ENV 797 - Time Series Analysis for Energy and Environment Applications | Spring 2026

Assignment 5 - Due date 02/17/26

Student Name

Directions

You should open the .rmd file corresponding to this assignment on RStudio. The file is available on our class repository on Github.

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_Sp26.Rmd"). Then change "Student Name" on line 3 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 Canvas.

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)
```

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

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

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

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

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

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr   1.1.4     v stringr 1.5.1
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.2     v tidyr   1.3.1
## v readr   2.1.5

## -- 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(readxl)
```

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 2025 Monthly Energy Review.

```r
#Importing data set - using readxl package
energy_data <- read_excel(
  path="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx",
  skip = 12,
  sheet="Monthly Data",
  col_names=FALSE
  )
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
```

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

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
```

```
colnames(energy_data) <- read_col_names
nobs <- nrow(energy_data)

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

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>, ...
```

Handling Missing Data

Q1

Using the original dataset, create a new data frame that includes only the following variables: **Date, Solar Energy Consumption and Wind Energy Consumption**. Check the class of columns, you will see that they are stored are characters instead of numbers. Because solar generation begins later in the sample, the early observations are recorded as "Not Available". Convert the data to numeric, the "Not Available" will became NAs.

You may either filter out the "Not Available" rows and then convert the column to numeric or convert first and then remove missing values using drop_na() (or na.omit()). If you are comfortable using pipes for data wrangling, please do so.

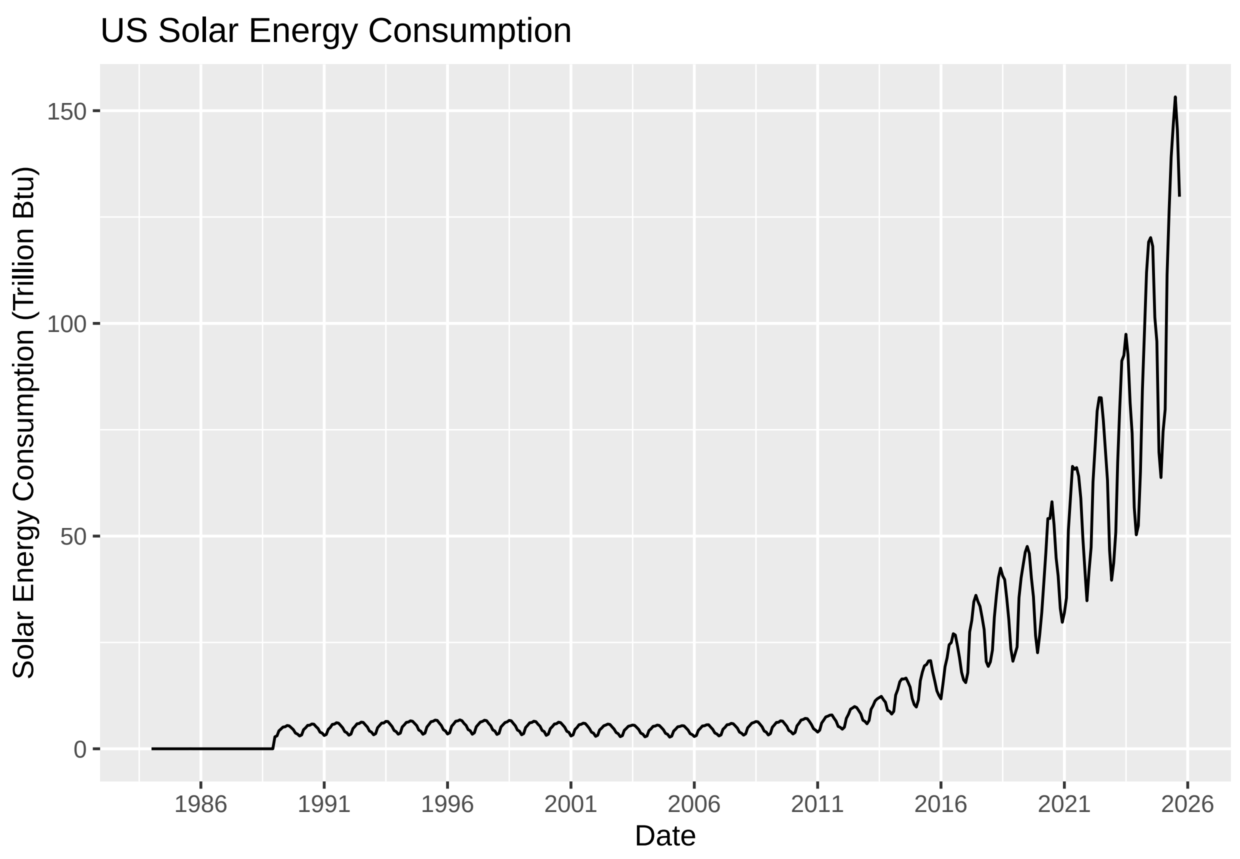
Important: Note that we dropping the missing observations instead of interpolating is becasue they only happen in the beginning of the series!

```r
solar_wind <- energy_data %>%
  select("Month","Solar Energy Consumption","Wind Energy Consumption") %>%
  filter(`Solar Energy Consumption` != "Not Available")

