2 Outlier detection and statistics

The attached csv file (data.csv) contains historical time series for different variables. The variables are GDP(GDP Level), interest rates (LIBOR 1M, %) and Equities: Equities Index 1 (index points), Equities Index 2 (index points) and Equities Index 3 (index points). Please complete the following tasks and provide code implemented, preferably in R. Apply appropriate data transformation if necessary.

2.1 Load the data of the csv file into a data format of your choice.

In this case, following the recommendations of the exercise, the code will be implemented in R (with exception of the algorithm of section 2.2, further details later).

- **Data Gathering.** First, the data is imported into data frame format using the R function read.csv().
- **Data Cleaning.** Secondly, the data is transformed into vertical format, transposing the initial table by changing the names of the rows and columns.

```r
# Code implemented in R
library(data.table)
library(stringr)
library(ggplot2)

# Data Gathering
raw_data <- read.csv('/Users/Documents/data.csv',header = TRUE)

# Data Cleaning
df <- transpose(horizontal_df)          # Dataframe Organized Vertically
rownames(df) <- colnames(horizontal_df) # Set Row Names
colnames(df) <- df[1,]                  # Set Column Names
df <- df[-1, ]                          # Eliminate repeated header row
```

2.2 Detect potential outliers for each time series based on a simple methodology. Do not use any outlier-detection package. You are free with respect to the methodology you choose and based on which criterion you define outliers. However, ensure that the approach is economically and statistically reasonable.

We will consider a outliers as a data point that differ significantly from other observations. There is no rigid mathematical definition of what constitutes an outlier; determining whether

or not an observation is an outlier is ultimately a subjective exercise. Therefore, I will define two types of outliers for our purposes:

- **Outliers caused by heavy-tailed distributions.** This type of outliers are located very far from the population mean in the probability distribution of the process. In our case, we will try to detect them with histograms that will reflect the probability distribution of each time-series.

  In order to represent histograms, the R function hist() has been used for each time series, as a method to detect outliers located very far from each other in the probability distribution. The following code has been used for each variable (for the sake of simplicity, here only GDP is presented):

```r
# Code implemented in R

# GDP | Histogram
GDP <- df[,1]
GDP <- GDP[!is.na(GDP)]
GDP <- as.numeric(GDP)
GDP_hist <- hist(GDP, main="GDP Histogram", col="darkmagenta", freq=TRUE)
text(GDP_hist\$mids,GDP_hist\$counts,labels=GDP_hist\$counts, adj=c(0.5, -0.5))
```

