ENV 790.30 - Time Series Analysis for Energy Data | Spring 2024

Assignment 4 - Due date 02/12/24

Shubhangi Gupta

Directions

You should open the .rmd file corresponding to this assignment on RStudio. The file is available on our class repository on Github. And to do so you will need to fork our repository and link it to your RStudio.

Once you have the file open on your local machine the first thing you will do is rename the file such that it includes your first and last name (e.g., "LuanaLima_TSA_A04_Sp23.Rmd"). Then change "Student Name" on line 4 with your name.

Then you will start working through the assignment by **creating code and output** that answer each question. Be sure to use this assignment document. Your report should contain the answer to each question and any plots/tables you obtained (when applicable).

When you have completed the assignment, **Knit** the text and code into a single PDF file. Submit this pdf using Sakai.

R packages needed for this assignment: "xlsx" or "readxl", "ggplot2", "forecast","tseries", and "Kendall". Install these packages, if you haven't done yet. Do not forget to load them before running your script, since they are NOT default packages.\

```r
#Load/install required package here
library(readxl)
library(ggplot2)
library(forecast)
```

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

```r
library(tseries)
library(Kendall)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr   1.1.3     v stringr 1.5.0
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.2     v tidyr   1.3.0
## v readr   2.1.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(cowplot)
```

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

Questions

Consider the same data you used for A3 from the spreadsheet "Table_10.1_Renewable_Energy_Production_and_Consumpti
The data comes from the US Energy Information and Administration and corresponds to the January 2021 Monthly Energy Review. For this assignment you will work only with the column "Total Renewable Energy Production".

```r
getwd()
```

```
## [1] "/home/guest/RStudio Project Folder/TSA_Sp24"
```

```r
#Importing data set - using readxl package
REDataset_Raw <- read.csv("Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.csv")

#Subsetting the date and RE columns into a new object
REDataset <- REDataset_Raw[,c(1, 5)]

#Changing the column names
colnames(REDataset)<-c("Date", "RE Production")

#Converting the date column to a date object
REDataset$Date <-ym(REDataset$Date)

#Checking results
glimpse(REDataset)
```

```
## Rows: 609
## Columns: 2
## $ Date            <date> 1973-01-01, 1973-02-01, 1973-03-01, 1973-04-01, 1973-~
## $ `RE Production` <dbl> 219.839, 197.330, 218.686, 209.330, 215.982, 208.249, ~
```

Stochastic Trend and Stationarity Tests

Q1

Difference the “Total Renewable Energy Production” series using function diff(). Function diff() is from package base and take three main arguments: * *x* vector containing values to be differenced; * *lag* integer indicating with lag to use; * *differences* integer indicating how many times series should be differenced.

Try differencing at lag 1 only once, i.e., make `lag=1` and `differences=1`. Plot the differenced series Do the series still seem to have trend?

