ARIMA\_Seasonal\_Forecasting

SOBOYEJO LUKMAN ADEBOYE

01/06/2020

## Packages Installed ————————————————————————-

# pkgs <-  
 (c("forecast", "patchwork", "ggplot2", "plotly", "tidyverse", "pander", "ggthemes", "reshape", "cowplot", "imputeTS", "dplyr"))

## [1] "forecast" "patchwork" "ggplot2" "plotly" "tidyverse" "pander"   
## [7] "ggthemes" "reshape" "cowplot" "imputeTS" "dplyr"

# install.packages(pkgs)  
  
library(forecast) # For the moving avearage timeseries

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

library(patchwork) # For combining plots  
library(ggplot2) # For Visualization  
library(plotly) # For interactive plot

##   
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

## The following object is masked from 'package:stats':  
##   
## filter

## The following object is masked from 'package:graphics':  
##   
## layout

library(lubridate) #

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(knitr)  
library(tidyverse)

## -- Attaching packages ---------------------------------------------------------------- tidyverse 1.3.0 --

## v tibble 3.0.0 v dplyr 0.8.5  
## v tidyr 1.1.0 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.5.0  
## v purrr 0.3.4

## -- Conflicts ------------------------------------------------------------------- tidyverse\_conflicts() --  
## x lubridate::as.difftime() masks base::as.difftime()  
## x lubridate::date() masks base::date()  
## x dplyr::filter() masks plotly::filter(), stats::filter()  
## x lubridate::intersect() masks base::intersect()  
## x dplyr::lag() masks stats::lag()  
## x lubridate::setdiff() masks base::setdiff()  
## x lubridate::union() masks base::union()

library(pander)  
library(ggthemes)  
library(reshape2)

##   
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':  
##   
## smiths

library(cowplot)

##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Note: As of version 1.0.0, cowplot does not change the

## default ggplot2 theme anymore. To recover the previous

## behavior, execute:  
## theme\_set(theme\_cowplot())

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##   
## Attaching package: 'cowplot'

## The following object is masked from 'package:ggthemes':  
##   
## theme\_map

## The following object is masked from 'package:lubridate':  
##   
## stamp

## The following object is masked from 'package:patchwork':  
##   
## align\_plots

library(imputeTS)  
library(dplyr)

## Read file into R —

# Monthly Time series  
rm(list=ls(all=TRUE))  
library(RCurl)

## Warning: package 'RCurl' was built under R version 4.0.3

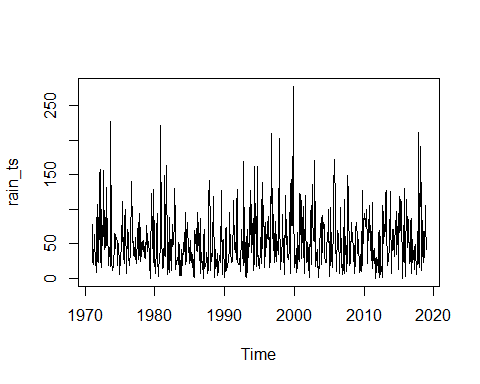
##   
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':  
##   
## complete

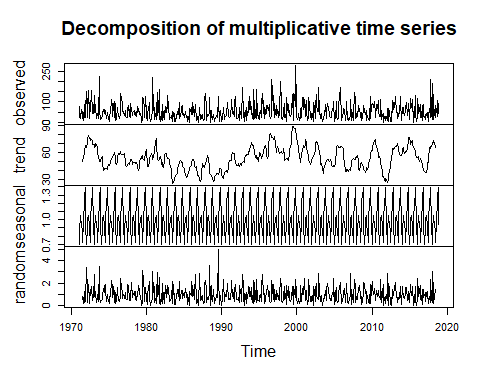
san.mon.data = read.table(text=getURL("https://raw.githubusercontent.com/lukeng200/MSc\_Coastal\_Drainage\_Analysis/master/data\_csv/san.mon.csv"),  
 header = T,sep = ",")  
# Date Conversion to POSXCIT  
san.parse.date <- mutate(san.mon.data, parse.date = parse\_date\_time(year, "by"))  
san.newdate <- mutate(san.parse.date, newdate = as.Date(parse.date))  
san.mon <- mutate(san.newdate, year = as.POSIXlt(newdate)$year + 1900)  
# str(san.mon)  
# head(san.mon)  
  
  
# Rasponi Basin Data  
ras.mon.data = read.table(text=getURL("https://raw.githubusercontent.com/lukeng200/MSc\_Coastal\_Drainage\_Analysis/master/data\_csv/ras.mon.csv"),  
 header = T,sep = ",")  
# Date Conversion to POSXCIT  
ras.parse.date <- mutate(ras.mon.data, parse.date = parse\_date\_time(year, "by"))  
ras.newdate <- mutate(ras.parse.date, newdate = as.Date(parse.date))  
ras.mon <- mutate(ras.newdate, year = as.POSIXlt(newdate)$year + 1900)  
# str(ras.mon)  
# head(ras.mon)  
  
  
  
# Quinto Basin Data  
quin.mon.data = read.table(text=getURL("https://raw.githubusercontent.com/lukeng200/MSc\_Coastal\_Drainage\_Analysis/master/data\_csv/quin.mon.csv"),  
 header = T,sep = ",")  
# Date Conversion to POSXCIT  
quin.parse.date <- mutate(quin.mon.data, parse.date = parse\_date\_time(year, "by"))  
quin.newdate <- mutate(quin.parse.date, newdate = as.Date(parse.date))  
quin.mon <- mutate(quin.newdate, year = as.POSIXlt(newdate)$year + 1900)  
# str(quin.mon)  
# head(quin.mon)  
  
# # kable()  
# # pander()

# Precipitation —

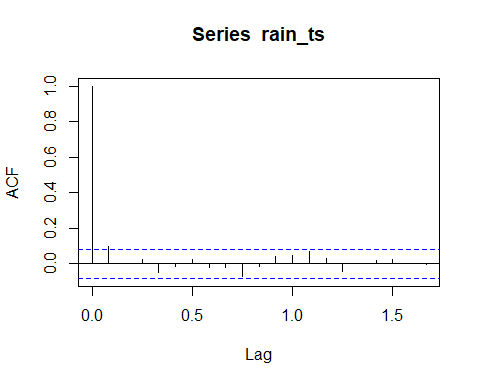
# Time Series  
rain\_ts = ts(san.mon$p, frequency = 12, start = c(1971,1))  
plot.ts(rain\_ts)



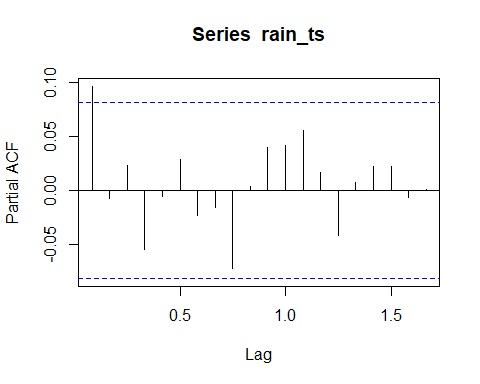
# Decomposing time series into trend,seasonality and randomness assuming additive model  
rain\_ts\_components= decompose(rain\_ts, type = "multiplicative")  
plot(rain\_ts\_components)



# ACF and PACF of data  
# par(mfrow=c(1,3))  
acf(rain\_ts, lag.max=20)



pacf(rain\_ts, lag.max=20)



# Building ARIMA model

# Calculating ndiffs and nsdiffs using forecast  
ndiffs(rain\_ts)

## [1] 0

nsdiffs(rain\_ts)

## [1] 0

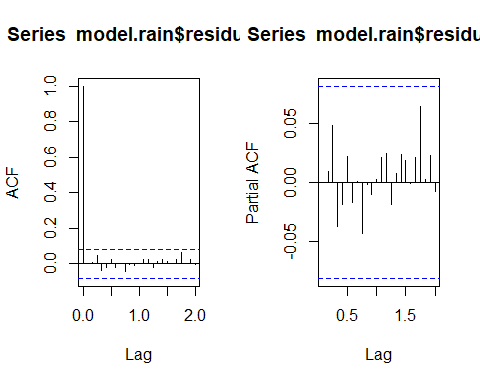
# # ACF and PACF plots show trend and seasonility  
# # Performing differences to make the data stationary  
# # calculating "d "for Trend  
# par(mfrow = c(1, 1))  
# rain\_ts\_diff1 = diff(rain\_ts, differences = 1)  
# plot(rain\_ts\_diff1)  
#   
# # Checking ACF and PACF for after differencing for above values   
# par(mfrow=c(1,2))  
# acf(rain\_ts\_diff1, lag.max=20)  
# pacf(rain\_ts\_diff1, lag.max=20)  
#   
# # Performing seasonal differencing and checking for value of D  
# par(mfrow = c(1, 1))  
# rain\_ts\_seas\_diff1 = diff(rain\_ts, lag = 12, differences=1)  
# plot(rain\_ts\_seas\_diff1)  
#   
# # Checking ACF and PACF for seasonally differenced data  
# par(mfrow = c(1, 2))  
# acf(rain\_ts\_seas\_diff1)  
# pacf(rain\_ts\_seas\_diff1)

# No Differencing seasonally differenced data again

# Implementing ARIMA model  
# auto.arima(rain\_ts)  
model.rain = Arima(rain\_ts, order = c(0,0,1),seasonal = c(2,0,2), include.drift = FALSE)  
  
library(forecast)  
forecast(model.rain)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Jan 2019 43.44806 -8.0043157 94.90044 -35.2415583 122.1377  
## Feb 2019 65.44155 13.9108717 116.97223 -13.3678213 144.2509  
## Mar 2019 57.43603 5.9053541 108.96671 -21.3733389 136.2454  
## Apr 2019 52.29632 0.7656465 103.82700 -26.5130466 131.1057  
## May 2019 56.08888 4.5582052 107.61956 -22.7204879 134.8983  
## Jun 2019 55.56832 4.0376460 107.09900 -23.2410470 134.3777  
## Jul 2019 38.39009 -13.1405867 89.92077 -40.4192798 117.1995  
## Aug 2019 47.29480 -4.2358798 98.82548 -31.5145728 126.1042  
## Sep 2019 56.38745 4.8567688 107.91812 -22.4219243 135.1968  
## Oct 2019 54.31739 2.7867108 105.84807 -24.4919823 133.1268  
## Nov 2019 79.12369 27.5930109 130.65437 0.3143179 157.9331  
## Dec 2019 46.14046 -5.3902177 97.67114 -32.6689107 124.9498  
## Jan 2020 41.71793 -9.8135728 93.24944 -37.0927048 120.5286  
## Feb 2020 79.48079 27.9492772 131.01230 0.6701440 158.2914  
## Mar 2020 60.87253 9.3410175 112.40404 -17.9381157 139.6832  
## Apr 2020 47.45902 -4.0724851 98.99053 -31.3516184 126.2697  
## May 2020 55.00389 3.4723798 106.53540 -23.8067535 133.8145  
## Jun 2020 58.79857 7.2670598 110.33008 -20.0120734 137.6092  
## Jul 2020 38.87961 -12.6518979 90.41112 -39.9310312 117.6903  
## Aug 2020 50.95053 -0.5809755 102.48204 -27.8601087 129.7612  
## Sep 2020 52.78881 1.2573034 104.32032 -26.0218299 131.5995  
## Oct 2020 60.60007 9.0685656 112.13158 -18.2105677 139.4107  
## Nov 2020 75.38669 23.8551783 126.91820 -3.4239550 154.1973  
## Dec 2020 46.73850 -4.7930105 98.27001 -32.0721437 125.5491

