2.5,     # Axis labels size
     cex.axis = 2)      # Axis tick labels size

# Add the mean curve with specified line width, type, and color
lines(b_spline_mean, lwd = 5, lty = 2, col = 2)

# Add a legend with larger text
legend("topright", legend = c("Smooth Data", "Mean Curve"),
       col = c(1, 2), lty = c(1, 2), lwd = c(3, 5),
       cex = 2) # Legend text size

# Close the graphics device
dev.off()

lines(b_spline_mean-b_spline_sd, lwd=3, lty=2, col=6)
lines(b_spline_mean+b_spline_sd, lwd=3, lty=2, col=6)
dev.off()

# The Bivariate Covariance Function v(s; t)

logprecvar.bifd = var.fd(smooth.fd)

print(range(logprecvar.bifd$argvals))

time = seq(1,156, length = 36)
logprecvar_mat = eval.bifd(weektime, weektime,
                           logprecvar.bifd)

persp(weektime, weektime, logprecvar_mat,
      theta=-20, phi=20, r=3, expand = 0.5,
      ticktype='detailed',
      xlab="Week",
      ylab="Week",
      zlab="variance")
png("Variance_perspective.png", width = 1200, height = 1000)

# Create the perspective plot
persp(weektime, weektime, logprecvar_mat,
      theta = -20, phi = 20, r = 3, expand = 0.5,
      ticktype = 'detailed',
      xlab = "Week",
      ylab = "Week",
      zlab = "variance",labcex=2)
```

```
# Close the device to finalize and save the file
dev.off()

contour(weektime, weektime, logprecvar_mat,
        xlab="Day",
        ylab="Day")

png("Coontour.png", width = 1200, height = 1000)
day5time = seq(1,156)
logprec.varmat = eval.bifd(day5time, day5time,
                           logprecvar.bifd)
contour(day5time, day5time, logprec.varmat,
        xlab="Day",
        ylab="Day", lwd=2.2,
        labcex=2)
dev.off()

### boxplot

boxplot(smooth$fd)

# 1. Define the correct domain for evaluation
domain <- smooth$fd$basis$rangeval
grid <- seq(domain[1], domain[2], length.out = 100)

# 2. Evaluate the functional data on the grid
eval_fd <- eval.fd(grid, smooth$fd)

# 3. Create a simple and nicer boxplot
boxplot(as.data.frame(t(eval_fd)),
        main = "Boxplot of Function Values",
        xlab = "Evaluation Points",
        ylab = "Function Values",
        col = "lightblue",         # Light blue color for boxes
        border = "darkblue",        # Dark blue borders
        outpch = 20,                # Outliers as solid points
        outcol = "red",             # Red color for outliers
        cex.main = 1.5,             # Bigger title size
        cex.lab = 1.2,              # Bigger label size
        cex.axis = 0.8)             # Slightly smaller axis text

### Kernel smoothing

out1 <- optim.np(fdata_obj , type.S = S.NW, h = 3:50,
                    par.CV = list(criteria = "GCV")) # Local regression
out2 <- optim.np(fdata_obj, type.S = S.LLR,h = 3:50,
                    par.CV = list(criteria = "GCV")) # Local kernel
out3 <- optim.np(fdata_obj, type.S = S.KNN, h = 3:50, Ker = Ker.norm) # Normal Kernel
out4 <- optim.np(fdata_obj, type.S = S.NW, h = 3:50,
                    Ker = Ker.tri, correl = FALSE) # Triweight Kernel
out5 <- optim.np(fdata_obj, type.S = S.NW, h = 3:50,
                    Ker = Ker.epa, correl = FALSE) # Epanechnikov Kerne
out6 <- optim.np(fdata_obj, type.S = S.NW, h = 3:50,
                    Ker = Ker.unif, correl = FALSE) # Uniform Kernel

SSE_out0 <-sum((fdata_obj - out0$fdata.est )^2)
SSE_out1 <-sum((fdata_obj - out1$fdata.est )^2)
```

