ENV 797 - Time Series Analysis for Energy and Environment Applications | Spring 2025

Assignment 5 - Due date 02/18/25 - Extension Granted

Rachael Stephan

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_Sp25.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.\

```r
#Load/install required package here
library(forecast)
library(openxlsx)
library(readxl)
library(tseries)
library(ggplot2)
library(Kendall)
library(lubridate)
library(tidyverse)  #load this package so yon clean the data frame using pipes

#set theme
mytheme <- theme_bw(base_size = 10)+
  theme(axis.title = element_text(size = 10, hjust = 0.5),
        plot.title.position = "panel",
        panel.border = element_rect(colour = "black", fill = NA, linewidth = 0.25),
        plot.caption = element_text(hjust = 0),
        legend.box = "vertical",
        legend.location = "plot",
        axis.gridlines = element_line(color = "grey", linewidth = 0.25),
        axis.ticks = element_line(color = "black", linewidth = 0.5),
        axis.grid = element_blank())
theme_set(mytheme)
```

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.

```
#Importing data set - using xlsx package **I used read_xlsl because the code
#provided would not run
energy_data <- read_xlsx("./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx"
                         sheet = 1,
                         skip = 12,   # Skip first 12 rows to start at row 13
                         col_names = FALSE)  # Prevents automatic column names

#Now let's extract the column names from row 11 only
read_col_names <- read_xlsx("./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xl
                            sheet = 1,
                            skip = 10,
                            n_max = 1,
                            col_names = FALSE)

colnames(energy_data) <- read_col_names
