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

Assignment 5 - Due date 02/13/24

Ina Liao

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.\

```
library(forecast)
library(tseries)
library(ggplot2)
library(Kendall)
library(lubridate)
library(tidyverse)
library(here)
library(patchwork)
library(tidyr)
library(knitr)
library(kableExtra)
library(ggthemes)
library(cowplot)
library(dplyr)
```

Decomposing Time Series

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

```
here()
```

```
## [1] "/Users/inaliao/Desktop/TSA_Spring24"
```

```
raw_energy<-read.csv(here("Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.csv"),s
```

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!

```
#select the needed columns
df_energy<-raw_energy[,c(1,8:9)]
df_energy$Month<-ym(df_energy$Month)

#rename the column names
new_colnames<-c("Date","Solar Energy Consumption","Wind Energy Consumption")
colnames(df_energy)<-new_colnames

#select the columns that do not have na values
df_energy<-df_energy %>%
  mutate("Solar Energy Consumption"= as.numeric(`Solar Energy Consumption`),
         "Wind Energy Consumption" = as.numeric(`Wind Energy Consumption`)) %>%
  select_if(~all(!is.na(.)))
head(df_energy)
```

```
##         Date Solar Energy Consumption Wind Energy Consumption
## 1 1973-01-01                      412                     411
## 2 1973-02-01                      412                     411
## 3 1973-03-01                      412                     411
## 4 1973-04-01                      412                     411
## 5 1973-05-01                      412                     411
## 6 1973-06-01                      412                     411
```

```
#create time series object
year1<-year(df_energy$Date[1])
month1<-month(df_energy$Date[1])
ts_energy<-ts(df_energy[,2:3],start=c(year1,month1),frequency=12)

#combine time series object with date as a dataframe
df_ts_energy<-data.frame("Date"=df_energy$Date,
                         "Solar Energy Consumption"=ts_energy[,1],
                         "Wind Energy Consumption"=ts_energy[,2])
#correct the column names
colnames(df_ts_energy)<-new_colnames
```

