Thota, Sunil Raj – Time Series in R.R

# Intermediate Analytics  
# ALY 6015  
# Module 5 - Time Series in R  
# 02/015/2021  
# Sunil Raj Thota  
# NUID: 001099670

# Get and set the working directories  
getwd()

## [1] "G:/NEU/Coursework/2021 Q1 Winter/ALY 6015 IA/Discussions & Assignments"

setwd('G:/NEU/Coursework/2021 Q1 Winter/ALY 6015 IA/Discussions & Assignments')  
getwd()

## [1] "G:/NEU/Coursework/2021 Q1 Winter/ALY 6015 IA/Discussions & Assignments"

# Installed the above packages into the work space  
install.packages("plyr")  
install.packages("dplyr")  
install.packages("tidyr")  
install.packages("TTR")  
install.packages("forecast")  
  
# Loaded the below libraries into the work space  
library(plyr)

library(dplyr)

library(tidyr)

library(TTR)

library(forecast)

# PART A  
# Problem 1  
  
tsData <- scan("http://robjhyndman.com/tsdldata/data/fancy.dat")  
tsData

## [1] 1664.81 2397.53 2840.71 3547.29 3752.96 3714.74 4349.61  
## [8] 3566.34 5021.82 6423.48 7600.60 19756.21 2499.81 5198.24  
## [15] 7225.14 4806.03 5900.88 4951.34 6179.12 4752.15 5496.43  
## [22] 5835.10 12600.08 28541.72 4717.02 5702.63 9957.58 5304.78  
## [29] 6492.43 6630.80 7349.62 8176.62 8573.17 9690.50 15151.84  
## [36] 34061.01 5921.10 5814.58 12421.25 6369.77 7609.12 7224.75  
## [43] 8121.22 7979.25 8093.06 8476.70 17914.66 30114.41 4826.64  
## [50] 6470.23 9638.77 8821.17 8722.37 10209.48 11276.55 12552.22  
## [57] 11637.39 13606.89 21822.11 45060.69 7615.03 9849.69 14558.40  
## [64] 11587.33 9332.56 13082.09 16732.78 19888.61 23933.38 25391.35  
## [71] 36024.80 80721.71 10243.24 11266.88 21826.84 17357.33 15997.79  
## [78] 18601.53 26155.15 28586.52 30505.41 30821.33 46634.38 104660.67

View(tsData)  
str(tsData)

## num [1:84] 1665 2398 2841 3547 3753 ...

head(tsData)

## [1] 1664.81 2397.53 2840.71 3547.29 3752.96 3714.74

tail(tsData)

## [1] 26155.15 28586.52 30505.41 30821.33 46634.38 104660.67

summary(tsData)

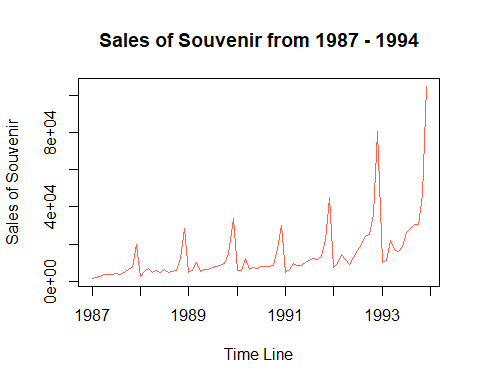
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1665 5884 8772 14316 16889 104661

# Let's perform some Exploratory Data Analysis and Time series analysis using "sales" data set. To get this data set we need to search on Google as Monthly Sales for a Souvenir Shop in Australia from Jan 1987 - Dec 1993. After that, i have loaded the data set using scan() function. I also installed all the necessary package from the packages tab which is right side to the work space in R Studio.  
  
# Or we can also install the packages by using install.packages("package name") command. Once it is loaded we can use it in the code for further analysis and calculations. Loaded the necessary library into the work space. Loaded the sales Data set into the Environment.  
  
# I have also installed TTR package enables us to use SMA function which is required to smooth time data series by using Simple Moving Average. I have also installed 'TTR', 'SMA', to perform Smooth Time Data series by simply moving the averages. Let's install all the above packages.  
  
# To View the diabetes Data set we use View() command, To observe the structure of the Data set we use str() command, and head () and tail() shows first and last few rows in the Data set. Summary() Provides the Descriptive Stats of the sales columns. We noticed 5 variables from the statistics given in the summary.

