CENTRAL UNIVERSITY OF RAJASTHAN

SCHOOL OF MATHEMATICS, STATISITCS AND COMPUTATION SCIENCES

DEPARTMENT OF STATISTICS

COURSE TITLE: Time Series Analysis and Forecasting

ENROLLEMENT NUMBER: 2018MSTA020

SUBMITTED TO: Dr. Jitendra Kumar

NAME OF STUDENT: Prabhat Kumar Trivedi

Time Series Analysis in R

About the Data:

The data is monthly temperature recorded for the Indian nation during the year 1950-2017. Temperatures values are recorded in Celsius. Total observations are 816.

The source of the data i.e. it is collected from an Indian Government Website.

URL: https://data.gov.in/catalog/all-india-seasonal-and-annual-minmax-temperature-series?filters%5Bfield_catalog_reference%5D=349321&format=json&offset=0&limit=6&sort%5Bcreated%5D=desc

Analysis Carried Out:

I have used time series forecasting models in R to analyze temperature of India during many years.

First load packages that are needed to analyze data and forecast model

```
library(tseries)
library(forecast)
library(TSA)
```

Load temperature recorded dataset

```
data<-read.csv("E:/R/3_Sem/Time_series_project/temperatures.csv",header = T)
```

The data is in 'csv' format. So we structure the data in time series format so that the analysis become easier.

```
data_trans<-as.data.frame(t(data))
data_trans<-data_trans[-1,]
data1<-as.vector(as.matrix(data_trans))
data.ts<-ts(data1, frequency = 12, start = c(1950,1), end = c(2017,12))
is.ts(data.ts)
```

```
[1] TRUE
```

Now 'data.ts' is our desired time series data of recorded temperature of India during the year 1950-2017.

```
head(data.ts)
```

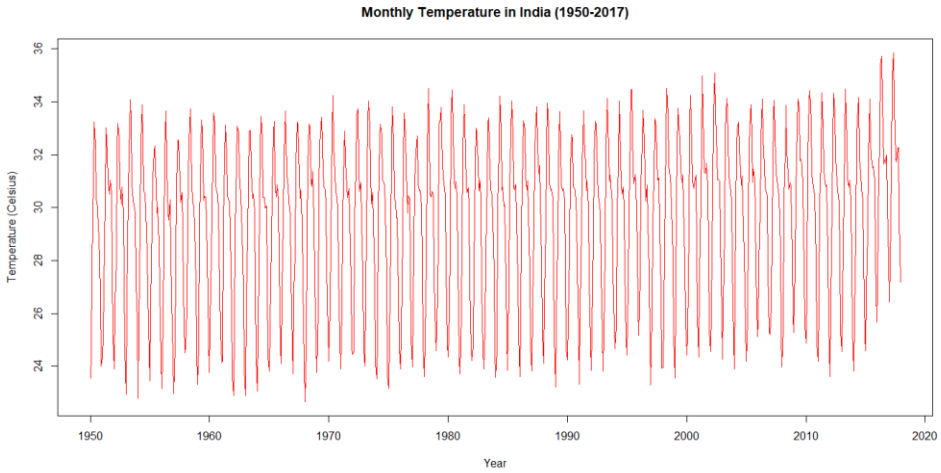
```
      Jan   Feb   Mar   Apr   May   Jun
1950 23.56 24.48 27.78 31.16 33.22 32.35
```

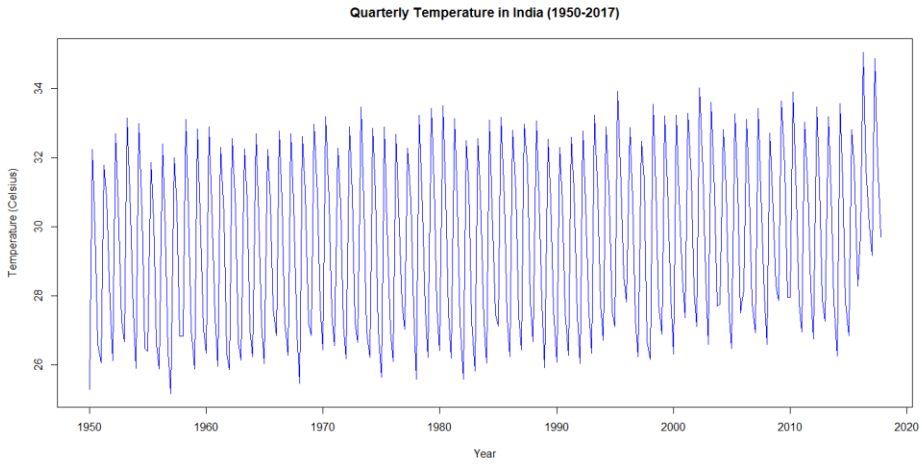
Time Series Plots

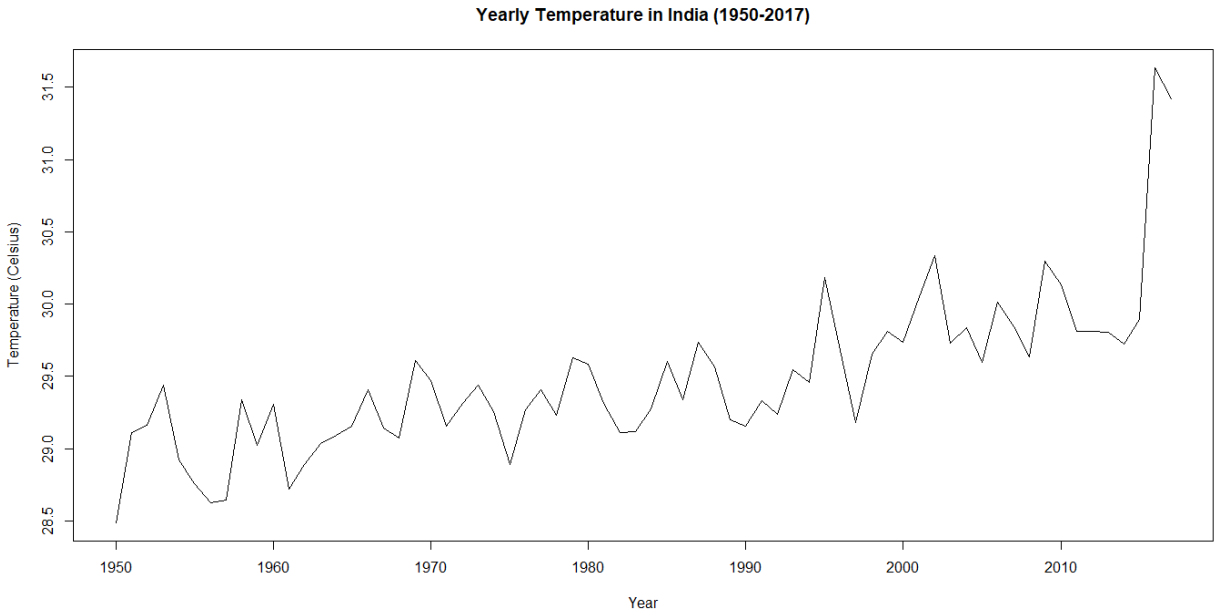
The first thing to do with any time series analysis is to plot the charts.