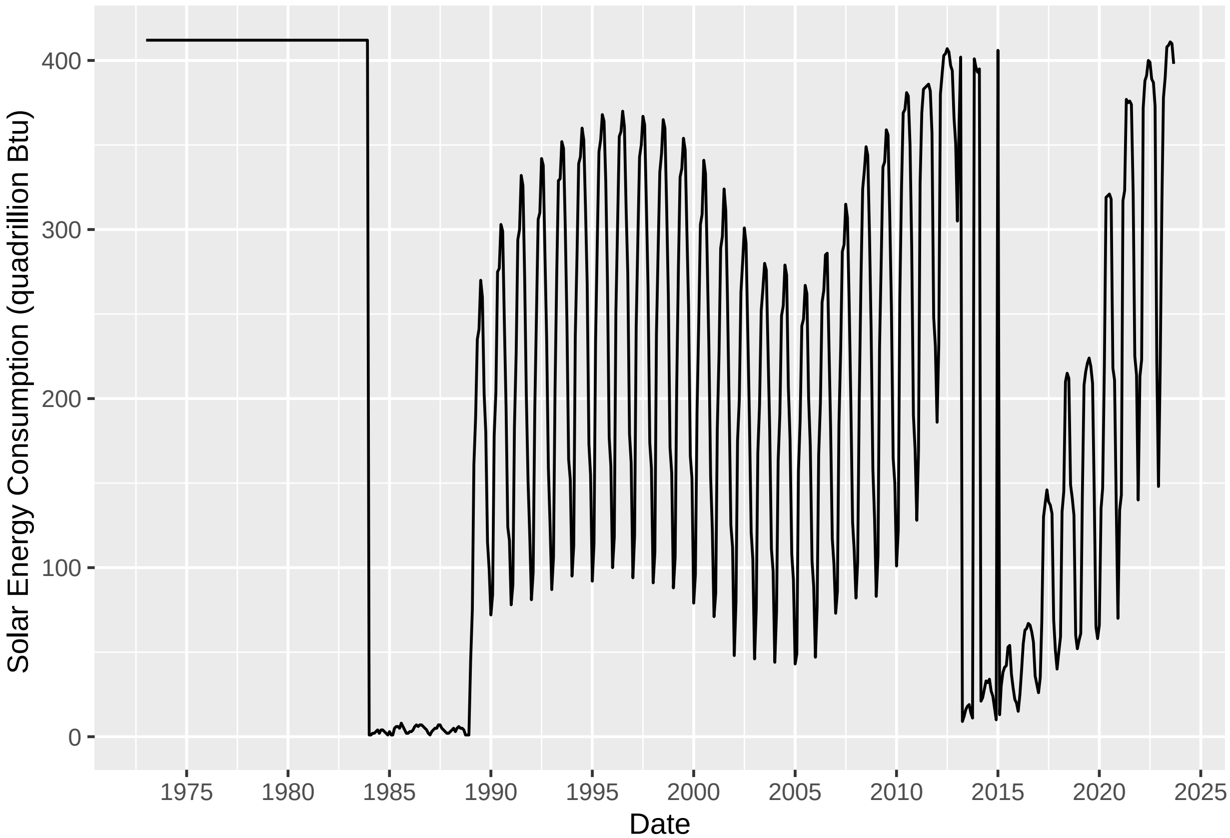
Q2

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")")`

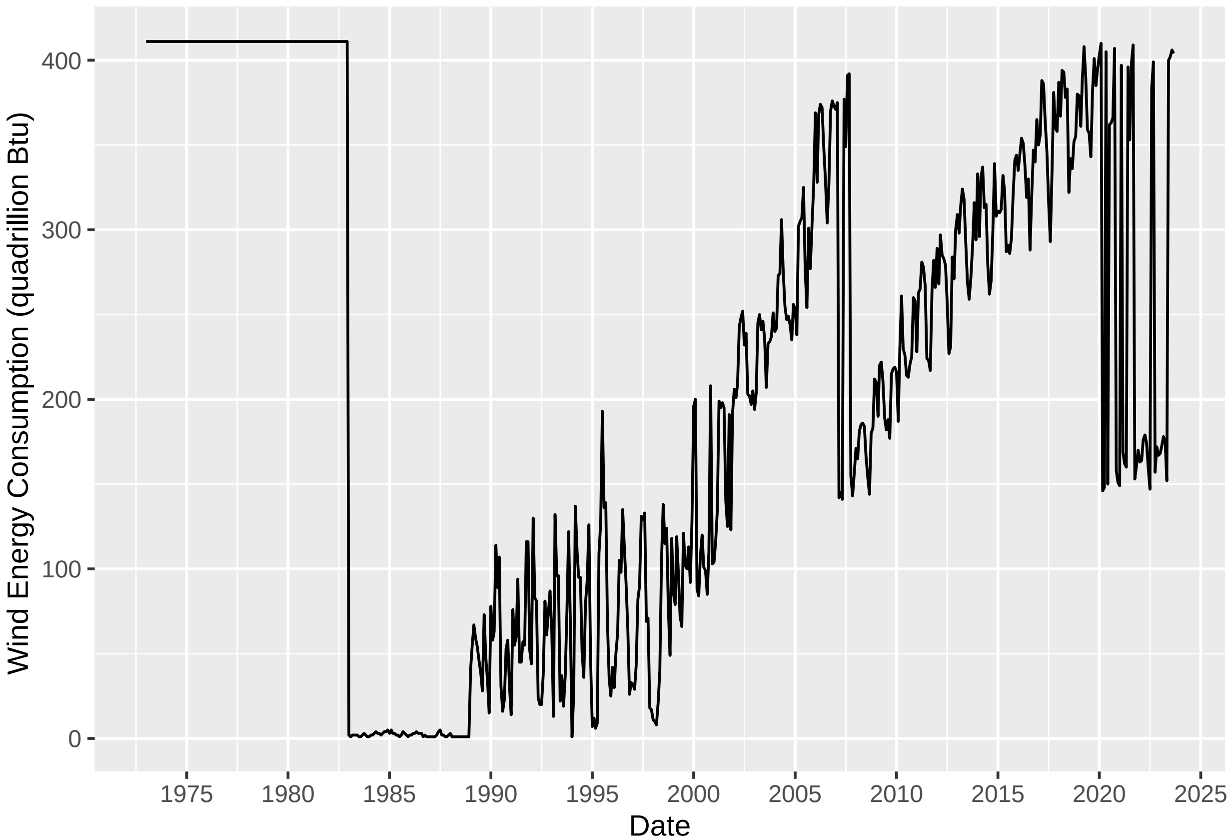
```
for (i in 2:3){
  print(ggplot(df_ts_energy)+
          geom_line(aes(x=Date,y=df_ts_energy[,i]))+
```

```
          ylab(paste0(colnames(df_ts_energy)[(i)]," (quadrillion Btu)",sep=" "))+
          scale_x_date(date_breaks = "5 years", date_labels = "%Y")
          )
}
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```




