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

Assignment 5 - Due date 02/13/24

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_A05_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: "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.\

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

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

```
library(tseries)
library(ggplot2)
library(Kendall)
library(lubridate)
```

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

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

```
library(tidyverse)  #load this package so yon clean the data frame using pipes
```

```
## -- 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 errors
```

```r
library(openxlsx)
library(cowplot)
```

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

Decomposing Time Series

Consider the same data you used for A04 from the spreadsheet "Table_10.1_Renewable_Energy_Production_and_Consump
The data comes from the US Energy Information and Administration and corresponds to the December 2023 Monthly Energy Review.

```r
#Importing data set - using xlsx package
#energy_data <- read.xlsx(file="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source

#Now let's extract the column names from row 11 only
#read_col_names <- read.xlsx(file="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Sou

#colnames(energy_data) <- read_col_names

#FOR TA: read.xlsx wasn't working so I have used read.csv which is giving me the same

energy_data <- read.csv(file = "./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source

head(energy_data)
```

```
##           Month Wood.Energy.Production Biofuels.Production
## 1  1973 January                129.630       Not Available
## 2 1973 February                117.194       Not Available
## 3    1973 March                129.763       Not Available
## 4    1973 April                125.462       Not Available
## 5      1973 May                129.624       Not Available
## 6     1973 June                125.435       Not Available
##   Total.Biomass.Energy.Production Total.Renewable.Energy.Production
## 1                         129.787                           219.839
## 2                         117.338                           197.330
## 3                         129.938                           218.686
```

```
## 4                        125.636                           209.330
## 5                        129.834                           215.982
## 6                        125.611                           208.249
##   Hydroelectric.Power.Consumption Geothermal.Energy.Consumption
## 1                          89.562                         0.490
## 2                          79.544                         0.448
## 3                          88.284                         0.464
## 4                          83.152                         0.542
## 5                          85.643                         0.505
## 6                          82.060                         0.579
##   Solar.Energy.Consumption Wind.Energy.Consumption Wood.Energy.Consumption
## 1            Not Available           Not Available                 129.630
## 2            Not Available           Not Available                 117.194
## 3            Not Available           Not Available                 129.763
## 4            Not Available           Not Available                 125.462
## 5            Not Available           Not Available                 129.624
## 6            Not Available           Not Available                 125.435
##   Waste.Energy.Consumption Biofuels.Consumption
## 1                    0.157        Not Available
## 2                    0.144        Not Available
## 3                    0.176        Not Available
## 4                    0.174        Not Available
## 5                    0.210        Not Available
## 6                    0.176        Not Available
##   Total.Biomass.Energy.Consumption Total.Renewable.Energy.Consumption
## 1                          129.787                            219.839
## 2                          117.338                            197.330
## 3                          129.938                            218.686
## 4                          125.636                            209.330
## 5                          129.834                            215.982
## 6                          125.611                            208.249
```

```r
nobs=nrow(energy_data)
nvar=ncol(energy_data)
```

Q1

For this assignment you will work only with the following columns: Solar Energy Consumption and Wind Energy Consumption. Create a data frame structure with these two time series only and the Date column. Drop the rows with *Not Available* and convert the columns to numeric. You can use filtering to eliminate the initial rows or convert to numeric and then use the drop_na() function. If you are familiar with pipes for data wrangling, try using it!

```r
#Creating dataframe with the three specified columns and renaming columns
energy_data_solar_wind <- energy_data[,c(1,8,9)]
head(energy_data_solar_wind)
```

```
##           Month Solar.Energy.Consumption Wind.Energy.Consumption
## 1  1973 January            Not Available           Not Available
## 2 1973 February            Not Available           Not Available
## 3    1973 March            Not Available           Not Available
## 4    1973 April            Not Available           Not Available
```

```
## 5      1973 May            Not Available           Not Available
## 6     1973 June            Not Available           Not Available
```

```
colnames(energy_data_solar_wind) <- c("Date", "Solar", "Wind")

#Converting columns to numeric data and the date column to a date object
energy_data_solar_wind$Date <- ym(energy_data_solar_wind$Date)
energy_data_solar_wind$Wind <- as.numeric(as.character(energy_data_solar_wind$Wind))
```

```
## Warning: NAs introduced by coercion
```

```
energy_data_solar_wind$Solar <- as.numeric(as.character(energy_data_solar_wind$Solar))
```

```
## Warning: NAs introduced by coercion
```

```
#Checking dataset
glimpse(energy_data_solar_wind)
```

```
## Rows: 609
## Columns: 3
## $ Date  <date> 1973-01-01, 1973-02-01, 1973-03-01, 1973-04-01, 1973-05-01, 197~
## $ Solar <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Wind  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