head(energy_data)
```

```
## # A tibble: 6 x 14
##   Month               `Wood Energy Production` `Biofuels Production`
##   <dttm>                                 <dbl> <chr>
## 1 1973-01-01 00:00:00                     130. Not Available
## 2 1973-02-01 00:00:00                     117. Not Available
## 3 1973-03-01 00:00:00                     130. Not Available
## 4 1973-04-01 00:00:00                     125. Not Available
## 5 1973-05-01 00:00:00                     130. Not Available
## 6 1973-06-01 00:00:00                     125. Not Available
## # i 11 more variables: `Total Biomass Energy Production` <dbl>,
## #   `Total Renewable Energy Production` <dbl>,
## #   `Hydroelectric Power Consumption` <dbl>,
## #   `Geothermal Energy Consumption` <dbl>, `Solar Energy Consumption` <chr>,
## #   `Wind Energy Consumption` <chr>, `Wood Energy Consumption` <dbl>,
## #   `Waste Energy Consumption` <dbl>, `Biofuels Consumption` <chr>,
## #   `Total Biomass Energy Consumption` <dbl>, ...
```

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

Q1

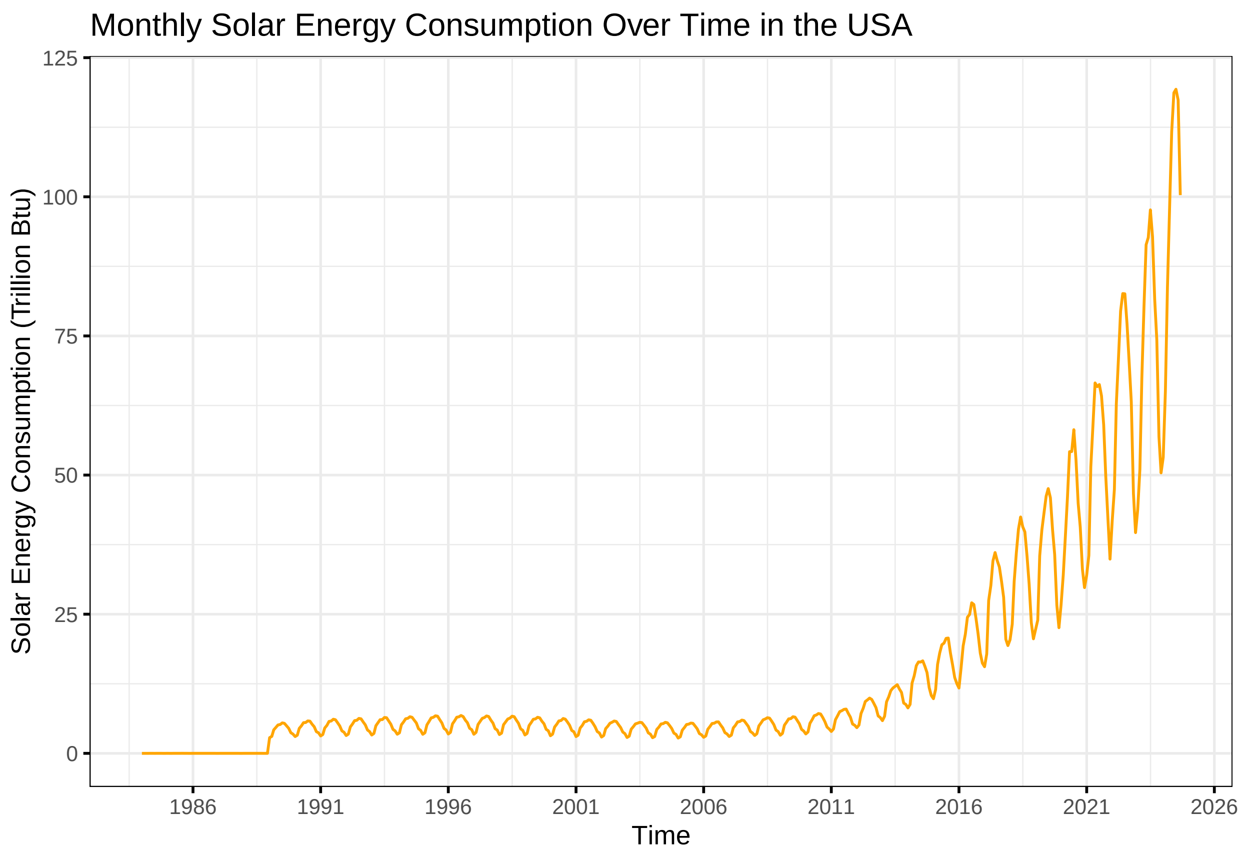
For this assignment you will work only with the following columns: Solar Energy Consumption and Wind Energy Consumption. Create a data frame structure with these two time series only and the Date column. Drop the rows with *Not Available* and convert the columns to numeric. You can use filtering to eliminate 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
energy_data <- energy_data %>%
  select(Month,
         `Solar Energy Consumption`,
         `Wind Energy Consumption`) %>%
  mutate(across(c(`Solar Energy Consumption`, `Wind Energy Consumption`), as.numeric)) %>%
  mutate(Month = as.Date(Month, format = "%Y-%m")) %>%
  na.omit()
```

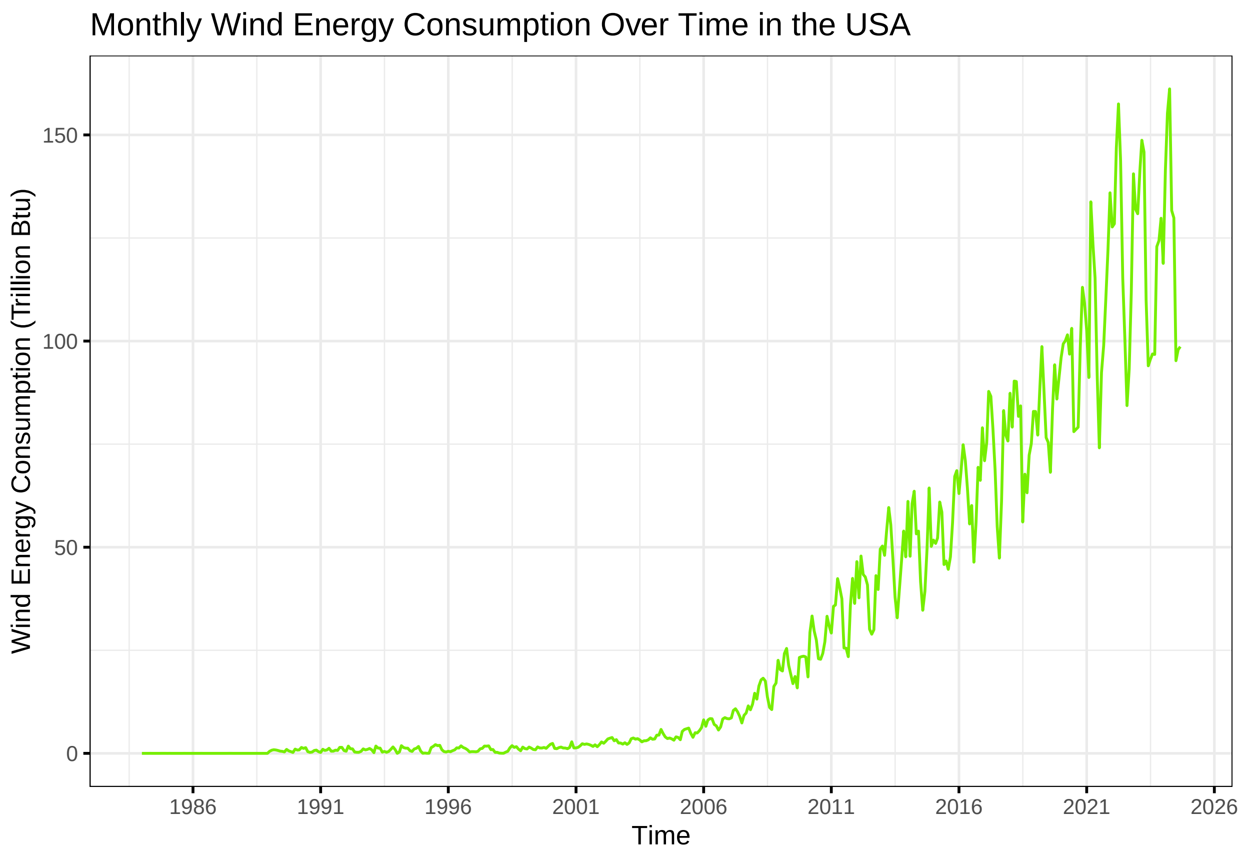
Q2

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")`

