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

Assignment 5 - Due date 02/18/25

Chloe Young

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.\

```
#Load/install required package here
library(openxlsx)
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library(lubridate)
```

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

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

```
library(forecast)
```

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

```
library(tseries)
library(ggplot2)
library(Kendall)
library(lubridate)
library(tidyverse)  #load this package so yon clean the data frame using pipes
```

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

## -- 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
```

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 without change the original file using read.xlsx
energy_data1 <- read_excel(path="/home/guest/TSA_Sp25/Data/Table_10.1_Renewable_Energy_Production_and_C
```

```
## 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
read_col_names <- read_excel(path="/home/guest/TSA_Sp25/Data/Table_10.1_Renewable_Energy_Production_and_
```

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

```
colnames(energy_data1) <- read_col_names
head(energy_data1)
```

```
## # 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_data1)
nvar=ncol(energy_data1)
```

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!

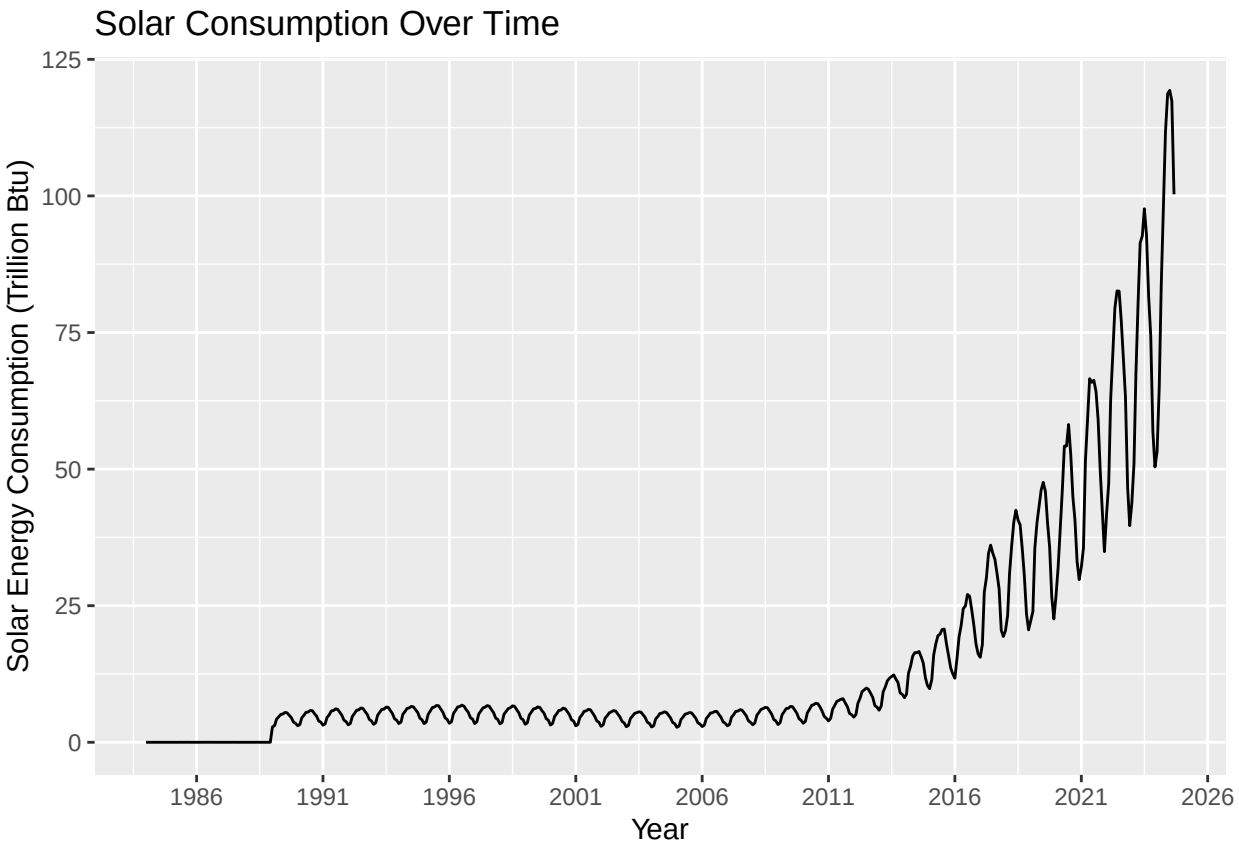
```r
energy_data_cleaned <- energy_data1 %>%
  select(Month, `Solar Energy Consumption`, `Wind Energy Consumption`) %>%
  filter(`Solar Energy Consumption` != "Not Available", `Wind Energy Consumption` != "Not Available") %
  mutate(
    `Solar Energy Consumption` = as.numeric(`Solar Energy Consumption`),
    `Wind Energy Consumption` = as.numeric(`Wind Energy Consumption`),
    Month = as.Date(Month)
  ) %>%
  drop_na()

head(energy_data_cleaned)
```

```
## # A tibble: 6 x 3
##   Month      `Solar Energy Consumption` `Wind Energy Consumption`
##   <date>                          <dbl>                     <dbl>
## 1 1984-01-01                      0                         0
## 2 1984-02-01                      0                         0.001
## 3 1984-03-01                      0.001                     0.001
## 4 1984-04-01                      0.001                     0.002
## 5 1984-05-01                      0.002                     0.003
## 6 1984-06-01                      0.003                     0.002
```