```
#Removing the first 132 rows (NAs)
energy_data_solar_wind_dropna <- drop_na(energy_data_solar_wind)
head(energy_data_solar_wind_dropna)
```

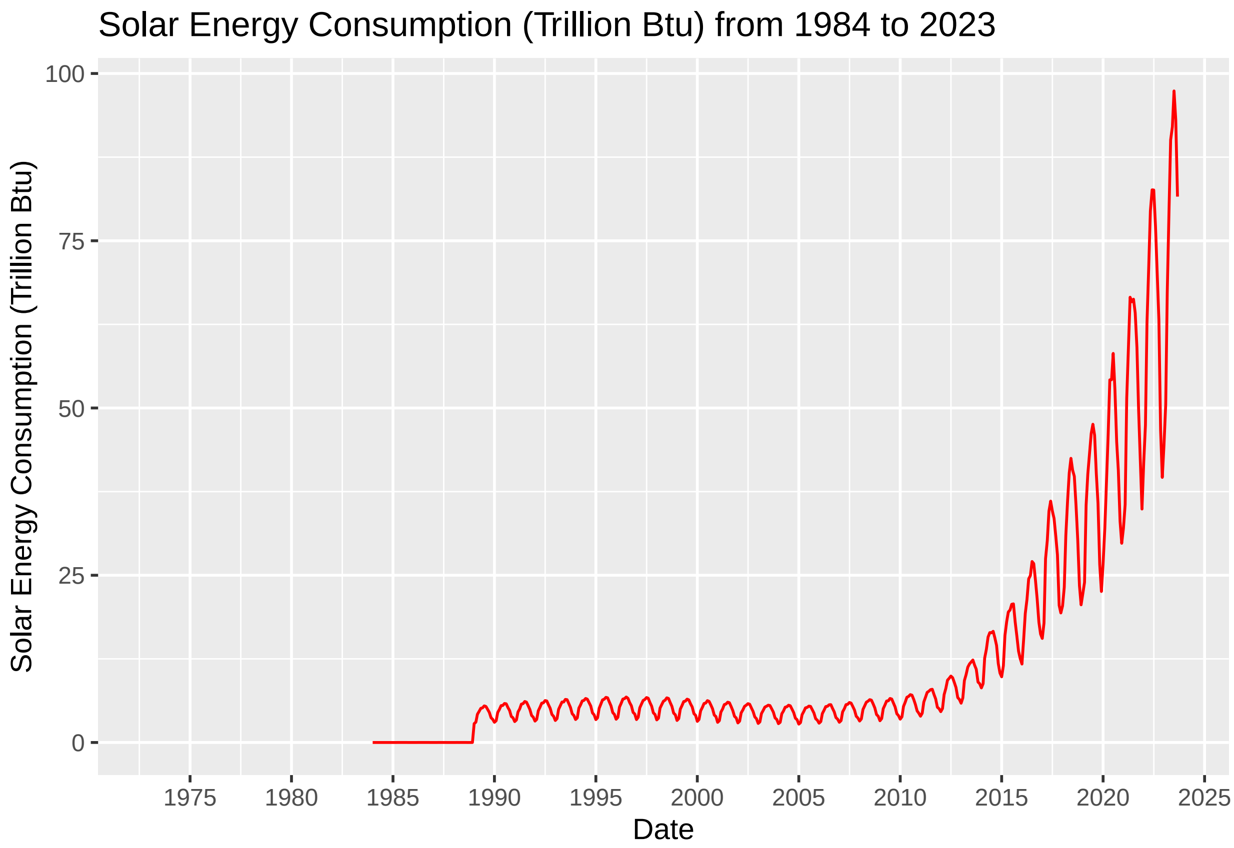
```
##         Date Solar  Wind
## 1 1984-01-01 0.000 0.000
## 2 1984-02-01 0.000 0.001
## 3 1984-03-01 0.001 0.001
## 4 1984-04-01 0.001 0.002
## 5 1984-05-01 0.002 0.003
## 6 1984-06-01 0.003 0.002
```

Q2

Plot the Solar and Wind energy consumption over time using ggplot. Plot each series on a separate graph. No need to add legend. Add informative names to the y axis using `ylab()`. Explore the function `scale_x_date()` on ggplot and see if you can change the x axis to improve your plot. Hint: use `scale_x_date(date_breaks = "5 years", date_labels = "%Y")`

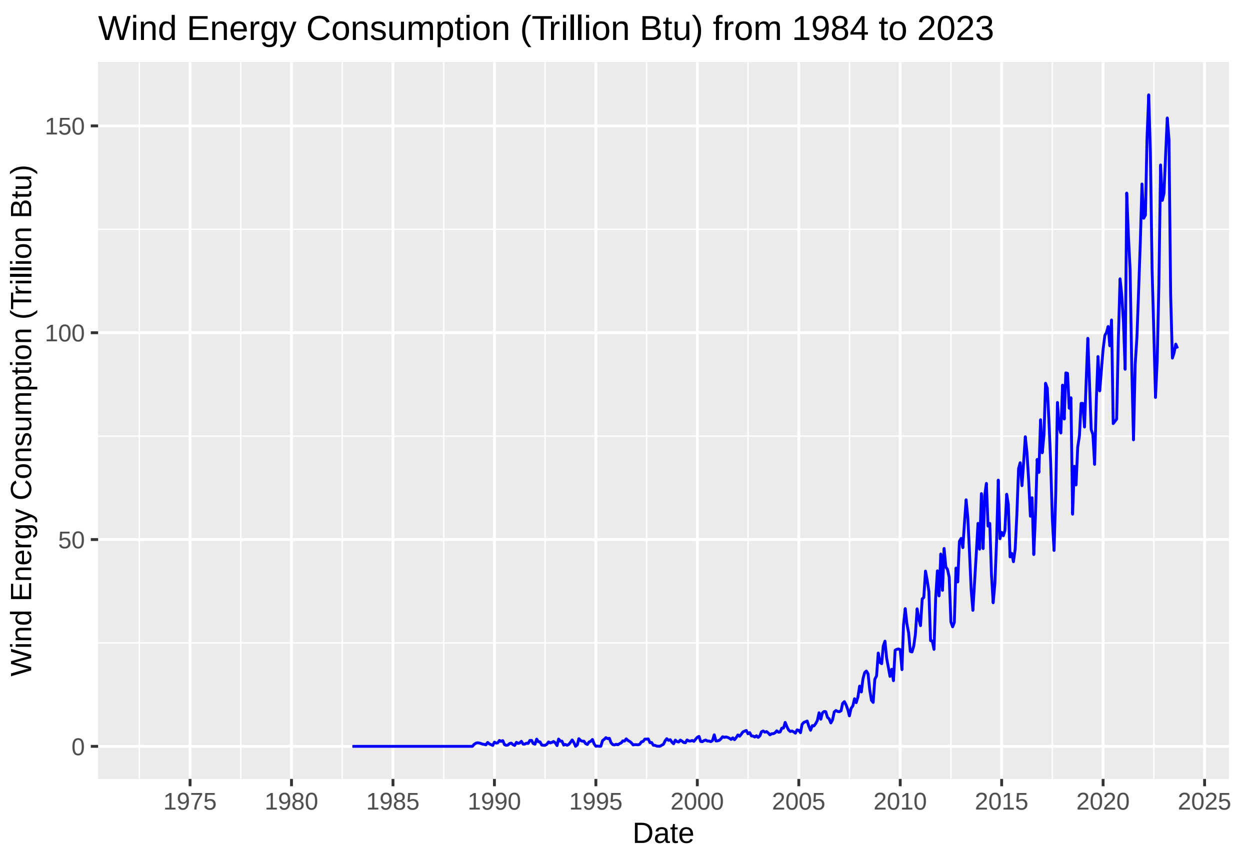
```
#Plotting Solar Energy Consumption
ggplot(energy_data_solar_wind)+
  geom_line(aes(x=Date, y=Solar), color="red")+
  ylab("Solar Energy Consumption (Trillion Btu)")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  ggtitle ("Solar Energy Consumption (Trillion Btu) from 1984 to 2023")
```

```
## Warning: Removed 132 rows containing missing values (`geom_line()`).
```




```
ggplot(energy_data_solar_wind)+
  geom_line(aes(x=Date, y=Wind), color="blue")+
  ylab("Wind Energy Consumption (Trillion Btu)")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  ggtitle ("Wind Energy Consumption (Trillion Btu) from 1984 to 2023")
```

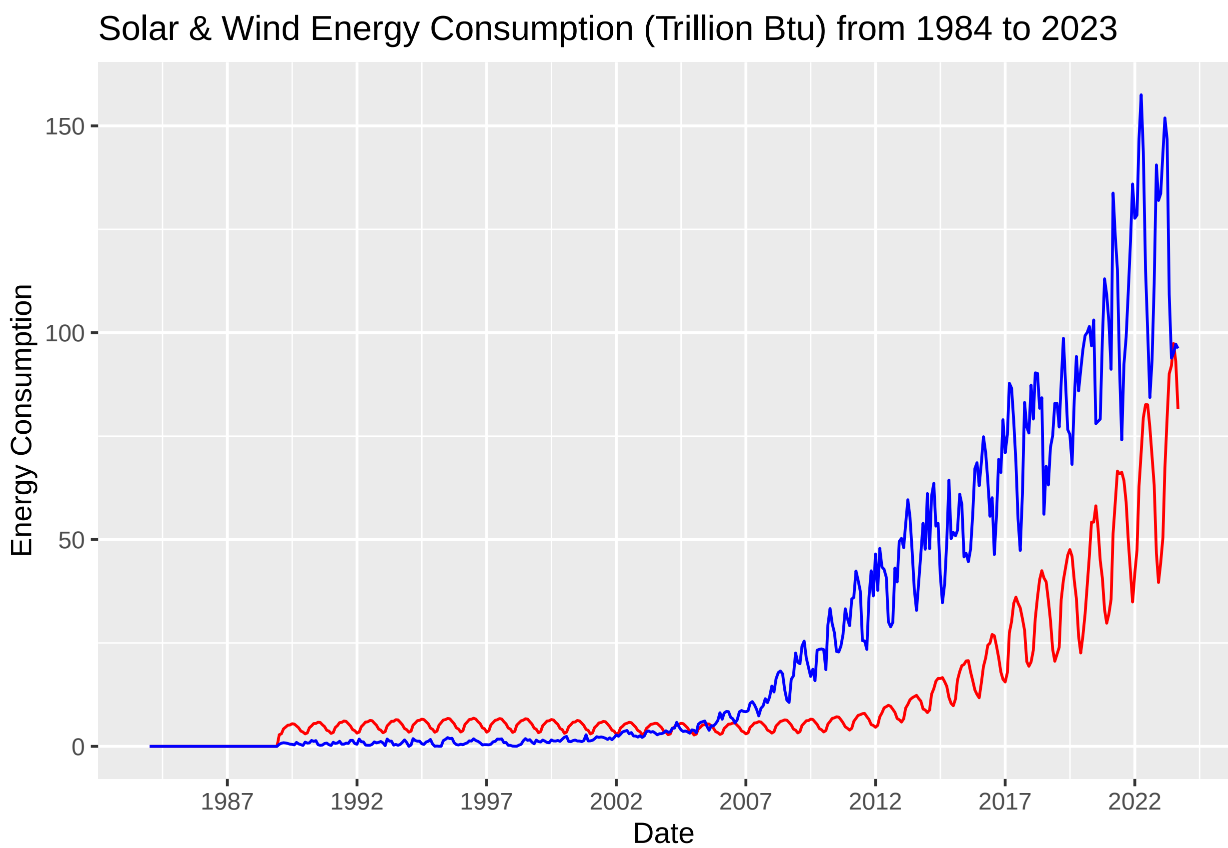
```
## Warning: Removed 120 rows containing missing values (`geom_line()`).
```



Q3

Now plot both series in the same graph, also using ggplot(). Use function `scale_color_manual()` to manually add a legend to ggplot. Make the solar energy consumption red and wind energy consumption blue. Add informative name to the y axis using `ylab("Energy Consumption)`. And use function `scale_x_date()` to set x axis breaks every 5 years.