```r
#solar plot
ggplot(data = energy_data, aes(x = Month, y = `Solar Energy Consumption`))+
  geom_line(colour = "orange")+
  labs(title = "Monthly Solar Energy Consumption Over Time in the USA",
       y = "Solar Energy Consumption (Trillion Btu)",
       x = "Time")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```



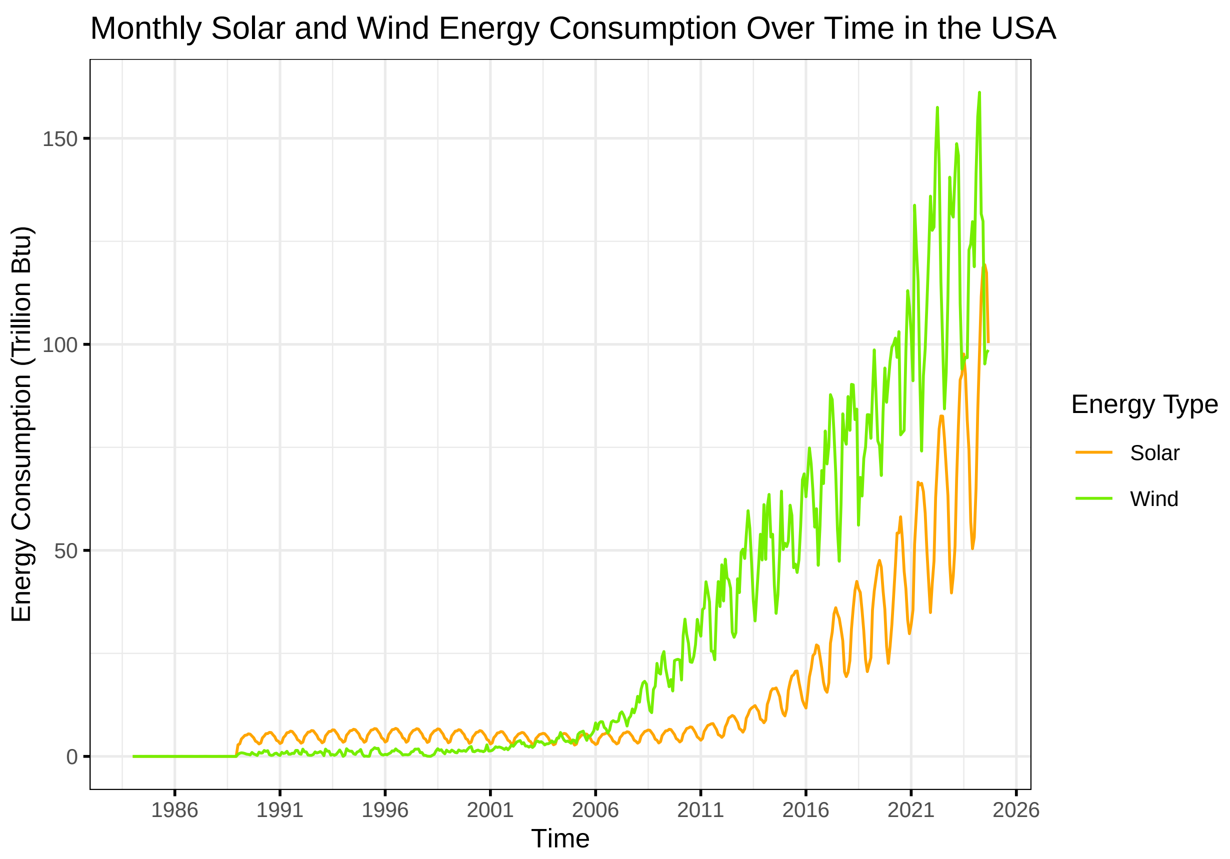
```r
#wind plot
  ggplot(data = energy_data, aes(x = Month, y = `Wind Energy Consumption`))+
  geom_line(colour = "chartreuse2")+
  labs(title = "Monthly Wind Energy Consumption Over Time in the USA",
       y = "Wind Energy Consumption (Trillion Btu)",
       x = "Time")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```



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.