solar_wind$`Solar Energy Consumption`<-as.numeric(solar_wind$`Solar Energy Consumption`)
solar_wind$`Wind Energy Consumption`<-as.numeric(solar_wind$`Wind Energy Consumption`)
solar_wind$Month <- as.Date(ymd(solar_wind$Month))
```

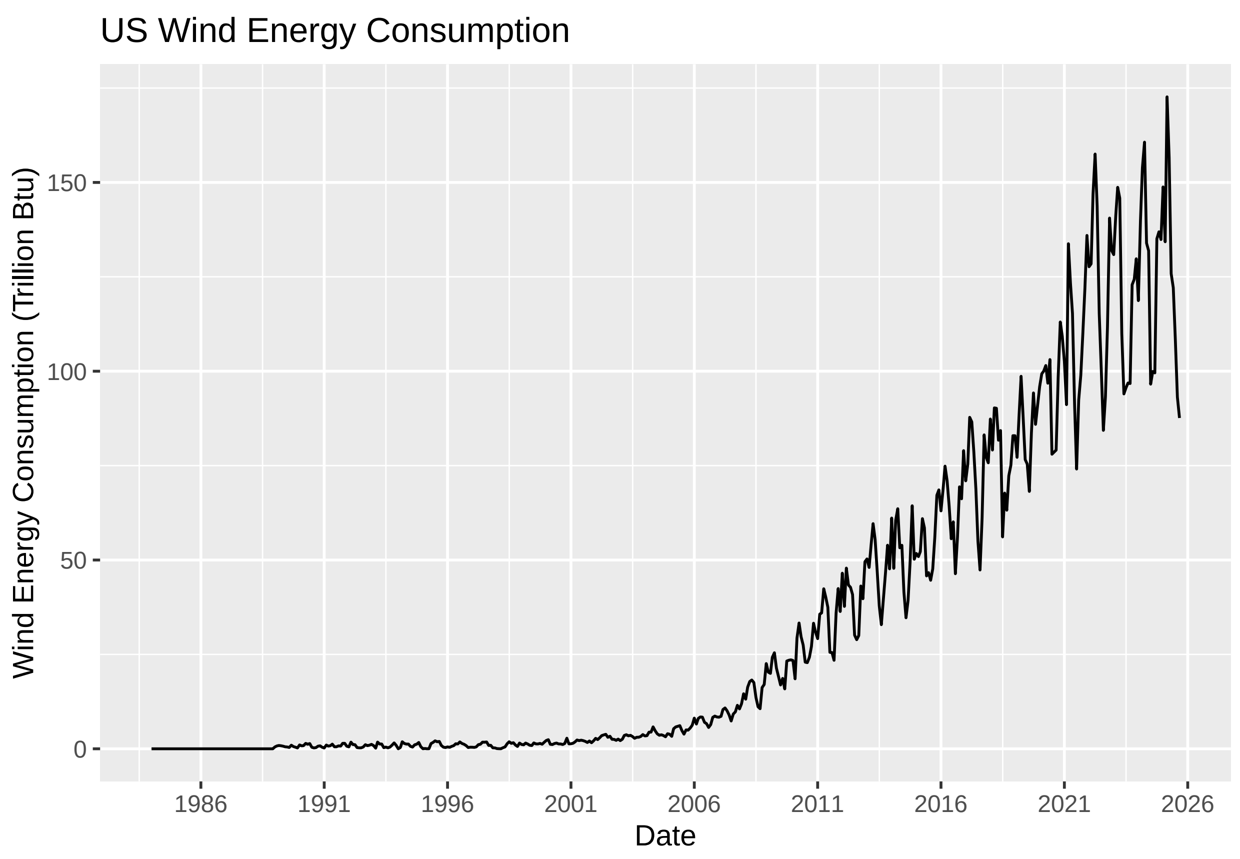
Q2

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_raw <- ggplot(solar_wind, aes(x = Month, y = `Solar Energy Consumption`))+
  geom_line()+
  xlab("Date")+
  ggtitle("US Solar Energy Consumption")+
  ylab("Solar Energy Consumption (Trillion Btu)")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")

solar_raw
```



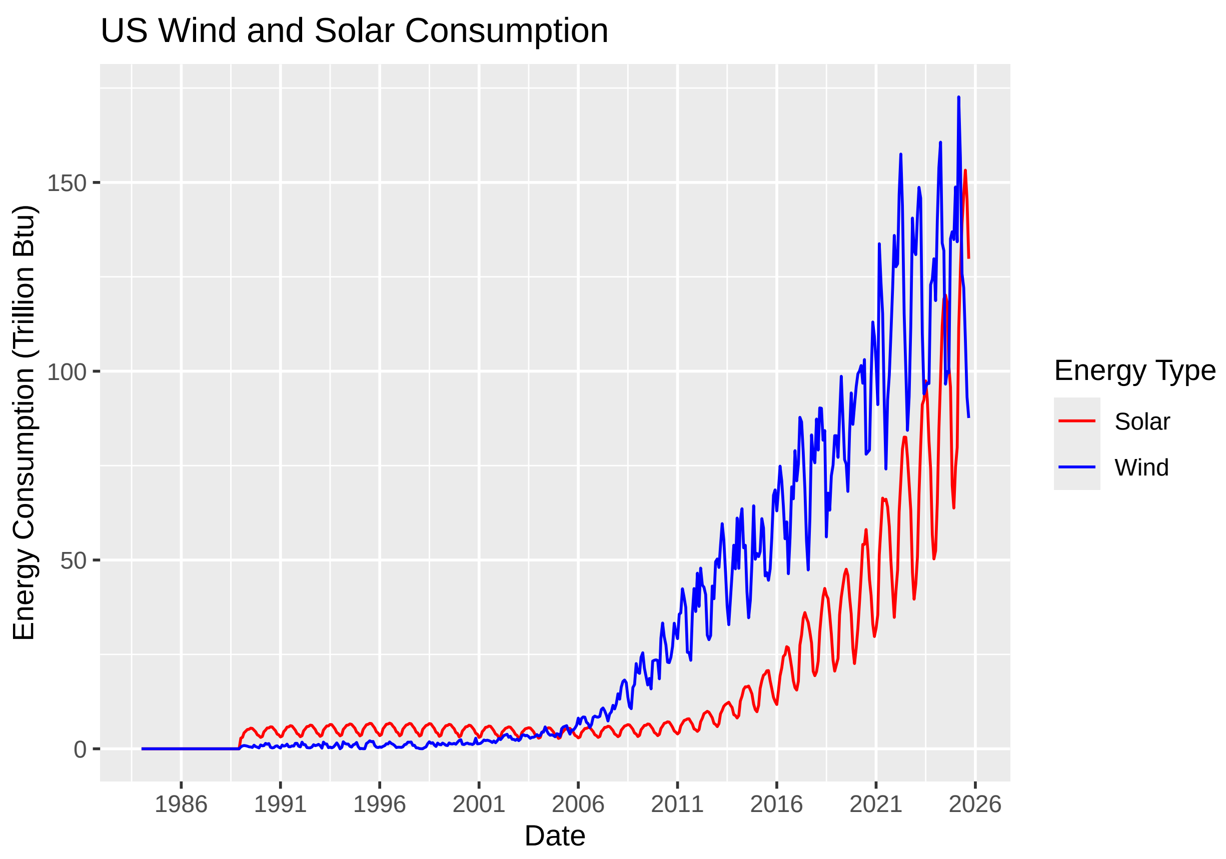
```r
wind_raw <- ggplot(solar_wind, aes(x = Month, y = `Wind Energy Consumption`))+
  geom_line()+
  xlab("Date")+
  ggtitle("US Wind Energy Consumption")+
  ylab("Wind Energy Consumption (Trillion Btu)")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")

wind_raw
```



