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

Assignment 5 - Due date 02/17/26

Joy Wu

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.\

```
#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
```

```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 errors
```

```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!

```
energy.solar.wind <- energy_data %>%
  select(Month, `Solar Energy Consumption`, `Wind Energy Consumption`) %>%
  mutate(
    `Solar Energy Consumption` = as.numeric(`Solar Energy Consumption`),
    `Wind Energy Consumption` = as.numeric(`Wind Energy Consumption`)
  ) %>%
  na.omit()
```

```
## Warning: There were 2 warnings in ‘mutate()‘.
## The first warning was:
## i In argument: ‘Solar Energy Consumption = as.numeric(‘Solar Energy
##   Consumption‘)‘.
## Caused by warning:
## ! NAs introduced by coercion
## i Run ‘dplyr::last_dplyr_warnings()‘ to see the 1 remaining warning.
```

```
summary(energy.solar.wind)
```

```
##      Month                         Solar Energy Consumption
##  Min.   :1984-01-01 00:00:00.00   Min.   :  0.000
##  1st Qu.:1994-06-01 00:00:00.00   1st Qu.:  3.924
##  Median :2004-11-01 00:00:00.00   Median :  5.658
##  Mean   :2004-10-31 01:26:13.65   Mean   : 16.623
##  3rd Qu.:2015-04-01 00:00:00.00   3rd Qu.: 15.758
##  Max.   :2025-09-01 00:00:00.00   Max.   :153.256
##  Wind Energy Consumption
##  Min.   :  0.000
##  1st Qu.:  0.844
##  Median :  3.999
##  Mean   : 31.305
##  3rd Qu.: 55.001
##  Max.   :172.670
```

Q2

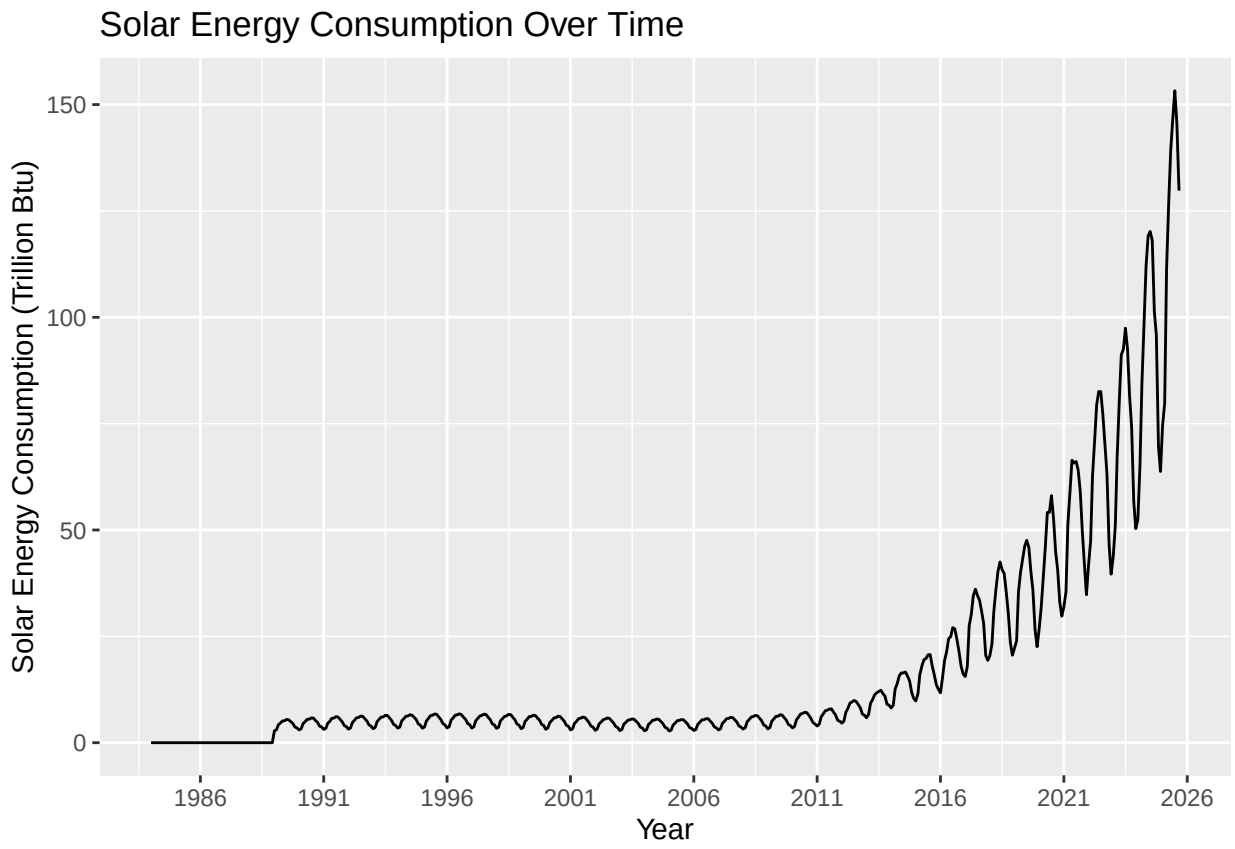
Plot the Solar and Wind energy consumption over time using ggplot. Plot each series on a separate graph. No need to add legend. Add informative names to the y axis using `ylab()`. Explore the function `scale_x_date()` on ggplot and see if you can change the x axis to improve your plot. Hint: use `scale_x_date(date_breaks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

