**1 |** P a g e

**TIME SERIES ANALYSIS OF US GDP FROM THE FEDERAL RESERVE ECONOMIC DATA (1947-2021)**

**INTRODUCTION**

***Problem Description:***

Here in this problem we are interested in,

Choose any non-seasonal non stationary data set and perform the MMSE forecast after examining the model adequancy.

***Objective:***

Here our main objective is to perform the time series analysis U.S. GDP from the Federal Reserve Economic data for the period of 1947/01/01-2021/07/01 fitting a suitable best model for the given dataset and by performing the residual analysis to check the model adequacy i.e. if the model provides adequate information about the data. Further we want to perform MMSE forecast if the model is adequate.

***In sample forecast:***

In-sample forecast is the process of formally evaluating the predictive capabilities of the models developed using observed data to see how effective the algorithms are in reproducing data. It is kind of similar to a training set in a machine learning algorithm.

***Out of Sample forecast:***

An out of sample forecast instead uses all available data in the sample to estimate a models. For example, here estimation would be performed over 1, Jan- 12,March, and the forecast(s) would commence in March, 13th- March, 17th.

***Holt’s Exponential smoothing:***

Exponential smoothing may readily be generalized to deal with time series containing trend and seasonal variation. The version for handling a trend with non-seasonal data is usually called Holt’s (two-parameter) exponential smoothing.

***Auto regressive moving average process:***

Moving average models capture the fact that returns depend not only on current information, but also on signals that have arrived over a previous stretch of time. This could happen if new information is only gradually absorbed or reaches market participants

**2 |** P a g e

at different points in time. As a consequence, any new signal has not only an immediate, but also a delayed, effect.

Autoregressive models assume that there is a linear relationship between current returns and their own history. This type of model can be used when (some) investors base their decisions on recent price movements: in a bull market, profits attract more buyers who will drive up the price even further; and falling prices are seen as a sell signal that will prolong the downward movement.

These two concepts can be combined; not surprisingly, the resulting model is then called autoregressive moving average model, ARMA(p,q)

***Moving average process:***

In time series analysis, the moving-average model, also known as moving-average process, is a common approach for modeling univariate time series. The moving-average model specifies that the output variable depends linearly on the current and various past values of a stochastic term.

*#Setting and getting the current working directory.*

setwd("E:/M.Sc/SEM III/TIME\_SERIES\_ANALYSIS(MST371)/Practical Labs") getwd()

## [1] "E:/M.Sc/SEM III/TIME\_SERIES\_ANALYSIS(MST371)/Practical Labs" ***Data Description:***

Gross domestic product (GDP), the featured measure of U.S. output, is the market value of the goods and services produced by labor and property located in the United States

Below is the quarterly data set that consist of record of U.S. GDP from the Federal Reserve Economic Data for the period of 1947/01/01-2021/07/01.

The dataset consist of 299 records of US GDP and two columns which are explained below, Date(t) - The day on which each value og gdp is recorded.

GDP (Zt)- GDP value that denotes U.S. GDP from the Federal Reserve Economic Data. **ANALYSIS**

*#Loading the 'readxl' package required to load the dataset from excel.* library(readxl)

*#Loading the Quarterly U.S. GDP dataset.*

US\_GDP <- read\_excel("E:/M.Sc/SEM

III/TIME\_SERIES\_ANALYSIS(MST371)/US\_GDP.xlsx")

*#Obtaining the first few records of the dataset.*

head(US\_GDP)

**3 |** P a g e

## # A tibble: 6 x 2

## Date GDP

## <dttm> <dbl>

## 1 1947-01-01 00:00:00 243.

## 2 1947-04-01 00:00:00 246.

## 3 1947-07-01 00:00:00 250.

## 4 1947-10-01 00:00:00 260.

## 5 1948-01-01 00:00:00 266.

## 6 1948-04-01 00:00:00 273.

Thus,the U.S. GDP dataset is loaded in R.

