

**May 15, 2022**

A picture containing laser

Description automatically generated

# Final Project Report

## Yash Admuthe

MA-641-A: Time Series Analysis

Stevens Institute of Technology-Spring’22

**Introduction:**

A time series is **a sequence taken at successive equally spaced points in time**. Thus it is a sequence of discrete-time data. Examples of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average.

**Four components of time series are:**

* Secular trend, which describe the movement along the term;
* Seasonal variations, which represent seasonal changes;
* Cyclical fluctuations, which correspond to periodical but not seasonal variations;
* Irregular variations, which are other non-random sources of variations of series.

Based on timeseries theories in statistics, this study analyzes two datasets and offers a plausible model that can persuasively explain its timeseries pattern. As a result, examining these datasets should provide a great opportunity for students, particularly those who are just starting to understand timeseries models, to practice their knowledge and develop their statistical analytic skills.

The project's seasonal and non-seasonal datasets are as follows:

1.Temperature change in Delhi, India, and forecasting future temperatures

2. Commercial banks' assets and liabilities in the United States to anticipate credit growth rate

**PROJECT 1**

**[US](https://ycharts.com/indicators/us_commercial_banks_bank_credit_annual_growth_rate_yearly) Commercial Banks Bank Credit Annual Growth Rate Prediction**

**Data Description :**

The information pertains to the assets and liabilities of commercial banks in the United States.

The data is obtained from federalreserve.gov via the Federal Reserve Board of Governors' data download program. We will do a univariate time series analysis and forecasting using data from US commercial banks' credit annual growth variables. From 1947 until 2022, this is quarterly data. So, except for dates and credit yearly growth rate, we will eliminate all other columns from the data. Entire analysis has been done in R.

**Exploring Data :**

After importing data , firstly we checked for NA’s in data . Fortunately there was no null values in the data.

data <- read.csv("/Users/yashadmuthe/Desktop/FRB\_H8.1.csv")

data

Table

Description automatically generated

Then we convert our data into time series data using ts() function.

The ts() function will convert a numeric vector into an R time series object. The format is ts(vector, start=, end=, frequency=) where start and end are the times of the first and last observation and frequency is the number of observations per unit time (1=annual, 4=quartly, 12=monthly, etc.).

# Converting data into TS data and checking dimentions :

datats <- ts(data2$Bank.credit, frequency = 4, start = c(1963, 1))

autoplot(datats)

**A picture containing antenna

Description automatically generated**

We check for mean and variance of dataset for better understanding :

> mean(datats)

[1] 7.989865

> var(datats)

[1] 13.23983

After plotting the time series data, we checked the box plot to see how the points in the data were distributed quarterly.

plot.ts(datats)

abline(reg=lm(datats~time(datats))) # This will fit in a line

plot(aggregate(datats,FUN=mean)) #This will aggregate the cycles and display a year on year trend

boxplot(datats~cycle(datats))

**Chart, box and whisker chart

Description automatically generated**

**Stationarity Test :**

Stationarity is a critical factor in time series analysis. Because a model cannot forecast on non-stationary time series data, the first step in ARIMA time series forecasting is to identify the number of differencing required to make the series stationary.

A stationary series is one in which the statistical features such as mean, variance, and covariance do not vary with time or are not a function of time.

Two statistical tests which we will be using to check stationarity are :

1. Augmented Dickey-Fuller (ADF) Test
2. Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test

**Augmented Dickey-Fuller (ADF) Test :**

The Augmented Dickey-Fuller test is a type of statistical test called a unit root test.

In probability theory and statistics, a unit root is a feature of some stochastic processes (such as random walks) that can cause problems in statistical inference involving time series models. In a simple term, the unit root is non-stationary but does not always have a trend component.

ADF test is conducted with the following assumptions.