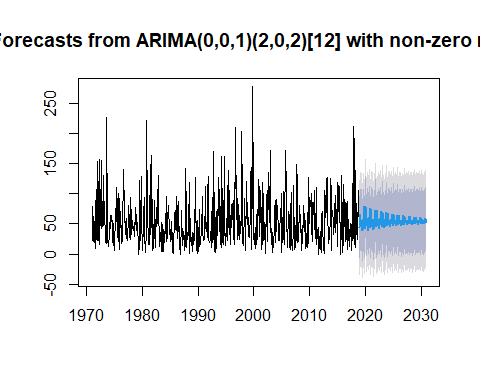
# Checking residuals to ensure they are white noise  
par(mfrow = c(1, 2))  
acf(model.rain$residuals, lag.max = 24)  
pacf(model.rain$residuals, lag.max = 24)



Box.test(model.rain$residuals, lag=24, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: model.rain$residuals  
## X-squared = 8.4772, df = 24, p-value = 0.9985

# Forecasting using Arima model  
par(mfrow = c(1, 1))  
rain\_ts\_forecast = forecast(model.rain, h= 144)  
plot(rain\_ts\_forecast)



rain\_ts\_forecast$mean

## Jan Feb Mar Apr May Jun Jul Aug  
## 2019 43.44806 65.44155 57.43603 52.29632 56.08888 55.56832 38.39009 47.29480  
## 2020 41.71793 79.48079 60.87253 47.45902 55.00389 58.79857 38.87961 50.95053  
## 2021 43.76921 73.83907 59.38402 49.04513 54.97608 57.64797 41.07180 50.92413  
## 2022 45.02694 71.56619 58.75005 49.57005 54.81700 57.23145 42.71660 51.37344  
## 2023 46.19743 69.20307 58.10182 50.15210 54.69864 56.78344 44.18449 51.69432  
## 2024 47.20330 67.20642 57.55277 50.63941 54.59283 56.40675 45.45459 51.98346  
## 2025 48.07919 65.46270 57.07348 51.06566 54.50131 56.07750 46.55927 52.23324  
## 2026 48.84019 63.94847 56.65723 51.43572 54.42170 55.79162 47.51923 52.45055  
## 2027 49.50160 62.63226 56.29543 51.75740 54.35252 55.54312 48.35355 52.63938  
## 2028 50.07644 61.48836 55.98099 52.03696 54.29240 55.32715 49.07866 52.80350  
## 2029 50.57604 60.49419 55.70771 52.27993 54.24014 55.13945 49.70886 52.94613  
## 2030 51.01024 59.63015 55.47020 52.49110 54.19473 54.97632 50.25657 53.07010  
## Sep Oct Nov Dec  
## 2019 56.38745 54.31739 79.12369 46.14046  
## 2020 52.78881 60.60007 75.38669 46.73850  
## 2021 53.41573 58.78783 72.63734 47.73645  
## 2022 53.40712 58.28488 70.17438 48.53328  
## 2023 53.48123 57.68973 68.04464 49.23620  
## 2024 53.53361 57.19577 66.19208 49.84558  
## 2025 53.58090 56.76304 64.58225 50.37542  
## 2026 53.62175 56.38746 63.18310 50.83588  
## 2027 53.65728 56.06096 61.96710 51.23607  
## 2028 53.68816 55.77721 60.91026 51.58387  
## 2029 53.71500 55.53060 59.99176 51.88615  
## 2030 53.73832 55.31627 59.19349 52.14886

library(pander)  
pander(rain\_ts\_forecast$mean)

Table continues below

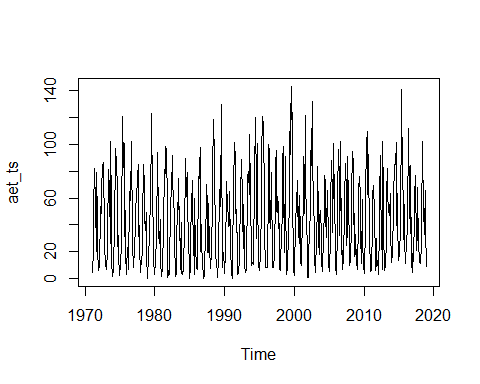
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug |
| **2019** | 43.45 | 65.44 | 57.44 | 52.3 | 56.09 | 55.57 | 38.39 | 47.29 |
| **2020** | 41.72 | 79.48 | 60.87 | 47.46 | 55 | 58.8 | 38.88 | 50.95 |
| **2021** | 43.77 | 73.84 | 59.38 | 49.05 | 54.98 | 57.65 | 41.07 | 50.92 |
| **2022** | 45.03 | 71.57 | 58.75 | 49.57 | 54.82 | 57.23 | 42.72 | 51.37 |
| **2023** | 46.2 | 69.2 | 58.1 | 50.15 | 54.7 | 56.78 | 44.18 | 51.69 |
| **2024** | 47.2 | 67.21 | 57.55 | 50.64 | 54.59 | 56.41 | 45.45 | 51.98 |
| **2025** | 48.08 | 65.46 | 57.07 | 51.07 | 54.5 | 56.08 | 46.56 | 52.23 |
| **2026** | 48.84 | 63.95 | 56.66 | 51.44 | 54.42 | 55.79 | 47.52 | 52.45 |
| **2027** | 49.5 | 62.63 | 56.3 | 51.76 | 54.35 | 55.54 | 48.35 | 52.64 |
| **2028** | 50.08 | 61.49 | 55.98 | 52.04 | 54.29 | 55.33 | 49.08 | 52.8 |
| **2029** | 50.58 | 60.49 | 55.71 | 52.28 | 54.24 | 55.14 | 49.71 | 52.95 |
| **2030** | 51.01 | 59.63 | 55.47 | 52.49 | 54.19 | 54.98 | 50.26 | 53.07 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sep | Oct | Nov | Dec |
| **2019** | 56.39 | 54.32 | 79.12 | 46.14 |
| **2020** | 52.79 | 60.6 | 75.39 | 46.74 |
| **2021** | 53.42 | 58.79 | 72.64 | 47.74 |
| **2022** | 53.41 | 58.28 | 70.17 | 48.53 |
| **2023** | 53.48 | 57.69 | 68.04 | 49.24 |
| **2024** | 53.53 | 57.2 | 66.19 | 49.85 |
| **2025** | 53.58 | 56.76 | 64.58 | 50.38 |
| **2026** | 53.62 | 56.39 | 63.18 | 50.84 |
| **2027** | 53.66 | 56.06 | 61.97 | 51.24 |
| **2028** | 53.69 | 55.78 | 60.91 | 51.58 |
| **2029** | 53.71 | 55.53 | 59.99 | 51.89 |
| **2030** | 53.74 | 55.32 | 59.19 | 52.15 |

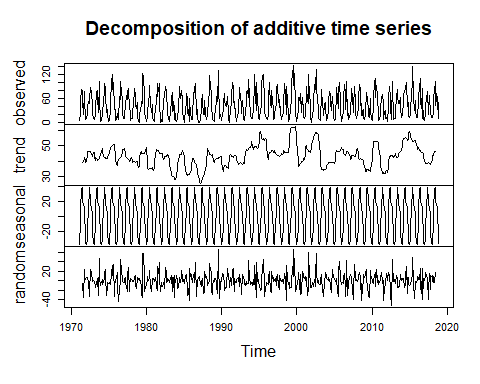
# write.csv(rain\_forecast, "D:/HYDRO COMPLEXITY/R software/Github\_Learning/MSc\_Coastal\_Drainage\_Analysis-master/MSc\_Coastal\_Drainage\_Analysis/mon\_timeseries\\seasonal\_rain\_forecast.csv")

# Actual ET —

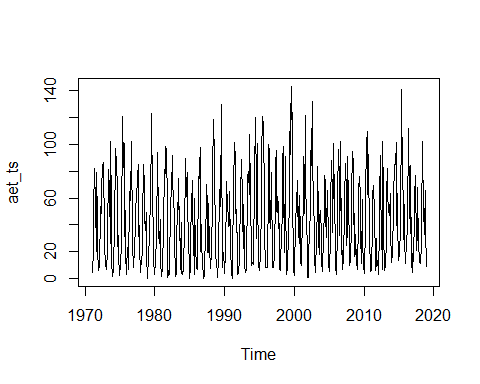
# Time Series  
aet\_ts = ts(san.mon$aet, frequency = 12, start = c(1971,1))  
plot.ts(aet\_ts)



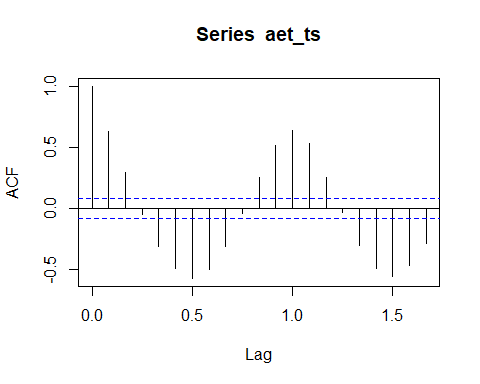
# Decomposing time series into trend,seasonality and randomness assuming additive model  
aet\_ts\_components= decompose(aet\_ts, type = "additive")  
plot(aet\_ts\_components)