# Problem 2  
  
tsTime <- ts(tsData, frequency = 12, start = c(1987, 1))  
tsTime

## Jan Feb Mar Apr May Jun Jul  
## 1987 1664.81 2397.53 2840.71 3547.29 3752.96 3714.74 4349.61  
## 1988 2499.81 5198.24 7225.14 4806.03 5900.88 4951.34 6179.12  
## 1989 4717.02 5702.63 9957.58 5304.78 6492.43 6630.80 7349.62  
## 1990 5921.10 5814.58 12421.25 6369.77 7609.12 7224.75 8121.22  
## 1991 4826.64 6470.23 9638.77 8821.17 8722.37 10209.48 11276.55  
## 1992 7615.03 9849.69 14558.40 11587.33 9332.56 13082.09 16732.78  
## 1993 10243.24 11266.88 21826.84 17357.33 15997.79 18601.53 26155.15  
## Aug Sep Oct Nov Dec  
## 1987 3566.34 5021.82 6423.48 7600.60 19756.21  
## 1988 4752.15 5496.43 5835.10 12600.08 28541.72  
## 1989 8176.62 8573.17 9690.50 15151.84 34061.01  
## 1990 7979.25 8093.06 8476.70 17914.66 30114.41  
## 1991 12552.22 11637.39 13606.89 21822.11 45060.69  
## 1992 19888.61 23933.38 25391.35 36024.80 80721.71  
## 1993 28586.52 30505.41 30821.33 46634.38 104660.67

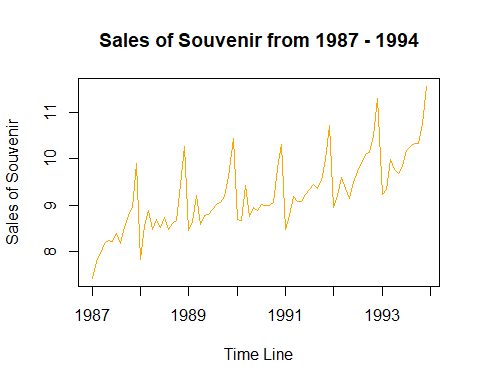
ts.plot(  
 tsTime,  
 xlab = "Time Line",  
 ylab = "Sales of Souvenir",  
 main = "Sales of Souvenir from 1987 - 1994",  
 col = "tomato"  
)



tsToLog <- log(tsTime)  
tsToLog

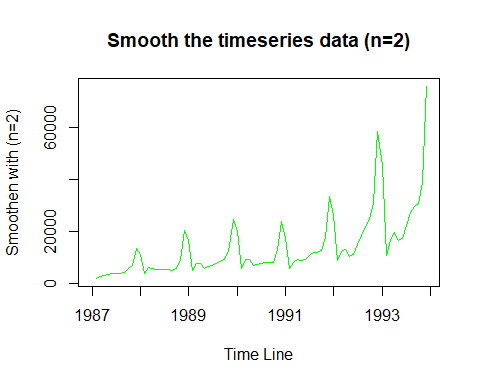
## Jan Feb Mar Apr May Jun Jul  
## 1987 7.417466 7.782194 7.951809 8.173939 8.230300 8.220064 8.377841  
## 1988 7.823970 8.556075 8.885322 8.477627 8.682857 8.507414 8.728931  
## 1989 8.458933 8.648683 9.206089 8.576364 8.778392 8.799481 8.902404  
## 1990 8.686278 8.668124 9.427164 8.759319 8.937103 8.885268 9.002236  
## 1991 8.481906 8.774967 9.173549 9.084910 9.073646 9.231072 9.330481  
## 1992 8.937879 9.195195 9.585923 9.357668 9.141265 9.478999 9.725125  
## 1993 9.234373 9.329623 9.990896 9.761770 9.680206 9.830999 10.171801  
## Aug Sep Oct Nov Dec  
## 1987 8.179295 8.521548 8.767715 8.935982 9.891223  
## 1988 8.466352 8.611854 8.671647 9.441458 10.259122  
## 1989 9.009034 9.056393 9.178901 9.625877 10.435909  
## 1990 8.984600 8.998762 9.045077 9.793375 10.312759  
## 1991 9.437653 9.361978 9.518332 9.990679 10.715766  
## 1992 9.897902 10.083029 10.142164 10.491963 11.298763  
## 1993 10.260691 10.325659 10.335962 10.750093 11.558479

ts.plot(  
 tsToLog,  
 xlab = "Time Line",  
 ylab = "Sales of Souvenir",  
 main = "Sales of Souvenir from 1987 - 1994",  
 col = "orange"  
)

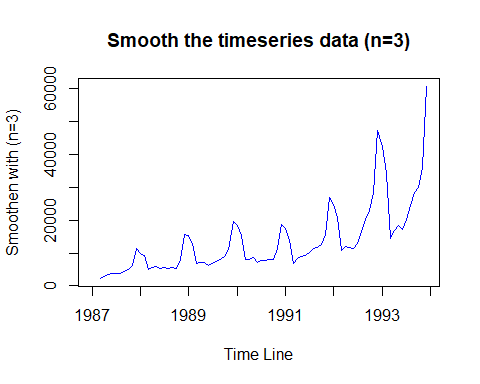


# I have utilized the "ts" function to time series data. I took frequency as 12 to format the data into 12 levels as the months are 12 from 1987 - 1993. I used "ts.plot" function to plot the time series data. In this plot, X Axis depicts the time line and Y Axis plots the number of sales of souvenir. The plot is of Additive type which is not applicable to utilize it for the time series analysis.  
  