```
SSE_out2 <-sum((fdata_obj - out2$fdata.est )^2)
SSE_out3 <-sum((fdata_obj - out3$fdata.est )^2)
SSE_out4 <-sum((fdata_obj - out4$fdata.est )^2)
SSE_out5 <-sum((fdata_obj - out5$fdata.est )^2)
SSE_out6 <-sum((fdata_obj - out6$fdata.est )^2)

# Combine GCV values into a vector
gcv_values <- c(out0$gcv.opt, out1$gcv.opt, out2$gcv.opt, out3$gcv.opt,
                out4$gcv.opt, out5$gcv.opt, out6$gcv.opt)

# Define labels for each method
methods <- c("B-splines", "NW", "LR", "Normal ", "Triweight ", "Epanech.", "Uniform")

# Plot GCV values

png("GCV_comparison_plot.png", width = 1200, height = 800)

par(mar = c(5, 7, 4, 2) + 0.1)  # Adjust margins for better text spacing

# Create the barplot without rotating labels
bp <- barplot(gcv_values, names.arg = methods, col = "skyblue",
              main = "GCV Comparison by smoothing methods",
              ylab = "GCV Value", cex.names = 2, cex.lab = 2, cex.main = 3,
              ylim = c(0, max(gcv_values) * 1.1), border = "white")

# Add horizontal grid lines for clarity
grid(nx = NULL, ny = NULL, col = "gray90", lty = 2)

dev.off()

# Compute Sum of Squared Errors (SSE) for each method
sse_values <- c(SSE_out0, SSE_out1, SSE_out2, SSE_out3, SSE_out4, SSE_out5, SSE_out6)

# Plot SSE values
png("SSE_comparison_plot.png", width = 1200, height = 800)

sse_values_adjusted <- sse_values[-1]  # Adjust SSE values if removing the first method
methods_adjusted <- methods[-1]  # Adjust method names to match
options(scipen = 999)

# Improved plot for SSE values
png("SSE_comparison_plot.png", width = 1200, height = 800)

par(mar = c(5, 7, 4, 2) + 0.1)  # Adjust margins

# Create the barplot without rotating labels
bp_sse <- barplot(sse_values, names.arg = methods, col = "lightcoral",
                  main = "SSE Comparison by smoothing methods",
                  ylab = "Sum of Squared Errors",
                  cex.names = 2, cex.lab = 2, cex.main = 3,
                  ylim = c(0, max(sse_values) * 1.1), border = "white")

# Add horizontal grid lines for better readability
grid(nx = NULL, ny = NULL, col = "gray90", lty = 2)

dev.off()  # Save the plot to file

png("gcv_criteria_plot.png", width = 1200, height = 800)
```

```
# Set plot margins and background for better visuals
par(mar = c(5, 6, 4, 8) + 0.1, bg = "white")  # Increased right margin for space

# Main plot with better aesthetics
plot(out1$h, out1$gcv, type = "l",
     main = "GCV Criteria",
     xlab = "Bandwidth (h) Values",
     ylab = "GCV Criteria",
     col = "forestgreen", lwd = 3,
     cex.lab = 1.8, cex.main = 2, cex.axis = 1.5)

# Add grid lines for better clarity
grid(lty = "dotted", col = "gray70")

# Add additional lines with unique colors and styles
lines(out2$h, out2$gcv, col = "royalblue", lwd = 3, lty = 2)
lines(out3$h, out3$gcv, col = "darkorange", lwd = 3, lty = 3)
lines(out4$h, out4$gcv, col = "purple", lwd = 3, lty = 4)
lines(out5$h, out5$gcv, col = "firebrick", lwd = 3, lty = 5)
lines(out6$h, out6$gcv, col = "goldenrod", lwd = 3, lty = 6)

