Australian Gas Production Analysis

- Vompolu Sai Tanuj

PGP-BABI(June 2019) G1

Table of Contents

1) Project Objective
2) Initial Data Analysis
   a) Invoking the required Packages to be used in R Code
   b) Calling the required dataset into R
   c) Inspection of Time Series Data
   d) Analysing the components of Time Series using Decomposition
3) Splitting of Time Series and finding periodicity
   a) Deseasonalizing the Time Series
   b) Splitting of Time Series into Training and Testing data
   c) Checking the periodicity of the data
4) Using Simple forecast methods
   a) Random Walk Model
   b) Simple Exponential Smoothing
   c) Double Exponential Smoothing
   d) Holt Winter's Model
5) Building Regression models
   a) Checking the time series for stationarity
   b) Converting the series into stationary
   c) Performing Manual ARIMA
   d) Performing Auto ARIMA

e) Model Comparison

f) Box Ljung Test

g) Making final forecasts

6) Project Conclusion

7) Appendix – A (Source Code)

1) Project Objective:

The project is based on the **Monthly Gas Production** of **Australia**. The data given to us contains the **quantity of gas** that has been **produced every month** for a course of time. We are required to use the same data, build various **time forecast models** and use the **best model** out of them to predict what would be the **quantity of the gas production** that would take place for **12 months** beyond the time period in the given data.

2) Initial Data Analysis:

The required dataset is present in one of the packages in the R called "Forecast". The dataset can be called onto to the session after importing the package.

a) Invoking the necessary libraries into the R session:

The following libraries have to invoke into the R session before we can begin the analysis. The libraries, if not installed, can be installed by using the function

install.packages() and called onto the R session using **library()**.

Forecast – This library contains the dataset "**Gas"** and several other **functions to be used for model building**

- **tseries** – This library contains various functions which will be **useful for modifying time series.**
- **ggplot2 –** This library contains functions to be used for **plotting time series data.**
- **dygraphs –** This library contains functions for **plotting time series object.**
- **xts –** This library contains functions useful for **various plots**.
- **fts –** This library is useful to find the **periodicity**
- **TSA –** This library is useful to build **regression models.**
- **Metrics –** This library contains functions for **model performance.**

b)Calling the required dataset into the R session:

The required dataset '**gas'** is downloaded along with the **forecast** package and can be called into the **session** using its variable name 'gas'. For our **further analysis**, we will call it into **another variable** namely '**gasprod' which will be used to call the original dataset when required.** The dataset can be viewed using the function **print()**. We can check the type of dataset using the function **class()**.

```
> ### As the dataset to be used for the analysis is present in the ####
> ### forecast library, the data can be called directly using ###
> ### the data name 'gas'. For our analysis, we import it and
> ### name it 'gasprod' ###
> gasprod = gas
```

```
> ### Viewing the data ####
> print(gasprod)
       Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
1956  1709  1646  1794  1878  2173  2321  2468  2416  2184  2121  1962  1825
1957  1751  1688  1920  1941  2311  2279  2638  2448  2279  2163  1941  1878
1958  1773  1688  1783  1984  2290  2511  2712  2522  2342  2195  1931  1910
1959  1730  1688  1899  1994  2342  2553  2712  2627  2363  2311  2026  1910
1960  1762  1815  2005  2089  2617  2828  2965  2891  2532  2363  2216  2026
1961  1804  1773  2015  2089  2627  2712  3007  2880  2490  2237  2205  1984
1962  1868  1815  2047  2142  2743  2775  3028  2965  2501  2501  2131  2015
1963  1910  1868  2121  2268  2690  2933  3218  3028  2659  2406  2258  2057
1964  1889  1984  2110  2311  2785  3039  3229  3070  2659  2543  2237  2142
1965  1962  1910  2216  2437  2817  3123  3345  3112  2659  2469  2332  2110
1966  1910  1941  2216  2342  2923  3229  3513  3355  2849  2680  2395  2205
1967  1994  1952  2290  2395  2965  3239  3608  3524  3018  2648  2363  2247
1968  1994  1941  2258  2332  3323  3608  3957  3672  3155  2933  2585  2384
1969  2057  2100  2458  2638  3292  3724  4652  4379  4231  3756  3429  3461
1970  3345  4220  4874  5064  5951  6774  7997  7523  7438  6879  6489  6288
1971  5919  6183  6594  6489  8040  9715  9714  9756  8595  7861  7753  8154
1972  7778  7402  8903  9742 11372 12741 13733 13691 12239 12502 11241 10829
1973 11569 10397 12493 11962 13974 14945 16805 16587 14225 14157 13016 12253
1974 11704 12275 13695 14082 16555 17339 17777 17592 16194 15336 14208 13116
1975 12354 12682 14141 14989 16159 18276 19157 18737 17109 17094 15418 14312
1976 13260 14990 15975 16770 19819 20983 22001 22337 20750 19969 17293 16498
1977 15117 16058 18137 18471 21398 23854 26025 25479 22804 19619 19627 18488
1978 17243 18284 20226 20903 23768 26323 28038 26776 22886 22813 22404 19795
1979 18839 18892 20823 22212 25076 26884 30611 30228 26762 25885 23328 21930
1980 21433 22369 24503 25905 30605 34984 37060 34502 31793 29275 28305 25248
1981 27730 27424 32684 31366 37459 41060 43558 42398 33827 34962 33480 32445
1982 30715 30400 31451 31306 40592 44133 47387 41310 37913 34355 34607 28729
1983 26138 30745 35018 34549 40980 42869 45022 40387 38180 38608 35308 30234
1984 28801 33034 35294 33181 40797 42355 46098 42430 41851 39331 37328 34514
1985 32494 33308 36805 34221 41020 44350 46173 44435 40943 39269 35901 32142
1986 31239 32261 34951 38109 43168 45547 49568 45387 41805 41281 36068 34879
1987 32791 34206 39128 40249 43519 46137 56709 52306 49397 45500 39857 37958
1988 35567 37696 42319 39137 47062 50610 54457 54435 48516 43225 42155 39995
```

```
1989 37541 37277 41778 41666 49616 57793 61884 62400 50820 51116 45731 42528
1990 40459 40295 44147 42697 52561 56572 56858 58363 45627 45622 41304 36016
1991 35592 35677 39864 41761 50380 49129 55066 55671 49058 44503 42145 38698
1992 38963 38690 39792 42545 50145 58164 59035 59408 55988 47321 42269 39606
1993 37059 37963 31043 41712 50366 56977 56807 54634 51367 48073 46251 43736
1994 39975 40478 46895 46147 55011 57799 62450 63896 57784 53231 50354 38410
1995 41600 41471 46287 49013 56624 61739 66600 60054
>
> ### Checking the class of the imported ####
> class(gasprod)
[1] "ts"
>
```

Inferences:

- We can see from the above results that the data contains values from **January of 1956** to the **August of 1995.**
- The data is already a **time series** object and need not be converted to one for further analysis.

c) Inspection of the Time Series data:

The Time Series data needs to be **inspected** using the **basic functions** before the actual analysis can be started. The inspection can be done using the following functions:

- The **summary()** function can be used to get **basic statistical information** on the **distribution of gas production values** in the data.

```
> ### Inspection of the time series data ####
> summary(gasprod)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1646    2675   16788   21415   38629   66600
```

- The **anyNA()** function can be used to check for **any missing values** in the data.

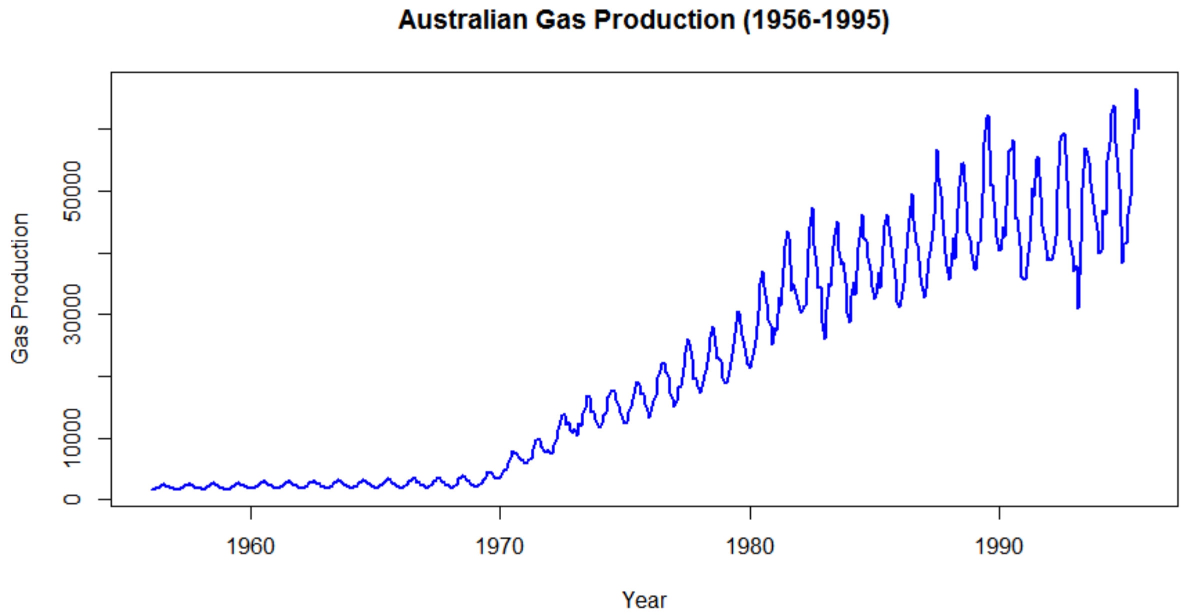
```
> anyNA(gasprod)
[1] FALSE
```

- The **findfrequency()** function can be used to check the **frequency of the time series.**

```
> findfrequency(gasprod)
[1] 12
```

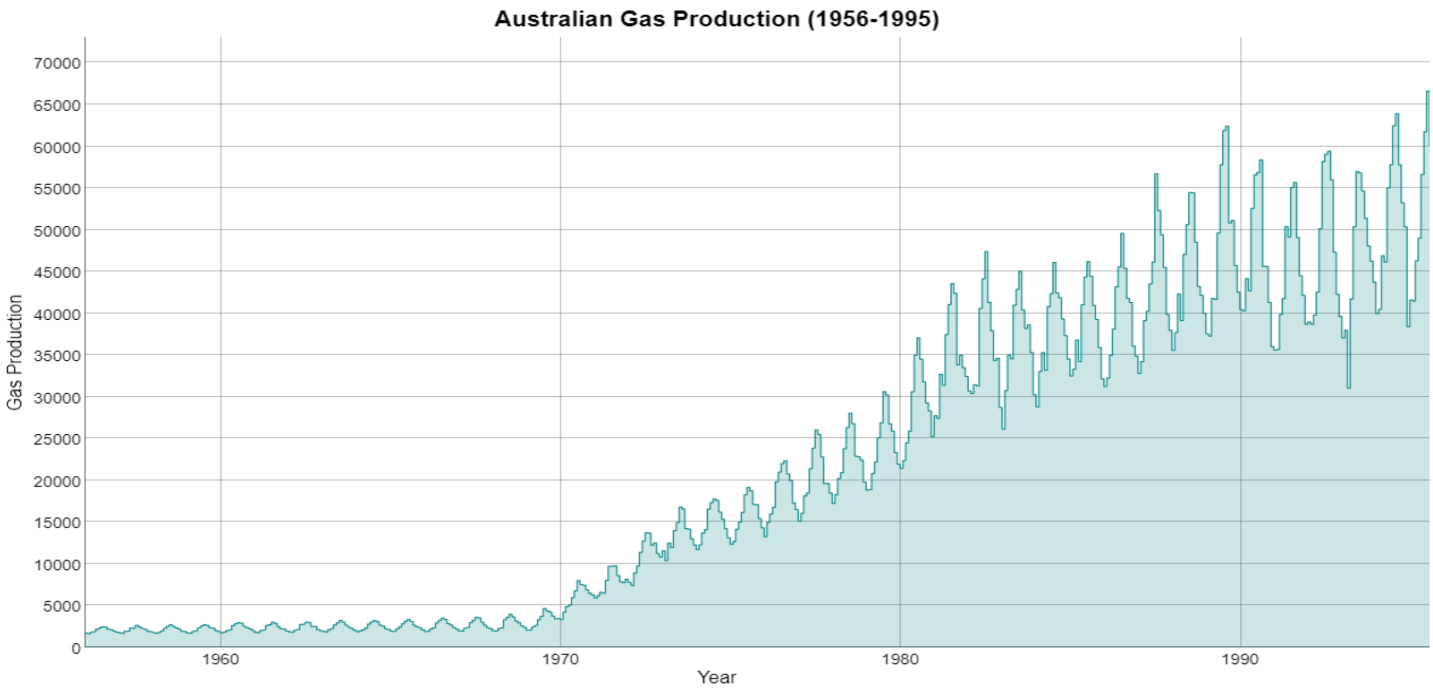
- The **ts.plot()** function can be used to **plot the time series.** The additional arguments like **colour, Title, etc.** can be included in a **list** and must be passed through the **gpars** argument.

```
> ts.plot(gasprod,gpars = list(xlab = "Year",ylab = "Gas Production",
+                              main = "Australian Gas Production (1956-1995)",
+                              col = c("Blue")),lwd = 2)
```



- The function **dygraph()** can be used to plot the **time series** as a **step plot.** An additional argument **stepPlot = TRUE** must be passed to obtain a step plot.

```
> stepplot = dygraph(gasprod,main = "Australian Gas Production (1956-1995)",
+                   xlab = "Year",ylab = "Gas Production") %>%dyOptions(stepPlot= TRUE,pointSize = 0,fillGraph = TRUE,
+                                fillAlpha = 0.2)
> stepplot
```



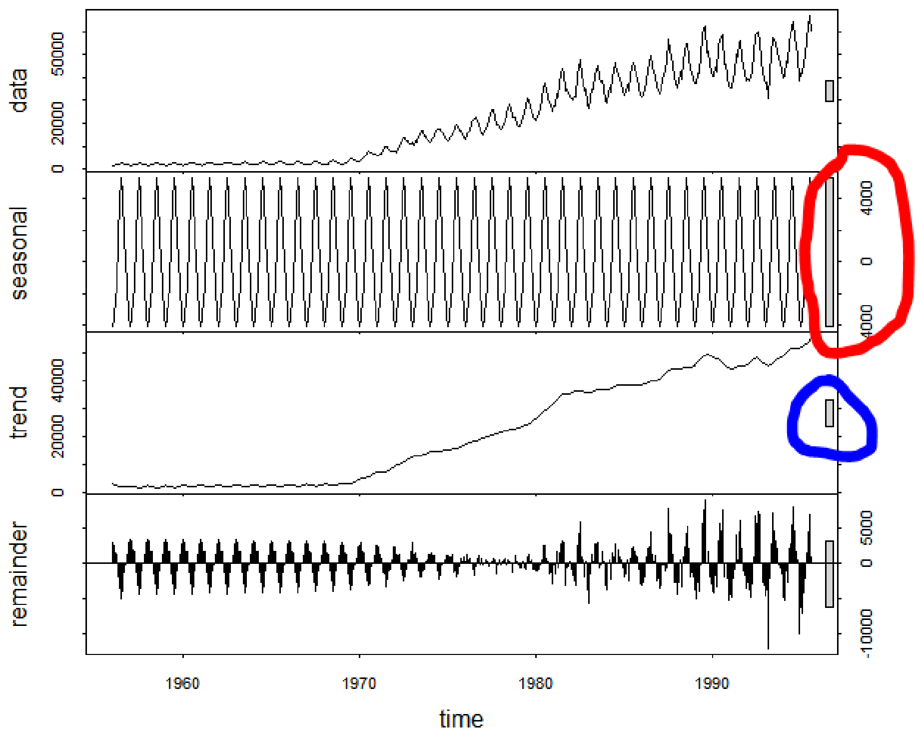
Inferences:

- We can see that **minimum amount of gas produced** was **1646 units.**
- **The Maximum amount** of **gas produced** was **66600 units.**
- The **median** of the **data** is **16788 units.**
- The **mean** of the **data** is at **21415 units.**
- In the given **time series data,** there are no **missing values.**
- The frequency of the **time series** is **12** which indicates it is a **monthly time series.**
- From the plot, we can say that the **lowest value** of **1646 units** should have been recorded in the year **1956.**
- The **highest value** of **66600 units** should have been recorded around the year **1995.**
- The **time series** doesn't display **any trend** until **1970.**
- The **time series** shows **seasonality** throughout the data.
- The **time series** shows an **upward trend** after **1970.**
- The **effect of seasonality** is **little** until **the year 1970.**
- The **seasonality** changes are very **drastic** after the year **1970.**

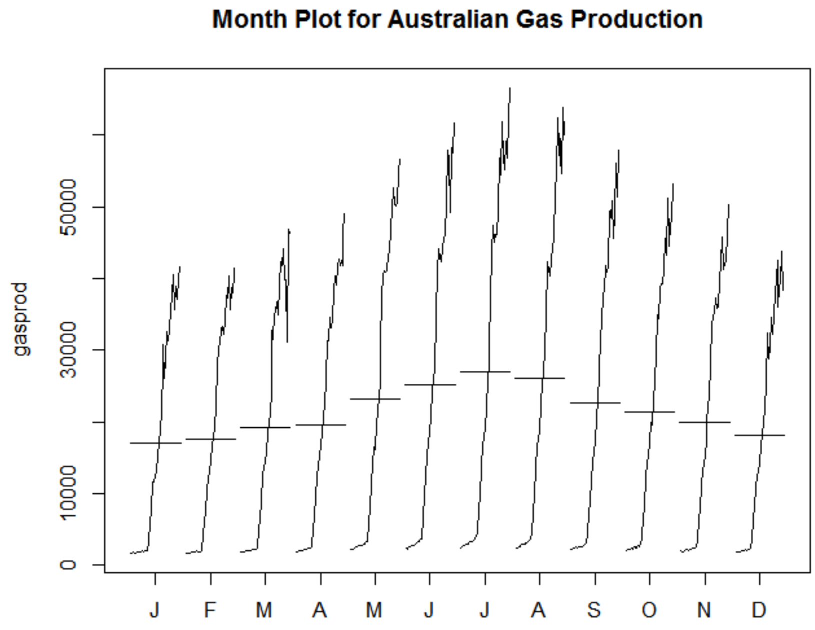
d) Analysing the components of Time Series using Decomposition:

The major **components** of **any time series data** are **Trend, seasonality and residuals.** These components would be difficult to see in a **normal time series plot**. Therefore a process called **decomposition** can help **analyse each of the components of the time series** by **separating** all the **three components** separately. The **decomposition** can be done using the function **stl()** with argument **s.window = "periodic"** since the **occurrence of the seasonality** is **periodic.** After creating a **decomposed object**, the same object can be used to **plot** various plots such as **monthplot and seasonalplot.** They can be plotted by using the functions **monthplot() and seasonplot()** respectively.