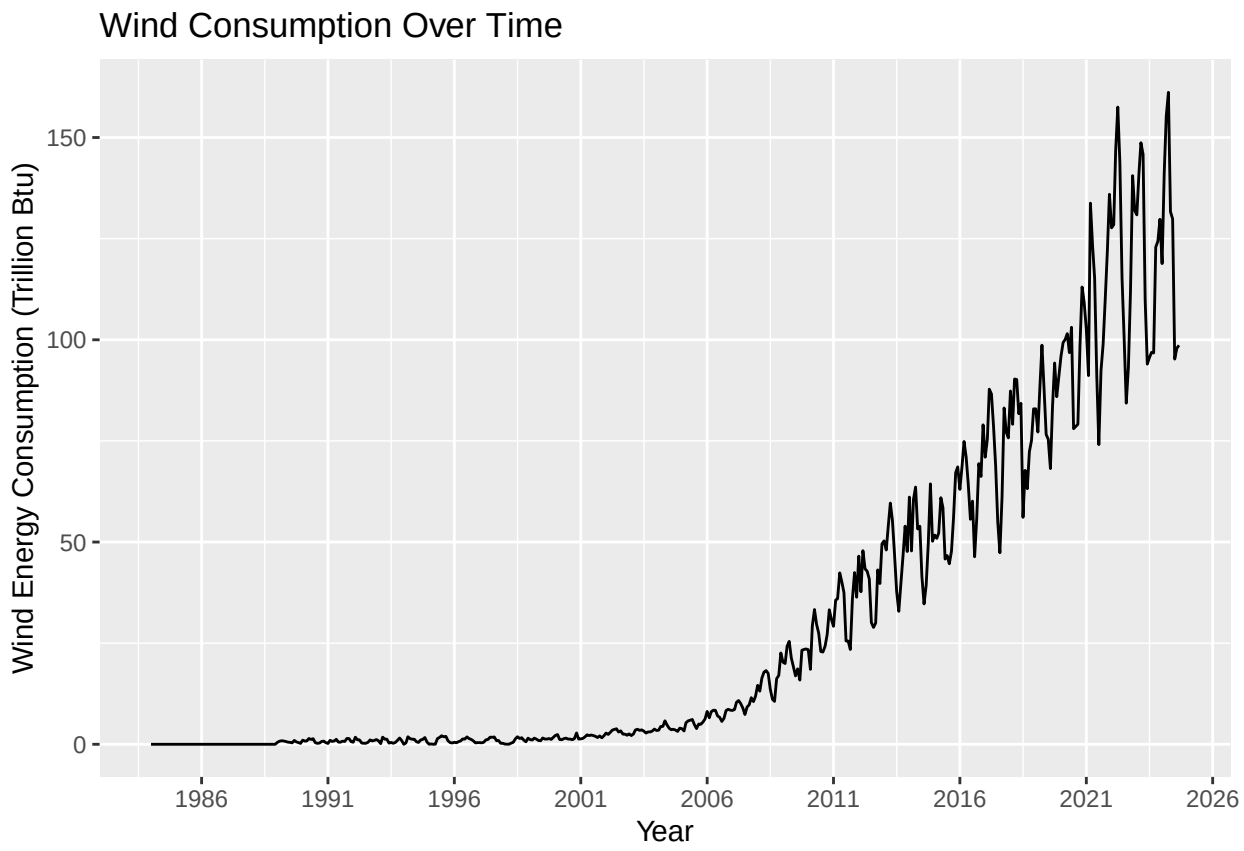
Q2

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
#Wind Plot
solarplot <- ggplot(energy_data_cleaned, aes(x = Month, y = `Solar Energy Consumption`)) +
  geom_line() +
  ylab("Solar Energy Consumption (Trillion Btu)") +
  xlab("Year") +
  ggtitle("Solar Consumption Over Time") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")

#Wind Plot
windplot <- ggplot(energy_data_cleaned, aes(x = Month, y = `Wind Energy Consumption`)) +
  geom_line() +
  ylab("Wind Energy Consumption (Trillion Btu)") +
  xlab("Year") +
  ggtitle("Wind Consumption Over Time") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")

print(solarplot)
```



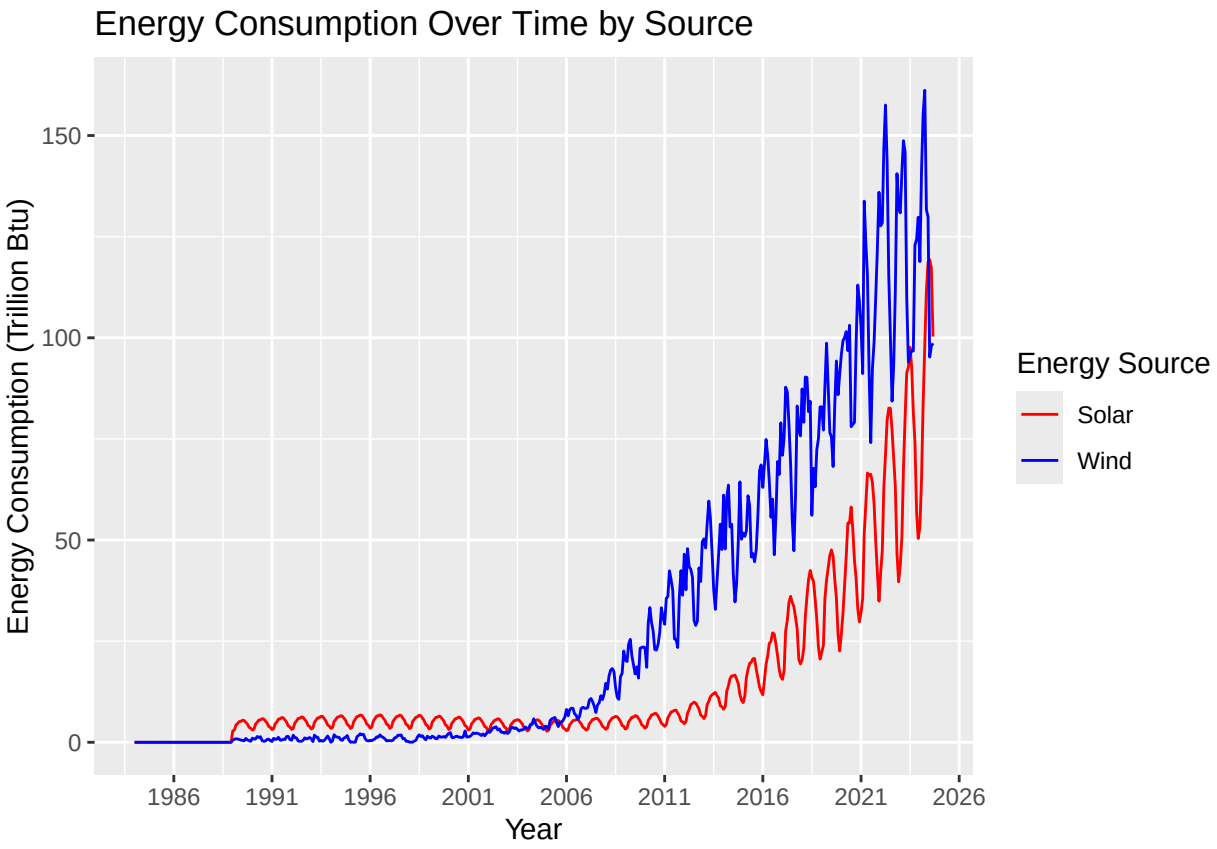

```
print(windplot)
```



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.