It is also a good idea to aggregate monthly recorded temperature into quarterly and yearly volume.

```
#Time Series Plots
data.ts.qtr <- aggregate(data.ts, FUN="mean",nfrequency=4)
data.ts.yr <- aggregate(data.ts,FUN="mean", nfrequency=1)
plot.ts(data.ts, col="red",main = "Monthly Temperature in India (1950-2017)", xlab = "Year", ylab = "Temperature (Celsius)")
plot.ts(data.ts.qtr,col="blue", main = "Quarterly Temperature in India (1950-2017)", xlab = "Year", ylab = "Temperature (Celsius)")
plot.ts(data.ts.yr, main = "Yearly Temperature in India (1950-2017)", xlab = "Year", ylab = "Temperature (Celsius)")
```



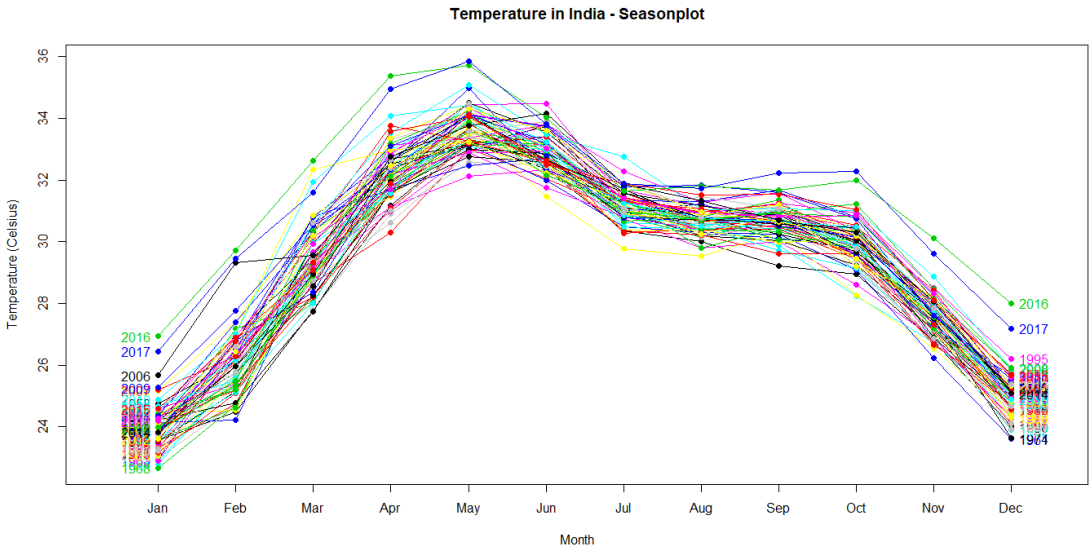


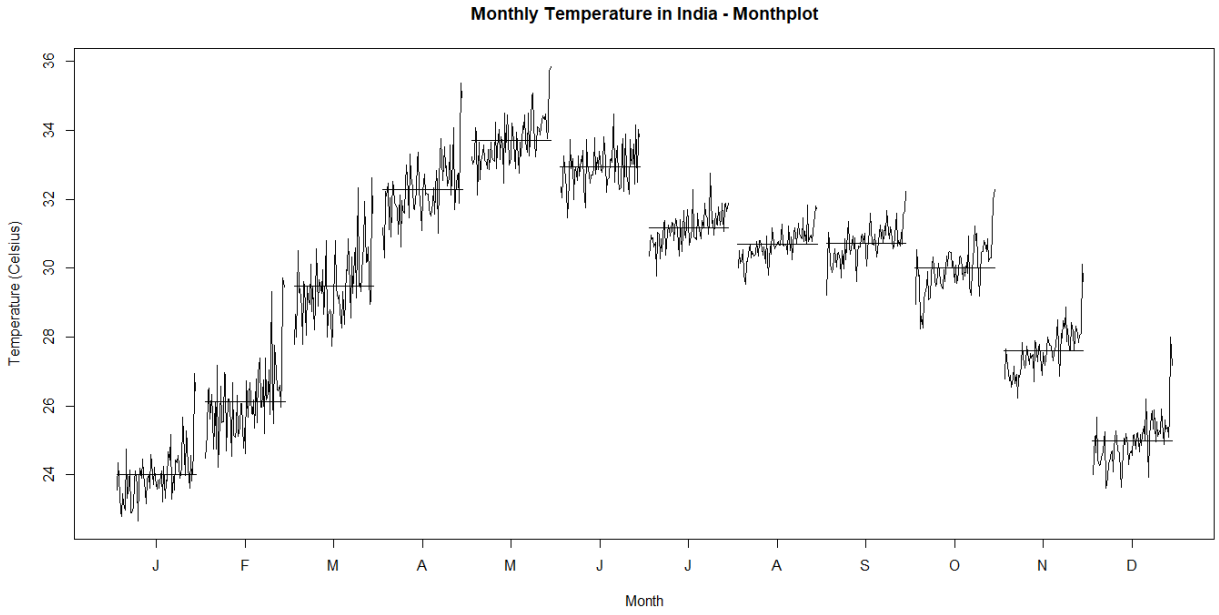


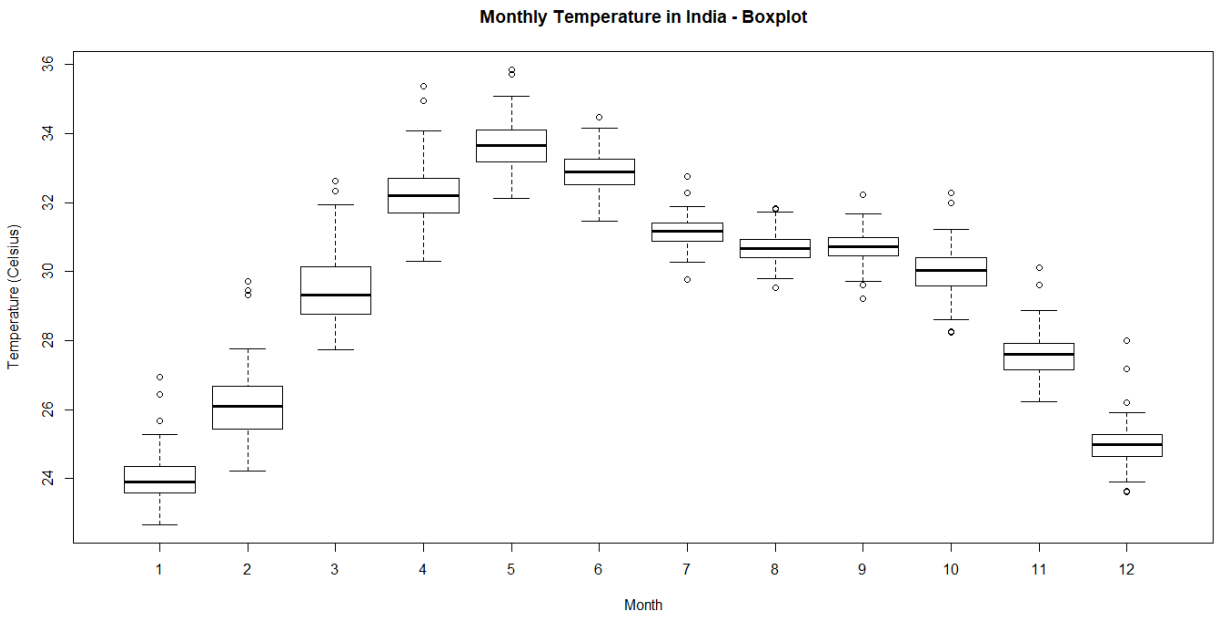
From the above plots we can observe that the data was slowly increasing with the time and during the interval of every five years there was fluctuations. After the year 2010 onwards there was a drastic increase in the temperature.

Next step we want to take a look at seasonality in more detail.

```
#Seasonality