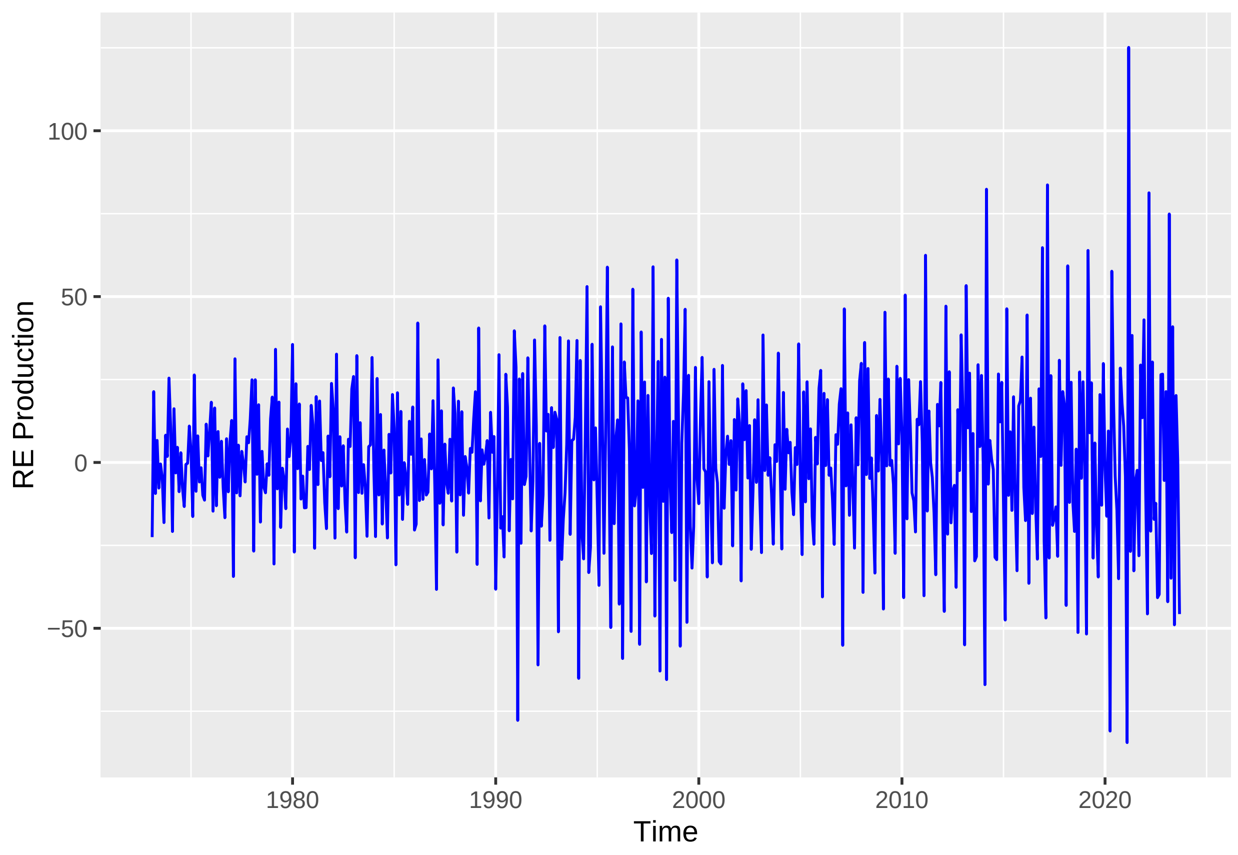
```
#Differencing the RE dataset
RE_diff <- diff(REDataset$`RE Production`, lag=1, differences=1)

#Converting to a df to be able to plot it
RE_diff_dates <- REDataset[c(2:609),1] #making a date column starting from Feb 1st 1973
RE_diff_df <- as.data.frame(RE_diff) #converting numeric vector to a df
RE_diff_df <- cbind(RE_diff_dates, RE_diff_df) #combining the date & differenced data columns
glimpse(RE_diff_df)
```

```
## Rows: 608
## Columns: 2
## $ RE_diff_dates <date> 1973-02-01, 1973-03-01, 1973-04-01, 1973-05-01, 1973-06~
## $ RE_diff       <dbl> -22.509, 21.356, -9.356, 6.652, -7.733, -0.449, -4.368, ~
```

```
#Plotting the differenced dataset

ggplot(RE_diff_df)+
  geom_line(aes(y=RE_diff, x=RE_diff_dates), col="blue")+
  xlab("Time")+
  ylab("RE Production")
```



```
#Converting the differenced dataset into a time series
RE_diff_ts <- ts(RE_diff, frequency = 12, start = c(1973, 01))

#Checking results
glimpse(RE_diff_ts)
```

```
##  Time-Series [1:608] from 1973 to 2024: -22.51 21.36 -9.36 6.65 -7.73 ...
```

Analysis: the differenced series does not seem to have a trend anymore as the points hover around 0 (with some outliers) with no obvious trend.

