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

Assignment 5 - Due date 02/17/26

Josh Salzberg

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 1.6.0
## v forcats 1.0.1     v tibble  3.3.1
## v purrr   1.2.1     v tidyr   1.3.1
## v readr   2.1.6

## -- 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
```

```
library(readxl)
```

Consider the same data you used for A04 from the spreadsheet "Table_10.1_Renewable_Energy_Production_and_Consumptio
The data comes from the US Energy Information and Administration and corresponds to the December 2025 Monthly Energy Review.

```
#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`
```

```
#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!

```
A5data <- energy_data %>%
  select(Date = Month,
         SolarEnergy = `Solar Energy Consumption`,
         WindEnergy = `Wind Energy Consumption`)
str(A5data)
```

```
## tibble [633 x 3] (S3: tbl_df/tbl/data.frame)
##  $ Date       : POSIXct[1:633], format: "1973-01-01" "1973-02-01" ...
##  $ SolarEnergy: chr [1:633] "Not Available" "Not Available" "Not Available" "Not Available" ...
##  $ WindEnergy : chr [1:633] "Not Available" "Not Available" "Not Available" "Not Available" ...
```

```r
A5data <- A5data %>%
  mutate(SolarEnergy = as.numeric(SolarEnergy),
         WindEnergy = as.numeric(WindEnergy),
         Date = ymd(Date)) %>%
  drop_na()
```

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

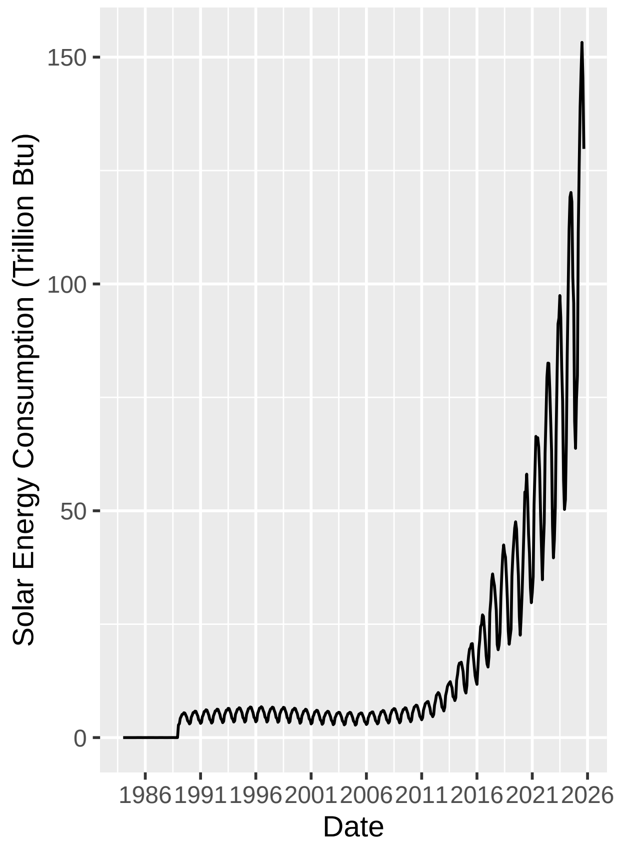
Q2

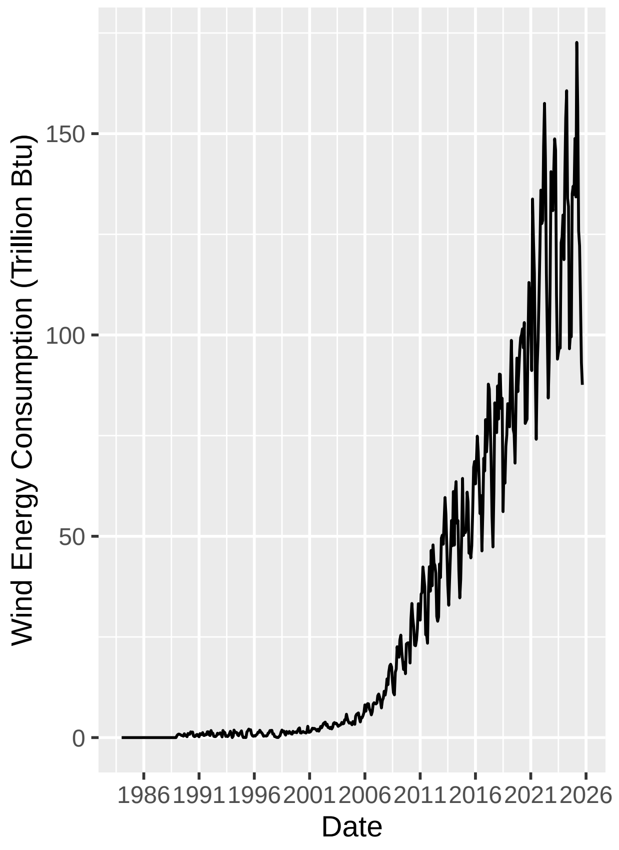
Plot the Solar and Wind energy consumption over time using ggplot. Plot each series on a separate graph. No need to add legend. Add informative names to the y axis using `ylab()`. Explore the function `scale_x_date()` on ggplot and see if you can change the x axis to improve your plot. Hint: use `scale_x_date(date_breaks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

```r
library(cowplot)
```

```
##
## Attaching package: 'cowplot'

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

```r
SolarPlot <- ggplot(data = A5data, aes(x = Date, y = SolarEnergy))+
  geom_line()+
  ylab("Solar Energy Consumption (Trillion Btu)")+
    scale_x_date(date_breaks = "5 years", date_labels = '%Y')