```
energy.solar.wind$Month <- as.Date(energy.solar.wind$Month)
class(energy.solar.wind$Month)
```

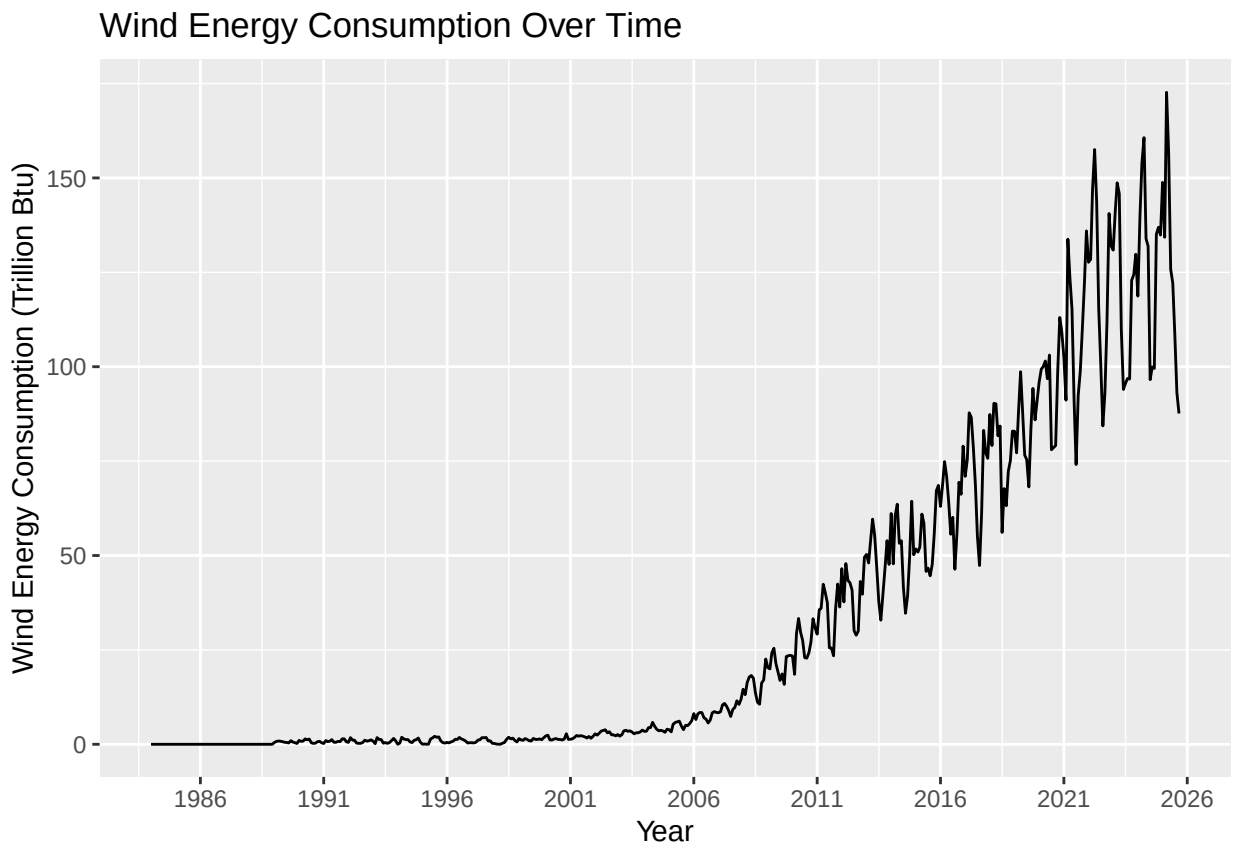
```
## [1] "Date"
```

```
solar.plot <- ggplot(energy.solar.wind, aes(x = Month, y = `Solar Energy Consumption`)) +
  geom_line() +
  ylab("Solar Energy Consumption (Trillion Btu)") +
```

```
  xlab("Year") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ggtitle("Solar Energy Consumption Over Time")
solar.plot
```



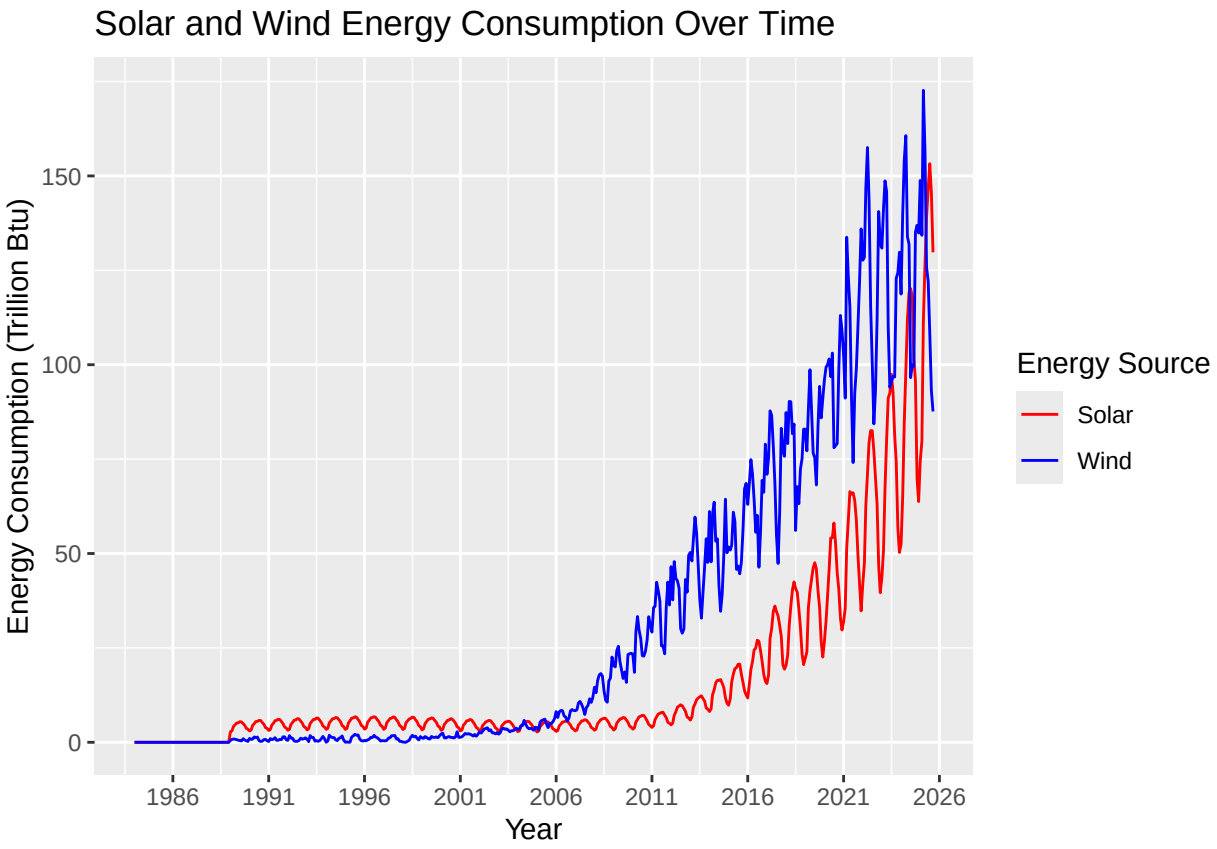

```
wind.plot <- ggplot(energy.solar.wind, aes(x = Month, y = `Wind Energy Consumption`)) +
  geom_line() +
  ylab("Wind Energy Consumption (Trillion Btu)") +
  xlab("Year") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ggtitle("Wind Energy Consumption Over Time")
wind.plot
```



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.