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
solar_wind_plot <- ggplot(solar_wind, aes(x = Month))+
  geom_line(aes( y = `Solar Energy Consumption`,color = "Solar"))+
  geom_line(aes( y = `Wind Energy Consumption`,color = "Wind"))+
  scale_color_manual(values = c(
    "Solar" = "red",
    "Wind" = "blue"))+
  labs(title = "US Wind and Solar Consumption",color = "Energy Type")+
  ylab("Energy Consumption (Trillion Btu)")+
  xlab("Date")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")


solar_wind_plot
```



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.

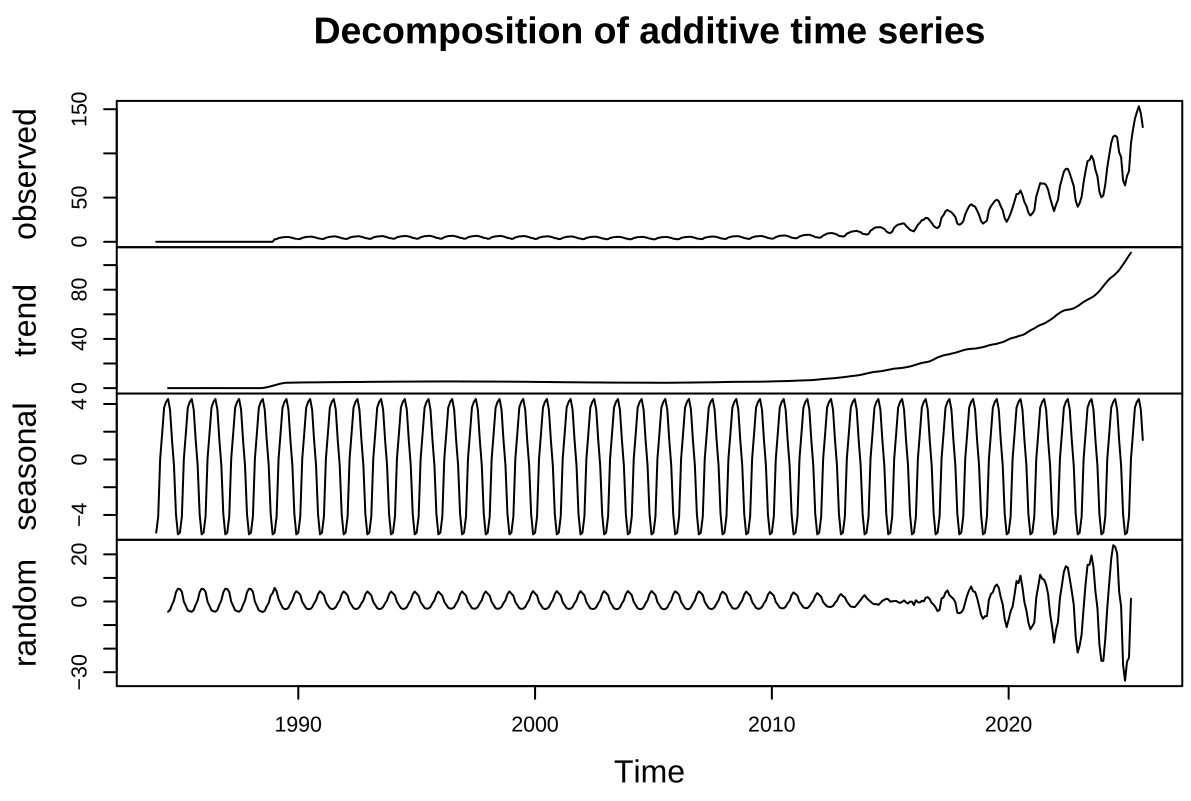
Additional info on `decompose()`.

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.

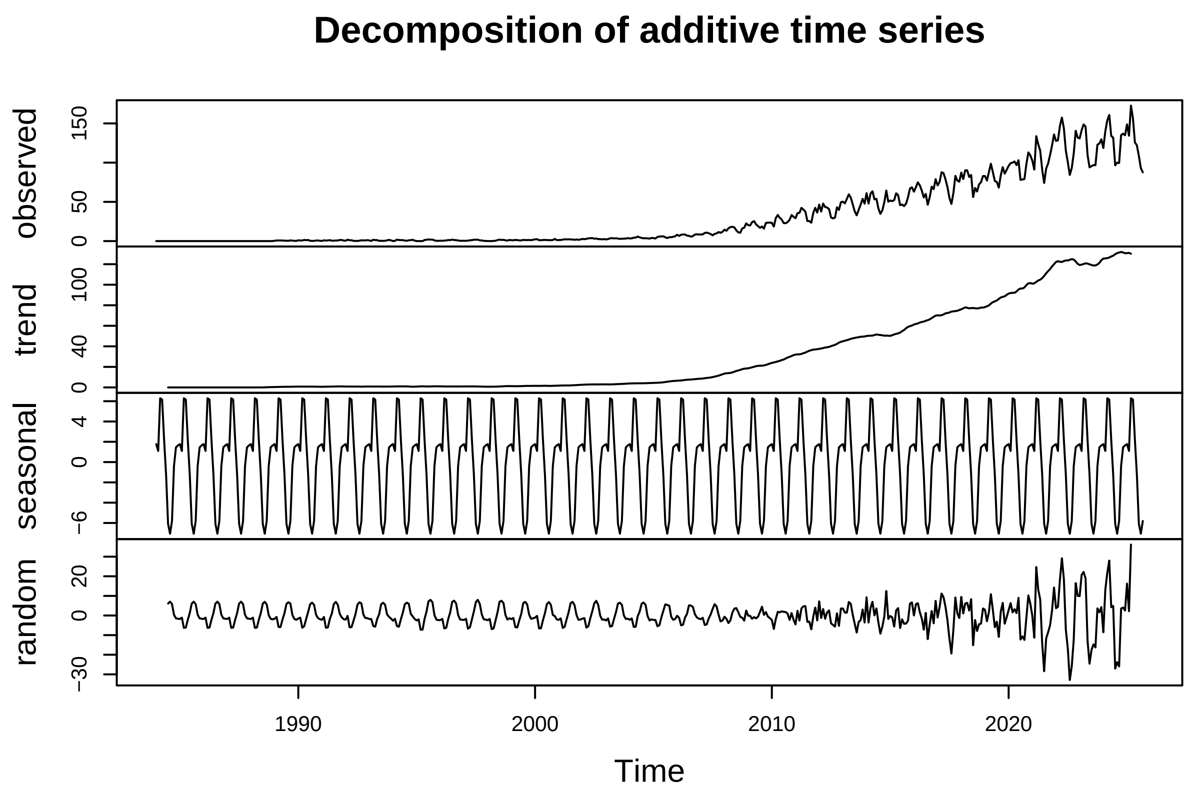
Q4

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_solar<- ts(solar_wind[,2],
                     start=c(1984,1),
                     frequency=12)

ts_wind<- ts(solar_wind[,3],
                     start=c(1984,1),
                     frequency=12)

solar_decompose_a <- decompose(ts_solar, "additive")
plot(solar_decompose_a)
```



```
wind_decompose_a <- decompose(ts_wind, "additive")
plot(wind_decompose_a)
```