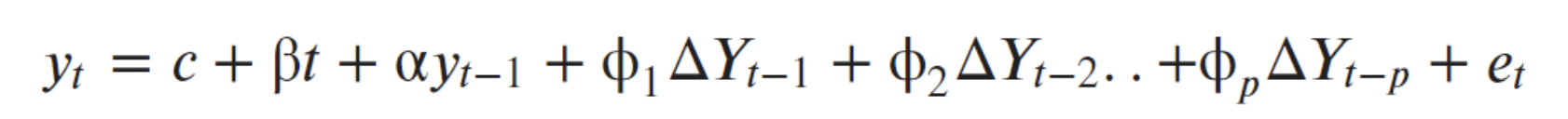
Null Hypothesis (HO): Series is non-stationary or series has a unit root.

Alternate Hypothesis(HA): Series is stationary or series has no unit root.

If the null hypothesis is failed to be rejected, this test may provide evidence that the series is non-stationary.

Conditions to Reject Null Hypothesis(HO)

If Test statistic < Critical Value and p-value < 0.05 – Reject Null Hypothesis(HO) i.e., time series does not have a unit root, meaning it is stationary. It does not have a time-dependent structure.



adf.test(datats)

adf.test(datats, k=2)

**Text

Description automatically generated**

We reject the null hypothesis since the p value is less than 0.05, and the alternative hypothesis shows that the data is stationary. We don't need to execute differencing because the data is steady, hence our differencing (d) equals 0.

**Kwiatkowski-Phillips-Schmidt-Shin (KPSS) :**

The KPSS test (Kwiatkowski-Phillips-Schmidt-Shin) is a sort of Unit root test that checks for the stationarity of a given series around a deterministic trend.

In other words, the test is conceptually comparable to the ADF test.

However, it is a frequent misconception that it can be used interchangeably with the ADF test. This can lead to misunderstandings about stationarity, which can easily go undetected, producing further problems down the road.

KPSS test is conducted with the following assumptions.

Null Hypothesis (HO): Series is trend stationary or series has no unit root.

Alternate Hypothesis(HA): Series is non-stationary or series has a unit root.

Note that Hypothesis is reversed in KPSS test compared to ADF Test.

If the null hypothesis is failed to be rejected, this test may provide evidence that the series is trend stationary.

Conditions to Fail to Reject Null Hypothesis(HO)

If Test statistic < Critical Value and p-value < 0.05 – Fail to Reject Null Hypothesis(HO) i.e., time series does not have a unit root, meaning it is trend stationary.

In order to reject the null hypothesis, the test statistic should be greater than the provided critical values. If it is in fact higher than the target critical value, then that should automatically reflect in a low p-value.

That is, if the p-value is less than 0.05, the kpss statistic will be greater than the 5% critical value.

kpss.test(datats)

Diagram, text

Description automatically generated

Here as the p-value is greater than 0.05, the data is stationary.

**ACF and PACF Evaluation :**

Autocorrelation Function (ACF)

Time series correlation with a delayed version of itself. The relationship between observations made in the current time and observations made at earlier times. The autocorrelation function begins with a lag of zero, which is the correlation of the time series with itself, resulting in a correlation of one.

The plot acf function from the statsmodels.graphics.tsaplots package will be used.

The following questions can be answered using the ACF plot:

Is the observed time series random or white noise?

Is one observation related to another, to an observation twice removed, and so on?

Can an MA model be used to model the observed time series? If so, what is the sequence?

Partial Autocorrelation Function (PACF)

Each each lagged term explains more correlation. Given that both observations are connected to observations at other time points, the correlation between pbservations at two time points.

The plot pacf function from the statsmodels.graphics.tsaplots package will be used, using the option method = "ols" (regression of time series on lags of it and on constant).

The following questions can be answered using the PACF plot:

Can an AR model be used to model the observed time series? If so, what is the sequence?

Both the ACF and PACF start with a lag of 0, which is the correlation of the time series with itself and therefore results in a correlation of 1.