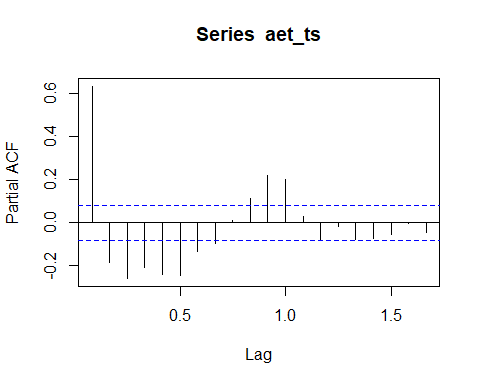
# ACF and PACF of data  
# par(mfrow=c(1,3))  
plot.ts(aet\_ts)



acf(aet\_ts, lag.max=20)



pacf(aet\_ts, lag.max=20)



# Building ARIMA model

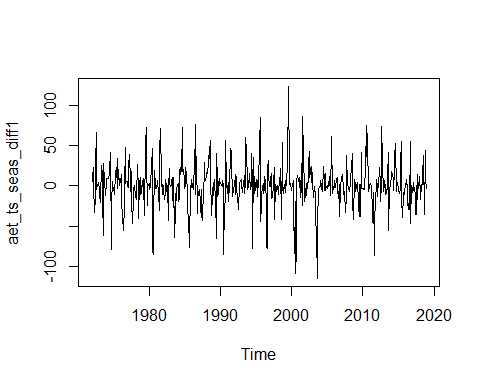
# Calculating ndiffs and nsdiffs  
ndiffs(aet\_ts)

## [1] 0

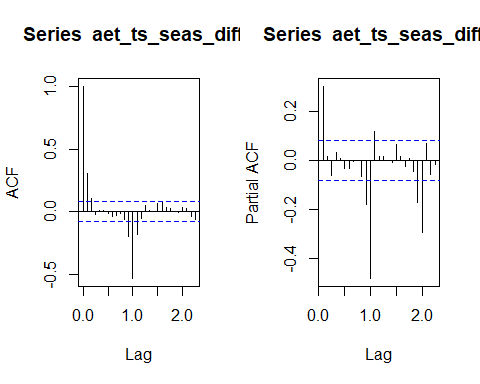
nsdiffs(aet\_ts)

## [1] 1

# # ACF and PACF plots show trend and seasonility  
# # Performing differences to make the data stationary  
# # calculating "d "for Trend  
# par(mfrow = c(1, 1))  
# aet\_ts\_diff1 = diff(aet\_ts, differences = 1)  
# plot(aet\_ts\_diff1)  
#   
# # Checking ACF and PACF for after differencing for above values   
# par(mfrow=c(1,2))  
# acf(aet\_ts\_diff1, lag.max=20)  
# pacf(aet\_ts\_diff1, lag.max=20)  
  
# Performing seasonal differencing and checking for value of D  
par(mfrow = c(1, 1))  
aet\_ts\_seas\_diff1 = diff(aet\_ts, lag = 12, differences = 1)  
plot(aet\_ts\_seas\_diff1)

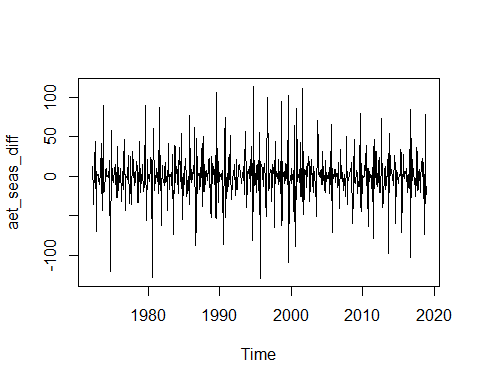


# Checking ACF and PACF for seasonally differenced data  
par(mfrow = c(1, 2))  
acf(aet\_ts\_seas\_diff1)  
pacf(aet\_ts\_seas\_diff1)

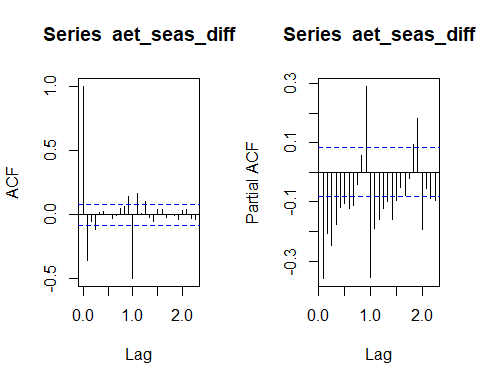


# Differencing seasonally differenced data again

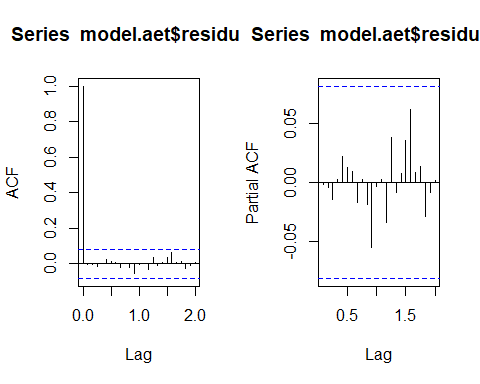
par(mfrow = c(1, 1))  
aet\_seas\_diff = diff(aet\_ts\_seas\_diff1, differences=1)  
plot(aet\_seas\_diff)



par(mfrow = c(1, 2))  
acf(aet\_seas\_diff)  
pacf(aet\_seas\_diff)



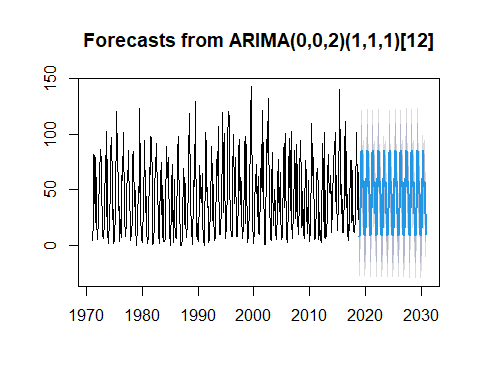
# Implementing ARIMA model  
# auto.arima(aet\_ts)  
model.aet = Arima(aet\_ts, order = c(0,0,2),seasonal = c(1,1,1), include.drift = FALSE)  
  
# Checking residuals to ensure they are white noise  
par(mfrow = c(1, 2))  
acf(model.aet$residuals, lag.max = 24)  
pacf(model.aet$residuals, lag.max = 24)



Box.test(model.aet$residuals, lag=24, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: model.aet$residuals  
## X-squared = 7.8092, df = 24, p-value = 0.9993

# Forecasting using Arima model  
par(mfrow = c(1, 1))  
aet\_ts\_forecast = forecast(model.aet, h= 144)  
plot(aet\_ts\_forecast)



aet\_ts\_forecast$mean

## Jan Feb Mar Apr May Jun Jul  
## 2019 7.737980 15.050059 33.715602 60.173571 84.224812 85.120575 53.771218  
## 2020 8.864028 15.303502 33.310926 60.298465 83.971035 85.936654 53.209568  
## 2021 8.809229 15.291169 33.330619 60.292387 83.983385 85.896940 53.236901  
## 2022 8.811896 15.291769 33.329661 60.292683 83.982784 85.898872 53.235571  
## 2023 8.811766 15.291740 33.329707 60.292669 83.982813 85.898778 53.235636  
## 2024 8.811773 15.291741 33.329705 60.292670 83.982812 85.898783 53.235632  
## 2025 8.811772 15.291741 33.329705 60.292669 83.982812 85.898783 53.235633  
## 2026 8.811772 15.291741 33.329705 60.292669 83.982812 85.898783 53.235633  
## 2027 8.811772 15.291741 33.329705 60.292669 83.982812 85.898783 53.235633  
## 2028 8.811772 15.291741 33.329705 60.292669 83.982812 85.898783 53.235633  
## 2029 8.811772 15.291741 33.329705 60.292669 83.982812 85.898783 53.235633  
## 2030 8.811772 15.291741 33.329705 60.292669 83.982812 85.898783 53.235633  
## Aug Sep Oct Nov Dec  
## 2019 52.561815 58.439232 45.882494 28.068089 9.680945  
## 2020 53.553999 57.182747 46.854206 28.484752 9.631748  
## 2021 53.505715 57.243894 46.806918 28.464475 9.634142  
## 2022 53.508065 57.240918 46.809219 28.465462 9.634026  
## 2023 53.507950 57.241063 46.809107 28.465414 9.634031  
## 2024 53.507956 57.241056 46.809113 28.465416 9.634031  
## 2025 53.507955 57.241056 46.809113 28.465416 9.634031  
## 2026 53.507955 57.241056 46.809113 28.465416 9.634031  
## 2027 53.507955 57.241056 46.809113 28.465416 9.634031  
## 2028 53.507955 57.241056 46.809113 28.465416 9.634031  
## 2029 53.507955 57.241056 46.809113 28.465416 9.634031  
## 2030 53.507955 57.241056 46.809113 28.465416 9.634031

library(pander)  
pander(aet\_ts\_forecast$mean)

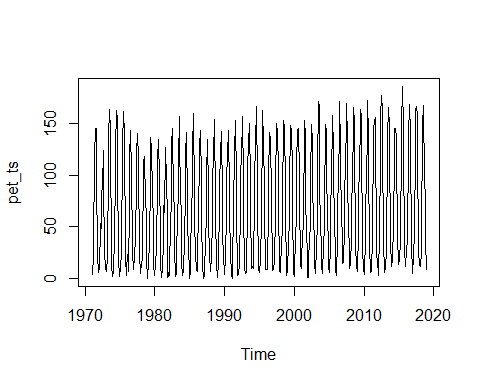
Table continues below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug |
| **2019** | 7.738 | 15.05 | 33.72 | 60.17 | 84.22 | 85.12 | 53.77 | 52.56 |
| **2020** | 8.864 | 15.3 | 33.31 | 60.3 | 83.97 | 85.94 | 53.21 | 53.55 |
| **2021** | 8.809 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |
| **2022** | 8.812 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |
| **2023** | 8.812 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |
| **2024** | 8.812 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |
| **2025** | 8.812 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |
| **2026** | 8.812 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |
| **2027** | 8.812 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |
| **2028** | 8.812 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |
| **2029** | 8.812 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |
| **2030** | 8.812 | 15.29 | 33.33 | 60.29 | 83.98 | 85.9 | 53.24 | 53.51 |