Q2

Copy and paste part of your code for A3 where you run the regression for Total Renewable Energy Production and subtract that from the orinal series. This should be the code for Q3 and Q4. make sure you use the same name for you time series object that you had in A3.

```
#Converting the dataset into a time series
RE_ts <- ts(REDataset[,2], frequency=12, start=c(1973,01))

#Fitting a regression model on the total RE production dataset
nobs <- nrow(REDataset)
t <- c(1:nobs)
RE_lm <- lm(RE_ts~t)
print(summary(RE_lm))
```

```
##
## Call:
## lm(formula = RE_ts ~ t)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -148.27  -35.63   11.58   41.51  144.27
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 180.98940    4.90151   36.92   <2e-16 ***
## t             0.70404    0.01392   50.57   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60.41 on 607 degrees of freedom
## Multiple R-squared:  0.8081, Adjusted R-squared:  0.8078
## F-statistic:  2557 on 1 and 607 DF,  p-value: < 2.2e-16
```

```
B0_RE <- as.numeric(RE_lm$coefficients[1])
B0_RE
```

```
## [1] 180.9894
```

```
B1_RE <- as.numeric(RE_lm$coefficients[2])
B1_RE
```

```
## [1] 0.7040391
```

```
#Detrending the RE dataset using the regression results
RE_detrended_vector <- REDataset[,2] - (B0_RE + B1_RE*t)
RE_detrended <- as.data.frame(RE_detrended_vector)
colnames(RE_detrended) <- "Detrended RE"
head(RE_detrended)
```

```
##   Detrended RE
## 1     38.14556
## 2     14.93252
## 3     35.58448
## 4     25.52444
## 5     31.47240
## 6     23.03536
```

```
#converting the detrended dataset into a time series
RE_detrended_ts <- ts(RE_detrended, frequency = 12, start=c(1973, 01))
glimpse(RE_detrended_ts)
```

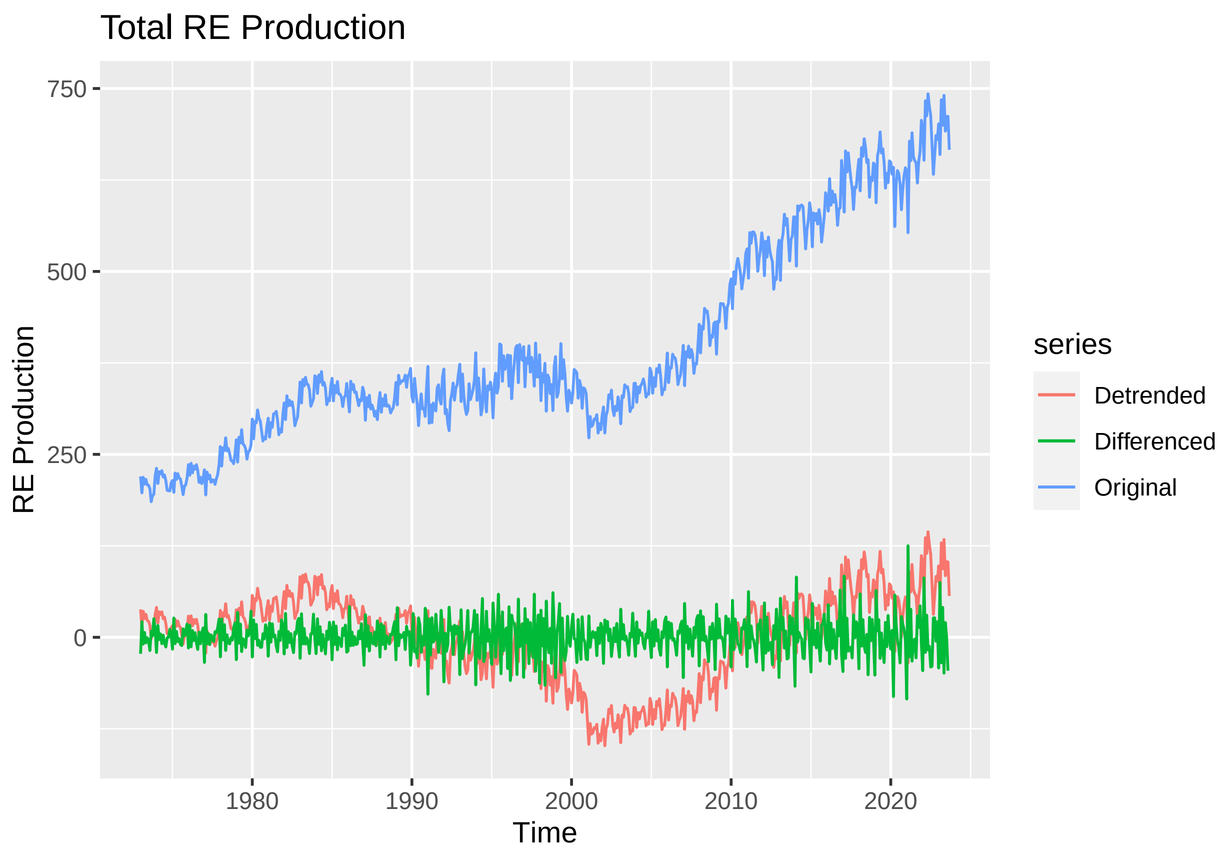
```
##  Time-Series [1:609, 1] from 1973 to 2024: 38.1 14.9 35.6 25.5 31.5 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "Detrended RE"
```

