May 15, 2022

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 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.

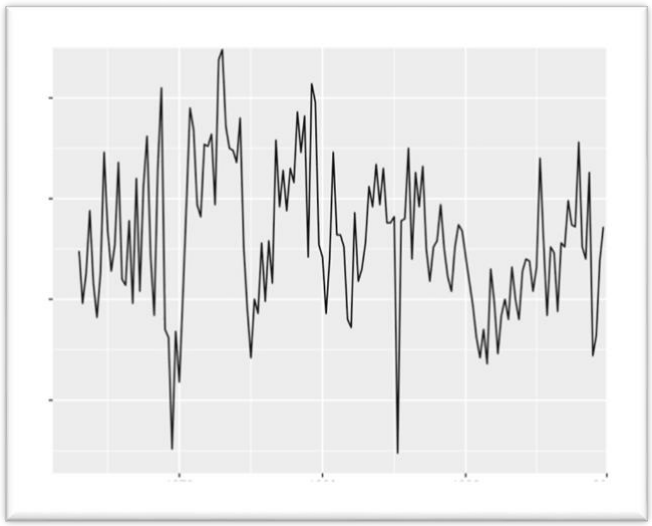
```
data <- read.csv("/Users/yashadmuthe/Desktop/FRB_H8.1.csv")
data
```

Series Description	commercial banks credit annual growth rate	Securities growth rate	Treasury and agency securities, annual growth rate	Loans and leases annual growth rate
1947Q2	-2.1	-9.1	-8	17.4
1947Q3	5.4	-9.7	-4.3	44.5
1947Q4	0	-1.5	-2.4	3.3
1948Q1	-3.3	-11	-12.3	14.1
1948Q2	-1.3	-12.1	-13.1	21.6
1948Q3	0.3	-5.1	-5.9	10.6
1948Q4	-4.4	-7.1	-7.3	0.7
1949Q1	0.5	-3.2	-3.5	7.4
1949Q2	5.2	1.2	0.9	11.6
1949Q3	13.9	20.2	20.2	3.5
1949Q4	5.2	11.2	11.3	-5.6

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)
```

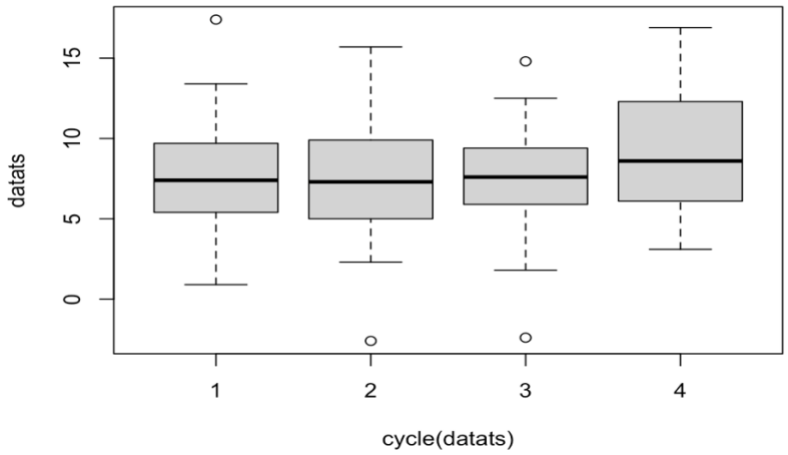


We check for mean and variance of dataset for better understanding :

```
> 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))
```



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.

$$y_t = c + \beta t + \alpha y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} . . + \phi_p \Delta Y_{t-p} + e_t$$

adf.test(datats)
adf.test(datats, k=2)

```
        Augmented Dickey-Fuller Test

data:  datats
Dickey-Fuller = -4.2304, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary
```

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)
```

```
        KPSS Test for Level Stationarity

data:  datats
KPSS Level = 0.3828, Truncation lag parameter = 4, p-value = 0.08457
```

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?

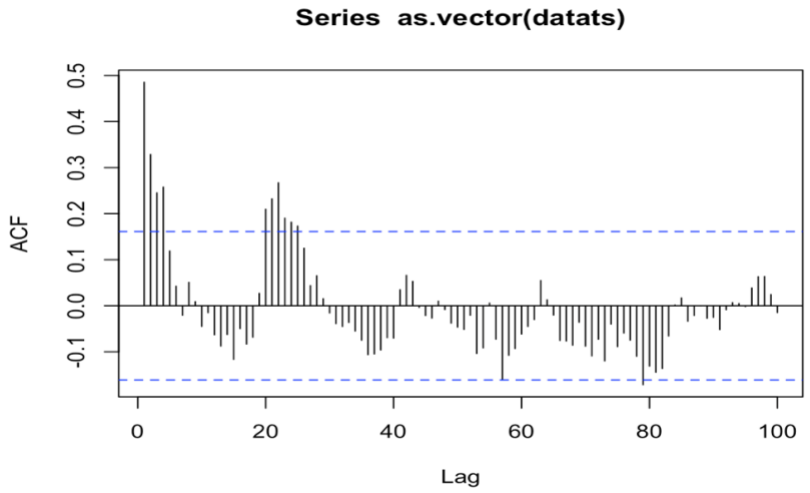
Partial Autocorrelation Function (PACF)

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

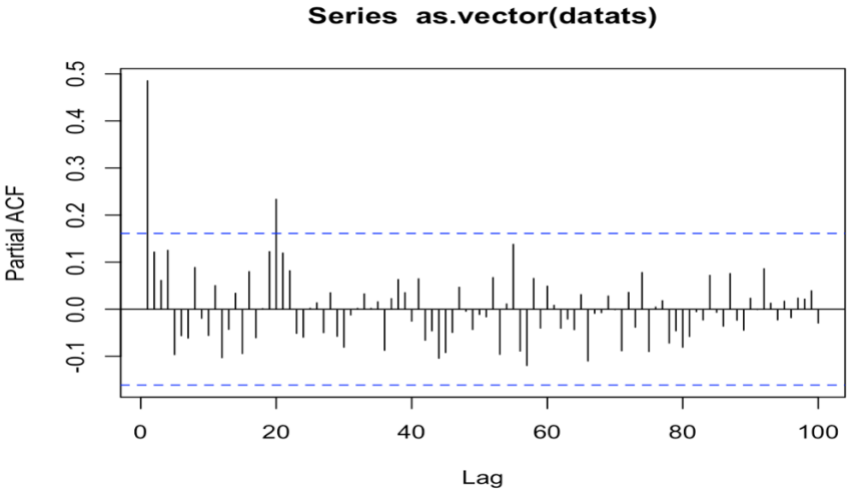
```
acf <- acf(as.vector(datats),plot = FALSE, lag.max = 100)
plot(acf)
```