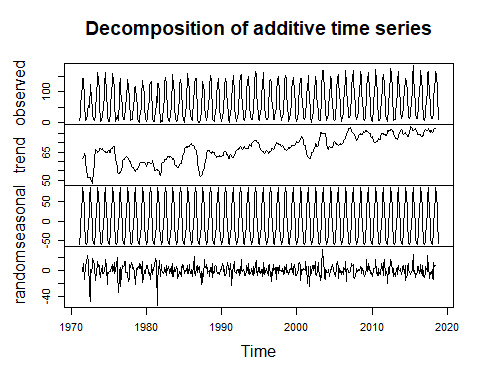
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sep | Oct | Nov | Dec |
| **2019** | 58.44 | 45.88 | 28.07 | 9.681 |
| **2020** | 57.18 | 46.85 | 28.48 | 9.632 |
| **2021** | 57.24 | 46.81 | 28.46 | 9.634 |
| **2022** | 57.24 | 46.81 | 28.47 | 9.634 |
| **2023** | 57.24 | 46.81 | 28.47 | 9.634 |
| **2024** | 57.24 | 46.81 | 28.47 | 9.634 |
| **2025** | 57.24 | 46.81 | 28.47 | 9.634 |
| **2026** | 57.24 | 46.81 | 28.47 | 9.634 |
| **2027** | 57.24 | 46.81 | 28.47 | 9.634 |
| **2028** | 57.24 | 46.81 | 28.47 | 9.634 |
| **2029** | 57.24 | 46.81 | 28.47 | 9.634 |
| **2030** | 57.24 | 46.81 | 28.47 | 9.634 |

# Potential ET —

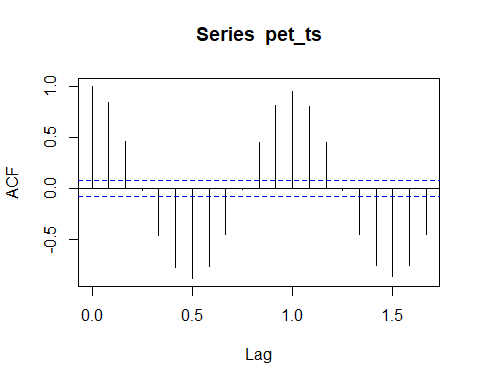
# Time Series  
pet\_ts = ts(san.mon$pet, frequency = 12, start = c(1971,1))  
plot.ts(pet\_ts)



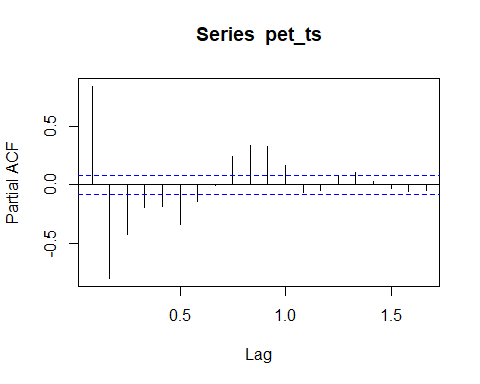
# Decomposing time series into trend,seasonality and randomness assuming additive model  
pet\_ts\_components= decompose(pet\_ts, type = "additive")  
plot(pet\_ts\_components)



# ACF and PACF of data  
# par(mfrow=c(1,3))  
acf(pet\_ts, lag.max=20)



pacf(pet\_ts, lag.max=20)



# Building ARIMA model

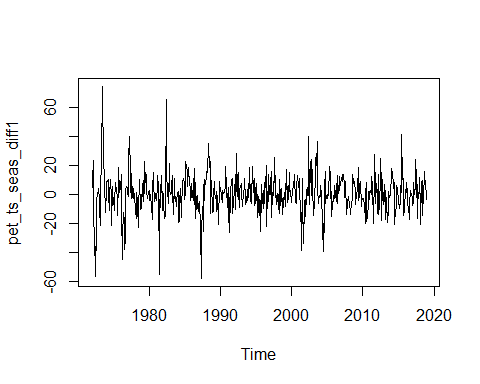
# Calculating ndiffs and nsdiffs  
ndiffs(pet\_ts)

## [1] 0

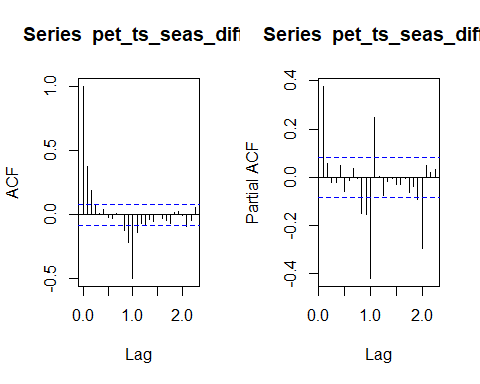
nsdiffs(pet\_ts)

## [1] 1

# # ACF and PACF plots show trend and seasonility  
# # Performing differences to make the data stationary  
# # calculating "d "for Trend  
# par(mfrow = c(1, 1))  
# pet\_ts\_diff1 = diff(pet\_ts, differences = 1)  
# plot(pet\_ts\_diff1)  
#   
# # Checking ACF and PACF for after differencing for above values   
# par(mfrow=c(1,2))  
# acf(pet\_ts\_diff1, lag.max=20)  
# pacf(pet\_ts\_diff1, lag.max=20)  
  
# Performing seasonal differencing and checking for value of D  
par(mfrow = c(1, 1))  
pet\_ts\_seas\_diff1 = diff(pet\_ts, lag = 12, differences=1)  
plot(pet\_ts\_seas\_diff1)

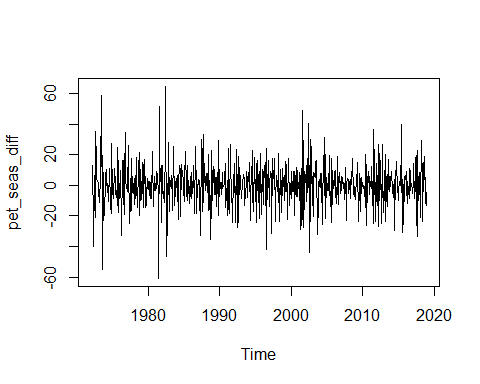


# Checking ACF and PACF for seasonally differenced data  
par(mfrow = c(1, 2))  
acf(pet\_ts\_seas\_diff1)  
pacf(pet\_ts\_seas\_diff1)

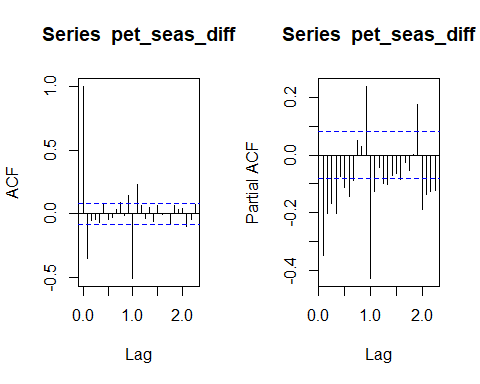


# Differencing seasonaaly differenced data again

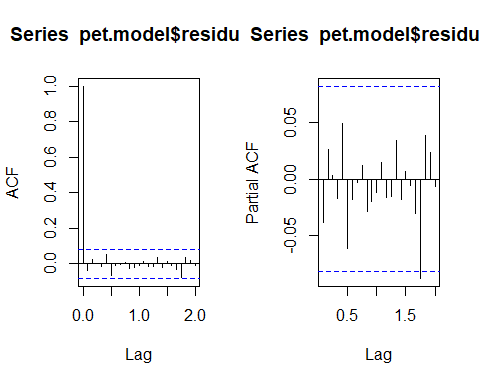
par(mfrow = c(1, 1))  
pet\_seas\_diff <- diff(pet\_ts\_seas\_diff1, differences = 1)  
plot(pet\_seas\_diff)



par(mfrow = c(1, 2))  
acf(pet\_seas\_diff)  
pacf(pet\_seas\_diff)



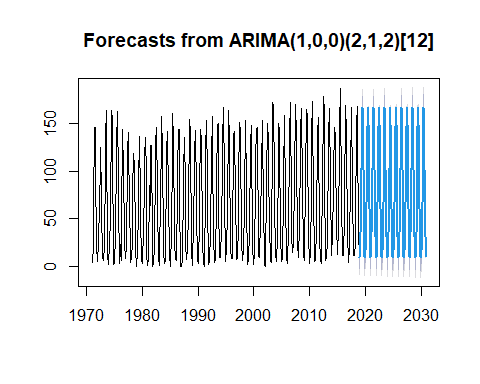
# Implementing ARIMA model  
# auto.arima(pet\_ts)  
pet.model <- Arima(pet\_ts, order = c(1,0,0),seasonal = c(2,1,2), include.drift = FALSE)  
  
# Checking residuals to ensure they are white noise  
par(mfrow = c(1, 2))  
acf(pet.model$residuals, lag.max = 24)  
pacf(pet.model$residuals, lag.max = 24)



Box.test(pet.model$residuals, lag=24, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: pet.model$residuals  
## X-squared = 13.044, df = 24, p-value = 0.9654

# Forecasting using Arima model  
par(mfrow = c(1, 1))  
pet\_ts\_forecast = forecast(pet.model, h= 144)  
plot(pet\_ts\_forecast)



pet\_ts\_forecast$mean

## Jan Feb Mar Apr May Jun  
## 2019 9.207557 16.556789 35.702246 66.910206 102.572792 145.858313  
## 2020 10.057640 16.649402 35.557343 67.327108 102.931708 146.188311  
## 2021 10.120392 16.575138 35.406252 67.435570 103.027980 146.153833  
## 2022 10.085794 16.593376 35.450246 67.397521 102.994438 146.156368  
## 2023 10.093885 16.589962 35.441432 67.405604 103.001550 146.156393  
## 2024 10.092405 16.590511 35.442914 67.404197 103.000313 146.156334  
## 2025 10.092639 16.590433 35.442694 67.404411 103.000502 146.156349  
## 2026 10.092606 16.590443 35.442723 67.404382 103.000476 146.156346  
## 2027 10.092610 16.590442 35.442720 67.404386 103.000479 146.156347  
## 2028 10.092610 16.590442 35.442720 67.404385 103.000479 146.156347  
## 2029 10.092610 16.590442 35.442720 67.404385 103.000479 146.156347  
## 2030 10.092610 16.590442 35.442720 67.404385 103.000479 146.156347  
## Jul Aug Sep Oct Nov Dec  
## 2019 166.106018 148.493831 101.945111 60.593404 31.459637 10.541348  
## 2020 166.179335 148.891655 102.053040 60.807907 31.592175 10.499957  
## 2021 166.190612 148.928303 102.161593 60.858386 31.663095 10.472511  
## 2022 166.186037 148.910136 102.129906 60.840254 31.641111 10.480814  
## 2023 166.187045 148.914308 102.136261 60.844131 31.645610 10.479128  
## 2024 166.186866 148.913551 102.135191 60.843454 31.644844 10.479414  
## 2025 166.186894 148.913670 102.135350 60.843557 31.644959 10.479371  
## 2026 166.186890 148.913654 102.135329 60.843543 31.644944 10.479377  
## 2027 166.186890 148.913656 102.135332 60.843545 31.644945 10.479376  
## 2028 166.186890 148.913656 102.135331 60.843545 31.644945 10.479376  
## 2029 166.186890 148.913656 102.135331 60.843545 31.644945 10.479376  
## 2030 166.186890 148.913656 102.135331 60.843545 31.644945 10.479376

library(pander)  
pander(pet\_ts\_forecast$mean)