seasonplot(data.ts, year.labels = TRUE, year.labels.left=TRUE, col=1:40, pch=19, main =
           "Temperature in India - Seasonplot", xlab = "Month", ylab = "Temperature (Celsius)")
monthplot(data.ts, main = "Monthly Temperature in India - Monthplot", xlab = "Month", ylab
= "Temperature (Celsius)")
boxplot(data.ts ~ cycle(data.ts), xlab = "Month", ylab = "Temperature (Celsius)", main =
"Monthly Temperature in India - Boxplot")
```


Temperature in India - Seasonplot
Temperature (Celsius)
Month
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
2016
2017
2006
1988
1995
1974


Monthly Temperature in India - Monthplot
Temperature (Celsius)
Month
J
F
M
A
M
J
J
A
S
O
N
D

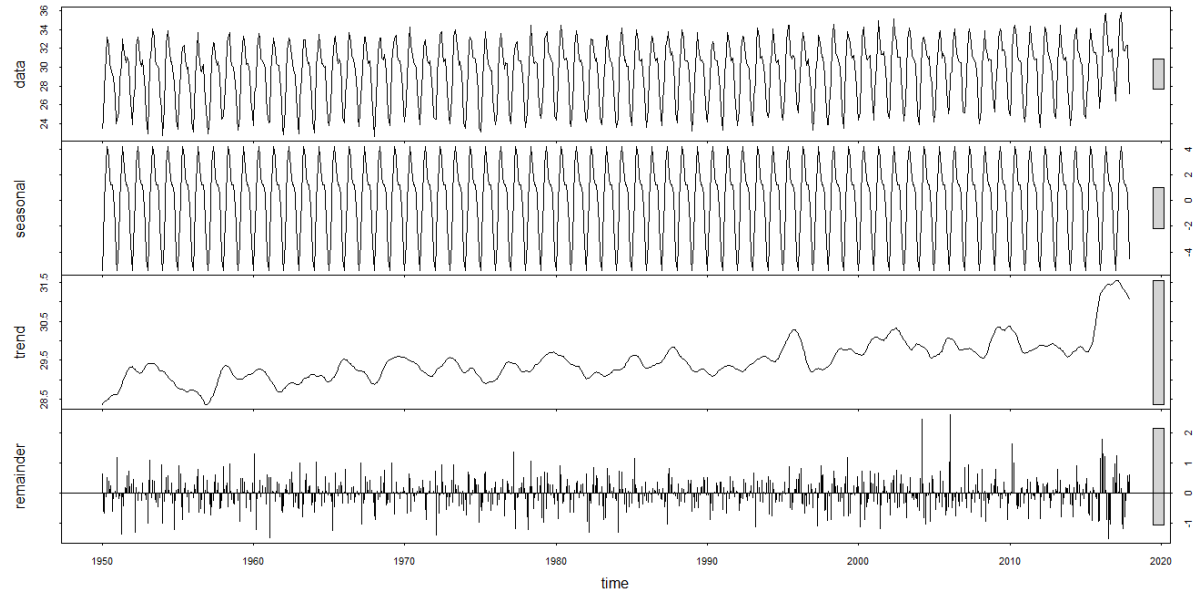


From the above three plots we can easily observe that temperature changes during the season (i.e. month wise) which is actually true in practical. All plots give us the level among different months, but also the range and variation. Temperature starts increasing from Feb and was maximum in the month May and then starts decreasing. During the months July-Oct temperature remains a bit consistent.