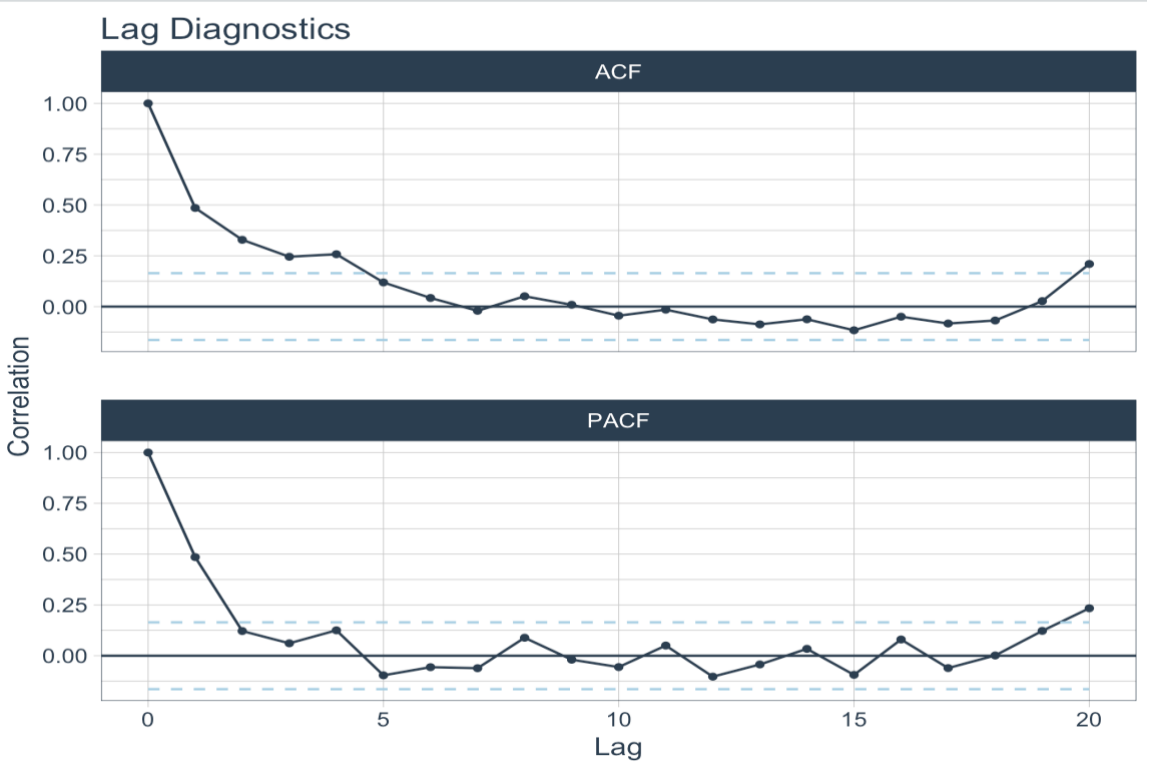


In ACF plot we see 11 spikes are significant, out of which 5 are more significant

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





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

eacf(as.vector(datats))

```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x x x o o o o o o o  o  o  o
1 x o o x o o o o o o o  o  o  o
2 x o o x o o o o o o o  o  o  o
3 x x x x o o o o o o o  o  o  o
4 x x x o o o o o o o o  o  o  o
5 x o x o o o o o o o o  o  o  o
6 x o x x o o o o o o o  o  o  o
7 x x o o x x o o o o o  o  o  o
```

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)
```

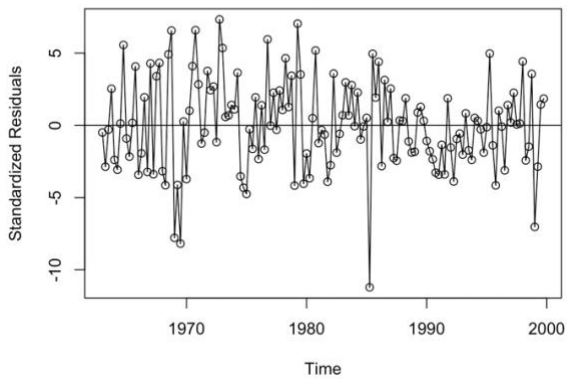
```
*----------------------------------*
*          GARCH Model Fit         *
*----------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model      : sGARCH(1,1)
Mean Model       : ARFIMA(1,0,1)
Distribution     : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error     t value Pr(>|t|)
mu       7.89726    0.613651   12.869301 0.000000
ar1      0.72616    0.113187    6.415575 0.000000
ma1     -0.32894    0.158019   -2.081683 0.037371
omega    0.00000    0.001653    0.000054 0.999957
alpha1   0.00000    0.000151    0.000006 0.999995
beta1    0.99819    0.000183 5458.243446 0.000000

Robust Standard Errors:
        Estimate  Std. Error     t value Pr(>|t|)
mu       7.89726    0.672492   11.743267 0.000000
ar1      0.72616    0.152249    4.769562 0.000002
ma1     -0.32894    0.207775   -1.583178 0.113381
omega    0.00000    0.000322    0.000276 0.999779
alpha1   0.00000    0.000240    0.000004 0.999997
beta1    0.99819    0.000292 3416.321391 0.000000

LogLikelihood : -378.5306
```

#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.

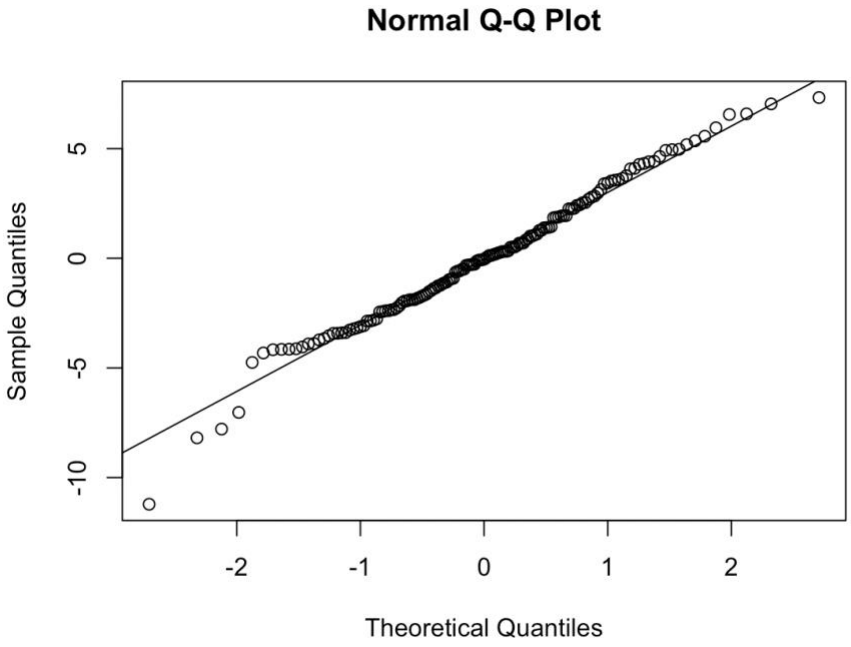
```
r<- residuals(tsmod)

plot(r,ylab ='Standardized Residuals',type='o');
abline(h=0)
```



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)
```



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:

$$W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n} (x_i - \bar{x})}$$

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)

```
        Shapiro-Wilk normality test

data:  datats
W = 0.9897, p-value = 0.3504
```

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:

H_0: The residuals are independently distributed.