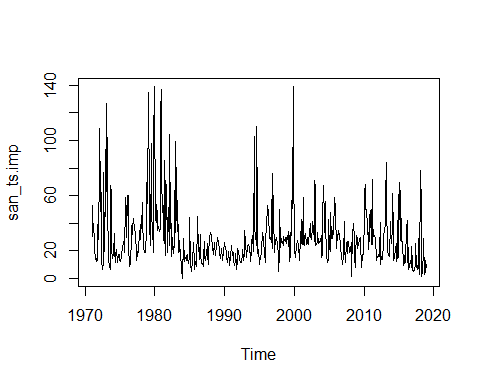
Table continues below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug |
| **2019** | 9.208 | 16.56 | 35.7 | 66.91 | 102.6 | 145.9 | 166.1 | 148.5 |
| **2020** | 10.06 | 16.65 | 35.56 | 67.33 | 102.9 | 146.2 | 166.2 | 148.9 |
| **2021** | 10.12 | 16.58 | 35.41 | 67.44 | 103 | 146.2 | 166.2 | 148.9 |
| **2022** | 10.09 | 16.59 | 35.45 | 67.4 | 103 | 146.2 | 166.2 | 148.9 |
| **2023** | 10.09 | 16.59 | 35.44 | 67.41 | 103 | 146.2 | 166.2 | 148.9 |
| **2024** | 10.09 | 16.59 | 35.44 | 67.4 | 103 | 146.2 | 166.2 | 148.9 |
| **2025** | 10.09 | 16.59 | 35.44 | 67.4 | 103 | 146.2 | 166.2 | 148.9 |
| **2026** | 10.09 | 16.59 | 35.44 | 67.4 | 103 | 146.2 | 166.2 | 148.9 |
| **2027** | 10.09 | 16.59 | 35.44 | 67.4 | 103 | 146.2 | 166.2 | 148.9 |
| **2028** | 10.09 | 16.59 | 35.44 | 67.4 | 103 | 146.2 | 166.2 | 148.9 |
| **2029** | 10.09 | 16.59 | 35.44 | 67.4 | 103 | 146.2 | 166.2 | 148.9 |
| **2030** | 10.09 | 16.59 | 35.44 | 67.4 | 103 | 146.2 | 166.2 | 148.9 |

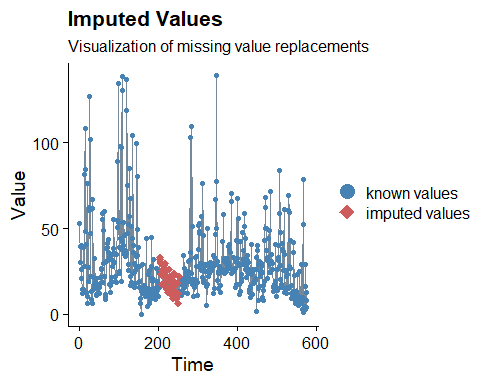
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sep | Oct | Nov | Dec |
| **2019** | 101.9 | 60.59 | 31.46 | 10.54 |
| **2020** | 102.1 | 60.81 | 31.59 | 10.5 |
| **2021** | 102.2 | 60.86 | 31.66 | 10.47 |
| **2022** | 102.1 | 60.84 | 31.64 | 10.48 |
| **2023** | 102.1 | 60.84 | 31.65 | 10.48 |
| **2024** | 102.1 | 60.84 | 31.64 | 10.48 |
| **2025** | 102.1 | 60.84 | 31.64 | 10.48 |
| **2026** | 102.1 | 60.84 | 31.64 | 10.48 |
| **2027** | 102.1 | 60.84 | 31.64 | 10.48 |
| **2028** | 102.1 | 60.84 | 31.64 | 10.48 |
| **2029** | 102.1 | 60.84 | 31.64 | 10.48 |
| **2030** | 102.1 | 60.84 | 31.64 | 10.48 |

# San Vitale —

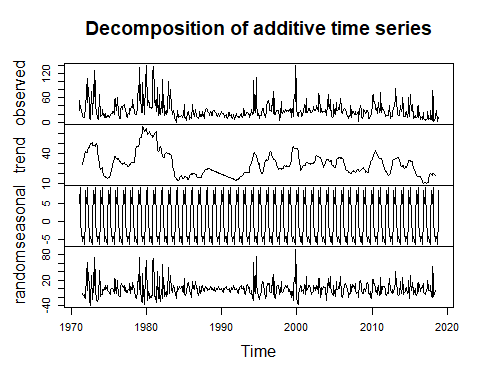
# Time Series  
san\_ts = ts(san.mon$pu, frequency = 12, start = c(1971,1))  
  
# imputeTS: Time Series Missing Value; Imputation in R by Steffen Moritz and Thomas Bartz-Beielstein  
san\_ts.imp <- na\_kalman(san\_ts)  
san\_imp <- ggplot\_na\_imputations(san\_ts, san\_ts.imp) + theme\_cowplot()  
plot.ts(san\_ts.imp)



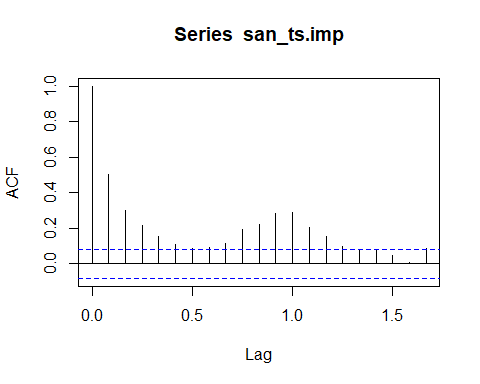
san\_imp



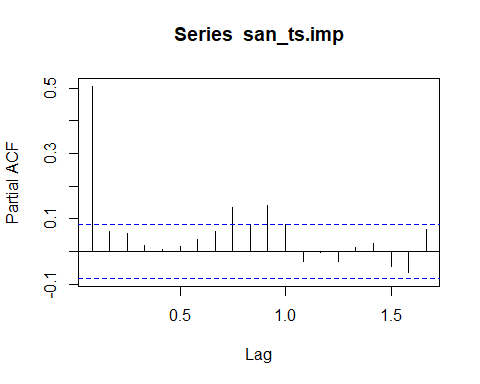
# Decomposing time series into trend,seasonality and randomness assuming multiplicative model  
san\_ts\_components= decompose(san\_ts.imp, type = "additive")  
plot(san\_ts\_components)



# ACF and PACF of data  
# par(mfrow=c(1,3))  
acf(san\_ts.imp, lag.max=20)



pacf(san\_ts.imp, lag.max=20)



# Building ARIMA model

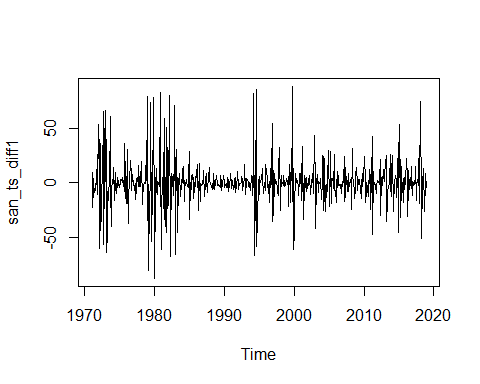
# Calculating ndiffs and nsdiffs  
ndiffs(san\_ts.imp)

## [1] 1

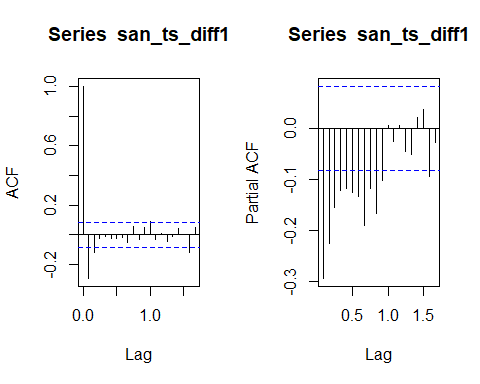
nsdiffs(san\_ts.imp)

## [1] 0

# ACF and PACF plots show trend and seasonility  
# Performing differences to make the data stationary  
# calculating "d "for Trend  
par(mfrow = c(1, 1))  
san\_ts\_diff1 = diff(san\_ts.imp, differences = 1)  
plot(san\_ts\_diff1)



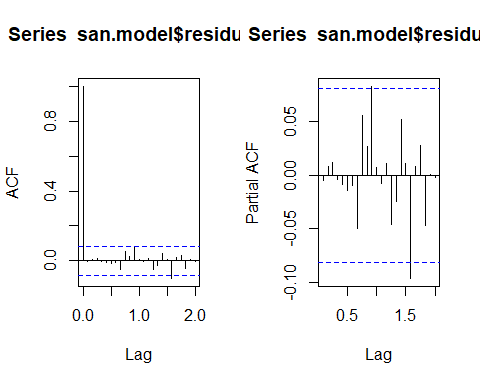
# Checking ACF and PACF for after differencing for above values   
par(mfrow=c(1,2))  
acf(san\_ts\_diff1, lag.max=20)  
pacf(san\_ts\_diff1, lag.max=20)



# # Performing seasonal differencing and checking for value of D  
# par(mfrow = c(1, 1))  
# san\_ts\_seas\_diff1 = diff(san\_ts.imp, lag = 12, differences = 1)  
# plot(san\_ts\_seas\_diff1)  
#   
# # Checking ACF and PACF for seasonally differenced data  
# par(mfrow = c(1, 2))  
# acf(san\_ts\_seas\_diff1)  
# pacf(san\_ts\_seas\_diff1)