```
combinedplot <- ggplot(energy_data_cleaned) +
  geom_line(aes(x = Month, y = `Solar Energy Consumption`, color = "Solar")) +
  geom_line(aes(x = Month, y = `Wind Energy Consumption`, color = "Wind")) +
  ylab("Energy Consumption (Trillion Btu)") +
  xlab("Year") +
  ggtitle("Energy Consumption Over Time by Source") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  scale_color_manual(values = c("Solar" = "red", "Wind" = "blue"),
                     name = "Energy Source")

print(combinedplot)
```



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.

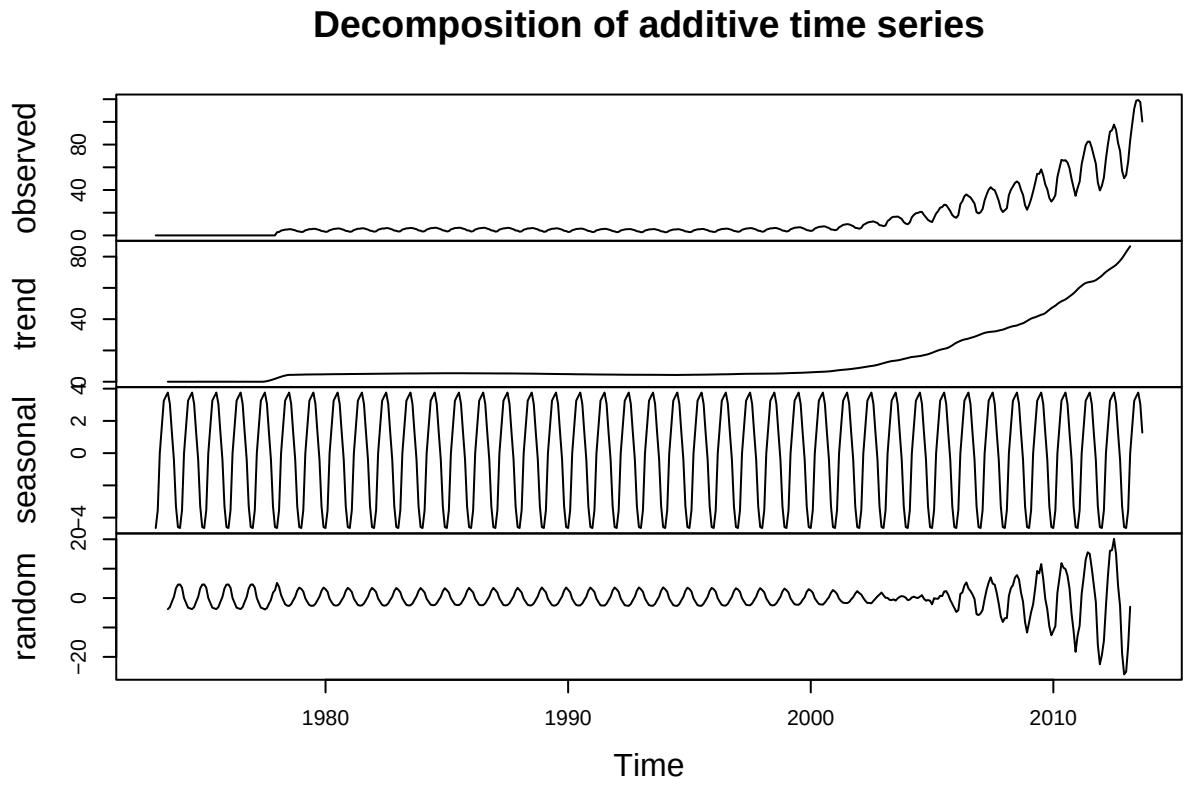
Additional info on `decompose()`.

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.

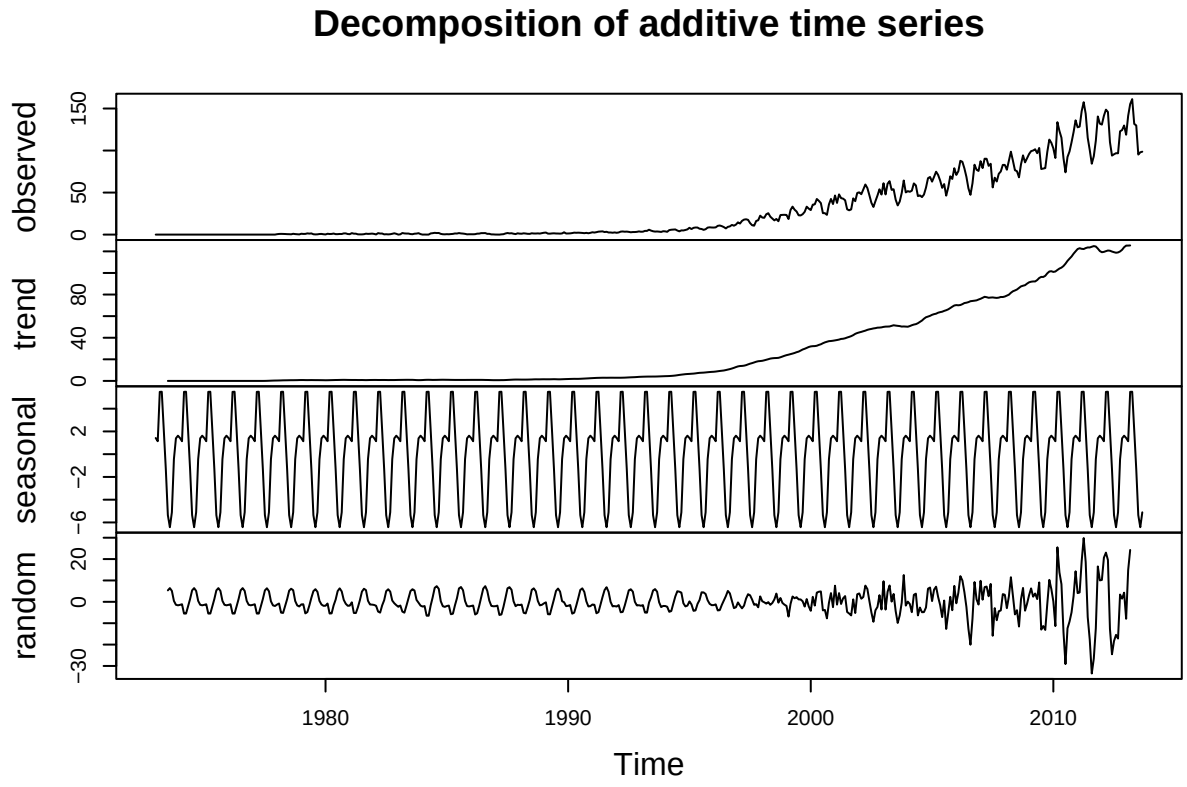
Q4

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
ts_solar <- ts(energy_data_cleaned$`Solar Energy Consumption`,frequency=12,start=c(1973,1))
ts_wind <- ts(energy_data_cleaned$`Wind Energy Consumption`,frequency=12,start=c(1973,1))
ts_data <- ts(energy_data_cleaned,frequency=12,start=c(1973,1))

decompose_solar <- decompose(ts_solar, type = "additive")
decompose_wind <- decompose(ts_wind, type = "additive")
decompose_energy <- decompose(ts_data, type = "additive")

plot(decompose_solar)
```




```r
plot(decompose_wind)
```