H_A: 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.

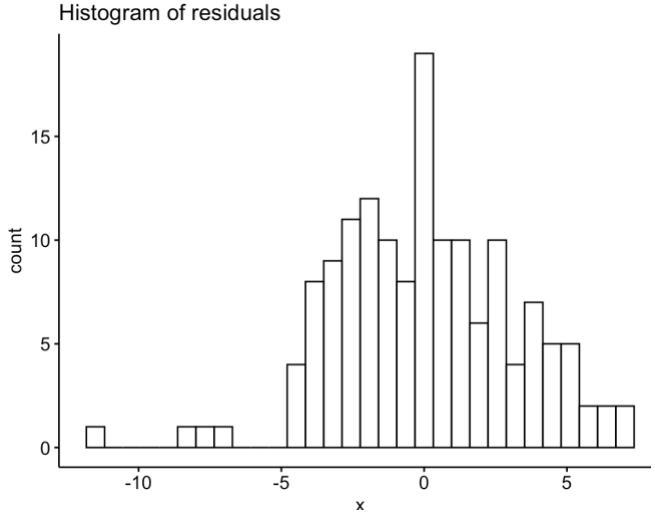
H0 : The series is i.i.d
H1 : The series exhibits serial correlation

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

```
        Box-Ljung test

data:  datats
X-squared = 74.74, df = 10, p-value = 5.345e-12
```

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.

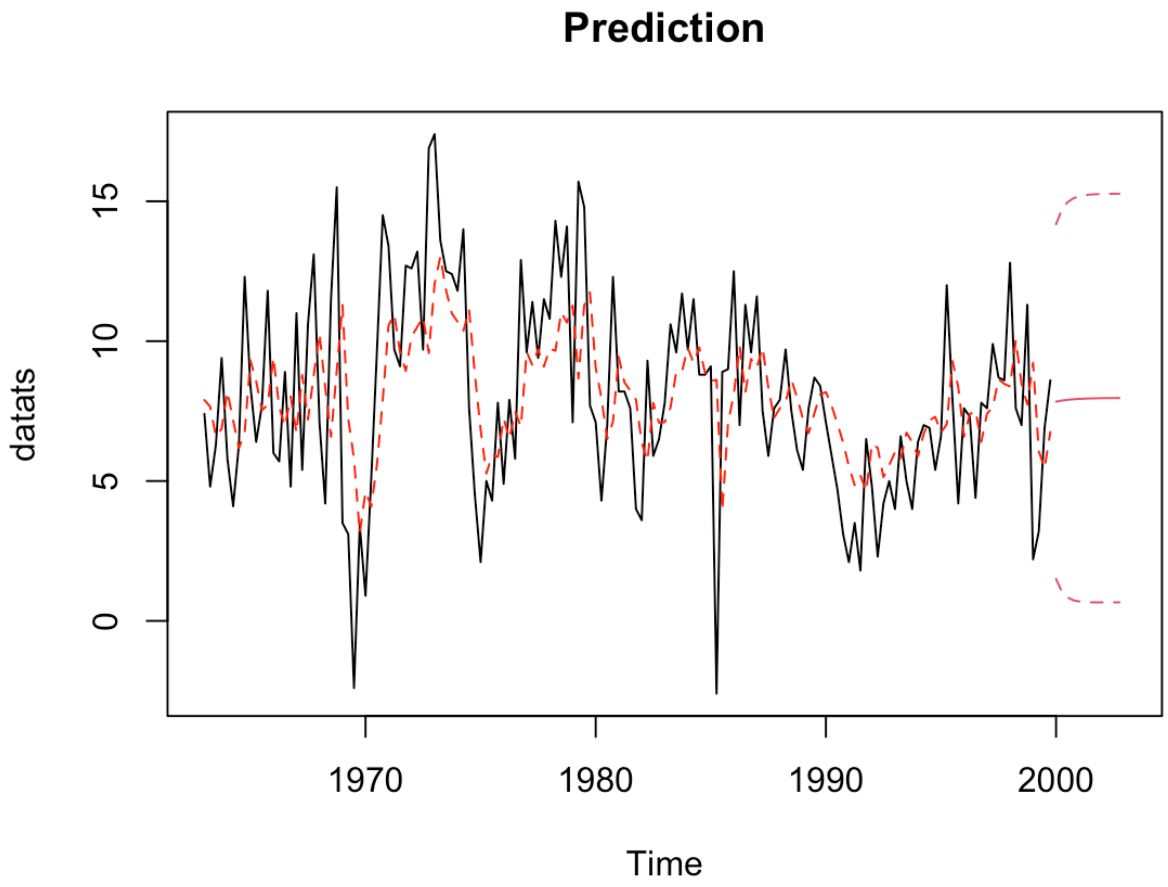
We plot the data to see if its normally distributed:
gghistogram(r) + ggtitle("Histogram of residuals")



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))
```



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
```

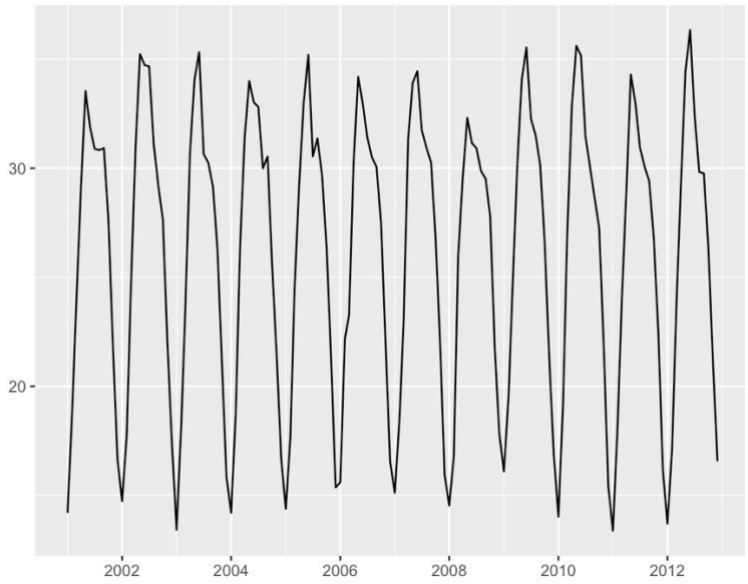
Date	Delhi_Temperature
01/01/01	14.197
01/02/01	18.882
01/03/01	23.918
01/04/01	29.155
01/05/01	33.541
01/06/01	31.87
01/07/01	30.888
01/08/01	30.832
01/09/01	30.92
01/10/01	27.687
01/11/01	21.806