# No Differencing seasonally differenced data again

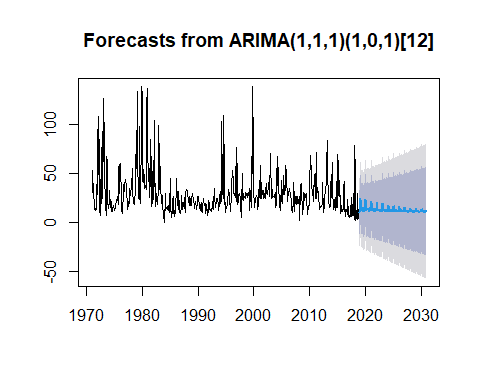
# par(mfrow = c(1, 1))  
# san\_seas\_diff = diff(san\_ts\_seas\_diff1, differences=1)  
# plot(san\_seas\_diff)  
# par(mfrow = c(1, 2))  
# acf(san\_seas\_diff)  
# pacf(san\_seas\_diff)  
  
# Implementing ARIMA model  
# auto.arima(san\_ts.imp)  
san.model= Arima(san\_ts.imp, order = c(1,1,1),seasonal = c(1,0,1), include.drift = FALSE)  
  
# Checking residuals to ensure they are white noise  
par(mfrow = c(1, 2))  
acf(san.model$residuals, lag.max = 24)  
pacf(san.model$residuals, lag.max = 24)



Box.test(san.model$residuals, lag=24, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: san.model$residuals  
## X-squared = 19.145, df = 24, p-value = 0.7442

# Forecasting using Arima model  
par(mfrow = c(1, 1))  
san\_ts\_forecast=forecast(san.model,h= 144)  
plot(san\_ts\_forecast)



san\_ts\_forecast$mean

## Jan Feb Mar Apr May Jun Jul  
## 2019 9.928154 24.268597 23.163267 12.340914 14.675905 13.676554 13.437616  
## 2020 12.128442 23.236614 21.771267 12.198063 14.146109 13.251171 13.032967  
## 2021 11.891579 21.516985 20.247231 11.951885 13.639899 12.864420 12.675342  
## 2022 11.686310 20.026888 18.926625 11.738566 13.201259 12.529294 12.365454  
## 2023 11.508441 18.735694 17.782297 11.553722 12.821170 12.238901 12.096931  
## 2024 11.354315 17.616853 16.790718 11.393551 12.491817 11.987271 11.864252  
## 2025 11.220762 16.647357 15.931498 11.254761 12.206427 11.769229 11.662631  
## 2026 11.105036 15.807273 15.186968 11.134496 11.959131 11.580292 11.487922  
## 2027 11.004757 15.079326 14.541821 11.030285 11.744845 11.416574 11.336535  
## 2028 10.917864 14.448547 13.982790 10.939984 11.559163 11.274711 11.205355  
## 2029 10.842569 13.901967 13.498380 10.861737 11.398266 11.151783 11.091685  
## 2030 10.777326 13.428345 13.078631 10.793935 11.258846 11.045265 10.993189  
## Aug Sep Oct Nov Dec  
## 2019 13.878947 10.831496 11.747103 15.294069 14.027815  
## 2020 13.411089 10.768767 11.561519 14.634779 13.537454  
## 2021 13.002992 10.713374 11.400307 14.063340 13.112489  
## 2022 12.649368 10.665376 11.260615 13.568178 12.744250  
## 2023 12.342948 10.623784 11.139569 13.139112 12.425164  
## 2024 12.077429 10.587744 11.034681 12.767319 12.148670  
## 2025 11.847352 10.556515 10.943793 12.445153 11.909084  
## 2026 11.647987 10.529455 10.865038 12.165992 11.701479  
## 2027 11.475234 10.506006 10.796795 11.924094 11.521585  
## 2028 11.325540 10.485688 10.737661 11.714485 11.365704  
## 2029 11.195828 10.468082 10.686421 11.532855 11.230631  
## 2030 11.083430 10.452826 10.642020 11.375470 11.113587

library(pander)  
pander(san\_ts\_forecast$mean)

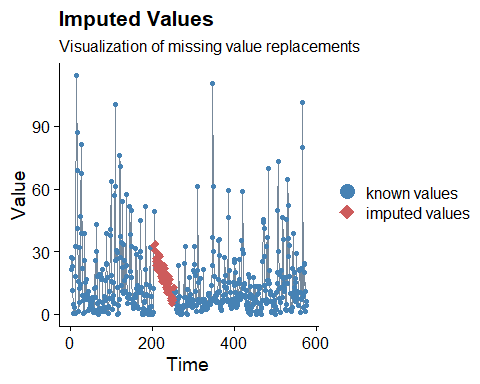
Table continues below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug |
| **2019** | 9.928 | 24.27 | 23.16 | 12.34 | 14.68 | 13.68 | 13.44 | 13.88 |
| **2020** | 12.13 | 23.24 | 21.77 | 12.2 | 14.15 | 13.25 | 13.03 | 13.41 |
| **2021** | 11.89 | 21.52 | 20.25 | 11.95 | 13.64 | 12.86 | 12.68 | 13 |
| **2022** | 11.69 | 20.03 | 18.93 | 11.74 | 13.2 | 12.53 | 12.37 | 12.65 |
| **2023** | 11.51 | 18.74 | 17.78 | 11.55 | 12.82 | 12.24 | 12.1 | 12.34 |
| **2024** | 11.35 | 17.62 | 16.79 | 11.39 | 12.49 | 11.99 | 11.86 | 12.08 |
| **2025** | 11.22 | 16.65 | 15.93 | 11.25 | 12.21 | 11.77 | 11.66 | 11.85 |
| **2026** | 11.11 | 15.81 | 15.19 | 11.13 | 11.96 | 11.58 | 11.49 | 11.65 |
| **2027** | 11 | 15.08 | 14.54 | 11.03 | 11.74 | 11.42 | 11.34 | 11.48 |
| **2028** | 10.92 | 14.45 | 13.98 | 10.94 | 11.56 | 11.27 | 11.21 | 11.33 |
| **2029** | 10.84 | 13.9 | 13.5 | 10.86 | 11.4 | 11.15 | 11.09 | 11.2 |
| **2030** | 10.78 | 13.43 | 13.08 | 10.79 | 11.26 | 11.05 | 10.99 | 11.08 |

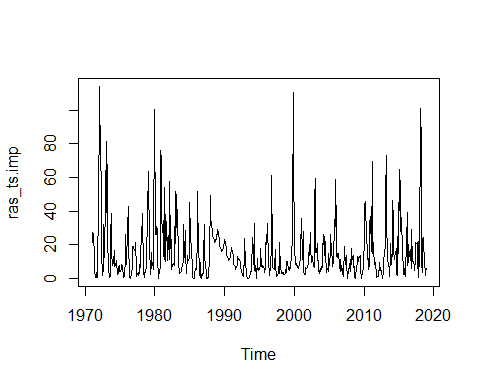
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sep | Oct | Nov | Dec |
| **2019** | 10.83 | 11.75 | 15.29 | 14.03 |
| **2020** | 10.77 | 11.56 | 14.63 | 13.54 |
| **2021** | 10.71 | 11.4 | 14.06 | 13.11 |
| **2022** | 10.67 | 11.26 | 13.57 | 12.74 |
| **2023** | 10.62 | 11.14 | 13.14 | 12.43 |
| **2024** | 10.59 | 11.03 | 12.77 | 12.15 |
| **2025** | 10.56 | 10.94 | 12.45 | 11.91 |
| **2026** | 10.53 | 10.87 | 12.17 | 11.7 |
| **2027** | 10.51 | 10.8 | 11.92 | 11.52 |
| **2028** | 10.49 | 10.74 | 11.71 | 11.37 |
| **2029** | 10.47 | 10.69 | 11.53 | 11.23 |
| **2030** | 10.45 | 10.64 | 11.38 | 11.11 |

# Rasponi —-

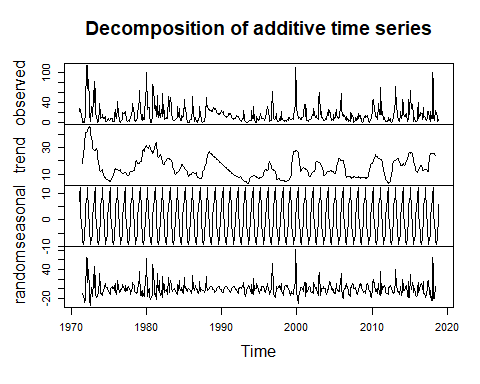
# Time Series  
ras\_ts = ts(ras.mon$pu, frequency = 12, start = c(1971,1))  
  
# imputeTS: Time Series Missing Value; Imputation in R by Steffen Moritz and Thomas Bartz-Beielstein  
ras\_ts.imp <- na\_kalman(ras\_ts)  
ggplot\_na\_imputations(ras\_ts, ras\_ts.imp) + theme\_cowplot()



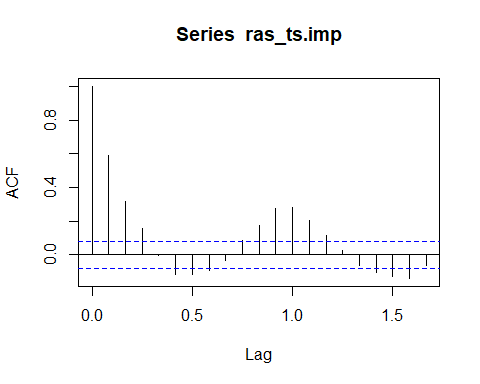
plot.ts(ras\_ts.imp)



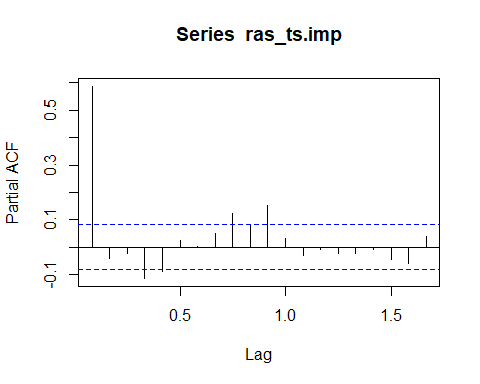
# Decomposing time series into trend,seasonality and randomness assuming additive model  
ras\_ts\_components= decompose(ras\_ts.imp, type = "additive")  
plot(ras\_ts\_components)