For solar, the trend component effectively captures that solar energy consumption is increasing starting in the mid 2000s. It doesn't show the seasonal variation throughout the year, but displays the overall increase which makes sense according to our knowledge of solar technology development. The seasonal graph displays a periodic pattern which likely represents how solar availability changes throughout the year with sunlight variation. The random panel shows the fliuctuations that aren't explained by trend or season. These fluctuations increase alongside solar energy consumption increase, likely displaying that there are certain factors affecting consumption, like technology changes or weather occurrences, that are irregular. The random component still looks slightly seasonal as it corresponds with peaks and troughs in the seasonal panel, indicating that unusual weather correlated with season may be a cause of some solar consumption extremes.

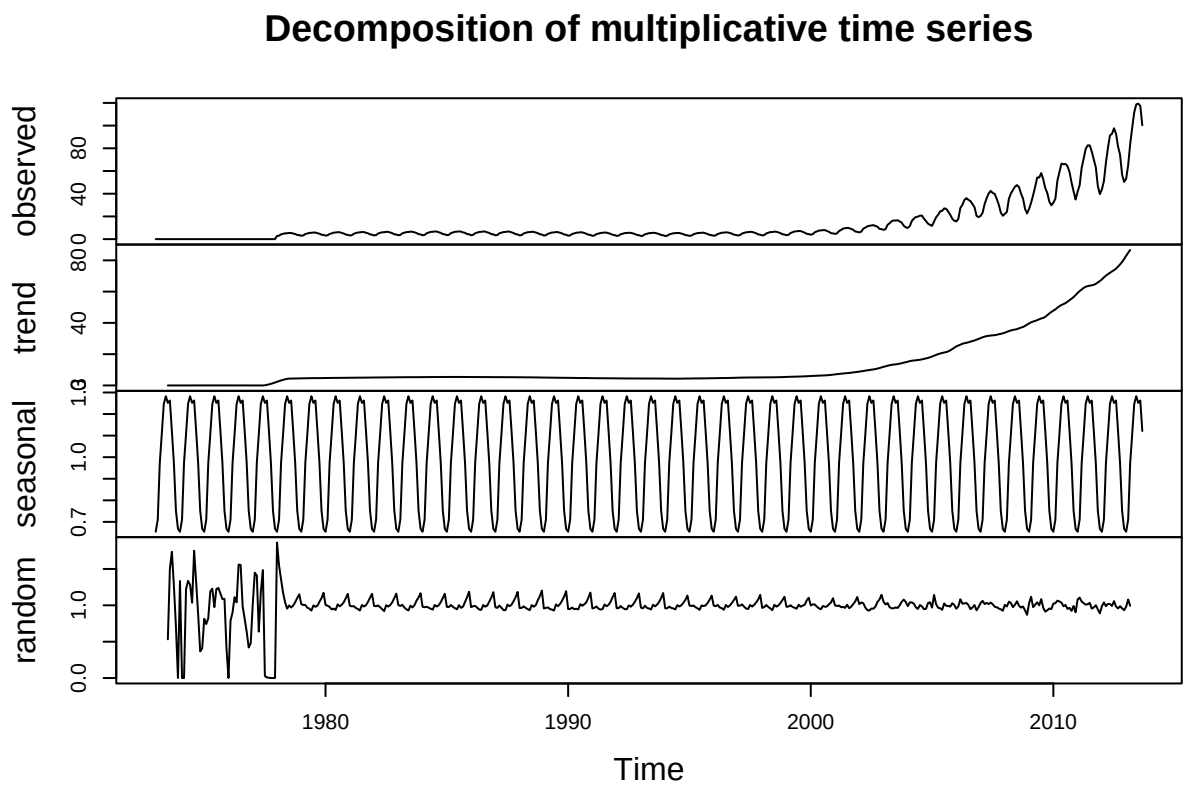
We see very similar patterns for wind. There is a clear increasing trend since the mid 2000s and there is also a seasonal pattern likely reflecting wind speed variation or electricity demand changes in different months of the year. The random panel shows more volatility as wind consumption increases, likely because as wind capacity increases, there is more volatility in wind production, as wind is inherently linked to weather so with more wind comes more inevitable fluctuations that aren't explained by seasonal or trend variation. This random panel also looks correlated to season, indicating that extreme wind or lack of wind during specific seasons may be a cause.