```r
ggplot(data = energy_data, aes(x = Month))+
  geom_line(aes(y = `Solar Energy Consumption`, colour = "Solar"))+
  geom_line(aes(y = `Wind Energy Consumption`, colour = "Wind"))+
  labs(title = "Monthly Solar and Wind Energy Consumption Over Time in the USA",
       y = "Energy Consumption (Trillion Btu)",
       x = "Time",
       colour = "Energy Type")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  scale_color_manual(values = c("Solar" = "orange", "Wind" = "chartreuse2"))
```



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.

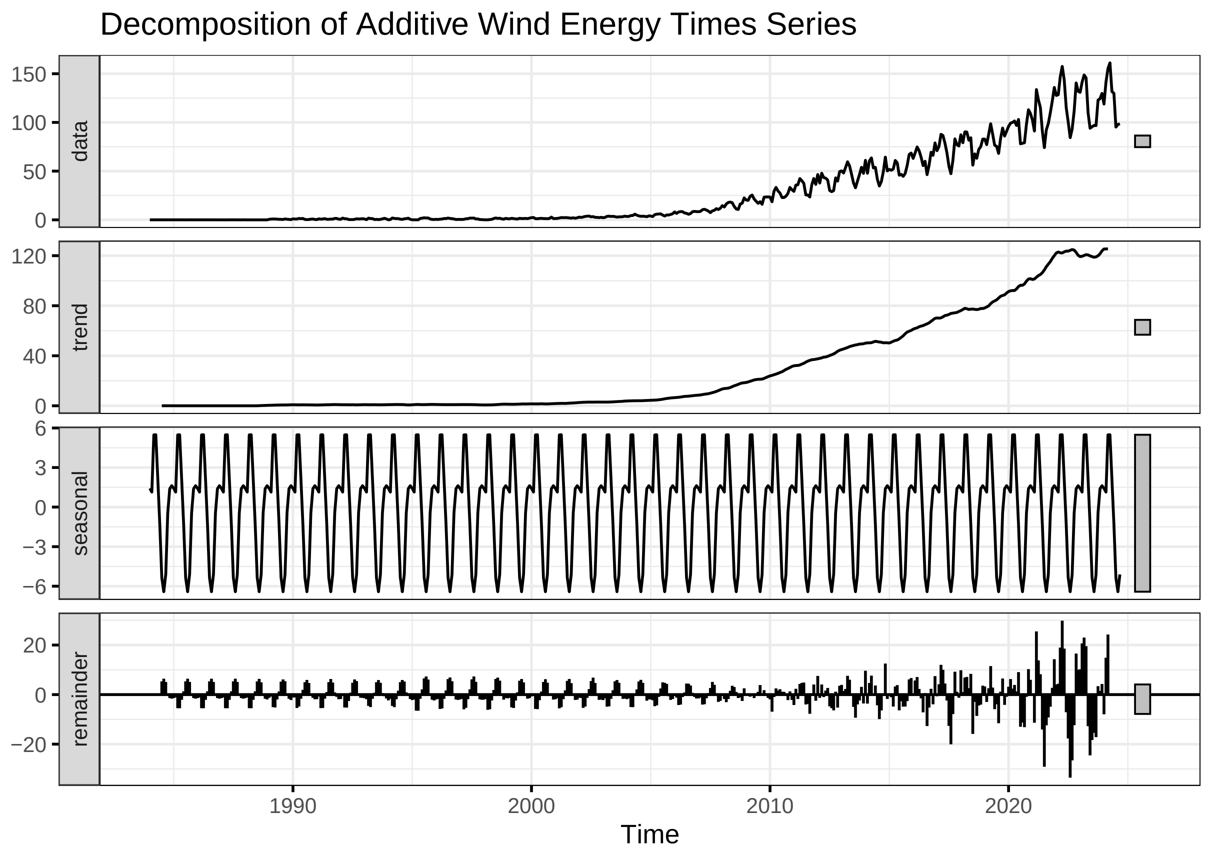
Q4

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?

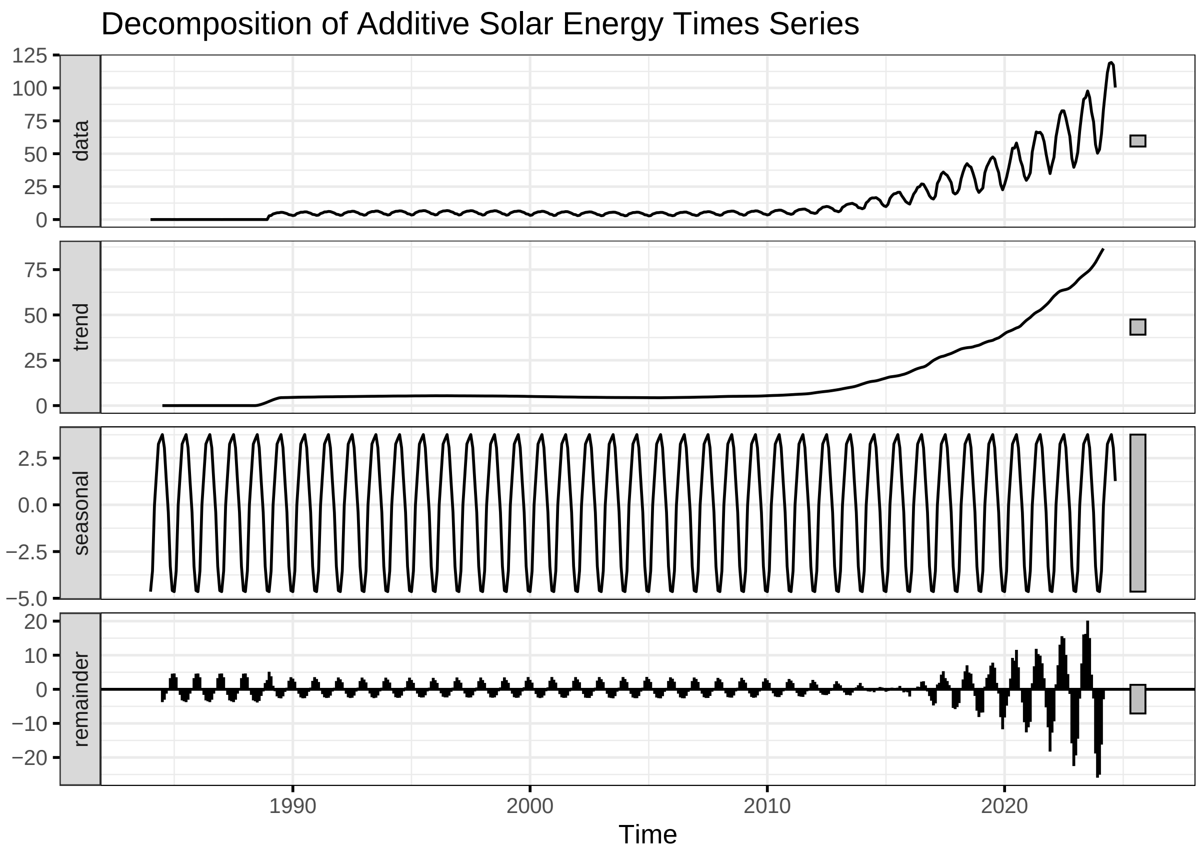
```r
#create ts
energy_ts <- ts(energy_data[,2:3],
                start=c(1984,1),
                frequency=12)