# Because the the difference between consecutive time series data is same. To analyze time series data, I need to convert it to multiplicative types by taking "Log" natural for the above dataset. Let's plot the data as shown above. I had used 'plot.ts' but it did not provide any differentiation and produced the same result.

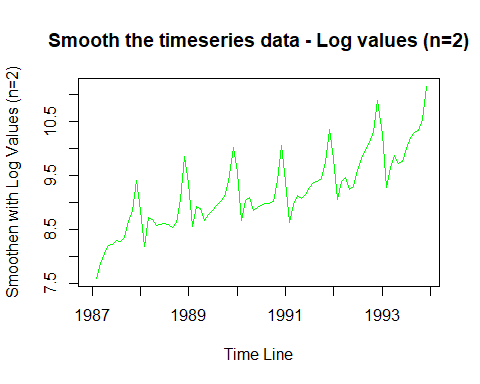
# Problem 3  
  
tsTimeSMAPlot1 <- plot(  
 SMA(tsTime, n = 2),  
 main = "Smooth the timeseries data (n=2)",  
 col = "green",  
 xlab = "Time Line",  
 ylab = "Smoothen with (n=2)"  
)



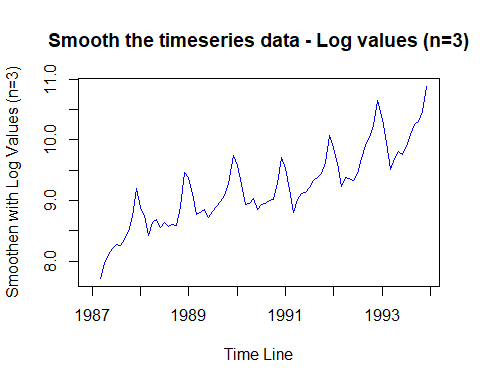
tsTimeSMAPlot2 <- plot(  
 SMA(tsTime, n = 3),  
 main = "Smooth the timeseries data (n=3)",  
 col = "blue",  
 xlab = "Time Line",  
 ylab = "Smoothen with (n=3)"  
)



tsTimeSMAPlot3 <- plot(  
 SMA(tsToLog, n = 2),  
 main = "Smooth the timeseries data - Log values (n=2)",  
 col = "green",  
 xlab = "Time Line",  
 ylab = "Smoothen with Log Values (n=2)"  
)



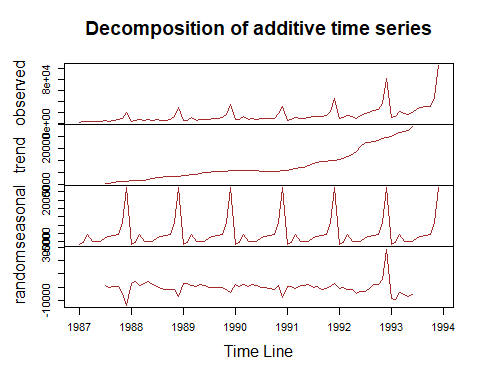
tsTimeSMAPlot4 <- plot(  
 SMA(tsToLog, n = 3),  
 main = "Smooth the timeseries data - Log values (n=3)",  
 col = "blue",  
 xlab = "Time Line",  
 ylab = "Smoothen with Log Values (n=3)"  
)



