ENV 790.30 - Time Series Analysis for Energy Data | Spring 2024

Assignment 5 - Due date 02/13/24

Aditi Jackson

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_Sp23.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(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
```

```
library(scales)
library(cowplot)
```

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

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

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

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr   1.1.4     v stringr 1.5.0
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.2     v tidyr   1.3.0
## v readr   2.1.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x cowplot::stamp()    masks lubridate::stamp()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

Decomposing Time Series

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 2023 Monthly Energy Review.

```r
# loading data using read.csv; note that I tried readxl, but it kept producing errors
energy_data <-
  read.csv(
  "/home/guest/ENV797_APJ_S24_NEW/Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source
  header = TRUE, dec = ".", sep=",",stringsAsFactors = TRUE)

# renaming the "Month" column to "Date"
colnames(energy_data)[colnames(energy_data) == "Month"] <- "Date_MY"

# converting the "Date" column to a date object using lubridate
energy_data$Date_MY <- ym(energy_data$Date_MY)

# Verifying data
head(energy_data)
```

```
##      Date_MY Wood.Energy.Production Biofuels.Production
## 1 1973-01-01                129.630       Not Available
## 2 1973-02-01                117.194       Not Available
## 3 1973-03-01                129.763       Not Available
## 4 1973-04-01                125.462       Not Available
## 5 1973-05-01                129.624       Not Available
## 6 1973-06-01                125.435       Not Available
##   Total.Biomass.Energy.Production Total.Renewable.Energy.Production
## 1                         129.787                           219.839
## 2                         117.338                           197.330
## 3                         129.938                           218.686
## 4                         125.636                           209.330
## 5                         129.834                           215.982
## 6                         125.611                           208.249
##   Hydroelectric.Power.Consumption Geothermal.Energy.Consumption
## 1                          89.562                         0.490
## 2                          79.544                         0.448
## 3                          88.284                         0.464
```