#decompose the ts
wind_ts_decomp <- decompose(energy_ts[,2], type = "additive")
solar_ts_decomp <- decompose(energy_ts[,1], type = "additive")

#plot the decomposed ts
autoplot(wind_ts_decomp)+
  labs(title = "Decomposition of Additive Wind Energy Times Series")
```



```r
autoplot(solar_ts_decomp)+
  labs(title = "Decomposition of Additive Solar Energy Times Series")
```



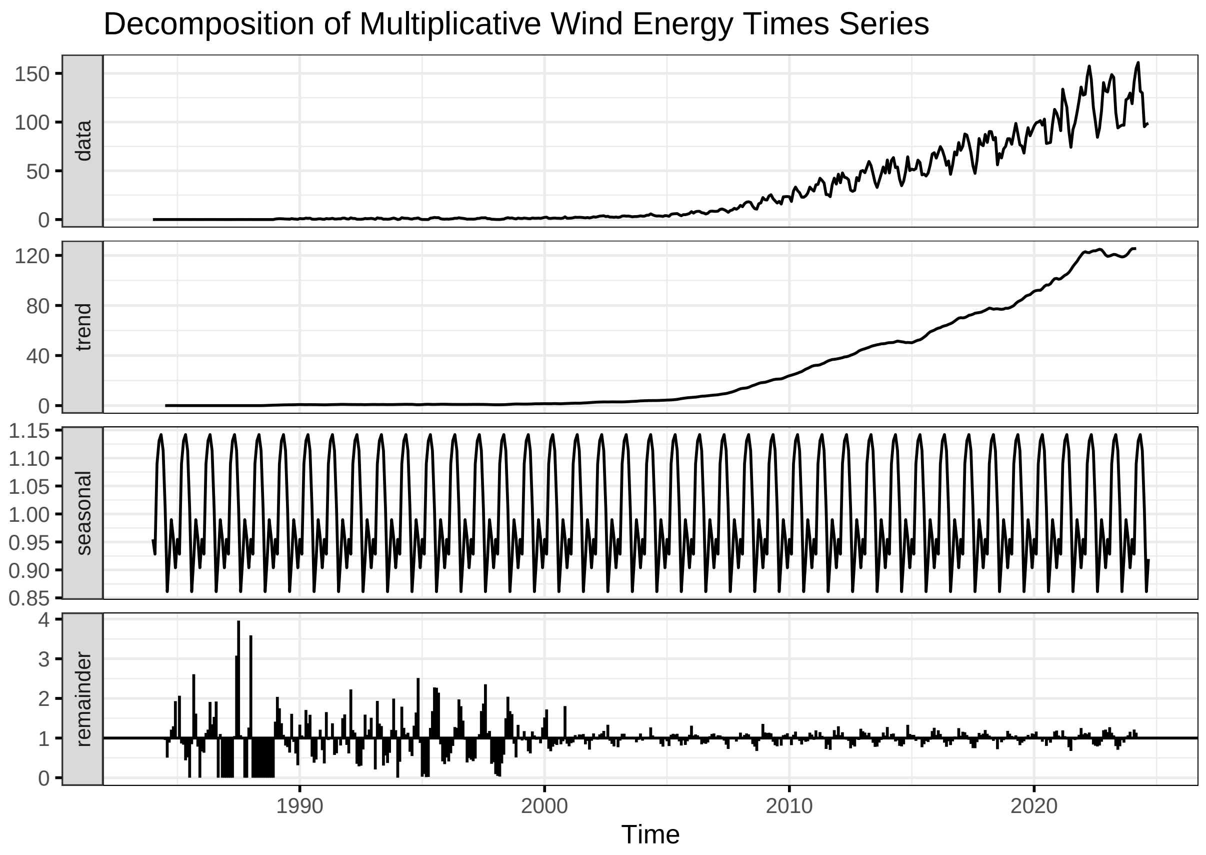
The trend component of the both energy time series have an initial lag period before they start to increase. The wind energy time series appears to increase at a relatively steady rate. Whereas the solar energy appears to have more of an exponential trend after the lag. Neither of the remainders look to be random. They both oscillate around zero with reltatively similar amplitudes, up until after 2010 for wind energy and 2015 for solar energy the random component amplitudes begin to increase. This random component may be because there is more than one trend within the timeseries or an additive model does not capture the seasonality.

Q5

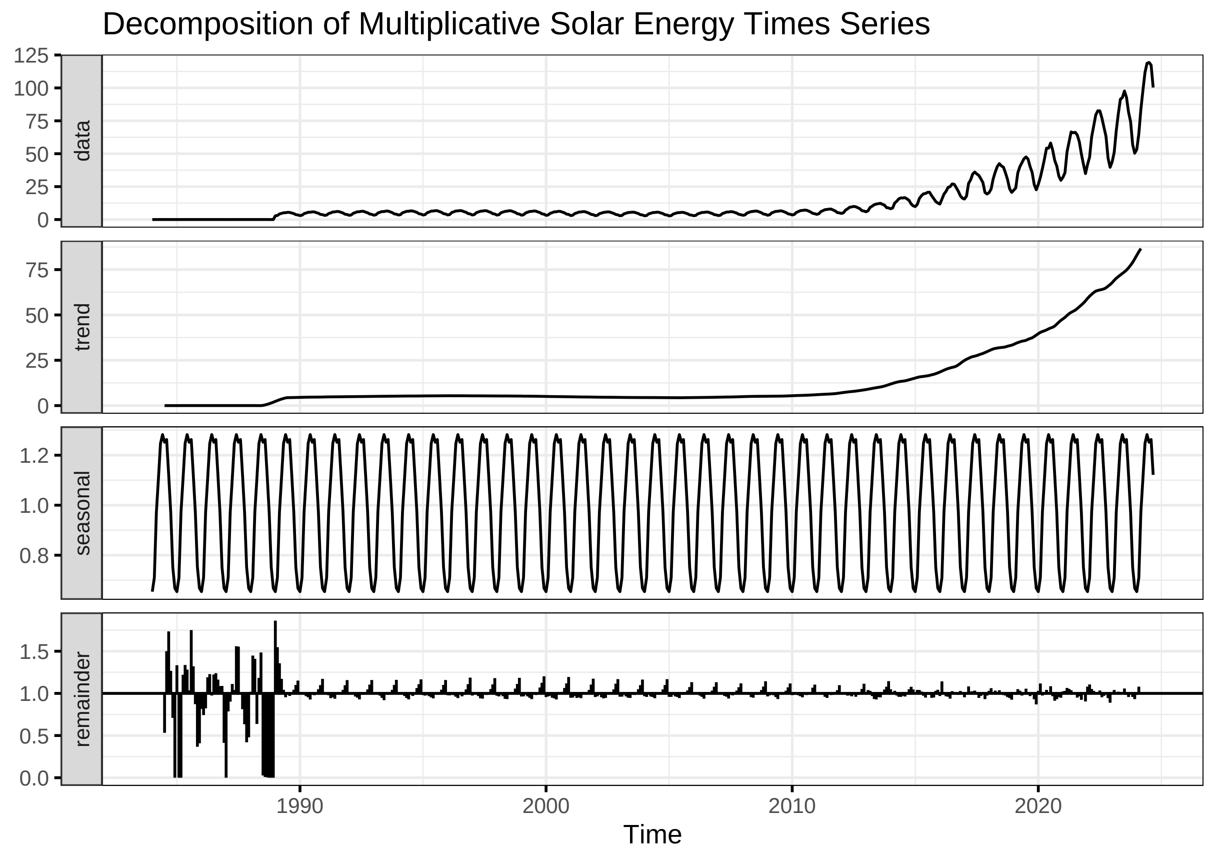
Use the decompose function again but now change the type of the seasonal component from additive to multiplicative. What happened to the random component this time?