```
ggplot(energy_data_solar_wind_dropna)+
  geom_line(aes(x=Date, y=Solar), color="red", )+
   geom_line(aes(x=Date, y=Wind), color="blue")+
  scale_color_manual(labels = c("Solar", "Wind"))+
  ylab("Energy Consumption")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  ggtitle ("Solar & Wind Energy Consumption (Trillion Btu) from 1984 to 2023")
```



Decomposing the time series

The stats package has a function called decompose(). This function only take time series object. As the name says the decompose function will decompose your time series into three components: trend, seasonal and random. This is similar to what we did in the previous script, but in a more automated way. The random component is the time series without seasonal and trend component.

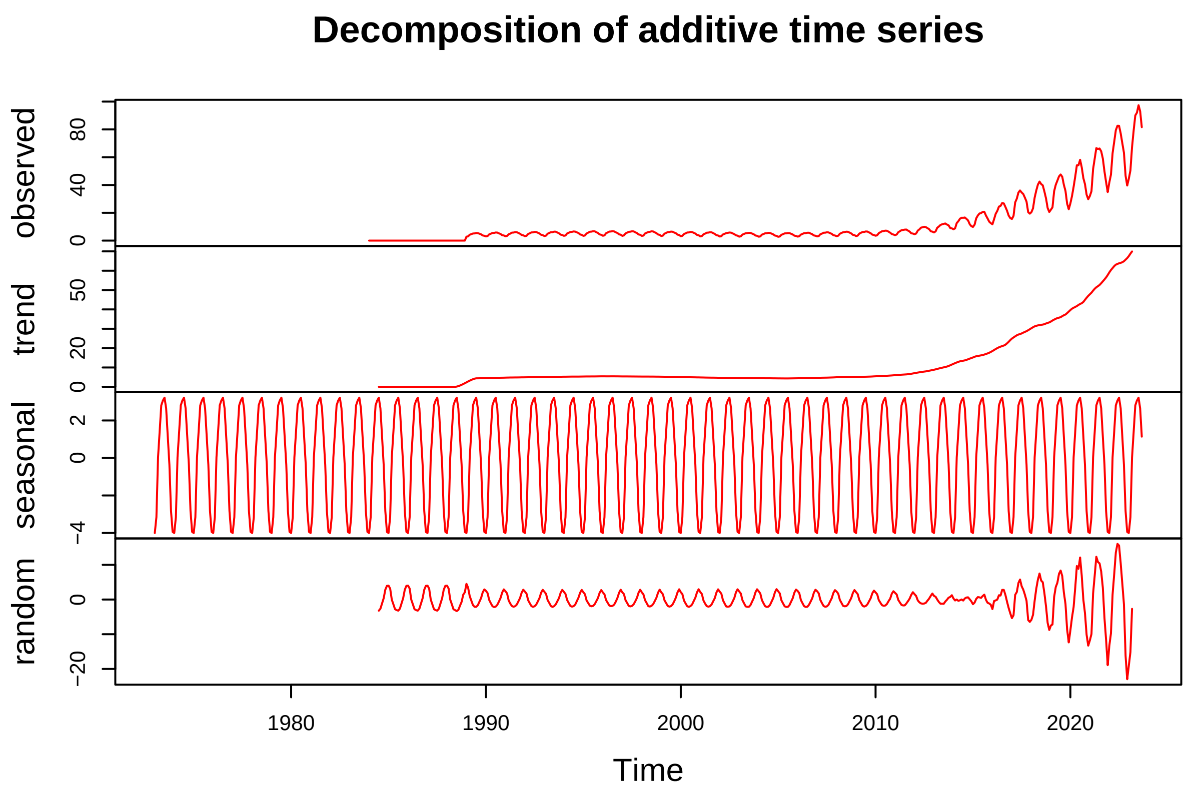
Additional info on `decompose()`.

1) You have two options: alternative and multiplicative. Multiplicative models exhibit a change in frequency over time.
2) The trend is not a straight line because it uses a moving average method to detect trend.
3) The seasonal component of the time series is found by subtracting the trend component from the original data then grouping the results by month and averaging them.
4) The random component, also referred to as the noise component, is composed of all the leftover signal which is not explained by the combination of the trend and seasonal component.

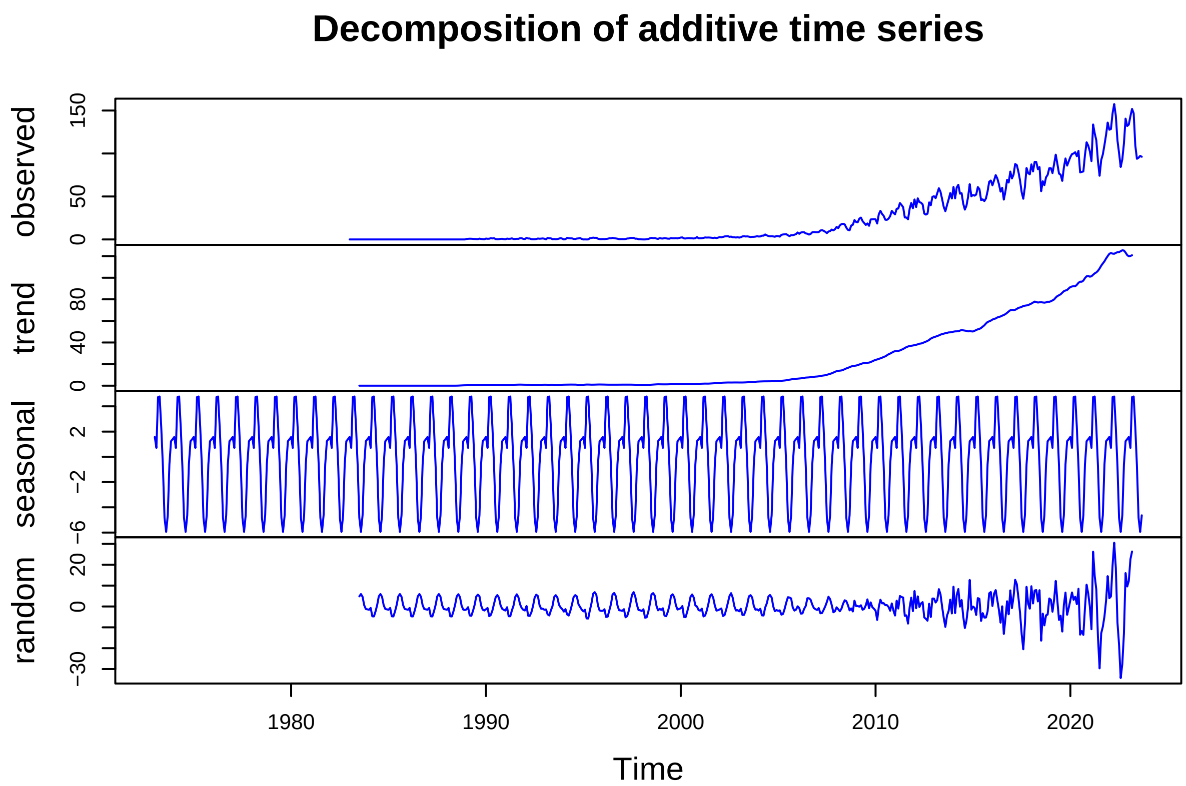
Q4

Transform wind and solar series into a time series object and apply the decompose function on them using the additive option, i.e., `decompose(ts_data, type = "additive")`. What can you say about the trend component? What about the random component? Does the random component look random? Or does it appear to still have some seasonality on it?