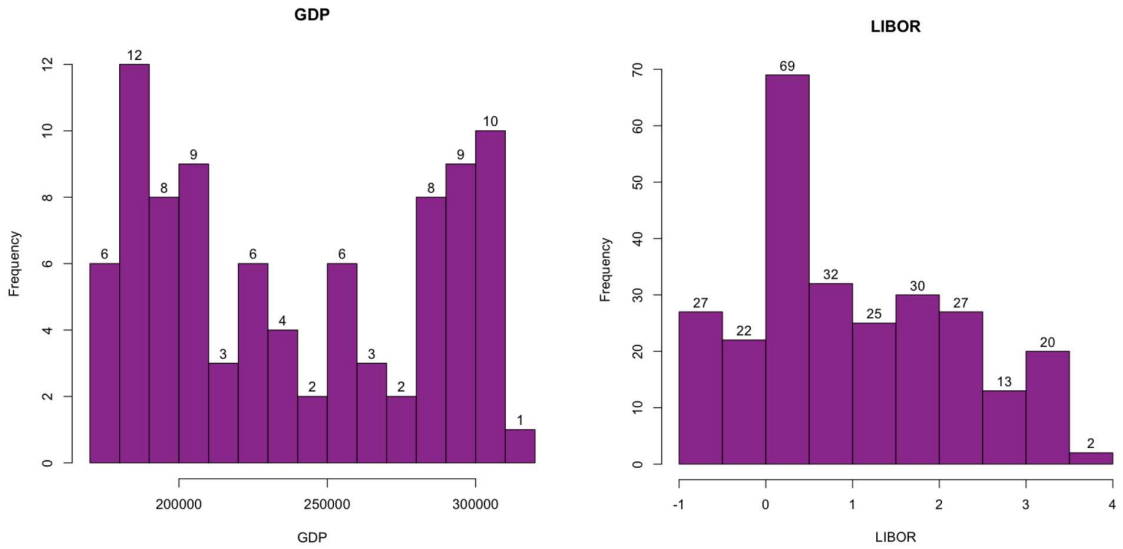

Figure 2: Histograms for the GDP Levels and the LIBOR

As Figures 2 and 3 depict, there are no outliers caused by heavy-tailed distributions that we can observe. While it is true that for the Equities II and III and for the GDP histograms there are alone-points located on the extreme of the frequency distribution, they cannot be considered as outliers because they are are part of the normal time-series, and cannot be considered as completely abnormal points. The same can be

said about Equities III regarding its cluster of isolated measures, which are not very correlated with the initial data points. However, even if we can distinguish two different clusters, it is not adequate to consider them as outliers because the number of data points is large, and they are probably due to a change in the index benchmark rather than related to statistical reasons.

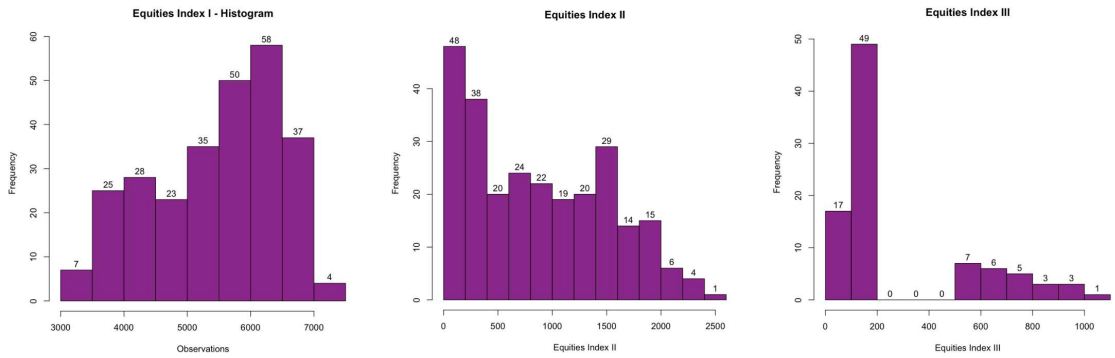

Figure 3: Histograms for Equity Indexes I, II and III

- **Outliers caused by statistical variation in the time-series.** These outliers are not necessarily very far from the population mean, but they are located far away from other data points of the time-series. In order to detect this type of outliers, we will apply to different techniques: i) the visual technique, representing the time-series in a graph; and ii) the analytical technique, applying an algorithm that will detect data points very far away from the rest, based on a confidence interval in the variability of the data of study.

I) Graphical Method

The graphical method consists on plotting the time-series for all the variables and try to visually spot the outliers that doesn't fit the rest of the series. The R-code to plot each time-series is the following:

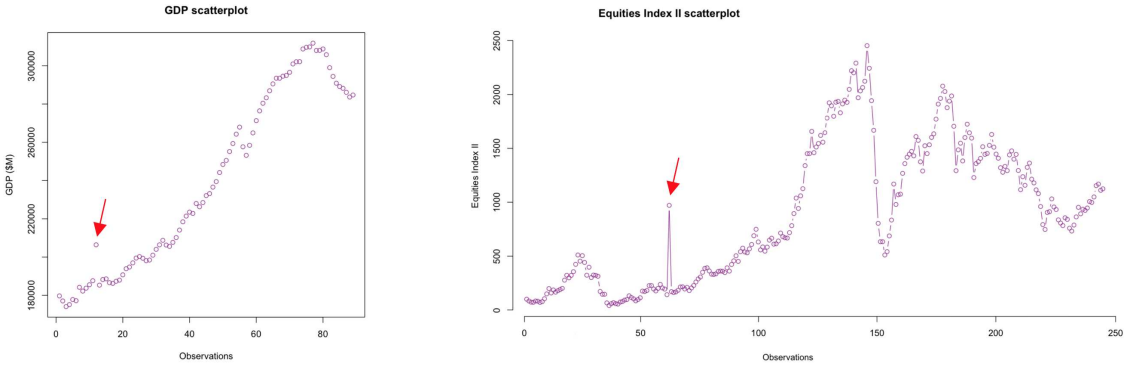

Figure 4: Scatter Line Plot for the GDP and the Equities II time-series

For the sake of simplicity, only the codes an graphics of the time-series that present outliers is shown here; all the codes and graphics are fully detailed in the coding-