tsTimeDec <- decompose(tsTime)  
tsTimeDec

## $x  
## Jan Feb Mar Apr May Jun Jul  
## 1987 1664.81 2397.53 2840.71 3547.29 3752.96 3714.74 4349.61  
## 1988 2499.81 5198.24 7225.14 4806.03 5900.88 4951.34 6179.12  
## 1989 4717.02 5702.63 9957.58 5304.78 6492.43 6630.80 7349.62  
## 1990 5921.10 5814.58 12421.25 6369.77 7609.12 7224.75 8121.22  
## 1991 4826.64 6470.23 9638.77 8821.17 8722.37 10209.48 11276.55  
## 1992 7615.03 9849.69 14558.40 11587.33 9332.56 13082.09 16732.78  
## 1993 10243.24 11266.88 21826.84 17357.33 15997.79 18601.53 26155.15  
## Aug Sep Oct Nov Dec  
## 1987 3566.34 5021.82 6423.48 7600.60 19756.21  
## 1988 4752.15 5496.43 5835.10 12600.08 28541.72  
## 1989 8176.62 8573.17 9690.50 15151.84 34061.01  
## 1990 7979.25 8093.06 8476.70 17914.66 30114.41  
## 1991 12552.22 11637.39 13606.89 21822.11 45060.69  
## 1992 19888.61 23933.38 25391.35 36024.80 80721.71  
## 1993 28586.52 30505.41 30821.33 46634.38 104660.67  
##   
## $seasonal  
## Jan Feb Mar Apr May Jun  
## 1987 -6650.1615 -5562.1051 -691.8707 -4601.8646 -5074.2387 -4827.4476  
## 1988 -6650.1615 -5562.1051 -691.8707 -4601.8646 -5074.2387 -4827.4476  
## 1989 -6650.1615 -5562.1051 -691.8707 -4601.8646 -5074.2387 -4827.4476  
## 1990 -6650.1615 -5562.1051 -691.8707 -4601.8646 -5074.2387 -4827.4476  
## 1991 -6650.1615 -5562.1051 -691.8707 -4601.8646 -5074.2387 -4827.4476  
## 1992 -6650.1615 -5562.1051 -691.8707 -4601.8646 -5074.2387 -4827.4476  
## 1993 -6650.1615 -5562.1051 -691.8707 -4601.8646 -5074.2387 -4827.4476  
## Jul Aug Sep Oct Nov Dec  
## 1987 -2452.6359 -2089.4193 -1309.5168 -425.8064 6341.6020 27343.4647  
## 1988 -2452.6359 -2089.4193 -1309.5168 -425.8064 6341.6020 27343.4647  
## 1989 -2452.6359 -2089.4193 -1309.5168 -425.8064 6341.6020 27343.4647  
## 1990 -2452.6359 -2089.4193 -1309.5168 -425.8064 6341.6020 27343.4647  
## 1991 -2452.6359 -2089.4193 -1309.5168 -425.8064 6341.6020 27343.4647  
## 1992 -2452.6359 -2089.4193 -1309.5168 -425.8064 6341.6020 27343.4647  
## 1993 -2452.6359 -2089.4193 -1309.5168 -425.8064 6341.6020 27343.4647  
##   
## $trend  
## Jan Feb Mar Apr May Jun Jul  
## 1987 NA NA NA NA NA NA 5421.133  
## 1988 6517.855 6643.493 6712.677 6707.937 6891.732 7466.107 7924.554  
## 1989 8566.257 8757.715 9028.598 9317.438 9584.403 9920.696 10200.837  
## 1990 10729.094 10753.020 10724.792 10654.212 10718.755 10669.431 10459.387  
## 1991 10913.802 11235.815 11574.035 11935.474 12312.042 13097.614 13836.559  
## 1992 15392.422 15925.448 16743.464 17746.816 18829.614 20907.268 22502.653  
## 1993 25224.785 25979.797 26616.045 27116.128 27784.443 29223.966 NA  
## Aug Sep Oct Nov Dec  
## 1987 5572.621 5872.002 6107.134 6249.078 6390.100  
## 1988 8037.954 8172.822 8307.455 8352.884 8447.509  
## 1989 10255.671 10362.989 10510.016 10600.920 10672.196  
## 1990 10441.103 10352.485 10338.690 10487.217 10657.966  
## 1991 14093.552 14439.348 14759.589 14900.270 15045.387  
## 1992 22671.211 23033.112 23576.381 24094.515 24602.210  
## 1993 NA NA NA NA NA  
##   
## $random  
## Jan Feb Mar Apr May Jun  
## 1987 NA NA NA NA NA NA  
## 1988 2632.1169 4116.8522 1204.3337 2699.9580 4083.3862 2312.6805  
## 1989 2800.9240 2507.0205 1620.8524 589.2071 1982.2662 1537.5514  
## 1990 1842.1673 623.6647 2388.3287 317.4221 1964.6037 1382.7668  
## 1991 562.9994 796.5205 -1243.3947 1487.5609 1484.5666 1939.3134  
## 1992 -1127.2306 -513.6528 -1493.1930 -1557.6212 -4422.8150 -2997.7307  
## 1993 -8331.3839 -9150.8120 -4097.3338 -5156.9337 -6712.4146 -5794.9882  
## Jul Aug Sep Oct Nov Dec  
## 1987 1381.1125 83.1381 459.3348 742.1522 -4990.0804 -13977.3547  
## 1988 707.2021 -1196.3844 -1366.8748 -2046.5482 -2094.4058 -7249.2538  
## 1989 -398.5808 10.3681 -480.3019 -393.7099 -1790.6816 -3954.6509  
## 1990 114.4692 -372.4336 -949.9082 -1436.1836 1085.8409 -7887.0209  
## 1991 -107.3729 548.0868 -1492.4411 -726.8928 580.2375 2671.8382  
## 1992 -3317.2370 -693.1819 2209.7843 2240.7755 5588.6825 28776.0353  
## 1993 NA NA NA NA NA NA  
##   
## $figure  
## [1] -6650.1615 -5562.1051 -691.8707 -4601.8646 -5074.2387 -4827.4476  
## [7] -2452.6359 -2089.4193 -1309.5168 -425.8064 6341.6020 27343.4647  
##   
## $type  
## [1] "additive"  
##   
## attr(,"class")  
## [1] "decomposed.ts"