*#Storing the U.S. GDP data in a seperate variable 'gdp'* gdp<-US\_GDP$GDP

gdp

## [1] 243.164 245.968 249.585 259.745 265.742 272.567 279.196

## [8] 280.366 275.034 271.351 272.889 270.627 280.828 290.383

## [15] 308.153 319.945 336.000 344.090 351.385 356.178 359.820

## [22] 361.030 367.701 380.812 387.980 391.749 391.171 385.970

## [29] 385.345 386.121 390.996 399.734 413.073 421.532 430.221

## [36] 437.092 439.746 446.010 451.191 460.463 469.779 472.025

## [43] 479.490 474.864 467.540 471.978 485.841 499.555 510.330

## [50] 522.653 525.034 528.600 542.648 541.080 545.604 540.197

## [57] 545.018 555.545 567.664 580.612 594.013 600.366 609.027

## [64] 612.280 621.672 629.752 644.444 653.938 669.822 678.674

## [71] 692.031 697.319 717.790 730.191 749.323 771.857 795.734

## [78] 804.981 819.638 833.302 844.170 848.983 865.233 881.439

## [85] 909.387 934.344 950.825 968.030 993.337 1009.020 1029.956

## [92] 1038.147 1051.200 1067.375 1086.059 1088.608 1135.156 1156.271

## [99] 1177.675 1190.297 1230.609 1266.369 1290.566 1328.904 1377.490

## [106] 1413.887 1433.838 1476.289 1491.209 1530.056 1560.026 1599.679

## [113] 1616.116 1651.853 1709.820 1761.831 1820.487 1852.332 1886.558

**4 |** P a g e

## [120] 1934.273 1988.648 2055.909 2118.473 2164.270 2202.760 2331.633

## [127] 2395.053 2476.949 2526.610 2591.247 2667.565 2723.883 2789.842

## [134] 2797.352 2856.483 2985.557 3124.206 3162.532 3260.609 3280.818

## [141] 3274.302 3331.972 3366.322 3402.561 3473.413 3578.848 3689.179

## [148] 3794.706 3908.054 4009.601 4084.250 4148.551 4230.168 4294.887

## [155] 4386.773 4444.094 4507.894 4545.340 4607.669 4657.627 4722.156

## [162] 4806.160 4884.555 5007.994 5073.372 5190.036 5282.835 5399.509

## [169] 5511.253 5612.463 5695.365 5747.237 5872.701 5960.028 6015.116

## [176] 6004.733 6035.178 6126.862 6205.937 6264.540 6363.102 6470.763

## [183] 6566.641 6680.803 6729.459 6808.939 6882.098 7013.738 7115.652

## [190] 7246.931 7331.075 7455.288 7522.289 7580.997 7683.125 7772.586

## [197] 7868.468 8032.840 8131.408 8259.771 8362.655 8518.825 8662.823

## [204] 8765.907 8866.480 8969.699 9121.097 9293.991 9411.682 9526.210

## [211] 9686.626 9900.169 10002.179 10247.720 10318.165 10435.744 10470.231

## [218] 10599.000 10598.020 10660.465 10783.500 10887.460 10984.040 11061.433

## [225] 11174.129 11312.766 11566.669 11772.234 11923.447 12112.815 12305.307

## [232] 12527.214 12767.286 12922.656 13142.642 13324.204 13599.160 13753.424

## [239] 13870.188 14039.560 14215.651 14402.082 14564.117 14715.058 14706.538

## [246] 14865.701 14898.999 14608.208 14430.901 14381.236 14448.882 14651.248

## [253] 14764.611 14980.193 15141.605 15309.471 15351.444 15557.535 15647.681

## [260] 15842.267 16068.824 16207.130 16319.540 16420.386 16629.050 16699.551

## [267] 16911.068 17133.114 17144.281 17462.703 17743.227 17852.540 17991.348

## [274] 18193.707 18306.960 18332.079 18425.306 18611.617 18775.459 18968.041

## [281] 19153.912 19322.920 19558.693 19882.965 20143.716 20492.492 20659.102

## [288] 20813.325 21001.591 21289.268 21505.012 21694.458 21481.367

**5 |** P a g e

19477.444

## [295] 21138.574 21477.597 22038.226 22740.959 23173.496

Above, we can see is displayed the U.S. GDP data in Billions of Dollars.

*#Coverting the dataset into time series data and further checking the type of the data.*

gdp=ts(gdp)

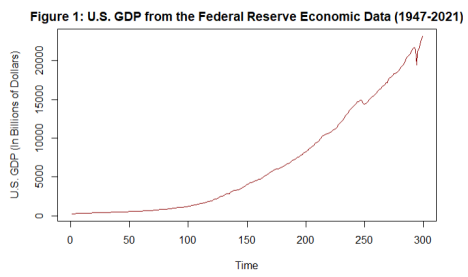
class(gdp)

## [1] "ts"

Now, we proceed to perform the exploratory data analysis of the time series data to understand the behaviour of the data.

*#Obtaining timeseries plot for US GDP time series data.*

ts.plot(gdp, main = "Figure 1: U.S. GDP from the Federal Reserve Economic Data (1947-2021)", xlab = "Time", ylab = "U.S. GDP (In Billions of Dollars)", col = "darkred")



From the above time series plot we observe that there exists a trend component in the dataset since there is observed a increase pattern for a longer period of time. Also we observe that there is some kind of irregularity in the dataset hence we can say that there also exists a error component in the dataset.

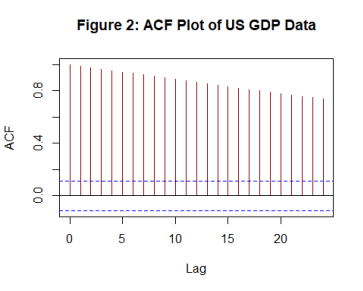
**MODELLING**

Now, we proceed to examine the stationarity of the time series data using acf plot and augmented dickey feller test. Here, since our dataset only consists of trend component in

**6 |** P a g e

the dataset without any seasonality thus we can go for checking stationarity using Augmented Dicky Feller test.

*#Obtaining the ACF plot of the above time series data.*

acf(gdp, main = "Figure 2: ACF Plot of US GDP Data", col = "darkred") 

From the above ACF plot in Figure 4 we observe that all the lag values are significant thus we can conclude that the US GDP time series data is non-stationary.

*#loading the package 'tseries'*

library(tseries)

## Registered S3 method overwritten by 'quantmod':

## method from

## as.zoo.data.frame zoo

*#Checking for the stationarity of the dataset.*

adf.test(gdp)

## Warning in adf.test(gdp): p-value greater than printed p-value

##

## Augmented Dickey-Fuller Test

##

## data: gdp

## Dickey-Fuller = 0.3477, Lag order = 6, p-value = 0.99

## alternative hypothesis: stationary

**7 |** P a g e

Thus, at 5 % level of significance, from the statistical test, augmented dickey fUller test we observe that the p value obtained for the dataset is 0.99 which is greater than 0.05 thus we conclude the the above time series data is non-stationary.

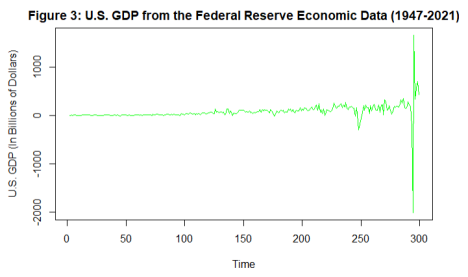
Now, since the data is non-stationary we try to extract stationary component of the dataset before fitting a suitable model.

Since we only have a trend component in our dataset therefore we detrend the time series data using the method of differencing.

*#Detrending the dataset to extract the stationary component from the dataset.* data=diff(gdp)

*#Obtaining the time series plot of detrended data.*

ts.plot(data, main = "Figure 3: U.S. GDP from the Federal Reserve Economic Data (1947-2021)", xlab = "Time", ylab = "U.S. GDP (In Billions of Dollars)", col = "green")

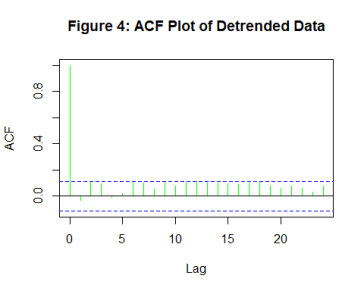


From the time series plot in Figure 5 we observe that the trend component is still there present in the dataset and thus the dataset is non-stationary.

Also, We crosscheck the stationarity of the dataset using the ACF plot and with the help of augmented dickey feller test.

*#Obtaining the ACF plot of the above detrended time series data.* acf(data, main = "Figure 4: ACF Plot of Detrended Data", col = "green")

**8 |** P a g e



From the above ACF plot in Figure 4 we observe that all the lag values are in significant thus we conclude that the US GDP time series data is stationary. We will also perform augmented dickey feller test to check for the stationarity of the dataset.

*#loading the package 'tseries'*

library(tseries)

*#Checking for the stationarity of the dataset.*

adf.test(data)

## Warning in adf.test(data): p-value smaller than printed p-value

##

## Augmented Dickey-Fuller Test

##

## data: data

## Dickey-Fuller = -6.3832, Lag order = 6, p-value = 0.01

## alternative hypothesis: stationary

From the augmented dickey-fuller test we observe that the p value associated with the adf test is less than 0.01 < 0.05 therefore we reject the the null hypothesis and conclude that the detrended dataset is stationary.

Now, since the stationary component of the time series data has been extracted, we will next fit the best suitable time series model to the dataset.