notebook attached with this document. Figure 4 depicts the time-series for the GDP and the Equities Index II, and the red arrow points towards the visually-detected outliers. After applying the process for all variables, only GDP and Equities II have presented outliers, so we will now try to apply the analytical procedure to see whether the same results are reached.

II) Analytic Algorithmic Method

The analytic method that I propose to detect outliers is based on a confidence interval in the variability of the data of study. Based on the probability distribution of the consecutive-days price difference for each variable, the algorithm will calculate the outliers as the points that exceed the 95% interval of the daily price variation.

The algorithm only needs a time-series dataset as input, and calculates its outliers following 7 steps:

1. **Time-Series Axis Scaling.**

   The first step of the process is to scale the x-axis of the time-series vector, in order to have the same range in both x and y axis, so measures can be taken directly in the plot. In order to correctly scale the x axis, we equal the total length of x equal to the total range of measures in y and divide it by the number of observations, in order to calculate the step in x (dx) between consecutive data points of the times-series:

   $$dx = \frac{\text{range}(y)}{\text{Nº of data points}} = \frac{\max(y) - \min(y)}{\text{len (dataset)}} \tag{6}$$

2. **Calculation of Δ_{CP} List.**

   The second step is to calculate the difference (delta Δ) between consecutive time-series data points, in order to have information about the variability of the data. For that, we create a python list called Δ_{CP} list (Δ Consecutive Points), where we save all the differences between consecutive data points. Each element in the list, contains:

   $$\Delta_{\text{CP}}[i] = \text{dataset}[i+1] - \text{dataset}[i]. \tag{7}$$

3. **Calculation of $\mu_{\Delta_{\text{CP}}}$ and $\sigma_{\Delta_{\text{CP}}}$.**

   The third step simply consists on computing the mean and standard deviation of the Δ_{CP} list.

4. **Calculation of Chebyshev's k.**

   The fourth step intends to quantify how many standard deviations should we deviate from the mean, in order to observe at least 95% of differences between consecutive data points. A confidence interval of 95% has been chosen as an accurate representation of "normal" Δ_{CP} between consecutive points.

   The main problem we face is that we don't know the distribution of the Δ_{CP} list. Therefore, to calculate the number of standard deviations of the 95% interval of

observations ($k_{95\%}$ we can make use of the Chebyshev's Inequality, which is a fast way to compute such statistic, and it is applicable to every distribution:

$$\mu \pm k\sigma \text{ interval must contain at least } 1 - \frac{1}{k^2} \text{ \% of observations} \tag{8}$$

For our purposes, a 95% of observations will have a Chebyshev's k equal to:

$$0.95 = 1 - \frac{1}{k^2}\ ; \quad k_{0.95} = \sqrt{\frac{1}{1 - 0.95}} = 4.47 \tag{9}$$

5. **Calculation of ε.**

   Epsilon (ε) represents the maximum distance between two points for one to be considered as in the neighborhood of the other. Once all the previous steps have been implemented, we can calculate (ε) as the maximum distance between a point, and a consecutive "normal" point (within the 95% of variation):

   $$\text{Max. distance between normal points: } = \sqrt{dx^2 + (\mu_{\Delta_{\text{CP}}} \pm k_{0.95}\sigma_{\Delta_{\text{CP}}})^2} \tag{10}$$

   For our purposes, as we want to define the maximum distance around a certain point (circle of radius=max. distance), we will calculate ε as two times the maximum distance between consecutive normal points:

   $$\varepsilon = 2\sqrt{dx^2 + (\mu_{\Delta_{\text{CP}}} \pm k_{0.95}\sigma_{\Delta_{\text{CP}}})^2} \tag{11}$$

6. **Selection of M.**

   M represents the number of points that we must find in a neighborhood for a point to be considered as a core point. This includes the point itself. For our purposes, in order to consider a given point as nuclear (part of the current's neighborhood), we want to have a minimum of 3 points within ε distance (the point itself, plus the previous and next points must lie within ε distance). Thus, we can be safe to consider:

   $$M = 3 \tag{12}$$

7. **Clustering Process.**

   This approach is based on the DBSCAN clustering algorithm, which groups datapoints according to their proximity, and detects the outliers that are left alone with no closer data points in their surroundings. The DBSCAN algorithm has only two inputs: ε and M.