```
## 4                          83.152                         0.542
## 5                          85.643                         0.505
## 6                          82.060                         0.579
##   Solar.Energy.Consumption Wind.Energy.Consumption Wood.Energy.Consumption
## 1            Not Available           Not Available                 129.630
## 2            Not Available           Not Available                 117.194
## 3            Not Available           Not Available                 129.763
## 4            Not Available           Not Available                 125.462
## 5            Not Available           Not Available                 129.624
## 6            Not Available           Not Available                 125.435
##   Waste.Energy.Consumption Biofuels.Consumption
## 1                    0.157        Not Available
## 2                    0.144        Not Available
## 3                    0.176        Not Available
## 4                    0.174        Not Available
## 5                    0.210        Not Available
## 6                    0.176        Not Available
##   Total.Biomass.Energy.Consumption Total.Renewable.Energy.Consumption
## 1                          129.787                            219.839
## 2                          117.338                            197.330
## 3                          129.938                            218.686
## 4                          125.636                            209.330
## 5                          129.834                            215.982
## 6                          125.611                            208.249
```

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!

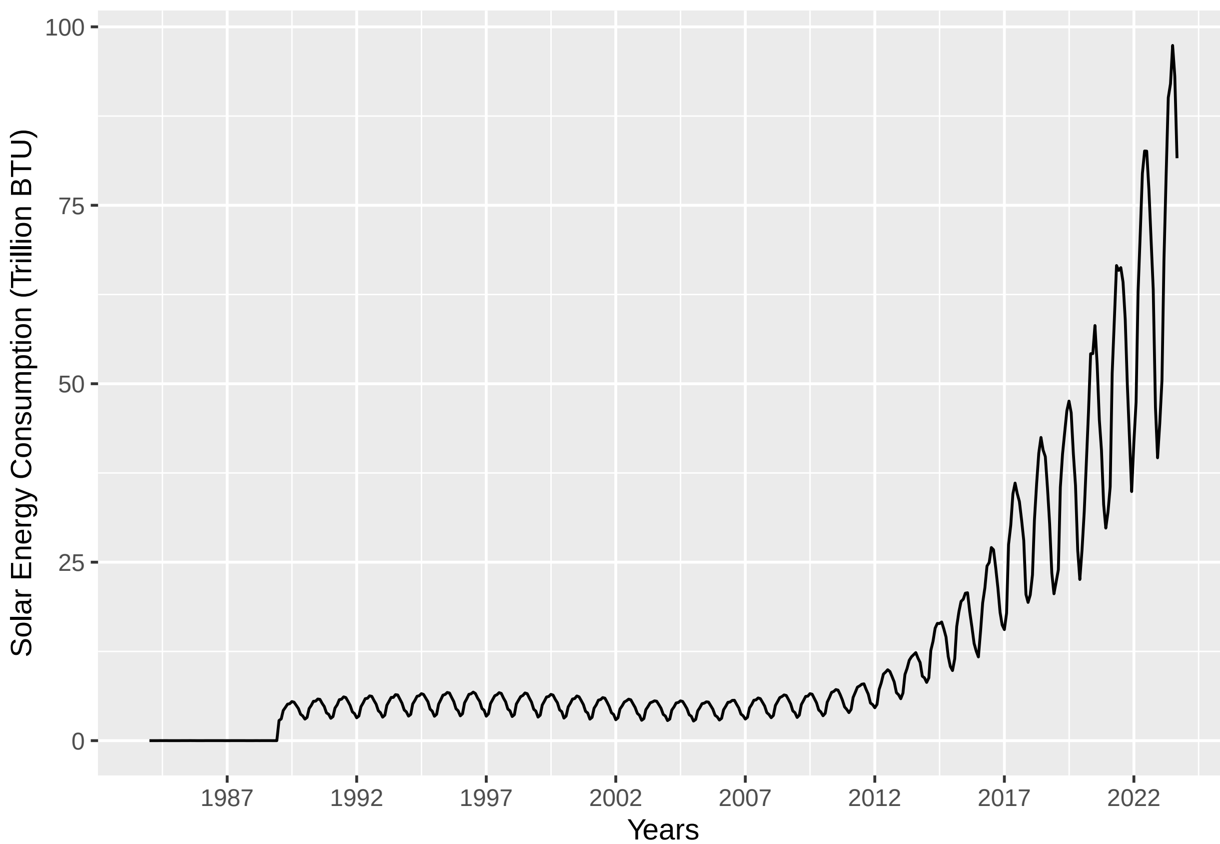
```r
# subsetting data into data frame with three variables: Date, Solar Energy Consumption
## and Wind Energy Consumption

# using mutate to convert solar and wind columns to character types in order to
## change data showing as "Not Available" to variable type that na_if function accepts

# dropping N/As and converting to numeric

# using drop_na() to take care of any N/A values
energy_data_sub <- energy_data %>%
  select(
    Date_MY,
    Solar.Energy.Consumption,
    Wind.Energy.Consumption) %>%
  mutate(
    Solar.Energy.Consumption = as.character(Solar.Energy.Consumption),
    Solar.Energy.Consumption = na_if(Solar.Energy.Consumption, "Not Available"),
    Wind.Energy.Consumption = as.character(Wind.Energy.Consumption),
    Wind.Energy.Consumption = na_if(Wind.Energy.Consumption, "Not Available")
  ) %>%
  drop_na()%>%
  mutate(
```