```r
#Transforming wind and solar into time series objects
Solar_ts <- ts(energy_data_solar_wind$Solar, frequency = 12, start = c(1973, 01))
Wind_ts <- ts(energy_data_solar_wind$Wind, frequency = 12, start = c(1973, 01))

#Decomposing the two time series using the additive method
Solar_decompose <- decompose(Solar_ts, type = "additive")
Wind_decompose <- decompose(Wind_ts, type = "additive")

#Plotting both the decomposed datasets
plot(Solar_decompose, col="red")
```



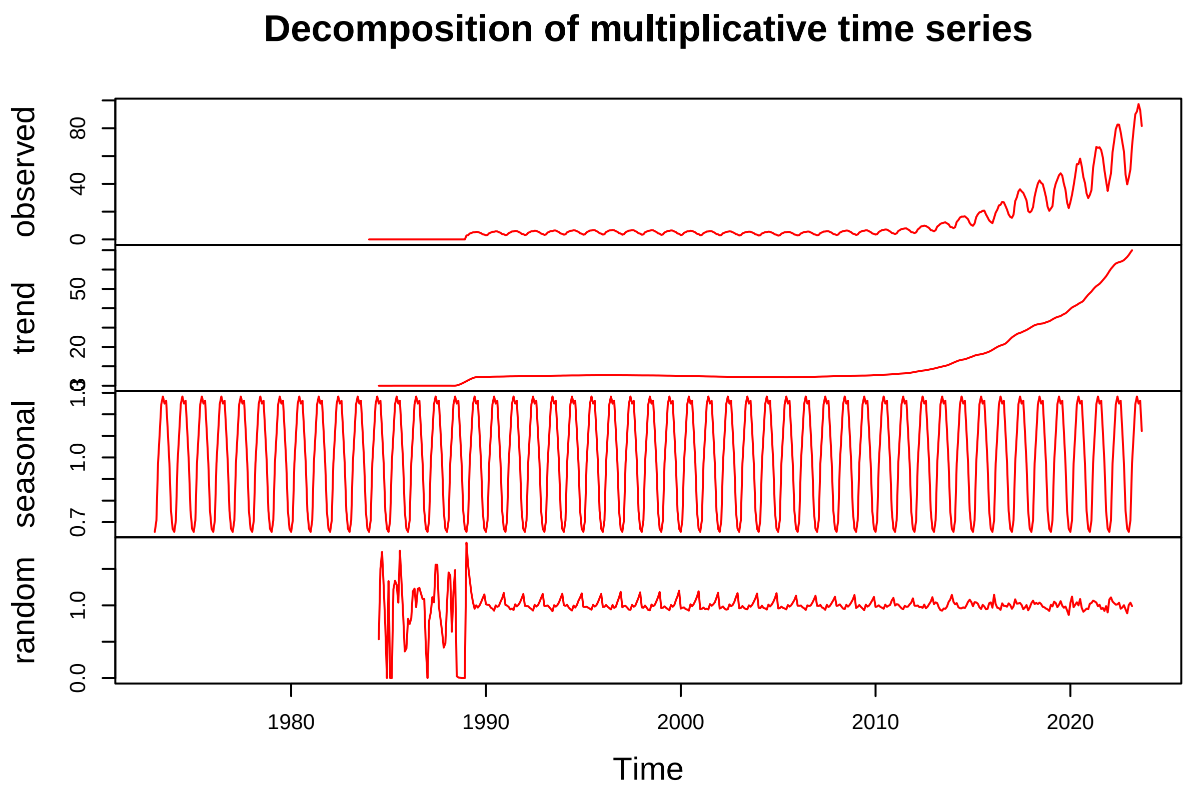

```r
plot(Wind_decompose, col="blue")
```



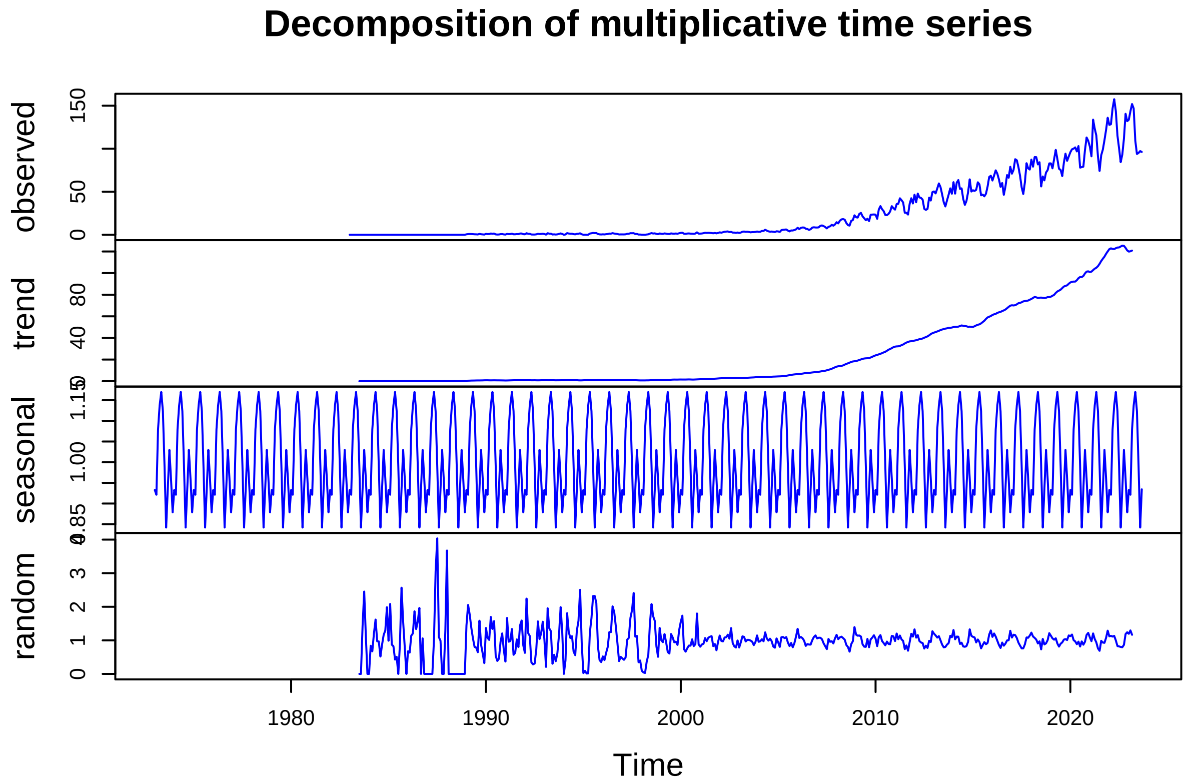
Analysis: Solar's random component is almost fully random except for the last few years post 2015 where the height of the spikes suddenly start increasing. In wind's random component as well there seems to be some seasonality - one, in the particular shape of the waves from 185 to 2010, and then the increasing height of the spikes thereafter. ### Q5

Use the decompose function again but now change the type of the seasonal component from additive to multiplicative. What happened to the random component this time?