# Improved legend with larger text and custom position (left side)
legend("topright", inset = c(0.01, 0),
       legend = c("Ker.norm-S.NW", "Ker.norm-S.LLR",
                  "Ker.norm-S.KNN", "Ker.tri-S.NW",
                  "Ker.epa-S.NW", "Ker.unif-S.NW"),
       col = c("forestgreen", "royalblue", "darkorange",
               "purple", "firebrick", "goldenrod"),
       lwd = 3, lty = 1:6,
       box.col = "white", cex = 2)

# Close the graphics device to save the image
dev.off()

plot(SSE_out1)

names(out1)

contour(nb, l, out0$gcv, ylab = "Lambda", xlab = "Number of basis",
        main = "GCV criteria by optim.basis()")

dev.new(width = 150, height = 110, units = "mm")
plot(out1$h, out1$gcv, type = "l", main = "GCV criteria  by optim.np() ",
     xlab = "Bandwidth (h) values",ylab = "GCV criteria", col = 3, lwd = 2)
legend(x = 3, y = 6, legend = c("Ker.norm-S.NW", "Ker.norm-S.LLR",
                                "Ker.norm-S.KNN", "Ker.tri-S.NW",
                                "Ker.epa-S.NW", "Ker.unif-S.NW"),
       box.col = "white", lwd = c(2, 2, 2), col = c(3, 4, 5, 6, 7, 8),cex = 0.75)
lines(out3$h,out3$gcv, col = 5, lwd = 2)
lines(out4$h,out4$gcv, col = 6, lwd = 2)
lines(out5$h,out5$gcv, col = 7, lwd = 2)
lines(out6$h,out6$gcv, col = 8, lwd = 2)

plot(out2$h, out2$gcv, type = "l", main = "GCV criteria  by optim.np() ",
     xlab = "Bandwidth (h) values",ylab = "GCV criteria", col = 3, lwd = 2)

### Plotting the differet smoothing
```

```
lines(st[,1], col = "red")
par(mfrow = c(1,2))
plot(out0$fdata.est[1,],col ="blue", lwd = 3)
points(st[,4], col = "red")

plot(out3$fdata.est[4,],col ="blue", lwd = 3)
points(st[,4], col = "red")
par(opar)
dev.off()

png("B-splines fitted.png", width = 1200, height = 800)
plot(out0$fdata.est[2,], col = "blue", lwd = 3,
     main = "B-splines fitted",
     xlab = "Time", ylab = "Price",
     cex.main = 3, cex.lab = 2, cex.axis = 2)

# Add red points for the second data
points(st[, 2], col = "red", cex = 2)
dev.off()
png("Normal kernel fitted.png", width = 1200, height = 800)
plot(out3$fdata.est[1,], col = "blue", lwd = 3,
     main = "Normal kernel fitted",
     xlab = "Time", ylab = "Price",
     cex.main = 3, cex.lab = 2, cex.axis = 2)

# Add red points for the second data
points(st[, 2], col = "red", cex = 2)
dev.off()

png("Triweight kernel fitted.png", width = 1200, height = 800)
plot(out4$fdata.est[2,], col = "blue", lwd = 3,
     main = "Triweight kernel fitted",
     xlab = "Time", ylab = "Price",
     cex.main = 3, cex.lab = 2, cex.axis = 2)

# Add red points for the second data
points(st[, 2], col = "red", cex = 2)
dev.off()

############################################ PCA ############################################

library(fda)
nharm = 6
pcalist = pca.fd(smooth$fd, nharm, centerfns = TRUE)
plot(pcalist)
plot(pcalist$harmonics)

for (i in 1:6) {
  png(paste0("pca_plot_", i, ".png"), width = 800, height = 600)
  plot(pcalist, harm = i, cex = 2,cex.lab = 2,    # Axis labels
       cex.axis = 1.5, # Axis tick labels
       cex.main = 2,   # Main title
       cex.sub = 1.5)  # 'harm' specifies which component to plot
  dev.off()
}

dev.off()
plotscores(pcalist, loc = 5)
```

```
fd.pca1.list <- list()
fd.pca2.list <- list()
fd.pca3.list <- list()
fd.pca4.list <- list()