Q5

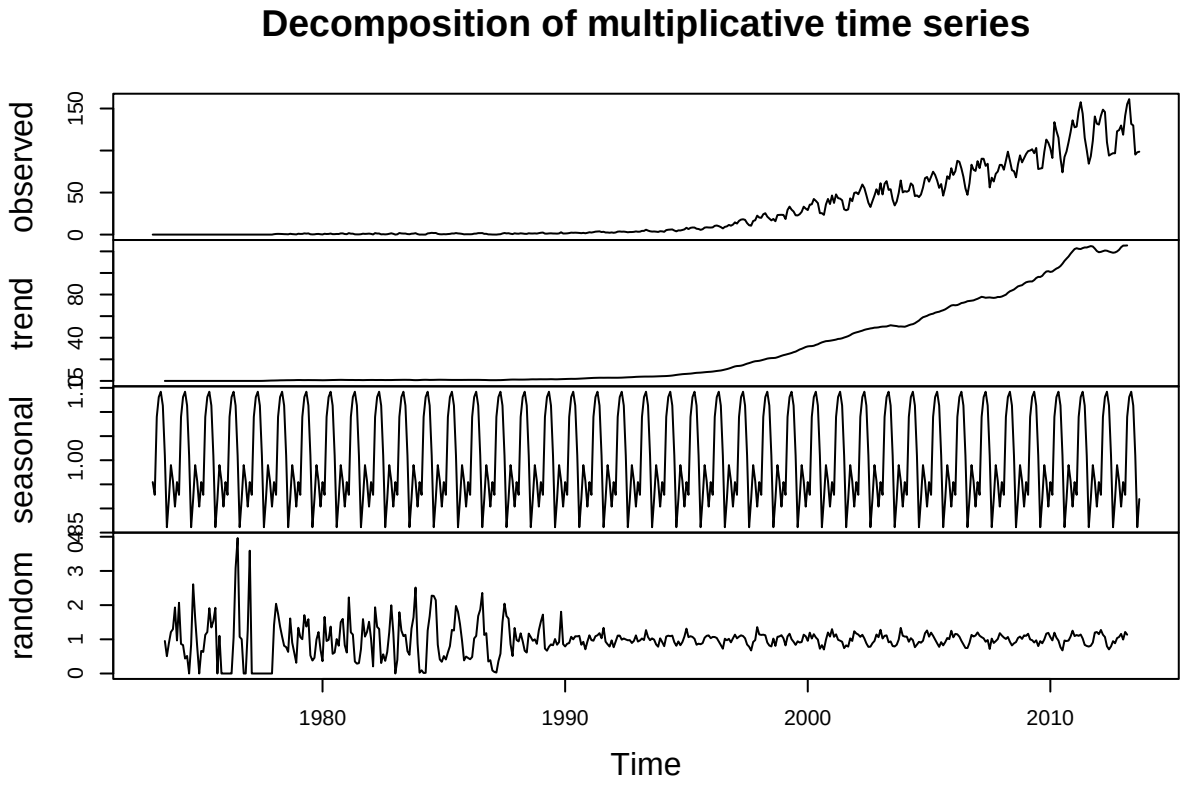
Use the decompose function again but now change the type of the seasonal component from additive to multiplicative. What happened to the random component this time?

```
decompose_solarM <- decompose(ts_solar, type = "multiplicative")
decompose_windM <- decompose(ts_wind, type = "multiplicative")
```

```r
decompose_energyM <- decompose(ts_data, type = "multiplicative")

plot(decompose_solarM)
```




```r
plot(decompose_windM)
```



In the multiplicative graph, the random component completely changed and the variation is now shown in earlier years, prior to the trend increasing. This is because in multiplicative models, each value on the random panel shows the ratio of the actual value to the expected value from the trend and seasonal components. This means that in early years when solar consumption is low, small fluctuations are more important, causing the random values to fluctuate more than in later years when the random panel centers around 1, indicating that the actual data aligns with what's expected based on trend and seasonality. A very similar conclusion can be drawn about wind energy, as the same pattern is shown just with more variation until the 90s.

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:

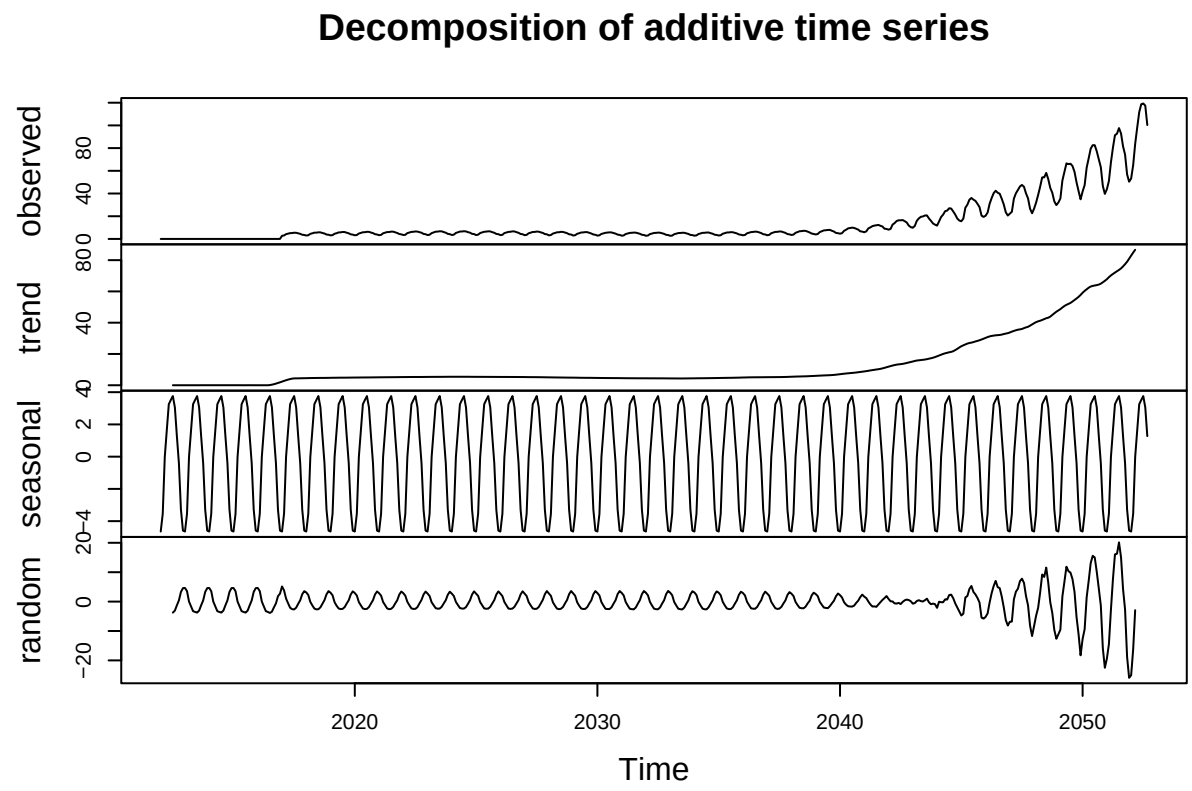
No, I do not think all the historical data is necessary because Solar and Wind consumption didn't start increasing until around 2006 when the technologies developed. Therefore, the years of the 90s and early 20s aren't showing the same trend as now or what the future hold, so this information isn't necessary to forecast the next 6 months.