)

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)
```

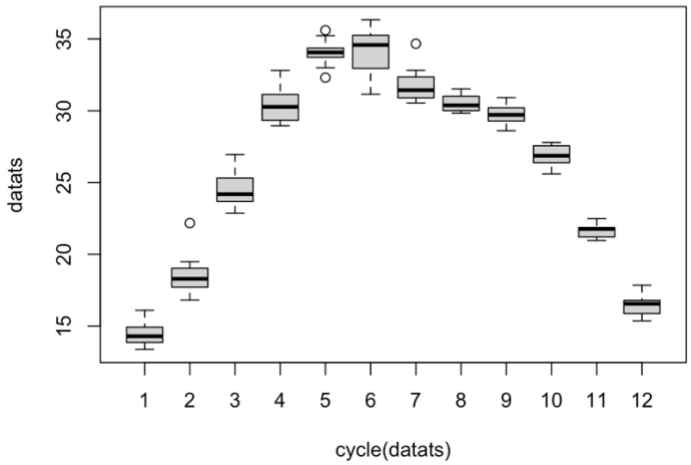


We check for mean and variance of dataset for better understanding :

```
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))
```



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.

$$y_t = c + \beta t + \alpha y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} . . + \phi_p \Delta Y_{t-p} + e_t$$

adf.test(datats)
adf.test(datats, k=2)

```
        Augmented Dickey-Fuller Test

data:  datats
Dickey-Fuller = -12.799, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary
```

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)
```

```
        KPSS Test for Level Stationarity

data:  datats
KPSS Level = 0.010244, Truncation lag parameter = 4, p-value = 0.1
```

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?

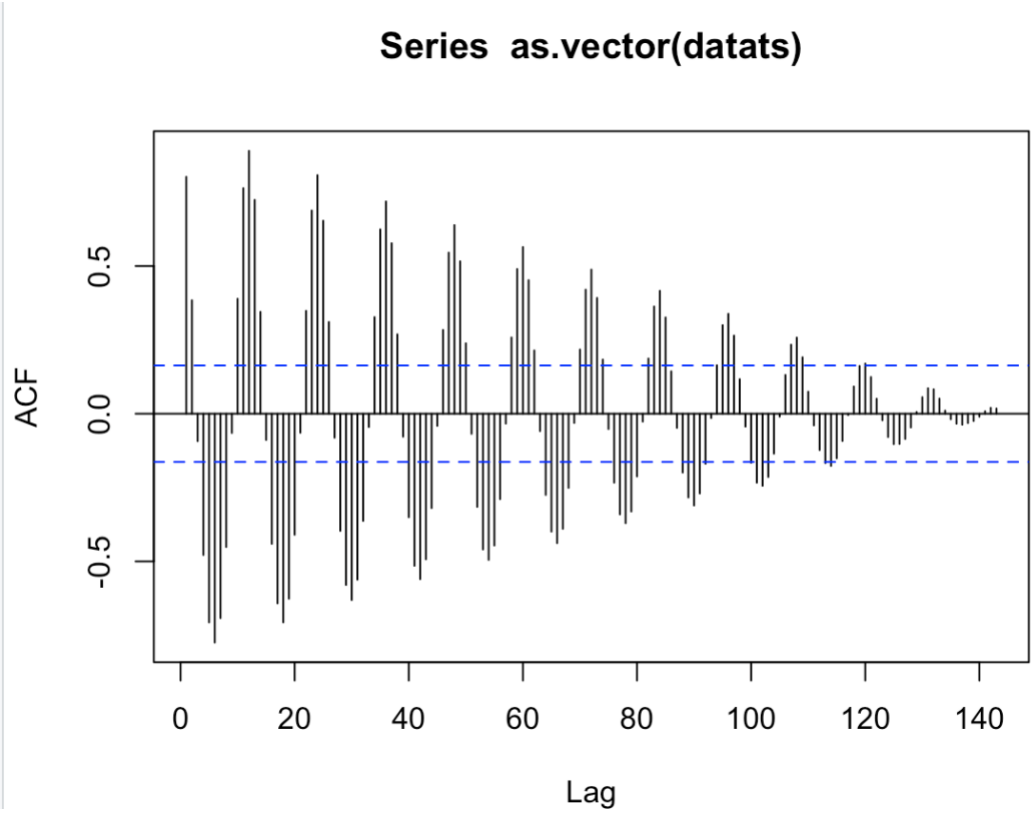
Partial Autocorrelation Function (PACF)

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.