WindPlot <- ggplot(data = A5data, aes(x = Date, y = WindEnergy))+
  geom_line()+
  ylab("Wind Energy Consumption (Trillion Btu)")+
  scale_x_date(date_breaks = "5 years", date_labels = '%Y')

plot_grid(SolarPlot,WindPlot)
```

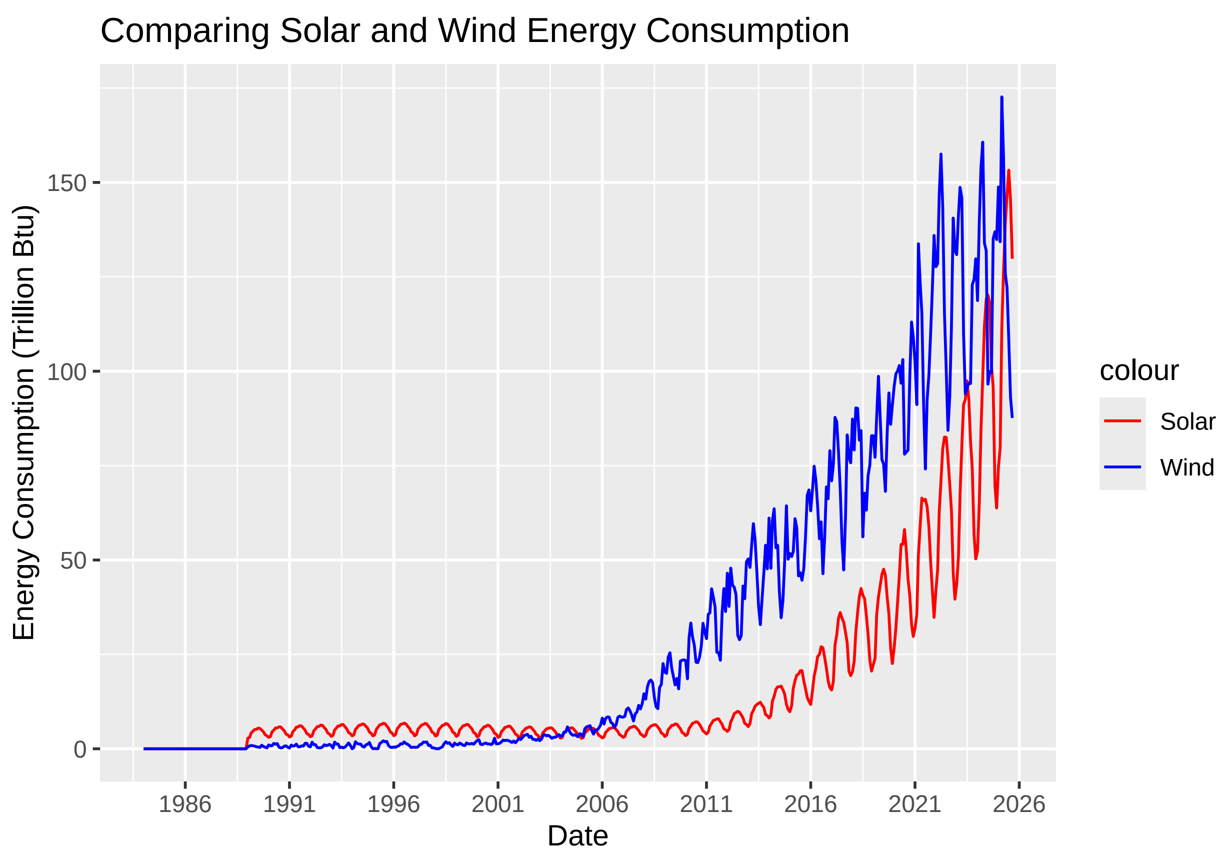




Q3

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.

```
ComboPlot <- ggplot(A5data)+
  geom_line(aes(x = Date, y = SolarEnergy, color = "Solar"))+
  geom_line(aes(x = Date, y = WindEnergy, color = "Wind"))+
  ylab("Energy Consumption (Trillion Btu)")+
  labs(title = "Comparing Solar and Wind Energy Consumption")+
  scale_x_date(date_breaks = "5 years", date_labels = '%Y')+
  scale_color_manual(values = c("Solar" = "red","Wind"= "blue"))