**9 |** P a g e

*#Loading the package 'forecast' required for fitting the suitable ARMA model.* library(forecast)

*#Fitting the suitable ARMA model using stationary data.*

fit=auto.arima(data, seasonal=FALSE)

summary(fit)

## Series: data

## ARIMA(5,1,1)

##

## Coefficients:

## ar1 ar2 ar3 ar4 ar5 ma1 ## -0.2246 -0.0889 -0.0926 -0.2117 -0.2515 -0.9442 ## s.e. 0.0616 0.0643 0.0655 0.0666 0.0816 0.0185 ##

## sigma^2 estimated as 28271: log likelihood=-1942.35

## AIC=3898.71 AICc=3899.1 BIC=3924.56

##

## Training set error measures:

## ME RMSE MAE MPE MAPE MASE ## Training set 18.70826 166.1536 57.30162 78.82615 120.4948 0.9469836 ## ACF1

## Training set -0.008923375

Thus, from the above summary it is observed that the model obtained using auto.arima command is ARMA(5,1) with order of non stationarity as 1 and with AIC = 3898.71.

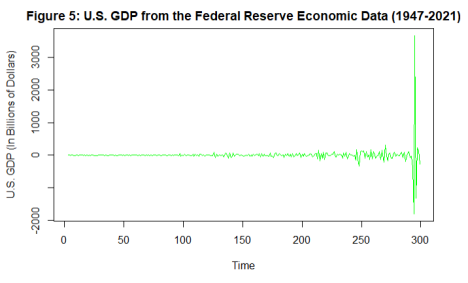
Thus, we further proceed to detrend the data again and fit the best suitable model.

*#Detrending the dataset to extract the stationary component from the dataset.* data1=diff(data)

*#Obtaining the time series plot of detrended data.*

ts.plot(data1, main = "Figure 5: U.S. GDP from the Federal Reserve Economic Data (1947-2021)", xlab = "Time", ylab = "U.S. GDP (In Billions of Dollars)", col = "green")

**10 |** P a g e

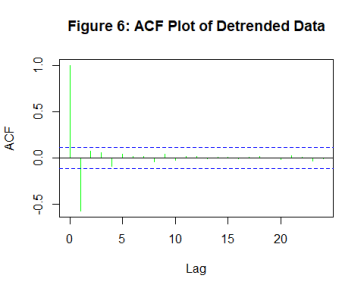


From the time series plot in Figure 5 we observe that the trend component is removed from the dataset and thus the dataset is stationary.

Also, We crosscheck the stationarity of the dataset using the ACF plot and with the help of augmented dickey feller test.

*#Obtaining the ACF plot of the above detrended time series data.* acf(data1, main = "Figure 6: ACF Plot of Detrended Data", col = "green")

**11 |** P a g e



From the above ACF plot in Figure 6 we observe that most of the lag values are in significant thus it seems that the US GDP time series data is stationary. We will also perform augmented dickey feller test to check for the stationarity of the dataset.

*#loading the package 'tseries'*

library(tseries)

*#Checking for the stationarity of the dataset.*

adf.test(data1)

## Warning in adf.test(data1): p-value smaller than printed p-value

##

## Augmented Dickey-Fuller Test

##

## data: data1

## Dickey-Fuller = -8.3077, Lag order = 6, p-value = 0.01

## alternative hypothesis: stationary

From the augmented dickey-fuller test we observe that the p value associated with the adf test less than 0.01 < 0.05 therefore we reject the null hypothesis and conclude that the detrended dataset is stationary.

Thus, we proceed to fit the model again.

**12 |** P a g e

*#Fitting the suitable ARMA model using stationary data.*

fit1=auto.arima(data1, seasonal=FALSE)

summary(fit1)

## Series: data1

## ARIMA(5,0,1) with zero mean

##

## Coefficients:

## ar1 ar2 ar3 ar4 ar5 ma1 ## -0.2246 -0.0889 -0.0926 -0.2117 -0.2515 -0.9442 ## s.e. 0.0616 0.0643 0.0655 0.0666 0.0816 0.0185 ##

## sigma^2 estimated as 28271: log likelihood=-1942.35

## AIC=3898.71 AICc=3899.1 BIC=3924.56

##

## Training set error measures:

## ME RMSE MAE MPE MAPE MASE ## Training set 18.77124 166.4331 57.49454 39.14401 350.9275 0.5499718 ## ACF1

## Training set -0.008964461

Thus, from the above summary it is observed that the model obtained using auto.arima command is ARMA(5,1) and with AIC = 3898.71.

Now, since we got the suitable time series model we proceed to check if a model has adequately captured the information in the data by performing the residual analysis.

**RESIDUAL ANALYSIS**

ASSUMPTIONS

1. Residuals are uncorrelated random variables.

2. Residuals are normally distributed random variables.

*#Obtaining the residuals.*

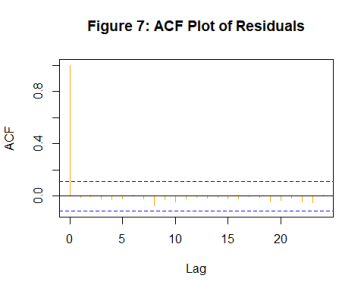
res <- resid(fit1)

Assumption 1. Residuals are uncorrelated random variables.

*#Obtaining the acf plot for residuals.*

acf(res, main = "Figure 7: ACF Plot of Residuals", col = "orange")

**13 |** P a g e



Thus, from the acf plot in Figure 7 we observe that almost all the lags lies inside the threshold line except therefore residuals are uncorrelated random variable, and we will check the same using statistical test.

To confirm the same we perform statistical test.

*#Computing Box-Pierce and Ljung-Box Tests to check if the residuals are uncorrelated.*

Box.test(res, lag=10, fitdf = 6)

##

## Box-Pierce test

##

## data: res

## X-squared = 3.1269, df = 4, p-value = 0.5368

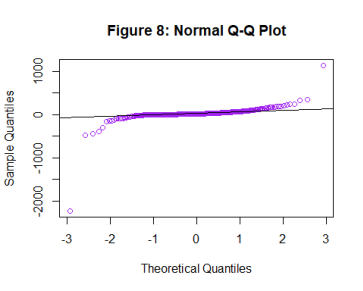
Since from the above test we observe that p value is 0.5368 which is greater than 0.05 thus we accept the null hypothesis and conclude that the residuals are uncorrelated random variables.

Thus the first assumption about the uncorrelated random variables is satisfied. Assumptions 2. Residuals are normally distributed random variables.

*#Checking for normality with the help of a qq plot.*

qqnorm(res, main = "Figure 8: Normal Q-Q Plot", col = "purple") qqline(res)

**14 |** P a g e



From the Q-Q plot in figure 8 we observe that the points in the graph do not form a straight line thus we can say that the residuals are not normally distributed random variables, and we check the same using statistical test.

*#Checking for normality with the help of shapiro wilks test.* shapiro.test(res)

##

## Shapiro-Wilk normality test

##

## data: res

## W = 0.36601, p-value < 2.2e-16

Since from the above test we observe that p value is 2.2e-16 which is less than 0.05 thus we reject the null hypothesis and conclude that the residuals are not normally distributed random variables.