```
    Solar.Energy.Consumption = as.numeric(Solar.Energy.Consumption),
    Wind.Energy.Consumption = as.numeric(Wind.Energy.Consumption))
```

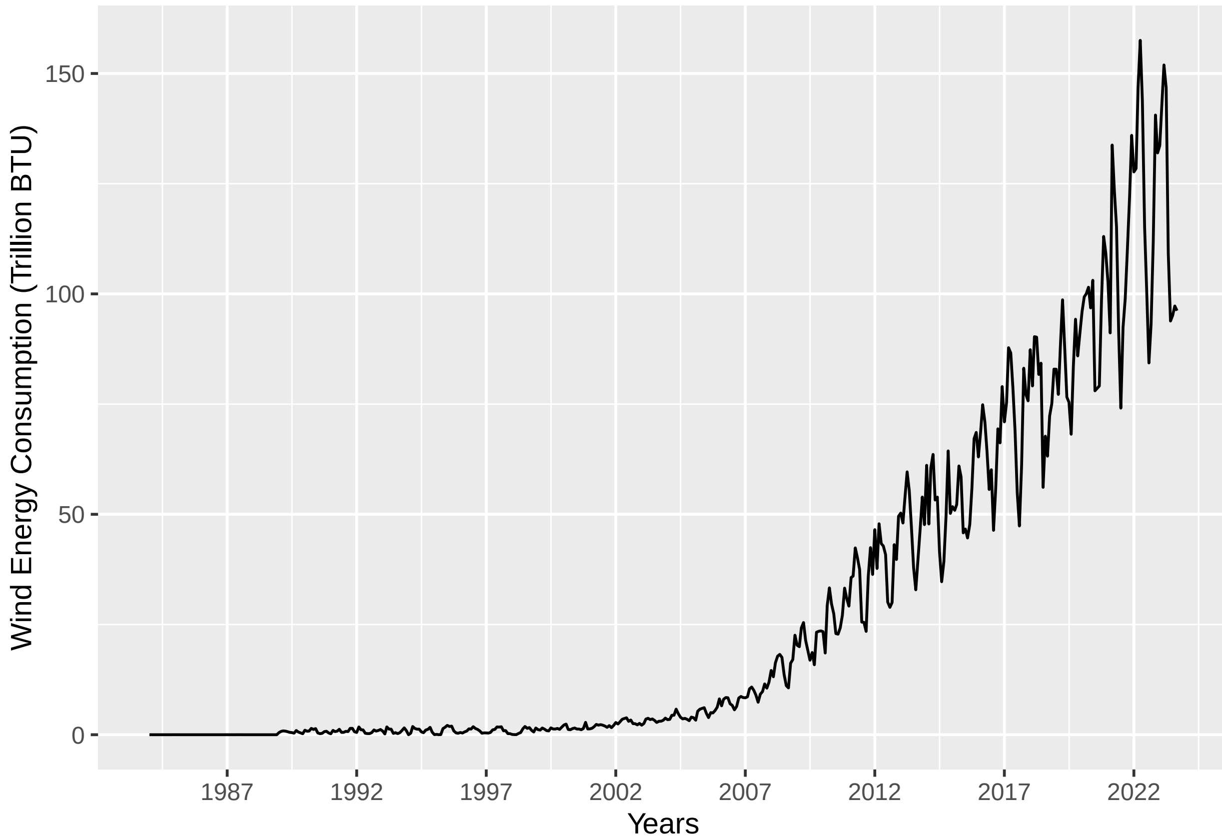
Q2

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"))`