ComboPlot
```



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()`.

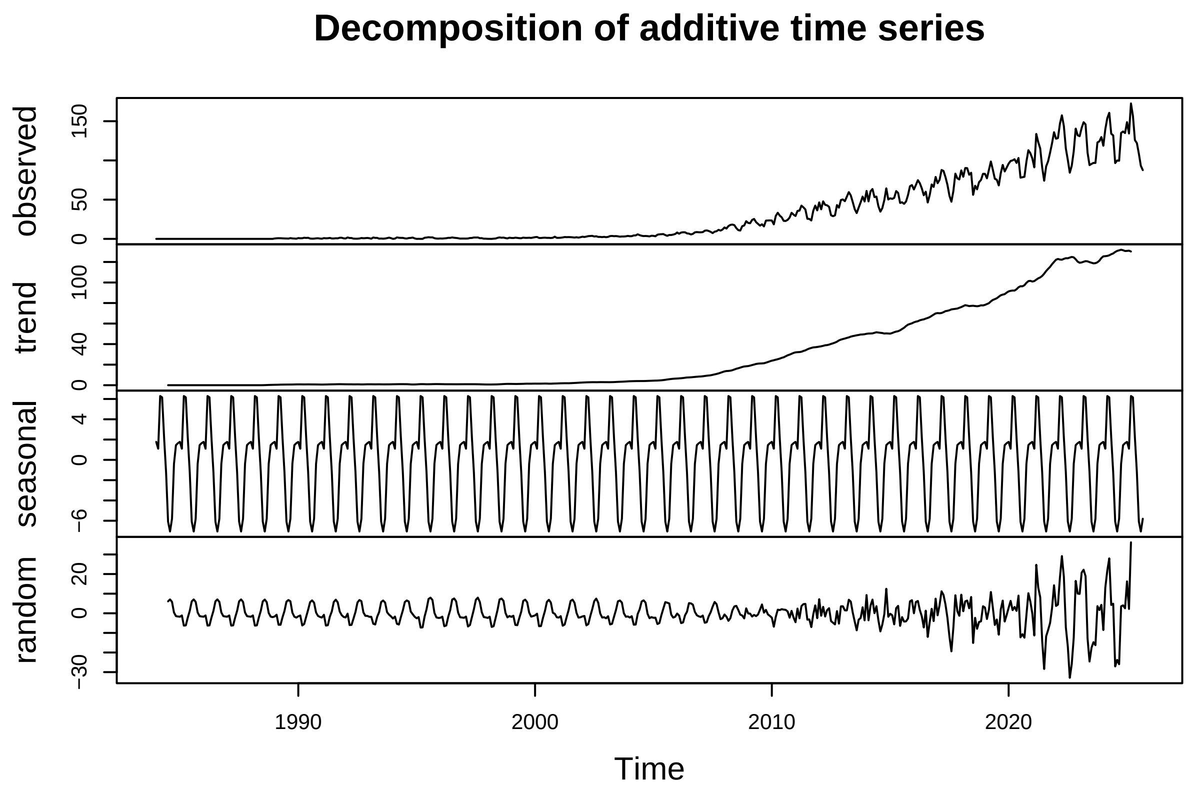
1) You have two options: alternative and multiplicative. Multiplicative models exhibit a change in frequency over time.
2) The trend is not a straight line because it uses a moving average method to detect trend.
3) The seasonal component of the time series is found by subtracting the trend component 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

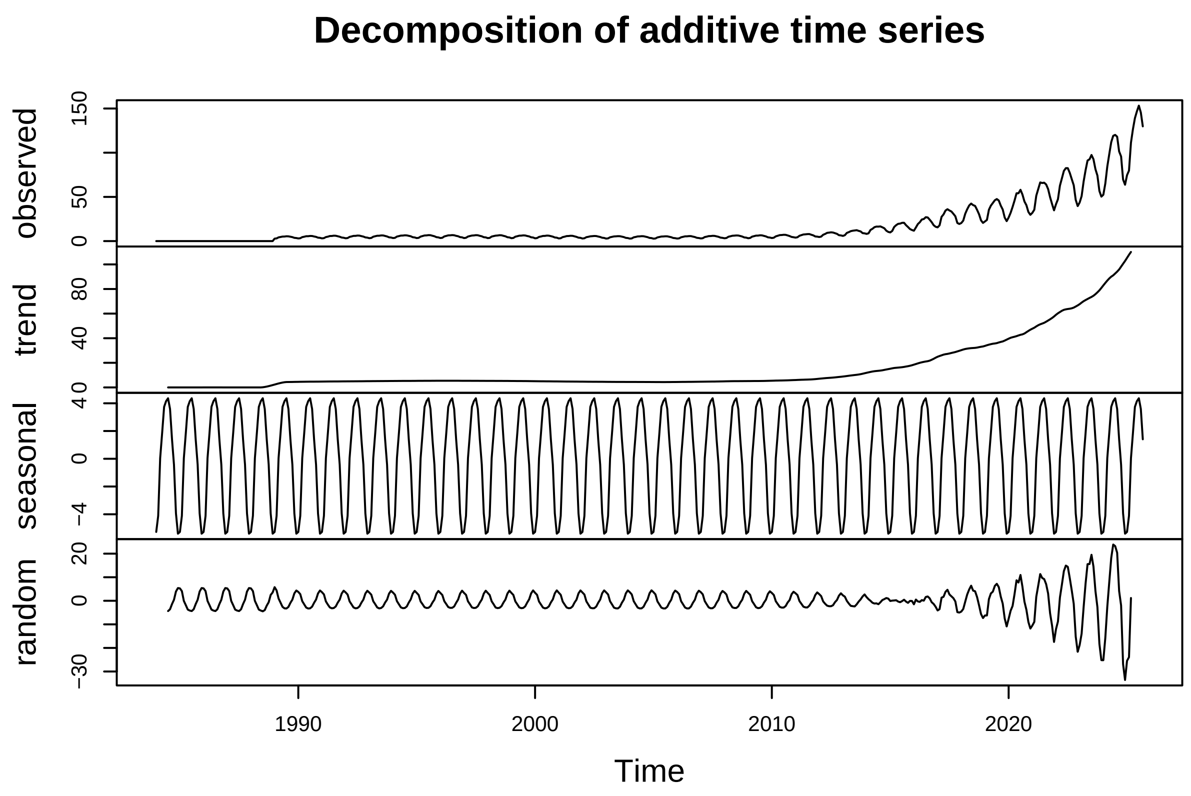
Transform wind and solar series into a time series object and apply the decompose function on them using the additive option, i.e., `decompose(ts_data, type = "additive")`. What can you say about the trend component? What about the random component? Does the random component look random? Or does it appear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

```
Windts <- ts(A5data$WindEnergy, start = c(1984, 1), frequency = 12)
Solarts <- ts(A5data$SolarEnergy, start = c(1984, 1), frequency = 12)

DecompWind <- decompose(Windts, type = "additive")
```

```
DecompSolar <- decompose(Solarts, type = "additive")