Q3

Now let's compare the differenced series with the detrended series you calculated on A3. In other words, for the "Total Renewable Energy Production" compare the differenced series from Q1 with the series you detrended in Q2 using linear regression.

Using autoplot() + autolayer() create a plot that shows the three series together. Make sure your plot has a legend. The easiest way to do it is by adding the `series=` argument to each autoplot and autolayer function. Look at the key for A03 for an example.

```r
autoplot(RE_ts, series = "Original")+
  autolayer(RE_detrended_ts, series = "Detrended")+
  autolayer(RE_diff_ts, series = "Differenced")+
  ylab("RE Production")+
  xlab("Time")+
  ggtitle("Total RE Production")
```



Q4

Plot the ACF for the three series and compare the plots. Add the argument `ylim=c(-0.5,1)` to the autoplot() or Acf() function - whichever you are using to generate the plots - to make sure all three y axis have the same limits. Which method do you think was more efficient in eliminating the trend? The linear regression or differencing?

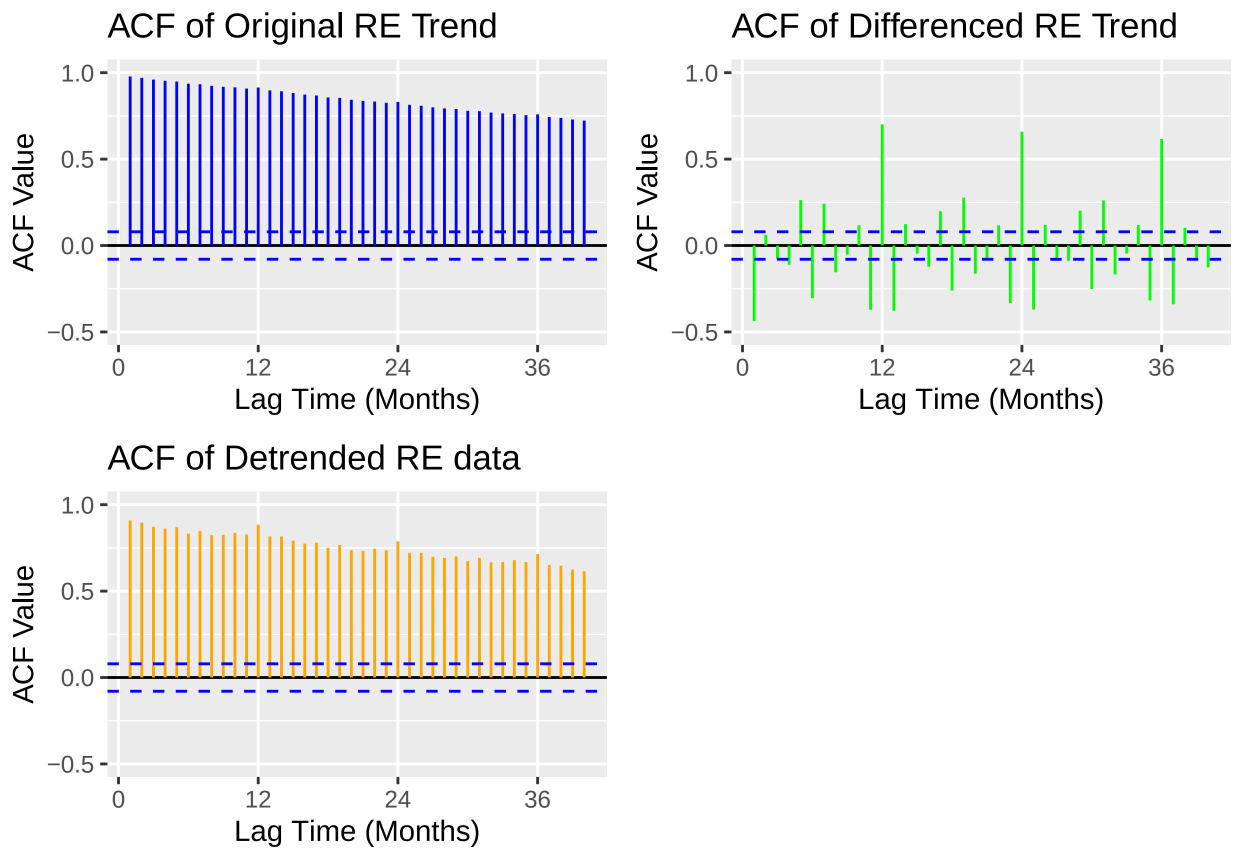
```r
RE_original_ACF <- Acf(RE_ts, lag.max=40, type="correlation", plot=FALSE)
RE_diff_ACF <- Acf(RE_diff_ts, lag.max=40, type="correlation", plot=FALSE, ylim=c(-0.5,1))
RE_detrended_ACF <- Acf(RE_detrended_ts, lag.max=40, type="correlation", plot=FALSE, ylim=c(-0.5,1))

RE_original_ACF_plot<- autoplot(RE_original_ACF,col = "blue")+
  xlab ("Lag Time (Months)") + ylab ("ACF Value") +
  ggtitle("ACF of Original RE Trend")+
  coord_cartesian(ylim=c(-0.5,1))

RE_diff_ACF_plot<- autoplot(RE_diff_ACF,col = "green")+
  xlab ("Lag Time (Months)") + ylab ("ACF Value") +
  ggtitle("ACF of Differenced RE Trend")+
  coord_cartesian(ylim=c(-0.5,1))

RE_detrended_ACF_plot<- autoplot(RE_detrended_ACF, col = "orange")+
  xlab ("Lag Time (Months)") + ylab ("ACF Value") +
  ggtitle("ACF of Detrended RE data")+
  coord_cartesian(ylim=c(-0.5,1))

plot_grid(RE_original_ACF_plot, RE_diff_ACF_plot, RE_detrended_ACF_plot)
```