```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

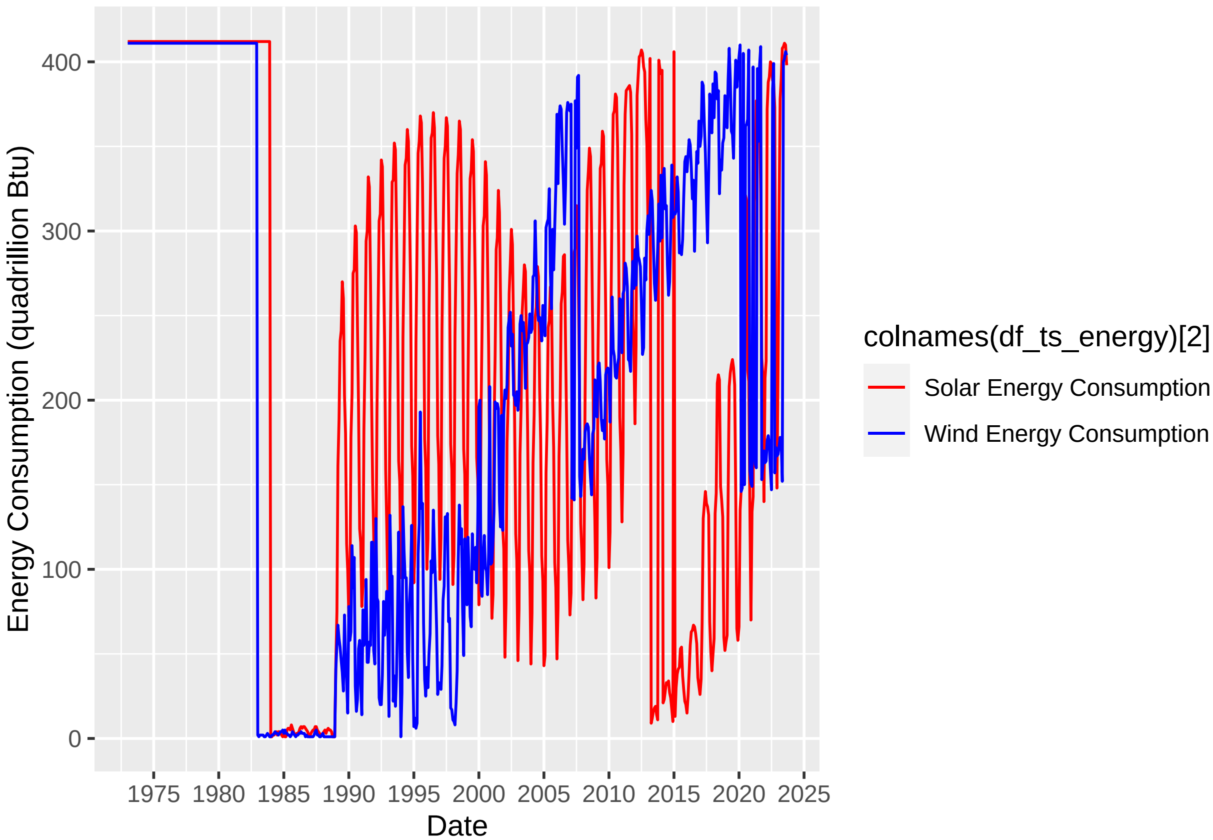


Q3

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.

```
print(ggplot(df_ts_energy)+
        geom_line(aes(x=Date,y=df_ts_energy[,2],color=colnames(df_ts_energy)[2]))+
        geom_line(aes(x=Date,y=df_ts_energy[,3],color=colnames(df_ts_energy)[3]))+
        ylab("Energy Consumption (quadrillion Btu)")+
        scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
        scale_color_manual(values = c("Solar Energy Consumption" = "red",
                                        "Wind Energy Consumption" = "blue"))
      )
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



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.

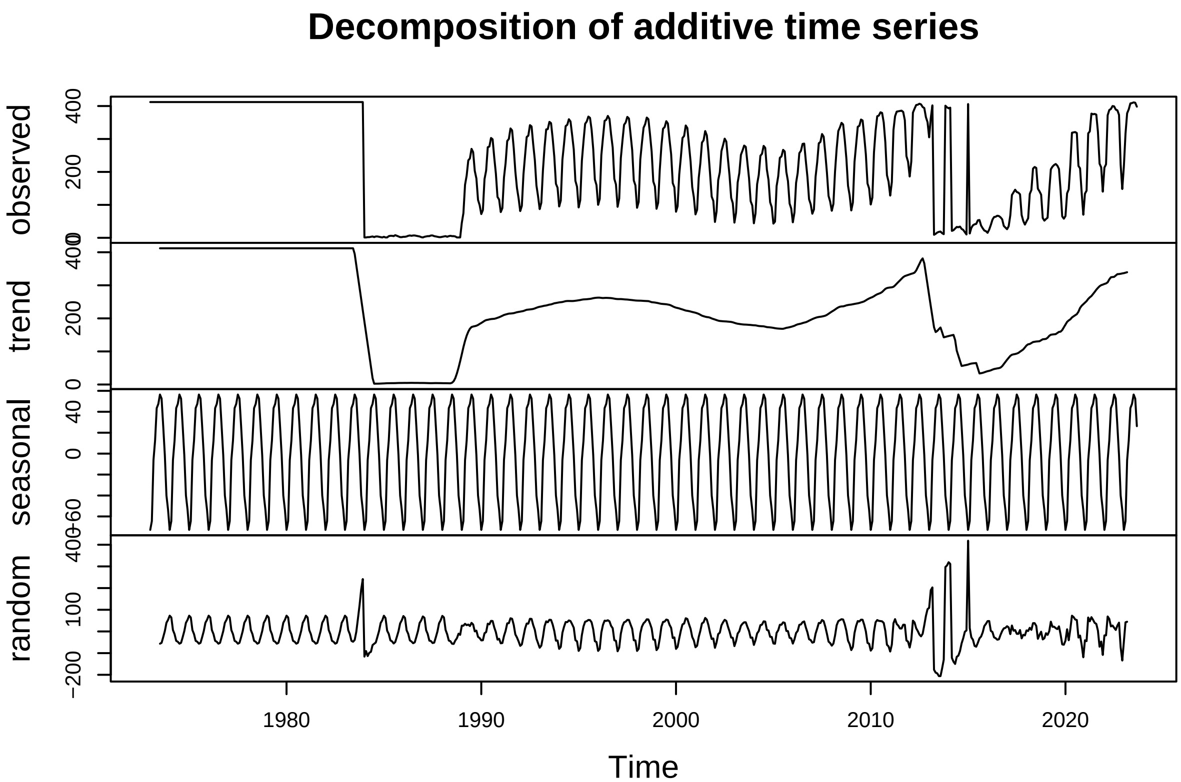
Additional info on `decompose()`.

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.

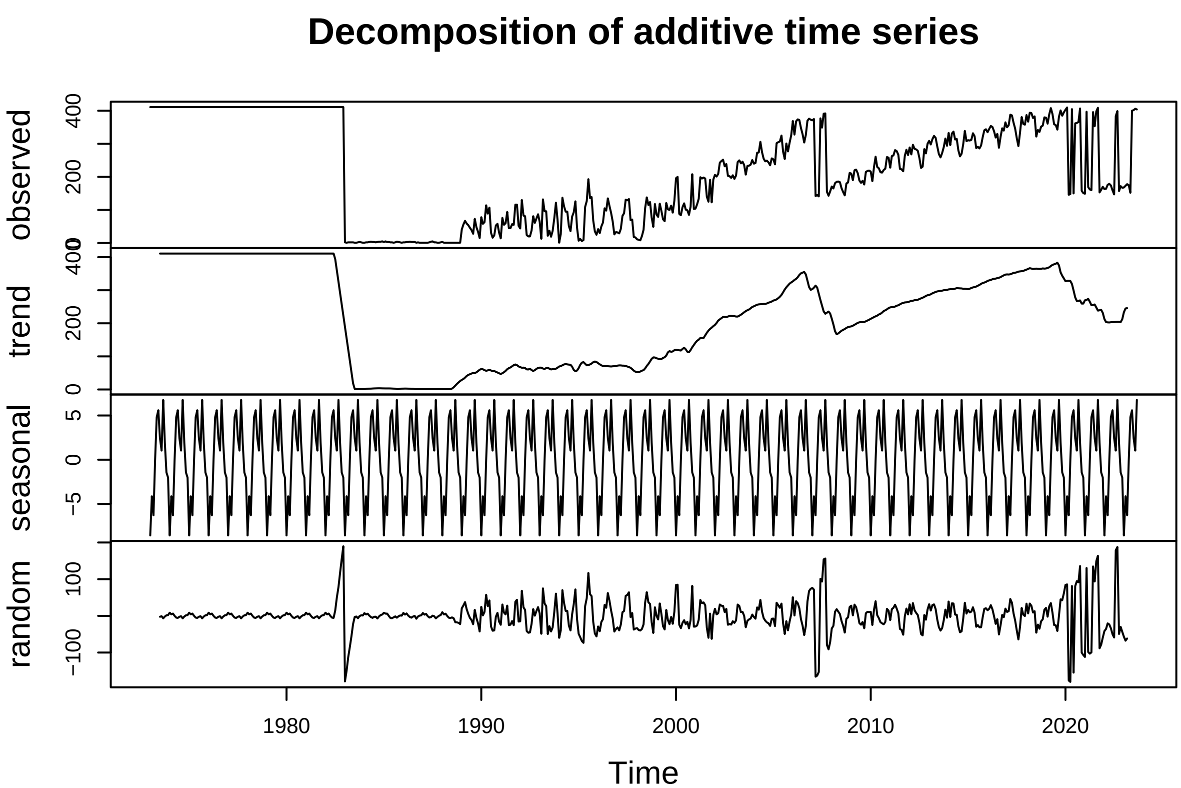
Q4

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?