```
# plotting solar
ggplot(energy_data_sub,aes(x=Date_MY,y=Solar.Energy.Consumption))+
  geom_line()+
  ylab("Solar Energy Consumption (Trillion BTU)")+
  scale_x_date(date_breaks = "5 years", labels = date_format("%Y"))+
  xlab("Years")
```



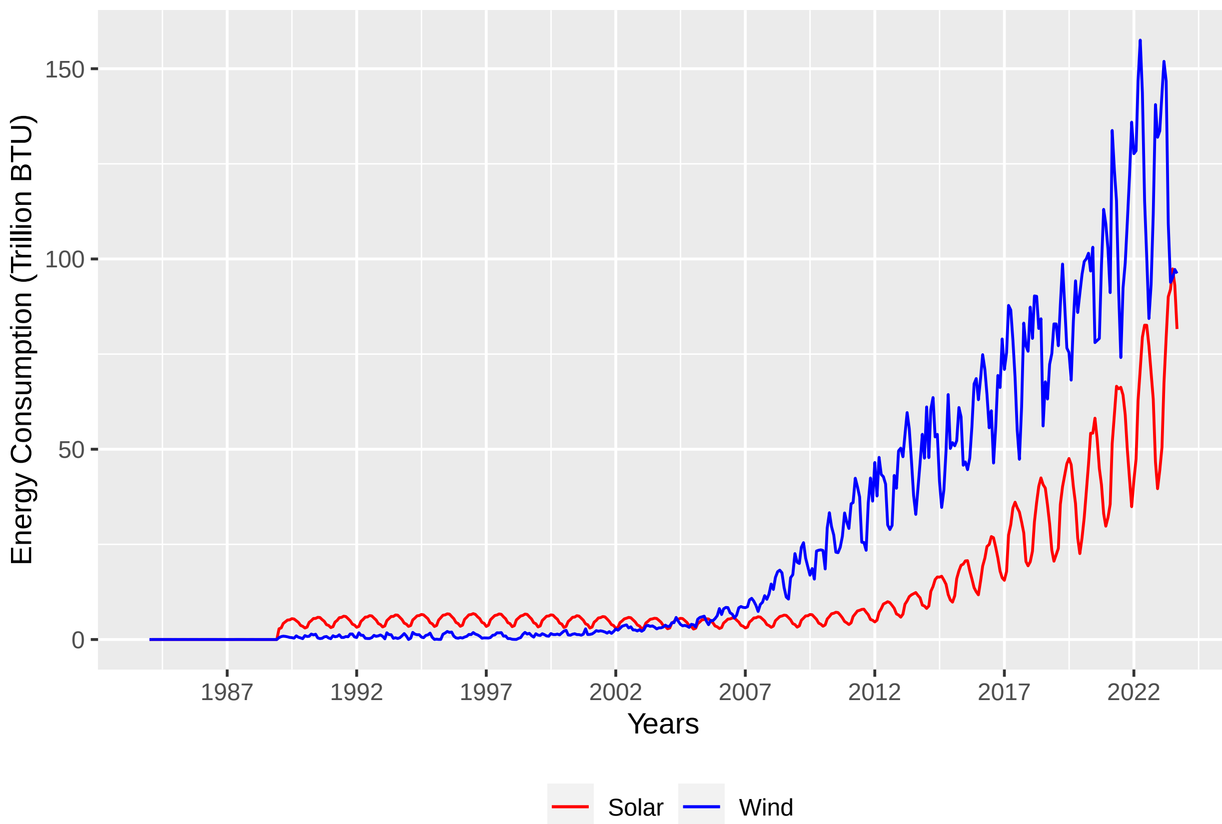

```
# plotting wind
ggplot(energy_data_sub,aes(x=Date_MY,y=Wind.Energy.Consumption))+
  geom_line()+
  ylab("Wind Energy Consumption (Trillion BTU)")+
  scale_x_date(date_breaks = "5 years", labels = date_format("%Y"))+
  xlab("Years")
```



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.