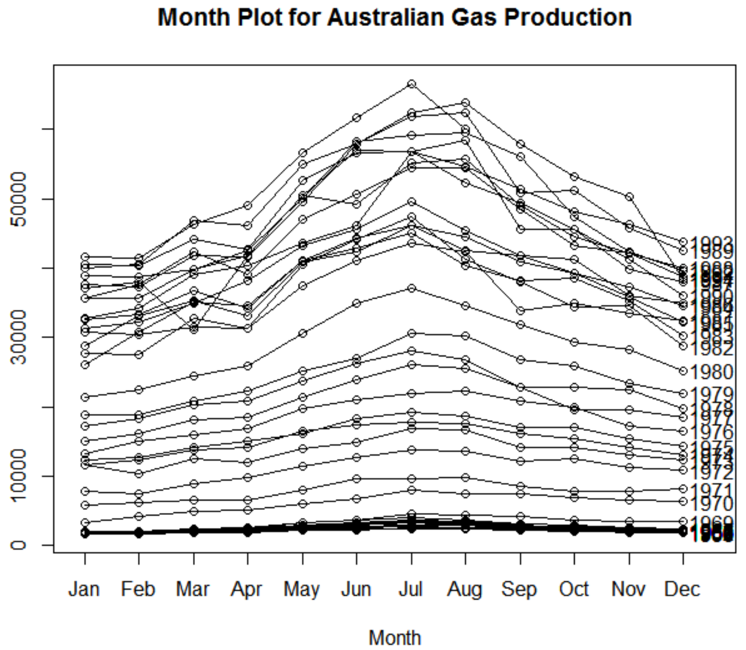
```
> ### Inspection of individual elements by decomposition of time series ####
> dc.gasprod = stl(gasprod,s.window = "periodic")
> plot(dc.gasprod)
```




```
> monthplot(gasprod,main = "Month Plot for Australian Gas Production")
```




```
> seasonplot(gasprod,year.labels = TRUE,
+             main = "Month Plot for Australian Gas Production")
```



Inferences:

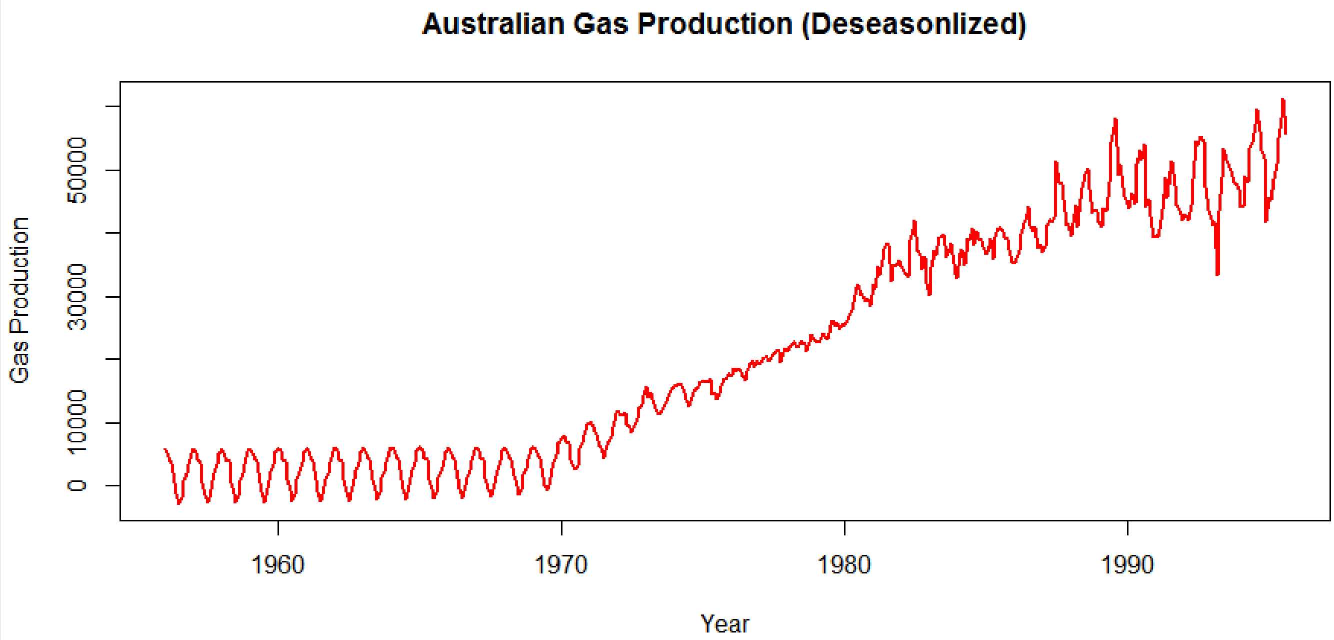
- The **seasonality** seems to be **periodic** almost **resembling** the **same pattern** for the whole time period.
- The **time series** did not **trend** until **1970**, and then after **1970,** it started showing an **upward trend.**
- The **residuals,** which is the remainder after the **trend and seasonality,** is **considerably low** compared to the **trend and seasonality.**
- **Even** though of **higher magnitude,** it actually is **less significant** in describing the **time series.**
- The **magnitude** of **trend** maybe **lower** compared to the **seasonality,** it is **very significant** parameter in determining the **time series data.**
- The **average values** keep increasing **every year** after **1970.**
- The **average values** for the **July** seems to be the highest.
- We can observe **periodic seasonal fluctuations** with an **increasing trend** indicates the **series** follows **a multiplicative seasonality**.

3) Splitting of Time Series and finding periodicity:

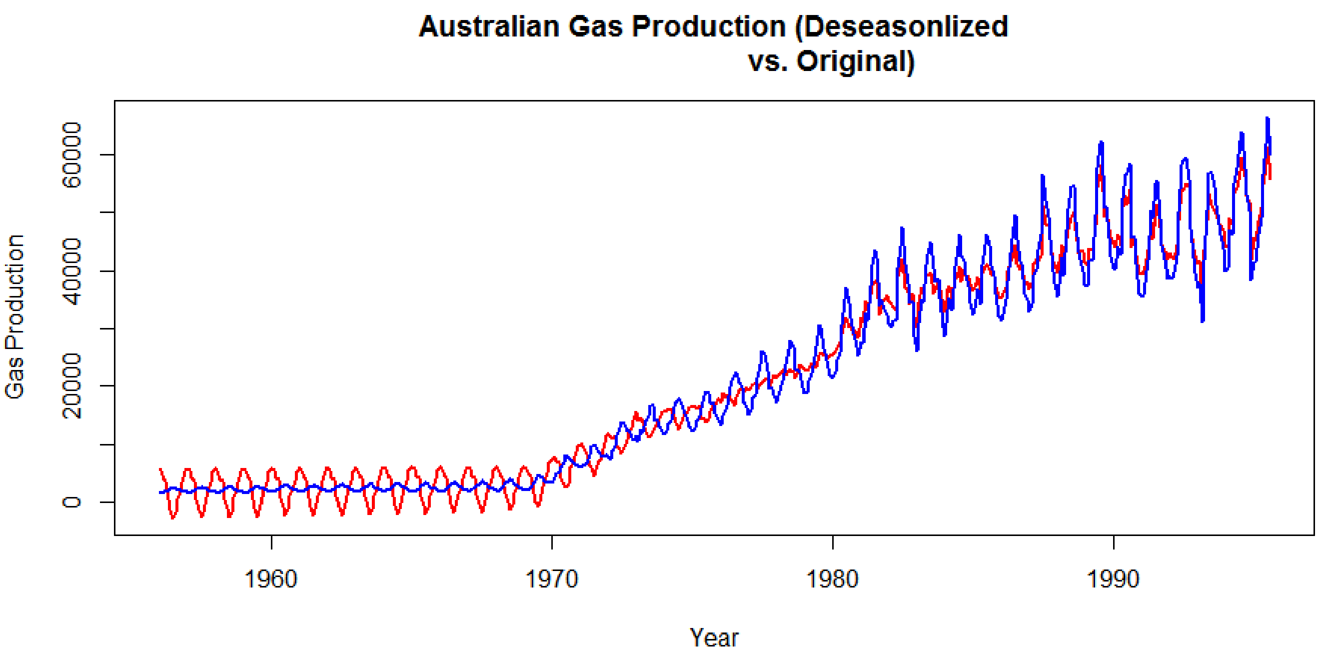
a) Deseasonalizing the Time Series:

As inferred above, the time series contains **periodic seasonality** with an **increasing trend.** This is indicative of **multiplicative seasonality** which is high in magnitude. To create certain time series model, the **time series** needs to be eliminated of all the seasonality. To do this, we can use the **time series object** that was created during the **decomposition** and **eliminate** the seasonality component from it. We name the **deseasonalized time series** as **‘ds.gasprod’.**

```
> ### De-Seasonalizing the time series from the decomposed time series####
> ds.gasprod = (dc.gasprod$time.series[,2]+dc.gasprod$time.series[,3])
> ts.plot(ds.gasprod,gpars = list(xlab = "Year",ylab = "Gas Production",
+                               main = "Australian Gas Production (Deseasonlized)",
+                               col = c("Red")),lwd = 2)
```



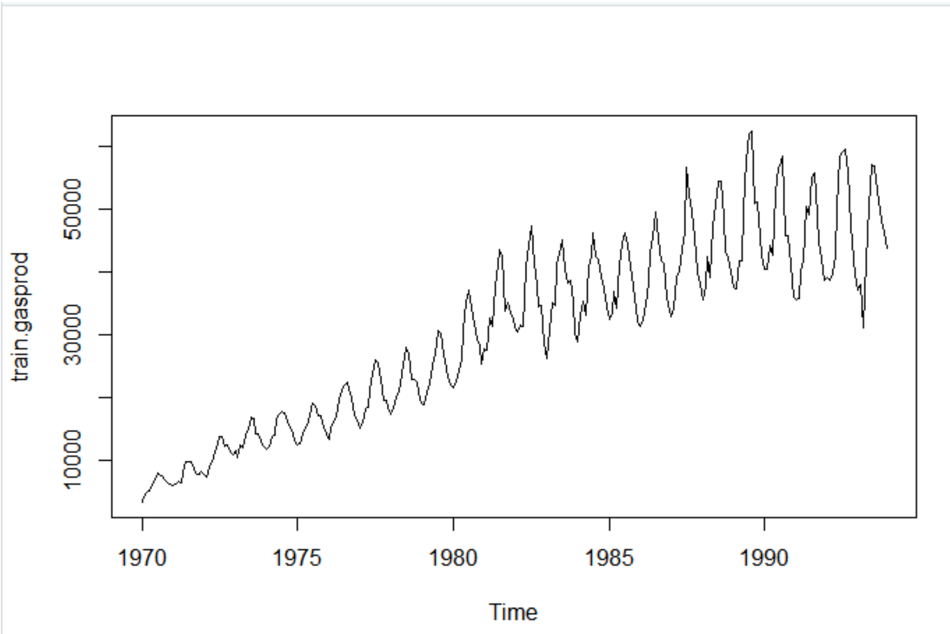

```
> ts.plot(ds.gasprod,gasprod,gpars = list(xlab = "Year",ylab = "Gas Production",
+                               main = "Australian Gas Production (Deseasonlized
+                               vs. Original)",
+                               col = c("Red","Blue")),lwd = 2)
```



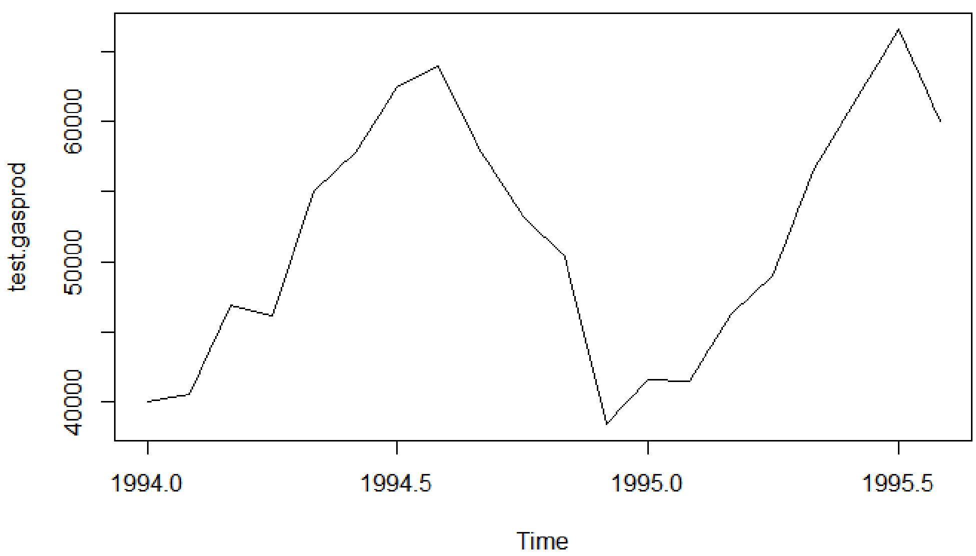
b) Splitting of Time Series into Training and Testing data:

The splitting of the time series data can be done using the **window**() function. We need to split both the **original(Seasonlized)** data and also the **deseasonalized data** into training and testing data. The **Training data** must be considered from **1970 January to 1993 December.** The **Testing data** must be considered from **1994 January to 1995 August.** The Training data for **original and deseasonalized** is named **'train.gasprod'** and **train.ds.gaspord'.** The Testing data for **original and deseasonalized** is named **'test.gasprod'** and **test.ds.gaspord'.**

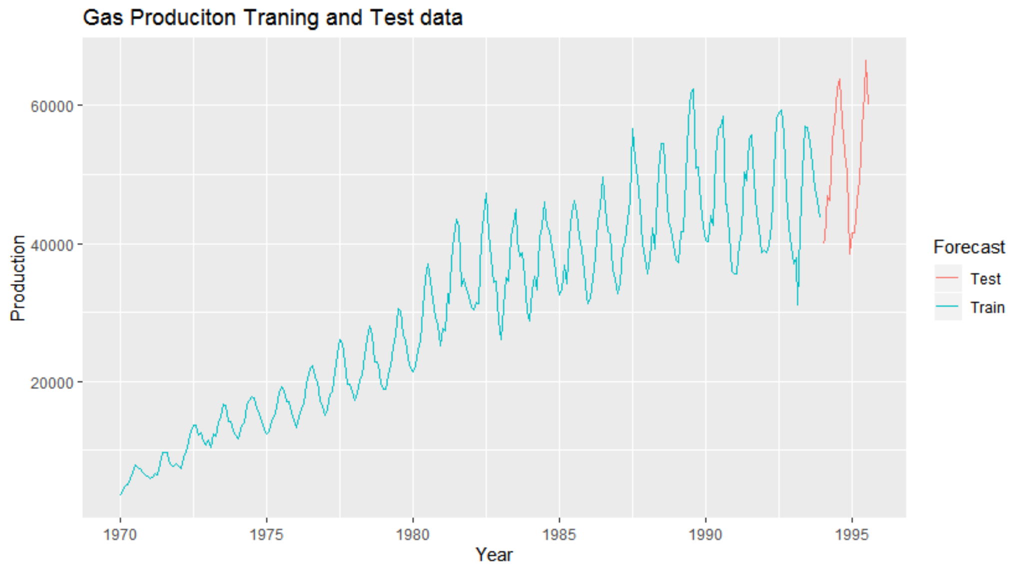
```
> ### splitting the time series into training and testing samples (Original) ####
> train.gasprod = window(gasprod,start=c(1970,1), end=c(1993,12), freq=12)
> ts.plot(train.gasprod)
```



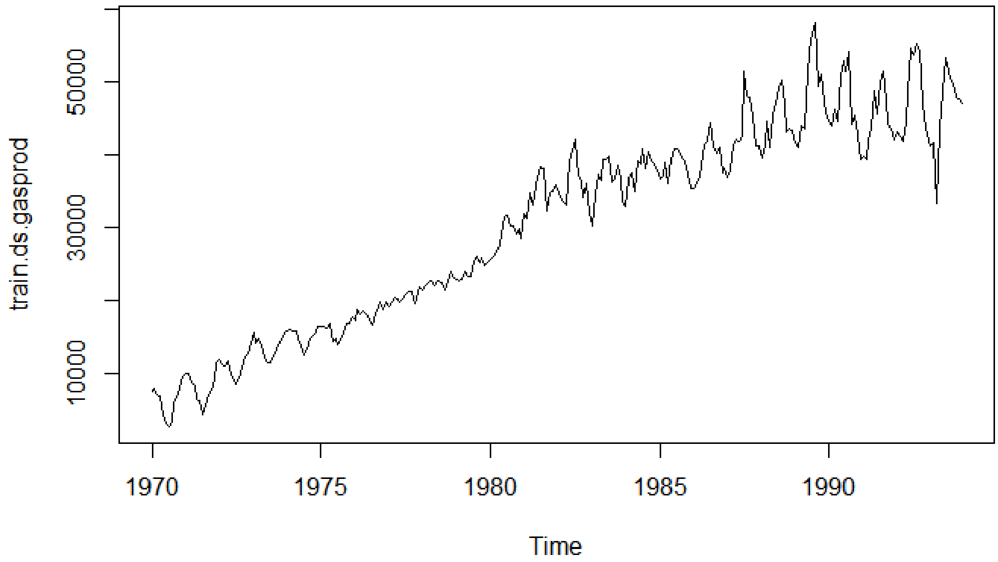

```
> test.gasprod = window(gasprod,start=c(1994,1),end=c(1995,8), freq=12)
> ts.plot(test.gasprod)
```



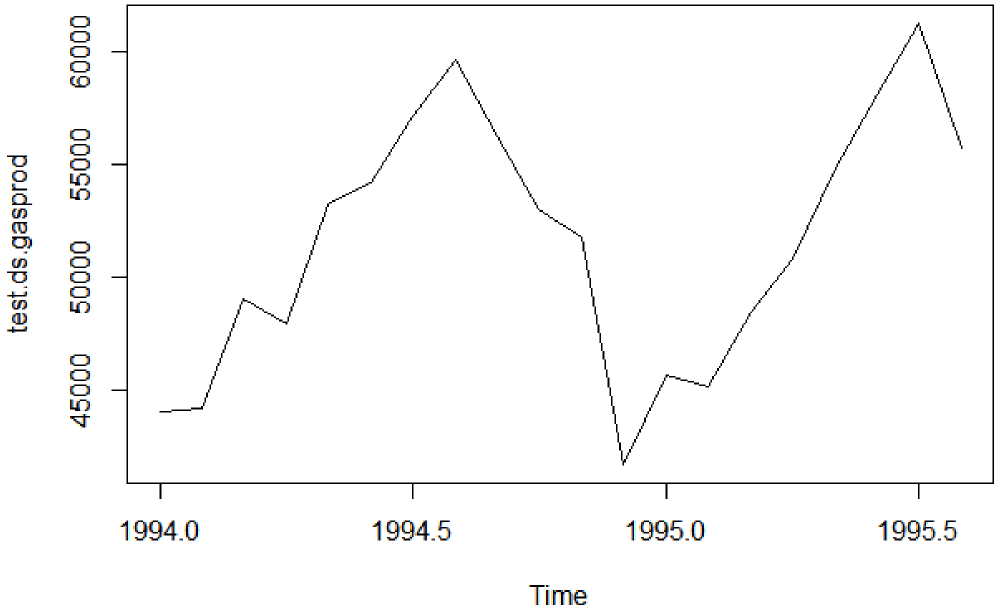

```
> autoplot(train.gasprod, series="Train") + autolayer(test.gasprod, series="Test") +
+   ggtitle("Gas Produciton Traning and Test data") +
+   xlab("Year") + ylab("Production") +
+   guides(colour=guide_legend(title="Forecast"))
```