```
solar.wind.plot <- ggplot(energy.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"),
    name = "Energy Source"
  ) +
  ylab("Energy Consumption (Trillion Btu)") +
  xlab("Year") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ggtitle("Solar and Wind Energy Consumption Over Time")
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.

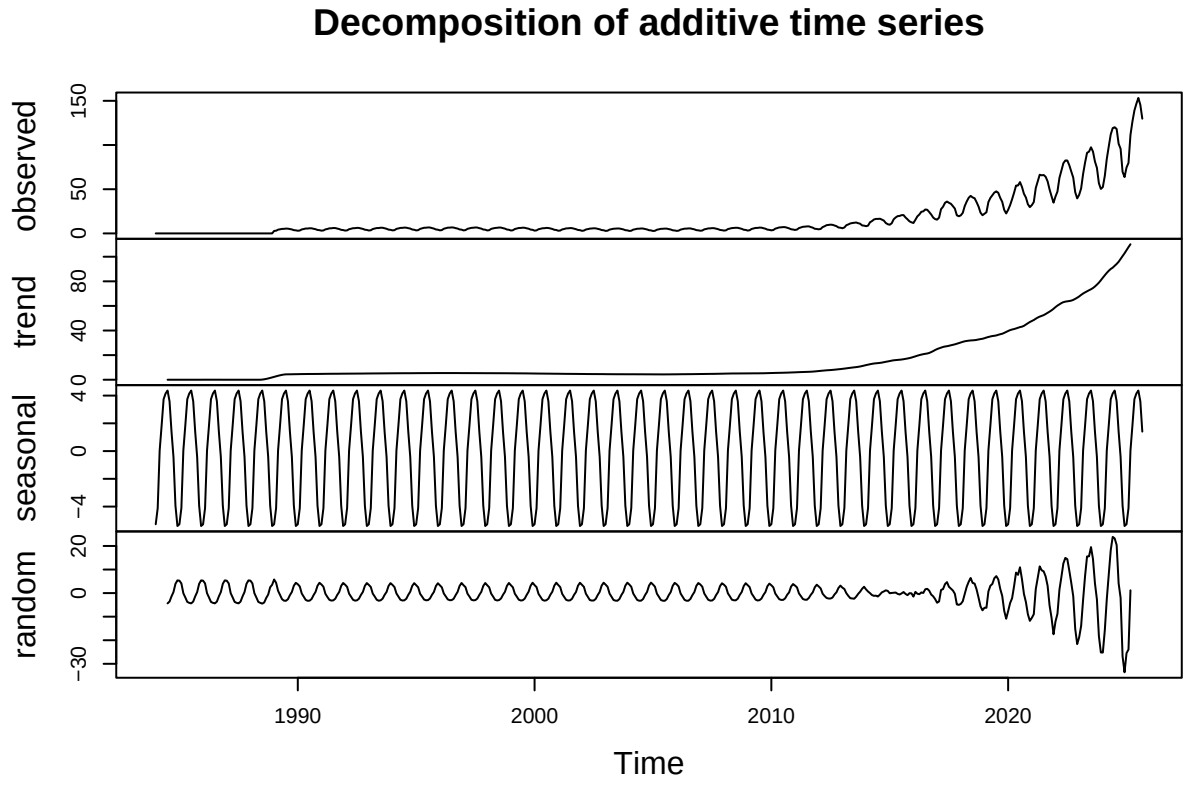
Additional info on `decompose()`.

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.