Since the residuals are not normally distributed now we proceed to perform transformation of the original data by taking the log of original data and perform the analysis of the transformed time series data again.

Transforming the data using log transformation.

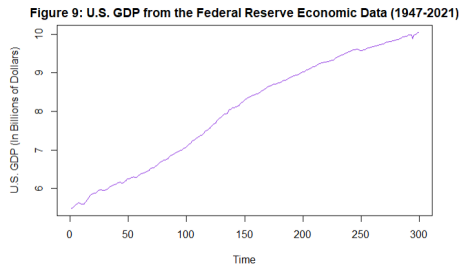
*#Transforming the US GDP time series data using log transformation.* log\_gdp<-log(gdp)

**15 |** P a g e

Now, we proceed to perform the exploratory data analysis of the transformed time series data to understand the behaviour of the data.

*#Obtaining timeseries plot for transformed U.S. GDP data.*

ts.plot(log\_gdp, main = "Figure 9: U.S. GDP from the Federal Reserve Economic Data (1947-2021)", xlab = "Time", ylab = "U.S. GDP (In Billions of Dollars)", col = "blueviolet")

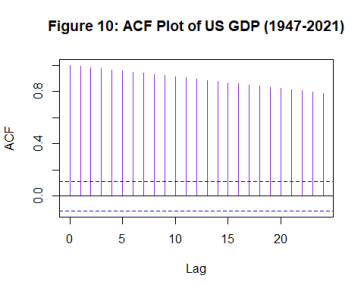


Thus, we observe from the above time series plot (Figure 9) that there exist a trend component in the dataset.

Now, we proceed to examine the stationarity of the time series data using acf plot and augmented dickey fuller test. Here, since our dataset only consists of trend component in the dataset without any seasonality thus we can go for checking stationarity using Augmented Dicky Feller test.

*#Obtaining the ACF plot of the above transformed time series data.* acf(log\_gdp, main = "Figure 10: ACF Plot of US GDP (1947-2021)", col = "blueviolet")

**16 |** P a g e



From the above ACF plot in Figure 10 we observe that all of the lag values are significant thus we can conclude that the transformed US GDP time series data is non-stationary. We check for the stationarity again using augmented dickey fuller test.

*#loading the package 'tseries'*

library(tseries)

*#Checking for the stationarity of the dataset.*

adf.test(log\_gdp)

## Warning in adf.test(log\_gdp): p-value greater than printed p-value

##

## Augmented Dickey-Fuller Test

##

## data: log\_gdp

## Dickey-Fuller = 0.81313, Lag order = 6, p-value = 0.99

## alternative hypothesis: stationary

Thus, at 5 % level of significance, from the statistical test, augmented dickey fuller test we observe that the p value obtained for the dataset is 0.99 which is greater than 0.05 thus we conclude that the above time series data is non-stationary.

Now, since the data is non-stationary we try to extract stationary component of the dataset before fitting a suitable model.

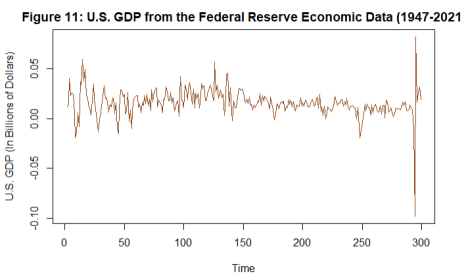
**17 |** P a g e

Since we only have a trend component in our dataset therefore we detrend the time series data using the method of differencing.

*#Detrending the dataset to extract the stationary component from the dataset.* data2=diff(log\_gdp)

*#Obtaining the time series plot of detrended data.*

ts.plot(data2, main = "Figure 11: U.S. GDP from the Federal Reserve Economic Data (1947-2021)", xlab = "Time", ylab = "U.S. GDP (In Billions of Dollars)", col = "chocolate4")

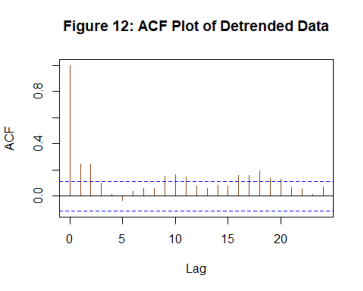


Thus, from the time series plot in Figure 11 we observe that the trend component is not removed and the data seems to be non-stationary.

However, we crosscheck the stationarity of the dataset using the ACF plot and with the help of augmented dickey fuller test.

*#Obtaining the ACF plot of the above detrended time series data.* acf(data2, main = "Figure 12: ACF Plot of Detrended Data", col = "chocolate4")

**18 |** P a g e



From the above ACF plot in Figure 12 we observe that few of the lag values are significant thus it seems that the transformed detrended US gdp time series data is non-stationary.

*#loading the package 'tseries'*

library(tseries)

*#Checking for the stationarity of the dataset.*

adf.test(data2)

## Warning in adf.test(data2): p-value smaller than printed p-value

##

## Augmented Dickey-Fuller Test

##

## data: data2

## Dickey-Fuller = -5.6, Lag order = 6, p-value = 0.01

## alternative hypothesis: stationary

Thus from the from the augmented dickey-fuller test we observe that the p value associated with the adf test is less than 0.01 > 0.05 therefore we reject the null hypothesis and conclude that the detrended dataset is stationary. Therefore, now since it can be seen from the above test that the detrended dataset is proved to be stationary we try fitting the best suitable model to the dataset.

*#Fitting the suitable ARMA model using stationary data.*

fit2=auto.arima(data2, seasonal=FALSE)

summary(fit2)

**19 |** P a g e

## Series: data2

## ARIMA(0,1,3)

##

## Coefficients:

## ma1 ma2 ma3

## -0.8339 0.0556 -0.1718

## s.e. 0.0574 0.0833 0.0587

##

## sigma^2 estimated as 0.0001585: log likelihood=878.33

## AIC=-1748.66 AICc=-1748.53 BIC=-1733.89

##

## Training set error measures:

## ME RMSE MAE MPE MAPE MASE

## Training set -0.0004040629 0.01250528 0.007313972 49.70683 129.9554 0.799446

## ACF1

## Training set 0.008616366

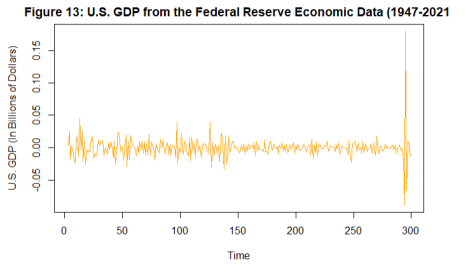
Now, from the above summary we observe that the model we got is MA(3) model with order of non stationarity is 1 which means our data was not stationary and the tests mgave a misleading result thus we need to again difference the data and fit the suitable model.