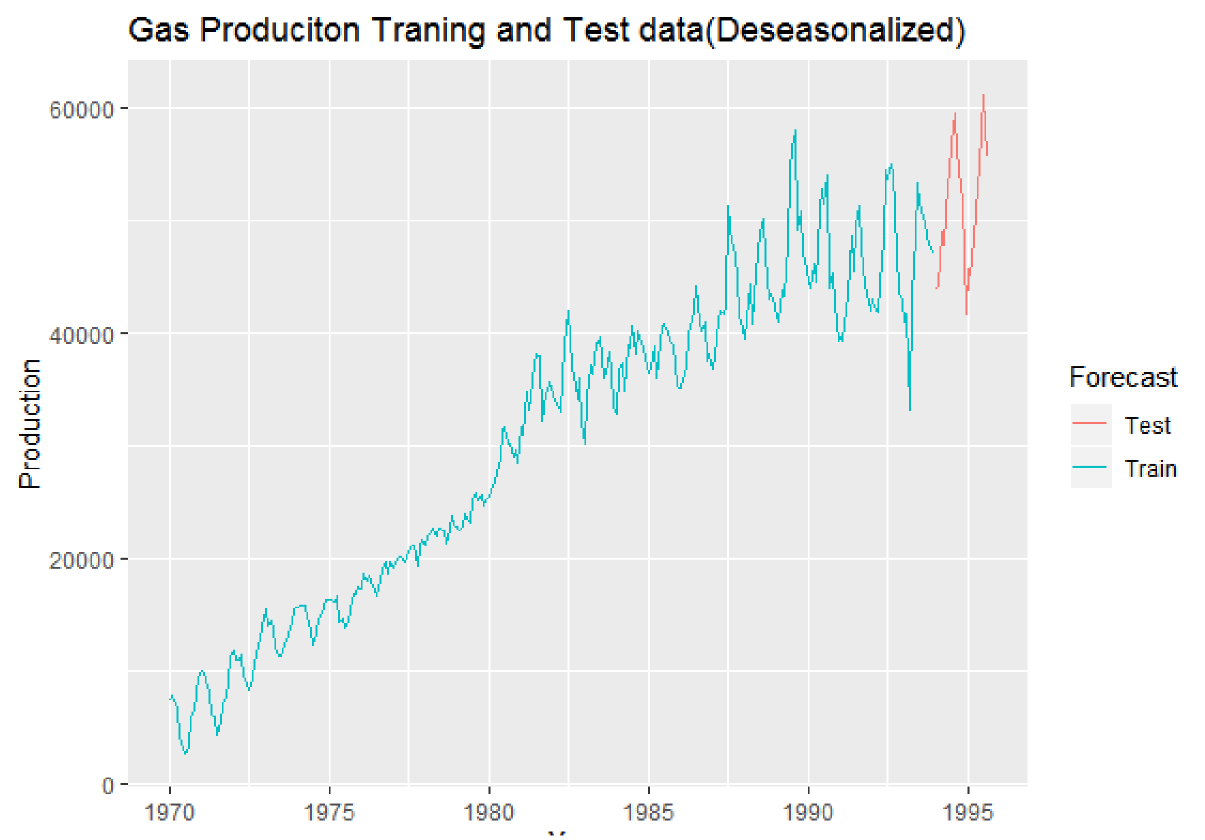

```
> ### Splitting the time series into training and testing samples(Deseasonalize) ####
> train.ds.gasprod = window(ds.gasprod,start=c(1970,1), end=c(1993,12), freq=12)
> ts.plot(train.ds.gasprod)
```




```
> test.ds.gasprod = window(ds.gasprod,start=c(1994,1),end=c(1995,8), freq=12)
> ts.plot(test.ds.gasprod)
```




```
> autoplot(train.ds.gasprod, series="Train") + autolayer(test.ds.gasprod, series="Test") +
+   ggtitle("Gas Produciton Traning and Test data(Deseasonalized)") +
+   xlab("Year") + ylab("Production") +
+   guides(colour=guide_legend(title="Forecast"))
```



c) Checking the periodicity of the Time Series:

The **periodicity** of a **time series data** is the measure of **regular intervals** at which the **observations** are recorded. We can check the **periodicity** of the time series data using the function **periodicity()**. We can also check the frequency of the data using **findfrequency() function.**

```
> ### Checking the periodicity of the Time Series ####
> periodicity(gasprod)
Monthly periodicity from Jan 1956 to Aug 1995
> findfrequency(gasprod)
[1] 12
> periodicity(train.gasprod)
Monthly periodicity from Jan 1970 to Dec 1993
> findfrequency(train.gasprod)
[1] 12
> periodicity(test.gasprod)
Monthly periodicity from Jan 1994 to Aug 1995
> findfrequency(test.gasprod)
[1] 12
```

Hence we can see that **periodicity** for **time series data** is **12**.

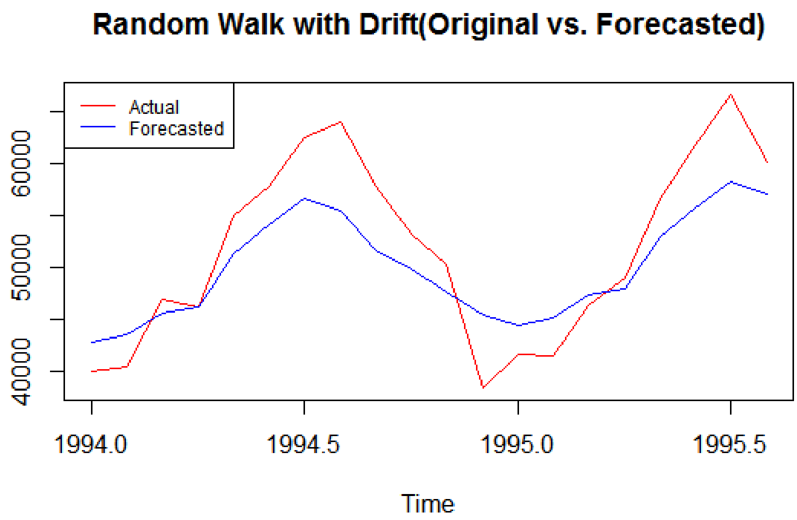
4) Using Simple Forecasting methods:

The **future values** for the **future** can be forecasted can be done using **various forecast models.** The models are basically of two types. One of them is **Simple Forecast Methods** which use the **pattern of the past values** to predict the **future values.**

a) Random Walk Model:

This method forecasts **next period value** as per the amount of **change over time (called the drift)** is evaluated the **average change** seen in past data. The **function forecast()** on the **decomposed object** along with the argument **method = “rwdrift”**.

```
> #### Naive Method ####
> gasprod.rw = stl(train.gasprod,s.window = 'p')
> gasprod.rw = forecast(de.gasprod.rw,method = "rwdrift",h = 20)
> ts.plot(test.gasprod,gasprod.rw$mean,gpars = list(col = c("Red","Blue"),
+          main = "Random Walk with Drift(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
> vec.rw = cbind(test.gasprod,gasprod.rw$mean)
> MAPE.rw = mean(abs(vec.rw[,1]-vec.rw[,2])/vec.rw[,1])
> print(MAPE.rw)
[1] 0.0737419
```



Inferences:

We can see that even though **MAPE** is very low, from the graphs, we can see that the **values forecasted** are always lower than the **Actuals.** We cannot use this **method of**

forecast just because of **MAPE** because there are not many **parameters** that can be **rectified** while making the model and this time series data **needs** the option of **parameter tuning** due to its **complexity.**

b) Simple Exponential Smoothing:

The **Simple Exponential Smoothing** is a **one-step** forecast method where all the **forecast values** are **identical.** The method can be expressed **mathematically** as,

$$\hat{Y}_{t+1}=\alpha Y_t+\alpha(1-\alpha)Y_{t-1}+\alpha(1-\alpha)^2 Y_{t-1}+\cdots,\ 0<\alpha<1$$

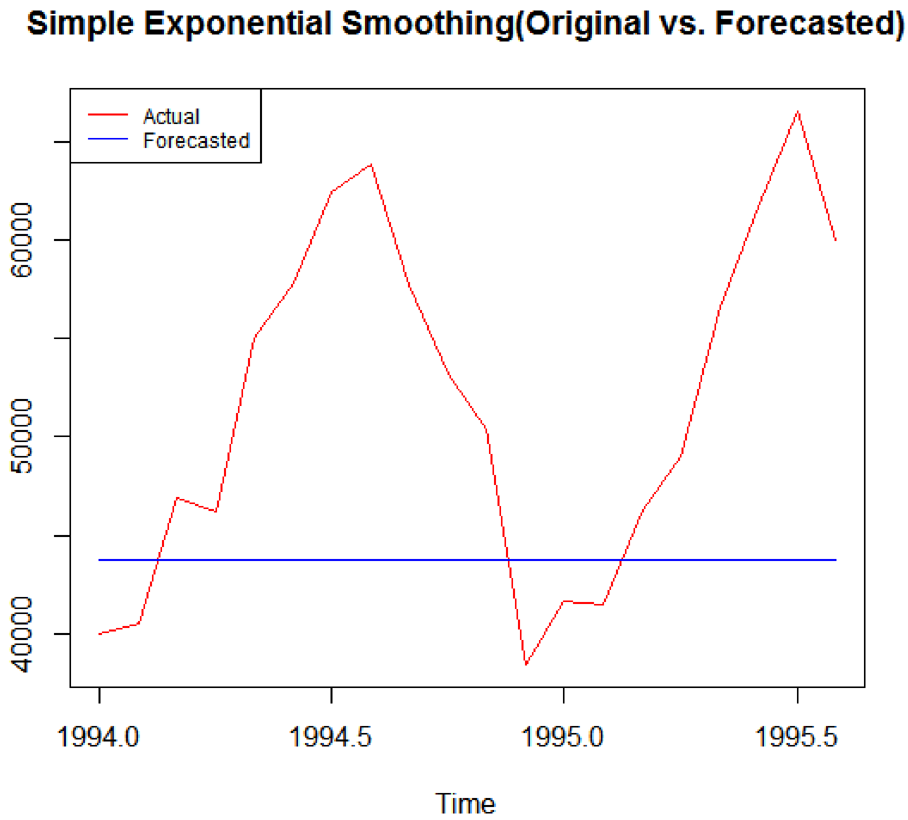
Where α is the smoothing parameter for the level.

The **simple exponential smoothing** can be done using the function **ses()**

```
> #### Simple Exponential Smoothing (Original) ####
> gasprod.ses = ses(train.gasprod,start = c(1970,1),end = c(1993,12),frequency = 12,h = 20)
> summary(gasprod.ses)

Forecast method: Simple exponential smoothing

Model Information:
Simple exponential smoothing

Call:
 ses(y = train.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), frequency = 12)

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = 6002.9039

  sigma:  3344.025

     AIC     AICc      BIC
6309.126 6309.210 6320.115

Error measures:
                   ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
Training set 131.0317 3332.394 2416.021 0.1509165 8.113316 0.9164853 0.3030855
```

```
> print(gasprod.ses$mean)
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
1994 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25
1995 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25
```

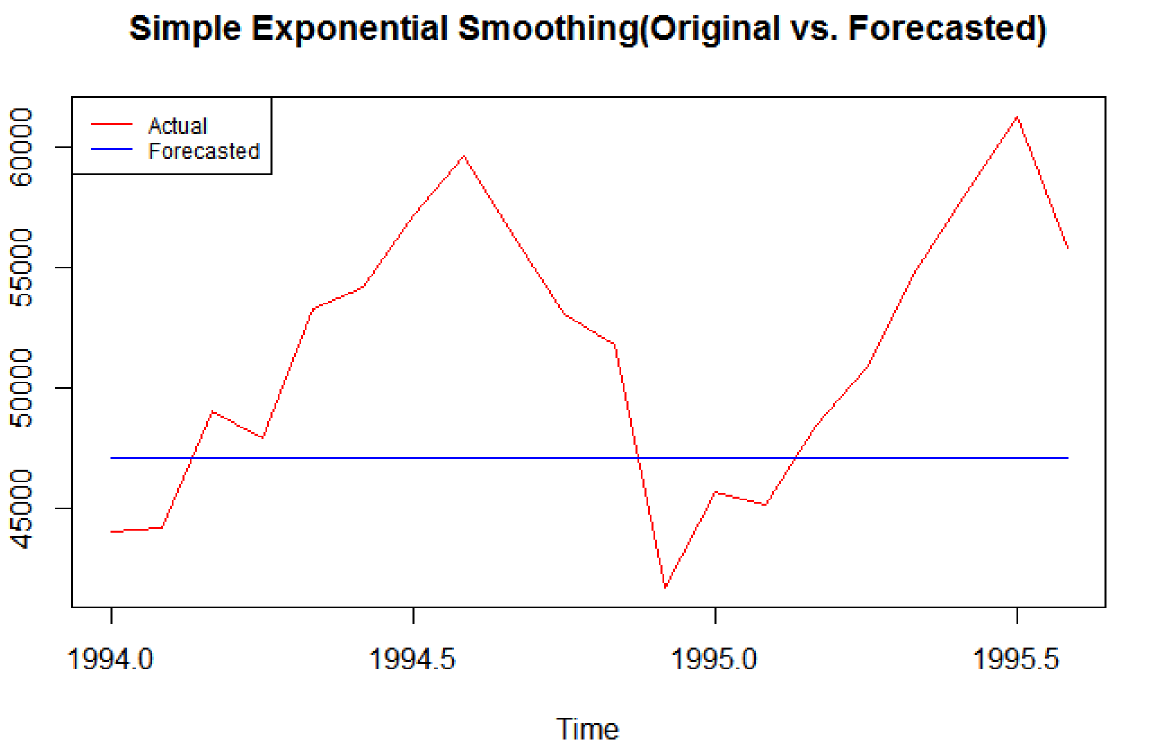
```
> ts.plot(test.gasprod,gasprod.ses$mean,gpars = list(col = c("Red","Blue"),
+                                       main = "Simple Exponential Smoothing(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
```




```
> vec.ses = cbind(test.gasprod+gasprod.ses$mean)
> MAPE.ses = mape(test.gasprod,gasprod.ses$mean)
> print(MAPE.ses)
[1] 0.1726068
```

```
> #### Simple Exponential Smoothing (Deseasonalized) ####
> gasprod.dses = ses(train.ds.gasprod,start = c(1970,1),end = c(1993,12),frequency = 12,h = 20)
> summary(gasprod.dses)

Forecast method: Simple exponential smoothing

Model Information:
Simple exponential smoothing

Call:
 ses(y = train.ds.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), frequency = 12)

  Smoothing parameters:
    alpha = 0.9833

  Initial states:
    l = 7432.9827

  sigma:  2319.347

     AIC     AICc      BIC
6098.373 6098.458 6109.362

Error measures:
                  ME     RMSE      MAE       MPE     MAPE      MASE         ACF1
Training set 139.963 2311.279 1591.754 0.2300846 6.077039 0.6038107 -0.005553574
```

```
> print(gasprod.dses$mean)
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov
1994 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77
1995 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77
          Dec
1994 47067.77
1995

> ts.plot(test.ds.gasprod,gasprod.dses$mean,gpars = list(col = c("Red","Blue"),
+                                         main = "Simple Exponential Smoothing(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
```



```
> MAPE.dses = mape(test.ds.gasprod,gasprod.dses$mean)
> print(MAPE.dses)
[1] 0.1110184
```

Inferences:

We can see that **MAPE value(0.11)** is **high** for this model and the graph very much deviates from the actual values. This is because the **time series** data has **both trend and seasonality** which is not suitable for **SES** method.

c) Double Exponential Smoothing:

This method is an **extension** of **simple exponential smoothing.** It contains **two parameters** instead of **one.** Mathematically, it can be expressed in the following way.

$$Y_{t+1}=l_t+b_t$$

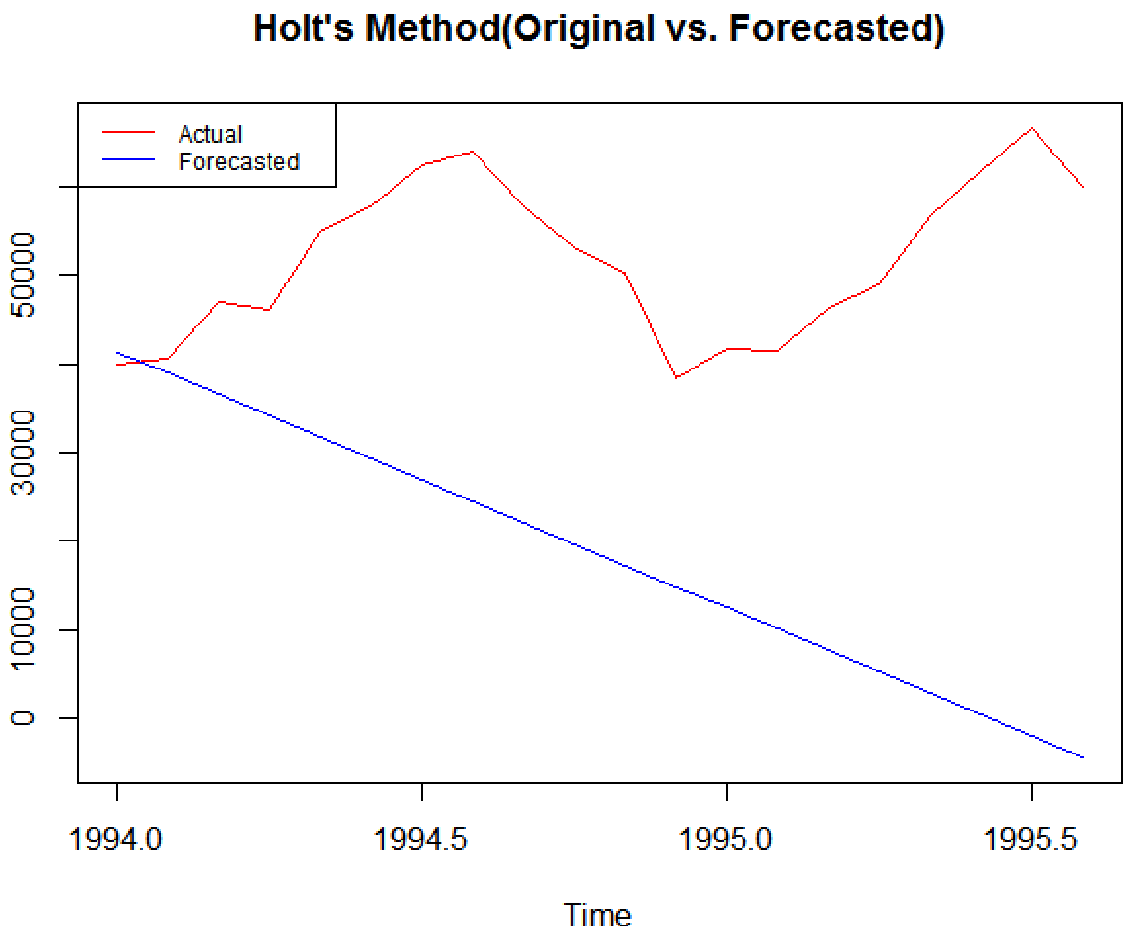
Where, l_t and b_t are **estimate of level** and **estimate of trend** respectively