```
#Decomposing the two time series using the multiplicative method
Solar_decompose <- decompose(Solar_ts, type = "multiplicative")
Wind_decompose <- decompose(Wind_ts, type = "multiplicative")

#Plotting both the decomposed datasets
plot(Solar_decompose, col="red")
```




```
plot(Wind_decompose, col="blue")
```



Analysis: the heightened spikes of both shift from the end of the dataset to the beginning and still show a slight reptitive trend in the shape of their cycles - post 1990 for solar and post 2000 for wind. However, it does seem to be largely detrended, compared to the original dataset.

Q6

When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: In general, the usefulness of the historical data depends on the nature of the data and what it represents. In this case, the historical data from the 90s and 2000s is not really relevant anymore as the values then were close to 0 whereas now, both solar and wind are growing rapidly, as seen in both their trends post 2010.

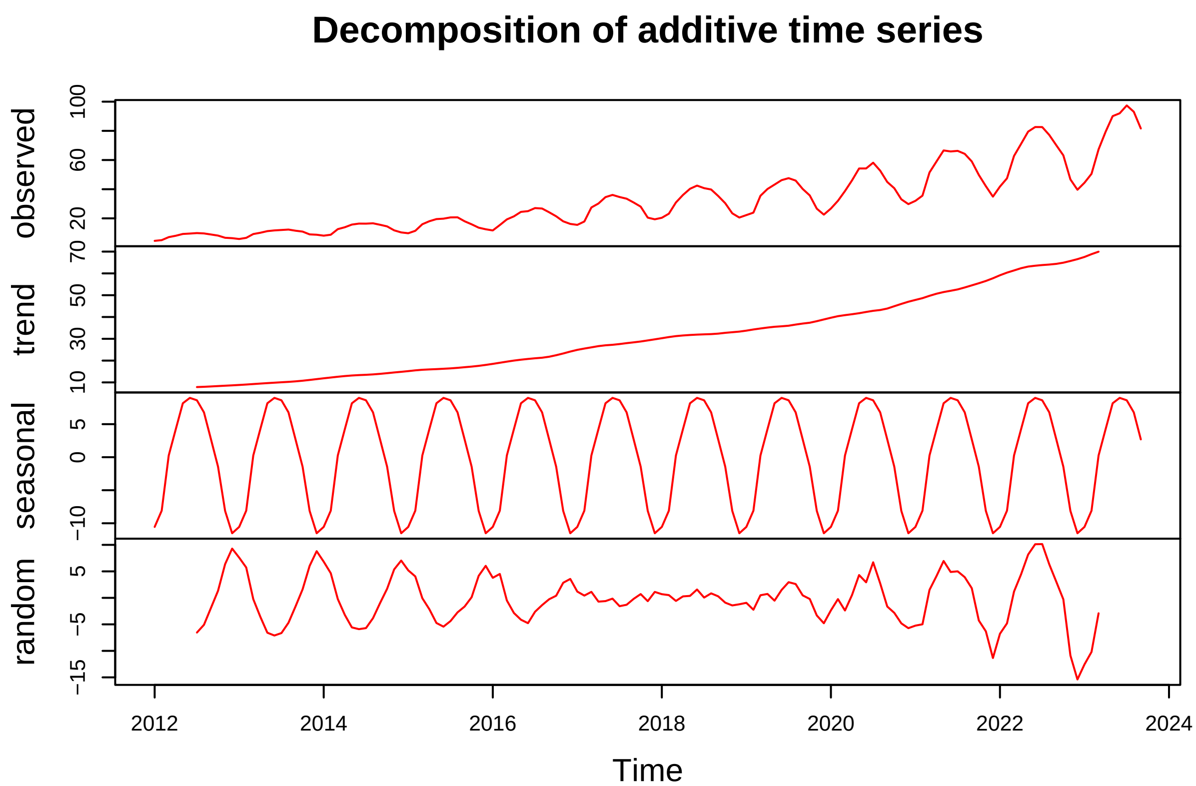
Q7

Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the decompose function `type=additive` to this new time series. Comment the results. Does the random component look random? Think about our discussion in class about seasonal components that depends on the level of the series.

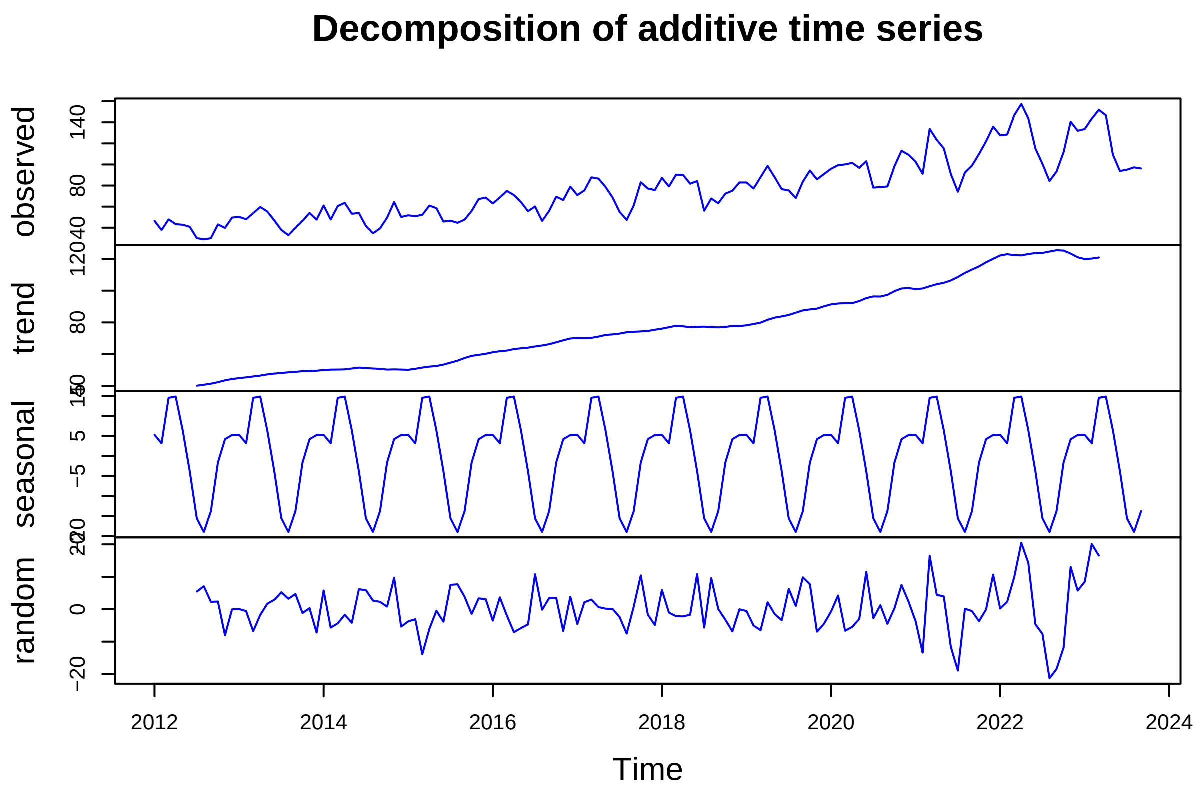
```
#Filtering data to post 2012
energy_data_solar_wind_dropna_2012 <- filter(energy_data_solar_wind_dropna, year(Date) >= 2012)
head(energy_data_solar_wind_dropna_2012)
```