plot(DecompWind)
```



```
plot(DecompSolar)
```

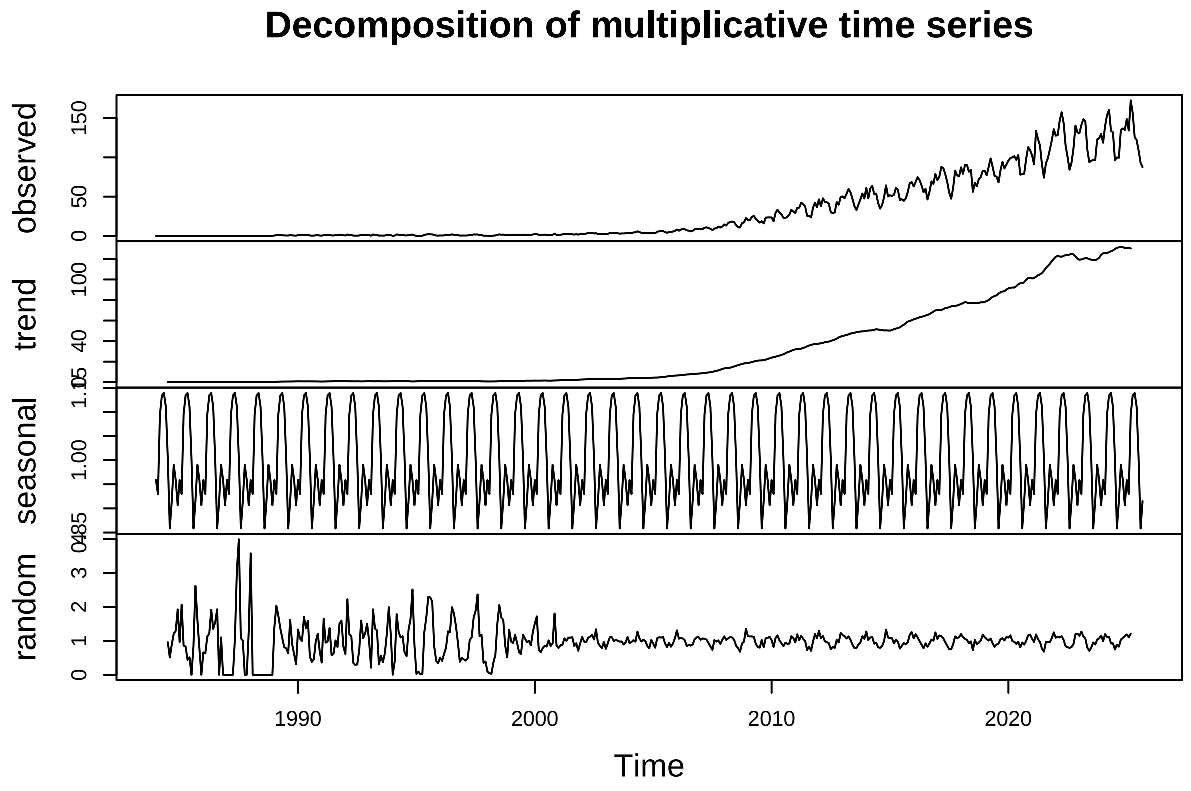


answer: The wind trend rises steadily until around 2020, when there appears to almost be a level shift to the trend. the Solar trend evokes a slow exponential curve to my eye. The random component for both series appears to have significant seasonality in them, though more true randomness comes in at the end of the wind series as the trend smoothes out.

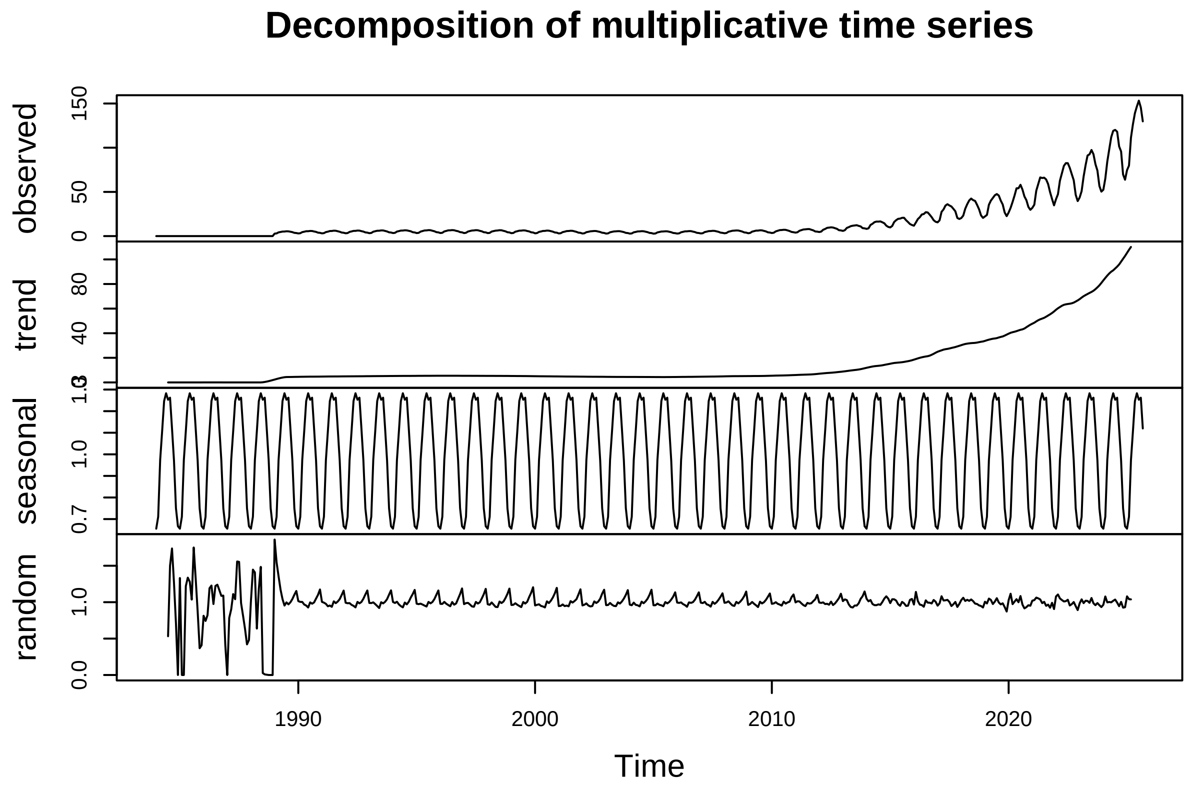
Q5

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
DecompWind2 <- decompose(Windts, type = "multiplicative")
DecompSolar2 <- decompose(Solarts, type = "multiplicative")

plot(DecompWind2)
```



```r
plot(DecompSolar2)
```



Answer: The Wind series random component truly seems to capture the randoness and irregularity of the data, particularly in the early portion of the series. Significantly less seasonality remains in the latter third of the plot. The solar plot is similar, there's intense randomness until about 1988 or 1989, and then a minute amount of seasonality remains. In both cases, the y-axis shifts from being centered on the 0-line to being centered on the 1-line.