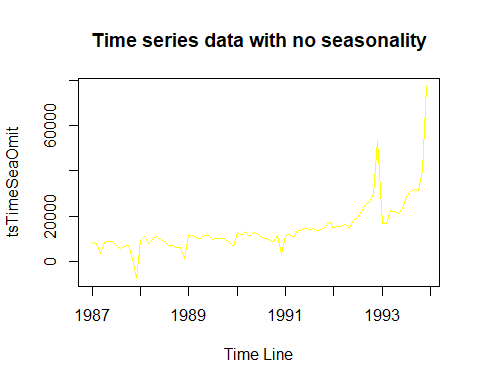
plot(tsTimeDec,  
 col = "brown",  
 xlab = "Time Line")



tsTimeSeaOmit <- tsTime - tsTimeDec$seasonal  
tsTimeSeaOmit

## Jan Feb Mar Apr May Jun Jul  
## 1987 8314.971 7959.635 3532.581 8149.155 8827.199 8542.188 6802.246  
## 1988 9149.971 10760.345 7917.011 9407.895 10975.119 9778.788 8631.756  
## 1989 11367.181 11264.735 10649.451 9906.645 11566.669 11458.248 9802.256  
## 1990 12571.261 11376.685 13113.121 10971.635 12683.359 12052.198 10573.856  
## 1991 11476.801 12032.335 10330.641 13423.035 13796.609 15036.928 13729.186  
## 1992 14265.191 15411.795 15250.271 16189.195 14406.799 17909.538 19185.416  
## 1993 16893.401 16828.985 22518.711 21959.195 21072.029 23428.978 28607.786  
## Aug Sep Oct Nov Dec  
## 1987 5655.759 6331.337 6849.286 1258.998 -7587.255  
## 1988 6841.569 6805.947 6260.906 6258.478 1198.255  
## 1989 10266.039 9882.687 10116.306 8810.238 6717.545  
## 1990 10068.669 9402.577 8902.506 11573.058 2770.945  
## 1991 14641.639 12946.907 14032.696 15480.508 17717.225  
## 1992 21978.029 25242.897 25817.156 29683.198 53378.245  
## 1993 30675.939 31814.927 31247.136 40292.778 77317.205

plot(tsTimeSeaOmit,  
 main = "Time series data with no seasonality",  
 col = "yellow",  
 xlab = "Time Line")



# Let's smoothen the above four plots by using simple moving averages "SMA()" function. This function usually requires 2, 3 consecutive numbers and avg. them and take the next consecutive by averaging the data set. Let's use various values of "n" to alter the smoothing level. The peaks in the time series analysis is determined by SMA. Let's now analyze the components of a time series by using the "decompose()" function to segregate various components.  
  
# After that, I have plotted the graph to analyze these components. In this, we already saw that there are four various components in this time series analysis as observed, trend, random, and seasonal. The seasonal attribute is recurring over the time line and is capable. To get a sure shot on this data we need to eradicate the seasonal aspect which does not give exact analysis of the trends  
  
# Once that component is eradicated from the analysis we can see more precise information on the time series data behavior to observe the rise and fall of the data. From the plot we can depict that the unseasoned hike in the end

# PART B  
# Problem 1  
  
volcanoData <-  
 scan("http://robjhyndman.com/tsdldata/annual/dvi.dat", skip = 1)  
volcanoData

View(volcanoData)  
str(volcanoData)

## num [1:470] 200 150 100 50 0 0 0 0 0 0 ...

head(volcanoData)

## [1] 200 150 100 50 0 0

tail(volcanoData)

## [1] 120 80 40 0 0 0

summary(volcanoData)