```
ts_decompose_solar<-decompose(ts_energy[,1], type = "additive")
plot(ts_decompose_solar)
```




```r
ts_decompose_wind<-decompose(ts_energy[,2], type = "additive")
plot(ts_decompose_wind)
```



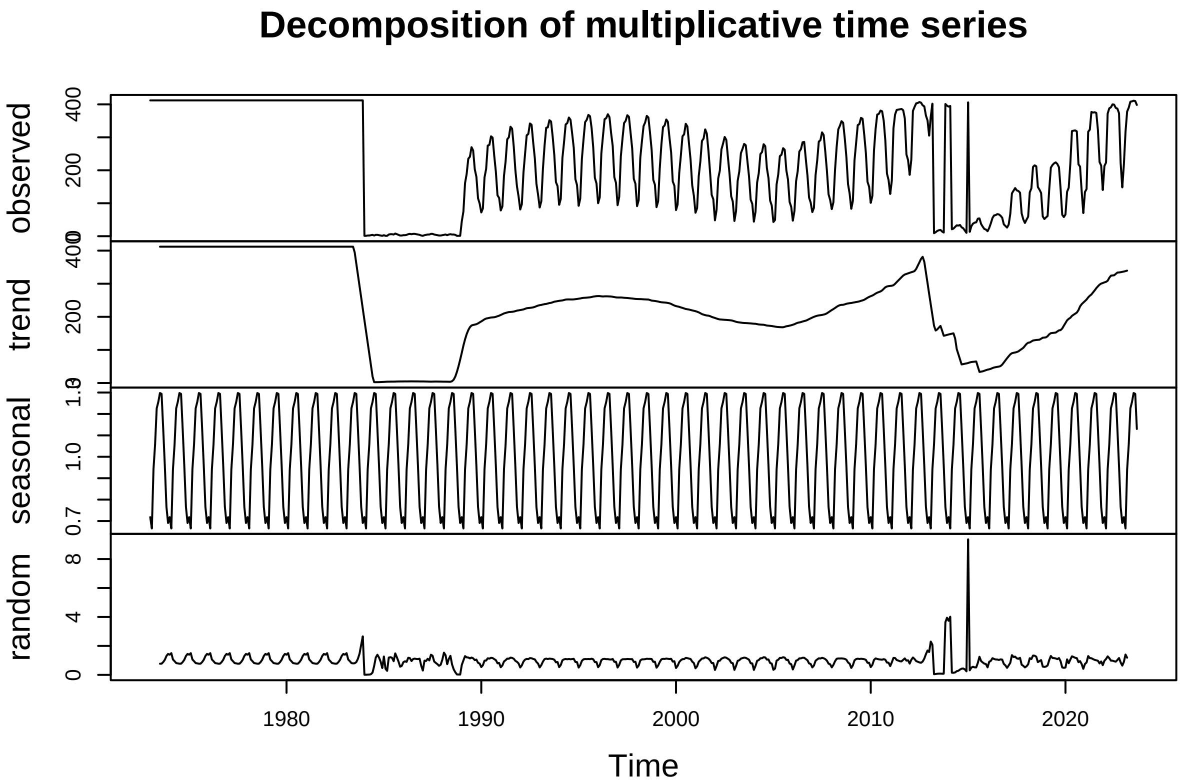
Solar energy consumption: 1) There was a shift in the level of solar energy consumption between 1980 and 1990. During 1990 and 2005, the consumption remained at approximately the same level. This was followed by a peak in consumption in 2013. There were some short-term shocks during 2013-2015, and the consumption presented an increased trend. 2) There are some spikes during the periods of 1983-1985 and 2013-2015. The wave-like pattern in the random component suggest the presence of seasonal components in it.

Wind energy consumption: 1) There was a shift in the level of wind energy consumption between 1980 and 1990. There were some short-term shocks during 2005-2010, followed by an increased trend from 2019 to 2018. The wind consumption shows a decreased trend after 2020. Overall, the wind energy consumption has an increased trend after 1990. 2) There seem to be some spikes during the periods of 1983-1985, 2005-2010, and after 2020. The wave-like pattern in the random component suggest the presence of seasonal components in it.

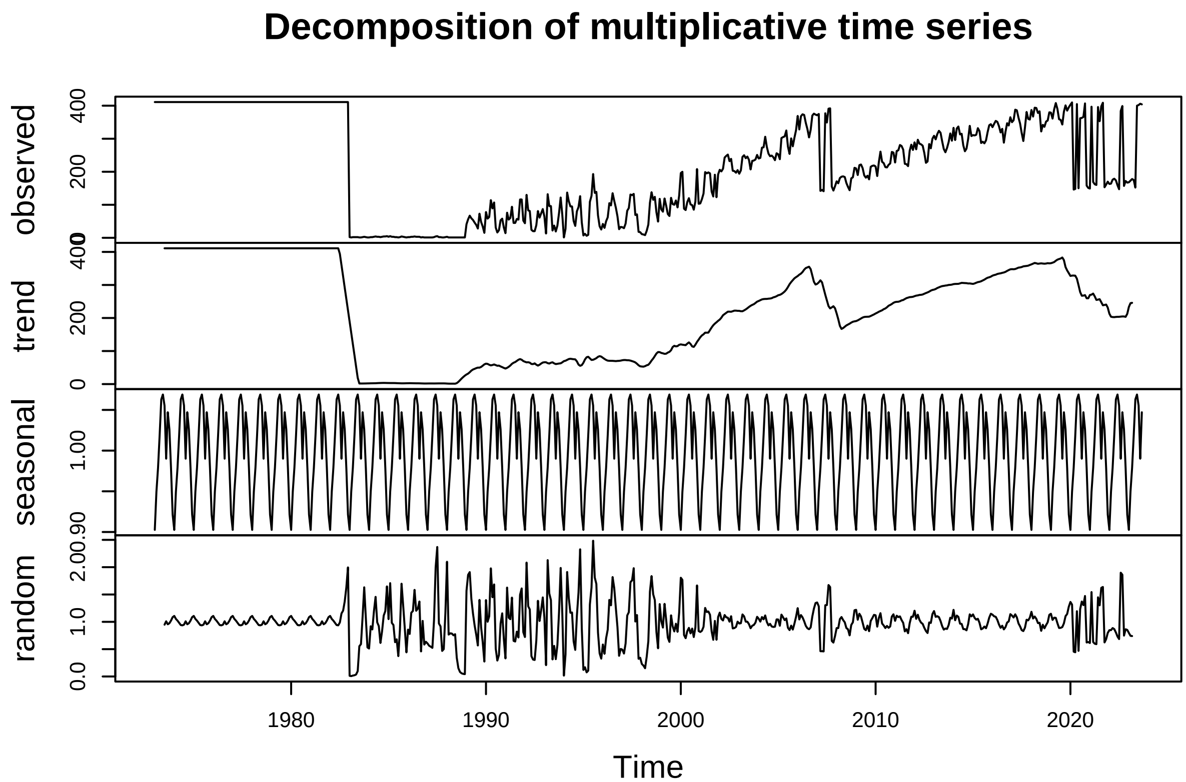
Q5

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?