for(i in 1:5) {
  fd.pca1.list[[i]] <- mean.fd(smooth$fd) +
    pcalist$scores[i,1]*pcalist$harmonics[1]

  fd.pca2.list[[i]] <- mean.fd(smooth$fd) +
    pcalist$scores[i,1]*pcalist$harmonics[1] +
    pcalist$scores[i,2]*pcalist$harmonics[2]

  fd.pca3.list[[i]]<- mean.fd(smooth$fd) +
    pcalist$scores[i,1]*pcalist$harmonics[1] +
    pcalist$scores[i,2]*pcalist$harmonics[2] +
    pcalist$scores[i,3]*pcalist$harmonics[3]

  fd.pca4.list[[i]]<- mean.fd(smooth$fd) +
    pcalist$scores[i,1]*pcalist$harmonics[1] +
    pcalist$scores[i,2]*pcalist$harmonics[2] +
    pcalist$scores[i,3]*pcalist$harmonics[3] +
    pcalist$scores[i,4]*pcalist$harmonics[4]
}
plot(mean.fd(smooth$fd) + pcalist$scores[1,1]*pcalist$harmonics[1])
plot(mean.fd(smooth$fd))

opar <- par(mfrow=c(2,2), ask = TRUE)
for(i in 1:5) {
  plot(fd.pca1.list[[i]], ylim=c(-1, 1), ylab = "1 PC")
  lines(smooth$fd[i], col = 2)

  plot(fd.pca2.list[[i]], ylim=c(-1, 1), ylab = "2 PC")
  lines(smooth$fd[i], col = 2)

  plot(fd.pca3.list[[i]], ylim=c(-1, 1), ylab = "3 PC")
  lines(smooth$fd[i], col = 2)

  plot(fd.pca4.list[[i]], ylim=c(-1, 1), ylab = "4 PC")
  lines(smooth$fd[i], col = 2)
}
par(opar)
opar <- par(mfrow=c(2,2), ask = TRUE)
plot(fd.pca1.list[[1]])
lines(smooth$fd[1], col = 2)

plot(fd.pca2.list[[1]])
lines(smooth$fd[1])

varmx <- varmx.pca.fd(pcalist)
plot(varmx)
par(mfrow= c(2,3))
for (i in 1:6) {
  png(paste0("varimx_plot_", i, ".png"), width = 1000, height = 800)
  plot(varmx, harm = i, cex = 2,cex.lab = 2,    # Axis labels
       cex.axis = 1.5, # Axis tick labels
       cex.main = 2,   # Main title
```

```
       cex.sub = 1.5)
  dev.off()
}

plot(varmx$harmonics)

png("harmonics_plot.png")  # Save as PNG
plot(varmx$harmonics)
dev.off()
plotscores(varmx, loc = 5)

fd.vrm1.list <- list()
fd.vrm2.list <- list()
fd.vrm3.list <- list()
fd.vrm4.list <- list()

for(i in 1:5) {
  fd.vrm1.list[[i]] <- mean.fd(smooth$fd) +
    varmx$scores[i,1]*varmx$harmonics[1]

  fd.vrm2.list[[i]] <- mean.fd(smooth$fd) +
    varmx$scores[i,1]*varmx$harmonics[1] +
    varmx$scores[i,2]*varmx$harmonics[2]

  fd.vrm3.list[[i]]<- mean.fd(smooth$fd) +
    varmx$scores[i,1]*varmx$harmonics[1] +
    varmx$scores[i,2]*varmx$harmonics[2] +
    varmx$scores[i,3]*varmx$harmonics[3]
}

opar <- par(mfrow=c(2,2), ask = TRUE)
for(i in 1:5) {
  plot(fd.vrm1.list[[i]], ylim=c(-1, 1), ylab = "1 PC")
  lines(smooth$fd[i], col = 2)

  plot(fd.vrm2.list[[i]], ylim=c(-1, 1), ylab = "2 PC")
  lines(smooth$fd[i], col = 2)

  plot(fd.vrm3.list[[i]], ylim=c(-1, 1), ylab = "3 PC")
  lines(smooth$fd[i], col = 2)
}

par(opar)

for (i in 1:5) {
  # Set the output to a PNG file (you can also use pdf(), jpeg(), etc.)
  png(paste0("reconstruction_", i, ".png"), width = 1200, height = 900)