   For instance, if we select an epsilon=5 and M=3, the algorithm will iterate through each point of the series and if within a distance less to 5 it finds at least 3 points (itself included), that point will be considered as nuclear point (part of the neighbourhood). The process will be repeated for every element. In the end, all the data points will be separated in different neighbourhoods, and the outliers will be spotted as the points with less than M points in a distance of epsilon=5.

The implementation of this algorithm has been programmed as a function in python:

```
def outlier_detection (dataset, name):
    # I) Configuration of x-axis scale: x=range(y)
    #    between consecutive elements of the dataset
    dx =(max(dataset)-min(dataset))/(len(dataset))

    # II) Calculation of difference between dataset consecutive elements
    dataset_delta=[]
    for i in range(0,len(dataset)-1):
        dataset_delta.append(abs(dataset[i]-dataset[i+1]))

    # III) Calculation of mean and std of the dataset differences
    dataset_delta_mean=np.mean(dataset_delta)
    dataset_delta_std=np.std(dataset_delta)**(1/2)

    # IV) Calculation of Chebyshev k: nº of std that
    #      contain at least 95% of observations
    k=(1/0.05)**(1/2)

    # V) Calculation of the epsilon: min distance
    #     between 2 points to be considered same cluster:
    #         Eps = 2*(dx^2 + 95% CI of daily variability)^(1/2)
    eps=2*(dx**2+(dataset_delta_mean+dataset_delta_std*k)**2)**(1/2)

    # VI) Determination of M: min 3 points within Eps. distance
    M=3

    # VII) Clustering process
    # Creation of the Dataset vector: [y,i*dx]
    dataset_vect=[]
    for i in range(len(dataset)):
        dataset_vect.append([dataset[i],i*dx])
    clustering = DBSCAN (eps=eps, min_samples=M).fit(dataset_vect)

    # VIII) Printing Results
    print (name+" Outliers:")
    print (" - Delta mean:",dataset_delta_mean)
    print (" - Delta std:",dataset_delta_std)
    print (" - k: ",k)
    print (" - Epsilon: ",eps)
    print (" - M: ",M)
    print (clustering.labels_)

    return clustering
```

The application of the outlier detection algorithm yields very interesting results. According to the explained selection of ε and M, only two outliers have been detected: one for the GDP time-series, and one for the Equities II time-series. Interestingly enough, this two outliers are THE SAME as the outliers detected following the visual approach. The output of the algorithm is shown hereafter for the GDP and Equities II and III Indexes, while results for the rest of the time-series are detailed in the Jupyter notebook attached jointly with the present document.

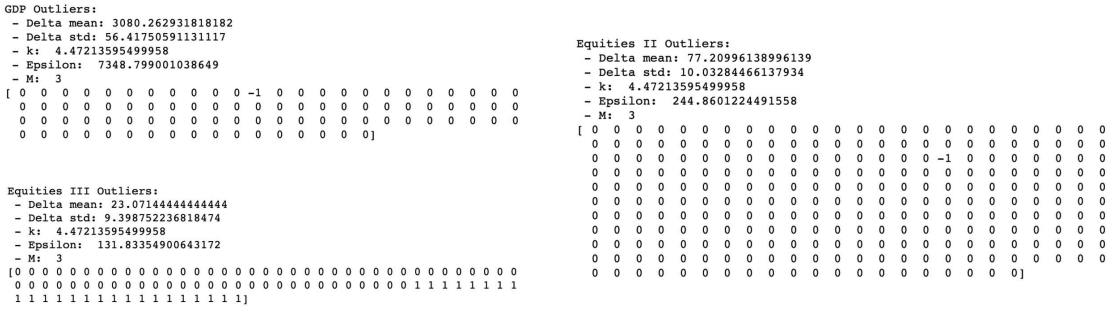

Figure 5: Output of the outlier detection algorithm, labeling each data point belonging to the same cluster with a number between 0 and n, and labeling outliers with a -1

Two additional observations can be made here. The first observation is related to the detection of several clusters within the same time-series. When we find two pieces of data that are very far from each other, but at the same time have more than 3 points within ε, the algorithm labels these two different groups of measurements as two different clusters. This is what happens with the Equities III data-set, which presents two differentiated clusters.