Decomposition

After just looking at the different plots, we normally can get a good sense on how the time series behaves and the different components within the data. Decomposition is a tool that we can separate different components in a time series data so we can see trend, seasonality, and random noises individually.

```
#Decomposition
plot(stl(data.ts, s.window="periodic"))
```



Seasonality refers to periodic fluctuations.

From this chart, we can see that the seasonality is strong but consistent. The trend is similar to what we saw when we aggerated the data into yearly, there was a slow growth during the years1950-2010 and after that, the temperature increased rapidly. Also, the noise was consistent throughout each year.

Descriptive Statistics

Mean

```
data.ts.yr <- aggregate(data.ts,FUN="mean", nfrequency=1)

mean<-as.data.frame(data.ts.yr)
```

Year	Annual Mean	Year	Annual Mean	Year	Annual Mean	Year	Annual Mean
1950	28.49	1967	29.14	1984	29.28	2001	30.05
1951	29.11	1968	29.07	1985	29.61	2002	30.33
1952	29.16	1969	29.61	1986	29.34	2003	29.73
1953	29.44	1970	29.47	1987	29.74	2004	29.84
1954	28.92	1971	29.15	1988	29.57	2005	29.6
1955	28.76	1972	29.32	1989	29.2	2006	30.01
1956	28.62	1973	29.44	1990	29.16	2007	29.84
1957	28.65	1974	29.26	1991	29.33	2008	29.64
1958	29.34	1975	28.89	1992	29.24	2009	30.3
1959	29.03	1976	29.27	1993	29.55	2010	30.13
1960	29.31	1977	29.41	1994	29.46	2011	29.81
1961	28.72	1978	29.23	1995	30.19	2012	29.81
1962	28.89	1979	29.63	1996	29.68	2013	29.81
1963	29.04	1980	29.58	1997	29.18	2014	29.72
1964	29.09	1981	29.32	1998	29.66	2015	29.9
1965	29.16	1982	29.11	1999	29.81	2016	31.63
1966	29.41	1983	29.12	2000	29.74	2017	31.42

Variance

```
data.ts.yr1 <- aggregate(data.ts,FUN="var", nfrequency=1)

variance<-as.data.frame(data.ts.yr1)
```

Year	Annual Variance	Year	Annual Variance	Year	Annual Variance	Year	Annual Variance
1950	10.44	1967	10.18	1984	11.57	2001	9.72
1951	8.43	1968	12.08	1985	10.09	2002	11.13
1952	9.50	1969	9.92	1986	10.43	2003	10.80
1953	10.67	1970	10.07	1987	10.53	2004	9.38
1954	11.94	1971	8.86	1988	9.43	2005	10.69
1955	8.56	1972	11.07	1989	10.35	2006	7.42
1956	10.16	1973	11.26	1990	8.43	2007	9.30
1957	10.53	1974	12.08	1991	10.43	2008	9.37
1958	9.60	1975	11.28	1992	9.94	2009	9.30
1959	11.12	1976	9.80	1993	10.06	2010	10.73
1960	9.52	1977	8.64	1994	9.56	2011	9.38
1961	11.06	1978	11.52	1995	9.96	2012	10.98
1962	11.46	1979	11.19	1996	8.19	2013	9.37
1963	10.40	1980	11.41	1997	11.67	2014	11.56
1964	11.02	1981	10.95	1998	11.33	2015	8.89
1965	9.57	1982	10.75	1999	10.00	2016	7.16
1966	9.22	1983	10.34	2000	10.02	2017	8.12

From the mean and variance table we can observe that the mean and variance values are approximately same for different years. Hence the stationarity is present in our time series data.

Stationarity

Stationarity is an important characteristic of time series. A time series is said to be stationary if its statistical properties do not change over time. In other words, it has constant mean and variance, and covariance is independent of time.

I have used ADF test to check the stationarity of data in R.

```
#Stationarity

adf.test(data.ts)
```

```
        Augmented Dickey-Fuller Test

data:  data.ts
Dickey-Fuller = -9.1816, Lag order = 9, p-value = 0.01
alternative hypothesis: stationary
```

The p-value (0.01) < 0.05 hence we reject the null hypothesis, i.e., the series is non-stationary. Hence, the time series data is stationary.

```
ndiffs(data.ts,test = "adf")
[1] 0
```

'ndiffs' is a function in R to estimate the number of differences required to make a given time series stationary. The output is zero, that is we don't have to difference our time series data since it is already stationary.

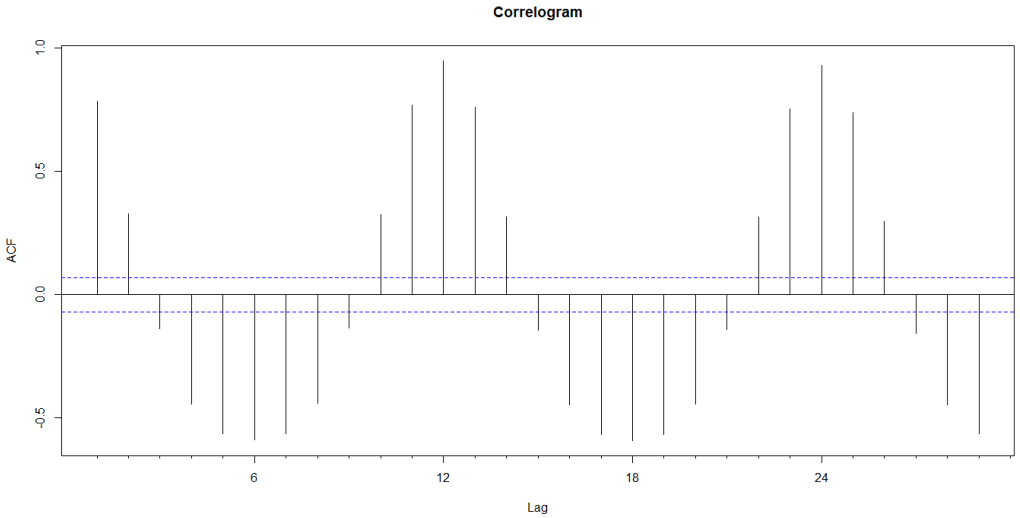
ACF, PACF and ACVF plots

Autocorrelation and partial autocorrelation plots are heavily used in time series analysis and forecasting. These are plots that graphically summarize the strength of a relationship with an observation in a time series with observations at prior time steps.