Q6

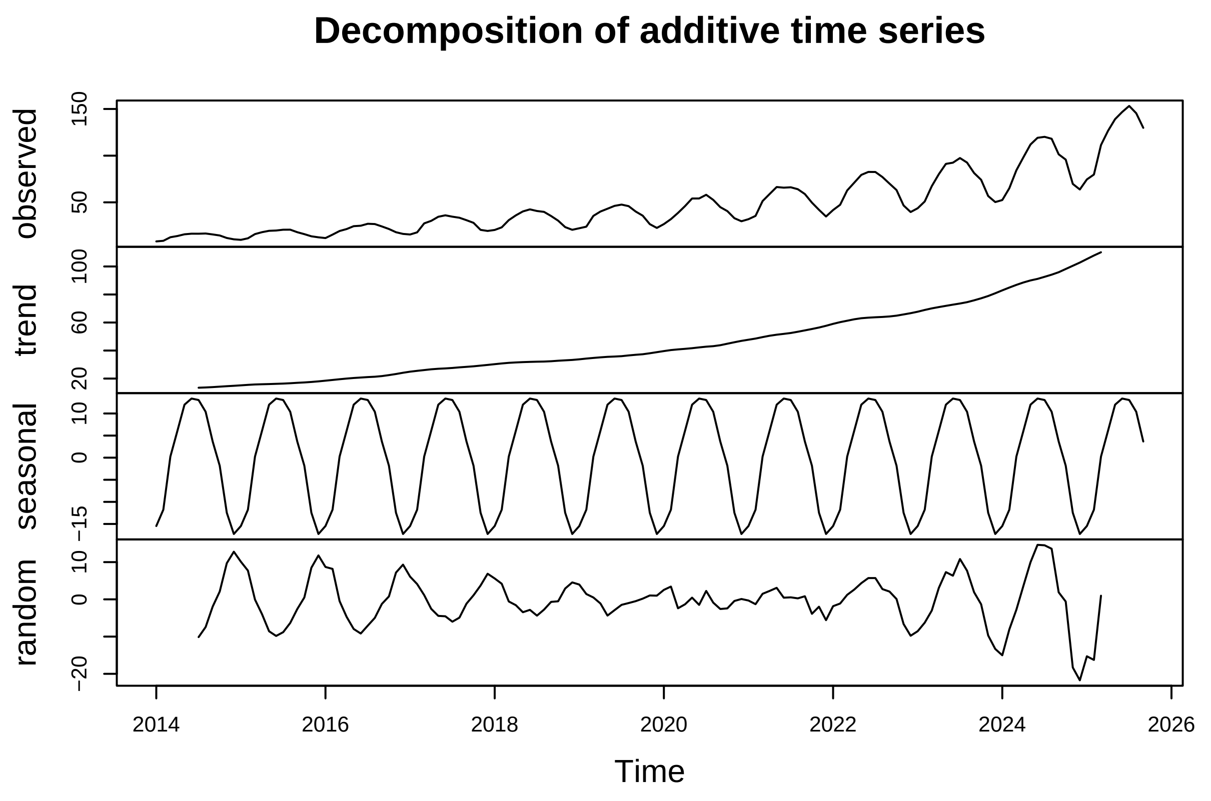
When fitting a model to this data, do you think you need all the historical data? Think about the data from 80s, 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

Answer: The complete historical data are most likely going to obfuscate the patterns in trend, seasonality and randomness that have the greatest bearing on the next six months of consumption. Through many different policy regime changes, technological advancements, and shifting societal preferences, the landscape of energy consumption looks very different from the 80s, 90s, and even the aughts. How we prioritize energy sources has also shifted dramatically, with some consumers being willing to pay premiums for certain kinds of energy over others. Less a matter of specific shocks to the energy consumption landscape (save the spillover effects of oil/gas booms & busts), there's been a long term series of shifts.