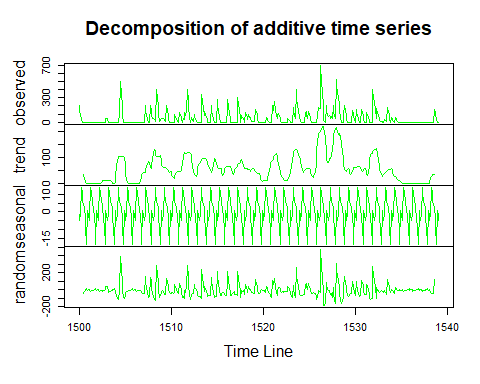
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 0.00 2.50 57.24 80.00 695.00

# Let's perform some Exploratory Data Analysis and Time series analysis using "volcano" data set. After that, i have loaded the data set using scan() function. I also installed all the necessary package from the packages tab which is right side to the work space in R Studio.  
  
# Or we can also install the packages by using install.packages("package name") command. Once it is loaded we can use it in the code for further analysis and calculations. Loaded the necessary library into the work space. Loaded the sales Data set into the Environment. I have also installed forecast package enables us to forecast. Let's install all the above packages.  
  
# To View the diabetes Data set we use View() command, To observe the structure of the Data set we use str() command, and head () and tail() shows first and last few rows in the Data set. Summary() Provides the Descriptive Stats of the volcano columns. We noticed 5 variables from the statistics given in the summary. Let's use the ARIMA to discover the correlations and its problems

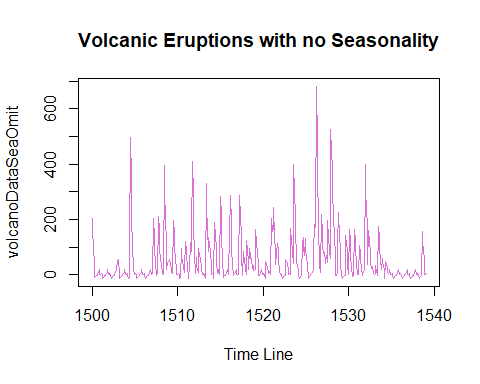
# Problem 2  
  
tsVolcanoData <- ts(volcanoData, start = c(1500), frequency = 12)  
tsVolcanoData

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
## 1500 200 150 100 50 0 0 0 0 0 0 0 0  
## 1501 0 0 0 0 0 0 0 0 0 0 0 0  
## 1502 0 0 0 0 0 0 0 0 0 0 0 50  
## 1503 50 50 0 0 0 0 0 0 0 0 0 0  
## 1504 0 0 0 0 0 100 500 350 200 100 0 0  
## 1505 0 0 0 0 0 0 0 0 0 0 0 0  
## 1506 0 0 0 0 0 0 0 0 0 0 0 0  
## 1507 0 0 200 150 100 50 0 0 0 200 150 100  
## 1508 50 40 30 20 10 400 300 210 110 10 20 50  
## 1509 50 50 40 30 20 10 200 150 100 50 0 0  
## 1510 0 0 0 0 0 100 75 50 25 0 0 120  
## 1511 90 60 30 0 40 30 120 85 150 400 275 175  
## 1512 75 0 60 45 30 15 100 75 50 25 0 0  
## 1513 0 0 0 0 340 255 170 85 130 100 65 30  
## 1514 0 0 0 0 200 150 100 50 0 0 0 0  
## 1515 280 210 140 70 0 0 0 0 0 0 0 0  
## 1516 0 140 285 205 105 45 0 0 0 0 0 0  
## 1517 0 0 0 300 225 150 75 0 80 60 40 20  
## 1518 0 120 90 60 30 100 75 50 55 15 15 15  
## 1519 15 15 160 130 90 50 0 0 0 0 0 0  
## 1520 0 0 0 0 60 45 30 15 0 0 0 0  
## 1521 200 150 160 255 150 95 40 80 110 77 45 13  
## 1522 0 0 0 0 0 0 0 0 50 37 25 13  
## 1523 0 0 0 180 135 90 45 400 300 200 160 45  
## 1524 30 15 0 0 0 0 0 120 130 90 50 130  
## 1525 90 60 30 0 0 0 0 0 0 0 0 80  
## 1526 180 170 170 695 490 375 195 30 15 0 200 150  
## 1527 100 70 80 65 50 75 50 200 130 80 40 525  
## 1528 450 375 300 225 150 75 0 0 0 100 205 140  
## 1529 90 30 0 0 0 0 0 0 140 105 70 35  
## 1530 0 160 120 80 40 0 0 0 160 120 80 40  
## 1531 0 0 0 120 90 60 30 0 0 0 0 400  
## 1532 300 240 170 50 170 125 85 45 20 15 10 5  
## 1533 0 0 30 25 15 5 180 135 90 45 0 60  
## 1534 45 30 15 0 60 45 30 15 0 0 0 0  
## 1535 0 0 0 0 0 0 0 0 0 0 0 0  
## 1536 0 0 0 0 0 0 0 0 0 0 0 0  
## 1537 0 0 0 0 0 0 0 0 0 0 0 0  
## 1538 0 0 0 0 0 0 0 160 120 80 40 0  
## 1539 0 0