```r
ts_decompose_multi_solar<-decompose(ts_energy[,1], type = "multiplicative")
plot(ts_decompose_multi_solar)
```




```r
ts_decompose_multi_wind<-decompose(ts_energy[,2], type = "multiplicative")
plot(ts_decompose_multi_wind)
```



1. Solar energy consumption: The random component level decreased, with the mean now varying around 0. The spike that occurred during 2013-2015 is now more evident.
2. Wind energy consumption: During the period of 1983-2000, the random component showed a higher degree of variability. In general, the variability tends to decrease with time.

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: It may not be necessary to include all historical data. For instance, records from the 1970s to the 1990s may contain errors that caused level drops. Data from the period between 1990 and 2000 could be useful in understanding the productivity of wind resources in different seasons; however, the current weather conditions might be different compared to 2 decades ago due to climate change. To predict the next six months of intermittent renewable energy consumption, more recent data might be more valuable.

Q7

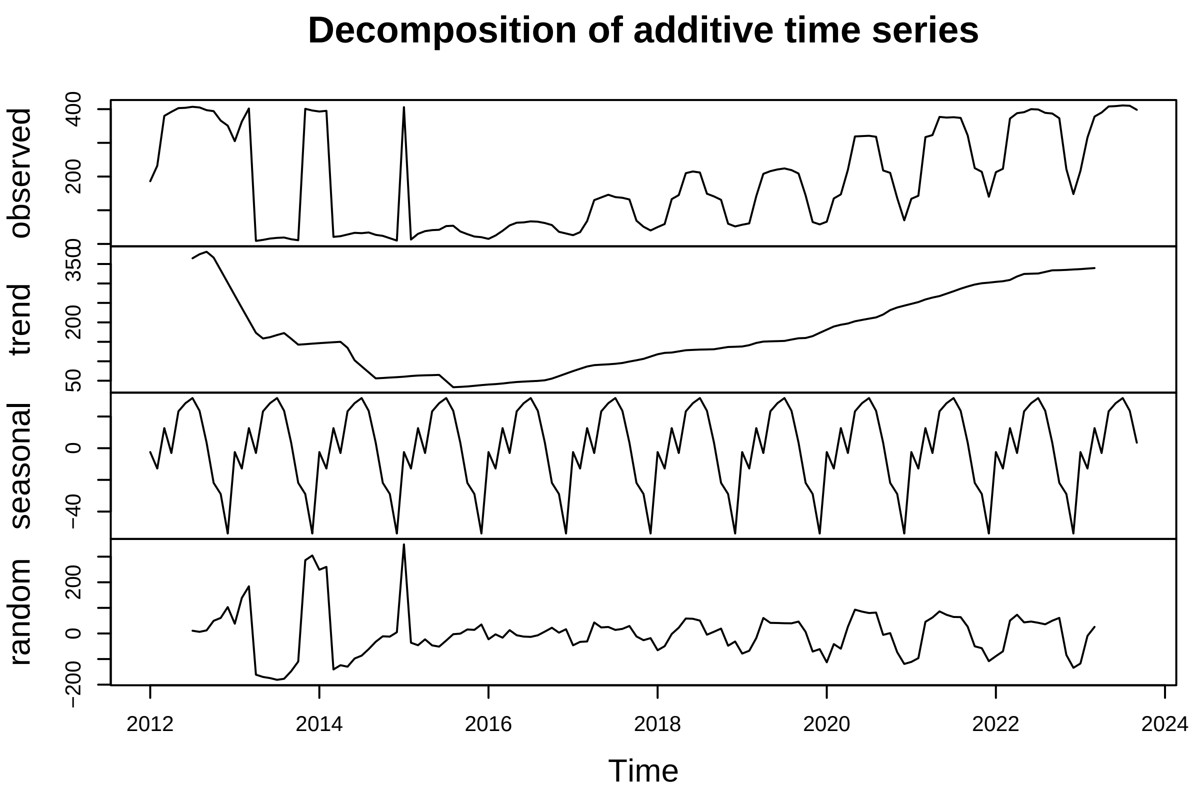
Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the decompose function `type=additive` to this new time series. Comment the results. Does the random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