# ACF and PACF of data  
# par(mfrow=c(1,3))  
acf(ras\_ts.imp, lag.max=20)



pacf(ras\_ts.imp, lag.max=20)



# Building ARIMA model

# Calculating ndiffs and nsdiffs  
ndiffs(ras\_ts.imp)

## [1] 0

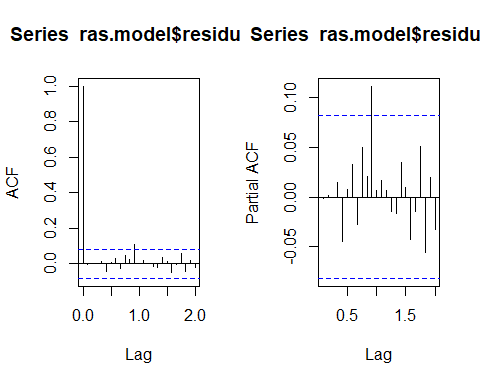
nsdiffs(ras\_ts.imp)

## [1] 0

# # ACF and PACF plots show trend and seasonility  
# # Performing differences to make the data stationary  
# # calculating "d "for Trend  
# par(mfrow = c(1, 1))  
# ras\_ts\_diff1 = diff(ras\_ts.imp, differences = 1)  
# plot(ras\_ts\_diff1)  
#   
# # Checking ACF and PACF for after differencing for above values   
# par(mfrow=c(1,2))  
# acf(ras\_ts\_diff1, lag.max=20)  
# pacf(ras\_ts\_diff1, lag.max=20)  
#   
# # Performing seasonal differencing and checking for value of D  
# par(mfrow = c(1, 1))  
# ras\_ts\_seas\_diff1=diff(ras\_ts.imp, lag = 12, differences=1)  
# plot(ras\_ts\_seas\_diff1)  
#   
# # Checking ACF and PACF for seasonally differenced data  
# par(mfrow = c(1, 2))  
# acf(ras\_ts\_seas\_diff1)  
# pacf(ras\_ts\_seas\_diff1)

# No Differencing seasonaaly differenced data again

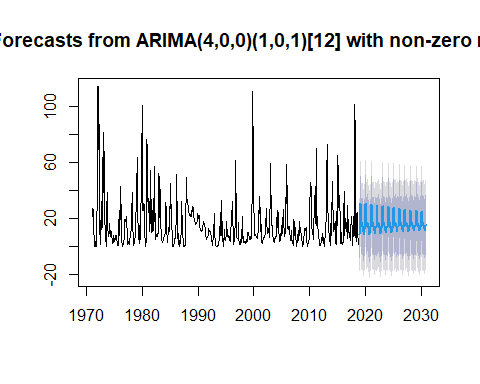
# Implementing ARIMA model  
# auto.arima(ras\_ts.imp)  
ras.model= Arima(ras\_ts.imp, order = c(4,0,0),seasonal = c(1,0,1), include.drift = FALSE)  
  
# Checking residuals to ensure they are white noise  
par(mfrow = c(1, 2))  
acf(ras.model$residuals, lag.max = 24)  
pacf(ras.model$residuals, lag.max = 24)



Box.test(ras.model$residuals, lag=24, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: ras.model$residuals  
## X-squared = 17.77, df = 24, p-value = 0.814

# Forecasting using Arima model  
par(mfrow = c(1, 1))  
ras\_ts\_forecast=forecast(ras.model,h= 144)  
plot(ras\_ts\_forecast)



ras\_ts\_forecast$mean

## Jan Feb Mar Apr May Jun Jul  
## 2019 8.549845 28.684461 30.664473 14.905360 14.318882 17.294337 14.852344  
## 2020 13.172176 30.398900 31.102435 15.465967 14.604887 17.297555 14.929581  
## 2021 13.333694 29.686749 30.354887 15.512171 14.694837 17.250848 15.003058  
## 2022 13.488156 29.011216 29.645442 15.556065 14.780215 17.206496 15.072792  
## 2023 13.634779 28.369969 28.972006 15.597731 14.861259 17.164395 15.138987  
## 2024 13.773960 27.761268 28.332749 15.637283 14.938190 17.124431 15.201822  
## 2025 13.906076 27.183462 27.725937 15.674827 15.011217 17.086495 15.261467  
## 2026 14.031487 26.634983 27.149925 15.710465 15.080537 17.050485 15.318086  
## 2027 14.150533 26.114341 26.603147 15.744295 15.146338 17.016302 15.371831  
## 2028 14.263537 25.620125 26.084121 15.776408 15.208800 16.983855 15.422848  
## 2029 14.370805 25.150992 25.591439 15.806891 15.268092 16.953054 15.471276  
## 2030 14.472629 24.705670 25.123762 15.835826 15.324374 16.923816 15.517245  
## Aug Sep Oct Nov Dec  
## 2019 13.342140 10.671085 8.348327 14.137957 14.086337  
## 2020 13.484191 10.949249 8.747857 14.246945 14.200171  
## 2021 13.631023 11.224738 9.135075 14.355056 14.310656  
## 2022 13.770395 11.486240 9.502637 14.457680 14.415532  
## 2023 13.902692 11.734469 9.851544 14.555094 14.515086  
## 2024 14.028275 11.970100 10.182742 14.647564 14.609587  
## 2025 14.147484 12.193771 10.497130 14.735342 14.699292  
## 2026 14.260643 12.406090 10.795562 14.818663 14.784443  
## 2027 14.368058 12.607632 11.078846 14.897756 14.865273  
## 2028 14.470021 12.798946 11.347753 14.972835 14.942000  
## 2029 14.566809 12.980549 11.603011 15.044103 15.014833  
## 2030 14.658685 13.152935 11.845314 15.111754 15.083970

library(pander)  
pander(ras\_ts\_forecast$mean)

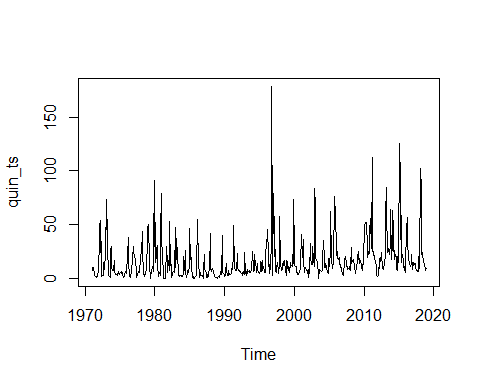
Table continues below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug |
| **2019** | 8.55 | 28.68 | 30.66 | 14.91 | 14.32 | 17.29 | 14.85 | 13.34 |
| **2020** | 13.17 | 30.4 | 31.1 | 15.47 | 14.6 | 17.3 | 14.93 | 13.48 |
| **2021** | 13.33 | 29.69 | 30.35 | 15.51 | 14.69 | 17.25 | 15 | 13.63 |
| **2022** | 13.49 | 29.01 | 29.65 | 15.56 | 14.78 | 17.21 | 15.07 | 13.77 |
| **2023** | 13.63 | 28.37 | 28.97 | 15.6 | 14.86 | 17.16 | 15.14 | 13.9 |
| **2024** | 13.77 | 27.76 | 28.33 | 15.64 | 14.94 | 17.12 | 15.2 | 14.03 |
| **2025** | 13.91 | 27.18 | 27.73 | 15.67 | 15.01 | 17.09 | 15.26 | 14.15 |
| **2026** | 14.03 | 26.63 | 27.15 | 15.71 | 15.08 | 17.05 | 15.32 | 14.26 |
| **2027** | 14.15 | 26.11 | 26.6 | 15.74 | 15.15 | 17.02 | 15.37 | 14.37 |
| **2028** | 14.26 | 25.62 | 26.08 | 15.78 | 15.21 | 16.98 | 15.42 | 14.47 |
| **2029** | 14.37 | 25.15 | 25.59 | 15.81 | 15.27 | 16.95 | 15.47 | 14.57 |
| **2030** | 14.47 | 24.71 | 25.12 | 15.84 | 15.32 | 16.92 | 15.52 | 14.66 |

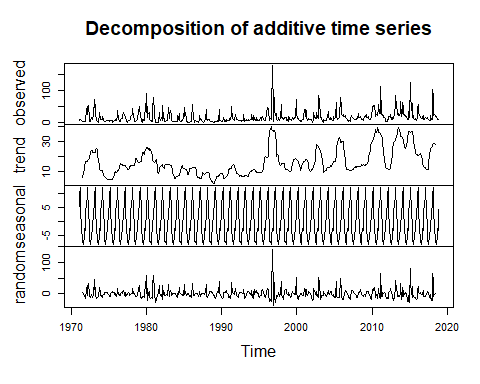
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sep | Oct | Nov | Dec |
| **2019** | 10.67 | 8.348 | 14.14 | 14.09 |
| **2020** | 10.95 | 8.748 | 14.25 | 14.2 |
| **2021** | 11.22 | 9.135 | 14.36 | 14.31 |
| **2022** | 11.49 | 9.503 | 14.46 | 14.42 |
| **2023** | 11.73 | 9.852 | 14.56 | 14.52 |
| **2024** | 11.97 | 10.18 | 14.65 | 14.61 |
| **2025** | 12.19 | 10.5 | 14.74 | 14.7 |
| **2026** | 12.41 | 10.8 | 14.82 | 14.78 |
| **2027** | 12.61 | 11.08 | 14.9 | 14.87 |
| **2028** | 12.8 | 11.35 | 14.97 | 14.94 |
| **2029** | 12.98 | 11.6 | 15.04 | 15.01 |
| **2030** | 13.15 | 11.85 | 15.11 | 15.08 |

# Quinto —

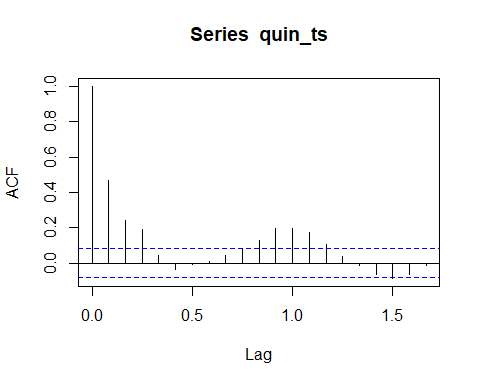
# Time Series  
quin\_ts = ts(quin.mon$pu, frequency = 12, start = c(1971,1))  
plot.ts(quin\_ts)