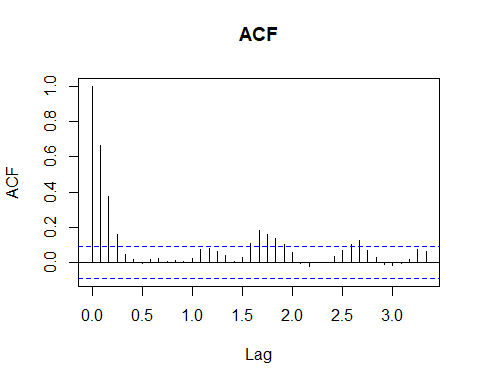
volcanoDataDec <- decompose(tsVolcanoData)  
plot(volcanoDataDec,  
 col = "green",  
 xlab = "Time Line")



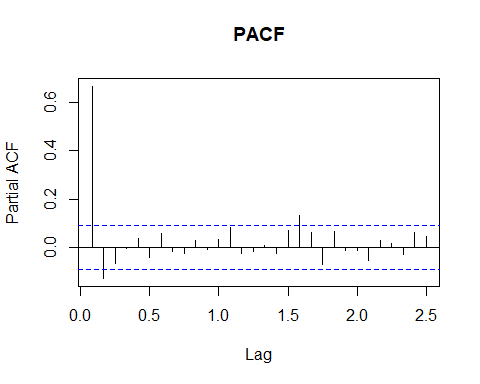
volcanoDataSeaOmit <- tsVolcanoData - volcanoDataDec$seasonal  
plot(volcanoDataSeaOmit,  
 main = "Volcanic Eruptions with no Seasonality",  
 xlab = "Time Line",  
 col = "orchid")



acf(tsVolcanoData, lag.max = 40, main = "ACF")



pacf(tsVolcanoData, lag.max = 30, main = "PACF")



acf(tsVolcanoData, lag.max = 40, plot = FALSE)

##   
## Autocorrelations of series 'tsVolcanoData', by lag  
##   
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333   
## 1.000 0.666 0.374 0.162 0.046 0.017 -0.007 0.016 0.021 0.006 0.010   
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667 1.7500   
## 0.004 0.024 0.075 0.082 0.064 0.039 0.005 0.028 0.108 0.182 0.159   
## 1.8333 1.9167 2.0000 2.0833 2.1667 2.2500 2.3333 2.4167 2.5000 2.5833 2.6667   
## 0.139 0.100 0.056 -0.005 -0.020 0.001 0.001 0.034 0.067 0.103 0.125   
## 2.7500 2.8333 2.9167 3.0000 3.0833 3.1667 3.2500 3.3333   
## 0.071 0.030 -0.010 -0.016 -0.004 0.017 0.075 0.060

pacf(tsVolcanoData, lag.max = 30, plot = FALSE)

##   
## Partial autocorrelations of series 'tsVolcanoData', by lag  
##   
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333 0.9167   
## 0.666 -0.126 -0.064 -0.005 0.040 -0.039 0.058 -0.016 -0.025 0.028 -0.008   
## 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667 1.7500 1.8333   
## 0.036 0.082 -0.025 -0.014 0.008 -0.025 0.073 0.131 0.063 -0.069 0.068   
## 1.9167 2.0000 2.0833 2.1667 2.2500 2.3333 2.4167 2.5000   
## -0.009 -0.010 -0.051 0.029 0.018 -0.026 0.062 0.049