*#Detrending the dataset to extract the stationary component from the dataset.* data3=diff(data2)

*#Obtaining the time series plot of detrended data.*

ts.plot(data3, main = "Figure 13: U.S. GDP from the Federal Reserve Economic Data (1947-2021)", xlab = "Time", ylab = "U.S. GDP (In Billions of Dollars)", col = "orange")

**20 |** P a g e

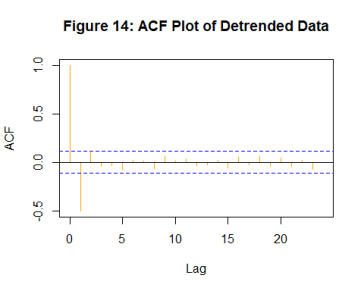


Now, from the time series plot in Figure 13 we observe that the trend component is removed and the data seems to be stationary now.

However, we crosscheck the stationarity of the dataset using the ACF plot and with the help of augmented dickey feller test.

*#Obtaining the ACF plot of the above detrended time series data.* acf(data3, main = "Figure 14: ACF Plot of Detrended Data", col = "orange")

**21 |** P a g e



From the above ACF plot in Figure 14 we observe that lag 1 is significant thus it seems that the US gdp time series data is non-stationary. But since statistical tests are more accurate than the graphical method let’s check the same with the augmented dickey fuller test.

*#loading the package 'tseries'*

library(tseries)

*#Checking for the stationarity of the dataset.*

adf.test(data3)

## Warning in adf.test(data3): p-value smaller than printed p-value

##

## Augmented Dickey-Fuller Test

##

## data: data3

## Dickey-Fuller = -10.487, Lag order = 6, p-value = 0.01

## alternative hypothesis: stationary

Thus from the from the augmented dickey-fuller test we observe that the p value associated with the adf test is less than 0.01 therefore we reject the null hypothesis and conclude that the detrended dataset is stationary.

*#Fitting the suitable ARMA model using stationary data.*

fit3=auto.arima(data3, seasonal=FALSE)

summary(fit3)

**22 |** P a g e

## Series: data3

## ARIMA(0,0,3) with zero mean

##

## Coefficients:

## ma1 ma2 ma3

## -0.8339 0.0556 -0.1718

## s.e. 0.0574 0.0833 0.0587

##

## sigma^2 estimated as 0.0001585: log likelihood=878.33

## AIC=-1748.66 AICc=-1748.53 BIC=-1733.89

##

## Training set error measures:

## ME RMSE MAE MPE MAPE MASE

## Training set -0.0004054622 0.01252631 0.007338559 176.284 266.1063 0.4748061

## ACF1

## Training set 0.00859121

Interpretation: Thus, from the above summary it is observed that the model obtained using auto.arima command is MA(3) with AIC = -1748.66. Thus, auto.arima gave the best suitable model for the data which is a MA(3) model. Also it is observed that rmse is 0.0125 which is a good value for rmse thus we got abest fitted model.

Now that we got the best fitted model we will check the adequacy of the model by performing the residual analysis.

**RESIDUAL ANALYSIS**

ASSUMPTIONS

1. Residuals are uncorrelated random variables.

2. Residuals are normally distributed random variables.

*#Obtaining the residuals.*

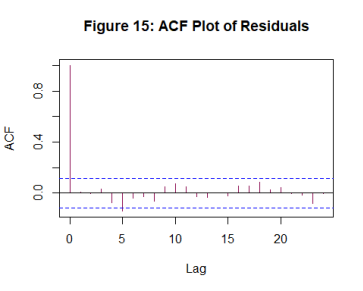
res1 <- resid(fit3)

Assumption 1. Residuals are uncorrelated random variables.

*#Obtaining the acf plot for residuals.*

acf(res1, main = "Figure 15: ACF Plot of Residuals", col = "deeppink4")

**23 |** P a g e



Thus, from the acf plot in Figure 15 we observe that all the lags lies inside the threshold line except for lag 1, therefore residuals are seems to be correlated random variables, let us check the same using statistical test.

To confirm the same we perform statistical test.

*#Computing Box-Pierce and Ljung-Box Tests to check if the residuals are uncorrelated.*

Box.test(res1, lag=10, fitdf = 3)

##

## Box-Pierce test

##

## data: res1

## X-squared = 11.92, df = 7, p-value = 0.1032

Since from the above test we observe that p value is 0.1032 which is greater than 0.05 thus we accept the null hypothesis and conclude that the residuals are uncorrelated random variables.

Assumptions 2. Residuals are normally distributed random variables.

*#Checking for normality with the help of a qq plot.*

qqnorm(res1, main = "Figure 16: Normal Q-Q Plot", col = "deeppink4") qqline(res1)

**24 |** P a g e



From the Q-Q plot in figure 16 we observe that the points in the graph do not form a straight line thus we can say that the residuals are not normally distributed random variables, but we check the same using statistical test.

*#Checking for normality with the help of shapiro wilks test.* shapiro.test(res1)

##

## Shapiro-Wilk normality test

##

## data: res1

## W = 0.77305, p-value < 2.2e-16

Since from the above test we observe that p value is 2.2e-16 which is less than 0.05 thus we reject the null hypothesis and conclude that the residuals are not normally distributed random variables.

**FORECASTING THE FUTURE BEHAVIOUR OF THE DATASET USING HOLT’S EXPONENTIAL TECHNIQUE**

Since, from the above time series plot we observed that our time series data has only trend component therefore we fit the predictive model for the dataset using Holt’s exponential smoothing technique.

Fitting a suitable predictive model for the dataset using Holt’s exponential smoothing technique.

**25 |** P a g e

*#Forecasting using Holt's exponential smoothing.*

gdp\_forecast<-HoltWinters(gdp, beta = TRUE, gamma = FALSE)

gdp\_forecast

## Holt-Winters exponential smoothing with trend and without seasonal component.

##

## Call:

## HoltWinters(x = gdp, beta = TRUE, gamma = FALSE)

##

## Smoothing parameters:

## alpha: 0.5272596

## beta : TRUE

## gamma: FALSE

##

## Coefficients:

## [,1]

## a 23394.5202

## b 723.3217

Thus, we observe from the above model that the alpha value = 0.5272596 which is closer to 1 which indicates that the weightage is not given to all the observations that means forecasting is just on the basis of recent past observation.

Obtaining the in sample and out of sample forecast for next 5 data points.