Analysis: the differencing was a lot more effective than the regression in detrending the data.

Q5

Compute the Seasonal Mann-Kendall and ADF Test for the original "Total Renewable Energy Production" series. Ask R to print the results. Interpret the results for both test. What is the conclusion from the Seasonal Mann Kendall test? What's the conclusion for the ADF test? Do they match what you observed in Q2? Recall that having a unit root means the series has a stochastic trend. And when a series has stochastic trend we need to use a different procedure to remove the trend.

```
#ADF Test: Tests for stochasticity
print(adf.test(REDataset$`RE Production`), alternative="stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  REDataset$‘RE Production‘
## Dickey-Fuller = -1.24, Lag order = 8, p-value = 0.9
## alternative hypothesis: stationary
```

```
#Seasonal Mann-Kendall Test
print(summary(SeasonalMannKendall(RE_ts)))
```

```
## Score =  11865 , Var(Score) = 179299
## denominator =  15149.5
## tau = 0.783, 2-sided pvalue =< 2.22e-16
## NULL
```

Analysis: The ADF test failed as the p-value = 0.9 <= 0.05 significance level. This indicates that the null hypothesis that there is a unit root is true meaning that the original RE dataset could be stochastic. To test if the data is stationary or follows a trend, we use the Seasonal Mann-Kendall test. In this case, the p-value of the SMK test is 2.22 x 10^-16 which is < 0.05 and thus we reject the null hypothesis that the data is stationary and the alternative hypothesis that the data may follow a trend may be true.