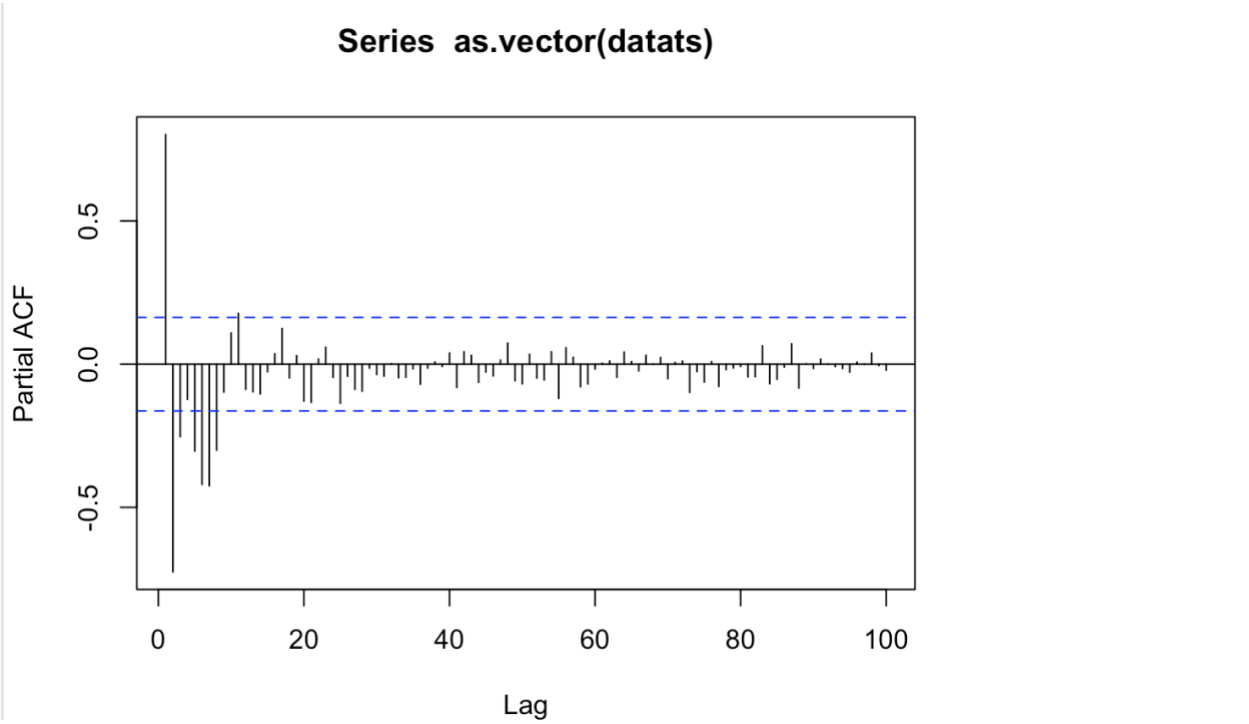
checking ACF AND PACF

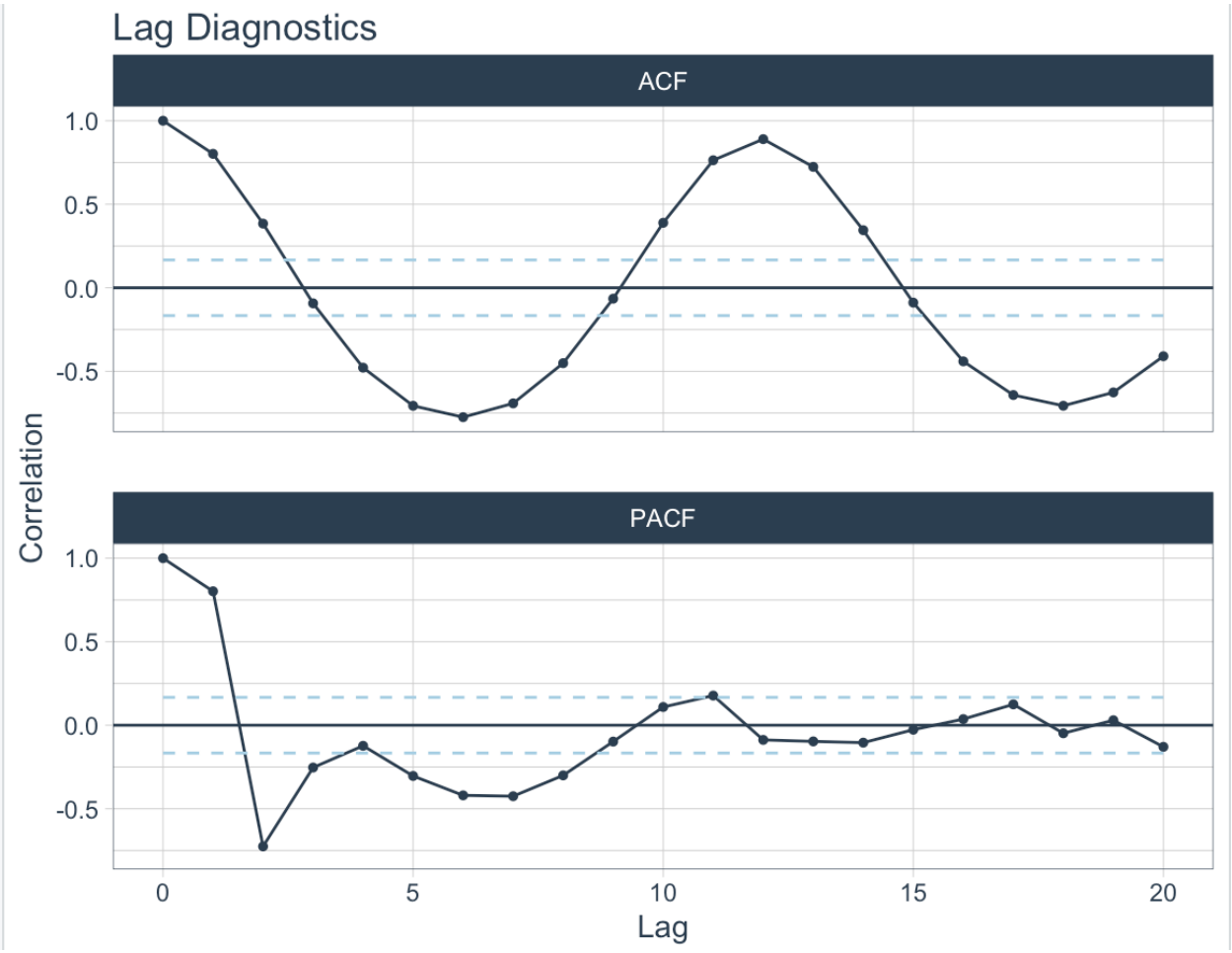
```
acf <- acf(as.vector(datats),plot = FALSE, lag.max = 100)
plot(acf)
```




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

From ACF and PACF we can say that the MA term (i.e q) can be 9 or 10 for non-seasonal part and 9 or 10 for seasonal part. Similarly we can also say AR term(p) can be 2/3/8 for non seasonal part and 0 for seasonal part .


Series as.vector(datats)
Partial ACF
0.5
0.0
-0.5
0
20
40
60
80
100
Lag


Lag Diagnostics
ACF
PACF
Correlation
1.0
0.5
0.0
-0.5
0
5
10
15
20
Lag

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

eacf(as.vector(datats))

```
> eacf(as.vector(datats))
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x o x x x x x o x x  x  x  x
1 x x o x x x x x o x x  x  x  x
2 x o x o o x o o x x x  x  x  o
3 x o x x o x o o o x x  o  x  o
4 x x x o o x o o o x x  o  x  o
5 x x o x o o o o o x x  o  o  o
6 x o x x o o o o o x x  o  o  o
7 x x x x x o o o o x o  o  o  o
```

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.