```r
#decompose the ts
wind_ts_decomp_multi <- decompose(energy_ts[,2], type = "multiplicative")
solar_ts_decomp_multi <- decompose(energy_ts[,1], type = "multiplicative")

#plot the decomposed ts
autoplot(wind_ts_decomp_multi)+
  labs(title = "Decomposition of Multiplicative Wind Energy Times Series")
```



```
autoplot(solar_ts_decomp_multi)+
  labs(title = "Decomposition of Multiplicative Solar Energy Times Series")
```



The random component still appears to contain some seasonality within the later part of the time series. However, the amplitude of these random components are much smaller than compared to the additive decomposition, indicating this decomposition did a better job at removing the seasonality. The initial part of the times series has larger amplitudes (but still smaller than additive amplitudes) in both time series. This could be due to the appearance of two different trends in the original data series (i.e., the lag period followed by the quick increase) not being able to handle little to no energy consumption in the earlier component of the time series.

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.

I do not think all historical data is required. Up until 2005 for wind energy and 2010 for solar energy, the energy consumption remained very minimal and close to zero. This is likely because the technologies were not in high use or highly developed at the time. The early data for solar and wind energy consumption before large scale consumption would likely not be helpful for predicting the future consumption of solar and wind energy. Thus, it may be more helpful to remove it to create a more uniform trend.

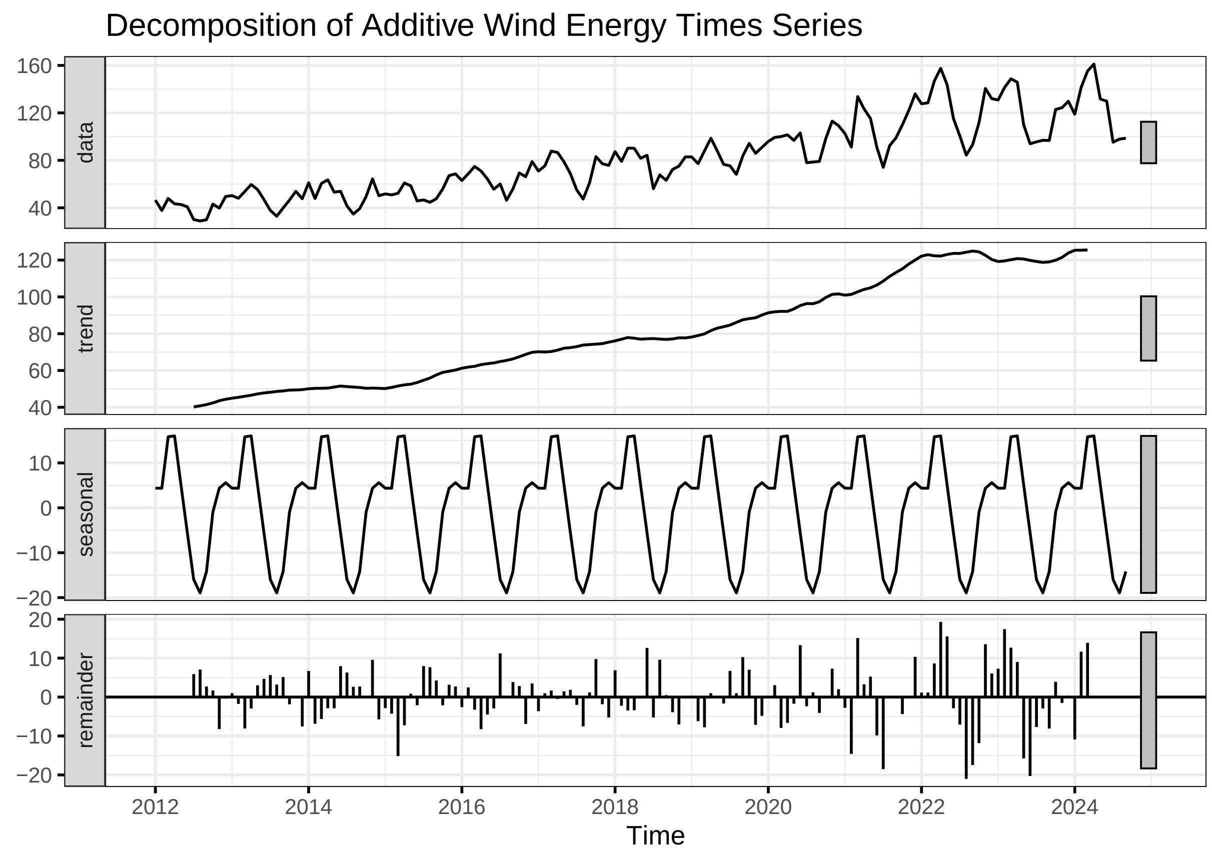
Q7

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.

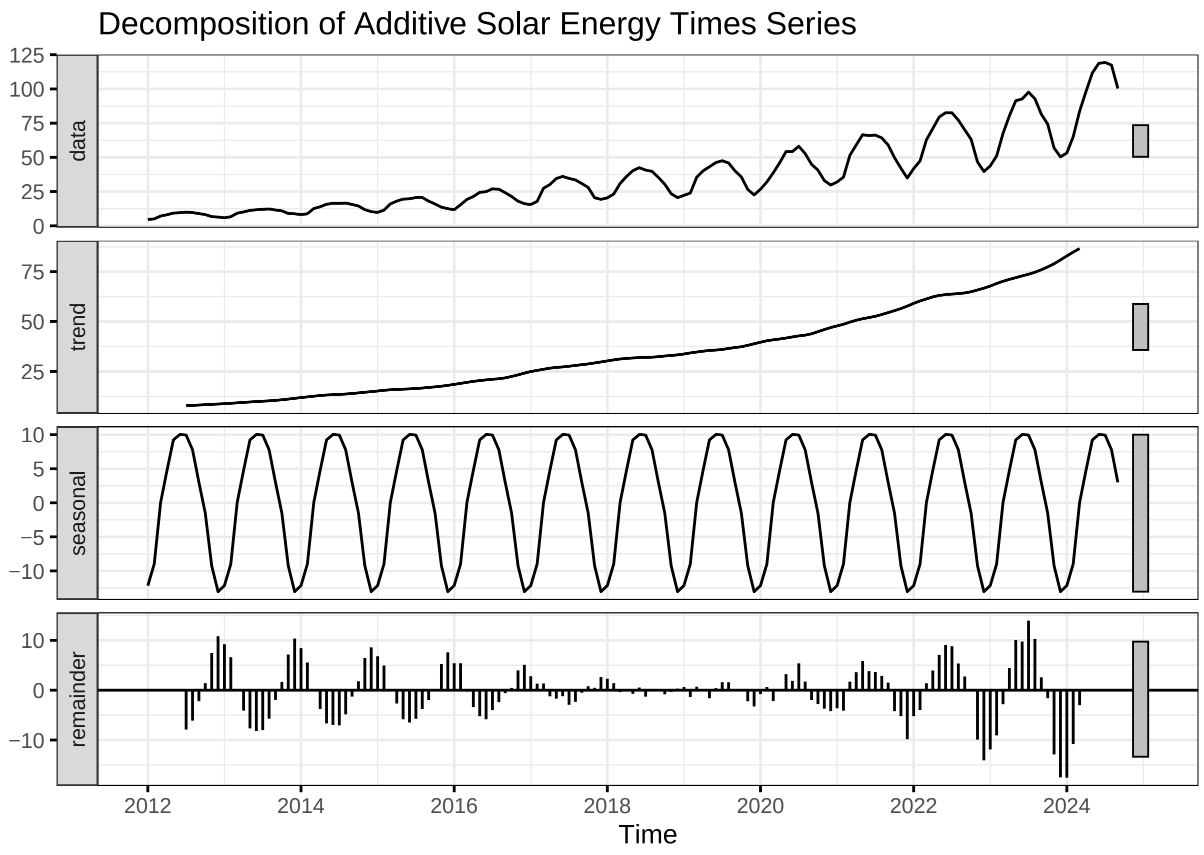
```r
energy_ts_recent <- energy_data %>%
  filter(year(Month) >= 2012) %>%
  select(`Solar Energy Consumption`, `Wind Energy Consumption`) %>%
  ts(start = c(2012, 1), frequency = 12)