Q7

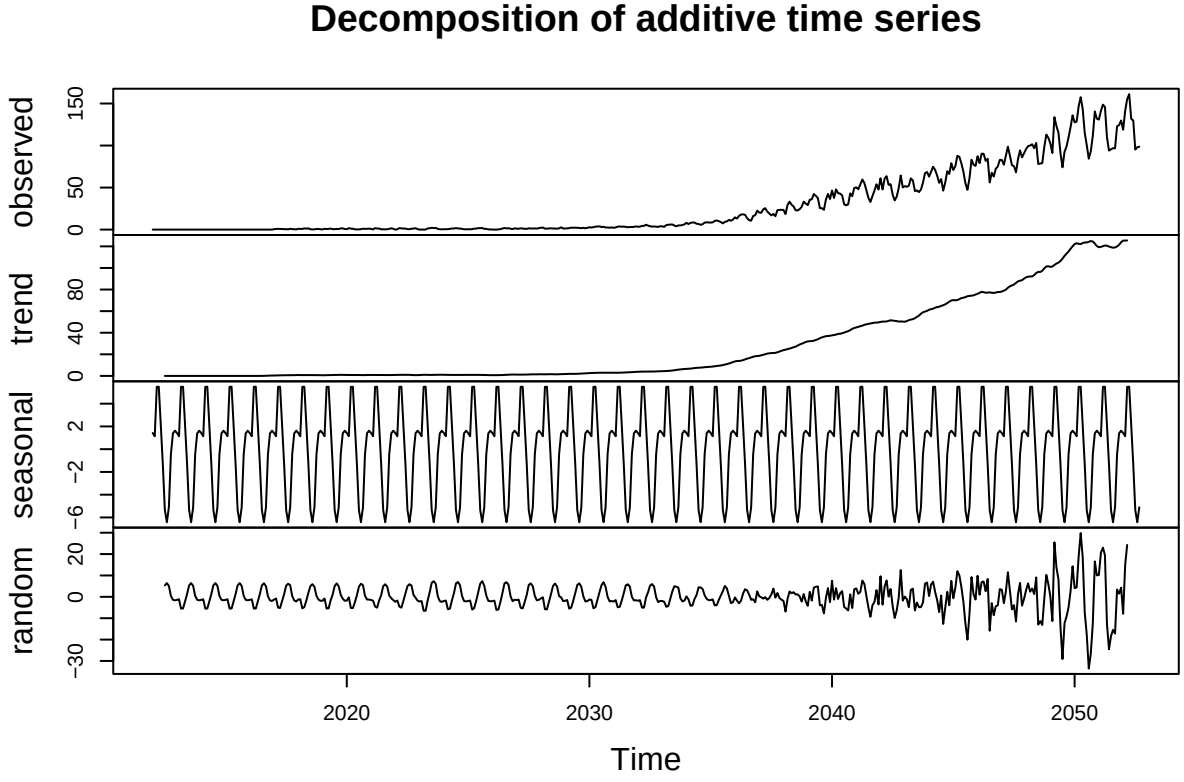
Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the

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.

```
energy_data_2012 <- energy_data_cleaned %>%
  filter(year(Month) >= 2012)

ts_solar_2012 <- ts(energy_data_cleaned$`Solar Energy Consumption`,frequency=12,start=c(2012,1))
ts_wind_2012 <- ts(energy_data_cleaned$`Wind Energy Consumption`,frequency=12,start=c(2012,1))

decompose_solar_2012 <- decompose(ts_solar_2012, type = "additive")
decompose_wind_2012 <- decompose(ts_wind_2012, type = "additive")

plot(decompose_solar_2012)
```




```
plot(decompose_wind_2012)
```



Answer:

For solar, from 2012 until around 2040, there are relatively constant fluctuations around 0, indicating that there is some variation in solar consumption that isn't explained by trend or season, but these variations aren't extreme. Then, the random panel shows extreme variation around 2045, which is also when the observed data shows an increase in solar consumption. This indicates that random variation is expected to increase as consumption increases, which makes sense because with more solar, extreme sun or lack of sunlight will be inevitably more frequent. However, the random componenent still correlates with seasons, so it doesn't look fully random.

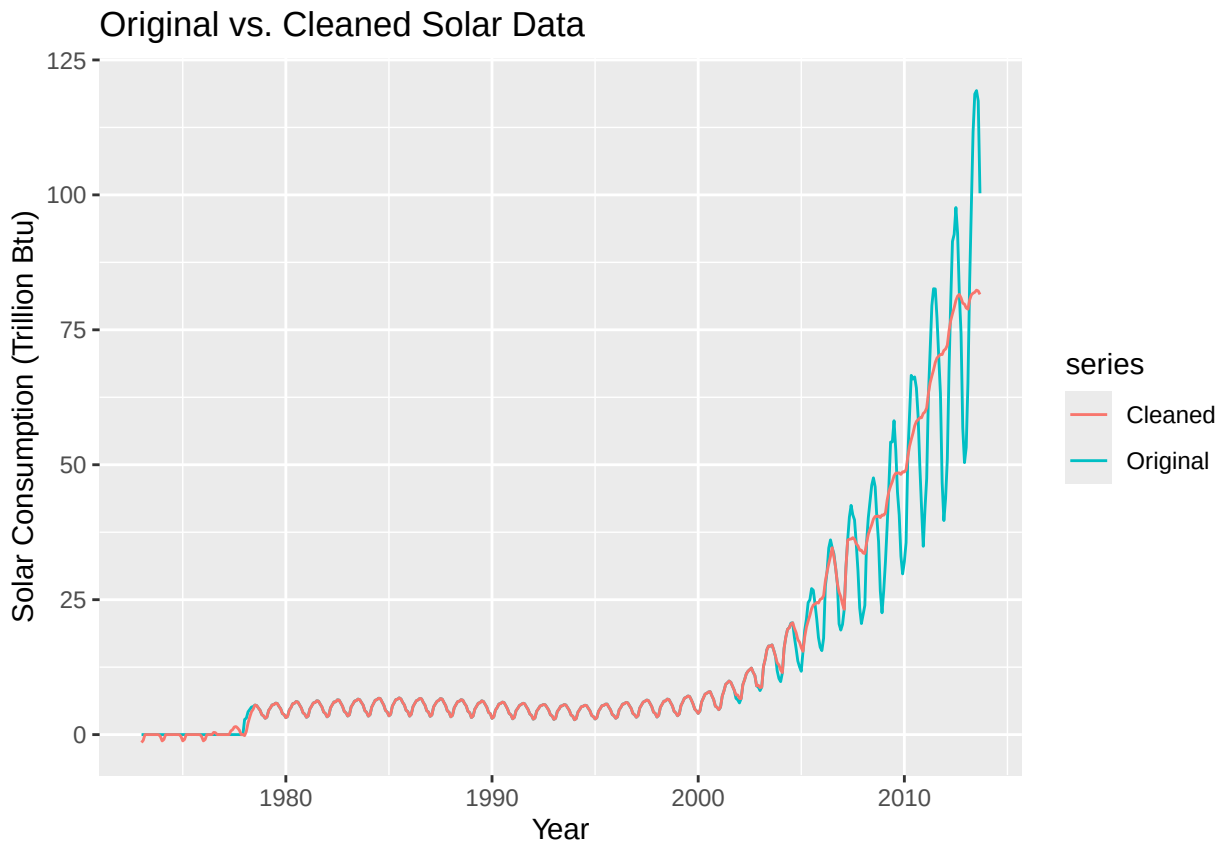
A similar analysis can be applied to wind, as variability in the random component also becomes less constant when wind consumption increases around 2040. This variation looks less seasonal/constant than the solar example, but still shows some seasonal dependence so it doesn't look fully random. This makes sense because wind consumption is dependent on wind speeds and weather patterns, so as more wind energy is used, there will be more variability.