tsVolcanoData1 <- ts(volcanoData, start = c(1500))  
tsVolcanoData1

## Time Series:  
## Start = 1500   
## End = 1969   
## Frequency = 1   
## [1] 200 150 100 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [19] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 50  
## [37] 50 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 100  
## [55] 500 350 200 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [73] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 200 150 100 50  
## [91] 0 0 0 200 150 100 50 40 30 20 10 400 300 210 110 10 20 50  
## [109] 50 50 40 30 20 10 200 150 100 50 0 0 0 0 0 0 0 100  
## [127] 75 50 25 0 0 120 90 60 30 0 40 30 120 85 150 400 275 175  
## [145] 75 0 60 45 30 15 100 75 50 25 0 0 0 0 0 0 340 255  
## [163] 170 85 130 100 65 30 0 0 0 0 200 150 100 50 0 0 0 0  
## [181] 280 210 140 70 0 0 0 0 0 0 0 0 0 140 285 205 105 45  
## [199] 0 0 0 0 0 0 0 0 0 300 225 150 75 0 80 60 40 20  
## [217] 0 120 90 60 30 100 75 50 55 15 15 15 15 15 160 130 90 50  
## [235] 0 0 0 0 0 0 0 0 0 0 60 45 30 15 0 0 0 0  
## [253] 200 150 160 255 150 95 40 80 110 77 45 13 0 0 0 0 0 0  
## [271] 0 0 50 37 25 13 0 0 0 180 135 90 45 400 300 200 160 45  
## [289] 30 15 0 0 0 0 0 120 130 90 50 130 90 60 30 0 0 0  
## [307] 0 0 0 0 0 80 180 170 170 695 490 375 195 30 15 0 200 150  
## [325] 100 70 80 65 50 75 50 200 130 80 40 525 450 375 300 225 150 75  
## [343] 0 0 0 100 205 140 90 30 0 0 0 0 0 0 140 105 70 35  
## [361] 0 160 120 80 40 0 0 0 160 120 80 40 0 0 0 120 90 60  
## [379] 30 0 0 0 0 400 300 240 170 50 170 125 85 45 20 15 10 5  
## [397] 0 0 30 25 15 5 180 135 90 45 0 60 45 30 15 0 60 45  
## [415] 30 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [433] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [451] 0 0 0 0 0 0 0 0 0 0 0 0 0 160 120 80 40 0  
## [469] 0 0

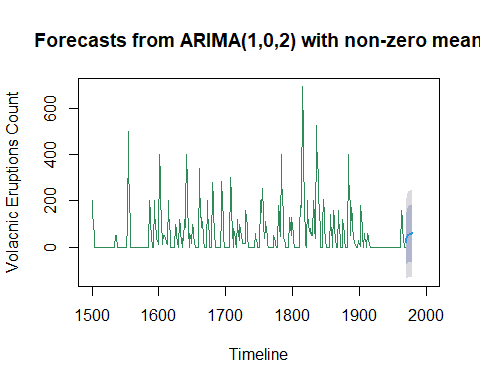
volcanoDataARIMA <- auto.arima(tsVolcanoData1)  
volcanoDataARIMA

## Series: tsVolcanoData1   
## ARIMA(1,0,2) with non-zero mean   
##   
## Coefficients:  
## ar1 ma1 ma2 mean  
## 0.4723 0.2694 0.1279 57.5178  
## s.e. 0.0936 0.0969 0.0752 8.4883  
##   
## sigma^2 estimated as 4897: log likelihood=-2661.84  
## AIC=5333.68 AICc=5333.81 BIC=5354.45

volcanoDataForecast <- forecast(volcanoDataARIMA)  
volcanoDataForecast

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 1970 22.67720 -67.00065 112.3550 -114.4732 159.8276  
## 1971 38.42748 -73.22746 150.0824 -132.3340 209.1890  
## 1972 48.50091 -71.10909 168.1109 -134.4268 231.4286  
## 1973 53.25886 -68.05471 174.5724 -132.2742 238.7920  
## 1974 55.50617 -66.18421 177.1965 -130.6032 241.6155  
## 1975 56.56763 -65.20665 178.3419 -129.6701 242.8053  
## 1976 57.06898 -64.72400 178.8620 -129.1973 243.3353  
## 1977 57.30579 -64.49137 179.1029 -128.9669 243.5785  
## 1978 57.41764 -64.38045 179.2157 -128.8565 243.6917  
## 1979 57.47046 -64.32783 179.2688 -128.8040 243.7449

plot(  
 volcanoDataForecast,  
 xlim = c(1500, 2000),  
 col = "seagreen",  
 xlab = "Timeline",  
 ylab = "Volacnic Eruptions Count"  
)



# Let's check the time series data for Volcano and took frequency as 12 because it has 12 levels for 12 months. To check for seasonality let's use decompose() and eradicate this component from the data and observe the volcanic eruptions trends  
  
# The plot depicts the total number of volcanic eruptions from 1500 - 1540. And, also let's use acf() and pacf() functions to check correlation. Where acf() is auto correlation and pacf() is partial correlation. acf() performs auto-correlation on the time series data with lagged attributes. Where pacf() is partial auto-correlation function that is used to observe the residuals correlation  
  
# Let's create time series with ARIMA function for the volcanoes data set. auto.arima() function checks the best value of q, p, and automatically. forecast() function is used to predict the volcanic eruptions trends. We can see that a prediction at the timeline end.