```
##         Date Solar   Wind
## 1 2012-01-01 4.607 46.514
## 2 2012-02-01 5.077 37.709
## 3 2012-03-01 7.148 47.858
## 4 2012-04-01 8.096 43.364
## 5 2012-05-01 9.316 42.788
## 6 2012-06-01 9.605 40.850
```

```r
#Converting this new df into two time series
Solar_2012_ts <- ts(energy_data_solar_wind_dropna_2012$Solar, frequency = 12, start = c(2012, 01))
Wind_2012_ts <- ts(energy_data_solar_wind_dropna_2012$Wind, frequency = 12, start = c(2012, 01))

#Decomposing the two time series using the additive method
Solar_2012_decompose <- decompose(Solar_2012_ts, type = "additive")
Wind_2012_decompose <- decompose(Wind_2012_ts, type = "additive")

#Plotting both the decomposed datasets
plot(Solar_2012_decompose, col="red")
```




```r
plot(Wind_2012_decompose, col="blue")
```



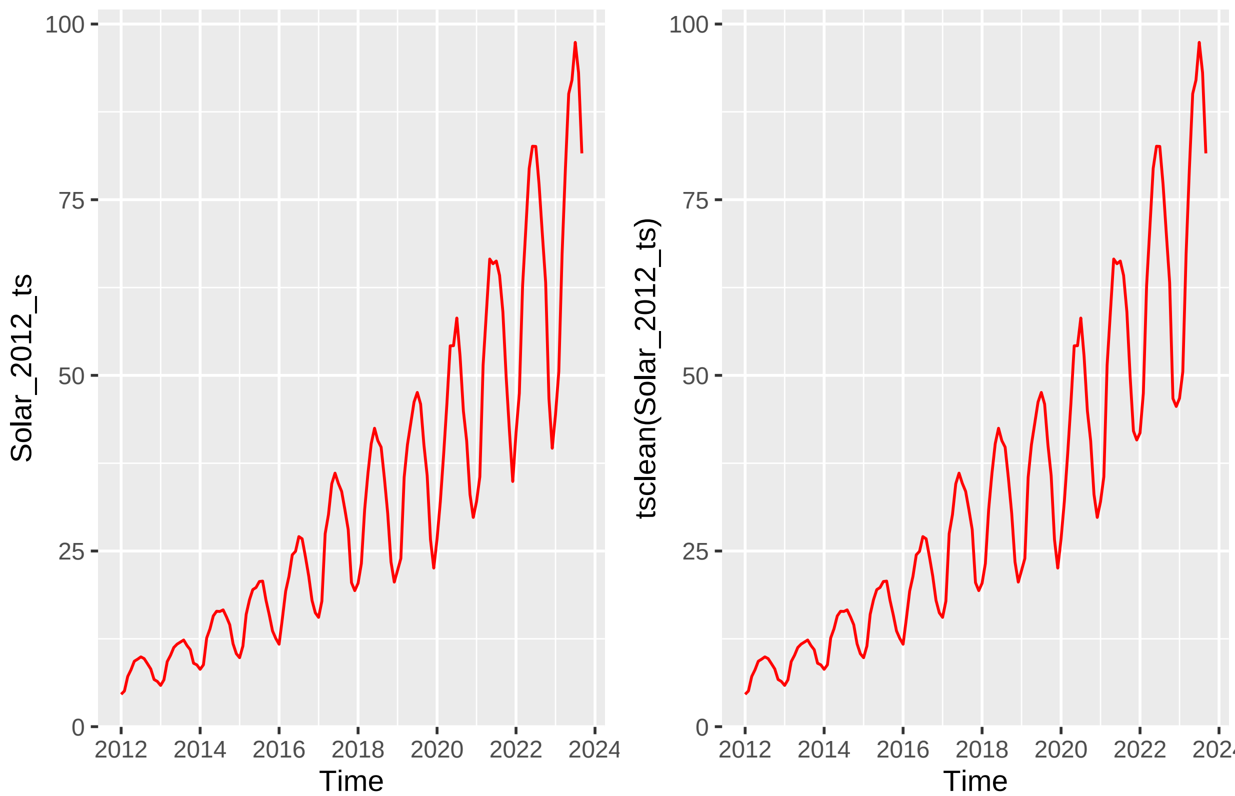
Answer: The random component looks a lot more random now!

Identify and Remove outliers

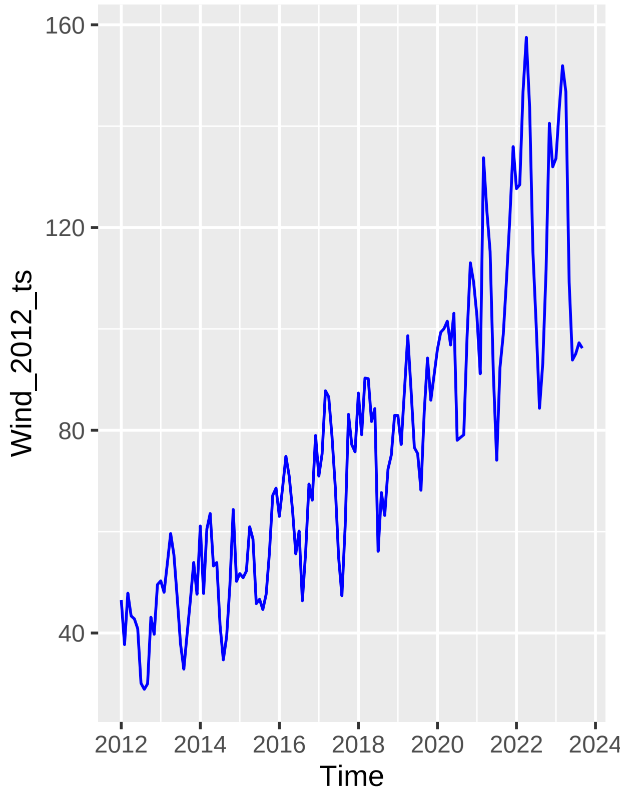
Q8

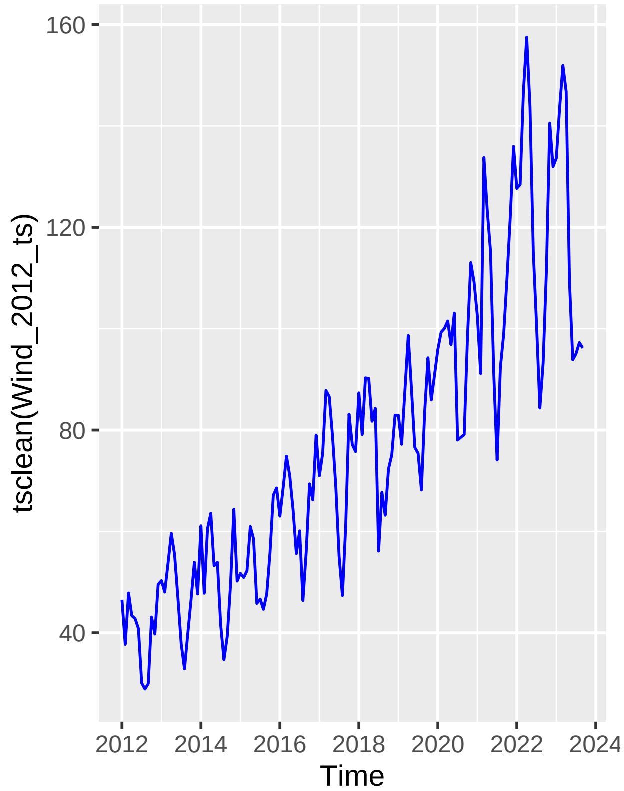
Apply the `tsclean()` to both series from Q7. Did the function removed any outliers from the series? Hint: Use `autoplot()` to check if there is difference between cleaned series and original series.