Q6

Aggregate the original "Total Renewable Energy Production" series by year. You can use the same procedure we used in class. Store series in a matrix where rows represent months and columns represent years. And then take the columns mean using function colMeans(). Recall the goal is the remove the seasonal variation from the series to check for trend. Convert the accumulates yearly series into a time series object and plot the series using autoplot().

```
#Removing 9 datapoints from 2023 that were preventing it from having 12 datapoints in the last column
RE_matrix_df <- REDataset[-c(601:609),]
tail(RE_matrix_df)
```

```
##           Date RE Production
## 595 2022-07-01       713.262
## 596 2022-08-01       672.494
## 597 2022-09-01       632.729
## 598 2022-10-01       659.196
## 599 2022-11-01       685.824
## 600 2022-12-01       680.383
```

```r
#Making a matrix with the rows as months and years as columns
RE_matrix <- matrix(RE_matrix_df$`RE Production`, byrow=FALSE, nrow=12)
glimpse(RE_matrix)
```

```
##  num [1:12, 1:50] 220 197 219 209 216 ...
```

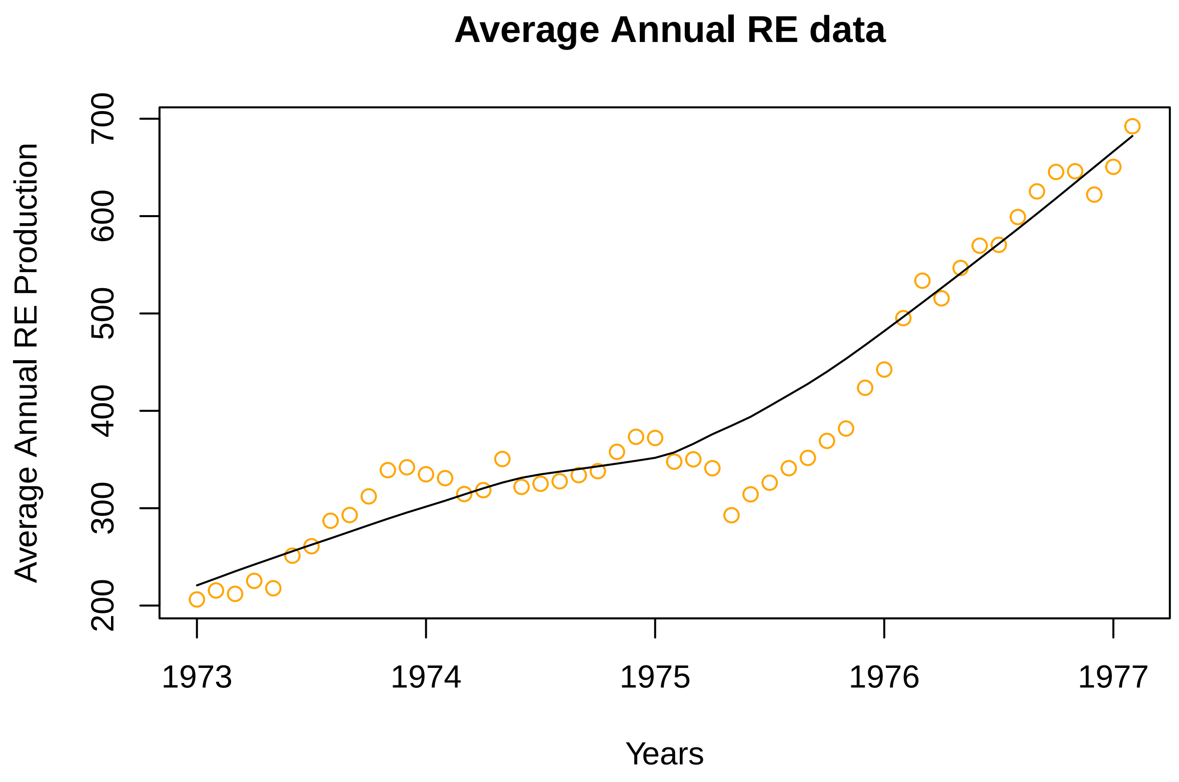
```r
#Renaming the rows and columns
rownames(RE_matrix) <- c("January", "February", "March", "April", "May", "June", "July", "August", "Sep
colnames(RE_matrix)<-c(1973:2022)
head(RE_matrix,12)
```

```
##              1973    1974    1975    1976    1977    1978    1979    1980
## January   219.839 231.010 214.319 236.073 228.907 260.677 270.000 298.221
## February  197.330 210.188 198.008 221.374 194.523 233.933 239.377 271.194
## March     218.686 226.384 224.384 237.807 225.781 258.863 273.485 294.931
## April     209.330 223.218 215.679 224.756 216.602 255.285 265.526 293.043
## May       215.982 227.793 223.695 234.082 221.823 272.691 283.727 310.682
## June      208.249 218.976 217.798 229.595 211.752 254.703 264.118 299.633
## July      207.800 221.909 216.202 235.984 215.097 258.056 262.394 295.537
## August    203.432 214.197 206.312 228.336 214.871 250.652 257.423 281.831
## September 185.300 200.900 194.934 211.665 208.974 241.494 243.468 268.204
## October   193.514 200.312 206.489 218.818 216.727 241.095 253.559 273.058
## November  195.326 200.068 208.436 209.968 222.663 237.214 255.317 270.913
## December  220.755 211.046 217.911 216.239 235.754 250.285 262.637 288.131
##              1981    1982    1983    1984    1985    1986    1987    1988
## January   299.483 320.311 348.969 355.607 353.933 326.552 334.890 334.583
## February  273.604 297.475 320.213 333.238 323.067 307.952 296.606 307.533
## March     293.454 330.131 352.422 358.566 344.083 349.995 327.541 326.015
## April     286.764 316.183 343.331 348.756 334.259 338.487 315.231 316.232
## May       305.297 323.939 355.330 363.212 349.644 345.587 330.797 331.539
## June      305.860 316.816 346.012 344.623 332.457 334.442 311.957 315.603
## July      308.821 321.854 345.359 348.366 332.393 335.334 317.495 317.391
## August    296.678 310.059 338.025 340.669 328.026 325.501 311.395 315.766
## September 276.720 289.054 315.758 317.887 315.367 316.539 302.090 