The function **holt()** can be used to build the model

```
> #### Double Exponential Method (Holt Model) (originial) ####
> gasprod.holt = holt(train.gasprod ,start=c(1970,1),end=c(1993,12), freq=12,h=20)
> summary(gasprod.holt)

Forecast method: Holt's method

Model Information:
Holt's method

Call:
 holt(y = train.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), freq = 12)

  Smoothing parameters:
    alpha = 0.9436
    beta  = 0.5881

  Initial states:
    l = 3775.0087
    b = 467.2663

  sigma:  3542.179

     AIC     AICc      BIC
6344.263 6344.476 6362.578

Error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE        ACF1
Training set -16.95919 3517.494 2436.963 0.2550371 7.909274 0.9244294 -0.01120745
```

```
> gasprod.holt$mean
           Jan        Feb        Mar        Apr        May        Jun        Jul        Aug        Sep        Oct
1994 41337.3687 38932.0524 36526.7362 34121.4199 31716.1037 29310.7874 26905.4711 24500.1549 22094.8386 19689.5224
1995 12473.5736 10068.2573  7662.9411  5257.6248  2852.3086   446.9923 -1958.3239 -4363.6402
           Nov        Dec
1994 17284.2061 14878.8899
1995
```

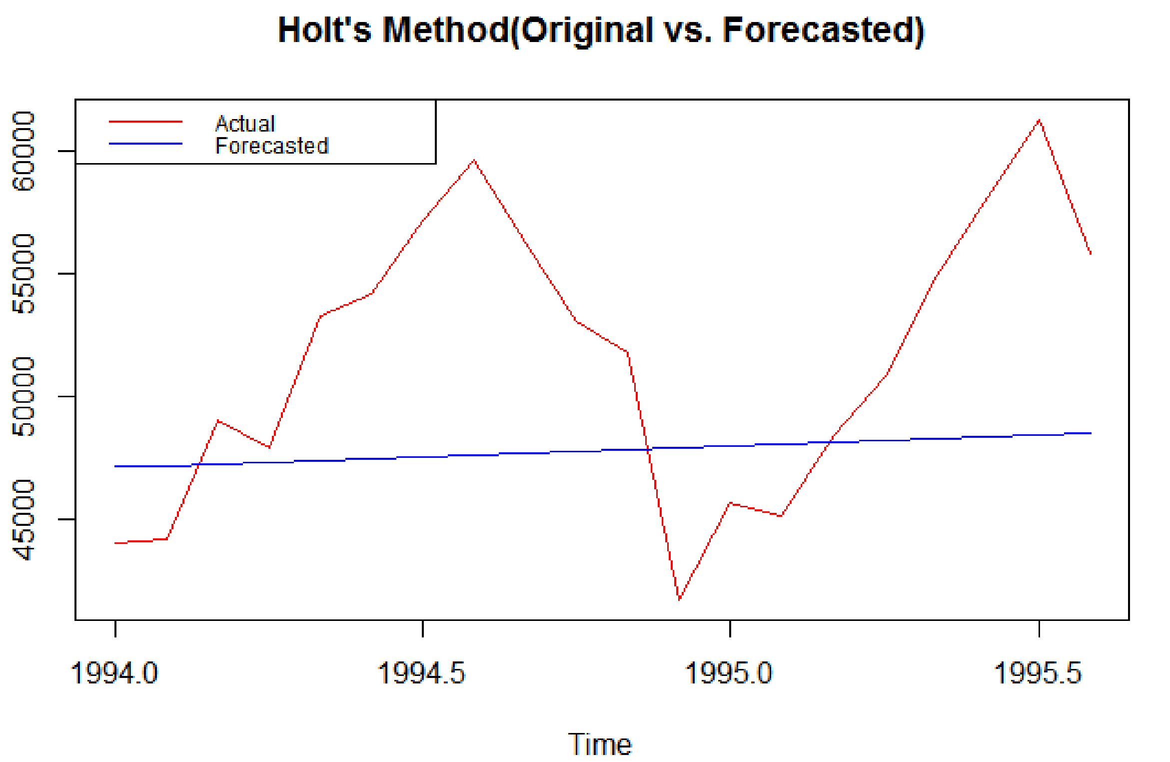
```
> ts.plot(test.gasprod,gasprod.holt$mean,gpars = list(col = c("Red","Blue"),
+                                              main = "Holt's Method(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
```




```
> MAPE.holt = mape(test.gasprod,gasprod.holt$mean)
> print(MAPE.holt)
[1] 0.6201079
```

```
> #### Double Exponential Method (Holt Model) (Deseasonlaize) ####
> gasprod.dholt = holt(train.ds.gasprod,start = c(1970,1),end = c(1993,12),freq = 12,h = 20)
> summary(gasprod.dholt)

Forecast method: Holt's method

Model Information:
Holt's method

Call:
 holt(y = train.ds.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), freq = 12)

  Smoothing parameters:
    alpha = 0.9809
    beta  = 1e-04

  Initial states:
    l = 7297.2832
    b = 70.1218

  sigma:  2324.336

     AIC     AICc      BIC
6101.590 6101.803 6119.905
```

```
> gasprod.dholt$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov
1994 47142.87 47214.95 47287.03 47359.11 47431.19 47503.27 47575.35 47647.43 47719.51 47791.59 47863.67
1995 48007.83 48079.91 48151.99 48224.07 48296.15 48368.23 48440.31 48512.39
          Dec
1994 47935.75
1995

> ts.plot(test.ds.gasprod,gasprod.dholt$mean,gpars = list(col = c("Red","Blue"),
+                                                         main = "Holt's Method(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
```




```
> MAPE.dholt = mape(test.ds.gasprod,gasprod.dholt$mean)
> print(MAPE.dholt)
[1] 0.1033223
```

Inferences:

We can see that **MAPE value(0.10)** is **high** for this model and the graph very much deviates from the actual values. This is because the **time series** data has **both trend and seasonality** which is not suitable for **Holt's** method.

d)Holt Winter's Model:

This model is an **extension** of the **Holt's model** where instead of **two**, we consider **three parameters.** Mathematically, this can be represented as,

Forecast equation: $Y't+1=lt+bt+st-m(k+1)$

Level Equation: $lt=\alpha(Yt-st-m)+\alpha(1-\alpha)Yt-1,\ 0<\alpha<1$

Trend Equation : $bt=\beta(lt-lt-1)+(1-\beta)bt-1,\ 0<\beta<1$

Seasonal Equation : $\gamma(Yt-lt-1-bt-1)+(1-\gamma)st-m,\ 0<\gamma<1$

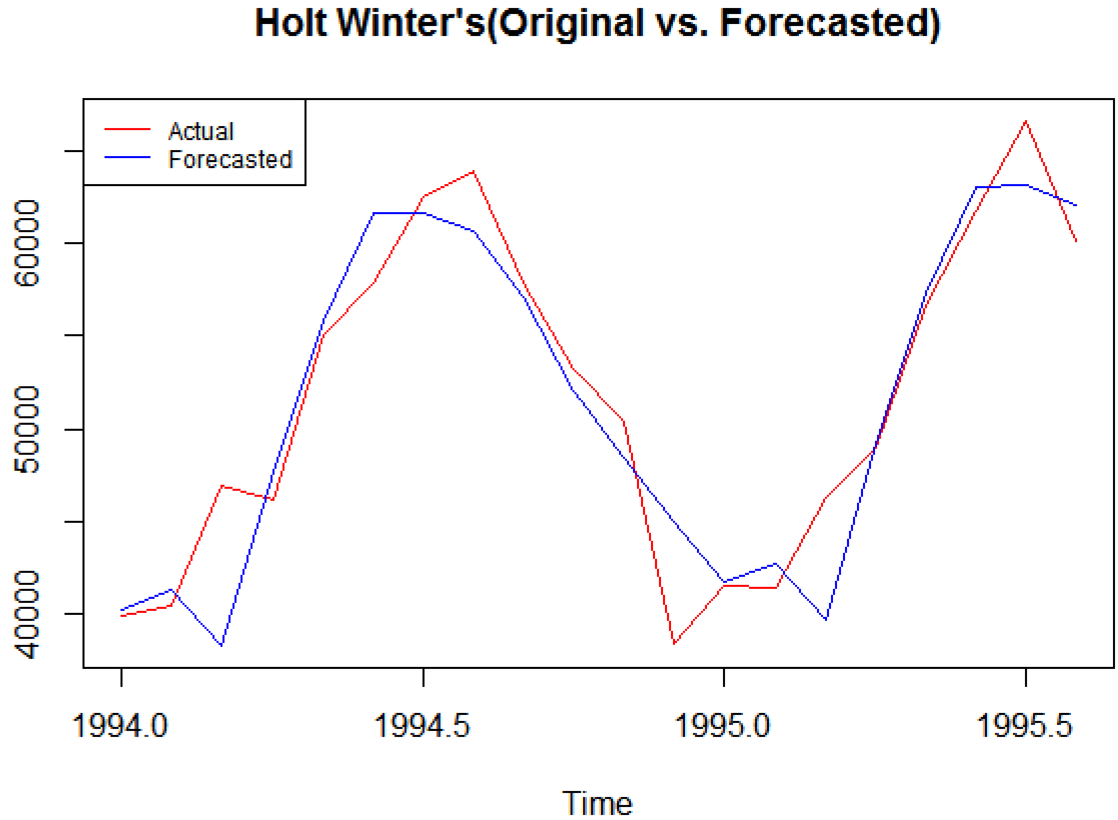
Here, α**,** β **and** γ are **smoothing parameters.**

The **Holt Winter's model** can be applied to the **time series data** using the function **hw()**.

```
> #### Holt Winter's method (Original) ####
> gasprod.hw = hw(train.gasprod,start = c(1970,1),end = c(1993,12),freq = 12,h = 20)
> summary(gasprod.hw)

Forecast method: Holt-Winters' additive method

Model Information:
Holt-Winters' additive method

Call:
 hw(y = train.gasprod, h = 20, start = c(1970, 1), end = c(1993, 

 Call:
     12), freq = 12)

  Smoothing parameters:
    alpha = 0.3408
    beta  = 1e-04
    gamma = 0.5936

  Initial states:
    l = 6253.203
    b = 119.5506
    s = -4511.742 -2141.073 234.0438 2010.202 5919.329 7284.465
           5272.426 2485.985 -2602.642 -3068.389 -5131.983 -5750.62

  sigma:  2109.356

     AIC     AICc      BIC
6057.255 6059.521 6119.525

Error measures:
                   ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
Training set 78.69136 2049.926 1551.842 0.4623734 7.505829 0.5886704 0.2738151
```

```
> gasprod.hw$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov
1994 40279.65 41289.25 38304.76 47712.40 55810.29 61602.40 61619.91 60595.61 57042.03 51962.18 48460.99
1995 41741.45 42751.05 39766.56 49174.20 57272.09 63064.20 63081.72 62057.41
          Dec
1994 44992.38
1995
```

```
> ts.plot(test.gasprod,gasprod.hw$mean,gpars = list(col = c("Red","Blue"),
+                                                    main = "Holt Winter's(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
```

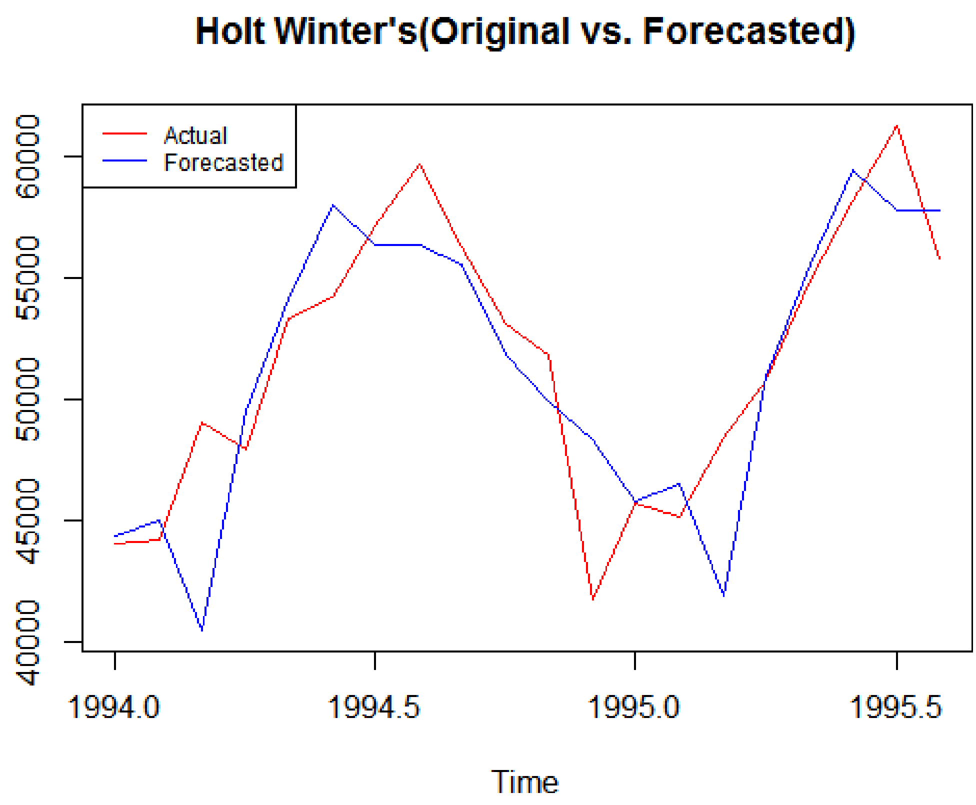



```
> MAPE.hw = mape(test.gasprod,gasprod.hw$mean)
> print(MAPE.hw)
[1] 0.04666037
```

```
> #### Holt Winter's method (Deseaonalize) ####
> gasprod.dhw = hw(train.ds.gasprod,start = c(1970,1),end = c(1993,12),freq = 12,h = 20)
> summary(gasprod.dhw)

Forecast method: Holt-Winters' additive method

Model Information:
Holt-Winters' additive method

Call:
 hw(y = train.ds.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), freq = 12)

  Smoothing parameters:
    alpha = 0.3408
    beta  = 1e-04
    gamma = 0.5934

  Initial states:
    l = 6263.4243
    b = 119.8431
    s = -1190.992 -669.5463 69.8913 481.8762 1663.536 1972.405
           1680.539 771.3246 -793.5156 -903.362 -1416.127 -1666.028

  sigma:  2109.394

     AIC     AICc      BIC
6057.265 6059.531 6119.535

Error measures:
                   ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
Training set 77.89129 2049.962 1551.948 -1.123422 8.141496 0.5887106 0.2739442
```

```
> gasprod.dhw$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
1994 44364.14 45005.85 40472.48 49522.52 54097.59 58011.54 56309.27 56342.08 55515.56 51799.98 49934.90 48315.63
1995 45829.18 46470.89 41937.52 50987.55 55562.62 59476.58 57774.30 57807.12
```

```
> ts.plot(test.ds.gasprod,gasprod.dhw$mean,gpars = list(col = c("Red","Blue"),
+                                             main = "Holt Winter's(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
```



```
> MAPE.dhw = mape(test.ds.gasprod,gasprod.dhw$mean)
> print(MAPE.dhw)
[1] 0.0458347
```

Inferences:

We can see that **MAPE value(0.04)** is **very low** for this model and the graph shows that **the forecasted values** almost **explain** the **fluctuations** that the **time series data.**

Even the **AIC and BIC values** are **good enough.**

Out of these both models where we used **original** or **deseasonalized,** we can go for **original** since there isn't much difference in **MAPE** between models **created** from **original** data and **deseasonalized** data. And also if we chose **deseasonalized model**, we are giving up **some of the original** data that was lost during the **deseasonalization.** Therefore choosing the **original model** would be the right decision over the **deseasonalized model.**

5) Building Regression models:

The concept of Regression can also be applied to the **time series data** and can be used for **forecasting**. The **forecasted values** can be termed as **Response variable** and the previous values can be termed as **Regressor variables.** But in this type of regression, instead of **correlation,** we talk about **auto-correlation** which showcases the **relation between the time series and the lagged version of itself over several time periods.** The most common regression model that can be built for this time series is **ARIMA**. There are **three components** in the **ARIMA model**. **The differencing factor(d), the Moving average factor(q) and the autoregressive term (p).** In **R**

programming, we have two types of processes to perform **ARIMA** namely **Manual ARIMA** and **Auto ARIMA**.

a) Checking the time series for stationarity:

A **series** is said to be **stationary** if it meets the following conditions.

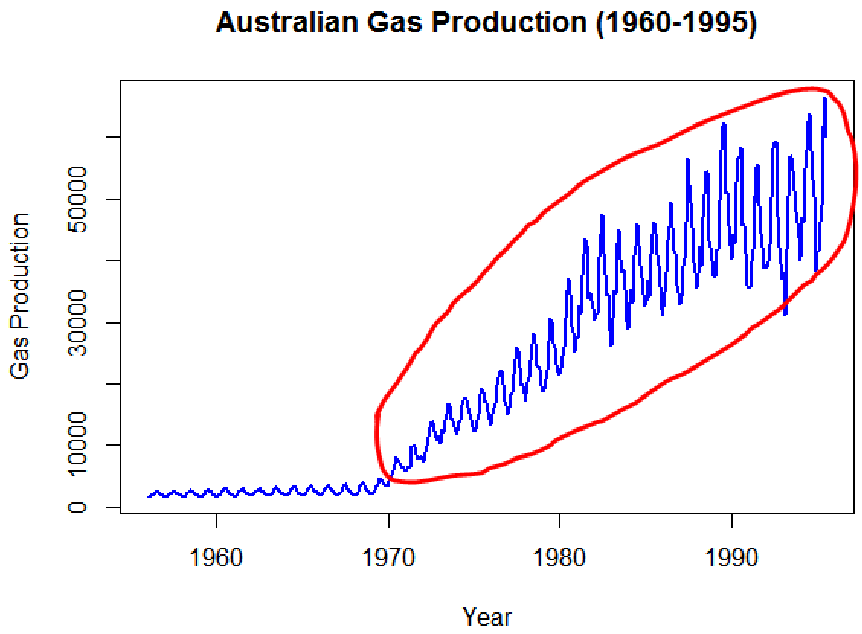
- It should not have any **trend**
- It should not have any **seasonality**
- It should not be **cyclic**
- It's values should not be dependent on **time**
- The **average values** should not change over time
- The **variance of the time series** should be constant.