# checking ACF AND PACF

acf <- acf(as.vector(datats),plot = FALSE, lag.max = 100)

plot(acf)

Chart, histogram

Description automatically generated

# In ACF plot we see 11 spikes are significant, out of which 5 are more significant

# Also note that we ignore the spike at lag 0.

# We also check how well the present value of the series is related with its past values.

# A time series can have components like trend, seasonality, cyclic and residual.

# ACF considers all these components while finding correlations hence it's a 'complete auto-correlation plot'.

# From ACF plot we can say MA can be 1,2 or 5

pacf <- pacf(as.vector(datats),lag.max = 100)

plot(pacf)

# In PACF plot we see 2 spikes are significant, out of which 1 are more significant

# From PACF, we can say AR can be 1

Chart, histogram

Description automatically generated

Graphical user interface, chart, line chart

Description automatically generated

We also checked for EACF, EACF allows for the identification of ARIMA models.

eacf(as.vector(datats))

Background pattern

Description automatically generated with medium confidence

**Model Building :**

The ACF and PACF of data suggests that the following ARIMA model could be the best fit. So, We will consider specifying the multiplicative,

ARIMA (1,0,1/2/4/5) model.

#Checking best model fit by comparing their AIC,BIC and log likelihood values.

mod<- arma(datats, order = c(1, 1),lag = NULL, coef = NULL,

include.intercept = TRUE)

summary(mod)

#Fit:sigma^2 estimated as 9.978, Conditional Sum-of-Squares = 1456.74, AIC = 766.46

mod2<- arma(datats, order = c(1, 2))

summary(mod2)

#Fit:sigma^2 estimated as 9.989, Conditional Sum-of-Squares = 1448.45, AIC = 768.63

mod3<- arma(datats, order = c(1, 4))

summary(mod3)

#Fit:sigma^2 estimated as 9.9, Conditional Sum-of-Squares = 1415.66, AIC = 771.3

mod4<- arma(datats, order = c(1, 5))

summary(mod4)

#Fit:sigma^2 estimated as 9.896, Conditional Sum-of-Squares = 1405.21, AIC = 773.24

tsmod <-Arima(datats, order=c(1,0,1))

summary(tsmod)

#Fit : log likelihood = -379.42 AIC=766.84 AICc=767.12 BIC=778.83

tsmod2 <-Arima(datats, order=c(1,0,2))

show(tsmod2)

#Fit : log likelihood = -379.32 AIC=768.63 AICc=769.05 BIC=783.62

tsmod3 <-Arima(datats, order=c(1,0,5))

summary(tsmod3)

#Fit : log likelihood = -377.62 AIC=771.25 AICc=772.28 BIC=795.23

tsmod4 <-Arima(datats, order=c(1,0,20))

show(tsmod4)

#Fit : log likelihood = -367.26 AIC=780.53 AICc=789.43 BIC=849.46

#Comparing all the models we can confidently say that ARIMA (1,0,1) fits very well

After analyzing all models, tsmod(i.e. ARIMA(1,0,1)) provided the best estimation parameters, resulting in a low AIC (766.84). In the following phase, I used residual analysis to determine whether a model adequately captured the information in the data.

**GARCH Model:**

GARCH processes are widely used in finance due to their effectiveness in modeling asset returns and inflation. GARCH aims to minimize errors in forecasting by accounting for errors in prior forecasting and enhancing the accuracy of ongoing predictions.

I used Garch model as the variance of the error term is not constant.

# install.packages("rugarch", repos=c("http://rstudio.org/\_packages", "http://cran.rstudio.com"))

s <- ugarchspec(mean.model=list(armaOrder = c(1,1)),

variance.model = list(model= 'sGARCH'),

distribution.model ="norm")

m <- ugarchfit(data = datats, spec = s, solver.control = list(trace=0))

#stargazer(list(m), title="Regression Results", type="text", keep.stat=c("n","ll","aic","bic"), out="kindaresults.doc")

show(m)

Table

Description automatically generated

#By comparing loglikelihood of ARIMA and Garch model, we conclude ARIMA(1,0,1) is still the best fit.