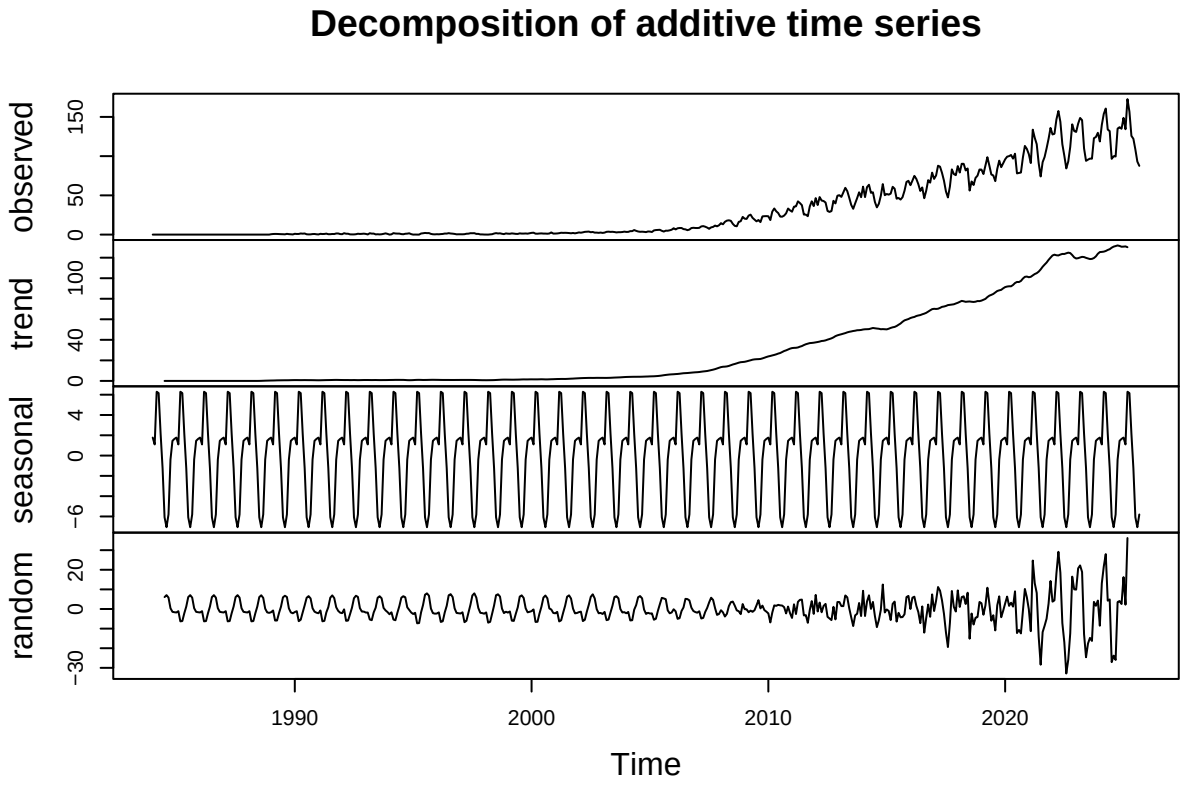
Q4

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?

```
solar.ts <- ts(energy.solar.wind$`Solar Energy Consumption`,
               frequency = 12,
               start = c(1984,1))
wind.ts <- ts(energy.solar.wind$`Wind Energy Consumption`,
               frequency = 12,
               start = c(1984,1))
solar.decompose.additive <- decompose(solar.ts, type = "additive")
wind.decompose.additive <- decompose(wind.ts, type = "additive")
plot(solar.decompose.additive)
```




```
plot(wind.decompose.additive)
```

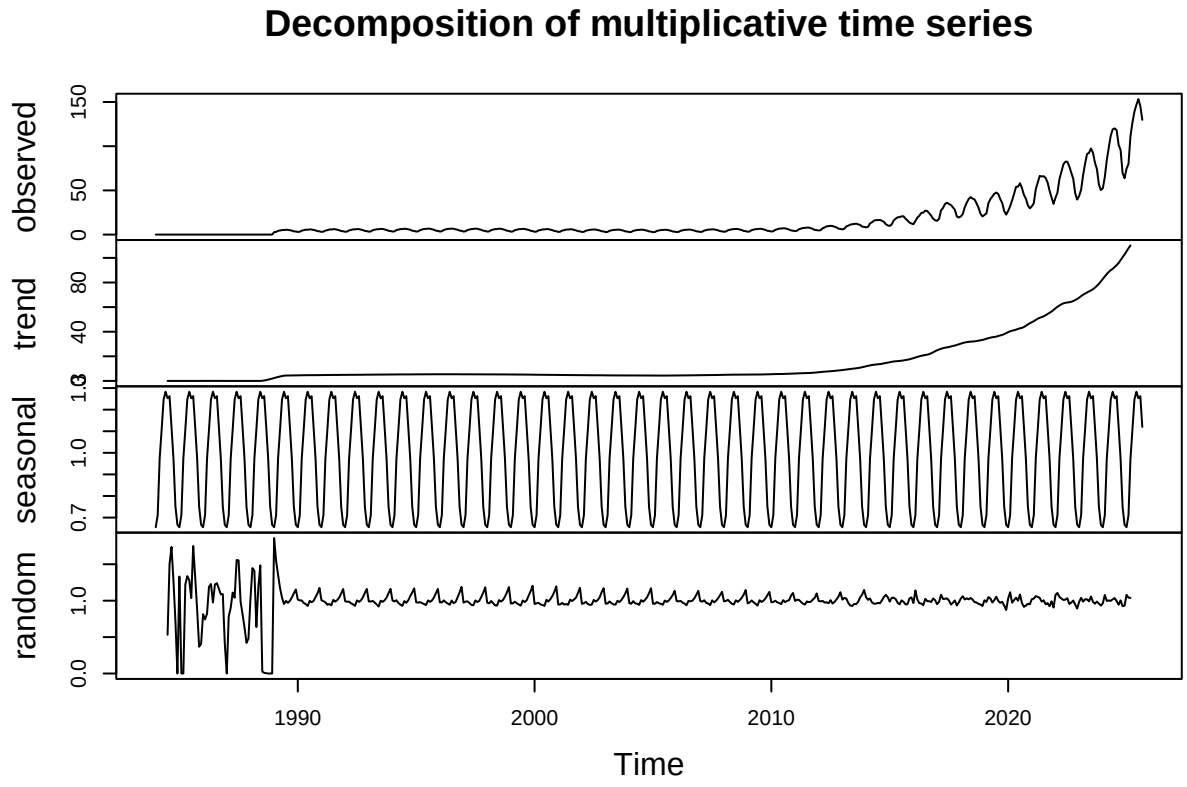


> Answer: Both solar and wind show a clear upward trend, especially in the later years after 2010s with a quick increase. The solar curve is more smooth with a dip at first, and the wind curve shows a more straight up increase. The random component for both solar and wind should show up as completely random, but from the plots, they still show some kind of seasonality patterns. This is especially true when the plot gets close to 2020s.