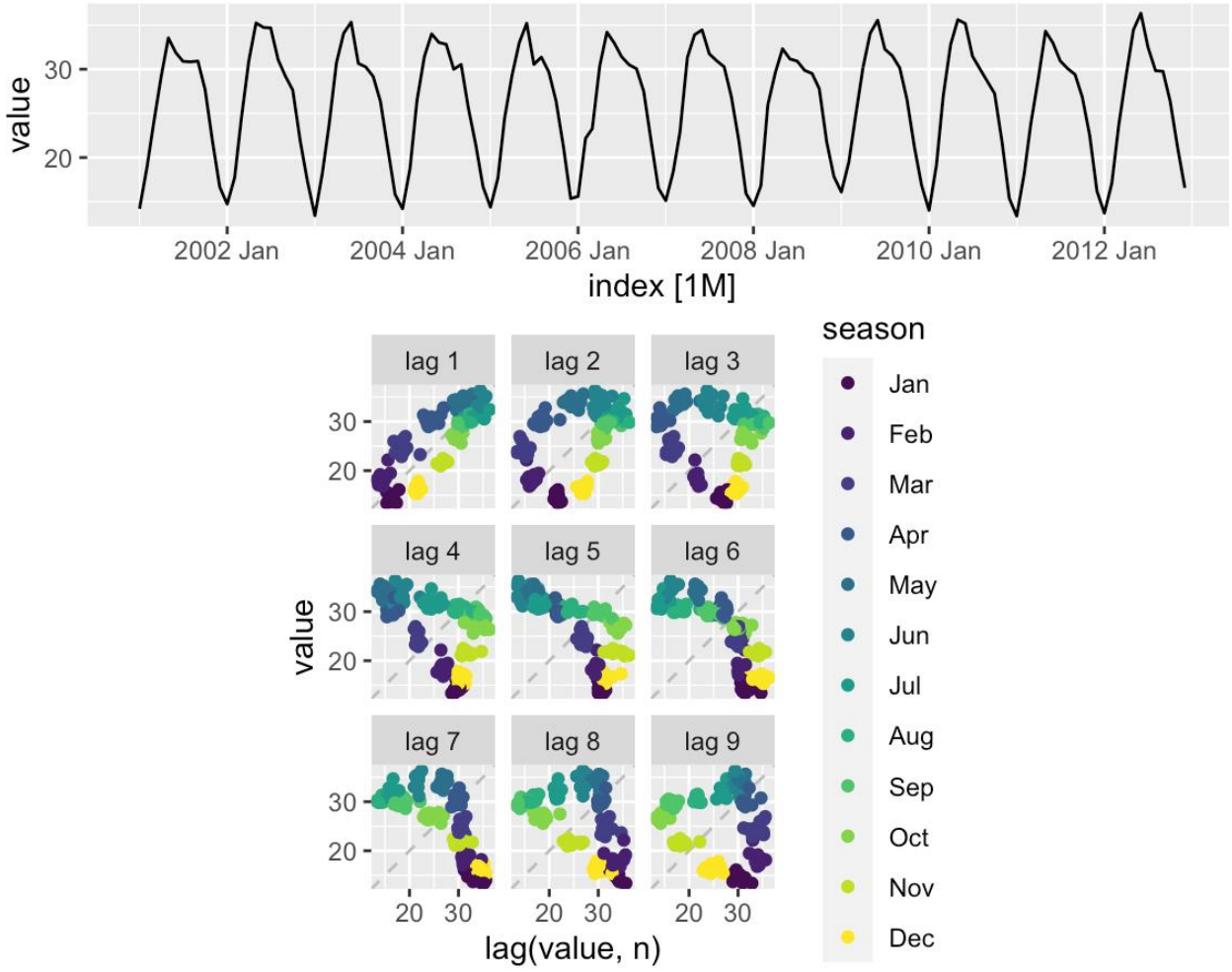
```
# install.packages("rugarch", repos=c("http://rstudio.org/_packages", "http://cran.rstudio.com"))


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.
```

```
*----------------------------------*
*          GARCH Model Fit         *
*----------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model      : sGARCH(1,1)
Mean Model       : ARFIMA(8,0,9)
Distribution     : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error     t value Pr(>|t|)
mu     26.062292    0.010512   2479.2374 0.000000
ar1     0.490504    0.000279   1759.7374 0.000000
ar2     0.286926    0.000259   1106.4852 0.000000
ar3    -0.692293    0.000478  -1448.7950 0.000000
ar4    -0.519762    0.000021 -24614.8031 0.000000
ar5     0.497756    0.000024  20750.0258 0.000000
ar6     0.039357    0.000015   2619.5017 0.000000
ar7    -0.712661    0.000142  -5021.1590 0.000000
ar8    -0.058361    0.000043  -1366.0291 0.000000
ma1    -0.054172    0.000088   -612.3903 0.000000
ma2    -0.294009    0.000342   -858.9777 0.000000
ma3     0.569127    0.000370   1539.7417 0.000000
ma4     0.661987    0.000083   7955.2757 0.000000
ma5    -0.595161    0.000387  -1538.2815 0.000000
ma6    -0.322092    0.000109  -2944.2221 0.000000
ma7     0.505262    0.000012  40449.1291 0.000000
ma8     0.079747    0.000126    632.4815 0.000000
ma9     0.043783    0.000039   1110.1793 0.000000
omega   0.560064    0.141235      3.9655 0.000073
alpha1  0.644277    0.186538      3.4539 0.000553
beta1   0.001415    0.000325      4.3604 0.000013

Robust Standard Errors:
        Estimate  Std. Error     t value Pr(>|t|)
mu     26.062292    0.090757   287.16443  0.00000
ar1     0.490504    0.002428   202.03362  0.00000
ar2     0.286926    0.002458   116.75387  0.00000
ar3    -0.692293    0.004453  -155.46017  0.00000
ar4    -0.519762    0.000116 -4463.03795  0.00000
ar5     0.497756    0.000193  2585.61381  0.00000
ar6     0.039357    0.000022  1790.64078  0.00000
ar7    -0.712661    0.001242  -573.86762  0.00000
```

```
ar7    -0.712661    0.001242  -573.86762  0.00000
ar8    -0.058361    0.000351  -166.05718  0.00000
ma1    -0.054172    0.001573   -34.43135  0.00000
ma2    -0.294009    0.003527   -83.36693  0.00000
ma3     0.569127    0.004188   135.90599  0.00000
ma4     0.661987    0.001368   483.93314  0.00000
ma5    -0.595161    0.003400  -175.02211  0.00000
ma6    -0.322092    0.000812  -396.78111  0.00000
ma7     0.505262    0.000079  6413.49556  0.00000
ma8     0.079747    0.001343    59.38080  0.00000
ma9     0.043783    0.000053   821.16407  0.00000
omega   0.560064    1.806937     0.30995  0.75660
alpha1  0.644277    3.024425     0.21302  0.83131
beta1   0.001415    0.001378     1.02698  0.30443

LogLikelihood : -255.8083

Information Criteria
------------------------------------

Akaike       3.8446
Bayes        4.2777
Shibata      3.8088
Hannan-Quinn 4.0205

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                         statistic   p-value
Lag[1]                       11.51 0.0006936
Lag[2*(p+q)+(p+q)-1][50]     37.89 0.0000000
Lag[4*(p+q)+(p+q)-1][84]     48.03 0.1262023
d.o.f=17
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic   p-value
Lag[1]                      15.84 6.905e-05
Lag[2*(p+q)+(p+q)-1][5]     24.84 8.824e-07
Lag[4*(p+q)+(p+q)-1][9]     33.21 5.697e-08
d.o.f=2
```