**Checking Residuals :**

The residuals plot do not show any significant auto correlation which means that our model is adequately built. Let us further examine the residuals for test of significant autocorrelation by examining performing the Box test and plotting histogram.

r<- residuals(tsmod)

plot(r,ylab ='Standardized Residuals',type='o');

abline(h=0)

A picture containing schematic

Description automatically generated

**Normality Test on Residuals :**

Normality of the residuals is an assumption of running a linear model. So, if our residuals are normal, it means that our assumption is valid and model inference (confidence intervals, model predictions) should also be valid.

Normality is the assumption that the underlying residuals are normally distributed, or approximately so. While a residual plot, or normal plot of the residuals can identify non-normality, we can formally test the hypothesis using the Shapiro-Wilk or similar test.

qqnorm(r,start=c(1963,1))

qqline(r,start=c(1963,1))

r

r2<- na.omit(r)

acf(as.vector(r2), lag.max = 100)

Chart, line chart, scatter chart

Description automatically generated

**Shapiro-wilk test**

When the distribution of a real continuous random variable is unknown, it is convenient to assume that it is normally distributed. However, this may not always be true leading to incorrect results. To avert this problem, there is a statistical test by the name of Shapiro-Wilk Test that gives us an idea whether a given sample is normally distributed or not. The test works as follows:

Specify the null hypothesis and the alternative hypothesis as:

H0 : the sample is normally distributed  
HA : the sample is not normally distributed

A test statistic is computed as follows:

**Text

Description automatically generated**

The purpose of this test is to see if the data is normally distributed or not

# Null Hypothesis : The data is normally distributed

# Alternative Hypothesis : The data is not normally distributed

shapiro.test(datats)

Graphical user interface, text, application

Description automatically generated

From the output, the p-value(0.3504) > 0.05 implying that the distribution of the data are not significantly different from normal distribution. In other words, we can assume the normality.

**Ljung-Box test :**

The Ljun-Box test is a hypothesis test that checks if a time series contains an autocorrelation. The null Hypothesis H0 is that the residuals are independently distributed. The alternative hypothesis is that the residuals are not independently distributed and exhibit a serial correlation.

The Ljung-Box test uses the following hypotheses:

H0: The residuals are independently distributed.

HA: The residuals are not independently distributed; they exhibit serial correlation.

Ideally, we would like to fail to reject the null hypothesis. That is, we would like to see the p-value of the test be greater than 0.05 because this means the residuals for our time series model are independent, which is often an assumption we make when creating a model.

It is a statistical test of whether any group of autocorrelations of a time series

# are different from 0. Instead of testing randomness of each distinct lag,

# it tests overall randomness based on number of lags.

# H0 : The series is i.i.d

# H1 : The series exhibits serial correlation

Box.test(datats, lag = 10, fitdf = 0, type = 'Lj')

Chart

Description automatically generated with low confidence

If p-value < 0.051: You can reject the null hypothesis assuming a 5% chance of making a mistake. So you can assume that your values are showing dependence on each other.

# As here we get a very small p-value, reject H0. The series is not white noise.

We plot the data to see if its normally distributed:

gghistogram(r) + ggtitle("Histogram of residuals")

Chart, histogram

Description automatically generated

**Forecasting :**

This plot demonstrates the next 12 months forecasts for the ARIMA (1,0,1) model that we fit before.

# Forecasting

ts.plot(datats, xlim=c(1963,2003),main = "Prediction")

fit = datats - r

points(fit, type = 'l', col='red', lty =2)

prediction = predict(tsmod)

prediction$pred[1]

predict(tsmod, n.ahead =12)

forecast <- predict(tsmod, n.ahead = 12)$pred

forecast\_se <- predict(tsmod, n.ahead = 12)$se

points(forecast, type = "l", col = 2)

points(forecast - 2\*forecast\_se, type = "l", col = 2, lty = 2)

points(forecast + 2\*forecast\_se, type = "l", col = 2, lty = 2)

fcst <- forecast(tsmod, h=6)

autoplot(fcst, include= 60)

print(summary(fcst))

Chart, line chart

Description automatically generated

Conclusion :