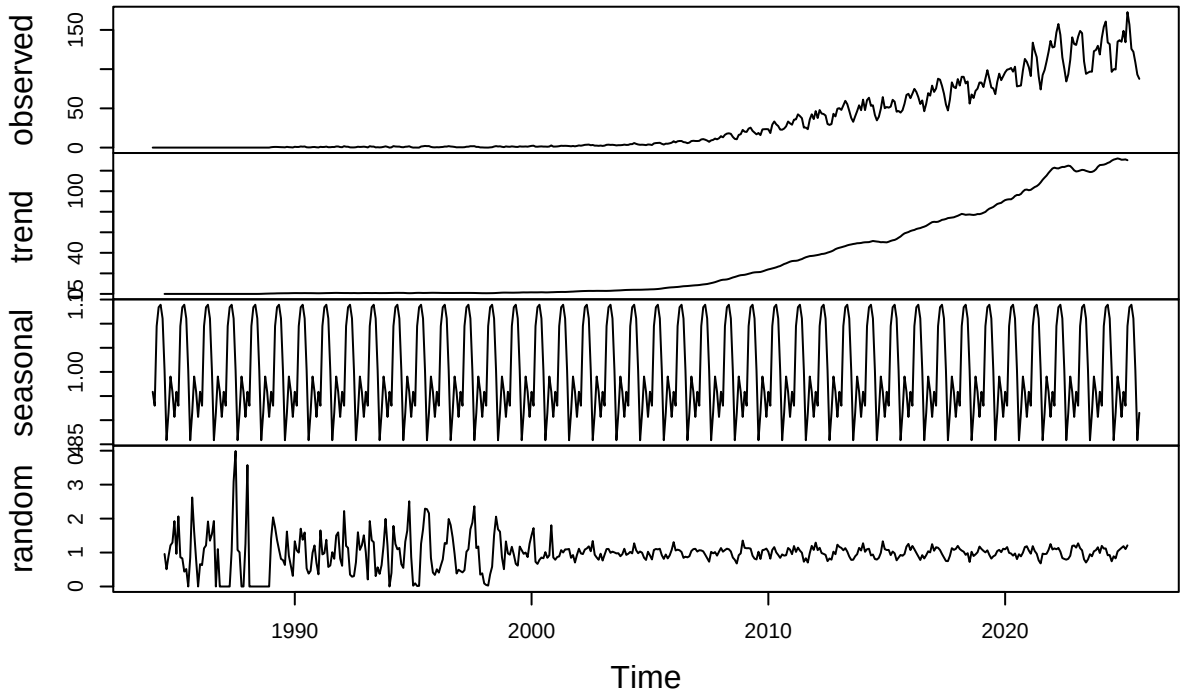
Q5

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?

```
solar.decompose.multiplicative <- decompose(solar.ts, type = "multiplicative")
wind.decompose.multiplicative <- decompose(wind.ts, type = "multiplicative")
plot(solar.decompose.multiplicative)
```




```
plot(wind.decompose.multiplicative)
```



> Answer: By changing the type from additive to multiplicative, the random component for both solar and wind look a lot more random than before. The plot flucutates around 1 rather than around 0 like the previous ones had, and the variability aren't incrasing with the series levels too.

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: To forecast the next six months of solar and/or wind consumption, I likely will not need all the historical data, especially the ones frok 80s and 90s. This needs to be put in context because there was not much solar production and consumption during that time frame as the expansion really only happened in mid 2000s. Although wind production goes back longer in history, it has a very different context in terms of policy and regualatory landscape, so things are different during that time frame and after the mid 2000s. Having data from that time frame would not be of much uses. In addition, if we want to get information on seasonal pattern and seasonality, we would only need info from the past few years rather than from decades ago since those are yearly patterns.

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?

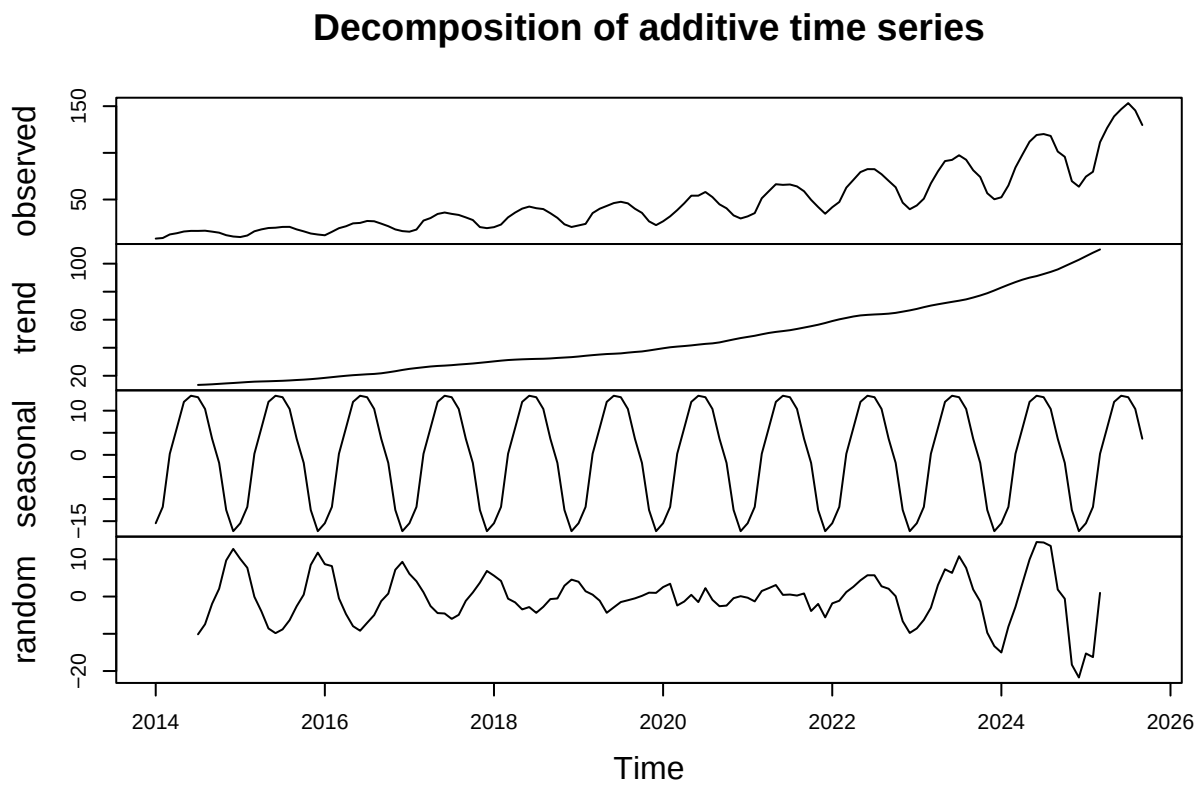
```r
energy.solar.wind.2014 <- energy.solar.wind %>%
  filter(year(Month) >= 2014)

head(energy.solar.wind.2014)
```