306.500
## October   284.684 296.056 320.524 326.373 327.776 325.125 309.095 310.737
## November  280.364 300.864 325.785 323.172 330.222 323.172 297.439 313.792
## December  304.193 323.054 357.437 343.652 346.947 341.787 319.908 326.992
##              1989    1990    1991    1992    1993    1994    1995    1996
## January   348.321 329.327 370.278 366.577 373.255 388.854 336.872 385.971
## February  317.572 321.465 292.511 305.537 322.185 323.751 299.810 343.243
## March     358.115 353.956 317.683 311.299 359.855 354.509 346.752 385.026
## April     346.511 334.136 293.309 292.073 330.605 332.955 361.046 325.915
## May       350.304 317.791 320.120 282.361 313.546 303.865 333.643 356.221
## June      349.753 289.276 313.437 323.546 304.450 313.708 342.092 375.816
## July      351.720 315.872 309.257 333.005 309.916 366.741 400.977 395.278
## August    358.320 332.580 340.813 347.510 346.577 333.540 399.583 398.870
## September 341.553 311.965 345.122 324.027 324.882 307.933 349.815 347.920
## October   356.682 312.873 324.454 340.565 331.480 343.569 384.663 400.155
## November  359.731 301.883 318.757 345.048 338.485 338.304 366.200 387.043
## December  367.555 341.584 355.690 360.200 352.074 348.732 373.129 378.537
##              1997    1998    1999    2000    2001    2002    2003    2004
## January   397.124 386.269 383.582 319.978 303.197 314.861 318.956 347.154
## February  342.279 323.378 328.183 334.369 272.585 279.136 291.767 321.055
## March     381.623 360.492 334.062 366.040 301.844 302.856 330.201 342.168
## April     374.093 348.763 355.198 364.110 288.028 309.709 327.749 334.068
## May       398.347 374.487 401.370 361.267 290.338 331.378 345.099 344.066
## June      362.325 309.019 353.158 326.724 298.272 326.674 341.209 346.968
## July      382.540 358.537 379.433 351.077 297.654 337.792 342.647 353.034
## August    370.673 354.150 360.215 343.214 304.239 311.593 333.101 344.004
## September 343.197 332.989 328.356 312.937 279.069 302.858 308.470 328.252
## October   402.188 345.379 308.985 341.025 292.015 315.739 313.818 332.739
## November  355.868 309.809 337.650 339.223 283.668 309.716 314.096 332.106
## December  355.807 370.867 332.407 333.069 302.843 328.629 347.074 367.856
##              2005    2006    2007    2008    2009    2010    2011    2012
## January   361.269 388.583 399.004 427.860 431.011 489.844 530.909 539.035
## February  333.479 348.049 343.865 388.671 386.812 449.090 490.715 494.125
## March     354.763 368.883 390.167 424.851 432.104 499.560 553.169 541.244
## April     342.863 367.940 383.102 421.184 431.059 482.552 538.506 519.621
## May       367.186 386.890 398.044 449.522 456.231 507.544 554.011 546.977
## June      362.264 383.011 382.096 444.695 455.356 517.750 553.885 528.771
## July      372.396 381.340 393.450 446.062 455.962 508.593 549.374 520.168
## August    356.107 370.019 386.428 431.761 449.335 497.073 534.036 513.269
## September 331.447 345.317 360.587 398.411 421.927 476.105 500.175 475.610
## October   339.018 353.690 374.075 412.573 450.940 489.125 517.691 491.516