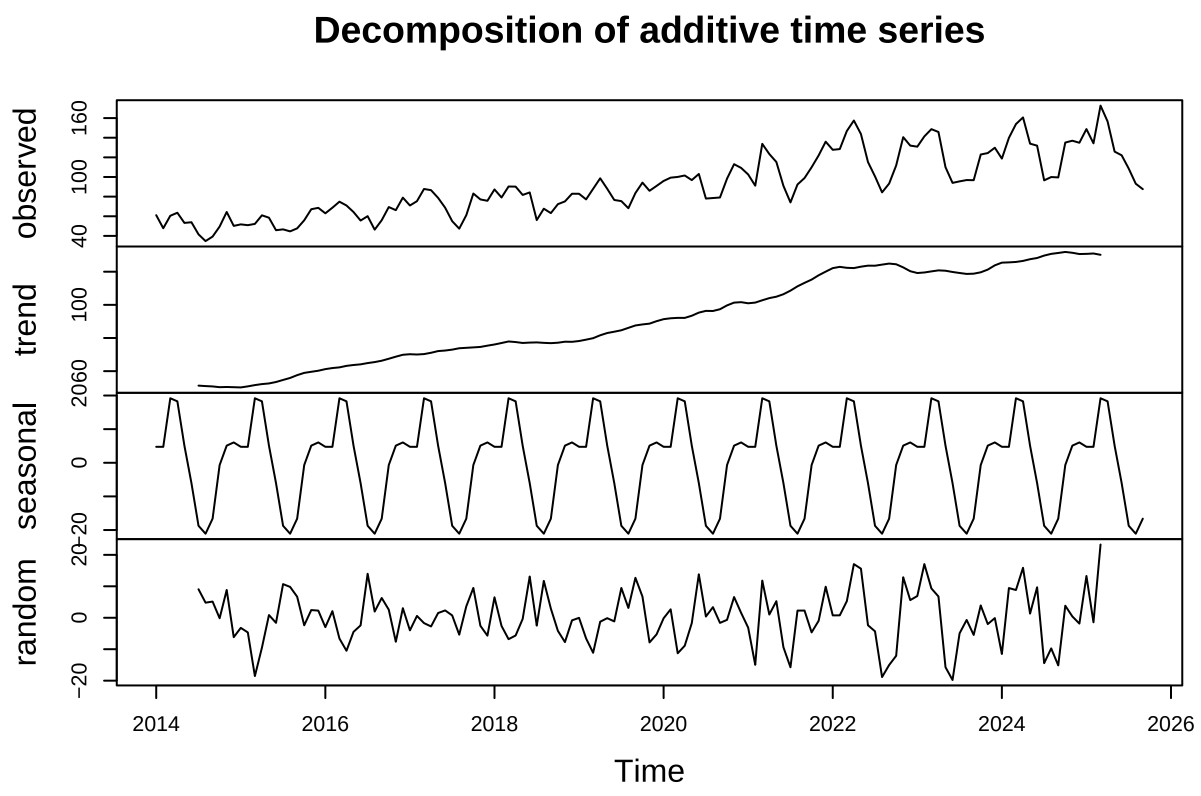
Q7

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
RecentData <- A5data %>%
  filter(year(Date) >=2014)

RecentData_ts <- ts(RecentData[,2:3], start=c(2014,1), frequency = 12)

RecentSolar_ts_decomp <- decompose(RecentData_ts[,1], type = 'additive')
RecentWind_ts_decomp <- decompose(RecentData_ts[,2], type = 'additive')

plot(RecentSolar_ts_decomp)
```



```r
plot(RecentWind_ts_decomp)
```



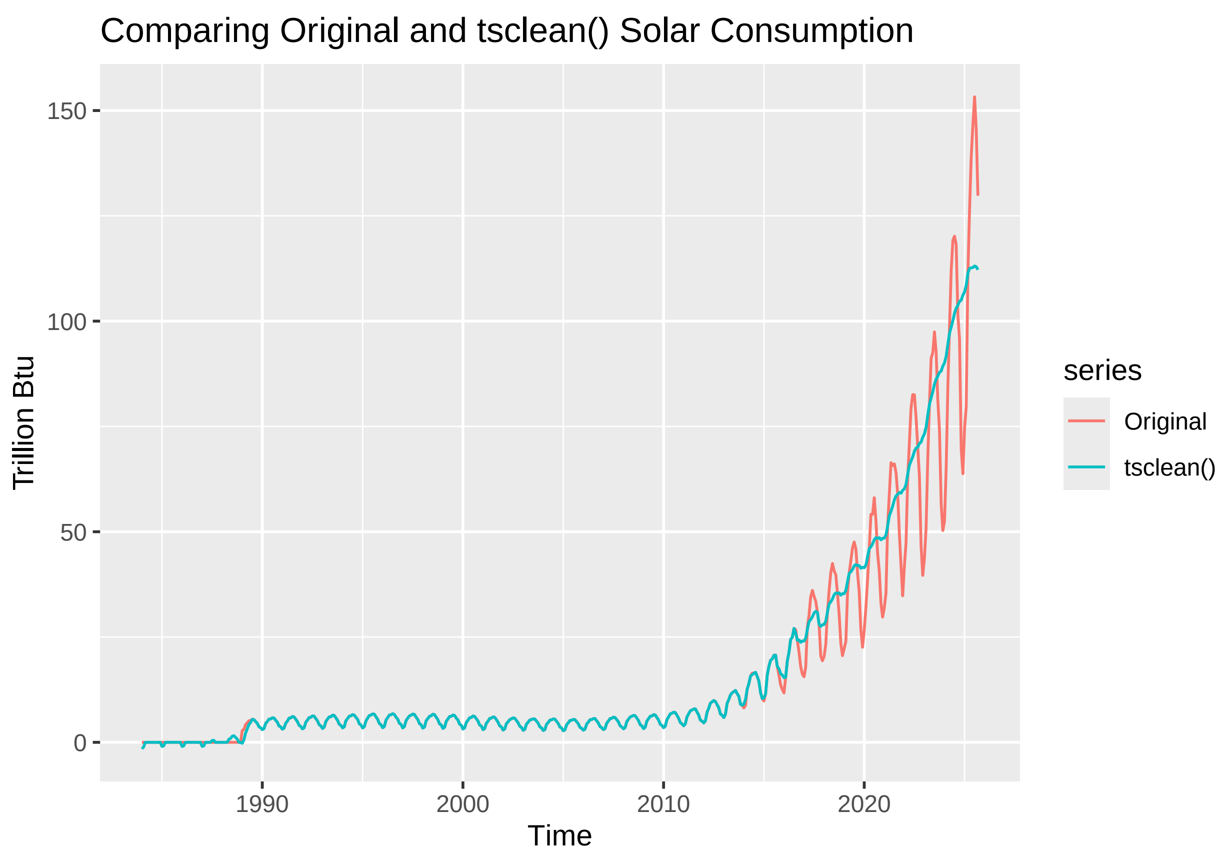
Answer: In the case of solar consumption, the random component is very aligned with the seasonal component until the mid-point of the series (2020ish) and returns to alignment in 2022. In the case of wind consumption, the random component appears negatively correlated