```
# plotting solar & wind together
ggplot(energy_data_sub,aes(x=Date_MY))+
  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"))+
   theme(legend.title = element_blank(), legend.position = "bottom")+
  scale_x_date(date_breaks = "5 years", labels = date_format("%Y"))+
  labs(x="Years",y="Energy Consumption (Trillion BTU)")
```



Decomposing the time series

The stats package has a function called decompose(). This function only takes 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()`.

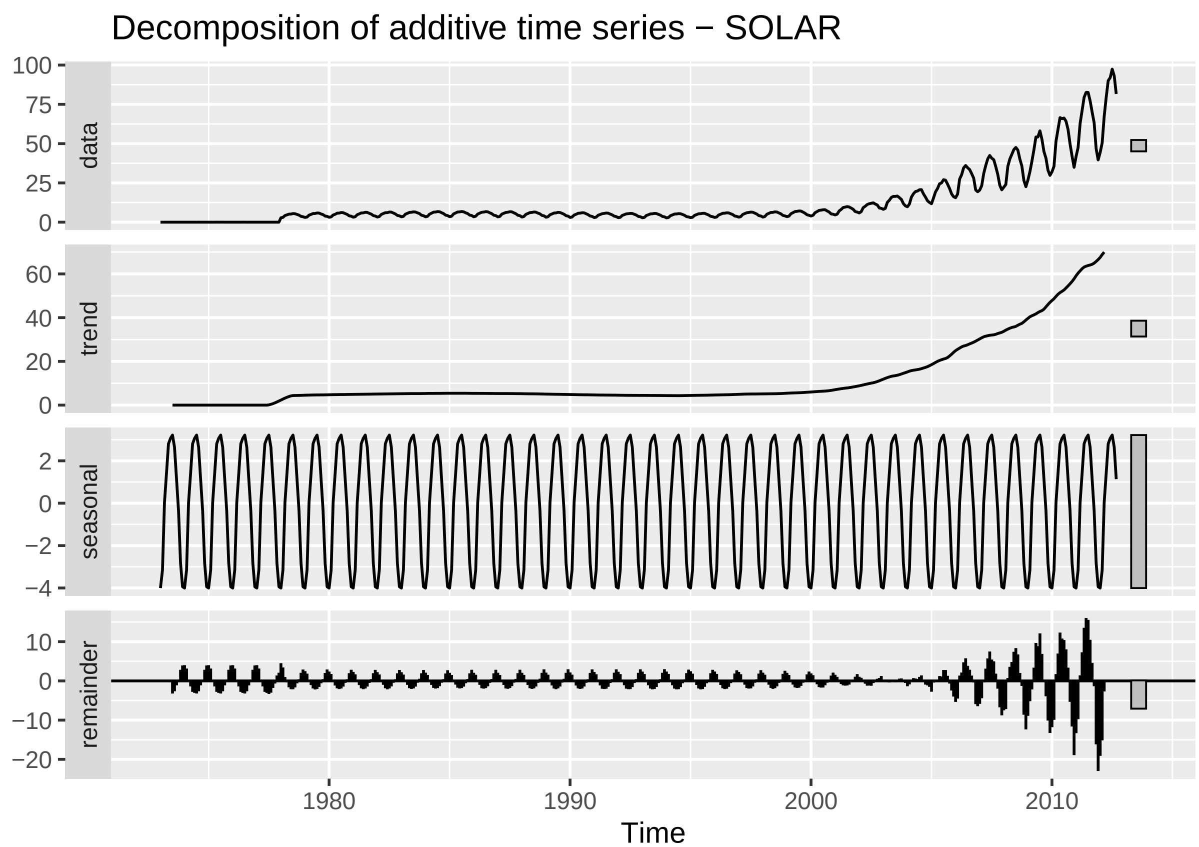
1) You have two options: alternative and multiplicative. Multiplicative models exhibit a change in frequency over time.