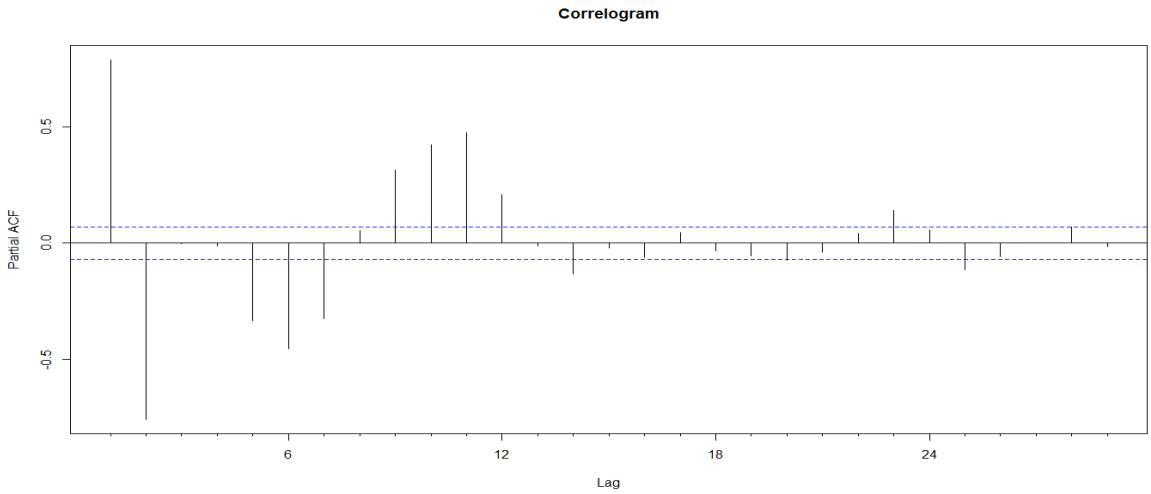
ACF describes how well the present value of the series is related with its past values.

PACF is a partial auto-correlation function. Basically, instead of finding correlations of present with lags like ACF, it finds correlation of the residuals (which remains after removing the effects which are already explained by the earlier lag(s)) with the next lag value hence 'partial' and not 'complete' as we remove already found variations before we find the next correlation.

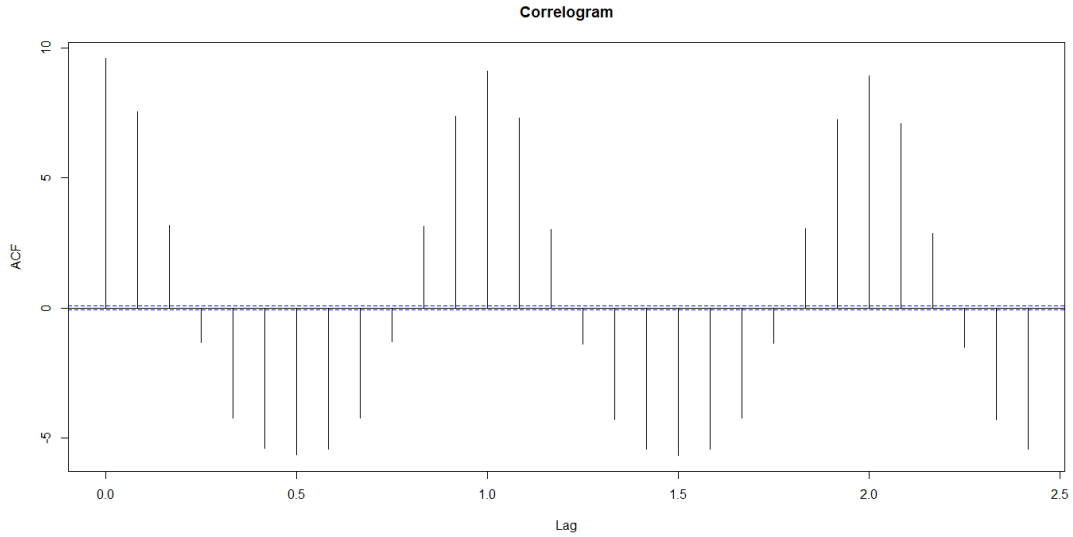
```
Acf(data.ts,main="Correlogram")                    #Auto-correlation function
Pacf(data.ts,main="Correlogram")                   #Partial auto-correlation function
acf(data.ts,type = "covariance",main="Correlogram")  #Auto-covariance function
```



Lag	ACF	Lag	ACF
0	1	15	-0.14501
1	0.786195	16	-0.448
2	0.329678	17	-0.56699
3	-0.13889	18	-0.59074
4	-0.44281	19	-0.56669
5	-0.56376	20	-0.44336
6	-0.58724	21	-0.14084
7	-0.56537	22	0.317127
8	-0.44152	23	0.753325
9	-0.13601	24	0.930431
10	0.325622	25	0.738583
11	0.768562	26	0.298528
12	0.948411	27	-0.15632
13	0.760227	28	-0.44744
14	0.314953	29	-0.56485



Lag	PACF	Lag	PACF
1	0.786195	15	-0.02013899
2	-0.75524	16	-0.06156074
3	-0.00041	17	0.044100874
4	-0.0126	18	-0.03171911
5	-0.3332	19	-0.05412129
6	-0.45364	20	-0.07280276
7	-0.32449	21	-0.03885196
8	0.054047	22	0.040793612
9	0.312215	23	0.14168232
10	0.422638	24	0.057399741
11	0.475767	25	-0.11277329
12	0.208289	26	-0.05709505
13	-0.01018	27	-7.51348E-05
14	-0.13326	28	0.068256438