```
#plotting the clean data
Solar_2012_ts_clean <- autoplot(tsclean(Solar_2012_ts), color = "red")
Wind_2012_ts_clean <- autoplot(tsclean(Wind_2012_ts), color = "blue")

#plotting the original data
Solar_2012_plot <- autoplot(Solar_2012_ts, color = "red")
Wind_2012_plot <- autoplot(Wind_2012_ts, color = "blue")

#plotting the original and cleaned plots together
plot_grid(Solar_2012_plot, Solar_2012_ts_clean)
```




```
plot_grid(Wind_2012_plot, Wind_2012_ts_clean)
```





There is no difference between the original plots and the cleaned plot

Q9

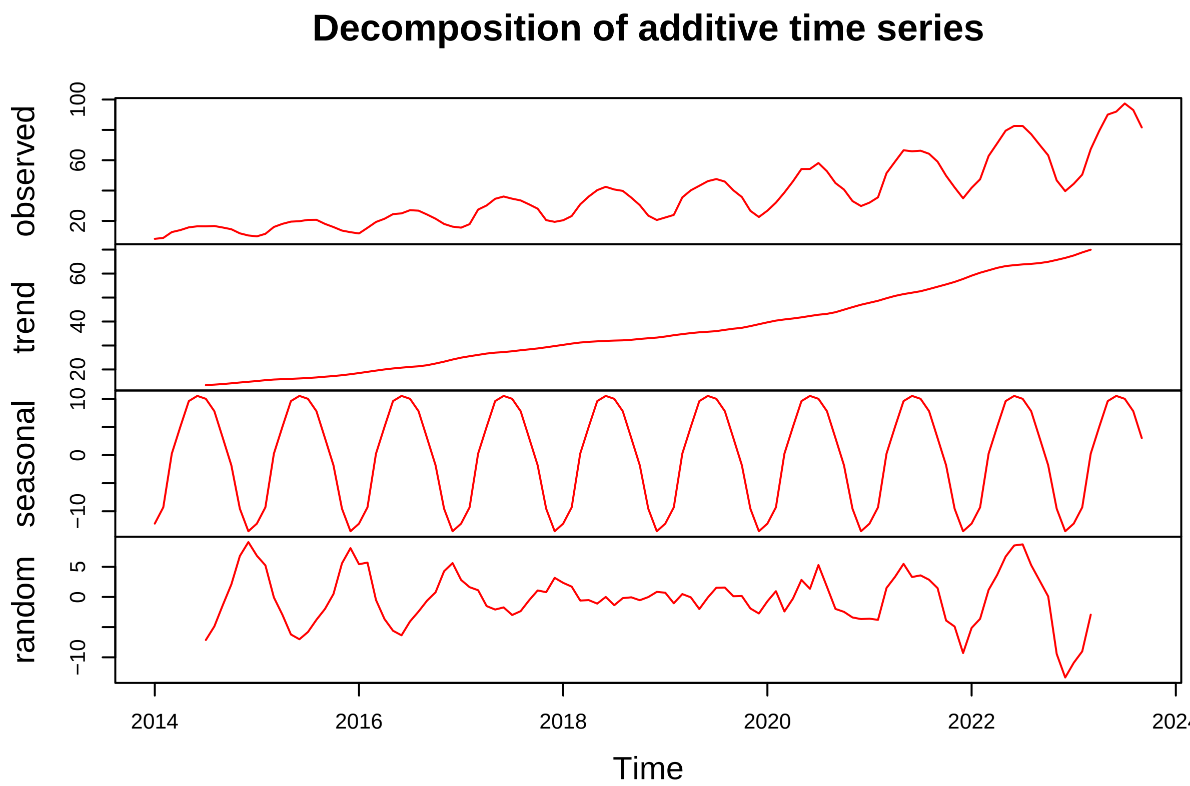
Redo number Q8 but now with the time series you created on Q7, i.e., the series starting in 2014. Using what `autoplot()` again what happened now?Did the function removed any outliers from the series?

```
#Filtering data to post 2014
energy_data_solar_wind_dropna_2014 <- filter(energy_data_solar_wind_dropna, year(Date) >= 2014)
head(energy_data_solar_wind_dropna_2014)
```

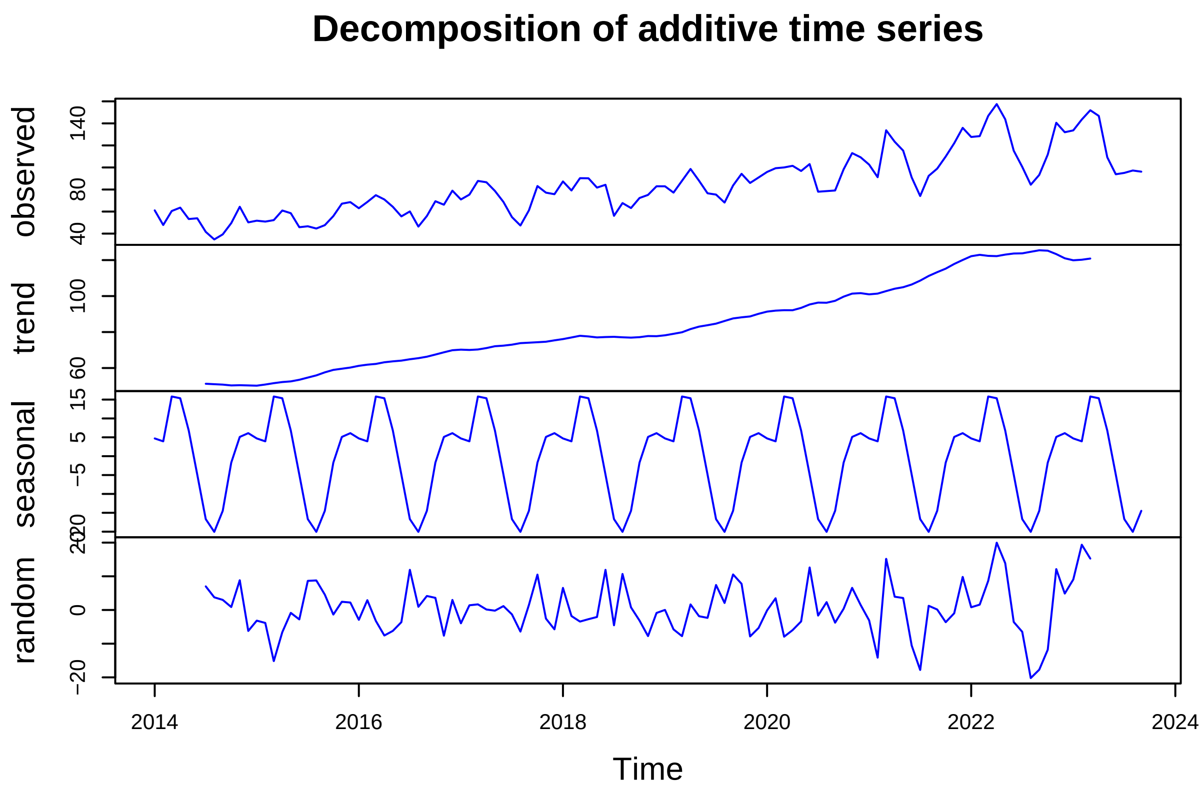
```
##         Date  Solar   Wind
## 1 2014-01-01  8.157 61.113
## 2 2014-02-01  8.799 47.798
## 3 2014-03-01 12.624 60.515
## 4 2014-04-01 13.934 63.584
## 5 2014-05-01 15.758 53.232
## 6 2014-06-01 16.428 53.906
```

```
#Converting this new df into two time series
Solar_2014_ts <- ts(energy_data_solar_wind_dropna_2014$Solar, frequency = 12, start = c(2014, 01))
Wind_2014_ts <- ts(energy_data_solar_wind_dropna_2014$Wind, frequency = 12, start = c(2014, 01))