#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 :

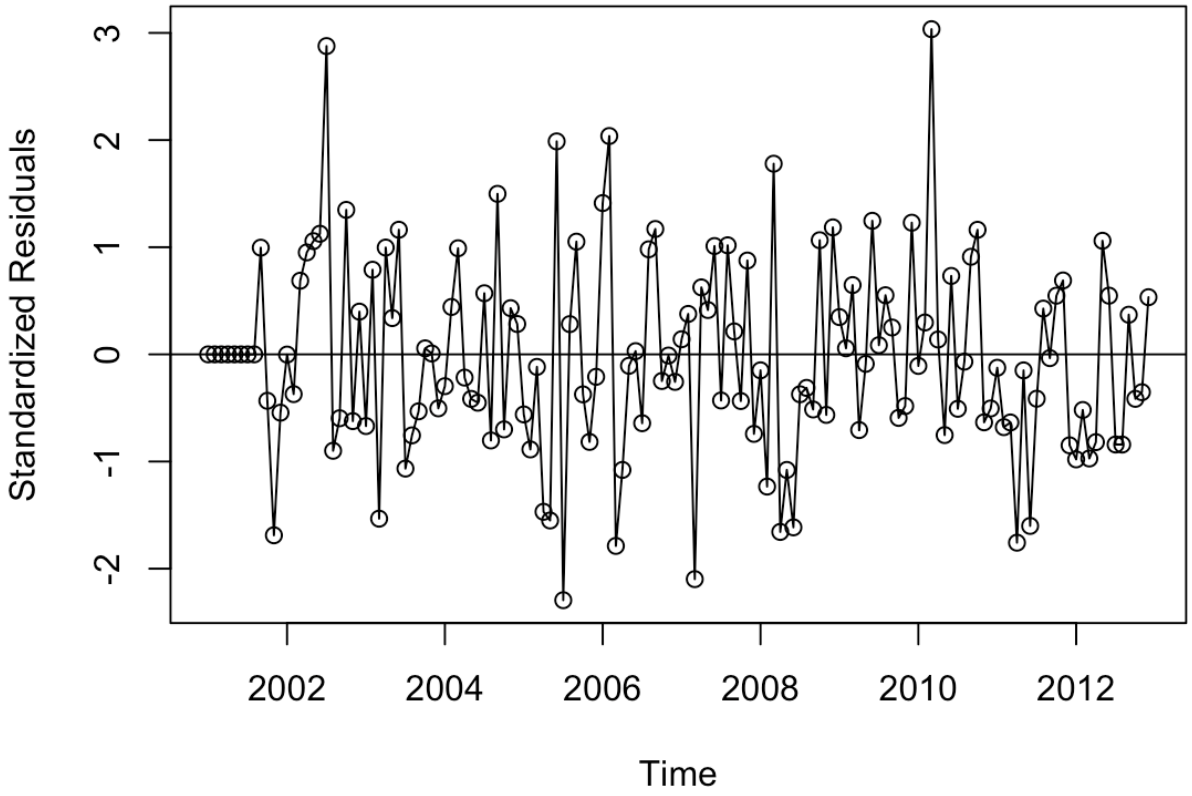


Checking Residuals :

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)

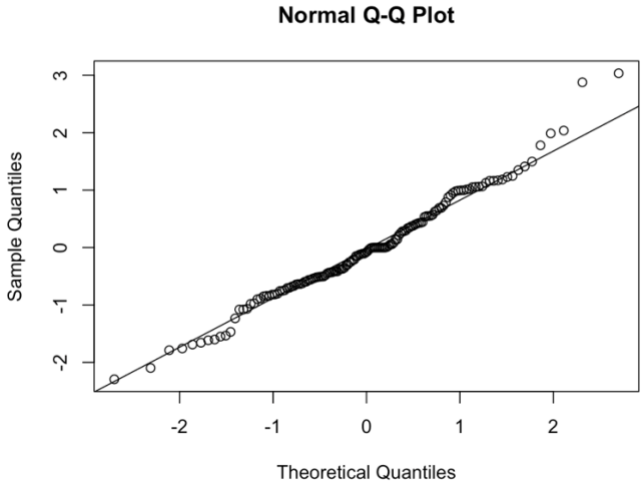
plot(r,ylab ='Standardized Residuals',type='o');
abline(h=0)