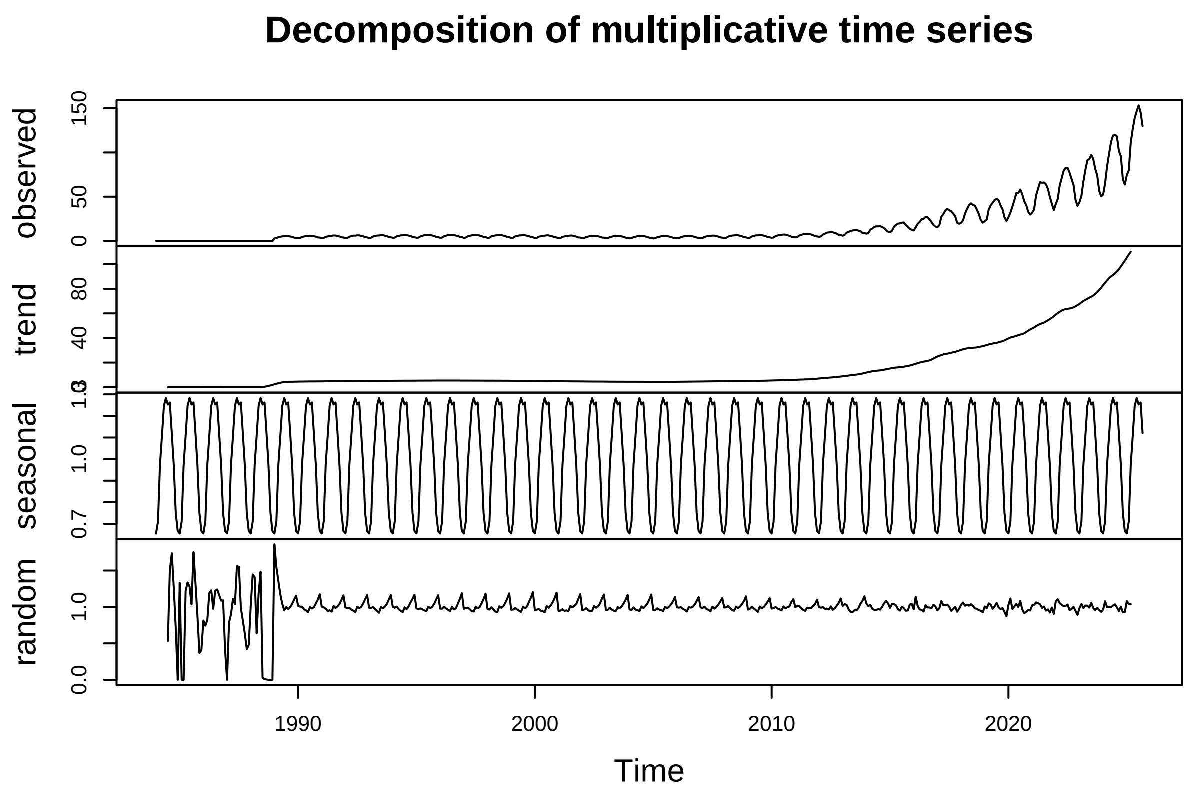


Both series seems to have a very clear upward trend component. While the random component is definitely centered around 0 in both series, there seems to be regular intervals of increasing and decreasing, though the peaks and troughs vary. In the solar data, this regularity seems very consistent, with the peaks and troughs steadily increasing in the last 10 years. In the wind data, there are very regular cycles until around 2008 and then while there is still some cycling it is less symmetrical. There may still be some seasonality in the random component