*#Performing in-sample forecast.*

gdp\_forecast$fitted

## Time Series:

## Start = 3

## End = 299

## Frequency = 1

## xhat level trend

## 3 248.7720 245.9680 2.8040000

## 4 252.4333 249.2007 3.2326621

## 5 263.3763 256.2885 7.0878134

## 6 272.9588 264.6236 8.3351573

## 7 280.8808 272.7522 8.1285822

## 8 287.2327 279.9925 7.2402567

## 9 287.2319 283.6122 3.6197082

## 10 277.9887 280.8004 -2.8117455

## 11 268.1774 274.4889 -6.3115307

## 12 266.8343 270.6616 -3.8272801

## 13 267.0065 268.8341 -1.8275655

## 14 279.7540 274.2940 5.4599570

## 15 296.4224 285.3582 11.0642141

## 16 319.8568 302.6075 17.2492628

## 17 337.1991 319.9033 17.2957910

## 18 353.2304 336.5669 16.6635677

## 19 360.2552 348.4110 11.8441944

**26 |** P a g e

## 20 362.7456 355.5783 7.1672755 ## 21 362.9872 359.2828 3.7044474 ## 22 363.3518 361.3173 2.0345027 ## 23 362.9379 362.1276 0.8103254 ## 24 368.7710 365.4493 3.3217043 ## 25 384.7902 375.1197 9.6704347 ## 26 397.8243 386.4720 11.3523033 ## 27 402.7701 394.6211 8.1490210 ## 28 398.6876 396.6544 2.0332944 ## 29 387.3099 391.9821 -4.6722078 ## 30 380.5657 386.2739 -5.7082404 ## 31 380.7156 383.4948 -2.7791368 ## 32 388.7773 386.1361 2.6412859 ## 33 402.9726 394.5543 8.4182898 ## 34 422.0420 408.2981 13.7438046 ## 35 435.2480 421.7731 13.4749266 ## 36 443.4219 432.5975 10.8243913 ## 37 447.5713 440.0844 7.4869127 ## 38 446.8063 443.4453 3.3609521 ## 39 449.3275 446.3864 2.9411037 ## 40 454.2337 450.3101 3.9236313 ## 41 464.7262 457.5182 7.2080905 ## 42 477.2626 467.3904 9.8722026 ## 43 481.6117 474.5010 7.1106474 ## 44 486.4850 480.4930 5.9919845 ## 45 480.2224 480.3577 -0.1352869 ## 46 466.7133 473.5355 -6.8222137 ## 47 465.4428 469.4892 -4.0463383 ## 48 482.9068 476.1980 6.7088008 ## 49 507.1714 491.6847 15.4867480 ## 50 525.9889 508.8368 17.1521239 ## 51 539.6233 524.2300 15.3932124 ## 52 539.6318 531.9309 7.7008899 ## 53 535.6994 533.8152 1.8842579 ## 54 544.9111 539.3631 5.5479508 ## 55 546.4191 542.8911 3.5279731 ## 56 549.0875 545.9893 3.0982127 ## 57 542.8105 544.3999 -1.5894074 ## 58 543.5489 543.9744 -0.4254860 ## 59 555.7735 549.8740 5.8995502 ## 60 574.2118 562.0429 12.1689149 ## 61 593.1298 577.5864 15.5434770 ## 62 609.6046 593.5955 16.0091272 ## 63 615.8714 604.7335 11.1379731 ## 64 619.7918 612.2626 7.5291739 ## 65 619.3996 615.8311 3.5684950 ## 66 625.3644 620.5978 4.7666210 ## 67 634.7578 627.6778 7.0800345 ## 68 652.0521 639.8650 12.1871604 ## 69 666.2280 653.0465 13.1815104

**27 |** P a g e

## 70 683.1995 668.1230 15.0764934 ## 71 693.5038 680.8134 12.6904045 ## 72 704.6411 692.7272 11.9138728 ## 73 708.8337 700.7805 8.0532198 ## 74 726.3315 713.5560 12.7755270 ## 75 743.1769 728.3665 14.8104795 ## 76 764.4686 746.4175 18.0510469 ## 77 790.3109 768.3642 21.9466741 ## 78 817.9763 793.1703 24.8060773 ## 79 829.0786 811.1244 17.9541598 ## 80 837.0775 824.1009 12.9765224 ## 81 846.0727 835.0868 10.9858713 ## 82 855.0521 845.0695 9.9826613 ## 83 858.6348 851.8521 6.7826499 ## 84 872.3754 862.1138 10.2616265 ## 85 892.1948 877.1543 15.0405073 ## 86 925.3648 901.2595 24.1052768 ## 87 958.9388 930.0992 28.8396388 ## 88 979.2223 954.6607 24.5615519 ## 89 991.9814 973.3210 18.6603148 ## 90 1012.0712 992.6961 19.3750902 ## 91 1028.2287 1010.4624 17.7663034 ## 92 1047.8165 1029.1395 18.6770179 ## 93 1056.2968 1042.7182 13.5786957 ## 94 1064.5008 1053.6095 10.8913348 ## 95 1078.4230 1066.0163 12.4067739 ## 96 1098.8821 1082.4492 16.4329110 ## 97 1104.4808 1093.4650 11.0158032 ## 98 1147.8442 1120.6546 27.1896095 ## 99 1183.9200 1152.2873 31.6327249 ## 100 1208.9673 1180.6273 28.3399716 ## 101 1217.6191 1199.1232 18.4959026 ## 102 1249.8131 1224.4681 25.3449604 ## 103 1292.6166 1258.5424 34.0742164 ## 104 1324.5284 1291.5354 32.9930311 ## 105 1362.1356 1326.8355 35.3000973 ## 106 1413.6272 1370.2313 43.3958605 ## 107 1457.2970 1413.7642 43.5328384 ## 108 1476.0919 1444.9280 31.1638426 ## 109 1507.4636 1476.1958 31.2677808 ## 110 1521.5906 1498.8932 22.6973921 ## 111 1553.2149 1526.0541 27.1608591 ## 112 1587.5582 1556.8061 30.7520667 ## 113 1631.0919 1593.9490 37.1428782 ## 114 1652.4424 1623.1957 29.2466998 ## 115 1681.0676 1652.1316 28.9359304 ## 116 1740.3235 1696.2276 44.0959274 ## 117 1807.0995 1751.6635 55.4359684 ## 118 1876.6528 1814.1582 62.4946565 ## 119 1913.5007 1863.8294 49.6712577