```r
#select the date starts on Jan 2012
df_energy_2012<-df_energy %>%
  filter(year(Date)>=2012)
head(df_energy_2012)
```

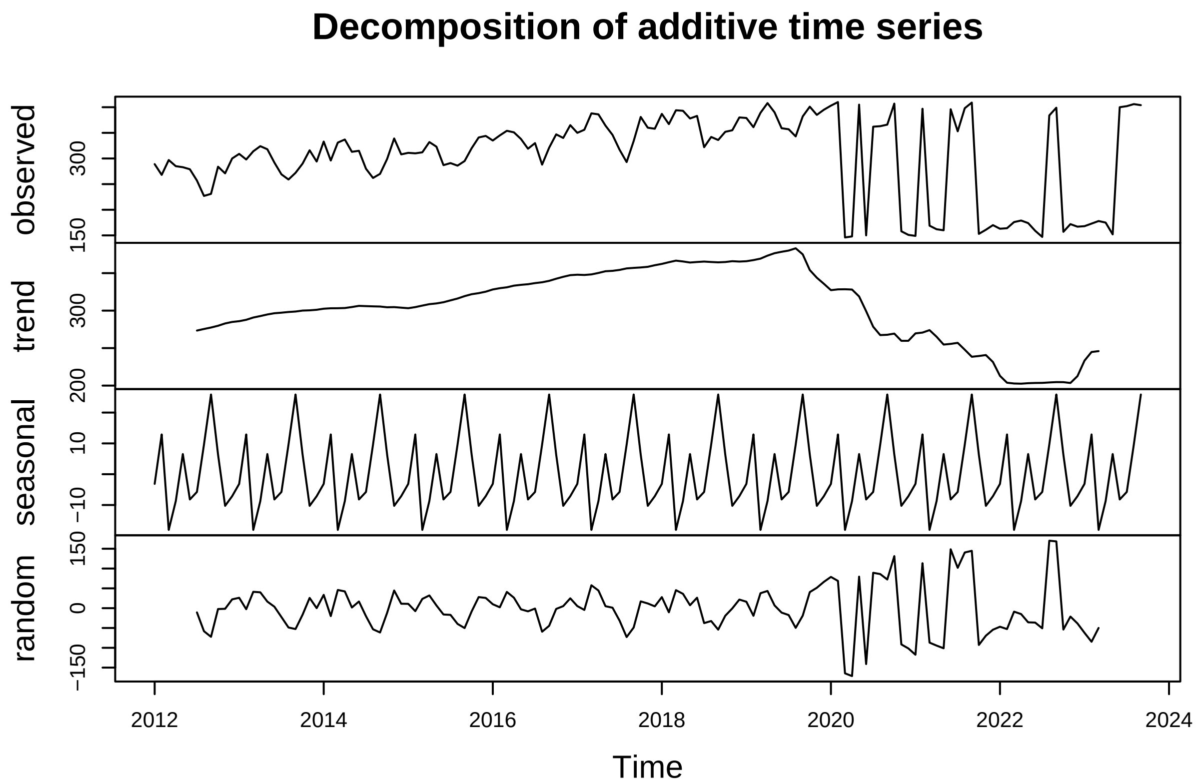
```
##         Date Solar Energy Consumption Wind Energy Consumption
## 1 2012-01-01                      186                     289
## 2 2012-02-01                      232                     268
## 3 2012-03-01                      380                     297
## 4 2012-04-01                      392                     285
## 5 2012-05-01                      403                     283
## 6 2012-06-01                      404                     279
```

```
#create time series object
year1_2012<-year(df_energy_2012$Date[1])
month1_2012<-month(df_energy_2012$Date[1])
ts_energy_2012<-ts(df_energy_2012[,2:3],start=c(year1_2012,month1_2012),frequency=12)

#decompose
ts_decompose_solar_2012<-decompose(ts_energy_2012[,1], type = "additive")
plot(ts_decompose_solar_2012)
```