Identify and Remove outliers

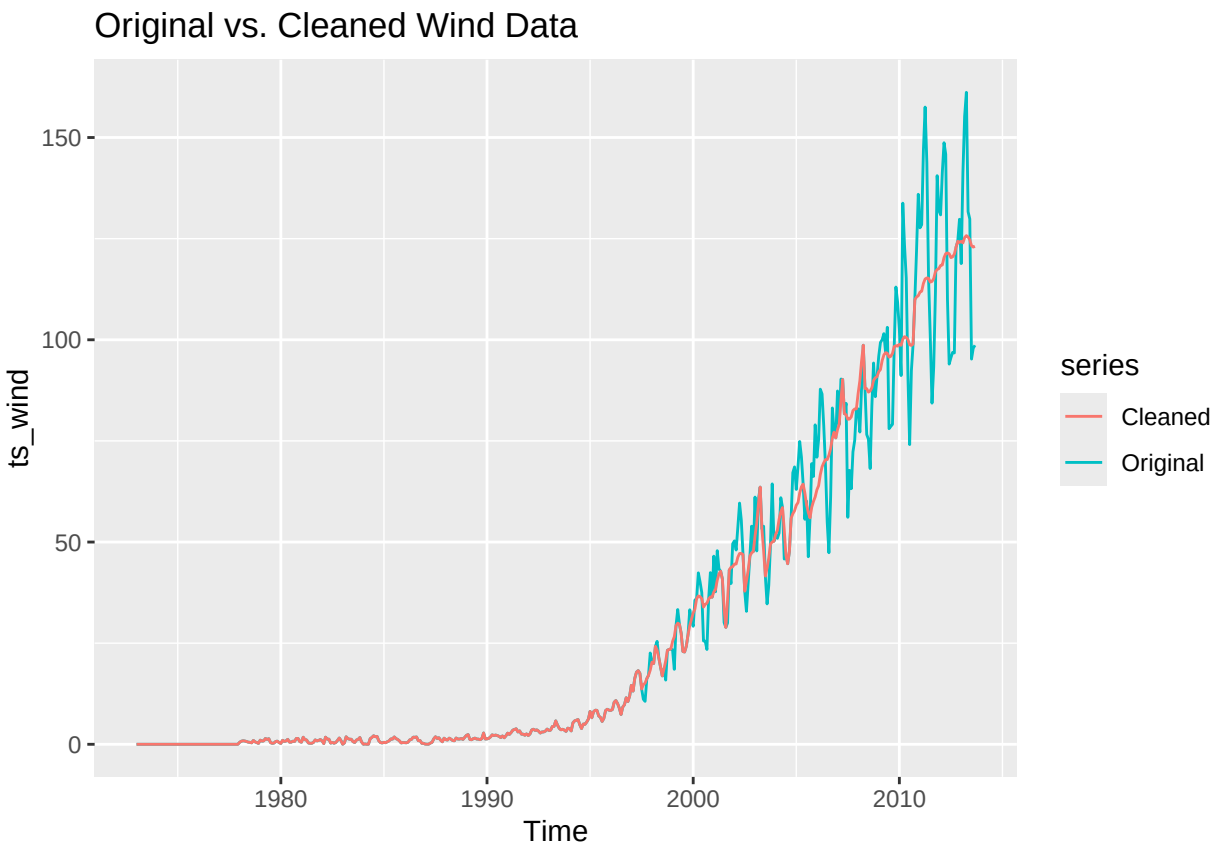
Q8

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.

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

autoplot(ts_solar,series="Original")+
  autolayer(clean_solar,series="Cleaned")+
  ggtitle("Original vs. Cleaned Solar Data")+
  ylab("Solar Consumption (Trillion Btu)")+
  xlab("Year")
```