**28 |** P a g e

## 120 1934.7604 1899.2949 35.4654586 ## 121 1969.7119 1934.5034 35.2084919 ## 122 2024.8889 1979.6961 45.1927388 ## 123 2102.7929 2041.2445 61.5483983 ## 124 2180.8763 2111.0604 69.8158650 ## 125 2233.1805 2172.1205 61.0600532 ## 126 2262.1616 2217.1410 45.0205500 ## 127 2380.4411 2298.7910 81.6500385 ## 128 2477.4997 2388.1454 89.3543145 ## 129 2566.2733 2477.2093 89.0639690 ## 130 2613.5116 2545.3604 68.1511172 ## 131 2658.1843 2601.7724 56.4119153 ## 132 2724.4883 2663.1304 61.3579947 ## 133 2785.2080 2724.1692 61.0388206 ## 134 2851.1335 2787.6513 63.4821460 ## 135 2857.9020 2822.7767 35.1253530 ## 136 2891.5310 2857.1538 34.3771594 ## 137 3025.0604 2941.1071 83.9532768 ## 138 3213.5646 3077.3359 136.2287555 ## 139 3295.9785 3186.6572 109.3213179 ## 140 3368.0020 3277.3296 90.6724106 ## 141 3366.7372 3322.0334 44.7038088 ## 142 3313.9663 3317.9999 -4.0335423 ## 143 3328.9201 3323.4600 5.4601283 ## 144 3373.8212 3348.6406 25.1806323 ## 145 3429.3085 3388.9746 40.3339417 ## 146 3516.1515 3452.5630 63.5884625 ## 147 3645.8546 3549.2088 96.6458027 ## 148 3788.1868 3668.6978 119.4889945 ## 149 3914.5504 3791.6241 122.9262983 ## 150 4030.6261 3911.1251 119.5010009 ## 151 4127.9557 4019.5404 108.4153050 ## 152 4190.2825 4104.9115 85.3710402 ## 153 4231.6469 4168.2792 63.3677042 ## 154 4293.4551 4230.8671 62.5879550 ## 155 4357.5530 4294.2101 63.3429499 ## 156 4451.7090 4372.9595 78.7494639 ## 157 4522.4283 4447.6939 74.7343820 ## 158 4581.8360 4514.7650 67.0710327 ## 159 4610.4213 4562.5931 47.8281749 ## 160 4655.3471 4608.9701 46.3769980 ## 161 4704.1283 4656.5492 47.5790858 ## 162 4770.7179 4713.6336 57.0843663 ## 163 4865.1766 4789.4051 75.7715318 ## 164 4961.3830 4875.3941 85.9889586 ## 165 5096.5242 4985.9591 110.5650430 ## 166 5182.6748 5084.3170 98.3578485 ## 167 5288.7952 5186.5561 102.2391087 ## 168 5384.7492 5285.6526 99.0965492 ## 169 5499.4102 5392.5314 106.8788142

**29 |** P a g e

## 170 5618.7775 5505.6545 113.1230211 ## 171 5725.2418 5615.4481 109.7936552 ## 172 5803.5298 5709.4890 94.0408460 ## 173 5838.2088 5773.8489 64.3599276 ## 174 5938.9414 5856.3951 82.5462676 ## 175 6043.7239 5950.0595 93.6643722 ## 176 6107.2207 6028.6401 78.5805866 ## 177 6077.7260 6053.1831 24.5429664 ## 178 6057.4013 6055.2922 2.1091043 ## 179 6132.7580 6094.0251 38.7329355 ## 180 6248.6596 6171.3424 77.3172409 ## 181 6342.7230 6257.0327 85.6903375 ## 182 6449.9034 6353.4680 96.4353461 ## 183 6568.3356 6460.9018 107.4337796 ## 184 6673.9824 6567.4421 106.5402876 ## 185 6787.7151 6677.5786 110.1365189 ## 186 6836.4194 6756.9990 79.4204087 ## 187 6886.8612 6821.9301 64.9310826 ## 188 6946.7694 6884.3498 62.4196414 ## 189 7079.8087 6982.0792 97.7294805 ## 190 7215.3356 7098.7074 116.6281959 ## 191 7365.2818 7231.9946 133.2871593 ## 192 7462.4972 7347.2459 115.2513215 ## 193 7570.1463 7458.6961 111.4501835 ## 194 7631.1300 7544.9131 86.2169731 ## 195 7664.4808 7604.6969 59.7838450 ## 196 7743.9253 7674.3111 69.6141999 ## 197 7843.7628 7759.0369 84.7258218 ## 198 7954.5407 7856.7888 97.7518980 ## 199 8134.8607 7995.8248 139.0359390 ## 200 8270.2557 8133.0402 137.2154634 ## 201 8396.4148 8264.7275 131.6873047 ## 202 8492.5017 8378.6146 113.8871017 ## 203 8634.1472 8506.3809 127.7662909 ## 204 8792.1527 8649.2668 142.8858688 ## 205 8907.3620 8778.3144 129.0475865 ## 206 8993.2987 8885.8066 107.4921722 ## 207 9075.9045 8980.8555 95.0489860 ## 208 9218.6098 9099.7327 118.8771481 ## 209 9416.9779 9258.3553 158.6225868 ## 210 9570.0159 9414.1856 155.8302890 ## 211 9679.6520 9546.9188 132.7332285 ## 212 9819.7394 9683.3291 136.4103221 ## 213 10040.9643 9862.1467 178.8175780 ## 214 10178.8820 10020.5144 158.3676678 ## 215 10409.8407 10215.1775 194.6631458 ## 216 10507.8301 10361.5038 146.3262740 ## 217 10578.1402 10469.8220 108.3182058 ## 218 10572.6661 10521.2440 51.4220443 ## 219 10651.8577 10586.5509 65.3068556

**30 |** P a g e

## 220 10660.3917 10623.4713 36.9203856 ## 221 10697.3894 10660.4303 36.9590526 ## 222 10825.1537 10742.7920 82.3617020 ## 223 10973.2186 10858.0053 115.2132785 ## 224 11099.8433 10978.9243 120.9189707 ## 225 11180.2579 11079.5911 100.6667958 ## 226 11274.4617 11177.0264 97.4352881 ## 227 11412.2896 11294.6580 117.6316221 ## 228 11692.7173 11493.6876 199.0296379 ## 229 11975.5988 11734.6432 240.9556005 ## 230 12161.5593 11948.1013 213.4580486 ## 231 12323.6155 12135.8584 187.7571356 ## 232 12492.0660 12313.9622 178.1037785 ## 233 12707.2340 12510.5981 196.6359159 ## 234 12967.1959 12738.8970 228.2988984 ## 235 13148.5266 12943.7118 204.8148071 ## 236 13347.1360 13145.4239 201.7120857 ## 237 13524.6659 13335.0449 189.6209777 ## 238 13792.8423 13563.9436 228.8987326 ## 239 13980.1737 13772.0586 208.1150344 ## 240 14072.3067 13922.1827 150.1240305 ## 241 14187.8987 14055.0407 132.8580171 ## 242 14350.0221 14202.5314 147.4906806 ## 243 14552.4109 14377.4712 174.9397891 ## 244 14739.6950 14558.5831 181.1119175 ## 245 14894.8267 14726.7049 168.1218259 ## 246 14864.3945 14795.5497 68.8447857 ## 247 14934.6170 14865.0834 69.5336643 ## 248 14966.5908 14915.8371 50.7537230 ## 249 14639.4230 14777.6300 -138.2070493 ## 250 14281.3255 14529.4778 -248.1522642 ## 251 14138.5308 14334.0043 -195.4734922 ## 252 14270.3286 14302.1664 -31.8378293 ## 253 14640.1776 14471.1720 169.0055770 ## 254 14940.4006 14705.7863 234.6142853 ## 255 15216.9767 14961.3815 255.5952168 ## 256 15393.0910 15177.2363 215.8547450 ## 257 15520.7668 15349.0016 171.7652920 ## 258 15513.9779 15431.4898 82.4881940 ## 259 15642.3979 15536.9438 105.4540699 ## 260 15753.4231 15645.1835 108.2396403 ## 261 15955.3503 15800.2669 155.0834394 ## 262 16230.0939 16015.1804 214.9135168 ## 263 16420.7915 16217.9860 202.8055623 ## 264 16516.8254 16367.4057 149.4197147 ## 265 16564.5479 16465.9768 98.5711104 ## 266 16731.1377 16598.5573 132.5804545 ## 267 16830.4094 16714.4833 115.9260567 ## 268 17031.3915 16872.9374 158.4540930 ## 269 17297.1139 17085.0257 212.0882575