The second observation is that the detection of outliers will be highly related to the variability between consecutive data points of the time-series. That means that time-series with higher variability in one period, will have higher tolerance to consider points as outliers. This is a crucial conclusion to obtain, because it indicates that in order to achieve a correct functioning of this algorithm, the time-series MUST NOT present notable DIFFERENCES IN VOLATILITY throughout different intervals of time. If this is the case, the tolerance towards outliers will increase, and outliers located in the period of lower volatility may remain undetected. Thus the algorithm must only be implemented in time-series with similar volatility of data-points (variation between consecutive data points is constant throughout the whole data-set).

2.3 Provide a measure of correlation for the different variables.

There are several correlation metrics that can be used in order to measure the degree of association between the different variables of study. However, taking into account that for our interests all metrics are measured in cardinal scales, and that we are mainly interested in measuring the linear association between variables, we will make use of the Pearson product-moment correlation coefficient (denoted by ρ for the population correlation and by the letter

"r" for a sample correlation).

The Pearson correlation measures the linear association between two different variables, and can be calculated as the ration between the covariance of the two variables and the product of their standard deviations, obtaining a normalized measure of correlation (ranging between -1 and 1). For our purposes:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} \approx \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{13}$$

In order to apply this formula to the five variables of our interest, we can build a correlation matrix to express the Pearson's correlation of each variable with respect to the others. The procedure used to construct the correlation matrix has been the following:

1. **Selection of the time-series data for each variable**

   We must take into account that each data have different time-series intervals. Even though it would be possible to use interpolation methods for missing values (for instance interpolate the GDP measures monthly, so they can be compared with Equity indexes), the approach that will be used here is more conservative, only making use of available data trying to simplify the computational level of the exercise, and avoid adding extra noise to the existing measures.

   As consequence, we will calculate the correlation coefficient between the GDP and the other variables using only quarterly data, and comparing the equity indexes only between the periods where we have available information.

2. **Calculation of the variance for each variable's time series**

   Once we have selected the time-series data for comparison, we can calculate the variation of each variable: $\sum_{i=1}^{n}(x_i - \bar{x})$.

3. **Construction of the correlation table**

   Applying Equation 13, we can construct the correlation table depicted in Table 1. The table has been implemented and calculated in R studio, by means of the function cor(data_set,"pairwise.complete.obs"). The pairwise.complete command allow us to compute the correlation between each pair of variables using only complete pairs of observations on those variables:

```r
# Code implemented in R
# Change Dataset to numeric values
df_clean_num <- as.data.frame(apply(df_clean, 2, as.numeric))
sapply(df_clean_num, class)

# Calculate Correlation Matrix
cor(df_clean_num, use="pairwise.complete.obs")
```

	GDP	LIBOR	Equities I	Equities II	Equities III
GDP	1	-0.68	0.56	0.74	-0.07
LIBOR	-0.68	1	-0.26	-0.26	-0.89
Equities I	0.56	-0.26	1	0.42	0.46
Equities II	0.74	-0.26	0.42	1	-0.65
Equities III	-0.07	-0.89	0.46	-0.65	1

Table 1: Correlation Table showing Pearson's correlation (ρ) between variables of study

It is possible to see how GDP Levels are the variable that presents higher correlation with the Equities II, with a Pearson's correlation coefficient of 0.74. Equities III present the next higher (negative) correlation with Equities II, with a Pearson's coefficient of -0.65. Finally, Equities I Index only presents a 0.42 correlation with Equities II. Based on these results, in the next sub-section we will construct a model to predict Equity 2.

2.4 Build a simple model that explains Equity 2 by one or more other variables provided. Provide an assessment of the out-of-sample performance for your model for the out-of-sample period 2015/01 to 2017/03.