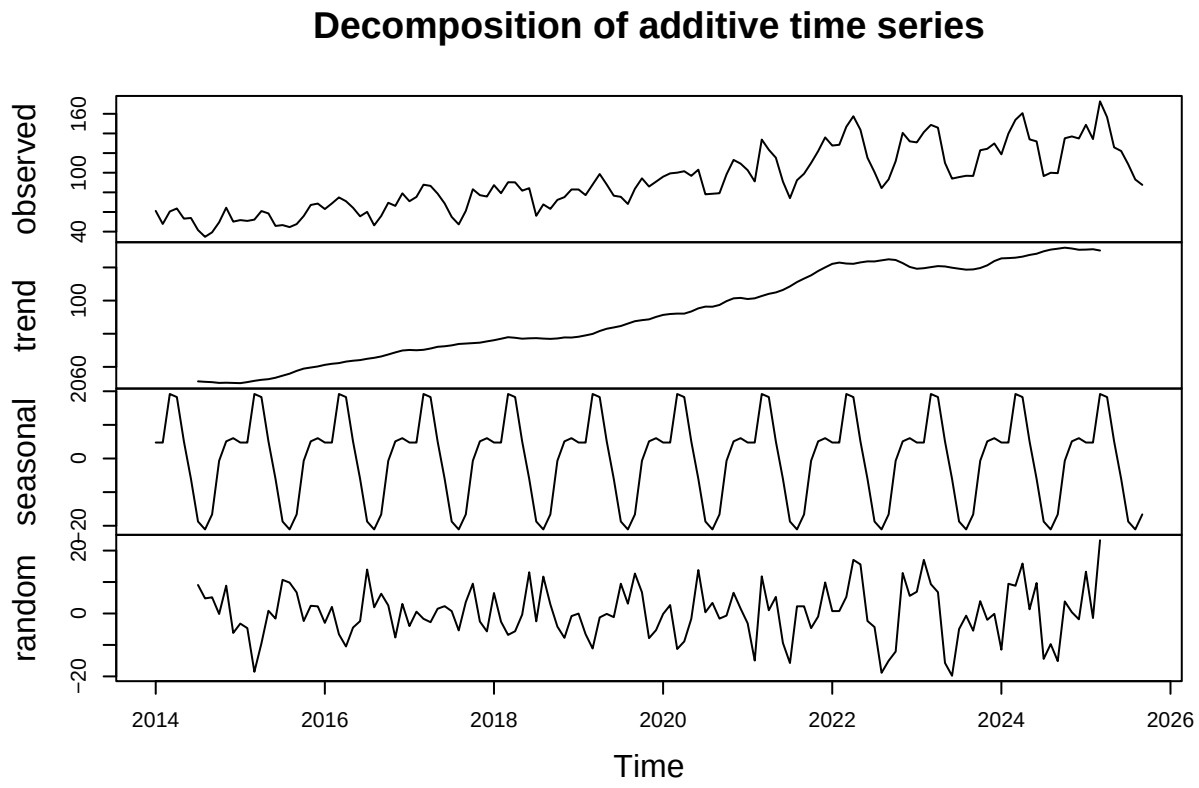
```
## # A tibble: 6 x 3
##   Month      `Solar Energy Consumption` `Wind Energy Consumption`
##   <date>                          <dbl>                     <dbl>
## 1 2014-01-01                       8.16                      61.1
## 2 2014-02-01                       8.80                      47.8
## 3 2014-03-01                      12.6                       60.5
## 4 2014-04-01                      13.9                       63.6
## 5 2014-05-01                      15.8                       53.2
## 6 2014-06-01                      16.4                       53.9
```

```r
solar.2014.ts <- ts(energy.solar.wind.2014$`Solar Energy Consumption`,
                    frequency = 12,
                    start = c(2014, 1))
wind.2014.ts <- ts(energy.solar.wind.2014$`Wind Energy Consumption`,
                    frequency = 12,
                    start = c(2014, 1))
solar.2014.decompose <- decompose(solar.2014.ts, type = "additive")
wind.2014.decompose <- decompose(wind.2014.ts, type = "additive")

plot(solar.2014.decompose)
```



```r
plot(wind.2014.decompose)
```



Answer: Since this time series only contains data post 2014, it is a more accurate representation of the recent solar and wind energy consumption. Looking at the trend, both solar and wind show much smoother and more linear upward trends compared to the full historical data set. The seasonal components for both series remain clear and stable. For the random component, solar seems to still have seasonality remaining, whereas wind shows less seasonality and is more random.