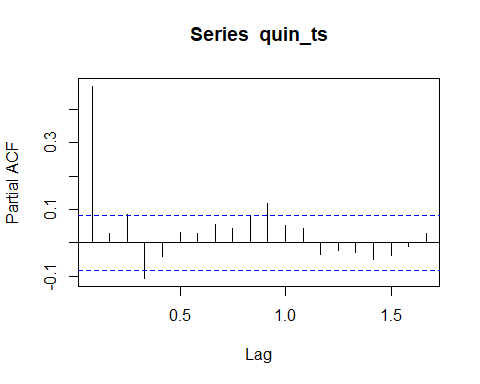
# Decomposing time series into trend,seasonality and randomness assuming additive model  
quin\_ts\_components= decompose(quin\_ts, type = "additive")  
plot(quin\_ts\_components)



# ACF and PACF of data  
# par(mfrow=c(1,3))  
acf(quin\_ts, lag.max=20)



pacf(quin\_ts, lag.max=20)



# Building ARIMA model

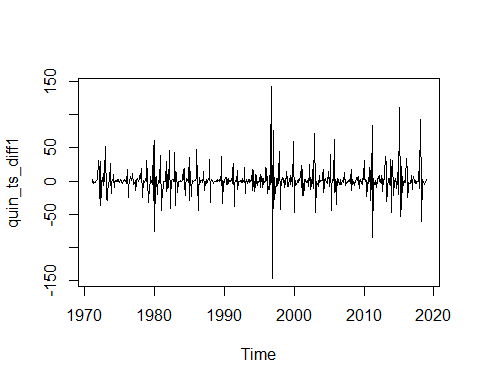
# Calculating ndiffs and nsdiffs  
ndiffs(quin\_ts)

## [1] 1

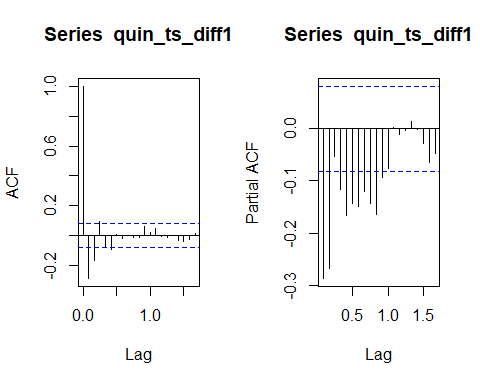
nsdiffs(quin\_ts)

## [1] 0

# ACF and PACF plots show trend and seasonility  
# Performing differences to make the data stationary  
# calculating "d "for Trend  
par(mfrow = c(1, 1))  
quin\_ts\_diff1 = diff(quin\_ts, differences = 1)  
plot(quin\_ts\_diff1)



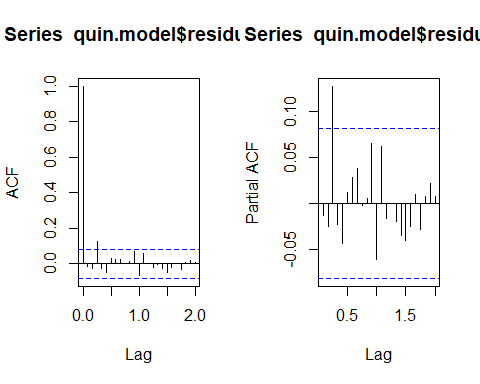
# Checking ACF and PACF for after differencing for above values   
par(mfrow=c(1,2))  
acf(quin\_ts\_diff1, lag.max=20)  
pacf(quin\_ts\_diff1, lag.max=20)



# # Performing seasonal differencing and checking for value of D  
# par(mfrow = c(1, 1))  
# quin\_ts\_seas\_diff1 = diff(quin\_ts, lag = 12, differences=1)  
# plot(quin\_ts\_seas\_diff1)  
#   
# # Checking ACF and PACF for seasonally differenced data  
# par(mfrow = c(1, 2))  
# acf(quin\_ts\_seas\_diff1)  
# pacf(quin\_ts\_seas\_diff1)

# No Differencing seasonally differenced data again

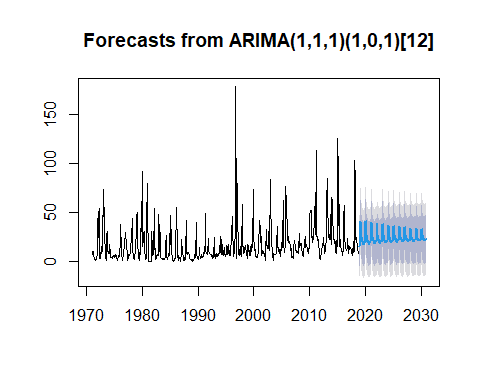
# par(mfrow = c(1, 1))  
# quin\_seas\_diff = diff(quin\_ts\_seas\_diff1, differences=1)  
# plot(quin\_seas\_diff)  
# par(mfrow = c(1, 2))  
# acf(quin\_seas\_diff)  
# pacf(quin\_seas\_diff)  
  
# Implementing ARIMA model  
# auto.arima(quin\_ts)  
quin.model <- Arima(quin\_ts, order = c(1,1,1),seasonal = c(1,0,1), include.drift = FALSE)  
  
# Checking residuals to ensure they are white noise  
par(mfrow = c(1, 2))  
acf(quin.model$residuals, lag.max = 24)  
pacf(quin.model$residuals, lag.max = 24)



Box.test(quin.model$residuals, lag=24, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: quin.model$residuals  
## X-squared = 23.445, df = 24, p-value = 0.4937

# Forecasting using Arima model  
par(mfrow = c(1, 1))  
quin\_ts\_forecast <- forecast(quin.model, h= 144)  
plot(quin\_ts\_forecast)



quin\_ts\_forecast$mean

## Jan Feb Mar Apr May Jun Jul Aug  
## 2019 15.20780 40.09764 40.89984 25.76465 24.94966 19.99713 20.00200 18.25393  
## 2020 19.39763 40.47735 40.37109 25.91321 25.01609 20.34228 20.32485 18.68573  
## 2021 19.74657 39.40638 39.30726 25.82322 24.98653 20.62753 20.61127 19.08256  
## 2022 20.07194 38.40752 38.31508 25.73929 24.95895 20.89356 20.87841 19.45266  
## 2023 20.37540 37.47594 37.38972 25.66101 24.93323 21.14168 21.12754 19.79783  
## 2024 20.65842 36.60711 36.52670 25.58800 24.90925 21.37308 21.35990 20.11975  
## 2025 20.92237 35.79680 35.72181 25.51991 24.88688 21.58890 21.57660 20.41999  
## 2026 21.16855 35.04107 34.97113 25.45641 24.86601 21.79018 21.77871 20.70000  
## 2027 21.39814 34.33624 34.27101 25.39718 24.84655 21.97790 21.96721 20.96116  
## 2028 21.61227 33.67889 33.61806 25.34195 24.82841 22.15298 22.14300 21.20472  
## 2029 21.81198 33.06582 33.00908 25.29043 24.81148 22.31626 22.30696 21.43188  
## 2030 21.99823 32.49404 32.44113 25.24238 24.79570 22.46855 22.45987 21.64373  
## Sep Oct Nov Dec  
## 2019 16.76948 17.61664 19.97059 18.92648  
## 2020 17.29774 18.08642 20.28125 19.30725  
## 2021 17.78806 18.52362 20.57061 19.66221  
## 2022 18.24536 18.93137 20.84048 19.99327  
## 2023 18.67185 19.31165 21.09217 20.30203  
## 2024 19.06961 19.66632 21.32691 20.58999  
## 2025 19.44059 19.99710 21.54584 20.85855  
## 2026 19.78657 20.30560 21.75002 21.10903  
## 2027 20.10925 20.59332 21.94044 21.34263  
## 2028 20.41020 20.86166 22.11804 21.56050  
## 2029 20.69087 21.11193 22.28368 21.76369  
## 2030 20.95264 21.34533 22.43816 21.95320

library(pander)  
pander(quin\_ts\_forecast$mean)

Table continues below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug |
| **2019** | 15.21 | 40.1 | 40.9 | 25.76 | 24.95 | 20 | 20 | 18.25 |
| **2020** | 19.4 | 40.48 | 40.37 | 25.91 | 25.02 | 20.34 | 20.32 | 18.69 |
| **2021** | 19.75 | 39.41 | 39.31 | 25.82 | 24.99 | 20.63 | 20.61 | 19.08 |
| **2022** | 20.07 | 38.41 | 38.32 | 25.74 | 24.96 | 20.89 | 20.88 | 19.45 |
| **2023** | 20.38 | 37.48 | 37.39 | 25.66 | 24.93 | 21.14 | 21.13 | 19.8 |
| **2024** | 20.66 | 36.61 | 36.53 | 25.59 | 24.91 | 21.37 | 21.36 | 20.12 |
| **2025** | 20.92 | 35.8 | 35.72 | 25.52 | 24.89 | 21.59 | 21.58 | 20.42 |
| **2026** | 21.17 | 35.04 | 34.97 | 25.46 | 24.87 | 21.79 | 21.78 | 20.7 |
| **2027** | 21.4 | 34.34 | 34.27 | 25.4 | 24.85 | 21.98 | 21.97 | 20.96 |
| **2028** | 21.61 | 33.68 | 33.62 | 25.34 | 24.83 | 22.15 | 22.14 | 21.2 |
| **2029** | 21.81 | 33.07 | 33.01 | 25.29 | 24.81 | 22.32 | 22.31 | 21.43 |
| **2030** | 22 | 32.49 | 32.44 | 25.24 | 24.8 | 22.47 | 22.46 | 21.64 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sep | Oct | Nov | Dec |
| **2019** | 16.77 | 17.62 | 19.97 | 18.93 |
| **2020** | 17.3 | 18.09 | 20.28 | 19.31 |
| **2021** | 17.79 | 18.52 | 20.57 | 19.66 |
| **2022** | 18.25 | 18.93 | 20.84 | 19.99 |
| **2023** | 18.67 | 19.31 | 21.09 | 20.3 |
| **2024** | 19.07 | 19.67 | 21.33 | 20.59 |
| **2025** | 19.44 | 20 | 21.55 | 20.86 |
| **2026** | 19.79 | 20.31 | 21.75 | 21.11 |
| **2027** | 20.11 | 20.59 | 21.94 | 21.34 |
| **2028** | 20.41 | 20.86 | 22.12 | 21.56 |
| **2029** | 20.69 | 21.11 | 22.28 | 21.76 |
| **2030** | 20.95 | 21.35 | 22.44 | 21.95 |