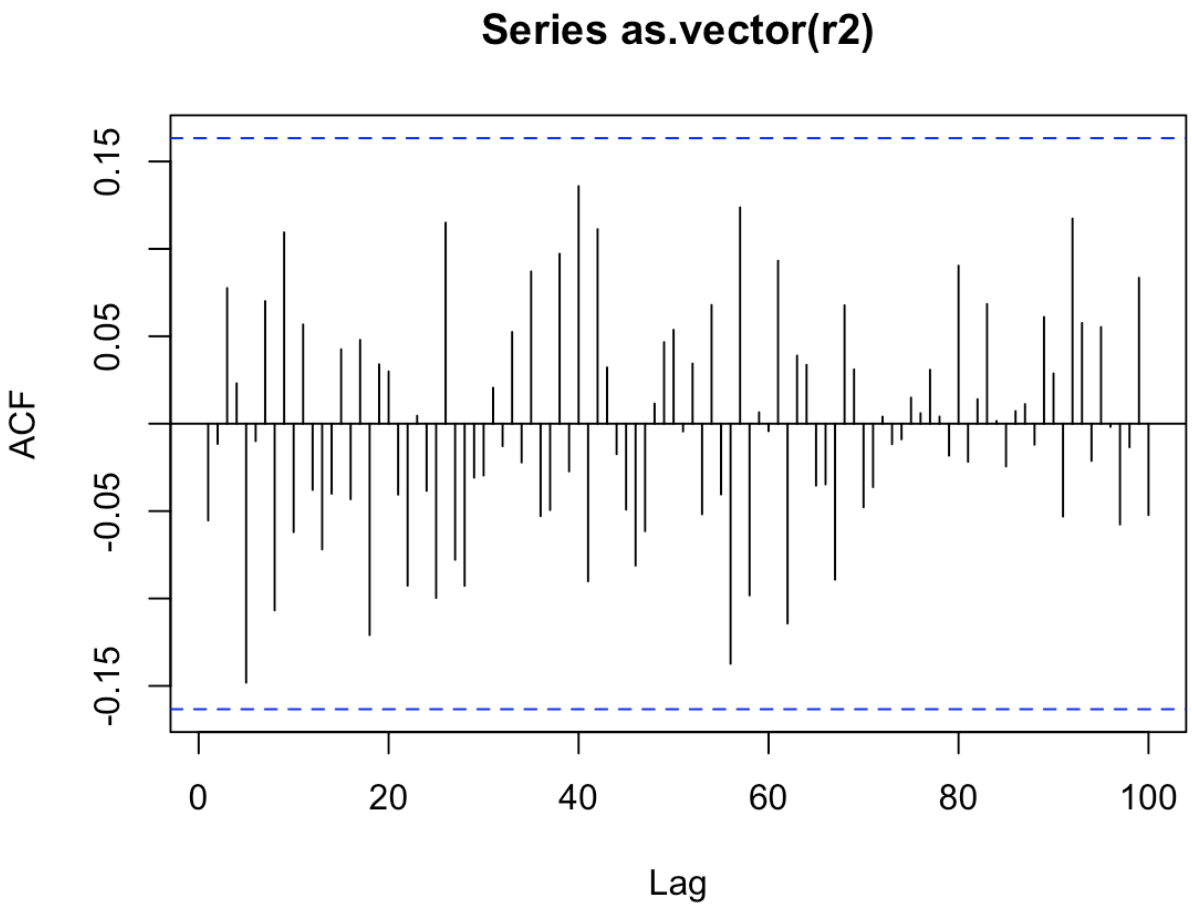
Normality Test on Residuals :

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

```
r
r2<- na.omit(r)
acf(as.vector(r2), lag.max = 100)
```



ACF of residuals :



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:

$$W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n} (x_i - \bar{x})}$$

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)

```
        Shapiro-Wilk normality test

data:  datats
W = 0.91873, p-value = 2.816e-07
```

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 Ljung-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:

H_0: The residuals are independently distributed.

H_A: 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

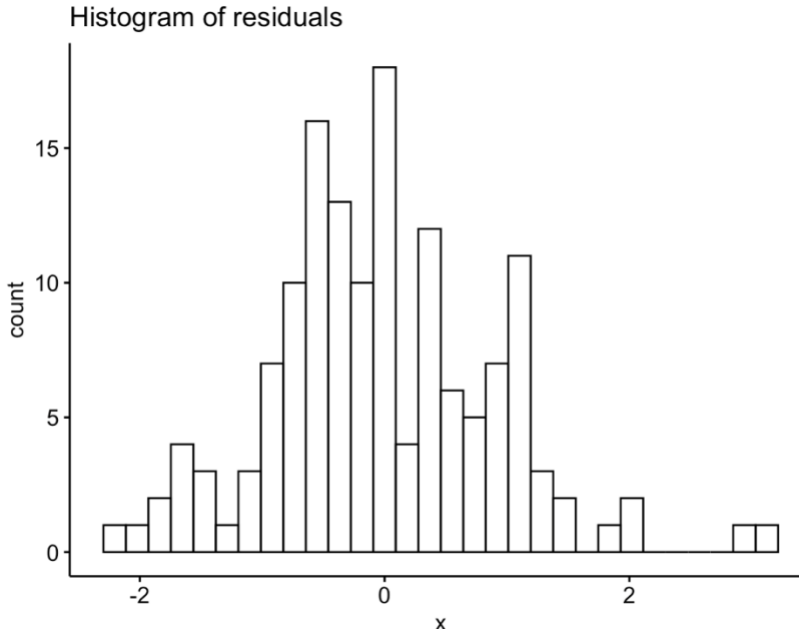
H1 : The series exhibits serial correlation

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

```
        Box-Ljung test

data:  datats
X-squared = 448.39, df = 10, p-value < 2.2e-16
```

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")



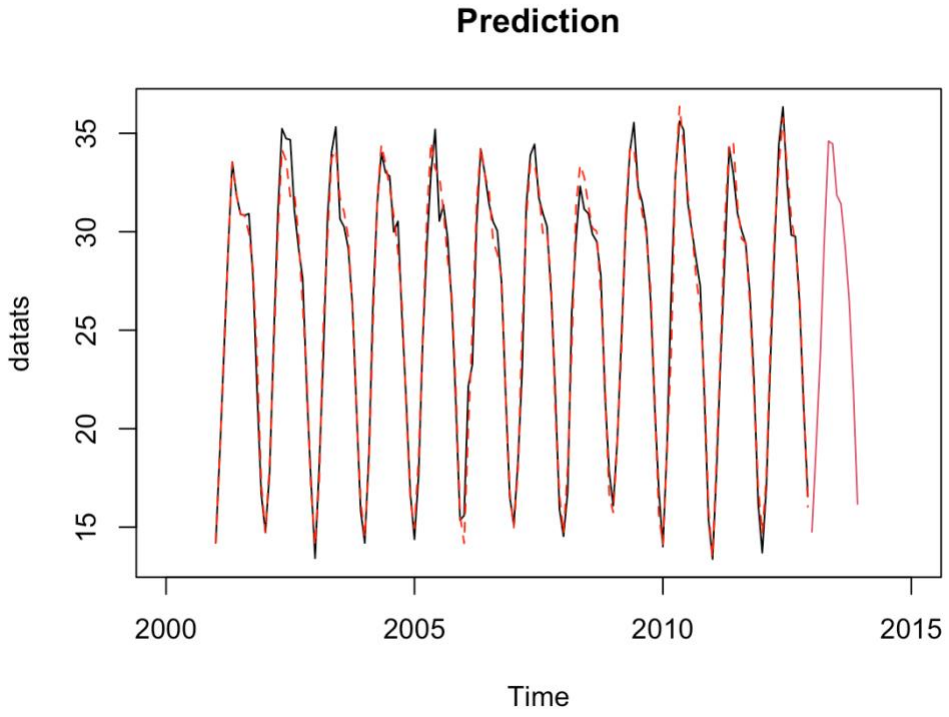
Forecasting :

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)
```



Conclusion :

Forecast for next 12 months 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
```