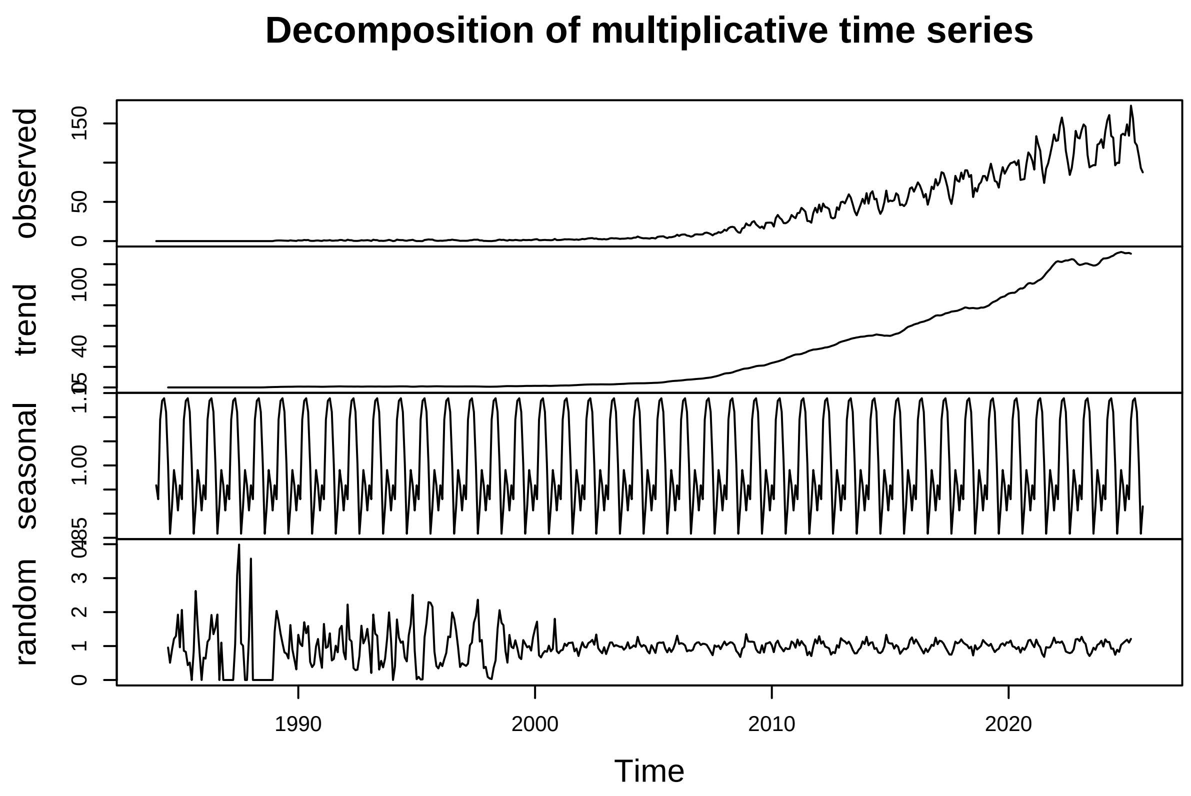
Q5

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
solar_decompose_m <- decompose(ts_solar, "multiplicative")
plot(solar_decompose_m)
```



```r
wind_decompose_m<- decompose(ts_wind, "multiplicative")
plot(wind_decompose_m)
```



The random component with the multiplicative method seems more random than the additive method. In the solar series, there still seem to be some symmetrical cycles between 1990 and 2010, but much less obvious than the additive method. The wind series random component still varies up and down but there does not seem to be any sort of regular pattern left.

Q6

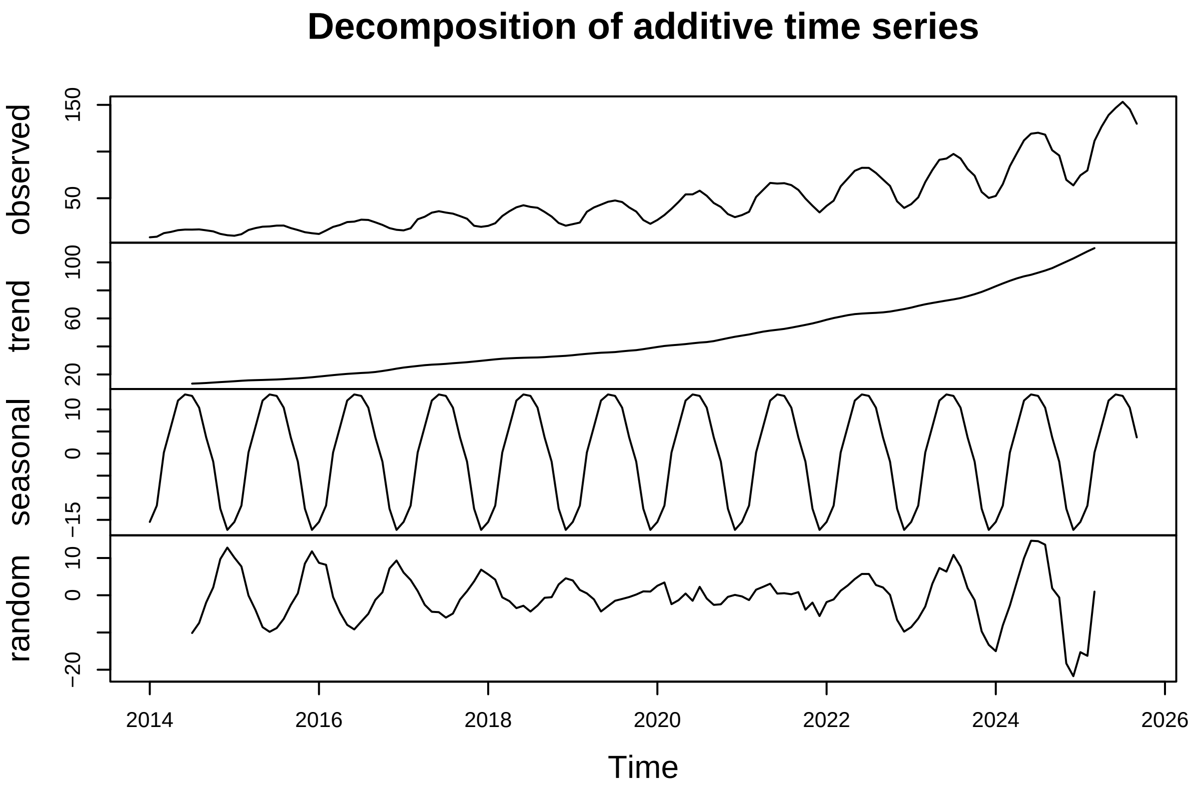
When fitting a model to this data, do you think you need all the historical data? Think about the data from 80s, 