It is possible to build a model that explains Equity 2 using linear regression. Linear regression aims to predict the variation of a specific dependent variable (Equities II), in terms of the variation of one (or more) independent variable(s).

In order to build a linear regression model, we will need to choose an independent variable (or number of variables) that may predict the dependent variable (Equities II) with as much exactitude as possible, and calculate the regression coefficients ($b_0, b_1, ...b_{j=k}$), under the form:

$$Y_i = b_0 + b_1 X_{1_i} + ... + b_{j=k} X_{j_i=k_i} + \epsilon_i \tag{14}$$

From the correlation matrix in Table 1, it is known that the variable that has a higher correlation with Equities II is the GDP. Thus, the approach to build the regression model will start with a simple linear regression between GDP (independent variable) and Equities II (dependent variable) as benchmark, and will subsequently proceed adding additional independent variables to analyze whether multi-variable regression improves the results of the simple regression. The process to decide which model to choose it is the one described in section 3.2 (variable stepwise selection). For all of these scenarios, data up to 2015 will be used in order to calculate the regression coefficients, and then the models will be examined with the out-of-sample period from 2015 to 2018.

I) Selection of the best regression model

We will make use of the variable stepwise selection process described in section 3.2 to select the best regression model. We will first make use of our predictor as benchmark, and begin with with a simple linear regression between GDP as independent variable, and Equities II as dependent variable.

$$\text{Benchmark Regression:} \qquad \text{Equities II} = b_0 + b_1 \cdot \text{GDP}. \tag{15}$$

```
# Code implemented in R

# Eliminate dataset values from 2015 to 2017 (out-of-sample period)
df_regression <- df_clean_num[-c(240:nrow(df_clean_num)), ]

# Creating variable names
GDP <- df_regression[,1]
LIBOR <- df_regression[,2]
EQ1 <- df_regression[,3]
EQ2 <- df_regression[,4]
EQ3 <- df_regression[,5]

# Benchmark Regression (Eq.II = b0 + b1*GDP)
lm_case1 = lm(EQ2~GDP)
anova(lm_case1)
summary (lm_case1)
```

- **Step 1: Effect of adding the most informational variable.**

  To the benchmark regression, we examine the effects of adding each additional variable (EQ1,EQ3,LIBOR). As Table 2 gathers, the best results for the regression are obtained for the GDP + LIBOR, achieving very low p-values (high statistical significance), and at the same time the highest values for R-squared (0.68), while the GDP alone as regression variable yielded a R-squared of 0.62.

- **Step 2: Effect of removing the most informational variable.**

  In our case, it makes no sense to remove any variable because both (GDP and LIBOR) are informational, and removing either of them would yield a worse model.

- **Step 3: Effect of adding the next most informational variable.**

  Finally, we can repeat the process and examine the effect of adding the next more informational variable. This case is represented with the model (GDP+LIBOR+Eq.I) Table 2. However, it is possible to see how not only the regression doesn't improve the results obtained for GDP and LIBOR, but the p-value of the Eq.1 is completely non-significant, so we are better off not adding the Eq. variable to the regression.

Regression Model	R-squared	F-statistic	P-value
GDP	0.62	123	2e-16
GDP+LIBOR	**0.69**	**81.7**	**2e-16 + 0.000153**
GDP+Eq.I	0.63	64	6.42e-14 + 0.114
GDP+Eq.III	0.24	3.05	0.1114 + 0.0583
GDP+LIBOR+Eq.I	0.69	53.73	1.01e-14 + 0.000635 + 0.991616

Table 2: Regression Models and their R-squared, F-statistic and p-values.

Therefore, we can conclude with our regression model to predict Equities II, by means of the independent variables GDP and LIBOR:

$$\text{Equities II} = -2774 + 0.01457 \cdot \text{GDP} + 209.5 \cdot \text{LIBOR} \tag{16}$$

In order to apply a best practice approach, it is mandatory to also verify the specification of the model, which will tell us whether the variables chosen define the dependent variable in an unbiased and precise way. This includes to examine the residuals of the regression, in order to avoid violating any assumptions of the linear regression (these concepts are further explained in section 3.2.1).