2) The trend is not a straight line because it uses a moving average method to detect trend.
3) The seasonal component of the time series is found by subtracting the trend component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

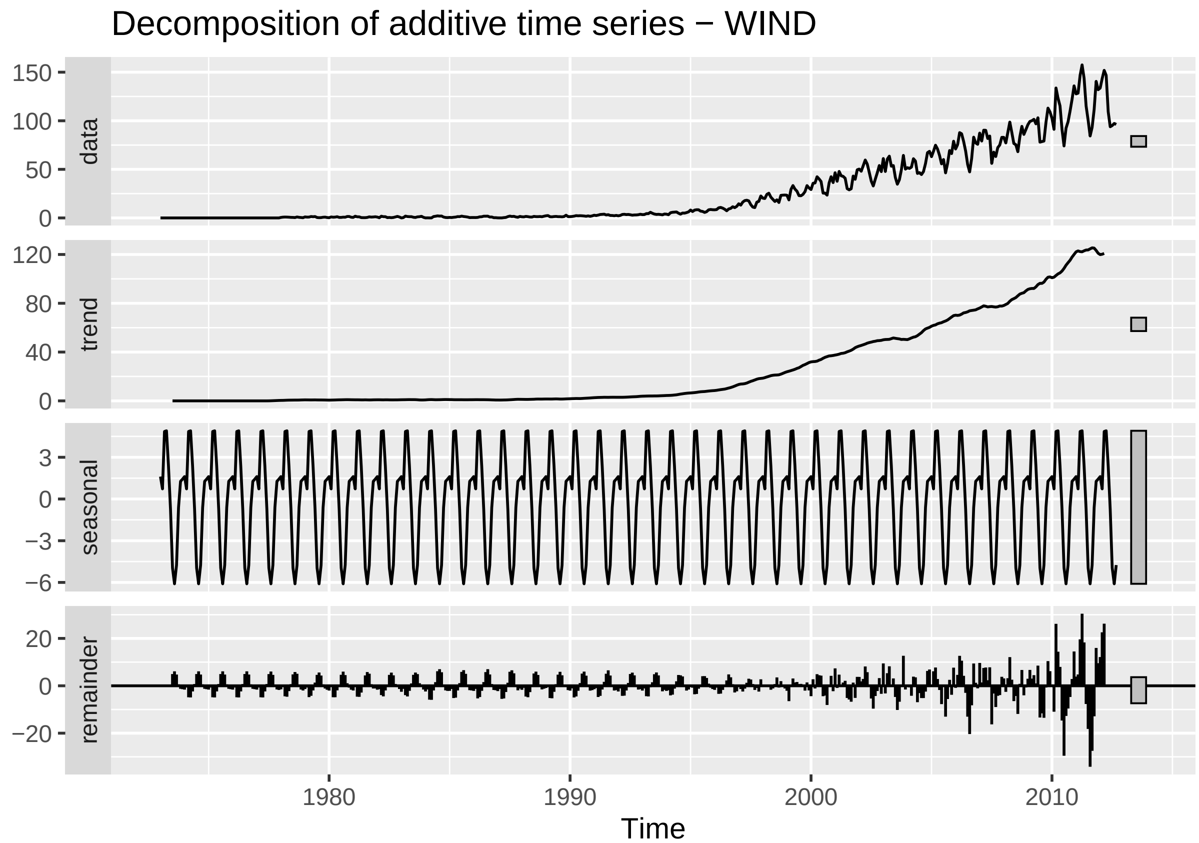
Transform wind and solar series into a time series object and apply the decompose function on them using the additive option, i.e., `decompose(ts_data, type = "additive")`. What can you say about the trend component? What about the random component? Does the random component look random? Or does it appear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