```r
autoplot(ts_wind,series="Original")+
  autolayer(clean_wind,series="Cleaned")+
  ggtitle("Original vs. Cleaned Wind Data")
```

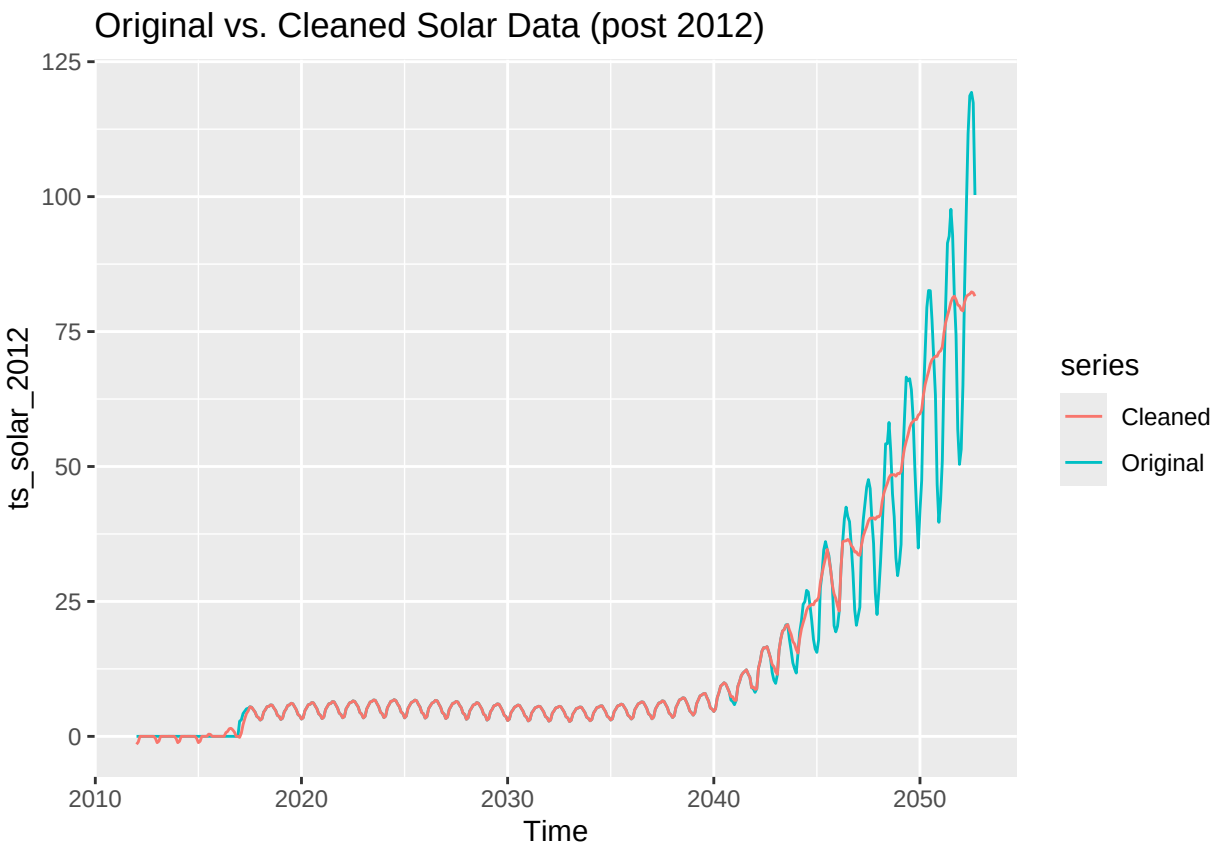


Yes, the function removed outliers from the series as there is a difference in the cleaned vs. original data. The original data has significantly larger peaks and troughs than the cleaned data.

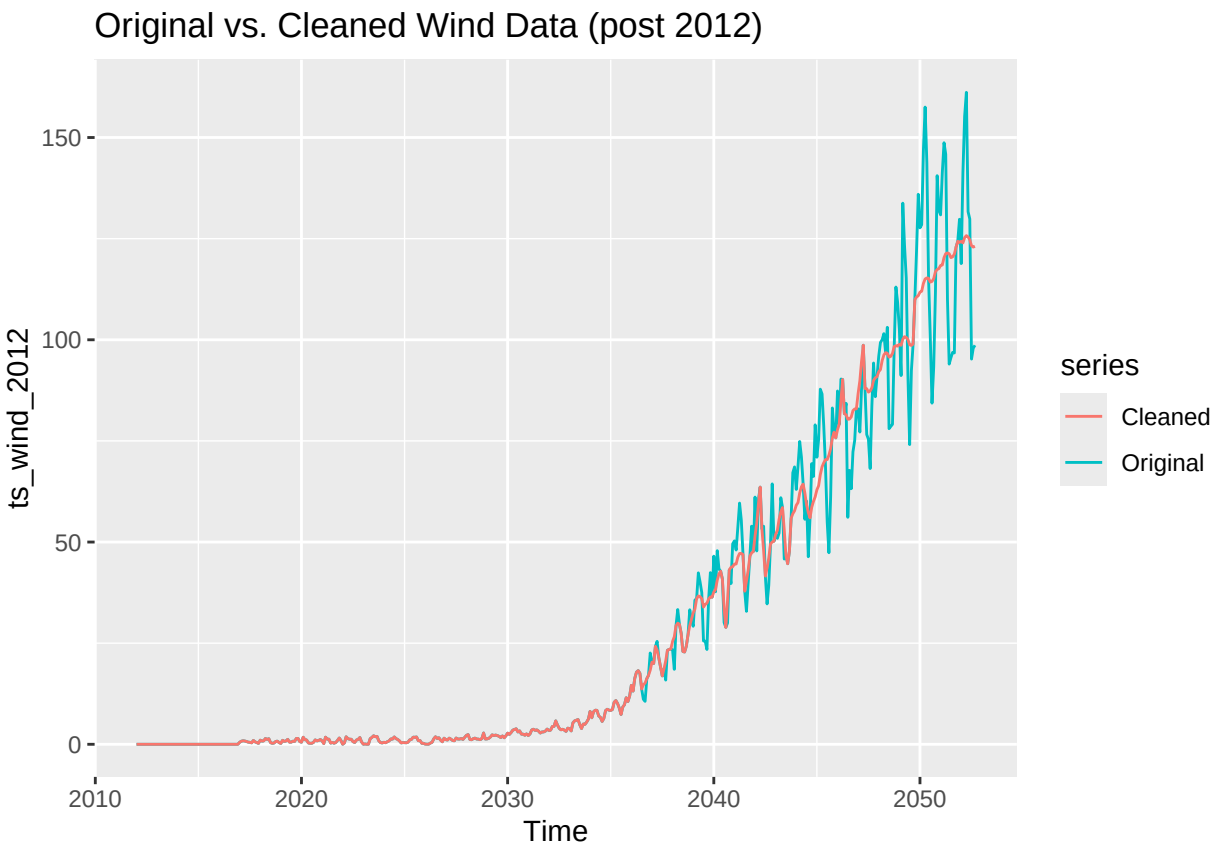
Q9

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?

```
clean_solar2012 <- tsclean(ts_solar_2012)
clean_wind2012 <- tsclean(ts_wind_2012)

autoplot(ts_solar_2012,series="Original")+
  autolayer(clean_solar2012,series="Cleaned")+
  ggtitle("Original vs. Cleaned Solar Data (post 2012)")
```




```
autoplot(ts_wind_2012,series="Original")+
  autolayer(clean_wind2012,series="Cleaned")+
  ggtitle("Original vs. Cleaned Wind Data (post 2012)")
```



Answer:

Yes, the function still removed outliers from the series at the end of the graph when there is more variation in the data. The original data has significantly larger peaks and troughs than the cleaned data