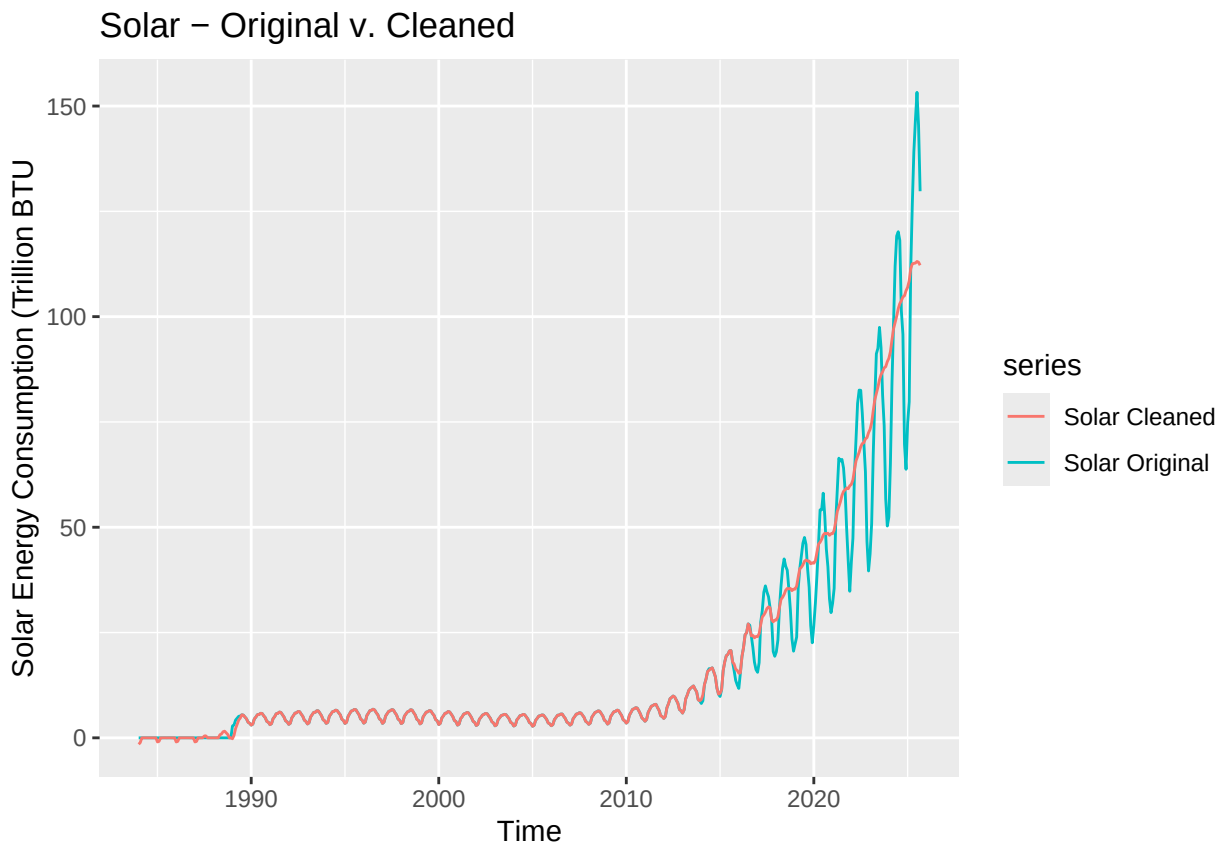
Identify and Remove outliers

Q8

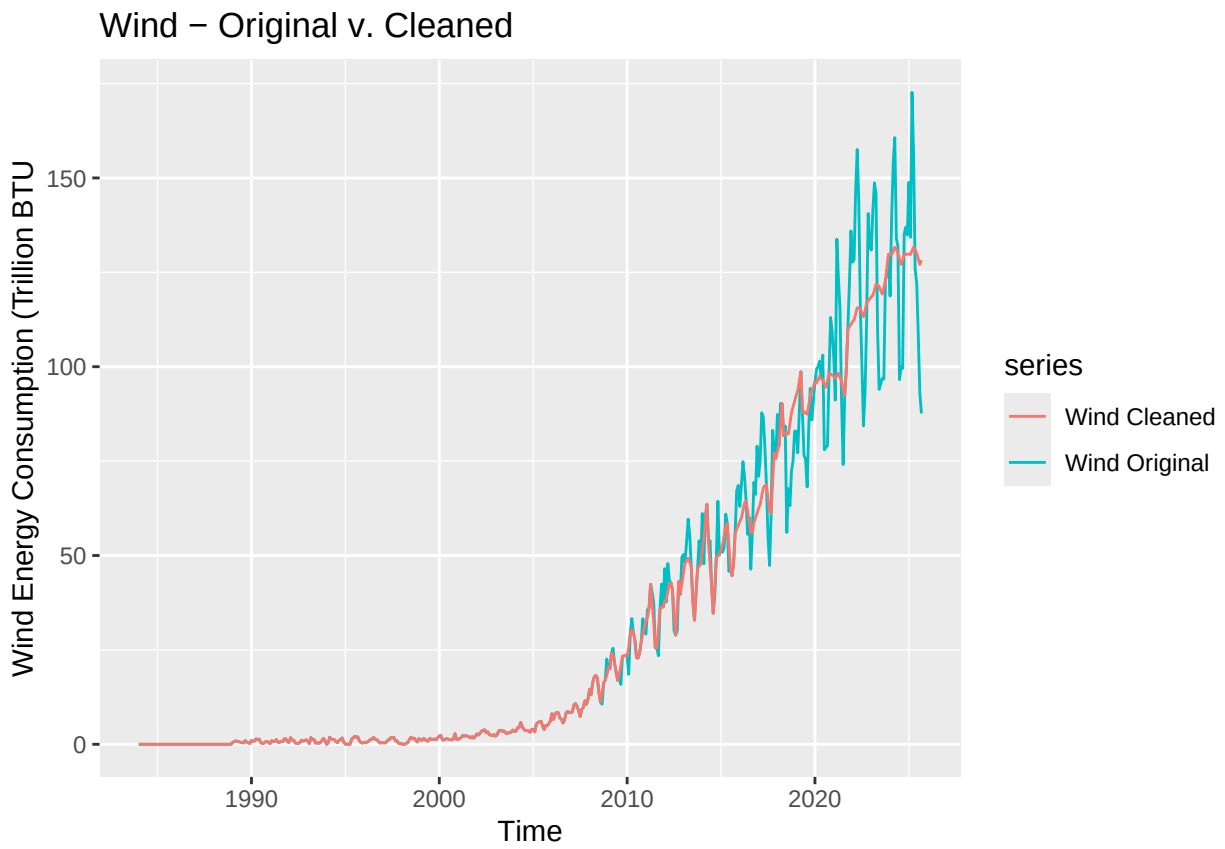
Apply the `tsclean()` to both time series object you created on Q4. Did the function removed any outliers from the series? Hint: Use `autoplot()` to check if there is difference between cleaned series and original series.

```r
solar.ts.clean <- tsclean(solar.ts)
wind.ts.clean <- tsclean(wind.ts)

autoplot(solar.ts, series = "Solar Original") +
  autolayer(solar.ts.clean, series = "Solar Cleaned") +
  ylab("Solar Energy Consumption (Trillion BTU") +
  ggtitle("Solar - Original v. Cleaned")
```




```
autoplot(wind.ts, series = "Wind Original") +
  autolayer(wind.ts.clean, series = "Wind Cleaned") +
  ylab("Wind Energy Consumption (Trillion BTU") +
  ggtitle("Wind - Original v. Cleaned")
```