Forecast for next six Quarters are as follows

Forecasts:

Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

2000 Q1 7.838334 3.775077 11.90159 1.6241192 14.05255

2000 Q2 7.875995 3.499985 12.25201 1.1834657 14.56852

2000 Q3 7.902917 3.375425 12.43041 0.9787158 14.82712

2000 Q4 7.922161 3.319188 12.52514 0.8825206 14.96180

2001 Q1 7.935918 3.294848 12.57699 0.8380132 15.03382

2001 Q2 7.945752 3.285334 12.60617 0.8182584 15.07325

**PROJECT 2**

**Seasonal Data Analysis of Delhi, India**

**Data Description :**

The information pertains temperature of Delhi City from India.

The data is obtained from federalreserve.gov via the Federal Reserve Board of Governors' data download program. We will do a univariate time series analysis and forecasting using data From 2001 until 2012, this is monthly data. So, except for dates and temperature of Delhi, we will eliminate all other columns from the data. Entire analysis has been done in R.

**Exploring Data :**

After importing data , firstly we checked for NA’s in data . Fortunately there was no null values in the data.

data <- read.csv("/Users/yashadmuthe/Desktop/Delhi\_temp.csv")

dataclass(datats

Table

Description automatically generated)

Then we convert our data into time series data using ts() function.

The ts() function will convert a numeric vector into an R time series object. The format is ts(vector, start=, end=, frequency=) where start and end are the times of the first and last observation and frequency is the number of observations per unit time (1=annual, 4=quartly, 12=monthly, etc.).

# Converting data into TS data and checking dimentions :

datats <- ts(data$Delhi\_Temperature, frequency = 12, start = c(2001, 1))

class(datats)

autoplot(datats)

**Diagram

Description automatically generated with medium confidence**

We check for mean and variance of dataset for better understanding :

mean(datats)

[1] 26.07953

> var(datats)

[1] 44.51562

After plotting the time series data, we checked the box plot to see how the points in the data were distributed quarterly.

plot.ts(datats)

abline(reg=lm(datats~time(datats))) # This will fit in a line

plot(aggregate(datats,FUN=mean))

boxplot(datats~cycle(datats))

**Chart, waterfall chart

Description automatically generated**

**Stationarity Test :**

Stationarity is a critical factor in time series analysis. Because a model cannot forecast on non-stationary time series data, the first step in ARIMA time series forecasting is to identify the number of differencing required to make the series stationary.

A stationary series is one in which the statistical features such as mean, variance, and covariance do not vary with time or are not a function of time.

Two statistical tests which we will be using to check stationarity are :

1. Augmented Dickey-Fuller (ADF) Test
2. Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test

**Augmented Dickey-Fuller (ADF) Test :**

The Augmented Dickey-Fuller test is a type of statistical test called a unit root test.

In probability theory and statistics, a unit root is a feature of some stochastic processes (such as random walks) that can cause problems in statistical inference involving time series models. In a simple term, the unit root is non-stationary but does not always have a trend component.

ADF test is conducted with the following assumptions.

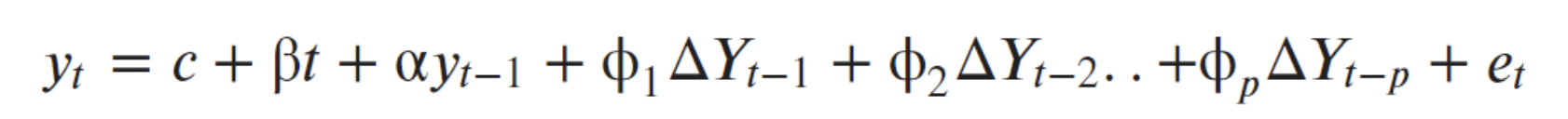
Null Hypothesis (HO): Series is non-stationary or series has a unit root.

Alternate Hypothesis(HA): Series is stationary or series has no unit root.

If the null hypothesis is failed to be rejected, this test may provide evidence that the series is non-stationary.