II) Assessment of out-of-sample model performance

To assess the performance of the regression, we can build a model and test it with the out-of-sample model. The code is implemented in R. First, the data is prepared eliminating the NA values. Then, the x and y axis values are created with the Equities II real and predicted data, and then a line plot is built to visually assess the performance of the model. Figure 6 depicts the results.

```
# Code implemented in R

# Preparing Data
df_clean_num_NA <- na.omit(df_clean_num)

# Defining x and y variables
y_real <- df_clean_num_NA[-c(1:22), 4]
y_prediction_1=-2774 + 0.01457*df_clean_num_NA[-c(1:22), 1]
               + 209.5*df_clean_num_NA[-c(1:22), 2]
x_axis=seq(1:9)

# Calculating the mean of the residuals for the out-of-sample regression
Residuals=(y_prediction_1-y_real)**2
mean(Residuals)

```

```
    # Plot
    plot(x_axis, y=y_real, type="l",ylim=range( c(700, 1600)),cex = 1, pch=21, xaxt='n',
                           col="green", xlab="Observations", ylab="Equities Index II",
                           main="Equities Index II Prediction")
    lines(x_axis,y=y_prediction_1, type="l",col="blue")
    axis(1, at=1:9, cex.axis=0.8 , labels=c("03-2015","06-2015","09-2015","12-2015",
                                    "03-2016","06-2016", "09-2016","12-2016","03-2017"))
    legend(6, 1600, legend=c("Equity II - Real Index", "Equity II - Prediction"),
                           col=c("green", "blue"), lty=1:1, cex=0.8)
```

Observing Figure 6, it is possible to observe how the predicted values are not as precise as we could expect in principle. However, we must take into account to factors here.

First, the mean squared error of the residuals during the out-of-sample period is $169,369.5$, while the expected mean squared error of the residuals for the whole regression is $142,739.7$. That means that for this particular example, the mean error is not as far as the mean of the whole regression. We could perform a test statistic to observe how extreme this result is, but in any case it is by no means a very extreme result.

Secondly, the out-of-sample period only represents 2-years of data, from which only GDP quarterly values have been taken into account, so the time interval is quite reduced in comparison to the training data period, to expect a highly correlated model. If the out of sample time-interval would've been longer, the total sum of the residuals should return to the expected value of the regression.

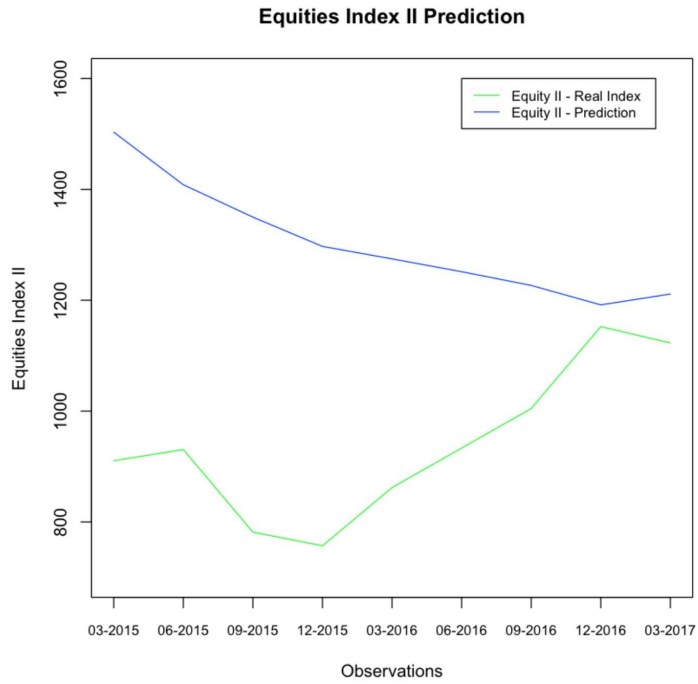

Figure 6: Prediction of Equities II Index (2015-17) with the regression model GDP+LIBOR

Additionally, we have to consider other factors affecting the relationship between variables, such as the presence of serial correlation or heteroskedasticity between the variance of the errors, which could bias the results of the regression. However, due to time constraints these studies will remain outside of the scope of this question.