  # Save 2x2 grid of plots
  par(mfrow = c(2, 2))

  # 1 PC Reconstruction
  plot(fd.vrm1.list[[i]], ylim = c(-1, 1), ylab = "1 PC")
  lines(smooth$fd[i], col = 2)

  # 2 PC Reconstruction
  plot(fd.vrm2.list[[i]], ylim = c(-1, 1), ylab = "2 PC")
  lines(smooth$fd[i], col = 2)
```

```
  # 3 PC Reconstruction
  plot(fd.vrm3.list[[i]], ylim = c(-1, 1), ylab = "3 PC")
  lines(smooth$fd[i], col = 2)

  # Close the current image file
  dev.off()
}

lines(b_spline_mean-2*b_spline_sd, lwd=4, lty=2, col=8)
lines(b_spline_mean+2*b_spline_sd, lwd=4, lty=2, col=8)

################################### Functional Clustering ###################################

library(funFEM)
library(fdacluster)
library(funData)
set.seed(123)

# --- FunFEM Clustering ---

plot(smooth$fd)
res_funfem <- funFEM(smooth$fd,K = 2:10,model = 'AkjBk',
                     init = 'kmeans', lambda = 0, disp = TRUE)
res_funfem$allCriterions

# Save FunFEM cluster memberships
funfem_clusters <- res_funfem$cls

# Plot BIC to choose K
plot(res_funfem$allCriterions$K, res_funfem$allCriterions$bic, type = 'b',
     xlab = 'Number of Clusters K', ylab = 'BIC', main = 'BIC - FunFEM')

# Discriminative Space Plot
plot(t(smooth$fd$coefs) %*% res_funfem$U, col = funfem_clusters,
        cex = 4, pch = 19, main = "Discriminative Space - FunFEM")
text(t(smooth$fd$coefs) %*% res_funfem$U,
     labels = colnames(st),
     pos = 3,                # position of text relative to point (above)
     cex = 1.2)

# Plot the smoothed functions colored by clusters
plot(smooth$fd, col = funfem_clusters, lwd = 2, lty = 1)
plot(smooth$fd[1:20], col = funfem_clusters, lwd = 2, lty = 1)

# Plot cluster mean functions
fdmeans_v2 <- smooth$fd
fdmeans_v2$coefs <- t(res_funfem$prms$my)
plot(fdmeans_v2, col = 1:max(funfem_clusters), lwd = 2, main = "Cluster Centroids - FunFEM")

# --- Convert to funData for other clustering methods ---

fn_w <- fd2funData(smooth$fd, argvals = day)
autoplot.funData(fn_w)
plot(fn_w)

# --- HCLUST Clustering ---
# "complete", "average", "single", "ward.D2"
```

```
w1 <- fdahclust(fn_w, centroid_type = "mean",
                metric = "l2", linkage_criterion = "complete", n_clusters = 5)

plot(w1)

# HCLUST Silhouette plot
plot(w1$silhouettes, type = "b", main = "Silhouettes - HCLUST")

# --- KMEANS Clustering ---

w2 <- fdakmeans(fn_w, n_clusters = 5, seeding_strategy = "hclust",
                centroid_type = "mean", metric = "l2")

# KMEANS plots
plot(w2, type = "amplitude", main = "Amplitude Plot - KMEANS")
plot(w2, type = "phase", main = "Phase Plot - KMEANS")

plot(w2$silhouettes, type = "b", main = "Silhouettes - KMEANS")


# --- DBSCAN Clustering ---

day_grid <- seq(min(day), max(day), length.out = 50)
fd_eval <- t(eval.fd(day_grid, smooth$fd))

w3 <- fdadbscan(x = day_grid, y = fd_eval)