The series can be checked for stationarity in the following methods:

➢ Checking the time series plot:

The time series can be plotted like above to check for **seasonality and trend.** The series can be plotted using the function **ts.plot()**

```
> #### Checking for stationarity using visualization ####
> ts.plot(gasprod,gpars = list(xlab = "Year",ylab = "Gas Production",
+                              main = "Australian Gas Production (1960-1995)",
+                              col = c("Blue")),lwd = 2)
>
```



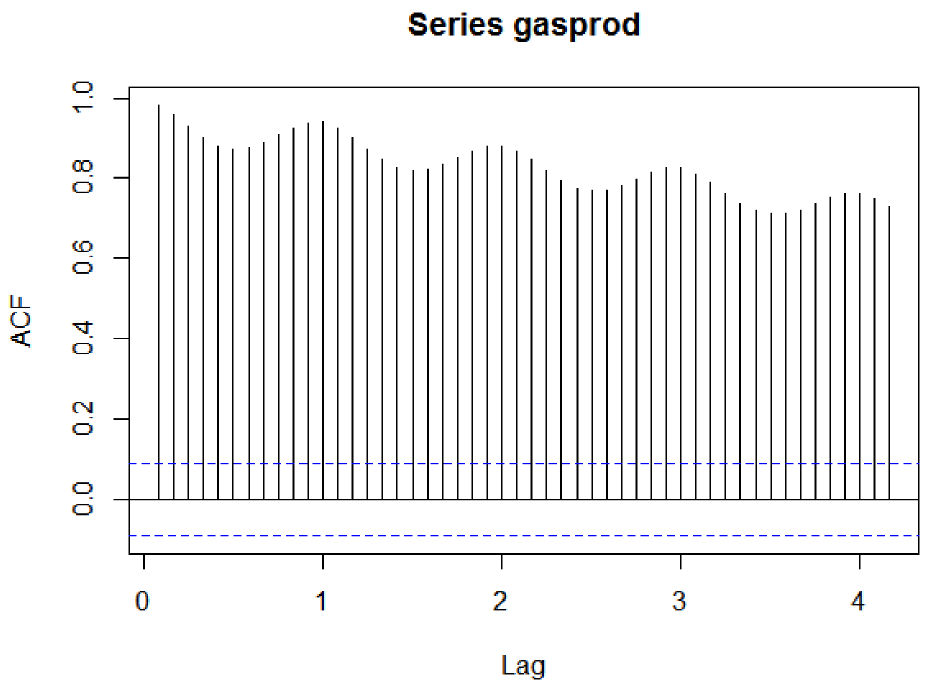
We can see that **the series** has **seasonality and trend.** Therefore by this we can rule out that the series is **non-stationary.**

- <u>**Checking the ACF and PACF plots:**</u>

  ACF(Auto Correlation Function) and PACF(Partial Auto Correlation Function) plots are **two** important plots which show to what extent the **time series** are auto correlated. The plot shows to how many number of **lags,** the correlation is **significant.** The function **acf()** and **pacf()** can be used to get the plots.

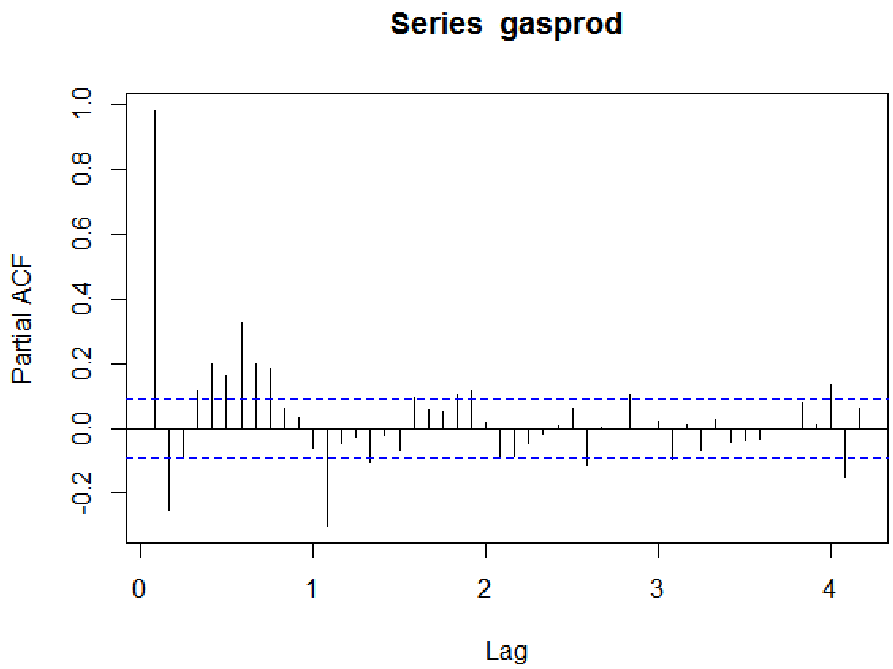
ACF plot:

```
> acf(gasprod, lag.max = 50)
```



PACF plot:

```
> pacf(gasprod, lag.max = 50)
```



From the **ACF** plot, we can see that the values of correlation have not dropped to **zero** even after **50 lags** which is indicative of **non-stationary series.**

- ## Augmented Dickey-Fuller Test:

It is a formal test to check whether a **time series** is **stationary or non-stationary.**

H_0: Time series is non-stationary
H_1: Time series is stationary

The function **adf()** can be used to perform the test. The **p-value** of the test determines whether the series is stationary or not.

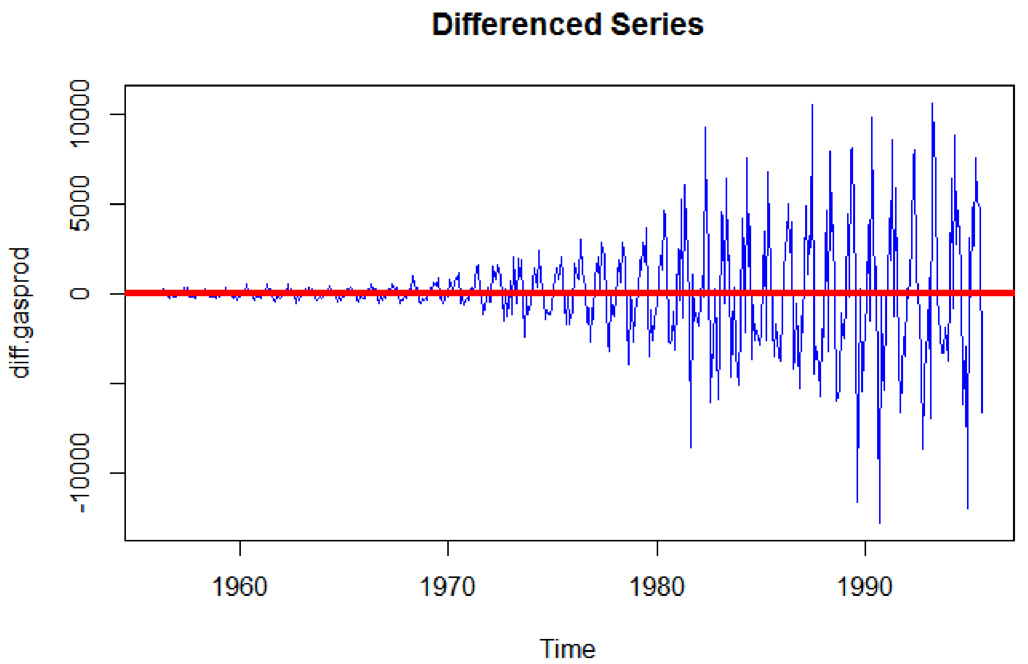
```
> #### Checking for  stationarity using Augmented Dicky- Fuller Test ####
> adf = adf.test(gasprod,alternative = "stationary")
> adf

	Augmented Dickey-Fuller Test

data:  gasprod
Dickey-Fuller = -2.7131, Lag order = 7, p-value = 0.2764
alternative hypothesis: stationary

> print(adf$alternative)
[1] "stationary"
> if(adf$p.value < 0.05){
+   print("The series is Stationary & Null Hypothesis is rejected")
+ } else{
+   print("The Series is not Stationary & Null Hypothesis is accepted")
+ }
[1] "The Series is not Stationary & Null Hypothesis is accepted"
```

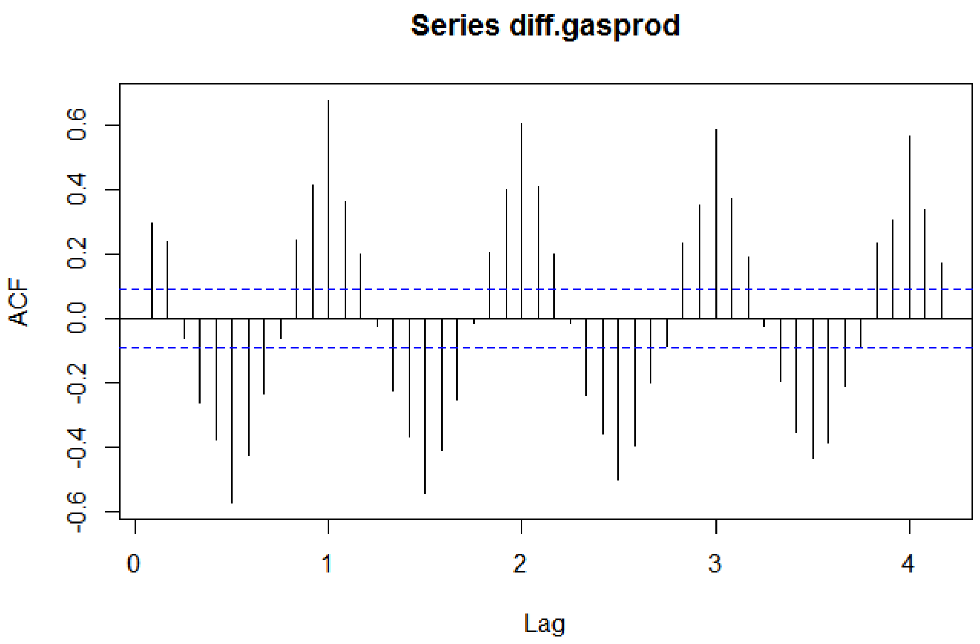
As we can see from the above results, the **p-value** is **0.27** and we **reject the alternative** **hypothesis** that the **series is stationary.**

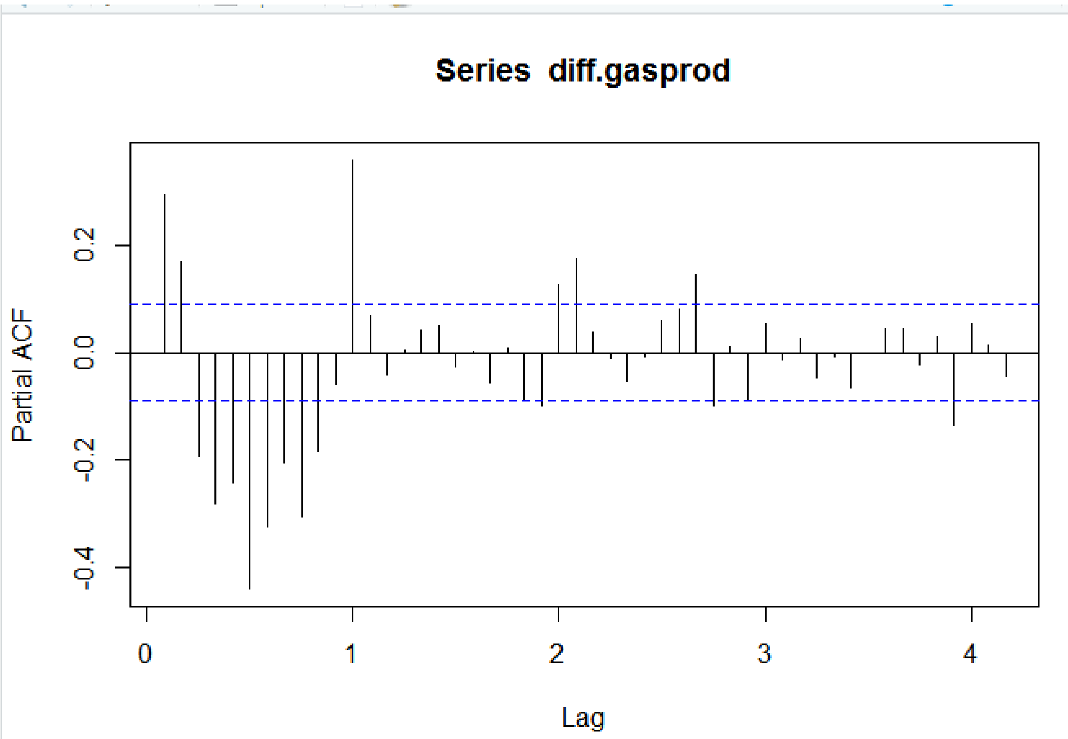
b) Converting the series into stationary:

The time series data needs to be converted to **stationary** in order to perform a **Manual ARIMA.** The **series** can be converted to **stationary series** by the process of **differencing.** It is the process in which a new series is got by computing the differences between consecutive observations. This can be done in R with the help of the function **diff()** with the **argument differences = 1** since from the **PACF and ACF** plots, it is evident the differencing value should be taken as **1.**

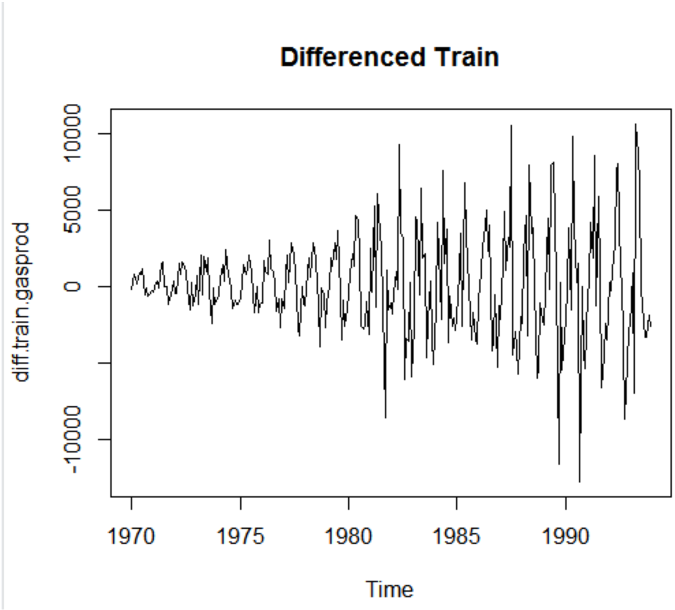
```
#### Stationarizing the series for performing Manual Arima ####
diff.gasprod = diff(gasprod, differences = 1)
ts.plot(diff.gasprod,col = c("Blue"),lwd = 1,main = "Differenced Series")
abline(a=1,b=0,col = c("Red"),lwd =4)
```

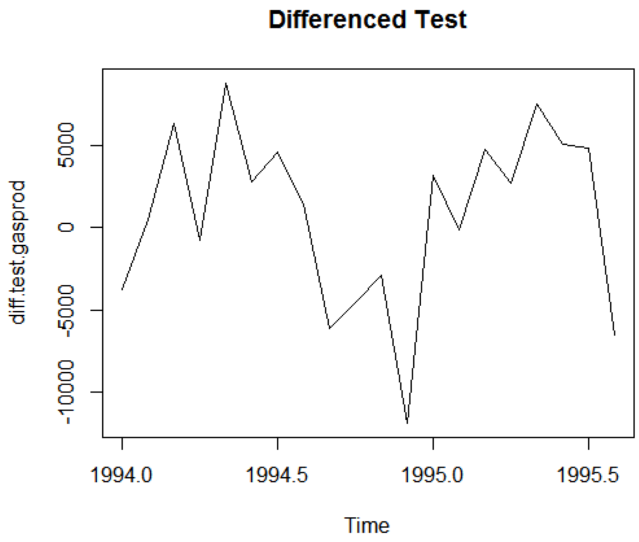







```
> adf.d = adf.test(diff.gasprod,alternative = "stationary")
Warning message:
In adf.test(diff.gasprod, alternative = "stationary") :
  p-value smaller than printed p-value
> adf.d

        Augmented Dickey-Fuller Test

data:  diff.gasprod
Dickey-Fuller = -19.321, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary

> print(adf.d$alternative)
[1] "stationary"
> if(adf.d$p.value < 0.05){
+   print("The series is Stationary & Null Hypothesis is rejected")
+ } else{
+   print("The Series is not Stationary & Null Hypothesis is accepted")
+ }
[1] "The series is Stationary & Null Hypothesis is rejected"
> ### Splitting of time series into Training and Testing Data ####
> diff.train.gasprod = window(diff.gasprod,start = c(1970,1),end = c(1993,12),frequency = 12)
> diff.test.gasprod = window(diff.gasprod,start = c(1994,1),end = c(1995,8),frequency = 12)
> plot.ts(diff.train.gasprod,main = "Differenced Train")
> plot.ts(diff.test.gasprod,main = "Differenced Test")
```