Lag	ACVF	Lag	ACVF
0.000	9.585	1.250	-1.390
0.083	7.536	1.333	-4.294
0.167	3.160	1.417	-5.434
0.250	-1.331	1.500	-5.662
0.333	-4.244	1.583	-5.432
0.417	-5.403	1.667	-4.249
0.500	-5.629	1.750	-1.350
0.583	-5.419	1.833	3.040
0.667	-4.232	1.917	7.220
0.750	-1.304	2.000	8.918
0.833	3.121	2.083	7.079
0.917	7.366	2.167	2.861
1.000	9.090	2.250	-1.498
1.083	7.287	2.333	-4.289
1.167	3.019	2.417	-5.414

The above three correlograms are of ACF (Auto-correlation function), PACF (Partial autocorrelation function) and ACVF (Auto covariance function) and we can observe that correlogram of ACF and ACVF is sinusoidal and correlogram of PACF spike at lag 2.

ARMA Model

ARMA model is simply the merger between AR(p) and MA(q) models.

For modelling the data, I have created a data frame to store the models along with their AIC (Akaike's Information Criteria) and BIC (Bayesian's Information Criteria) resp., and chose that model having minimum AIC and BIC.

```
t<-data.frame(matrix(NA, ncol = 3))
for(i in 0:5)
{
  for (j in 0:5)
  {
    a<-arima(data.ts,order=c(i,0,j),method="ML")
    t<-rbind(t,c(paste("ARMA(",i,",",j,")"),a$aic,BIC(a)))
  }


}
t<-t[-1,]
```

	Model	AIC	BIC		Model	AIC	BIC
1	ARMA(0 , 0)	4162.01	4173.42	19	ARMA(3 , 0)	2661.01	2686.53
2	ARMA(0 , 1)	3390.58	3406.69	20	ARMA(3 , 1)	2661.70	2691.93
3	ARMA(0 , 2)	2975.72	2996.54	21	ARMA(3 , 2)	2650.22	2685.15
4	ARMA(0 , 3)	2777.05	2802.57	22	ARMA(3 , 3)	2645.14	2684.77
5	ARMA(0 , 4)	2718.68	2748.91	23	ARMA(3 , 4)	2564.67	2609.01
6	ARMA(0 , 5)	2695.92	2730.85	24	ARMA(3 , 5)	2564.50	2613.54
7	ARMA(1 , 0)	3372.62	3388.74	25	ARMA(4 , 0)	2662.96	2693.19
8	ARMA(1 , 1)	2983.85	3004.67	26	ARMA(4 , 1)	2659.93	2694.86
9	ARMA(1 , 2)	2815.14	2840.66	27	ARMA(4 , 2)	2269.20	2308.83
10	ARMA(1 , 3)	2727.59	2757.82	28	ARMA(4 , 3)	2067.13	2111.47
11	ARMA(1 , 4)	2696.68	2731.61	29	ARMA(4 , 4)	1801.94	1850.98
12	ARMA(1 , 5)	2634.81	2674.45	30	ARMA(4 , 5)	1674.79	1728.54
13	ARMA(2 , 0)	2659.11	2679.92	31	ARMA(5 , 0)	2558.00	2592.93
14	ARMA(2 , 1)	2661.01	2686.53	32	ARMA(5 , 1)	2570.41	2610.04
15	ARMA(2 , 2)	2648.27	2678.49	33	ARMA(5 , 2)	2012.51	2056.85
16	ARMA(2 , 3)	2643.23	2678.16	34	ARMA(5 , 3)	2303.76	2352.81
17	ARMA(2 , 4)	2562.97	2602.61	35	ARMA(5 , 4)	1638.75	1692.50
18	ARMA(2 , 5)	2565.83	2610.17	36	ARMA(5 , 5)	1804.30	1862.76

```
t[which.min(t[,2]),]
                X1                   X2                   X3
36 ARMA( 5 , 4 ) 1638.75242425718 1692.50098216178
t[which.min(t[,3]),]
                X1                   X2                   X3
36 ARMA( 5 , 4 ) 1638.75242425718 1692.50098216178
```

ARMA(5,4) is the best model to forecast.