Conditions to Reject Null Hypothesis(HO)

If Test statistic < Critical Value and p-value < 0.05 – Reject Null Hypothesis(HO) i.e., time series does not have a unit root, meaning it is stationary. It does not have a time-dependent structure.



adf.test(datats)

adf.test(datats, k=2)

Text

Description automatically generated

We reject the null hypothesis since the p value is less than 0.05, and the alternative hypothesis shows that the data is stationary. We don't need to execute differencing because the data is steady, hence our differencing (d) equals 0.

**Kwiatkowski-Phillips-Schmidt-Shin (KPSS) :**

The KPSS test (Kwiatkowski-Phillips-Schmidt-Shin) is a sort of Unit root test that checks for the stationarity of a given series around a deterministic trend.

In other words, the test is conceptually comparable to the ADF test.

However, it is a frequent misconception that it can be used interchangeably with the ADF test. This can lead to misunderstandings about stationarity, which can easily go undetected, producing further problems down the road.

KPSS test is conducted with the following assumptions.

Null Hypothesis (HO): Series is trend stationary or series has no unit root.

Alternate Hypothesis(HA): Series is non-stationary or series has a unit root.

Note that Hypothesis is reversed in KPSS test compared to ADF Test.

If the null hypothesis is failed to be rejected, this test may provide evidence that the series is trend stationary.

Conditions to Fail to Reject Null Hypothesis(HO)

If Test statistic < Critical Value and p-value < 0.05 – Fail to Reject Null Hypothesis(HO) i.e., time series does not have a unit root, meaning it is trend stationary.

In order to reject the null hypothesis, the test statistic should be greater than the provided critical values. If it is in fact higher than the target critical value, then that should automatically reflect in a low p-value.

That is, if the p-value is less than 0.05, the kpss statistic will be greater than the 5% critical value.

kpss.test(datats)

Text

Description automatically generated with medium confidence

Here as the p-value is greater than 0.05, the data is stationary.

**ACF and PACF Evaluation :**

Autocorrelation Function (ACF)

Time series correlation with a delayed version of itself. The relationship between observations made in the current time and observations made at earlier times. The autocorrelation function begins with a lag of zero, which is the correlation of the time series with itself, resulting in a correlation of one.

The plot acf function from the statsmodels.graphics.tsaplots package will be used.

The following questions can be answered using the ACF plot:

Is the observed time series random or white noise?

Is one observation related to another, to an observation twice removed, and so on?

Can an MA model be used to model the observed time series? If so, what is the sequence?

Partial Autocorrelation Function (PACF)

Each each lagged term explains more correlation. Given that both observations are connected to observations at other time points, the correlation between pbservations at two time points.

The plot pacf function from the statsmodels.graphics.tsaplots package will be used, using the option method = "ols" (regression of time series on lags of it and on constant).

The following questions can be answered using the PACF plot:

Can an AR model be used to model the observed time series? If so, what is the sequence?

Both the ACF and PACF start with a lag of 0, which is the correlation of the time series with itself and therefore results in a correlation of 1.

# checking ACF AND PACF

acf <- acf(as.vector(datats),plot = FALSE, lag.max = 100)

plot(acf)

Chart, box and whisker chart

Description automatically generated

# In ACF plot we see 11 spikes are significant, out of which 5 are more significant

# Also note that we ignore the spike at lag 0.

# We also check how well the present value of the series is related with its past values.

# A time series can have components like trend, seasonality, cyclic and residual.

# ACF considers all these components while finding correlations hence it's a 'complete auto-correlation plot'.

# From ACF plot we can say MA can be 1,2 or 5

pacf <- pacf(as.vector(datats),lag.max = 100)

plot(pacf)

# In PACF plot we see 2 spikes are significant, out of which 1 are more significant

# From PACF, we can say AR can be 1

Chart, histogram

Description automatically generated

Chart, line chart

Description automatically generated

We also checked for EACF, EACF allows for the identification of ARIMA models.

eacf(as.vector(datats))