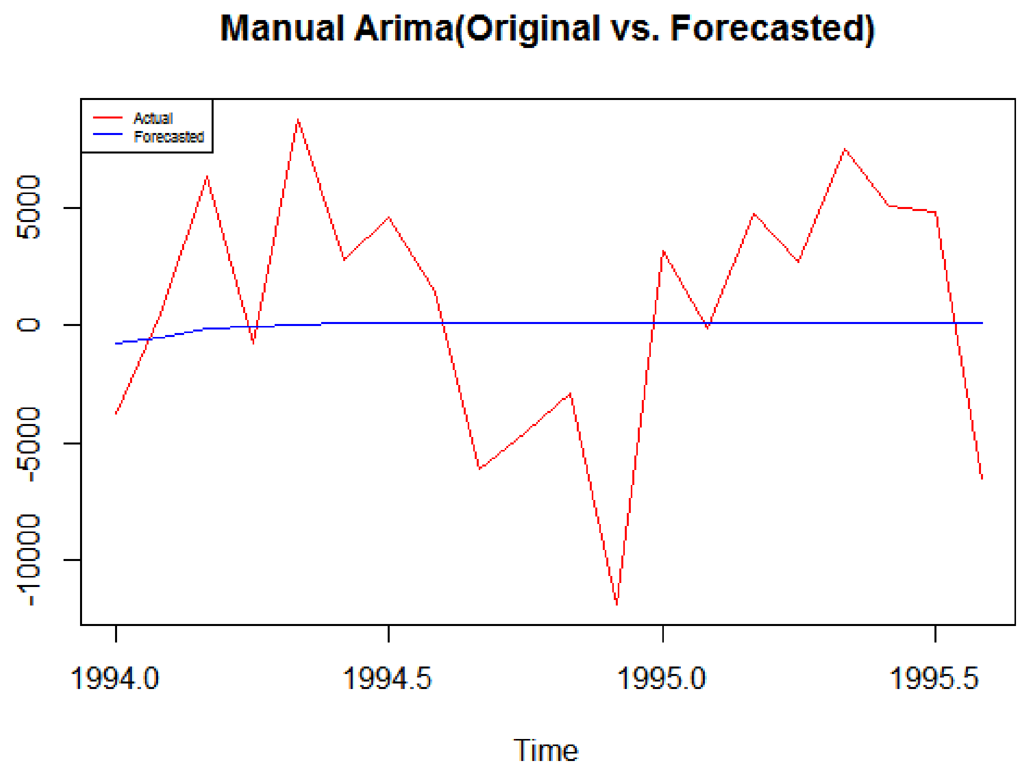


c) Performing Manual ARIMA:

The **Manual ARIMA** requires three parameters namely **the differencing factor(d), the Moving average factor(q) and the autoregressive term (p).** The **d term** has already been calculated during **differencing** which is **1.** From the **PACF and ACF, the p term and q term are 2 and 2 respectively since the lag values in ACF are significant only till 2.** We can use the function **arima()** with the arguments **(2,1,2)**.

```
> marima.gasprod = Arima(diff.train.gasprod,order = c(2,1,2))
> marima.gasprod
Series: diff.train.gasprod
ARIMA(2,1,2)

Coefficients:
          ar1     ar2      ma1      ma2
      -0.0131  0.2545  -0.7212  -0.2788
s.e.   0.1623  0.0695   0.1595   0.1593

sigma^2 estimated as 9895599:  log likelihood=-2719.06
AIC=5448.12   AICc=5448.33   BIC=5466.41
> forc.marima.gasprod = forecast(marima.gasprod,h = 20)
```

```
> ts.plot(diff.test.gasprod,forc.marima.gasprod$mean,gpars = list(col = c("Red","Blue"),
+                                                      main = "Manual Arima(Original vs. Forecaste
d)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
```




```
> vec.marima = cbind(diff.test.gasprod,forc.marima.gasprod$mean)
> MAPE.marima = mean(abs(vec.marima[,1]-vec.marima[,2])/vec.marima[,1])
> print(MAPE.marima)
[1] 0.1915037
```

Inferences:

We can see that the **MAPE(0.19)** is **very high** and **AIC and BIC** values are on the **higher side.** The **plot** shows that **forecasted**

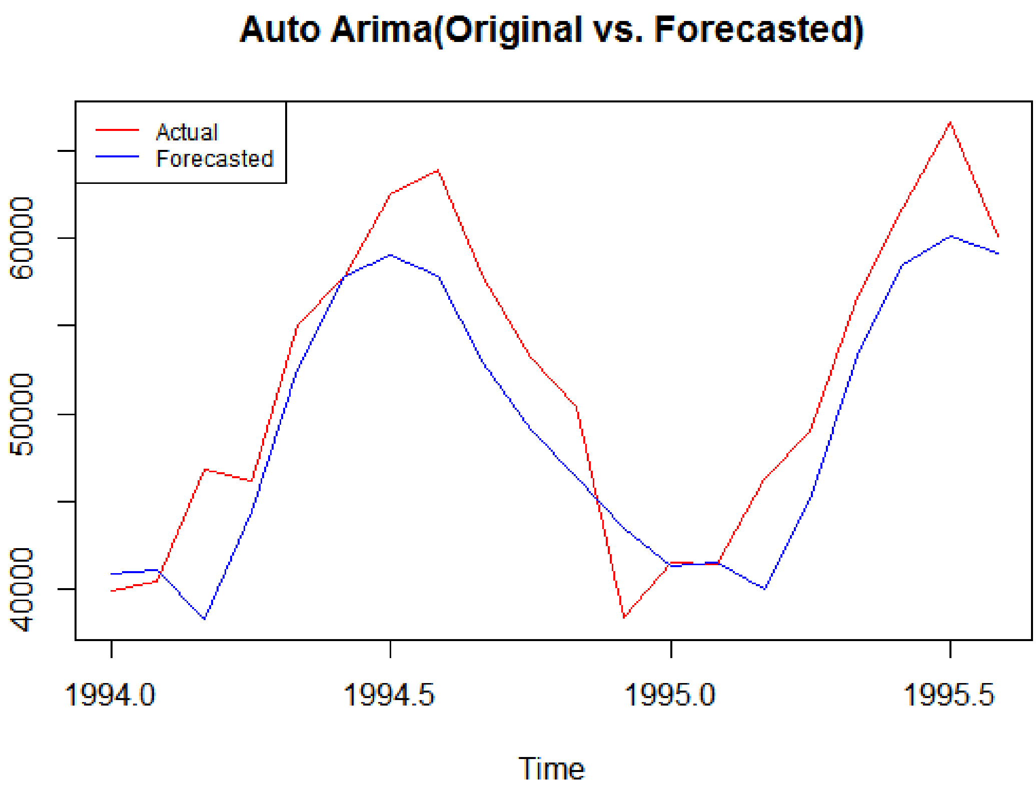
values and **actual values** don't match. This is due to **multiplicative seasonality** that is present in the dataset.

d)Performing Auto ARIMA:

In the **Auto ARIMA,** we need not convert the series into **stationery series** and also we need not determine the values of **p, d and q.** The function **auto.arima() with argument seasonal = TRUE** can be used to perform **Auto ARIMA**.

```
> ###USING AUTO ARIMA (Original) ####
> aarima.gasprod = auto.arima(train.gasprod,seasonal = TRUE)
> aarima.gasprod
Series: train.gasprod
ARIMA(2,1,1)(0,1,2)[12]

Coefficients:
         ar1     ar2      ma1     sma1     sma2
      0.5017  0.2057  -0.9583  -0.4404  -0.1236
s.e.  0.0738  0.0722   0.0426   0.0676   0.0639

sigma^2 estimated as 3535010:  log likelihood=-2463.67
AIC=4939.33   AICc=4939.64   BIC=4961.03
> forc.aarima = forecast(aarima.gasprod,h = 20)
```

```
> forc.aarima = forecast(aarima.gasprod,h = 20)
> ts.plot(test.gasprod,forc.aarima$mean,gpars = list(col = c("Red","Blue"),
+                                                   main = "Auto Arima(Original vs. Fo
recasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
> |
```




```
> vec.aarima = cbind(test.gasprod,forc.aarima$mean)
> MAPE.aarima = mean(abs(vec.aarima[,1]-vec.aarima[,2])/vec.aarima[,:
> print(MAPE.aarima)
[1] 0.06370687
```

Inferences:

We can see that the **MAPE(0.06)** is **very low** and the **plot** shows that **forecasted values** and **actual values** actually match. Even the **AIC and BIC values** are **pretty low.** This is due to the reason being the **auto.arima()** using the **best iterations of p,d and q** values to determine the best model.

e) Model Comparison:

For the **time series** data, the following **model measures** are to be kept in mind for comparison:

- **AIC – Akaike Information Criterion** is an important **model measure** which actually helps in model comparison rather than giving us the model performance as it is. **AIC** is the measure of how much information is being lost by the model while it tries to explain the process. **The lesser the value, the better the model is in prediction.**

- **BIC – Bayesian Information Criterion is** an important **model comparison measure.** It is the **posterior probability in the Bayesian setup,** which gives the **likelihood of the model** to the **true model.** Therefore, **lesser the value, better the model is in likelihood.**
- **MAPE – Mean Absolute Percentage Error** is the measure of the prediction accuracy of a forecasting method which is represented in the form of **percentage. The lesser the value, better the model that fits.**

We have created several **forecasting methods** to **forecast values** using the **training data, original and deseasonalized,** to forecast values and compare it with **testing data, original and deseasonalized.** We have also created several **regression models** using the **differenced data** for **Manual ARIMA** and **original data** for **Auto ARIMA.**

Model	AIC	BIC	MAPE
Random Walk with Drift	NA	NA	7%
Simple Exponential Smoothing - Original Data	6309	6320	17%
Simple Exponential Smoothing - Deseasonalized Data	6098	6109	11%
Double Exponential Smoothing - Original Data	6344	6362	62%
Double Exponential Smoothing - Deseaonalized Data	6101	6119	10%
Holt-Winter's Method - Original Data	**6057**	**6119**	**4.60%**
Holt-Winter's Method - Deseasonalized Data	**6057**	**6119**	***4.50%***
Manual ARIMA - Differenced Data	5448	5466	19%
Auto ARIMA - Original Data	***4939***	***4961***	**6%**

Inferences:

We can see that out of all these models, the **Holt-Winter's Model** with **Deseasonalized data** has the **lowest MAPE value of 4.5%**. But comparing the **Holt-Winter's Model** with **Original data,** it isn't much difference, therefore we can consider **the model with original data over the deseasonalized one.**

The lowest AIC and BIC values belong to the **Auto ARIMA model with 4939 and 4961 respectively.**

So we are now faced with the question of whether to choose whether **Holt-Winter's model or Auto ARIMA.**

Model	AIC	BIC	MAPE
Holt-Winter's Method - Original Data	6057	6119	4.60%
Auto ARIMA - Original Data	4939	4961	6%

If we compare both of them side-by-side, we can see that for a **small change in MAPE value (1.4%),** we are compromising a **huge amount of AIC and BIC values respectively (around 1000).** And also we need to consider that real low values of **MAPE** can cause **over fitting.** While **Holt-Winter's model take only trend and seasonality into account, ARIMA considers Moving Average, Difference Term, Auto Regressive Term and also Error term** which is an important aspect since as we saw in the decomposition plot that **error term had a significant impact on the time series data.**

Keeping in mind the above reasons, we consider Auto ARIMA model to be the best model to be used for forecasting.

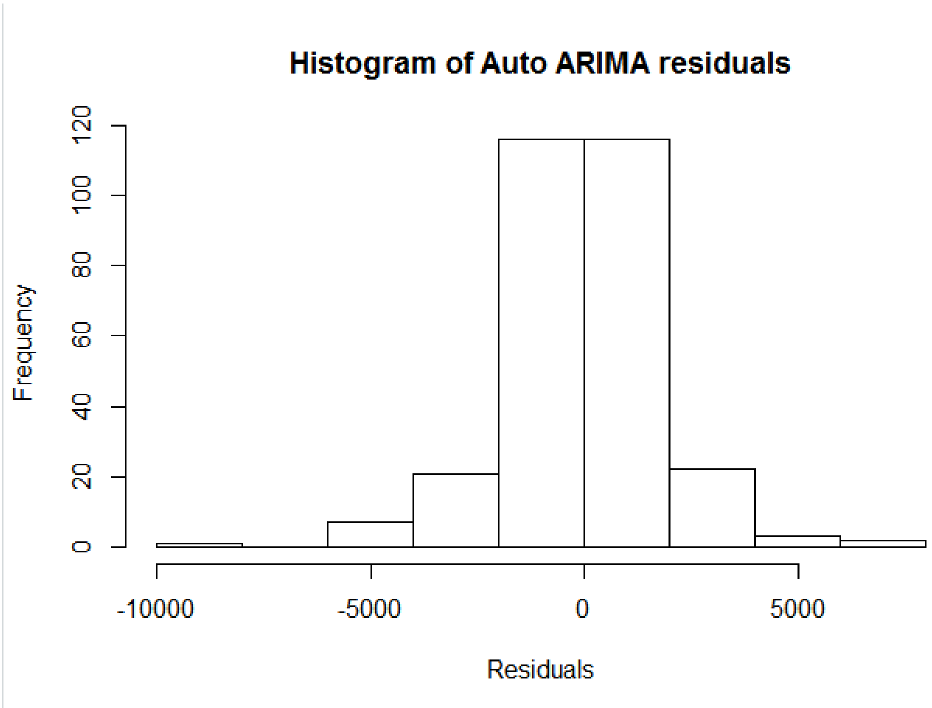
f) Box Ljung Test:

The **Box Ljung Test** is used to check whether the **residuals are following the white noise or not.** It checks whether the **residuals are stationary or not.**

H0: Residuals are stationary

H1: Residuals are not stationary

```
> ### Ljung box test ####
> Box.test(aarima.gasprod$residuals,type = "Ljung-Box")

	Box-Ljung test

data:  aarima.gasprod$residuals
X-squared = 0.009919, df = 1, p-value = 0.9207

> hist(aarima.gasprod$residuals,main = "Histogram of Auto ARIMA residuals",
+      xlab = "Residuals")
```



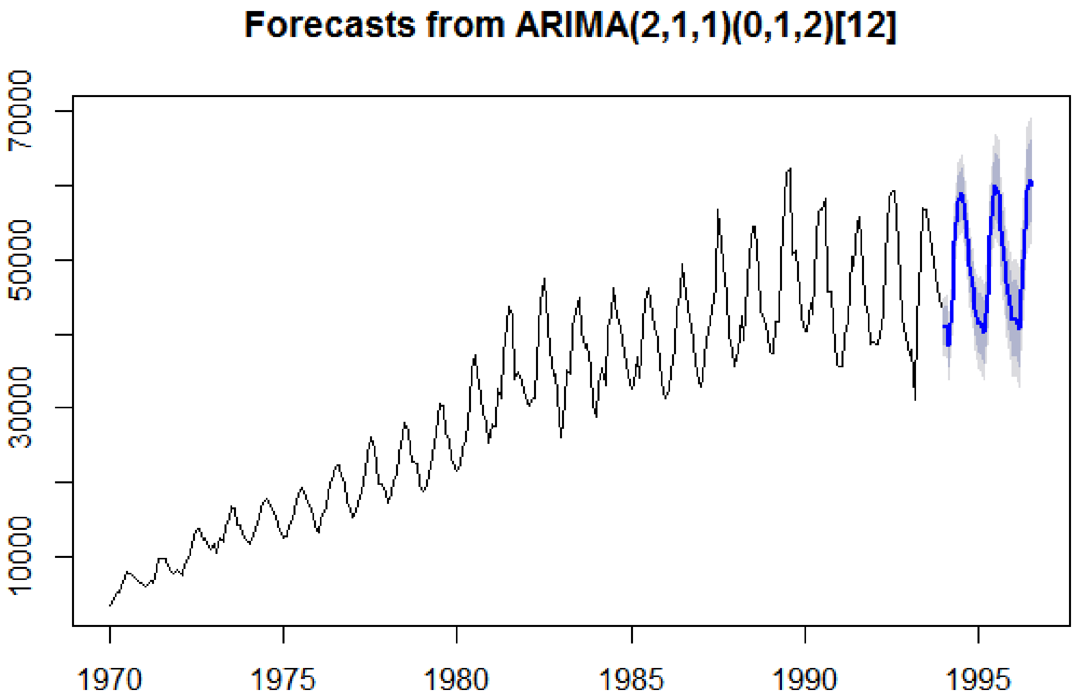
From the above results, we can see that the **P-Value** is more than **0.05** and hence we **reject null hypothesis** and conclude that the **residuals are stationary** concluding they are **independent**. We can also see that the **residuals** follow a

normal distribution (Bell Curve) meaning the **residuals** are **stationary.**

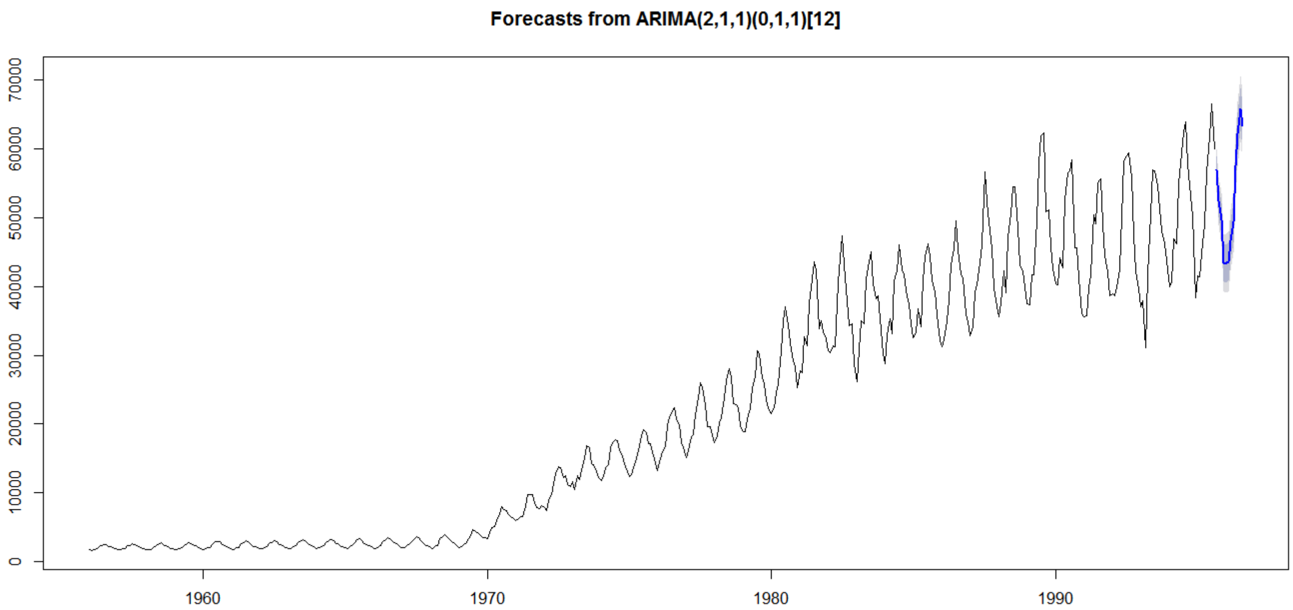
g) Making final forecasts:

As we have concluded that **Auto ARIMA gives the best model,** we use it to create two models, the one with data ranging from **1970 January to 1993 December (Original Training Data) and 1970 January to 1995 August (Original Training Data and Testing Data)**. We make the **forecast** for **next 12 time periods** i.e. **1995 September to 1996 August.**

```
> ### Making the future forecastusing the best model ####
> future = forecast(aarima.gasprod,h = 32)
> plot(future)
> future$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
1994 40911.17 41136.45 38318.03 44418.69 52575.33 57827.42 59054.39 57800.39
1995 41314.97 41577.63 40068.39 45141.64 53284.34 58496.89 60063.56 59142.29
1996 41882.79 42199.58 40744.41 45855.92 54028.93 59264.56 60849.03 59941.44
          Sep      Oct      Nov      Dec
1994 52923.33 49148.06 46382.99 43491.75
1995 54005.17 49942.73 46875.98 43928.71
1996
>
```



```
> ### Making the future forecast using the model created from gasprod ####
> aarima.fgasprod = auto.arima(gasprod,seasonal = TRUE)
> fgasprod.forc = forecast(aarima.fgasprod,h = 12)
> plot(fgasprod.forc)
> fgasprod.forc$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov
1995                                                                            56907.83 52476.28 49719.79
1996 43318.41 43601.71 46668.11 49376.36 57536.25 62184.69 65795.74 63391.54
          Dec
1995 43473.70
1996
```



6) Project Conclusion:

We were given a **time series data** of the **monthly Australian Gas Production** and were asked to **build a model to best forecast the future values beyond the time periods that data contained.** Various methods such as **decomposition** were used to analyse every component of the time series of data and understand the data better before we go for building a model. Then we applied **deseasonalization** method on the time series and **split the data into training and testing data for both original and seasonalized data.**

After **using many forecasting methods and building regression models,** we came to conclude that **Auto ARIMA** gave us the **best model** when compared with **other** models using **model measures such as AIC, BIC and MAPE.** Then using this method, we presented the **forecasts** for the **next 12 time periods.**

As we performed the analysis, following things were noted,

- There was **major change** in the **time series data** from the **year 1970.** An **upward trend** started from that time period onwards. This could be due to the **energy crisis of 1970** where the **prices of all the energy sources went up** and the **countries were forced to put more effort into increasing the production of energy sources.**
- The **upward trend** from the year **1970** could also be due to the **technology advancements** that might have occurred in the year **1970** helping in more production of gas.
- There has never been a **decrease** in the **production value** of **gas** from **its preceding year.**
- The **highest value** is in the **year 1995** which was **66600 units** whereas **the lowest value** was in the **year 1956** which was **1646 units.** Even though it has been **39 years,** the **production values** increased by **margin of only 2.4%**
- We can see that there is a **seasonal increase** in the **months of July** which are the **coldest winter months** in **Australia.** The production **increases** in **winter** because **people require more natural gas to fight the cold weather in winters as compared to other months in other seasons.** Hence to meet the **demand,** the production of the natural gas **increases.**

The following **suggestions** can be provided based on the analysis of the data:

- The data collected could have been a **daily production data** as compared to **monthly data** to make **better forecasts.**
- The data had a **major incident affecting** the **analysis** and also the **model building** which could have been avoided by taking a **different time period.**
- It was nowhere mentioned that whether the **monthly production values** taken were either **average** of every day **gas production** or **total** of **30 days gas production.** A better analysis could have been done had this **information** been known.