## November  338.541 359.164 373.327 409.976 456.527 500.488 528.710 489.078
## December  360.826 376.761 397.970 428.996 481.882 524.855 552.823 527.556
##              2013    2014    2015    2016    2017    2018    2019    2020
## January   542.692 574.319 581.002 600.004 627.929 653.150 645.530 649.110
## February  487.697 507.326 533.488 582.468 581.036 610.036 593.795 632.884
## March     541.012 589.693 579.817 626.931 664.711 669.313 657.711 642.361
## April     551.448 583.144 569.898 590.484 635.896 657.253 666.642 561.380
## May       578.378 589.777 579.138 609.870 662.077 681.427 690.669 619.030
## June      563.561 590.788 564.673 594.461 643.105 669.472 661.828 637.875
## July      572.289 588.697 584.483 605.124 626.343 648.661 667.695 633.731
## August    542.610 560.101 572.778 592.159 612.943 652.676 648.350 619.356
## September 514.219 530.782 540.124 562.996 584.631 601.408 613.803 584.297
## October   543.689 557.457 557.167 585.197 615.446 628.690 634.265 612.749
## November  548.475 569.677 575.787 586.984 614.560 623.898 621.355 630.734
## December  574.712 593.827 607.572 651.738 635.919 648.213 651.174 641.695
##              2021    2022
## January   637.388 697.568
## February  552.930 651.899
## March     678.059 733.189
## April     651.233 712.467
## May       689.525 742.753
## June      656.848 725.578
## July      651.269 713.262
## August    648.898 672.494
## September 620.767 632.729
## October   650.143 659.196
## November  663.620 685.824
## December  706.622 680.383
```

```
#Calculating the yearly averages
RE_yearly <- colMeans(RE_matrix)
head(RE_yearly)
```

```
##      1973     1974     1975     1976     1977     1978
## 206.2953 215.5001 212.0139 225.3914 217.7895 251.2457
```

```
#Converting the yearly means into a time series and plotting
RE_yearly_ts <- ts(RE_yearly, frequency = 12, start = c(1973, 01))
scatter.smooth(RE_yearly_ts, xlab = "Years", ylab = "Average Annual RE Production", main = " Average Ann
```



Q7

Apply the Mann Kendal, Spearman correlation rank test and ADF. Are the results from the test in agreement with the test results for the monthly series, i.e., results for Q6?

```
#ADF Test
print(adf.test(RE_yearly), alternative="stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  RE_yearly
## Dickey-Fuller = -1.0881, Lag order = 3, p-value = 0.9156
## alternative hypothesis: stationary
```

```
#Mann Kendall Test
print(summary(MannKendall(RE_yearly_ts)))
```

```
## Score =  983 , Var(Score) = 14291.67
## denominator =  1225
## tau = 0.802, 2-sided pvalue =< 2.22e-16
## NULL
```

```
#Spearman correlation rank test
RE_year <- c(1973:2022)
cor(RE_yearly, RE_year, method="spearman")
```

```
## [1] 0.9110684
```

Analysis: The ADF test on the annual average data fails as well as the p-value > 0.05. On the other hand the Mann Kendall test is successful as the p-value < 0.05, and so there is deterministic trend in the data. This is confirmed by the Spearman correlation test that finds a very high 0.91 correlation between the annual average RE production values and time (years).