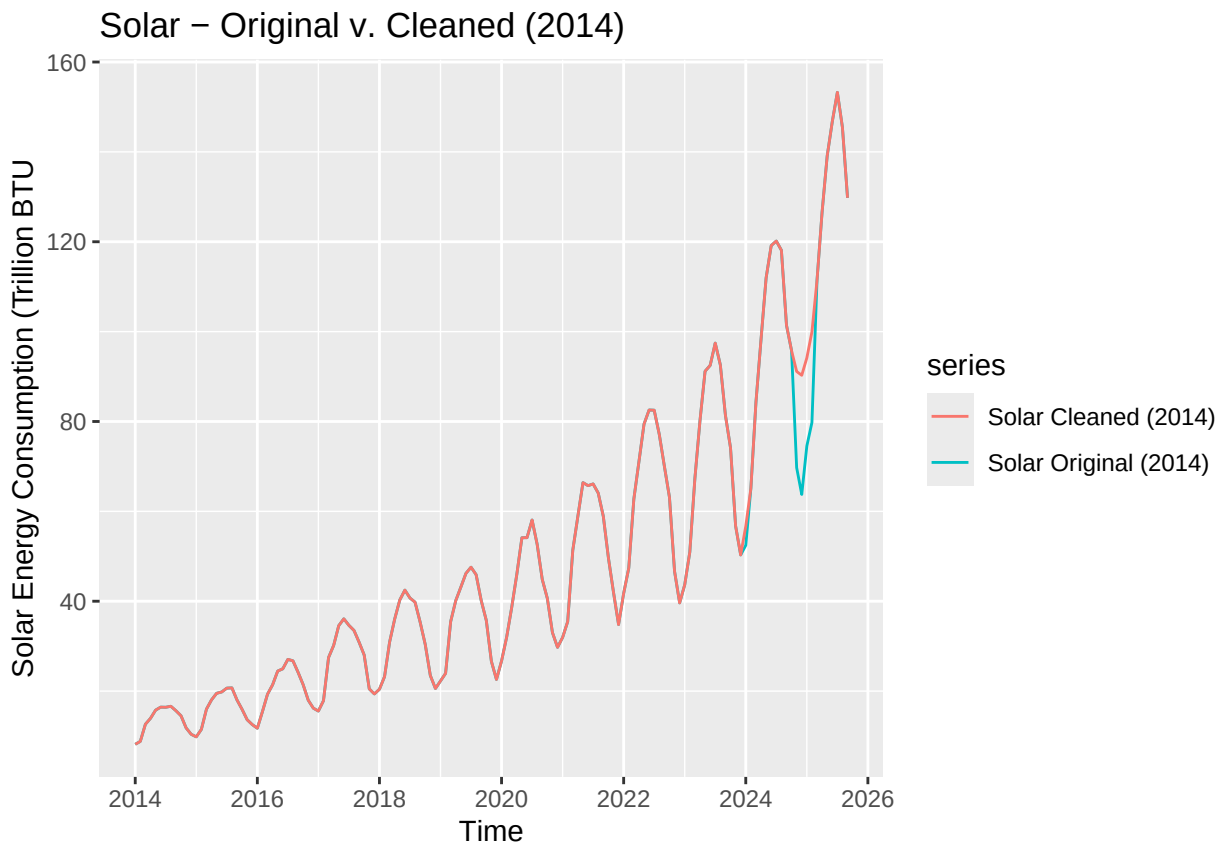
> Answer: For both solar and wind, there are differences between the original and cleaned series, indicating that the tsclean() function has replaced those specific points where there are extreme values. This in turn made the plots for cleaned series have a much rounder/smoothed turns at spikes and/or dips that appear in the original series. From the plots, wind seems to have more outliers than solar does.

Q9

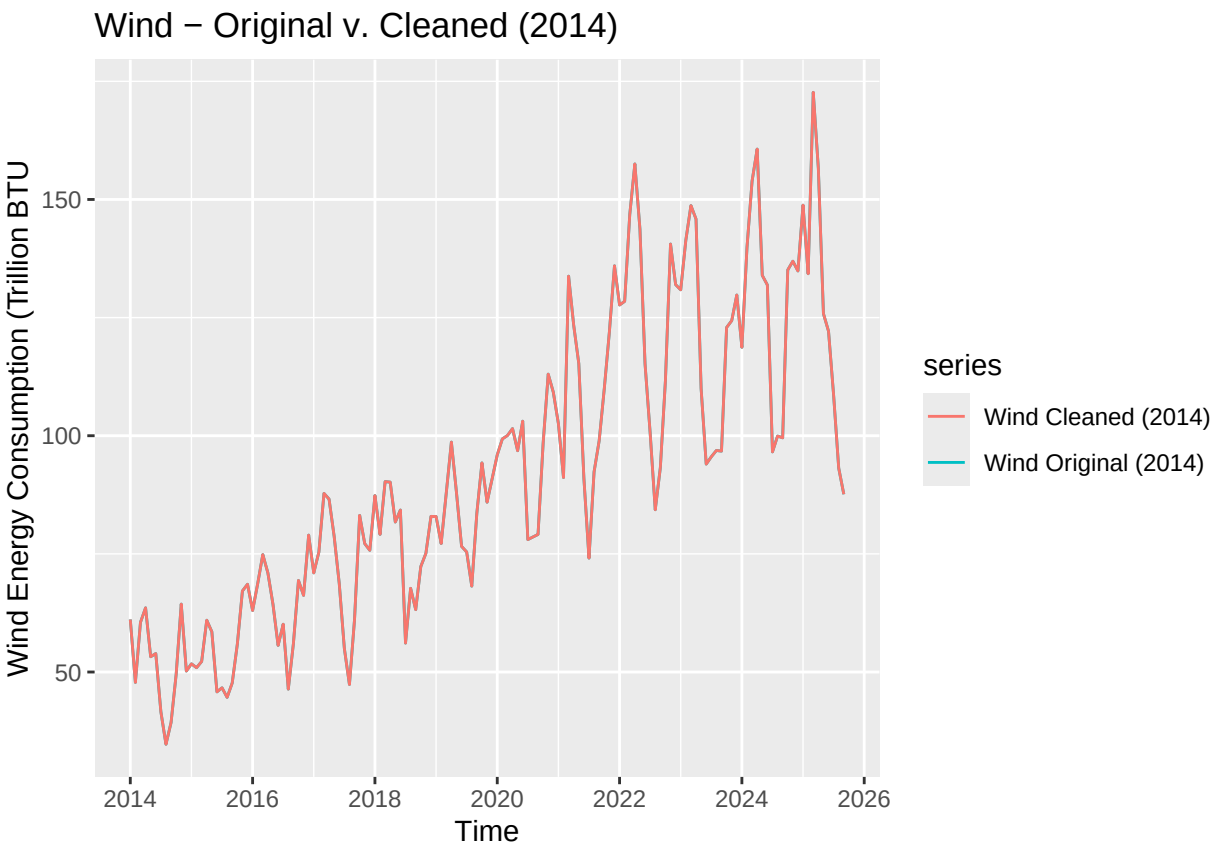
Redo number Q8 but now with the time series you created on Q7, i.e., the series starting in 2014. Using what `autoplot()` again what happened now? Did the function removed any outliers from the series?

```
solar.2014.ts.clean <- tsclean(solar.2014.ts)
wind.2014.ts.clean <- tsclean(wind.2014.ts)

autoplot(solar.2014.ts, series = "Solar Original (2014)") +
  autolayer(solar.2014.ts.clean, series = "Solar Cleaned (2014)") +
  ylab("Solar Energy Consumption (Trillion BTU") +
  ggtitle("Solar - Original v. Cleaned (2014)")
```




```
autoplot(wind.2014.ts, series = "Wind Original (2014)") +
  autolayer(wind.2014.ts.clean, series = "Wind Cleaned (2014)") +
  ylab("Wind Energy Consumption (Trillion BTU") +
  ggtitle("Wind - Original v. Cleaned (2014)")
```



Answer: From the plots, we can see that solar only has 1-2 outlier, and wind has no outliers. This is likely due to the fact that we are working with a more recent dataset that has eliminated rapid growth in the early timer periods.