7) Appendix – A (Source Code):

AUSTRALIAN GAS PRODUCTION

```
############# ANALYSIS OF AUSTRALIAN GAS PRODUCTION #########


### Invoking of the necessary libraries ####
install.packages('forecast')
library(forecast)
install.packages('tseries')
library(tseries)
install.packages('ggplot2')
library(ggplot2)
install.packages('dygraphs')
library(dygraphs)
install.packages('xts')
library(xts)
install.packages('fts')
library(fts)
install.packages('TSA')
library(TSA)
install.packages('Metrics')
library(Metrics)
> ### As the dataset to be used for the analysis is present in the ####
> ### forecast library, the data can be called directly using ###
> ### the data name 'gas'. For our analysis, we import it and
> ### name it 'gasprod' ###
> gasprod = gas
> ### Viewing the data ####
> print(gasprod)
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1956	1709	1646	1794	1878	2173	2321	2468	2416	2184	2121	1962	1825
1957	1751	1688	1920	1941	2311	2279	2638	2448	2279	2163	1941	1878
1958	1773	1688	1783	1984	2290	2511	2712	2522	2342	2195	1931	1910
1959	1730	1688	1899	1994	2342	2553	2712	2627	2363	2311	2026	1910
1960	1762	1815	2005	2089	2617	2828	2965	2891	2532	2363	2216	2026
1961	1804	1773	2015	2089	2627	2712	3007	2880	2490	2237	2205	1984
1962	1868	1815	2047	2142	2743	2775	3028	2965	2501	2501	2131	2015
1963	1910	1868	2121	2268	2690	2933	3218	3028	2659	2406	2258	2057
1964	1889	1984	2110	2311	2785	3039	3229	3070	2659	2543	2237	2142
1965	1962	1910	2216	2437	2817	3123	3345	3112	2659	2469	2332	2110
1966	1910	1941	2216	2342	2923	3229	3513	3355	2849	2680	2395	2205
1967	1994	1952	2290	2395	2965	3239	3608	3524	3018	2648	2363	2247
1968	1994	1941	2258	2332	3323	3608	3957	3672	3155	2933	2585	2384
1969	2057	2100	2458	2638	3292	3724	4652	4379	4231	3756	3429	3461
1970	3345	4220	4874	5064	5951	6774	7997	7523	7438	6879	6489	6288
1971	5919	6183	6594	6489	8040	9715	9714	9756	8595	7861	7753	8154
1972	7778	7402	8903	9742	11372	12741	13733	13691	12239	12502	11241	10829
1973	11569	10397	12493	11962	13974	14945	16805	16587	14225	14157	13016	12253
1974	11704	12275	13695	14082	16555	17339	17777	17592	16194	15336	14208	13116
1975	12354	12682	14141	14989	16159	18276	19157	18737	17109	17094	15418	14312
1976	13260	14990	15975	16770	19819	20983	22001	22337	20750	19969	17293	16498
1977	15117	16058	18137	18471	21398	23854	26025	25479	22804	19619	19627	18488
1978	17243	18284	20226	20903	23768	26323	28038	26776	22886	22813	22404	19795
1979	18839	18892	20823	22212	25076	26884	30611	30228	26762	25885	23328	21930
1980	21433	22369	24503	25905	30605	34984	37060	34502	31793	29275	28305	25248
1981	27730	27424	32684	31366	37459	41060	43558	42398	33827	34962	33480	32445
1982	30715	30400	31451	31306	40592	44133	47387	41310	37913	34355	34607	28729
1983	26138	30745	35018	34549	40980	42869	45022	40387	38180	38608	35308	30234
1984	28801	33034	35294	33181	40797	42355	46098	42430	41851	39331	37328	34514
1985	32494	33308	36805	34221	41020	44350	46173	44435	40943	39269	35901	32142
1986	31239	32261	34951	38109	43168	45547	49568	45387	41805	41281	36068	34879
1987	32791	34206	39128	40249	43519	46137	56709	52306	49397	45500	39857	37958
1988	35567	37696	42319	39137	47062	50610	54457	54435	48516	43225	42155	39995

```
1989 37541 37277 41778 41666 49616 57793 61884 62400 50820 51116 45731 425
28
1990 40459 40295 44147 42697 52561 56572 56858 58363 45627 45622 41304 360
16
1991 35592 35677 39864 41761 50380 49129 55066 55671 49058 44503 42145 386
98
1992 38963 38690 39792 42545 50145 58164 59035 59408 55988 47321 42269 396
06
1993 37059 37963 31043 41712 50366 56977 56807 54634 51367 48073 46251 437
36
1994 39975 40478 46895 46147 55011 57799 62450 63896 57784 53231 50354 384
10
1995 41600 41471 46287 49013 56624 61739 66600 60054
> ### Checking the class of the imported ####
 class(gasprod)
[1] "ts"
 ### Inspection of the time series data ####
 summary(gasprod)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1646    2675   16788   21415   38629   66600
 anyNA(gasprod)
[1] FALSE
> findfrequency(gasprod)
[1] 12
> ts.plot(gasprod,gpars = list(xlab = "Year",ylab = "Gas Production",
+                               main = "Australian Gas Production (1956-199
5)",
+                               col = c("Blue")),lwd = 2)
> stepplot = dygraph(gasprod,main = "Australian Gas Production (1956-1995)
",
+                     xlab = "Year",ylab = "Gas Production") %>%dyOptions(s
tepPlot= TRUE,pointSize = 0,fillGraph = TRUE,
+                                           fillAlpha = 0.2)
> stepplot
> ### Inspection of individual elements by decomposition of time series ##
##
> dc.gasprod = stl(gasprod,s.window = "periodic")
> plot(dc.gasprod)
> monthplot(gasprod,main = "Month Plot for Australian Gas Production")
> seasonplot(gasprod,year.labels = TRUE,
+            main = "Month Plot for Australian Gas Production")
> ### De-Seasonalizing the time series from the decomposed time series####
> ds.gasprod = (dc.gasprod$time.series[,2]+dc.gasprod$time.series[,3])
> ts.plot(ds.gasprod,gpars = list(xlab = "Year",ylab = "Gas Production",
+                               main = "Australian Gas Production (Deseason
lized)",
+                               col = c("Red")),lwd = 2)
> ts.plot(ds.gasprod,gasprod,gpars = list(xlab = "Year",ylab = "Gas Produc
tion",
+                                 main = "Australian Gas Production (Deseas
onlized
+                                 vs. Original)",
+                                 col = c("Red","Blue")),lwd = 2)
> ### Splitting the time series into training and testing samples (Origina
l) ####
> train.gasprod = window(gasprod,start=c(1970,1), end=c(1993,12), freq=12)
> ts.plot(train.gasprod)
> test.gasprod = window(gasprod,start=c(1994,1),end=c(1995,8), freq=12)
> ts.plot(test.gasprod)
> autoplot(train.gasprod, series="Train") + autolayer(test.gasprod, series
="Test") +
+   ggtitle("Gas Produciton Traning and Test data") +
+   xlab("Year") + ylab("Production") +
+   guides(colour=guide_legend(title="Forecast"))
> ### Splitting the time series into training and testing samples(Deseason
alize) ####
> train.ds.gasprod = window(ds.gasprod,start=c(1970,1), end=c(1993,12), fr
eq=12)
> ts.plot(train.ds.gasprod)
```

```
> test.ds.gasprod = window(ds.gasprod,start=c(1994,1),end=c(1995,8), freq=
12)
> ts.plot(test.ds.gasprod)
> autoplot(train.ds.gasprod, series="Train") + autolayer(test.ds.gasprod,
series="Test") +
+   ggtitle("Gas Produciton Traning and Test data(Deseasonalized)") +
+   xlab("Year") + ylab("Production") +
+   guides(colour=guide_legend(title="Forecast"))
> ### Checking the periodicity of the Time Series ####
> periodicity(gasprod)
Monthly periodicity from Jan 1956 to Aug 1995
> findfrequency(gasprod)
[1] 12
> periodicity(train.gasprod)
Monthly periodicity from Jan 1970 to Dec 1993
> findfrequency(train.gasprod)
[1] 12
> periodicity(test.gasprod)
Monthly periodicity from Jan 1994 to Aug 1995
> findfrequency(test.gasprod)
[1] 12
> #### Naive Method ####
> gasprod.rw = stl(train.gasprod,s.window = 'p')
> gasprod.rw = forecast(de.gasprod.rw,method = "rwdrift",h = 20)
> ts.plot(test.gasprod,gasprod.rw$mean,gpars = list(col = c("Red","Blue"),
+         main = "Random Walk with Drift(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
> vec.rw = cbind(test.gasprod,gasprod.rw$mean)
> MAPE.rw = mean(abs(vec.rw[,1]-vec.rw[,2])/vec.rw[,1])
> print(MAPE.rw)
[1] 0.0737419
> #### Simple Exponential Smoothing (Original) ####
> gasprod.ses = ses(train.gasprod,start = c(1970,1),end = c(1993,12),frequ
ency = 12,h = 20)
> summary(gasprod.ses)

Forecast method: Simple exponential smoothing

Model Information:
Simple exponential smoothing

Call:
 ses(y = train.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), frequency = 12)

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = 6002.9039

  sigma:  3344.025

     AIC     AICc      BIC
6309.126 6309.210 6320.115

Error measures:
                   ME     RMSE      MAE       MPE     MAPE      MASE
ACF1
Training set 131.0317 3332.394 2416.021 0.1509165 8.113316 0.9164853 0.303
0855

Forecasts:
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jan 1994       43736.25 39450.71 48021.79 37182.08 50290.42
```

```
Feb 1994       43736.25 37675.88 49796.62 34467.72 53004.78
Mar 1994       43736.25 36313.97 51158.53 32384.85 55087.65
Apr 1994       43736.25 35165.81 52306.69 30628.90 56843.61
May 1994       43736.25 34154.26 53318.25 29081.86 58390.65
Jun 1994       43736.25 33239.74 54232.76 27683.22 59789.28
Jul 1994       43736.25 32398.75 55073.75 26397.04 61075.47
Aug 1994       43736.25 31615.97 55856.53 25199.88 62272.62
Sep 1994       43736.25 30880.77 56591.73 24075.49 63397.01
Oct 1994       43736.25 30185.40 57287.10 23012.02 64460.49
Nov 1994       43736.25 29524.01 57948.49 22000.51 65471.99
Dec 1994       43736.25 28892.06 58580.44 21034.03 66438.48
Jan 1995       43736.25 28285.94 59186.56 20107.04 67365.46
Feb 1995       43736.25 27702.72 59769.79 19215.07 68257.43
Mar 1995       43736.25 27139.97 60332.53 18354.43 69118.07
Apr 1995       43736.25 26595.70 60876.81 17522.03 69950.47
May 1995       43736.25 26068.18 61404.33 16715.26 70757.24
Jun 1995       43736.25 25555.96 61916.54 15931.89 71540.61
Jul 1995       43736.25 25057.78 62414.72 15170.00 72302.50
Aug 1995       43736.25 24572.55 62899.95 14427.90 73044.60
> print(gasprod.ses$mean)
          Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug      Sep      Oct
1994 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.
25 43736.25 43736.25
1995 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.25 43736.
25
          Nov      Dec
1994 43736.25 43736.25
1995
> ts.plot(test.gasprod,gasprod.ses$mean,gpars = list(col = c("Red","Blue")
,
+                                                   main = "Simple Exponen
tial Smoothing(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
> vec.ses = cbind(test.gasprod+gasprod.ses$mean)
> MAPE.ses = mape(test.gasprod,gasprod.ses$mean)
> print(MAPE.ses)
[1] 0.1726068
> #### Simple Exponential Smoothing (Deseasonalized) ####
> gasprod.dses = ses(train.ds.gasprod,start = c(1970,1),end = c(1993,12),f
requency = 12,h = 20)
> summary(gasprod.dses)

Forecast method: Simple exponential smoothing

Model Information:
Simple exponential smoothing

Call:
 ses(y = train.ds.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), frequency = 12)

  Smoothing parameters:
    alpha = 0.9833

  Initial states:
    l = 7432.9827

  sigma:  2319.347

     AIC     AICc      BIC
6098.373 6098.458 6109.362

Error measures:
```

```
                      ME     RMSE      MAE       MPE     MAPE      MASE
ACF1
Training set 139.963 2311.279 1591.754 0.2300846 6.077039 0.6038107 -0.005
553574

Forecasts:
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+         box.lwd = 0.1,cex = 0.75)
> vec.ses = cbind(test.gasprod+gasprod.ses$mean)
> MAPE.ses = mape(test.gasprod,gasprod.ses$mean)
> print(MAPE.ses)
[1] 0.1726068
> #### Simple Exponential Smoothing (Deseasonalized) ####
> gasprod.dses = ses(train.ds.gasprod,start = c(1970,1),end = c(1993,12),f
requency = 12,h = 20)
> summary(gasprod.dses)

Forecast method: Simple exponential smoothing

Model Information:
Simple exponential smoothing

Call:
 ses(y = train.ds.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), frequency = 12)

  Smoothing parameters:
    alpha = 0.9833

  Initial states:
    l = 7432.9827

  sigma:  2319.347

     AIC     AICc      BIC
6098.373 6098.458 6109.362

Error measures:
                      ME     RMSE      MAE       MPE     MAPE      MASE
ACF1
Training set 139.963 2311.279 1591.754 0.2300846 6.077039 0.6038107 -0.005
553574

Forecasts:
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jan 1994       47067.77 44095.41 50040.13 42521.93 51613.60
Feb 1994       47067.77 42899.24 51236.30 40692.55 53442.99
Mar 1994       47067.77 41976.76 52158.78 39281.74 54853.80
Apr 1994       47067.77 41197.50 52938.04 38089.96 56045.58
May 1994       47067.77 40510.19 53625.35 37038.82 57096.72
Jun 1994       47067.77 39888.39 54247.15 36087.85 58047.69
Jul 1994       47067.77 39316.30 54819.24 35212.92 58922.62
Aug 1994       47067.77 38783.63 55351.91 34398.27 59737.27
Sep 1994       47067.77 38283.20 55852.34 33632.93 60502.61
Oct 1994       47067.77 37809.78 56325.76 32908.89 61226.65
Nov 1994       47067.77 37359.42 56776.12 32220.12 61915.41
Dec 1994       47067.77 36929.04 57206.50 31561.92 62573.62
Jan 1995       47067.77 36516.20 57619.34 30930.54 63205.00
Feb 1995       47067.77 36118.92 58016.62 30322.95 63812.59
Mar 1995       47067.77 35735.56 58399.98 29736.65 64398.89
Apr 1995       47067.77 35364.75 58770.79 29169.54 64966.00
May 1995       47067.77 35005.33 59130.21 28619.86 65515.68
Jun 1995       47067.77 34656.32 59479.22 28086.09 66049.45
Jul 1995       47067.77 34316.85 59818.69 27566.92 66568.61
Aug 1995       47067.77 33986.19 60149.34 27061.23 67074.31
> print(gasprod.dses$mean)
```

```
          Jan       Feb       Mar       Apr       May       Jun       Jul       A
ug       Sep       Oct
1994 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.
77 47067.77 47067.77
1995 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.77 47067.
77
          Nov       Dec
1994 47067.77 47067.77
1995
> ts.plot(test.ds.gasprod,gasprod.dses$mean,gpars = list(col = c("Red","Bl
ue"),
+                                                     main = "Simple Exponen
tial Smoothing(Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+         box.lwd = 0.1,cex = 0.75)
> MAPE.dses = mape(test.ds.gasprod,gasprod.dses$mean)
> print(MAPE.dses)
[1] 0.1110184
> #### Double Exponential Method (Holt Model) (Originial) ####
> gasprod.holt = holt(train.gasprod ,start=c(1970,1),end=c(1993,12), freq=
12,h=20)
> summary(gasprod.holt)

Forecast method: Holt's method

Model Information:
Holt's method

Call:
 holt(y = train.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), freq = 12)

  Smoothing parameters:
    alpha = 0.9436
    beta  = 0.5881

  Initial states:
    l = 3775.0087
    b = 467.2663

  sigma:  3542.179

     AIC     AICc      BIC
6344.263 6344.476 6362.578

Error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE
ACF1
Training set -16.95919 3517.494 2436.963 0.2550371 7.909274 0.9244294 -0.0
1120745