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: The seasonal component of these earlier data points might be helpful since they likely depend a lot on weather, which should (ignoring climate change) have similar patterns in the future. However, I would imagine that the industry trends are very different now than compared to the early 2000s and before. Likely looking at the more recent overall trend pattern in the last 10 years will be more helpful for predicting the future than the trend before then.

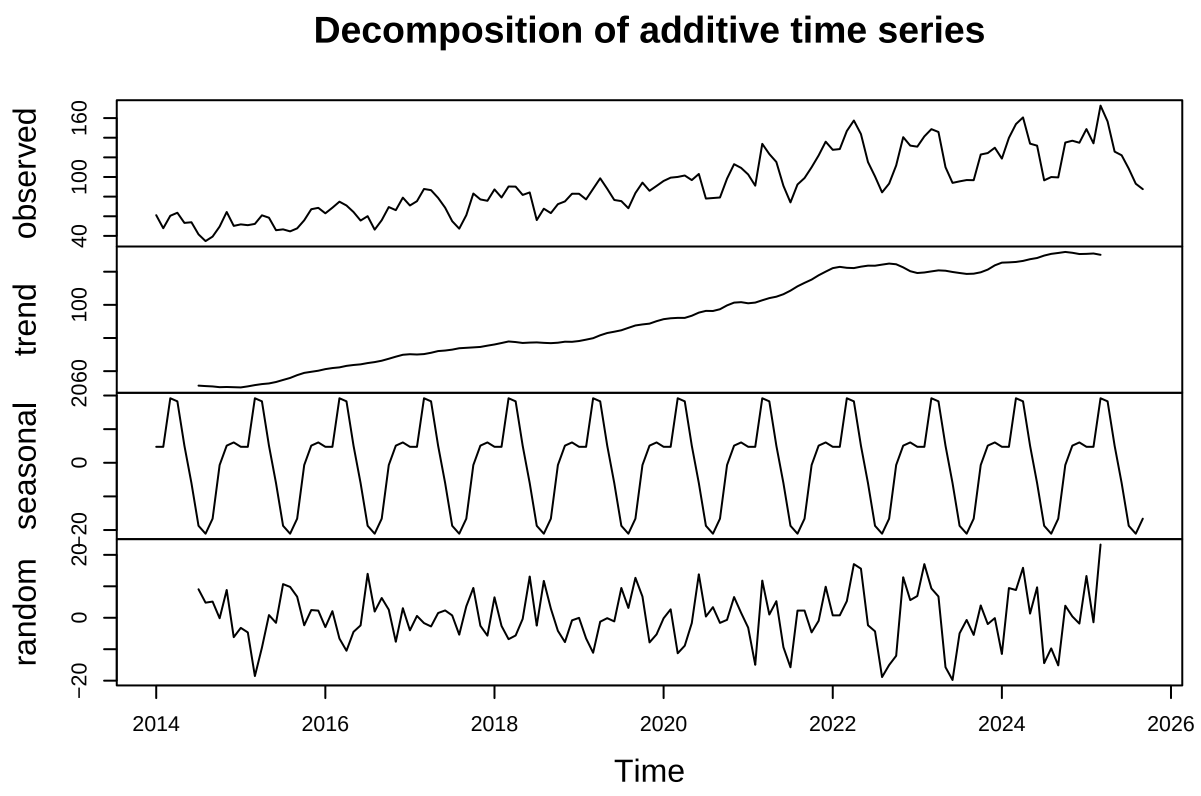
Q7

Create a new time series object where historical data starts on January 2014. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2014 )`. Apply the decompose function `type=additive` to this new time series. Comment on the results. Does the random component look random?