#decompose the ts
wind_ts_decomp_recent <- decompose(energy_ts_recent[,2], type = "additive")
solar_ts_decomp_recent <- decompose(energy_ts_recent[,1], type = "additive")

#plot the decomposed ts
autoplot(wind_ts_decomp_recent)+
  labs(title = "Decomposition of Additive Wind Energy Times Series")
```



```r
autoplot(solar_ts_decomp_recent)+
  labs(title = "Decomposition of Additive Solar Energy Times Series")
```



The random components appears to contain seasonality within the earlier and later parts of the solar time series. This may be because the series is not additive, as was specified in the decomposition. In the original data, the oscillations around the trend line get larger as time goes on. This may indicate that the solar energy time series is multiplicative. If this is the case and the seasonal trend was taken to be the average difference between peaks and troughs, the seasonality component would be too small and too large for the early and late parts of the time series, respectively. The wind time series remainders have less of a clear trend, but it still appears like there may be a pattern to the remainder that could relate to seasonality. Therefore, the wind series may also not be additive, like the solar series.

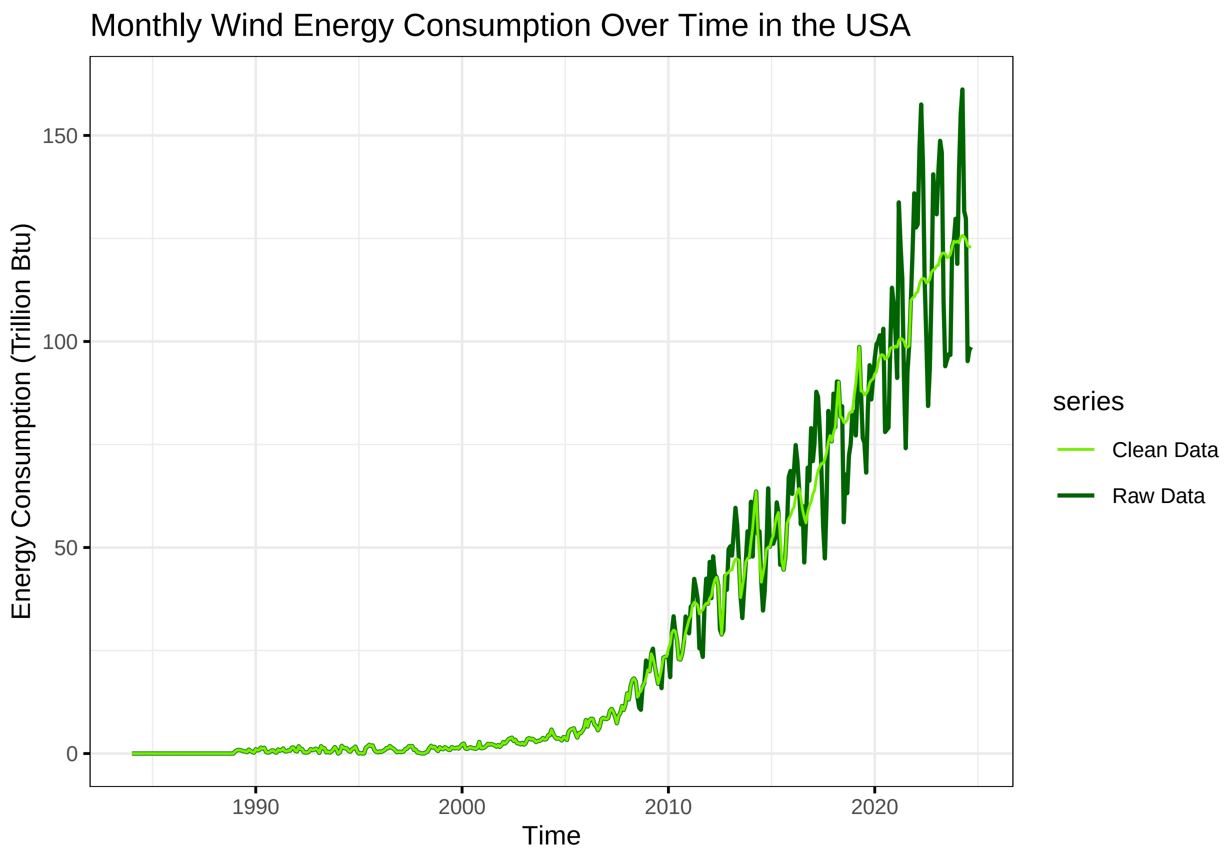
Identify and Remove outliers

Q8

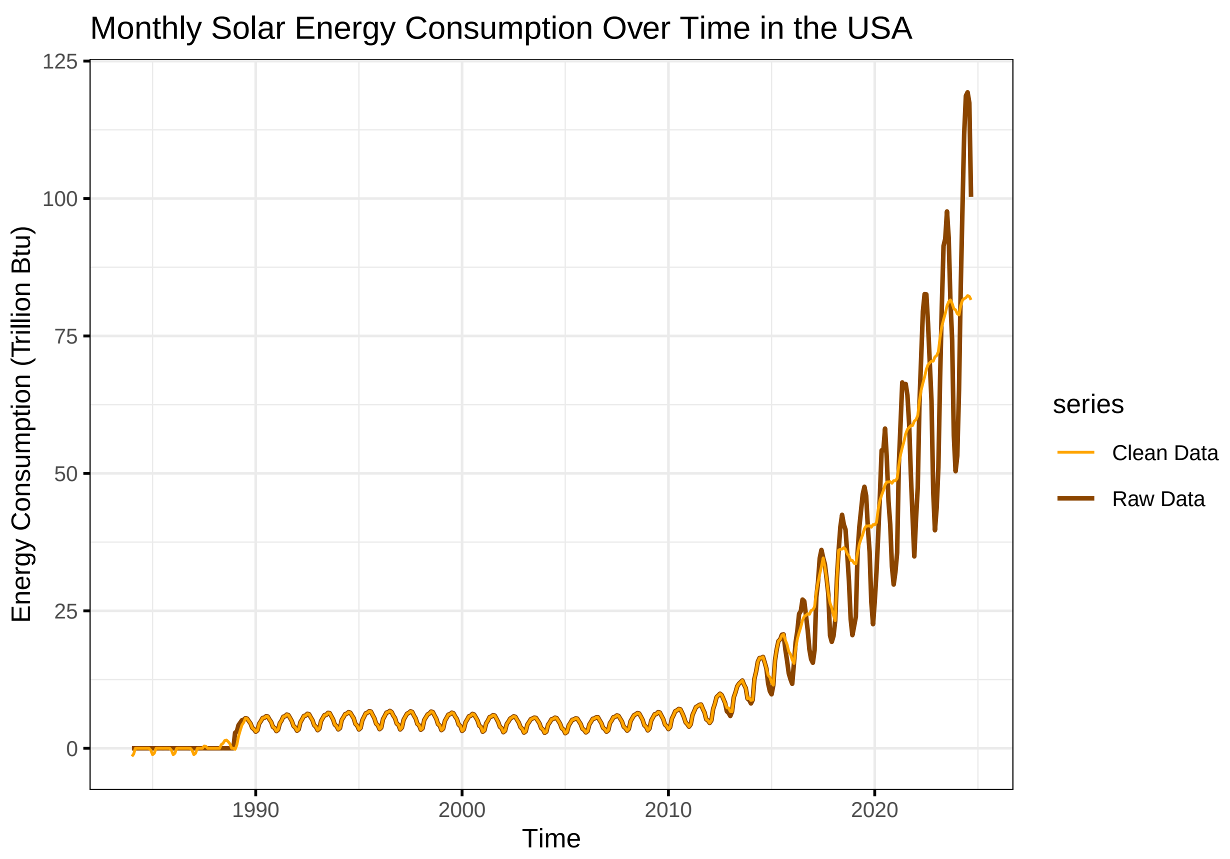
Apply the tsclean() to both time series object you created on Q4. Did the function removed any outliers from the series? Hint: Use autoplot() to check if there is difference between cleaned series and original series.

```r
#clean the timeseries
wind_ts_clean <- tsclean(energy_ts[,2])
solar_ts_clean <- tsclean(energy_ts[,1])

#plot the series against the original
autoplot(energy_ts[,2], series = "Raw Data", size = 0.75)+
  autolayer(wind_ts_clean, series = "Clean Data")+
  labs(title = "Monthly Wind Energy Consumption Over Time in the USA",
      y = "Energy Consumption (Trillion Btu)")+
  scale_color_manual(values = c("Raw Data" = "darkgreen", "Clean Data" = "chartreuse2"))
```