```
ts_decompose_wind_2012<-decompose(ts_energy_2012[,2], type = "additive")
plot(ts_decompose_wind_2012)
```



Solar energy consumption 1) Trend component: The solar energy consumption decrease from 2012 to 2016. After 2016, the consumption present an increased trend. 2) Seasonal component: The seasonal component does not seem to show a correlation with the overall level of the series. 3) Random component: There are some spikes during the periods of 2013-2015. We can see a wave-like pattern in the random component after 2018, suggesting the presence of seasonal components in it.

Wind energy consumption 1) Trend component: Before 2020, the consumption of wind energy showed an increasing trend, reaching its peak in 2020 and then decreasing thereafter. 2) Seasonal component: The seasonal component deos not seem to show a correlation with the overall level of the series. 3) Random component: There are some spikes during the periods of 2020-2022. We can see a wave-like pattern in the random component from 2012 to 2019, suggesting there might be seasonal components in the series.

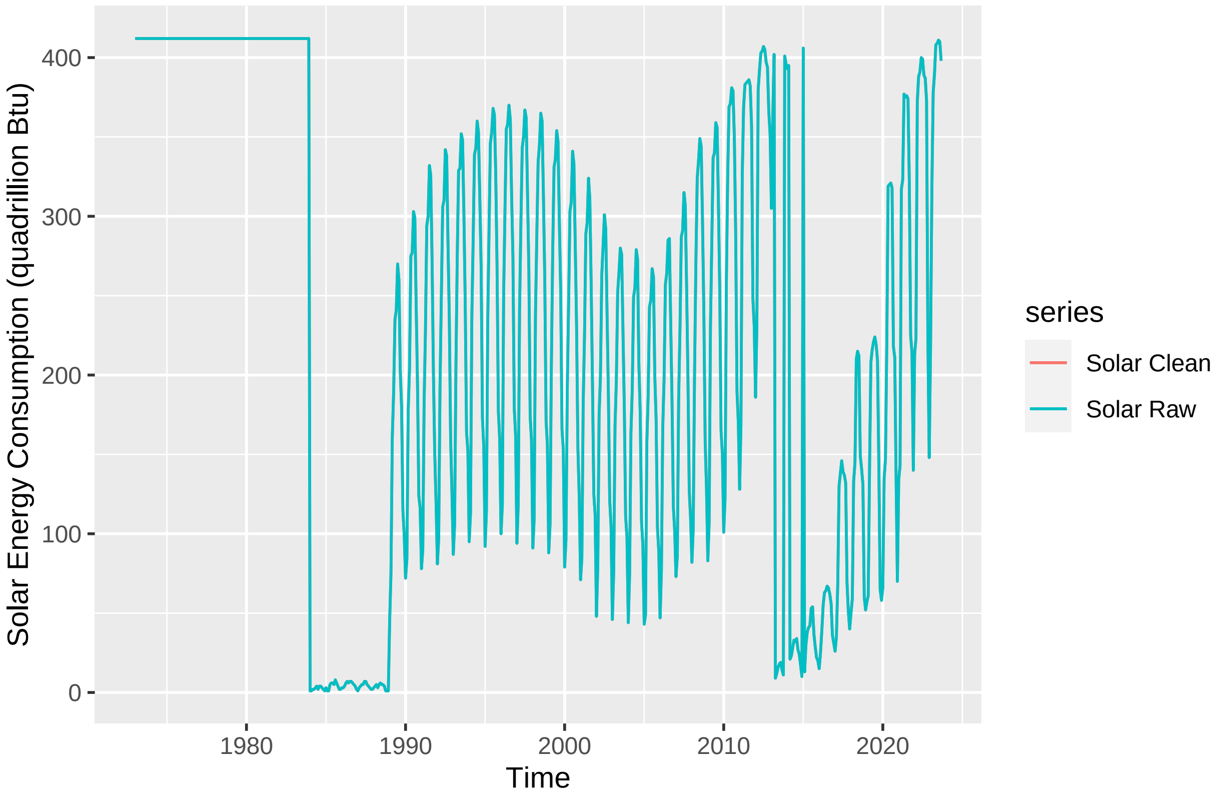
Identify and Remove outliers

Q8

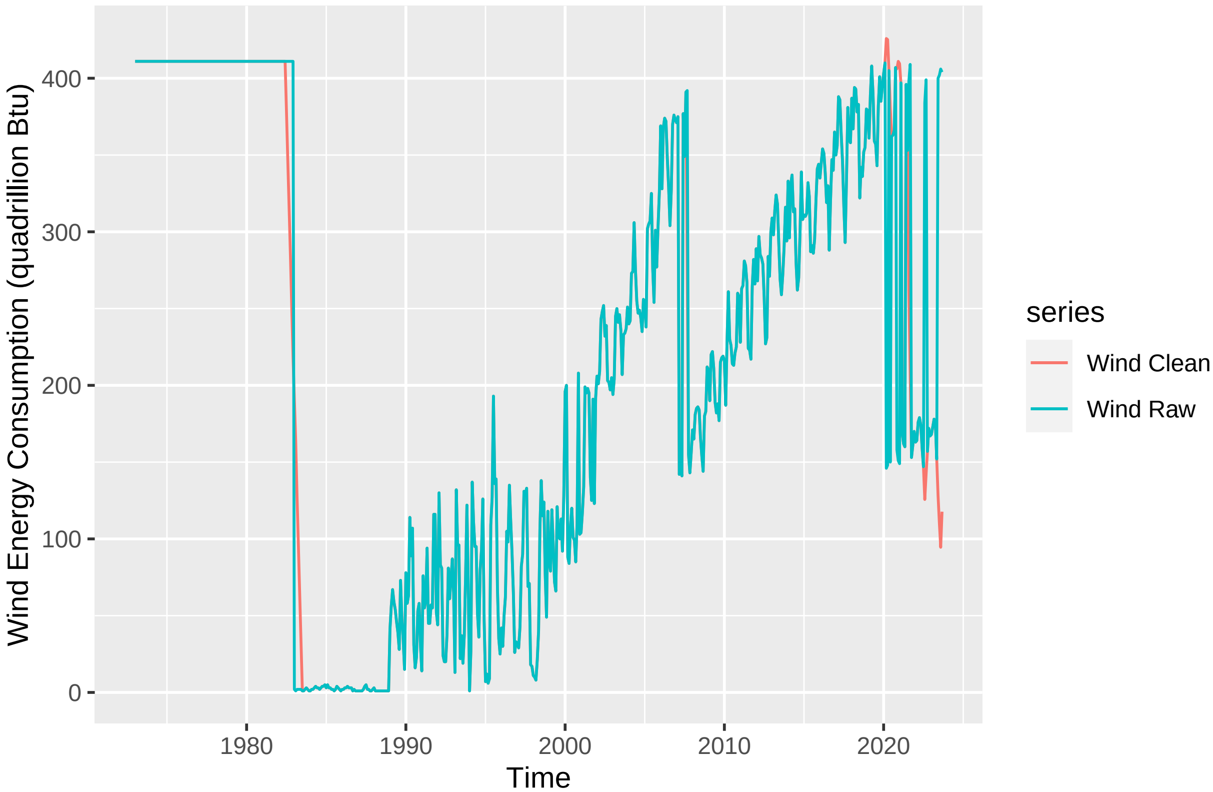
Apply the `tsclean()` to both series from Q7. Did the function removed any outliers from the series? Hint: Use `autoplot()` to check if there is difference between cleaned series and original series.

```
#ts-clean can be used for missing data and outliers
#object should be a time series object

#solar
clean_solar<-tsclean(ts_energy[,1])
plot_solar<-autoplot(clean_solar, series="Solar Clean") +
  autolayer(ts_energy[,1], series="Solar Raw") +
  ylab("Solar Energy Consumption (quadrillion Btu)")
plot_solar
```