```r
recent_solarwind <- solar_wind %>%
  filter(year(Month)>=2014)

ts_solar_recent<- ts(recent_solarwind[,2],
                     start=c(2014,1),
                     frequency=12)
ts_wind_recent<- ts(recent_solarwind[,3],
                     start=c(2014,1),
                     frequency=12)

solar_decomp_recent <- decompose (ts_solar_recent, "additive")
plot(solar_decomp_recent)
```



```r
wind_decomp_recent <- decompose (ts_wind_recent, "additive")
plot(wind_decomp_recent)
```



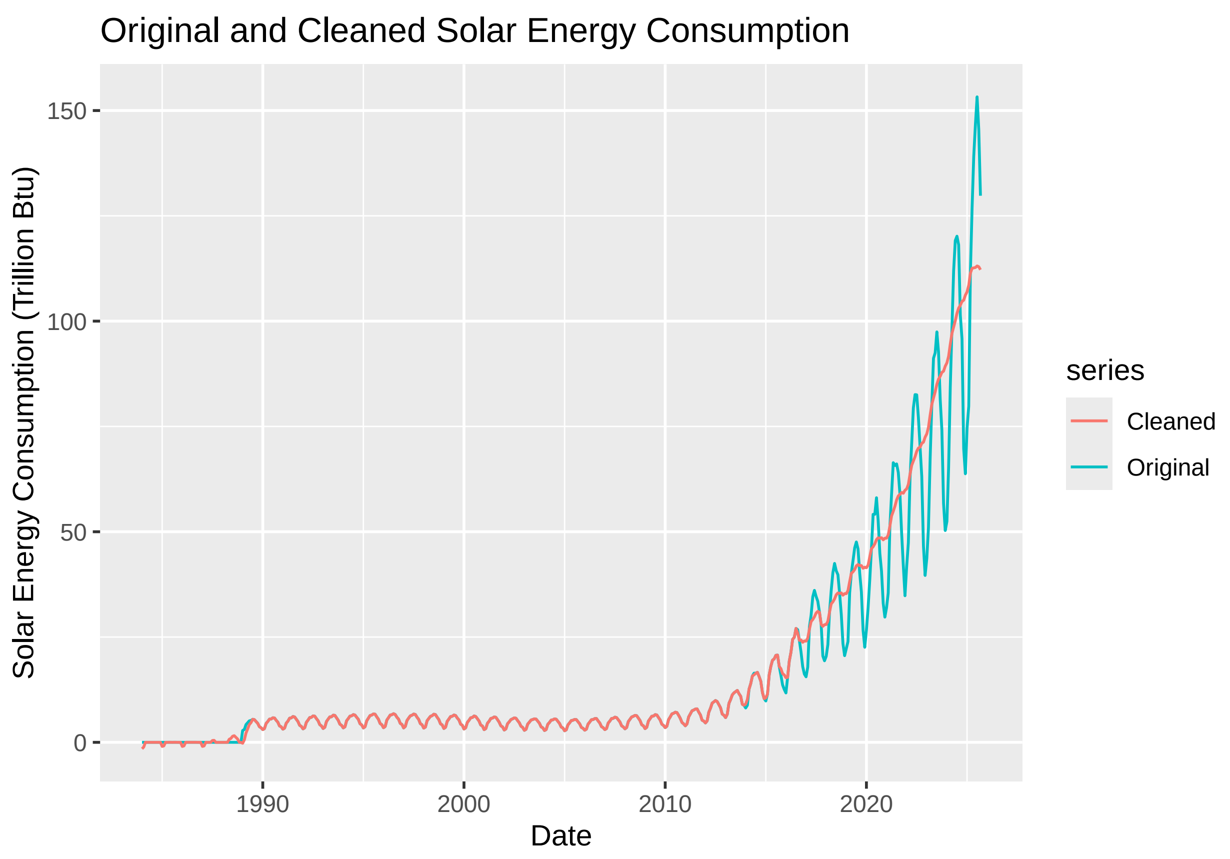
Answer:The random component looks much more random. There don't seem to be any periods in either series of regular patterns or cycles. There still seems to be a significant trend component in both series

Identify and Remove outliers

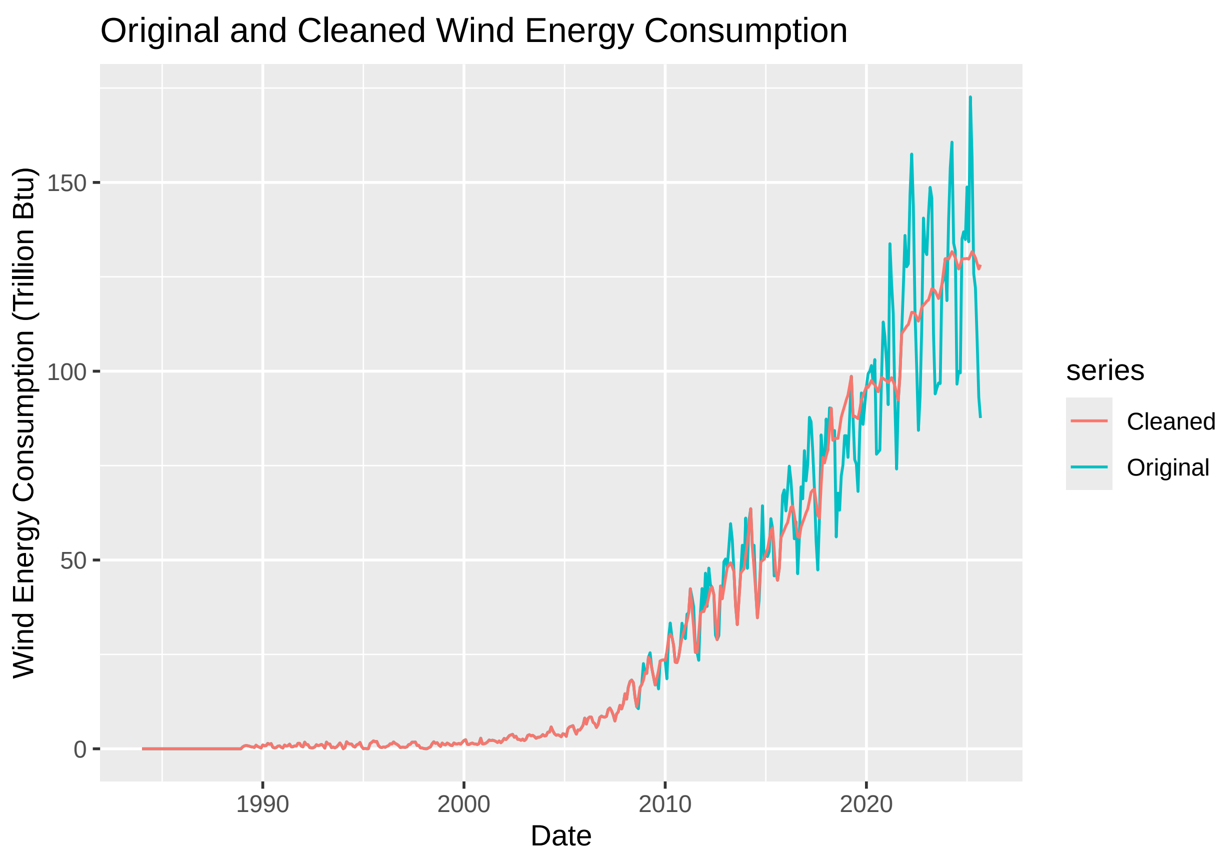
Q8

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.

```
clean_solar <- tsclean(ts_solar)
clean_wind <- tsclean(ts_wind)

solar_comparison<- autoplot(ts_solar, series = "Original")+
  autolayer(clean_solar, series = "Cleaned")+
  xlab("Date")+
  ylab("Solar Energy Consumption (Trillion Btu)")+
  ggtitle("Original and Cleaned Solar Energy Consumption")
solar_comparison
```