```
# creating time series objects fo solar and wind
ts_solar = ts(energy_data_sub[,2],start=c(1973,1),frequency=12)
ts_wind = ts(energy_data_sub[,3],start=c(1973,1),frequency=12)
```

```r
# decomposing solar and wind using additive type
decomp_add_solar <- decompose(ts_solar, type = "additive")
decomp_add_wind <- decompose(ts_wind, type = "additive")

# plotting decomposed solar and wind objects
autoplot(decomp_add_solar)+
  ggtitle("Decomposition of additive time series - SOLAR")
```




```r
autoplot(decomp_add_wind)+
    ggtitle("Decomposition of additive time series - WIND")
```



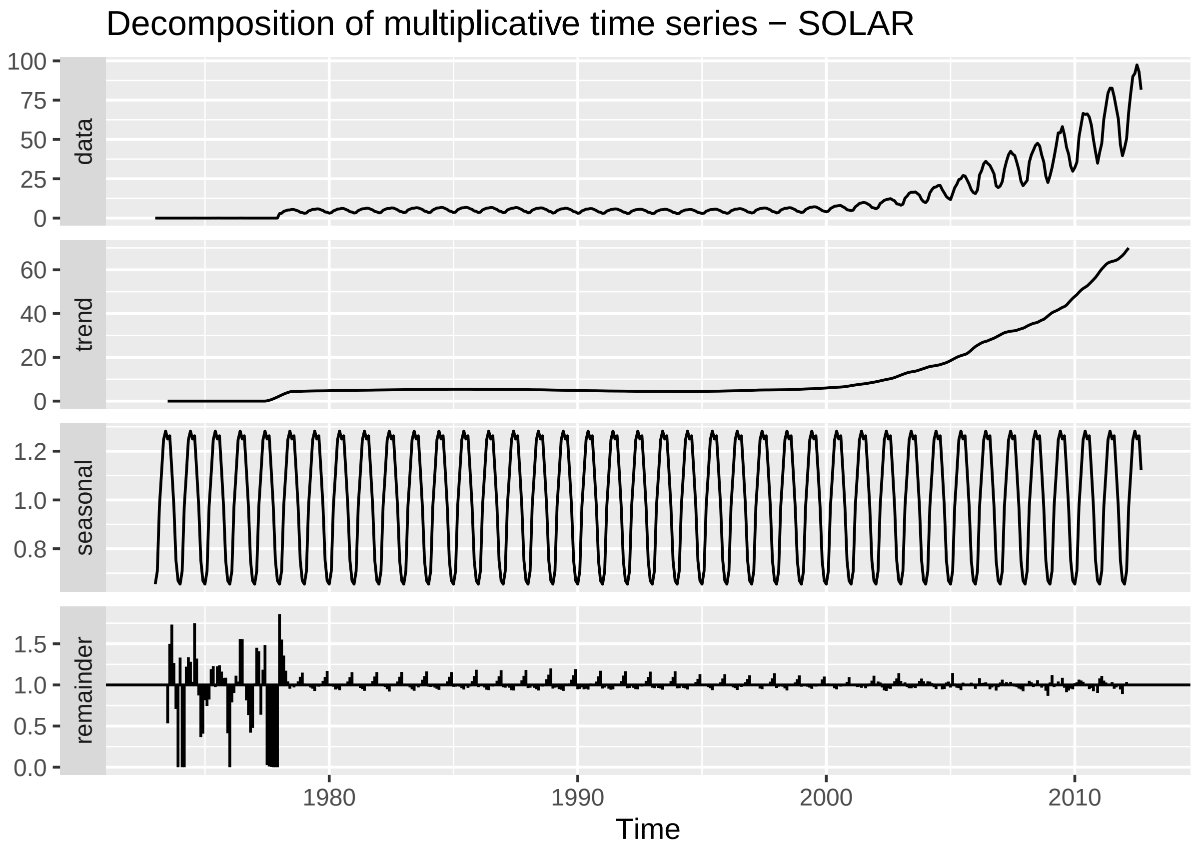
SOLAR: The trend component of solar shows a generally increasing non-linearly (trend appears exopnential). The random component of solar does not appear to be truly "random" There seems to be some seasonality present given the equally spaced sine wave pattern over time.