# DBSCAN plot
plot(w3, main = "DBSCAN Clustering")

# --- Save clustering results ---

# Collect clustering memberships
dt_w <- data.frame(
  FunFEM = funfem_clusters,
  Hclust = w1$memberships,
  Kmeans = w2$memberships,
  Dbscan = w3$memberships
)

write.csv(dt_w, "cluster_results.csv", row.names = FALSE)

# See results
head(dt_w)
dt_w
st
dim(dt_w)
dim(st)

dt_w$StockName <- colnames(st)

dt_w <- dt_w[, c("StockName", "FunFEM", "Hclust", "Kmeans", "Dbscan")]
write.csv(dt_w, "cluster_results_with_names.csv", row.names = FALSE)

dt_w
dt_w[dt_w$FunFEM == 1,]
dt_w[dt_w$FunFEM == 2,]
```

```
dt_w[dt_w$FunFEM == 3,]
dt_w[dt_w$FunFEM == 4,]
dt_w[dt_w$FunFEM == 5,]

dt_w[dt_w$Hclust == 1,]
dt_w[dt_w$Hclust == 2,]
dt_w[dt_w$Hclust == 3,]
dt_w[dt_w$Hclust == 4,]
dt_w[dt_w$Hclust == 5,]

dt_w[dt_w$Kmeans == 1,]
dt_w[dt_w$Kmeans == 2,]
dt_w[dt_w$Kmeans == 3,]
dt_w[dt_w$Kmeans == 4,]
dt_w[dt_w$Kmeans == 5,]
#_________________________fANOVA___________________
basis <- create.bspline.basis(c(1,156),nbasis= out0$numbasis.opt, norder = 4)
tD3fdPar = fdPar(basis,Lfdobj=int2Lfd(2),lambda=out0$lambda.opt)
smooth <- smooth.basis(day,st,tD3fdPar)
mean(smooth$y[,1:8])
mean(smooth$y[9:18])

time_grid <- 1:156

original_names <- c(
  "it_stocks", "automobile_stocks", "fashion_stocks", "healthcare_stocks",
  "food_stoks", "oil_stocks", "travel_stocks", "logistics_stocks"
)

# Simplified names (same order)
short_names <- c(
  "IT", "Auto", "Fashion", "Health",
  "Food", "Oil", "Travel", "Logistics"
)

groups <- factor(rep(short_names, each = 8))
head(groups, 16)
dim(fdata_stocks)

fANOVA.pointwise <- function(data, groups, t.seq = NULL, alpha = 0.05) {
  # Check inputs
  if (length(groups) != ncol(data))
    stop("Length of 'groups' must match number of columns in 'data'.")
  if (!is.factor(groups))
    groups <- factor(groups)}
library(dplyr)

n_time <- nrow(st)
n_groups <- length(levels(groups))
group_levels <- levels(groups)
pvals <- numeric(n_time)
mean_vals <- matrix(NA, nrow = n_time, ncol = length(levels(groups)))
colnames(mean_vals) <- levels(groups)

combs <- combn(group_levels, 2)
perm <- ncol(combs)
pvals <- numeric(n_time)
```

```
Tukey.posthoc <- matrix(NA, nrow = n_time, ncol = perm)
colnames(Tukey.posthoc) <- apply(combs, 2, paste, collapse = " vs. ")
data <- eval.fd(1:156, smooth$fd)

for (i in 1:n_time) {
  # Fit ANOVA for week i
  dt <- data.frame(
    Price = data[i, ],  # Stock prices at week i
    Industry = groups   # Industry labels (IT, Auto, ...)
  )
  av <- aov(Price ~ Industry, data = dt)