```r
autoplot(energy_ts[,1], series = "Raw Data", size = 0.8)+
  autolayer(solar_ts_clean, series = "Clean Data")+
  labs(title = "Monthly Solar Energy Consumption Over Time in the USA",
      y = "Energy Consumption (Trillion Btu)")+
  scale_color_manual(values = c("Raw Data" = "darkorange4", "Clean Data" = "orange"))
```



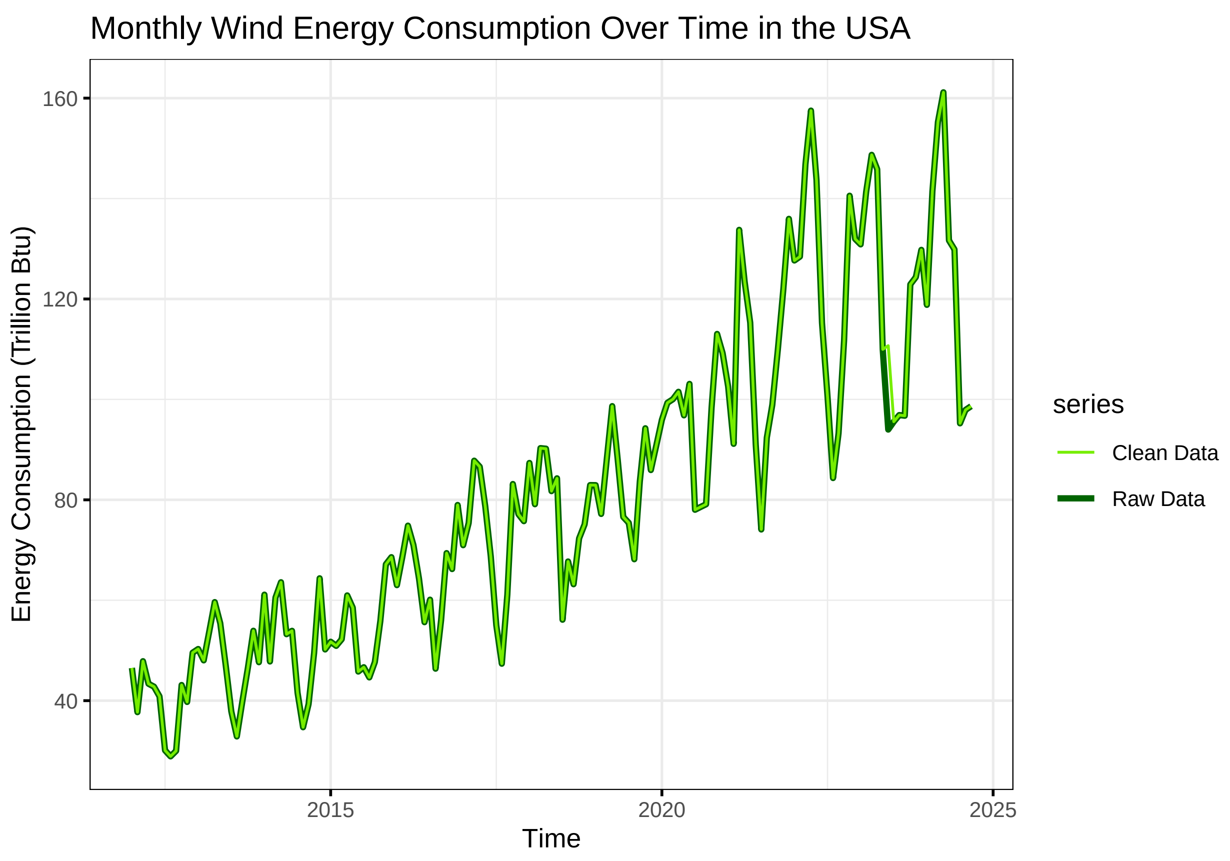
The `ts_clean` function removed many outliers from the both energy data sets. The `ts_clean` function does not do much to the early parts of the time series (i.e., before 2010 in the solar series and before 2005 in the wind series). Once the trend of the time series start to increase, the `ts_clean` function removes the peaks and the troughs caused by the seasonality. This may be because the `ts_clean` function cannot handle the different trends in the early vs. later parts of the time series, especially due to potential multiplicative trends.

Q9

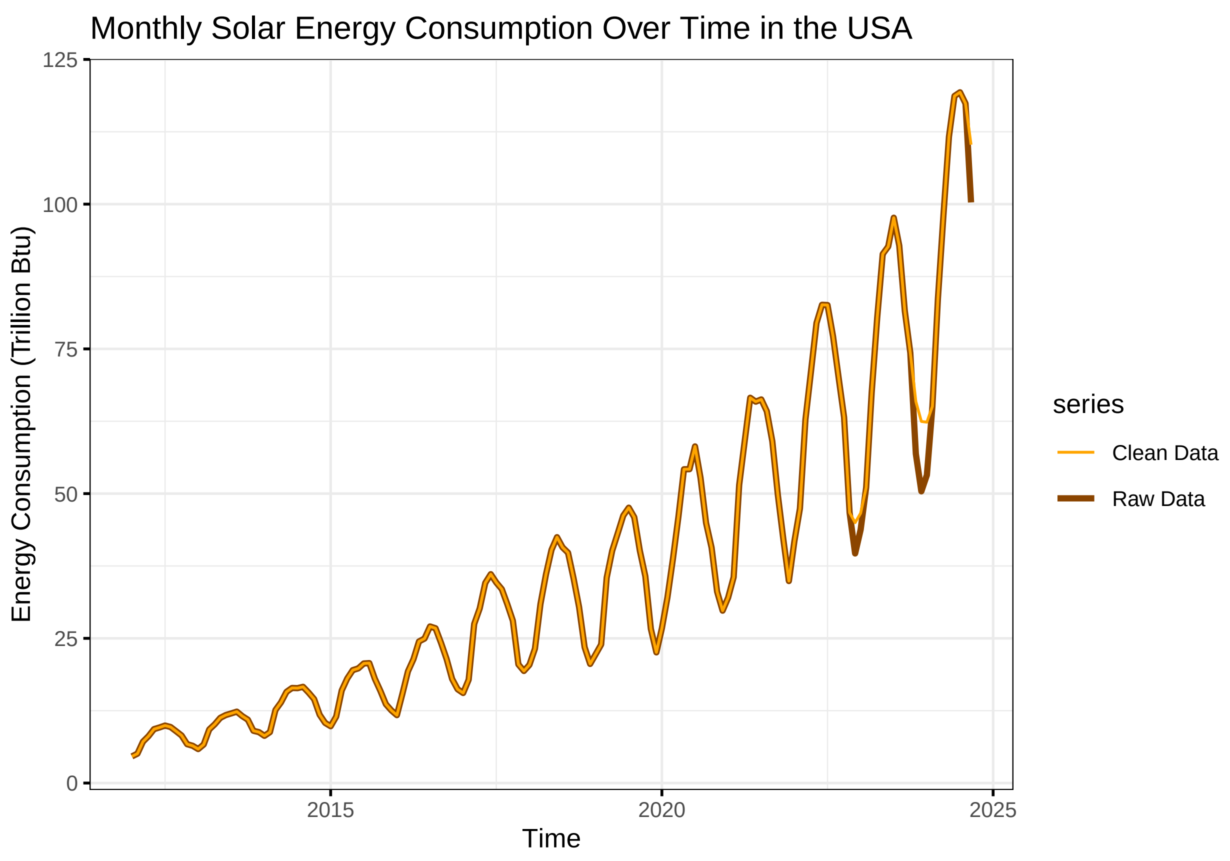
Redo number Q8 but now with the time series you created on Q7, i.e., the series starting in 2012. Using what autoplot() again what happened now? Did the function removed any outliers from the series?

```r
#clean the timeseries
wind_ts_recent_clean <- tsclean(energy_ts_recent[,2])
solar_ts_recent_clean <- tsclean(energy_ts_recent[,1])

#plot the series against the original
autoplot(energy_ts_recent[,2], series = "Raw Data", size = 1.1)+
  autolayer(wind_ts_recent_clean, series = "Clean Data")+
  labs(title = "Monthly Wind Energy Consumption Over Time in the USA",
       y = "Energy Consumption (Trillion Btu)")+
  scale_color_manual(values = c("Raw Data" = "darkgreen", "Clean Data" = "chartreuse2"))
```



```r
autoplot(energy_ts_recent[,1], series = "Raw Data", size = 1.1)+
  autolayer(solar_ts_recent_clean, series = "Clean Data")+
  labs(title = "Monthly Solar Energy Consumption Over Time in the USA",
       y = "Energy Consumption (Trillion Btu)")+
  scale_color_manual(values = c("Raw Data" = "darkorange4", "Clean Data" = "orange"))
```



The `ts_clean` function removed very few outliers from either series, and retains much more of the time series compared to the previous question. In the solar data, the two most recent troughs were removed. In the wind series, the second most recent trough was removed.