A picture containing calendar

Description automatically generated

**Model Building :**

The ACF and PACF of data suggests that the following SARIMA model could be the best fit. So, We will consider specifying the multiplicative,

Arima-(2/3/8,0,9/10) (0,0,9/10)s=12

#Checking best model fit

mod1 <- arima(datats,order=c(2,0,9), seasonal = list(order = c(0,0,9), period = 12),method='CSS')

summary(mod1)

# sigma^2 estimated as 2.221: part log likelihood = -261.79

mod2<- arima(datats,order=c(8,0,9), seasonal = list(order = c(0,0,9), period = 12),method='CSS')

summary(mod2)

# sigma^2 estimated as 0.8826: part log likelihood = -195.34

mod3<- arima(datats,order=c(3,0,9), seasonal = list(order = c(0,0,9), period = 12),method='CSS')

summary(mod3)

# sigma^2 estimated as 1.843: part log likelihood = -248.35

mod4<- arima(datats,order=c(2,0,9), seasonal = list(order = c(0,0,10), period = 12),method='CSS')

summary(mod4)

#sigma^2 estimated as 2.153: part log likelihood = -259.54

tsmod <-arima(datats,order=c(8,0,9), seasonal = list(order = c(0,0,10), period = 12),method='CSS')

show(tsmod)

# sigma^2 estimated as 0.8902: part log likelihood = -195.95

tsmod2 <-arima(datats,order=c(3,0,9), seasonal = list(order = c(0,0,10), period = 12),method='CSS')

show(tsmod2)

# sigma^2 estimated as 1.519: part log likelihood = -234.41

tsmod3 <-arima(datats,order=c(2,0,10), seasonal = list(order = c(0,0,9), period = 12),method='CSS')

summary(tsmod3)

# sigma^2 estimated as 2.285: part log likelihood = -263.83

tsmod4 <-arima(datats,order=c(8,0,10), seasonal = list(order = c(0,0,9), period = 12),method='CSS')

show(tsmod4)

# sigma^2 estimated as 1.509: part log likelihood = -233.97

tsmod5 <-arima(datats,order=c(3,0,10), seasonal = list(order = c(0,0,9), period = 12),method='CSS')

show(tsmod5)

# sigma^2 estimated as 2.132: part log likelihood = -258.85

# Comparing all the models we can confidently say that ARIMA(8,0,9)(0,0,9)s=12 fits very well

After analyzing all models, tsmod(i.e. ARIMA(8,0,9)(0,0,9)s=12 )provided the best estimation parameters, resulting in a log likelihood (-195.95). In the following phase, I used residual analysis to determine whether a model adequately captured the information in the data.

**GARCH Model:**

GARCH processes are widely used in finance due to their effectiveness in modeling asset returns and inflation. GARCH aims to minimize errors in forecasting by accounting for errors in prior forecasting and enhancing the accuracy of ongoing predictions.

I used Garch model as the variance of the error term is not constant.

# install.packages("rugarch", repos=c("http://rstudio.org/\_packages", "http://cran.rstudio.com"))

s <- ugarchspec(mean.model=list(armaOrder = c(8,9)),

variance.model = list(model= 'sGARCH'),

distribution.model ="norm")

m <- ugarchfit(data = datats, spec = s, solver.control = list(trace=0))

show(m)

# Checked fitting different Garch models(8,1)(8,9)(2,2)(1,9) and after comparing with ARIMA

# found still the best model to be ARIMA(8,0,9)(0,0,9)s=12.

Table

Description automatically generated

Table

Description automatically generated

#By comparing loglikelihood of ARIMA and Garch model, we conclude ARIMA(8,0,9)(0,0,9)s=12 ) is still the best fit.

Lag plots :

**Diagram

Description automatically generated**

**Checking Residuals :**

The residuals plot do not show any significant auto correlation which means that our model is adequately built. Let us further examine the residuals for test of significant autocorrelation by examining performing the Box test and plotting histogram.

r<- residuals(tsmod)

plot(r,ylab ='Standardized Residuals',type='o');

abline(h=0)

A picture containing diagram

Description automatically generated

**Normality Test on Residuals :**

Normality of the residuals is an assumption of running a linear model. So, if our residuals are normal, it means that our assumption is valid and model inference (confidence intervals, model predictions) should also be valid.