with the seasonal component, but follows a fairly consistent pattern throughout, with peaks and valleys appearing in sequence.

Identify and Remove outliers

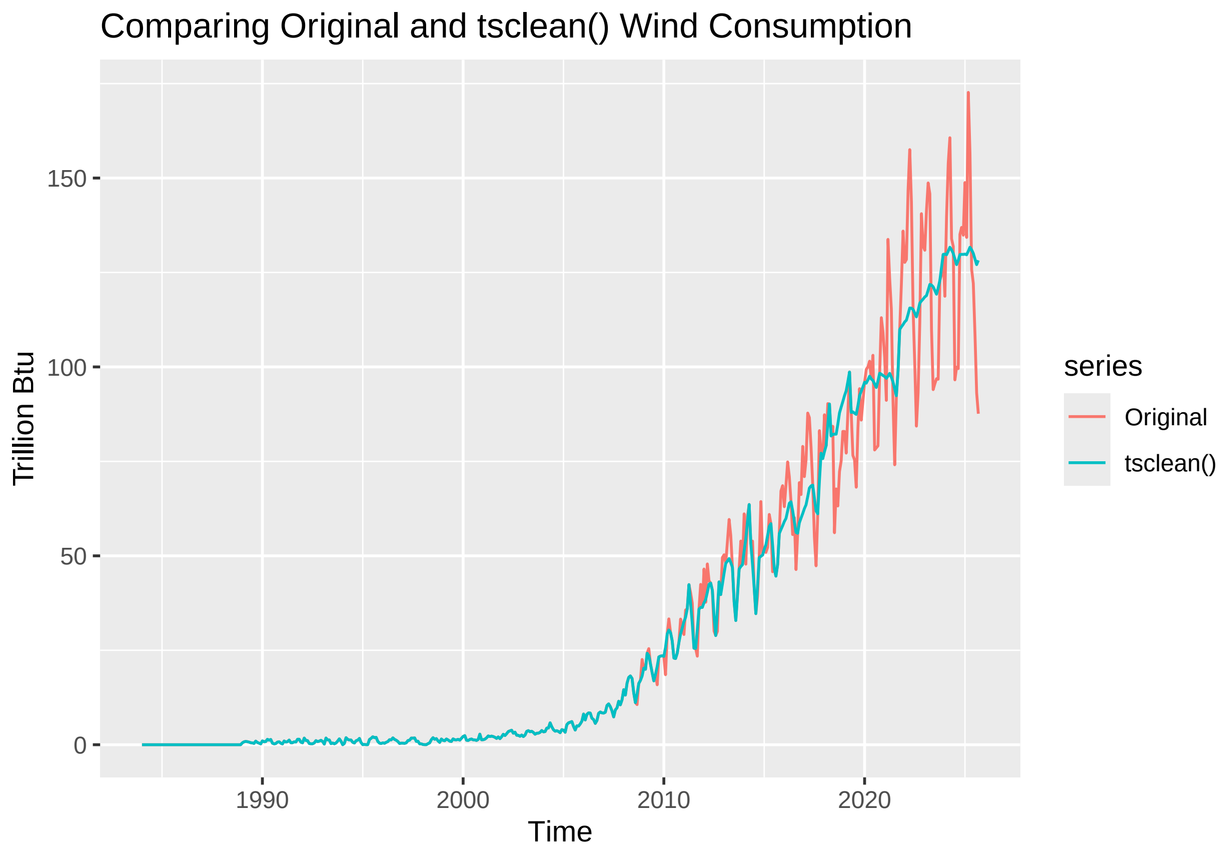
Q8

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
CleanSolar_ts <- tsclean(Solarts)
CleanWind_ts <- tsclean(Windts)

autoplot(Solarts, series = "Original")+
  autolayer(CleanSolar_ts, series = "tsclean()")+
  labs(title = "Comparing Original and tsclean() Solar Consumption",
       y = "Trillion Btu",
       x = "Time")
```



```r
autoplot(Windts, series = "Original")+
  autolayer(CleanWind_ts, series = "tsclean()")+
  labs(title = "Comparing Original and tsclean() Wind Consumption",
       y = "Trillion Btu",
       x = "Time")
```