#Decomposing the two time series using the additive method
Solar_2014_decompose <- decompose(Solar_2014_ts, type = "additive")
```

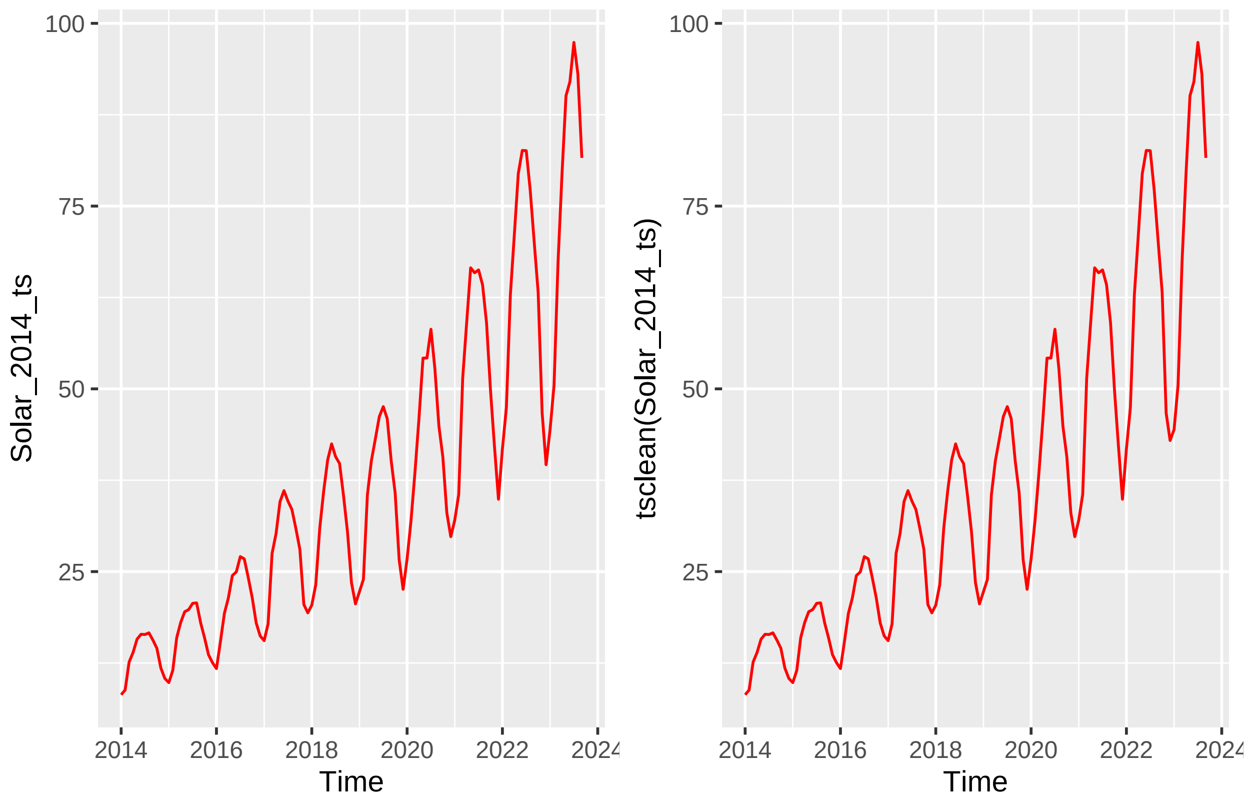
```r
Wind_2014_decompose <- decompose(Wind_2014_ts, type = "additive")

#Plotting both the decomposed datasets
plot(Solar_2014_decompose, col="red")
```



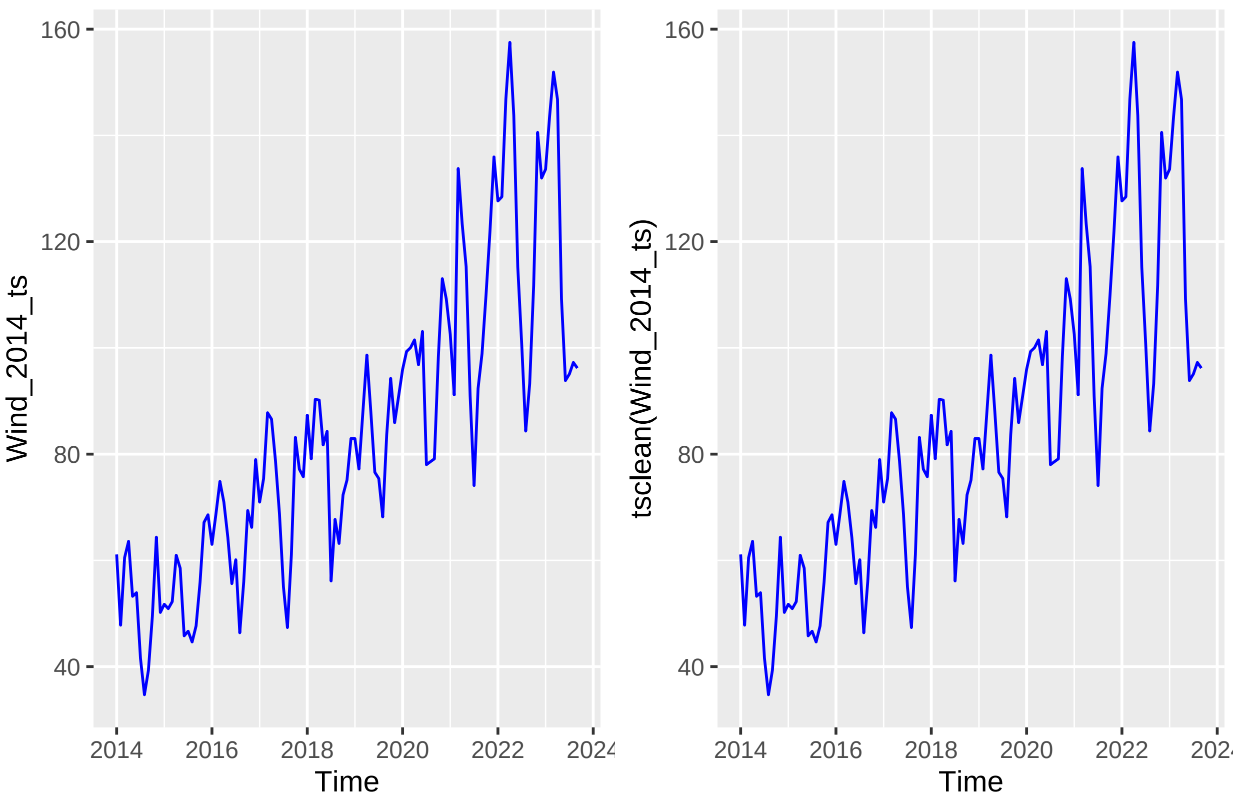

```r
plot(Wind_2014_decompose, col="blue")
```




```r
#plotting the clean data
Solar_2014_ts_clean <- autoplot(tsclean(Solar_2014_ts), color = "red")
Wind_2014_ts_clean <- autoplot(tsclean(Wind_2014_ts), color = "blue")

#plotting the original data
Solar_2014_plot <- autoplot(Solar_2014_ts, color = "red")
Wind_2014_plot <- autoplot(Wind_2014_ts, color = "blue")

#plotting the original and cleaned plots together
plot_grid(Solar_2014_plot, Solar_2014_ts_clean)
```




```
plot_grid(Wind_2014_plot, Wind_2014_ts_clean)
```



Answer: No outliers seem to have been removed, as there were none present post 2010 in the dataset. the original and cleaned datasets for both solar nad wind are thus the same, when begun in 2012 and 2014.