Normality is the assumption that the underlying residuals are normally distributed, or approximately so. While a residual plot, or normal plot of the residuals can identify non-normality, we can formally test the hypothesis using the Shapiro-Wilk or similar test.

qqnorm(r,start=c(2001,1))

qqline(r,start=c(2001,1))

r

r2<- na.omit(r)

acf(as.vector(r2), lag.max = 100)

Chart, line chart

Description automatically generated

ACF of residuals :

Chart

Description automatically generated

**Shapiro-wilk test**

When the distribution of a real continuous random variable is unknown, it is convenient to assume that it is normally distributed. However, this may not always be true leading to incorrect results. To avert this problem, there is a statistical test by the name of Shapiro-Wilk Test that gives us an idea whether a given sample is normally distributed or not. The test works as follows:

Specify the null hypothesis and the alternative hypothesis as:

H0 : the sample is normally distributed  
HA : the sample is not normally distributed

A test statistic is computed as follows:

**Text

Description automatically generated**

The purpose of this test is to see if the data is normally distributed or not

# Null Hypothesis : The data is normally distributed

# Alternative Hypothesis : The data is not normally distributed

shapiro.test(datats)

Text

Description automatically generated

From the output, the p-value(0.3504) > 0.05 implying that the distribution of the data are not significantly different from normal distribution. In other words, we can assume the normality.

**Ljung-Box test :**

The Ljun-Box test is a hypothesis test that checks if a time series contains an autocorrelation. The null Hypothesis H0 is that the residuals are independently distributed. The alternative hypothesis is that the residuals are not independently distributed and exhibit a serial correlation.

The Ljung-Box test uses the following hypotheses:

H0: The residuals are independently distributed.

HA: The residuals are not independently distributed; they exhibit serial correlation.

Ideally, we would like to fail to reject the null hypothesis. That is, we would like to see the p-value of the test be greater than 0.05 because this means the residuals for our time series model are independent, which is often an assumption we make when creating a model.

It is a statistical test of whether any group of autocorrelations of a time series

# are different from 0. Instead of testing randomness of each distinct lag,

# it tests overall randomness based on number of lags.

# H0 : The series is i.i.d

# H1 : The series exhibits serial correlation

Box.test(datats, lag = 10, fitdf = 0, type = 'Lj')

Chart

Description automatically generated with medium confidenceIf p-value < 0.051: You can reject the null hypothesis assuming a 5% chance of making a mistake. So you can assume that your values are showing dependence on each other.

# As here we get a very small p-value, reject H0. The series is not white noise.

We plot the data to see if its normally distributed:

gghistogram(r) + ggtitle("Histogram of residuals")

Chart, histogram

Description automatically generated

**Forecasting :**

This plot demonstrates the next 12 months forecasts for the ARIMA(8,0,9)(0,0,9)s=12 ) model that we fit before.

# Forecasting

ts.plot(datats, xlim=c(2000,2015),main = "Prediction")

fit = datats - r

points(fit, type = 'l', col='red', lty =2)

prediction = predict(mod2)

prediction$pred[1]

predict(mod2, n.ahead =12)

forecast <- predict(mod2, n.ahead = 12)$pred

forecast\_se <- predict(mod2, n.ahead = 12)$se

points(forecast, type = "l", col = 2)

points(forecast - 2\*forecast\_se, type = "l", col = 2, lty = 2)

points(forecast + 2\*forecast\_se, type = "l", col = 2, lty = 2)Chart, histogram

Description automatically generated

Conclusion :

Forecast for next six Quarters are as follows

Forecasts:

$pred

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov

2013 14.76415 19.08078 23.63291 29.63971 34.60511 34.46101 31.85293 31.42585 29.31588 26.48900 22.00603

Dec

2013 16.17856