Forecasts:
         Point Forecast      Lo 80    Hi 80      Lo 95     Hi 95
Jan 1994     41337.3687  36797.884 45876.85  34394.826  48279.91
Feb 1994     38932.0524  30628.005 47236.10  26232.108  51632.00
Mar 1994     36526.7362  23815.946 49237.53  17087.260  55966.21
Apr 1994     34121.4199  16438.554 51804.29   7077.810  61165.03
May 1994     31716.1037   8552.086 54880.12  -3710.204  67142.41
Jun 1994     29310.7874    199.210 58422.36 -15211.528  73833.10
Jul 1994     26905.4711  -8586.447 62397.39 -27374.733  81185.68
Aug 1994     24500.1549 -17777.552 66777.86 -40158.018  89158.33
Sep 1994     22094.8386 -27351.339 71541.02 -53526.565  97716.24
Oct 1994     19689.5224 -37288.475 76667.52 -67450.805 106829.85
Nov 1994     17284.2061 -47572.276 82140.69 -81905.224 116473.64
Dec 1994     14878.8899 -58188.160 87945.94 -96867.519 126625.30
Jan 1995     12473.5736 -69123.236 94070.38 -112317.977 137265.12
```

```
Feb 1995     10068.2573  -80366.005 100502.52 -128239.012 148375.53
Mar 1995      7662.9411  -91906.127 107232.01 -144614.807 159940.69
Apr 1995      5257.6248 -103734.234 114249.48 -161431.039 171946.29
May 1995      2852.3086 -115841.794 121546.41 -178674.656 184379.27
Jun 1995       446.9923 -128220.987 129114.97 -196333.701 197227.69
Jul 1995     -1958.3239 -140864.616 136947.97 -214397.166 210480.52
Aug 1995     -4363.6402 -153766.027 145038.75 -232854.874 224127.59
> gasprod.holt$mean
            Jan        Feb        Mar        Apr        May        Jun
Jul        Aug
1994 41337.3687 38932.0524 36526.7362 34121.4199 31716.1037 29310.7874 269
05.4711 24500.1549
1995 12473.5736 10068.2573  7662.9411  5257.6248  2852.3086   446.9923 -19
58.3239 -4363.6402
            Sep        Oct        Nov        Dec
1994 22094.8386 19689.5224 17284.2061 14878.8899
1995
> ts.plot(test.gasprod,gasprod.holt$mean,gpars = list(col = c("Red","Blue"
),
+                                                     main = "Holt's Method(
Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+         box.lwd = 0.1,cex = 0.75)
> MAPE.holt = mape(test.gasprod,gasprod.holt$mean)
> print(MAPE.holt)
[1] 0.6201079
> #### Double Exponential Method (Holt Model) (Deseasonlaize) ####
> gasprod.dholt = holt(train.ds.gasprod,start = c(1970,1),end = c(1993,12)
,freq = 12,h = 20)
> summary(gasprod.dholt)

Forecast method: Holt's method

Model Information:
Holt's method

Call:
 holt(y = train.ds.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), freq = 12)

  Smoothing parameters:
    alpha = 0.9809
    beta  = 1e-04

  Initial states:
    l = 7297.2832
    b = 70.1218

  sigma:  2324.336

     AIC     AICc      BIC
6101.590 6101.803 6119.905

Error measures:
                   ME     RMSE      MAE        MPE     MAPE      MASE
ACF1
Training set 67.98246 2308.139 1589.453 -0.1147602 6.078146 0.6029375 -0.0
02865917

Forecasts:
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jan 1994       47142.87 44164.12 50121.63 42587.26 51698.49
Feb 1994       47214.95 43042.25 51387.66 40833.36 53596.55
Mar 1994       47287.03 42192.76 52381.31 39496.01 55078.06
Apr 1994       47359.11 41485.99 53232.24 38376.95 56341.28
May 1994       47431.19 40870.91 53991.47 37398.11 57464.27
```

```
Jun 1994       47503.27 40321.17 54685.38 36519.19 58487.36
Jul 1994       47575.35 39821.01 55329.70 35716.11 59434.60
Aug 1994       47647.43 39360.17 55934.70 34973.16 60321.71
Sep 1994       47719.51 38931.49 56507.54 34279.39 61159.64
Oct 1994       47791.59 38529.75 57053.44 33626.82 61956.36
Nov 1994       47863.67 38151.00 57576.34 33009.43 62717.92
Dec 1994       47935.75 37792.19 58079.31 32422.52 63448.99
Jan 1995       48007.83 37450.87 58564.79 31862.36 64153.31
Feb 1995       48079.91 37125.06 59034.76 31325.91 64833.91
Mar 1995       48151.99 36813.13 59490.86 30810.69 65493.29
Apr 1995       48224.07 36513.70 59934.44 30314.61 66133.53
May 1995       48296.15 36225.64 60366.67 29835.89 66756.41
Jun 1995       48368.23 35947.94 60788.52 29373.04 67363.43
Jul 1995       48440.31 35679.76 61200.86 28924.73 67955.89
Aug 1995       48512.39 35420.35 61604.43 28489.84 68534.94
> gasprod.dholt$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug      Sep      Oct
1994 47142.87 47214.95 47287.03 47359.11 47431.19 47503.27 47575.35 47647.
43 47719.51 47791.59
1995 48007.83 48079.91 48151.99 48224.07 48296.15 48368.23 48440.31 48512.
39
          Nov      Dec
1994 47863.67 47935.75
1995
> ts.plot(test.ds.gasprod,gasprod.dholt$mean,gpars = list(col = c("Red","B
lue"),
+                                                 main = "Holt's Method(
Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+         box.lwd = 0.1,cex = 0.75)
> MAPE.dholt = mape(test.ds.gasprod,gasprod.dholt$mean)
> print(MAPE.dholt)
[1] 0.1033223
> #### Holt Winter's method (Original) ####
> gasprod.hw = hw(train.gasprod,start = c(1970,1),end = c(1993,12),freq =
12,h = 20)
> summary(gasprod.hw)

Forecast method: Holt-Winters' additive method

Model Information:
Holt-Winters' additive method

Call:
 hw(y = train.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), freq = 12)

  Smoothing parameters:
    alpha = 0.3408
    beta  = 1e-04
    gamma = 0.5936

  Initial states:
    l = 6253.203
    b = 119.5506
    s = -4511.742 -2141.073 234.0438 2010.202 5919.329 7284.465
           5272.426 2485.985 -2602.642 -3068.389 -5131.983 -5750.62

  sigma:  2109.356

     AIC     AICc      BIC
6057.255 6059.521 6119.525

Error measures:
```

```
                   ME      RMSE       MAE        MPE      MAPE       MASE
ACF1
Training set 78.69136 2049.926 1551.842 0.4623734 7.505829 0.5886704 0.273
8151

Forecasts:
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jan 1994       40279.65 37576.40 42982.90 36145.38 44413.91
Feb 1994       41289.25 38433.25 44145.25 36921.37 45657.13
Mar 1994       38304.76 35303.68 41305.84 33715.01 42894.51
Apr 1994       47712.40 44572.87 50851.93 42910.90 52513.90
May 1994       55810.29 52538.08 59082.50 50805.88 60814.70
Jun 1994       61602.40 58202.61 65002.18 56402.87 66801.92
Jul 1994       61619.91 58097.10 65142.73 56232.23 67007.60
Aug 1994       60595.61 56953.85 64237.37 55026.02 66165.20
Sep 1994       57042.03 53285.03 60799.04 51296.18 62787.88
Oct 1994       51962.18 48093.29 55831.07 46045.23 57879.14
Nov 1994       48460.99 44483.30 52438.68 42377.64 54544.34
Dec 1994       44992.38 40908.73 49076.03 38746.98 51237.78
Jan 1995       41741.45 36938.07 46544.84 34395.31 49087.59
Feb 1995       42751.05 37859.46 47642.65 35270.01 50232.10
Mar 1995       39766.56 34788.27 44744.85 32152.93 47380.20
Apr 1995       49174.20 44110.65 54237.76 41430.17 56918.24
May 1995       57272.09 52124.64 62419.54 49399.75 65144.44
Jun 1995       63064.20 57834.15 68294.25 55065.53 71062.88
Jul 1995       63081.72 57770.30 68393.13 54958.61 71204.83
Aug 1995       62057.41 56665.82 67449.01 53811.68 70303.15
> gasprod.hw$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug      Sep      Oct
1994 40279.65 41289.25 38304.76 47712.40 55810.29 61602.40 61619.91 60595.
61 57042.03 51962.18
1995 41741.45 42751.05 39766.56 49174.20 57272.09 63064.20 63081.72 62057.
41
          Nov      Dec
1994 48460.99 44992.38
1995
> ts.plot(test.gasprod,gasprod.hw$mean,gpars = list(col = c("Red","Blue"),
+                                               main = "Holt Winter's(
Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+          box.lwd = 0.1,cex = 0.75)
> MAPE.hw = mape(test.gasprod,gasprod.hw$mean)
> print(MAPE.hw)
[1] 0.04666037
> #### Holt Winter's method (Deseaonalize) ####
> gasprod.dhw = hw(train.ds.gasprod,start = c(1970,1),end = c(1993,12),fre
q = 12,h = 20)
> summary(gasprod.dhw)

Forecast method: Holt-Winters' additive method

Model Information:
Holt-Winters' additive method

Call:
 hw(y = train.ds.gasprod, h = 20, start = c(1970, 1), end = c(1993,

 Call:
     12), freq = 12)

  Smoothing parameters:
    alpha = 0.3408
    beta  = 1e-04
    gamma = 0.5934

  Initial states:
    l = 6263.4243
```

```
    b = 119.8431
    s = -1190.992 -669.5463 69.8913 481.8762 1663.536 1972.405
           1680.539 771.3246 -793.5156 -903.362 -1416.127 -1666.028

  sigma:  2109.394

     AIC     AICc      BIC
6057.265 6059.531 6119.535

Error measures:
                   ME     RMSE      MAE       MPE     MAPE      MASE
ACF1
Training set 77.89129 2049.962 1551.948 -1.123422 8.141496 0.5887106 0.273
9442

Forecasts:
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jan 1994       44364.14 41660.84 47067.44 40229.80 48498.47
Feb 1994       45005.85 42149.82 47861.88 40637.93 49373.77
Mar 1994       40472.48 37471.40 43473.56 35882.73 45062.24
Apr 1994       49522.52 46383.00 52662.03 44721.04 54323.99
May 1994       54097.59 50825.41 57369.76 49093.23 59101.95
Jun 1994       58011.54 54611.81 61411.28 52812.09 63210.99
Jul 1994       56309.27 52786.52 59832.02 50921.68 61696.85
Aug 1994       56342.08 52700.40 59983.76 50772.61 61911.55
Sep 1994       55515.56 51758.64 59272.48 49769.85 61261.27
Oct 1994       51799.98 47931.20 55668.77 45883.19 57716.78
Nov 1994       49934.90 45957.33 53912.47 43851.73 56018.07
Dec 1994       48315.63 44232.11 52399.15 42070.43 54560.84
Jan 1995       45829.18 41026.13 50632.22 38483.56 53174.79
Feb 1995       46470.89 41579.64 51362.13 38990.37 53951.40
Mar 1995       41937.52 36959.58 46915.45 34324.42 49550.61
Apr 1995       50987.55 45924.36 56050.74 43244.07 58731.04
May 1995       55562.62 50415.54 60709.71 47690.83 63434.41
Jun 1995       59476.58 54246.90 64706.26 51478.47 67474.69
Jul 1995       57774.30 52463.27 63085.34 49651.77 65896.84
Aug 1995       57807.12 52415.90 63198.33 49561.97 66052.27
> gasprod.dhw$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug      Sep      Oct
1994 44364.14 45005.85 40472.48 49522.52 54097.59 58011.54 56309.27 56342.
08 55515.56 51799.98
1995 45829.18 46470.89 41937.52 50987.55 55562.62 59476.58 57774.30 57807.
12
          Nov      Dec
1994 49934.90 48315.63
1995
> ts.plot(test.ds.gasprod,gasprod.dhw$mean,gpars = list(col = c("Red","Blu
e"),
+                                                     main = "Holt Winter's(
Original vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+        box.lwd = 0.1,cex = 0.75)
> MAPE.dhw = mape(test.ds.gasprod,gasprod.dhw$mean)
> print(MAPE.dhw)
[1] 0.0458347
> #### Checking for stationarity using visualization ####
> ts.plot(gasprod,gpars = list(xlab = "Year",ylab = "Gas Production",
+                              main = "Australian Gas Production (1960-199
5)",
+                              col = c("Blue")),lwd = 2)
> #### Plotting ACF and PACF plots ####
> acf(gasprod, lag.max = 50)
> pacf(gasprod, lag.max = 50)
> #### Checking for  stationarity using Augmented Dicky- Fuller Test ####
> adf = adf.test(gasprod,alternative = "stationary")
> adf
```

```
        Augmented Dickey-Fuller Test

data:  gasprod
Dickey-Fuller = -2.7131, Lag order = 7, p-value = 0.2764
alternative hypothesis: stationary

> print(adf$alternative)
[1] "stationary"
> if(adf$p.value < 0.05){
+   print("The series is Stationary & Null Hypothesis is rejected")
+ } else{
+   print("The Series is not Stationary & Null Hypothesis is accepted")
+ }
[1] "The Series is not Stationary & Null Hypothesis is accepted"
> #### Stationarizing the series for performing Manual Arima ####
> diff.gasprod = diff(gasprod, differences = 1)
> ts.plot(diff.gasprod,col = c("Blue"),lwd = 1,main = "Differenced Series"
)
> abline(a=1,b=0,col = c("Red"),lwd =4)
> acf(diff.gasprod, lag.max = 50)
> pacf(diff.gasprod, lag.max = 50)
> adf.d = adf.test(diff.gasprod,alternative = "stationary")
Warning message:
In adf.test(diff.gasprod, alternative = "stationary") :
  p-value smaller than printed p-value
> adf.d

        Augmented Dickey-Fuller Test

data:  diff.gasprod
Dickey-Fuller = -19.321, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary

> print(adf.d$alternative)
[1] "stationary"
> if(adf.d$p.value < 0.05){
+   print("The series is Stationary & Null Hypothesis is rejected")
+ } else{
+   print("The Series is not Stationary & Null Hypothesis is accepted")
+ }
[1] "The series is Stationary & Null Hypothesis is rejected"
> ### Splitting of time series into Training and Testing Data ####
> diff.train.gasprod = window(diff.gasprod,start = c(1970,1),end = c(1993,
12),frequency = 12)
> diff.test.gasprod = window(diff.gasprod,start = c(1994,1),end = c(1995,8
),frequency = 12)
> plot.ts(diff.train.gasprod,main = "Differenced Train")
> plot.ts(diff.test.gasprod,main = "Differenced Test")
> ### Finding the P and Q values ####
> acf(diff.train.gasprod)
> pacf(diff.train.gasprod)
> ### Building the Manual ARIMA model ####
> marima.gasprod = Arima(diff.train.gasprod,order = c(2,1,2))
> marima.gasprod
Series: diff.train.gasprod
ARIMA(2,1,2)

Coefficients:
          ar1     ar2      ma1      ma2
      -0.0131  0.2545  -0.7212  -0.2788
s.e.   0.1623  0.0695   0.1595   0.1593

sigma^2 estimated as 9895599:  log likelihood=-2719.06
AIC=5448.12   AICc=5448.33   BIC=5466.41
> forc.marima.gasprod = forecast(marima.gasprod,h = 20)
> ts.plot(diff.test.gasprod,forc.marima.gasprod$mean,gpars = list(col = c(
"Red","Blue"),
+                                                    main = "Manual Arima(O
riginal vs. Forecasted)"))
```

```
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+         box.lwd = 0.1,cex = 0.50)
> vec.marima = cbind(diff.test.gasprod,forc.marima.gasprod$mean)
> MAPE.marima = mean(abs(vec.marima[,1]-vec.marima[,2])/vec.marima[,1])
> print(MAPE.marima)
[1] 0.1915037
> ###USING AUTO ARIMA (Original) ####
> aarima.gasprod = auto.arima(train.gasprod,seasonal = TRUE)
> aarima.gasprod
Series: train.gasprod
ARIMA(2,1,1)(0,1,2)[12]

Coefficients:
         ar1     ar2      ma1     sma1     sma2
      0.5017  0.2057  -0.9583  -0.4404  -0.1236
s.e.  0.0738  0.0722   0.0426   0.0676   0.0639

sigma^2 estimated as 3535010:  log likelihood=-2463.67
AIC=4939.33   AICc=4939.64   BIC=4961.03
> forc.aarima = forecast(aarima.gasprod,h = 20)
> ts.plot(test.gasprod,forc.aarima$mean,gpars = list(col = c("Red","Blue")
,
+                                                    main = "Auto Arima(Ori
ginal vs. Forecasted)"))
> legend("topleft", legend = c("Actual","Forecasted"),col = c("Red","Blue"
),lty = 1,
+         box.lwd = 0.1,cex = 0.75)
> vec.aarima = cbind(test.gasprod,forc.aarima$mean)
> MAPE.aarima = mean(abs(vec.aarima[,1]-vec.aarima[,2])/vec.aarima[,1])
> print(MAPE.aarima)
[1] 0.06370687
> ### Ljung box test ####
> Box.test(aarima.gasprod$residuals,type = "Ljung-Box")

        Box-Ljung test

data:  aarima.gasprod$residuals
X-squared = 0.009919, df = 1, p-value = 0.9207

> hist(aarima.gasprod$residuals,main = "Histogram of Auto ARIMA residuals"
,
+       xlab = "Residuals")
> ### Making the future forecastusing the best model ####
> future = forecast(aarima.gasprod,h = 32)
> plot(future)
> future$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug      Sep      Oct
1994 40911.17 41136.45 38318.03 44418.69 52575.33 57827.42 59054.39 57800.
39 52923.33 49148.06
1995 41314.97 41577.63 40068.39 45141.64 53284.34 58496.89 60063.56 59142.
29 54005.17 49942.73
1996 41882.79 42199.58 40744.41 45855.92 54028.93 59264.56 60849.03 59941.
44
          Nov      Dec
1994 46382.99 43491.75
1995 46875.98 43928.71
1996
> ### Making the future forecast using the model created from gasprod ####
> aarima.fgasprod = auto.arima(gasprod,seasonal = TRUE)
> fgasprod.forc = forecast(aarima.fgasprod,h = 12)
> plot(fgasprod.forc)
> fgasprod.forc$mean
          Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug      Sep      Oct
1995
56907.83 52476.28
```

1996 43318.41 43601.71 46668.11 49376.36 57536.25 62184.69 65795.74 63391.
54

	Nov	Dec
1995	49719.79	43473.70
1996		