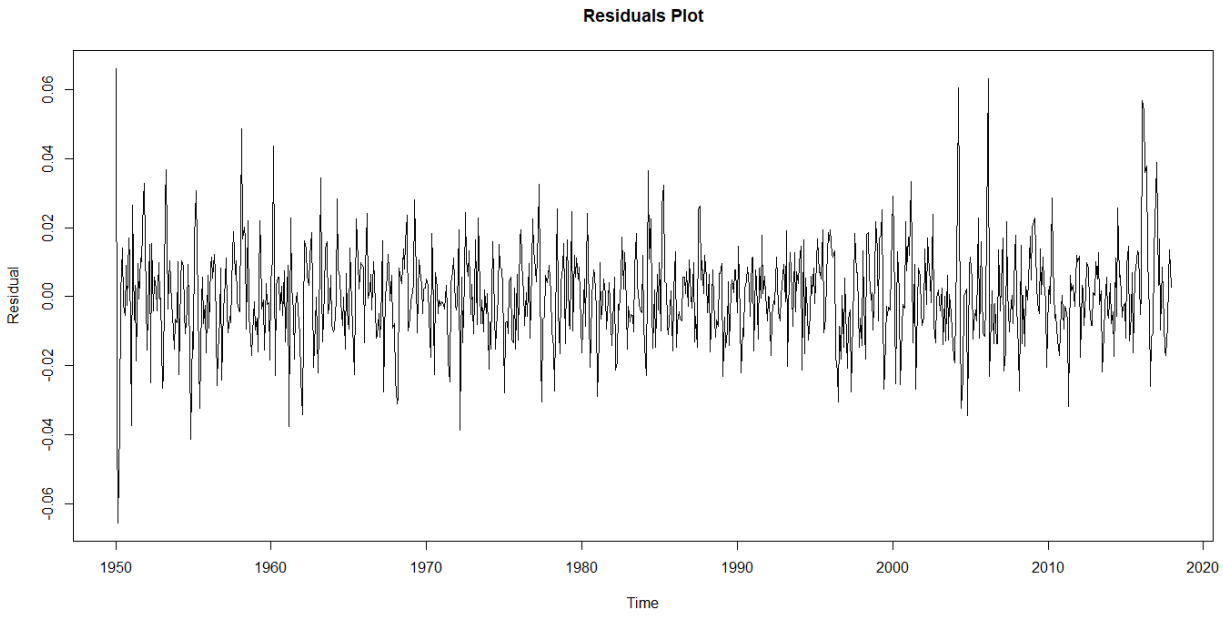
Yule-Walker Estimates

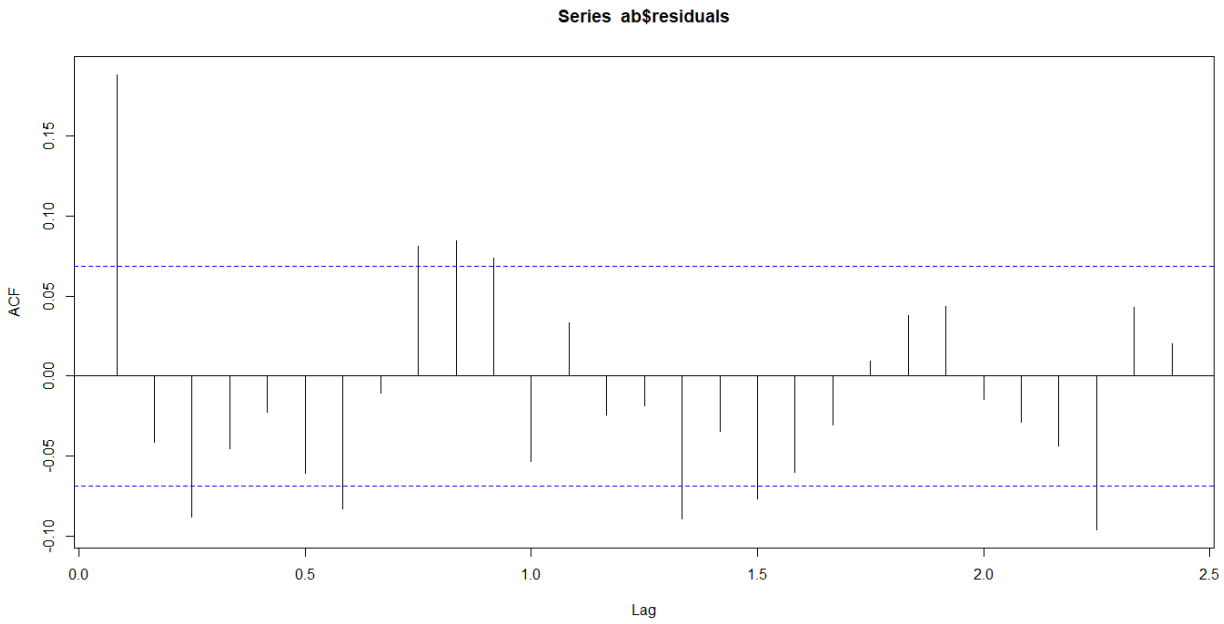
Yule walker equation can be used to estimate AR models from data by first estimating the auto covariance sequence for the data and then using it to solve for the autoregressive parameters.

```
yw<-yw(data.ts,5)  #Yule walker to estimate the AR coefficients.
yw$phi
[1]  1.3754529 -0.7585363 -0.2376447  0.4470918 -0.3331971
```

Forecasting

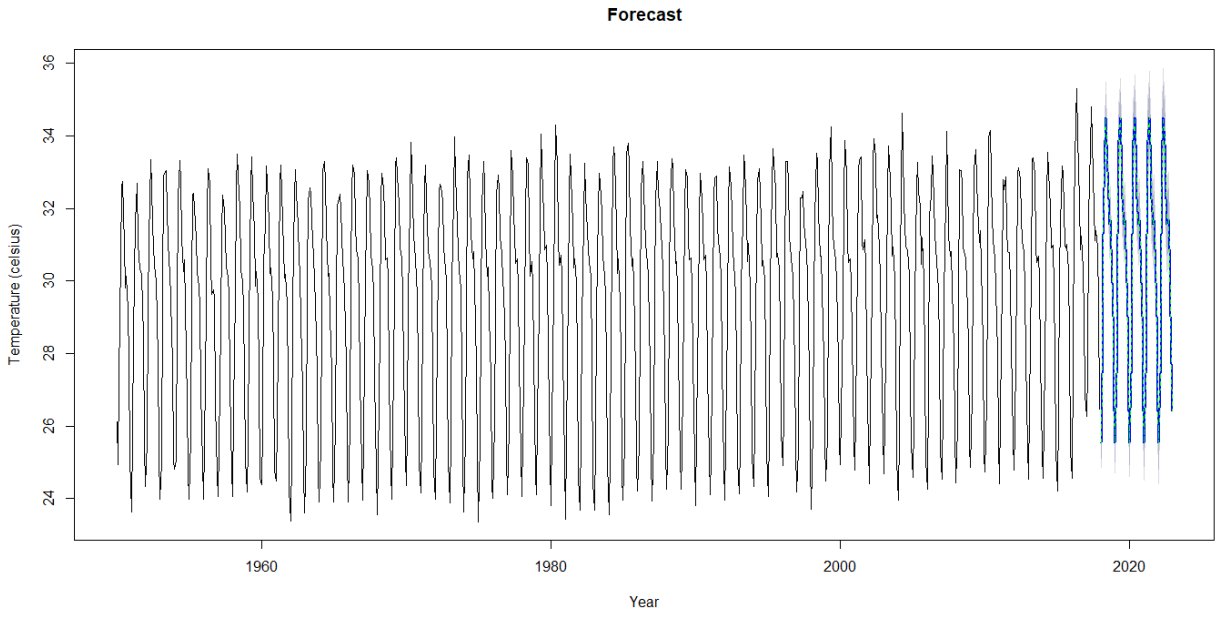
```
a<-arima(data.ts,order=c(4,2,5))
fit<-fitted(a)
ab<-forecast::forecast(fit,h=60)
```





The standardized residual shows no obvious patterns, the residuals behave similar as white noise and the model fits well.

```
plot(ab,main = "Forecast",ylab="Temperature (celsius)",xlab="Year")
lines(ab$mean,col='green',lty=2)
```



The above plot shows the forecasting of 5 years with the green dotted line.