```
#wind
clean_wind<-tsclean(ts_energy[,2])
plot_wind<-autoplot(clean_wind, series="Wind Clean") +
  autolayer(ts_energy[,2], series="Wind Raw") +
  ylab("Wind Energy Consumption (quadrillion Btu)")
plot_wind
```

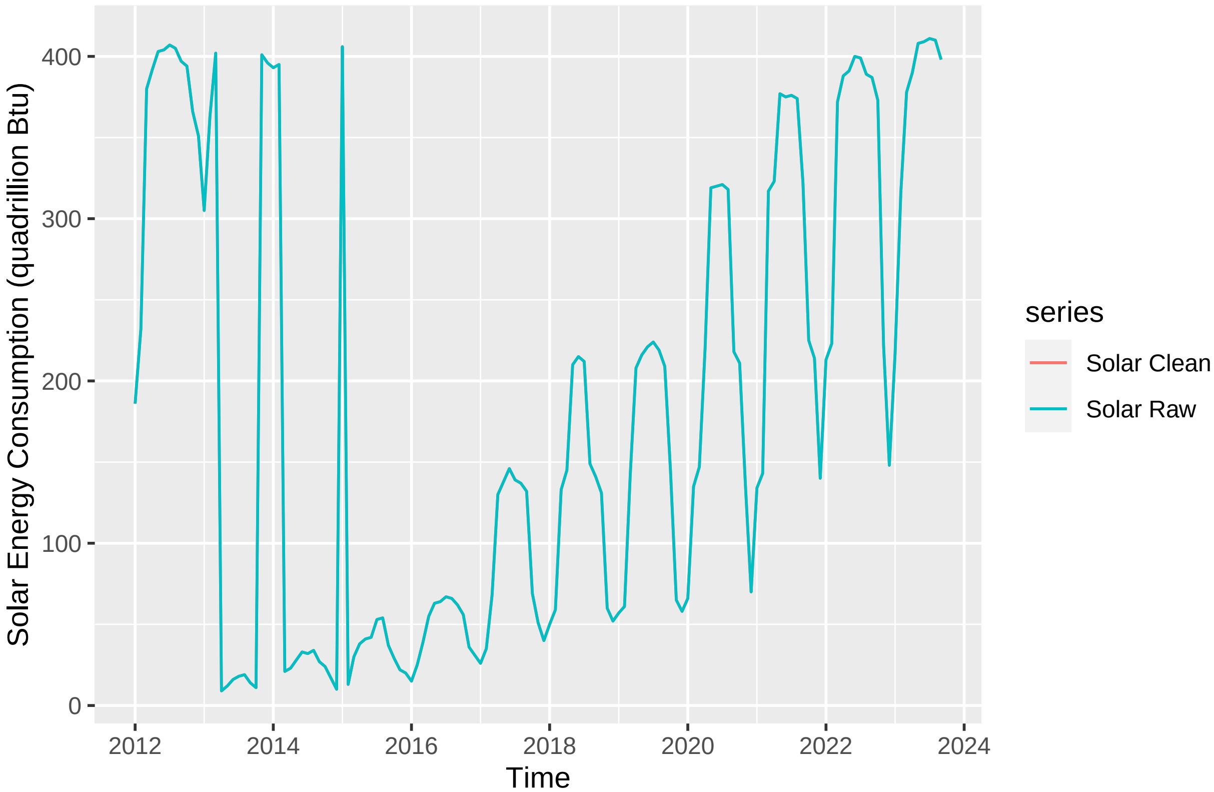


Answer: From the plots, we can see that outliers were not removed from the solar energy consumption data, whereas outliers were removed from the wind energy consumption data.

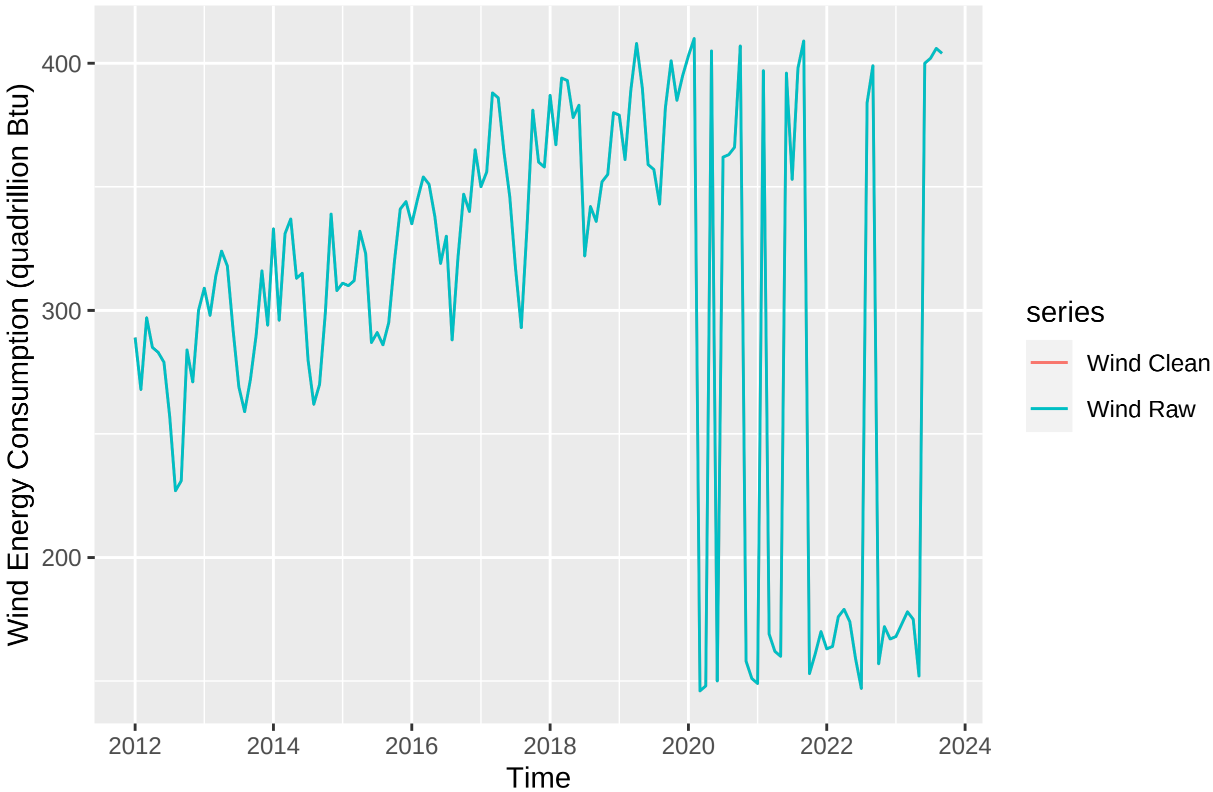
Q9

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?

```
#solar
clean_solar_2012<-tsclean(ts_energy_2012[,1])
plot_solar_2012<-autoplot(clean_solar_2012, series="Solar Clean") +
  autolayer(ts_energy_2012[,1], series="Solar Raw") +
  ylab("Solar Energy Consumption (quadrillion Btu)")
plot_solar_2012
```




```
#wind
clean_wind_2012<-tsclean(ts_energy_2012[,2])
plot_wind_2012<-autoplot(clean_wind_2012, series="Wind Clean") +
  autolayer(ts_energy_2012[,2], series="Wind Raw") +
  ylab("Wind Energy Consumption (quadrillion Btu)")
plot_wind_2012
```



Answer: From the plots, we can see that outliers were removed from the solar energy consumption data, whereas outliers were not removed from the wind energy consumption data.