```r
wind_comparison<- autoplot(ts_wind, series = "Original")+
  autolayer(clean_wind, series = "Cleaned")+
  xlab("Date")+
  ylab("Wind Energy Consumption (Trillion Btu)")+
  ggtitle("Original and Cleaned Wind Energy Consumption")
wind_comparison
```

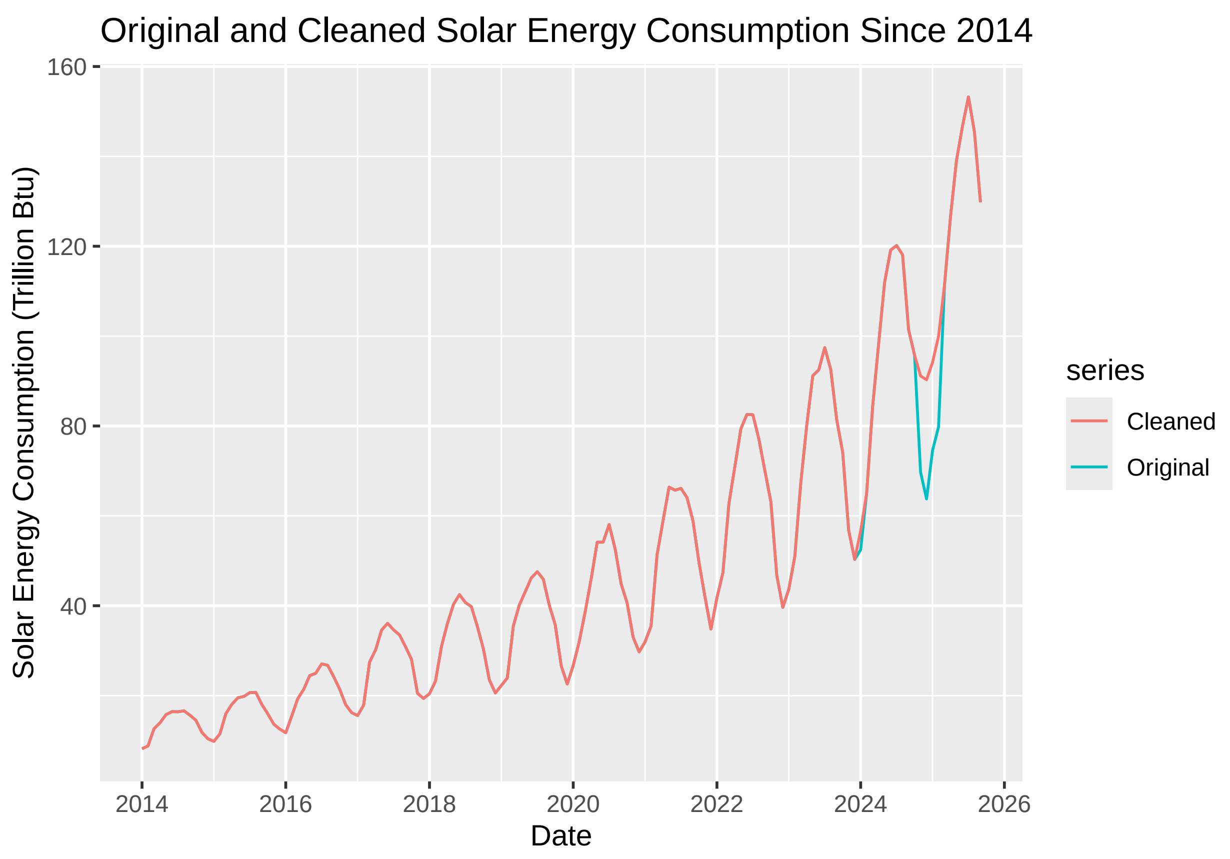


Both series removed a lot of outliers, particularly in the years since 2015. The cleaned lines are far more linear than the original lines which oscillate up and down significantly.

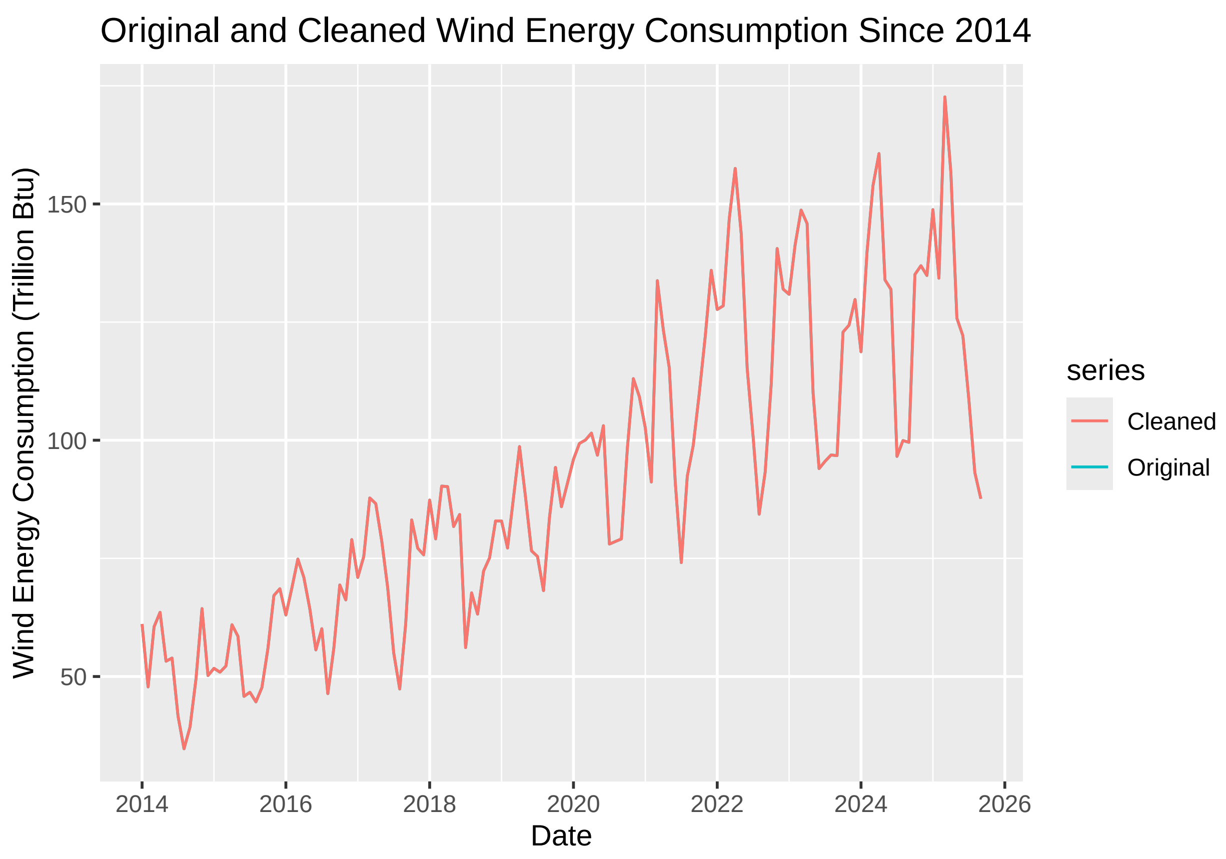
Q9

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?

```
clean_solar_recent <- tsclean(ts_solar_recent)
clean_wind_recent <- tsclean(ts_wind_recent)

solar_comparison_recent<- autoplot(ts_solar_recent, series = "Original")+
  autolayer(clean_solar_recent, series = "Cleaned")+
  xlab("Date")+
  ylab("Solar Energy Consumption (Trillion Btu)")+
  ggtitle("Original and Cleaned Solar Energy Consumption Since 2014")
solar_comparison_recent
```



```
wind_comparison_recent<- autoplot(ts_wind_recent, series = "Original")+
  autolayer(clean_wind_recent, series = "Cleaned")+
  xlab("Date")+
  ylab("Wind Energy Consumption (Trillion Btu)")+
  ggtitle("Original and Cleaned Wind Energy Consumption Since 2014")
wind_comparison_recent
```



Answer: Using data only since 2014, the tsclean() function did not remove as many outliers. In the wind data, by looking at the comparison plot, there doesn't seem to be any outliers that were removed. In the solar data, there were some outliers removed in 2025 but otherwise the cleaned and original series look the same.