R-Code

```
> rm(list=ls())
> library(tseries)
> library(forecast)
> library(TSA)
> library(itsmr)
> data<-read.csv("E:/R/3_Sem/Time_series_project/temperatures.csv",header = T)
> data<-data[,-(14:18)]
> data_trans<-as.data.frame(t(data))
> data_trans<-data_trans[-1,]
> data1<-as.vector(as.matrix(data_trans))
> data.ts<-ts(data1, frequency = 12, start = c(1950,1), end = c(2017,12))
> is.ts(data.ts)
[1] TRUE
> head(data.ts)
     Jan   Feb   Mar   Apr   May   Jun
1950 23.56 24.48 27.78 31.16 33.22 32.35
> #Time Series Plots
> data.ts.qtr <- aggregate(data.ts, FUN="mean",nfrequency=4)
> data.ts.yr <- aggregate(data.ts,FUN="mean", nfrequency=1)
> plot.ts(data.ts, col="red",main = "Monthly Temperature in India (1950-2017)", xlab = "Year", ylab =
"Temperature (Celsius)")
> plot.ts(data.ts.qtr,col="blue", main = "Quarterly Temperature in India (1950-2017)", xlab = "Year", y
lab = "Temperature (Celsius)")
> plot.ts(data.ts.yr, main = "Yearly Temperature in India (1950-2017)", xlab = "Year", ylab = "Temper
ature (Celsius)")
> #Seasonality
> seasonplot(data.ts, year.labels = TRUE, year.labels.left=TRUE, col=1:40, pch=19, main =
+            "Temperature in India - Seasonplot", xlab = "Month", ylab = "Temperature (Celsius)")
> monthplot(data.ts, main = "Monthly Temperature in India - Monthplot", xlab = "Month", ylab = "Te
mperature (Celsius)")
> boxplot(data.ts ~ cycle(data.ts), xlab = "Month", ylab = "Temperature (Celsius)", main = "Monthly T
emperature in India - Boxplot")
> #Decomposition
> plot(stl(data.ts, s.window="periodic"))
> #Descriptive statistics
> mean<-as.data.frame(data.ts.yr)
> write.csv(mean,file="E:/R/3_Sem/Time_series_project/mean.csv")
> data.ts.yr1 <- aggregate(data.ts,FUN="var", nfrequency=1)
> variance<-as.data.frame(data.ts.yr1)
> write.csv(variance,file="E:/R/3_Sem/Time_series_project/variance.csv")
> #Stationarity
> adf.test(data.ts)

        Augmented Dickey-Fuller Test

data:  data.ts
Dickey-Fuller = -9.1816, Lag order = 9, p-value = 0.01
alternative hypothesis: stationary
> #Acf and Pacf plots
```

```
> Acf(data.ts,main="Correlogram")                    #Auto-correlation function
> Pacf(data.ts,main="Correlogram")                   #Partial auto-correlation function
> acf(data.ts,type = "covariance",main="Correlogram")      #Auto-covariance function
> a1<-Acf(data.ts,plot=F)
> acf<-data.frame(a1$lag,a1$acf)
> a2<-Pacf(data.ts,plot=F)
> pacf<-data.frame(a2$lag,a2$acf)
> a3<-acf(data.ts,type="covariance",plot=F)
> acvf<-data.frame(a3$lag,a3$acf)
> write.csv(acf,file="E:/R/3_Sem/Time_series_project/acf.csv")
> write.csv(pacf,file="E:/R/3_Sem/Time_series_project/pacf.csv")
> write.csv(acvf,file="E:/R/3_Sem/Time_series_project/avcf.csv")
> ndiffs(data.ts,test = "adf")
[1] 0
>
> #ARMA model
> t<-data.frame(matrix(NA, ncol = 3))
> for(i in 0:5)
+ {
+   for (j in 0:5)
+   {
+     a<-arima(data.ts,order=c(i,0,j),method="ML")
+     t<-rbind(t,c(paste("ARMA(",i,",",j,")"),a$aic,BIC(a)))
+   }
+ }
> t<-t[-1,]
> write.csv(t,file="E:/R/3_Sem/Time_series_project/Models.csv")
> t[which.min(t[,2]),]
           X1             X2             X3
36 ARMA( 5 , 4 ) 1638.75242425718 1692.50098216178
> t[which.min(t[,3]),]
           X1             X2             X3
36 ARMA( 5 , 4 ) 1638.75242425718 1692.50098216178
> yw<-yw(data.ts,5)  #Yule walker to estimate the AR coefficients.
> yw$phi
[1]  1.3754529 -0.7585363 -0.2376447  0.4470918 -0.3331971
> a<-arima(data.ts,order=c(5,0,4),method = "ML")
> fit<-fitted(a)
> ab<-forecast::forecast(fit,h=60)
> plot(ab$residuals,main="Residuals Plot",ylab="Residual",xlab="Time")
> plot(acf(ab$residuals))
> plot(ab,main = "Forecast",ylab="Temperature (celsius)",xlab="Year")
> lines(ab$mean,col='green',lty=2)
```

Summary

- Load the packages in R required for time series analysis and forecasting.
- Read the data.
- Structure the data i.e., make it time series data.
- Observed the time series plot monthly, quarterly and annually to visualize the data.
- Checked the seasonality within data by plotting season-plot, month-plot and box-plot.
- Decomposed the time series data to check the trend, stationarity and seasonality.
- Found out descriptive statistics to analyze the data like mean and variance and came to a conclusion that the data is stationary.
- Further to check stationarity we test it using Augmented Dickey Fuller Test (ADF-Test).
- Plot the autocorrelations, partial autocorrelations and autocovariance for the data modelling.
- Found the best model for forecasting the data. In our case ARMA (5,4) fits well in the data and used for forecasting future five years predictions.