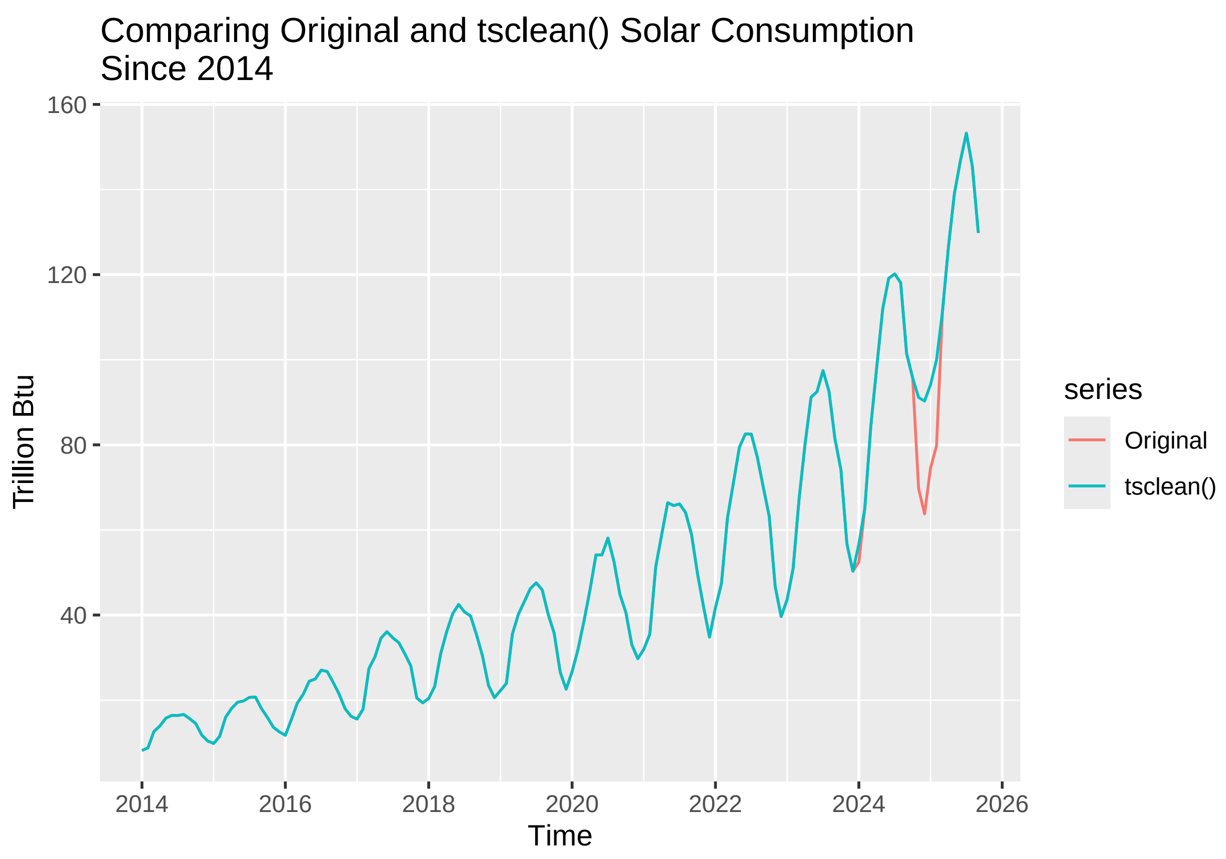


answer: In both series we see a significant number of outlier dropped. In the solar series, the tsclean() function dropped outliers from the beginning of the data all the way through the end. The most dramatic effect is seen in the years following 2015: the tsclean() series is smoothed to mostly just a trend, while the original data has sizable peaks and troughs. In the case of the wind series, the tsclean() function removed outliers only beginning in ~2009. The series is not quite as smooth as in the solar series, but the majority of the significant peaks and troughs are removed.

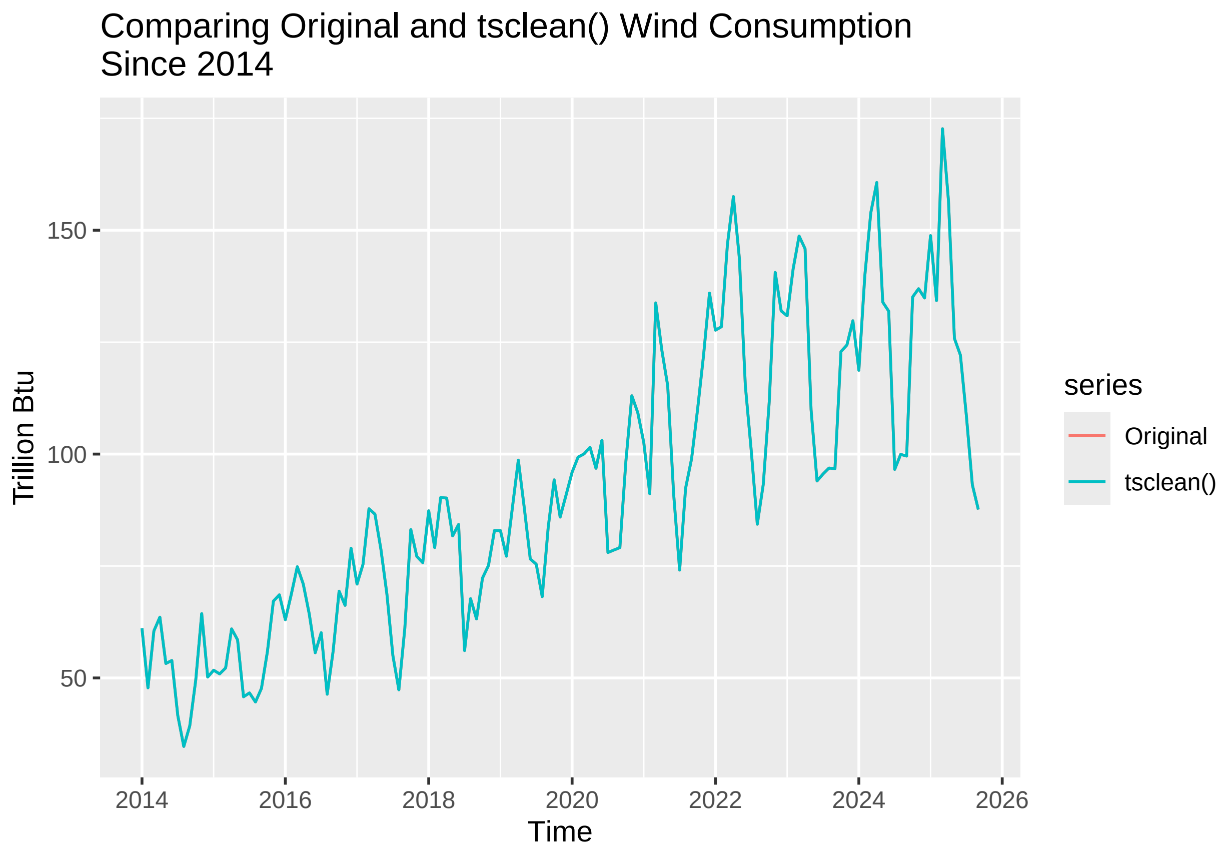
Q9

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?

```
Recent_CleanSolar_ts <- tsclean(RecentData_ts[,1])
Recent_CleanWind_ts <- tsclean(RecentData_ts[,2])

autoplot(RecentData_ts[,1], series = "Original")+
  autolayer(Recent_CleanSolar_ts, series = "tsclean()")+
  labs(title = "Comparing Original and tsclean() Solar Consumption \nSince 2014",
       y = "Trillion Btu",
       x = "Time")
```



```
autoplot(RecentData_ts[,2], series = "Original")+
  autolayer(Recent_CleanWind_ts, series = "tsclean()")+
  labs(title = "Comparing Original and tsclean() Wind Consumption \nSince 2014",
       y = "Trillion Btu",
       x = "Time")
```



Answer: The solar series had some outliers removed, corresponding to the troughs in the winters of 2024 and 2025. The wind series did not appear to have any outlier removed. This is significantly different from the previous exercise.