  # Extract ANOVA p-value
  pvals[i] <- summary(av)[[1]]$`Pr(>F)`[1]

  # Calculate industry means for week i
  mean_vals[i, ] <- tapply(dt$Price, dt$Industry, mean)

  # Tukey HSD post-hoc test
  tukey_res <- TukeyHSD(av)$Industry[, "p adj"]
  Tukey.posthoc[i, ] <- tukey_res
}
sig_level <- 0.05
alpha <- 0.05
t.seq <- 1:156
# Plot p-values with significance highlights
png("ANOVA_1.png", width = 1200, height = 800, res = 150)
plot(1:156, pvals, type = "l", lwd = 2, col = "darkred",
     main = "Pointwise ANOVA",
     xlab = "Time (weeks)", ylab = "p-value", ylim = c(0, 1),
     cex.lab = 1.8,     # axis label size
     cex.axis = 1.5,    # tick number size
     cex.main = 2)
abline(h = alpha, col = "blue", lty = 2, lwd = 2)

pvals_adj <- p.adjust(pvals, method = "fdr")
# Highlight significant weeks (FDR-adjusted)
sig_weeks <- which(pvals_adj < alpha)
points(t.seq[sig_weeks], pvals[sig_weeks], col = "red", pch = 19, cex = 0.6)
legend("topright", legend = c("p-values", "Statistically significant "),
       col = c("darkred", "red"), lwd = c(2, NA), pch = c(NA, 19), bty = "n")
dev.off()
overall_mean <- rowMeans(data)
col_set <- rainbow(n_groups)
ylim_range <- range(c(mean.p, overall_mean), na.rm = TRUE) * c(0.95, 1.05)

png("industry_mean_plot.png", width = 1200, height = 800, res = 150)
industry_means <- t(sapply(1:nrow(data), function(i) {
  tapply(data[i, ], groups, mean, na.rm = TRUE)
}))

plot(1:156, industry_means[, 1], type = "n",
     ylim = range(industry_means, na.rm = TRUE),
     xlab = "Week", ylab = "Mean Stock Price",
     main = "Industry Mean Trajectories",
     cex.lab = 1.8,     # axis label size
     cex.axis = 1.5,    # tick number size
     cex.main = 2 )
```

```
col_set <- rainbow(ncol(industry_means))  # Color palette
for (i in 1:ncol(industry_means)) {
  lines(1:156, industry_means[, i], col = col_set[i], lwd = 2)
}

# Add legend
legend("bottomright", legend = colnames(industry_means),
       col = col_set, lwd = 2, cex = 0.9)
dev.off()

opar2 <- par(mfrow = c(1, 1), ask = TRUE)
for (i in 1:perm) {
  plot(t.seq, Tukey.posthoc[, i], type = "l", col = "purple", lwd = 2,
       main = paste("Tukey HSD p-values for", colnames(Tukey.posthoc)[i]),
       xlab = "Time", ylab = "p-value", ylim = c(0, 1))
  abline(h = alpha, col = "blue", lty = 2, lwd = 2)
}
par(opar2)
## --- Return --- ##
install.packages("funFEM")
library(funFEM)

res_v = funFEM(smooth$fd,K=5,model="AkjBk",init="kmeans",lambda=0,disp=TRUE)

plot(t(smooth$fd$coefs) %*% res_v$U,col=res_v$cls,pch=19,main="Discriminative space")
text(t(smooth$fd$coefs) %*% res_v$U)

data <- eval.fd(1:156, smooth$fd)

group.label <- factor(res_v$cls)

fANOVA.pointwise(data=data, groups=group.label,
                 t.seq=tsq, alpha=0.05)


group_means <- t(apply(data, 1, function(x) {
  tapply(x, group.label, mean)
}))

colors <- rainbow(length(levels(group.label)))

# Plot all group means
n.groups <- length(levels(group.label))

# Colors for each group
colors <- rainbow(n.groups)  # or choose manually

# Now plot
matplot(1:156, group_means, type = "l", lty = 1, col = colors,
        xlab = "Time", ylab = "Mean Value", main = "Mean Curves per Group")
legend("topright", legend = paste("Group", levels(group.label)),
       col = colors, lty = 1, cex = 0.8)

p.values <- sapply(1:156, function(i) {
  summary(aov(data[i, ] ~ group.label))[[1]][["Pr(>F)"]][1]
})
```