**31 |** P a g e

## 270 17348.0369 17216.5313 131.5056309

## 271 17600.4601 17408.4957 191.9644189

## 272 17942.9750 17675.7353 267.2396200

## 273 18114.8492 17895.2923 219.5569212

## 274 18204.1717 18049.7320 154.4397390

## 275 18347.5762 18198.6541 148.9221072

## 276 18453.6677 18326.1609 127.5068204

## 277 18452.9569 18389.5589 63.3979876

## 278 18487.1965 18438.3777 48.8187859

## 279 18667.2191 18552.7984 114.4206987

## 280 18895.7809 18724.2896 171.4912287

## 281 19143.4718 18933.8807 209.5910854

## 282 19364.0723 19148.9765 215.0957839

## 283 19535.7722 19342.3743 193.3978505

## 284 19753.3405 19547.8574 205.4830655

## 285 20095.5151 19821.6863 273.8288414

## 286 20420.1727 20120.9295 299.2432331

## 287 20795.6780 20458.3038 337.3742697

## 288 20989.0303 20723.6670 265.3632498

## 289 21069.1089 20896.3880 172.7209676

## 290 21170.6309 21033.5095 137.1214861

## 291 21432.8575 21233.1835 199.6740138

## 292 21708.6198 21470.9016 237.7181751

## 293 21931.4041 21701.1529 230.2512190

## 294 21687.0826 21694.1177 -7.0351557

## 295 19349.9411 20522.0294 -1172.0883049

## 296 20064.0005 20293.0150 -229.0144295

## 297 21325.6507 20809.3329 516.3178778

## 298 22593.3929 21701.3629 892.0300325

## 299 23641.0342 22671.1986 969.8356668

*#Visualizing the in-sample forecast.*

plot(gdp\_forecast, main = "Figure 17: Time Series Plot for In Sample Forecast")

**32 |** P a g e



Thus, the in-sample forecasted values are obtained in the above table and are plotted in figure 17.

*#Loading the library 'forecast'.*

library(forecast)

*#Forecasting the last 2 observations to compare with actual out-sample forecast result if the model obtained is good or not.*

gdp\_forecast1<-HoltWinters(gdp[1:297], beta=TRUE, gamma = FALSE) gdp\_forecast1

## Holt-Winters exponential smoothing with trend and without seasonal component.

##

## Call:

## HoltWinters(x = gdp[1:297], beta = TRUE, gamma = FALSE) ##

## Smoothing parameters:

## alpha: 0.5170384

## beta : TRUE

## gamma: FALSE

##

## Coefficients:

## [,1]

## a 21670.2463

## b 882.5642

**33 |** P a g e

forecast1<-forecast(gdp\_forecast1,h=2)

forecast1

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

## 298 22552.81 22295.28 22810.34 22158.95 22946.67

## 299 23435.37 23064.91 23805.84 22868.80 24001.95

Thus, from the above for table it is observed that the forecasted values are very different from the actual values in the dataset which indicate the need to check for the accuracy of the predictive model since the accuracy seems to be low on comparison of above predicted value with actual value. We will further check the accuracy using the accuracy measure rmse in the final step.

*#Performing out-sample 2 step ahead forecast for 01-10-2019 and 01-01-2022.* forecast2<-forecast(gdp\_forecast, h=2)

forecast2

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

## 300 24117.84 23858.55 24377.14 23721.28 24514.40

## 301 24841.16 24464.33 25217.99 24264.85 25417.47

Thus, the out of sample forecast for next 2 data points, i.e. US GDP in billions of dollars in 01-10-2019 and 01-01-2022 are obtained above.

*#Plotting the forecasted value.*

plot(forecast2, main = "Figure 18: Plot of Out of Sample Forecasted US GDP", xlab ="Time", ylab="US GDP (In Billions of Dollars)")



**34 |** P a g e

Thus, the forecasted values are plotted in the graph.

Obtaining the accuracy measures and commenting about the findings. Root mean square value

*#Loading the package 'Metrics'.*

library(Metrics)

##

## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':

##

## accuracy

*#Obtaining the accuracy measure for the forecasting model obtained by removing the last value.*

ma<-rmse(gdp,gdp\_forecast$fitted)

ma

## [1] 5355.106

It is observed from the above calculations that the values obtained using forecasting model are different from the actual values also the root mean square for the above model is 5355.106 which is evident to the fact that the above predictive model is not that accurate to predict the future behaviour of the dataset.

**CONCLUSION**

In the above analysis, on obtaining the time series plot of the dataset we observed that our dataset for Quarterly US GDP has a increasing trend component as there’s some kind of increasing pattern is witnessed in the time series plot.

In modelling section we observed that on stationarizing the dataset the best suitable model that we obtained was MA(3) with AIC = -1748.66. Thus, auto.arima gave the best suitable model for the data which is a MA(3) model. It was also observed that even on transformation of the original data the residual analysis gave only one assumption to be true that is the residuals are uncorrelated random variables while the residuals are not normally distributed that is the normality assumption was violated.However, a forecasting method that does not satisfy normality property cannot necessarily be improved. Sometimes applying a Box-Cox transformation may assist with these properties which we have done already, but otherwise there is usually little that you can do to ensure that your residuals have constant variance and a normal distribution. This means that the results we obtained using auto,arima are misleading because for auto.arima we make normality assumption of residuals and estimate the parameters of the time series model which is violated in residual analysis stage that means we made a wrong assumption about the residuals that they were normally distributed and thus got the misleading results. Therefore since the model is not adequate we cannot perform MMSE forecast for this data.

**35 |** P a g e

we obtained the predictive model to forecast the US GDP in 01-10-2019 and 01-01-2022 using Holt’s exponential technique and we observed that the prediction was based on the recent past observations in the dataset. On observing the rmse value we could conclude that the accuracy is not very appropriate for the predictive model for future prediction which is because the model we obtained is not adequate about which a detailed interpretation is given in the above paragraph.