```
plot(1:156, p.values, type = "l", lwd = 2, col = "blue",
     xlab = "Time", ylab = "P-value", main = "Pointwise ANOVA p-values")
abline(h = 0.05, col = "red", lty = 2)

install.packages("fdANOVA")
library(fdANOVA)

plotFANOVA(x = data,
           group.label = group.label,
           int = c(0.025, 0.975),
           means = TRUE)
plotFANOVA(x = data,
           group.label = group.label,
           int = c(0.025, 0.975))

plotFANOVA(x = data, group.label = group.label,
           int = c(0.025, 0.975),separately = TRUE)

#_______________________________FAR(1)_________________________
library(far)
basis <- create.bspline.basis(c(1,156),nbasis= out0$numbasis.opt, norder = 4)
tD3fdPar = fdPar(basis,Lfdobj=int2Lfd(2),lambda=out0$lambda.opt)
smooth <- smooth.basis(day,st,tD3fdPar)

# Transpose


time_grid <- 1:156

eval_matrix <- t(eval.fd(1:156, smooth$fd))
fdata_obj[1, ]$var
fdata_obj <- far::as.fdata(eval_matrix,argvals = 1:64)
str(fdata_obj$var)
dim(fdata_obj$var)
train_weeks <- 1:146
test_weeks <- 147:156
fdata_train <- fdata_obj$var[, train_weeks]
fdata_train <- far::as.fdata(fdata_train,argvals = 1:64)
fdata_test <- fdata_obj$var[, test_weeks]
fdata_test <- far::as.fdata(fdata_test,argvals = 1:64)
far::multplot(fdata_obj$"var", type = "l")
# Fit FAR(1)

model1 <- far(data=fdata_train, y="var", center=TRUE,na.rm=FALSE,kn=4)
print(model1)

print(model1)
pred_far1 <- predict(model1, newdata=fdata_train, na.rm=FALSE)

pred1 <-predict(model1, newdata=fdata_test, na.rm=FALSE)

actual_fd <- select.fdata(fdata_test)
error_fd <- actual_fd[[1]] - pred1[[1]]

model2.cv <- far.cv(data=fdata_train, y="var",ncv=40,
                    cvcrit = "var",
                    center=TRUE, na.rm = FALSE)
print(model2.cv)
```

```
k2 <- model2.cv$minL2[1]

error_mat <- unclass(as.fdata(error_fd))$var  # extract error values matrix (64 x 155)
actual_mat <- unclass(as.fdata(actual_fd[[1]]))$var

mse <- mean(error_mat^2)

# MAPE: average absolute percentage error (x100%) over all stocks and weeks
mape <- mean(abs(error_mat / actual_mat)) * 100

ise_per_week <- colSums(error_mat^2)
mean_ise <- mean(ise_per_week)

matplot(actual_fd, type = "l")

dev.off()
plot(rowMeans(eval.fd(fdobj)))

#___________________fPCA + FAR(3)___________________________

library(ftsa)
library(vars)

stock_grid <- 1:64

# Create functional time series object
train_weeks <- 1:146
test_weeks <- 147:156

fts_train <- fts(x = stock_grid, y = eval_matrix[,train_weeks])

fts_test <- fts(x = stock_grid, y = eval_matrix[,test_weeks])

model.ftsm <- ftsm(fts_train, order=3)
model.ftsm$varprop

fts_test$y
forecast.ftsm <- predict(model.ftsm, h = 10)
names(forecast.ftsm)


train_resid <- model.ftsm$residuals

# Mean squared error on training set
mse_train <- mean(train_resid$y^2, na.rm = TRUE)
print(paste("Train MSE:", round(mse_train, 6)))

y_forecast <- forecast.ftsm$mean$y     # Forecasted: matrix [stock points x 10]
y_actual   <- fts_test$y               # Actual: matrix [stock points x 10]

mse_test <- mean((y_actual - y_forecast)^2, na.rm = TRUE)
print(paste("Test MSE:", round(mse_test, 6)))
```