WIND: The trend component of wind shows a steadily increasing upward slow, which looks more linear than solar (but I'm not 100% certain that we can is monotonic). The random component of wind also appears to have some seasonality still present (equally spaced sine waves) - at least for most of the 1970s through mid 1990s. However, in the later 1990s it seems as if the randomness increases.

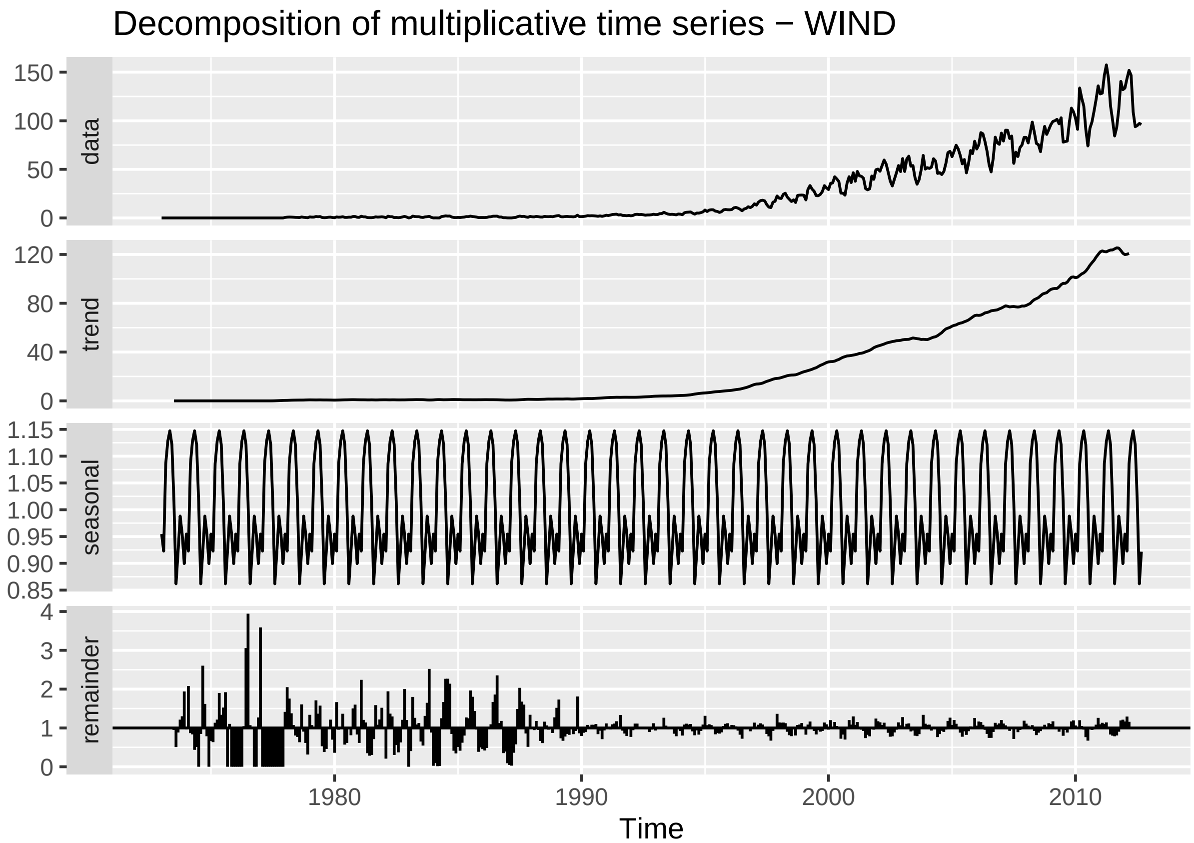
Q5

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?

```
# decomposing solar and wind using multiplicative type
decomp_mult_solar <- decompose(ts_solar, type = "multiplicative")
decomp_mult_wind <- decompose(ts_wind, type = "multiplicative")

# plotting decomposed solar and wind objects
autoplot(decomp_mult_solar)+
    ggtitle("Decomposition of multiplicative time series - SOLAR")
```




```
autoplot(decomp_mult_wind)+
 ggtitle("Decomposition of multiplicative time series - WIND")
```



For both wind and solar, there seems to be less residual seasonality present in the random component when using the multiplicative model.

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.

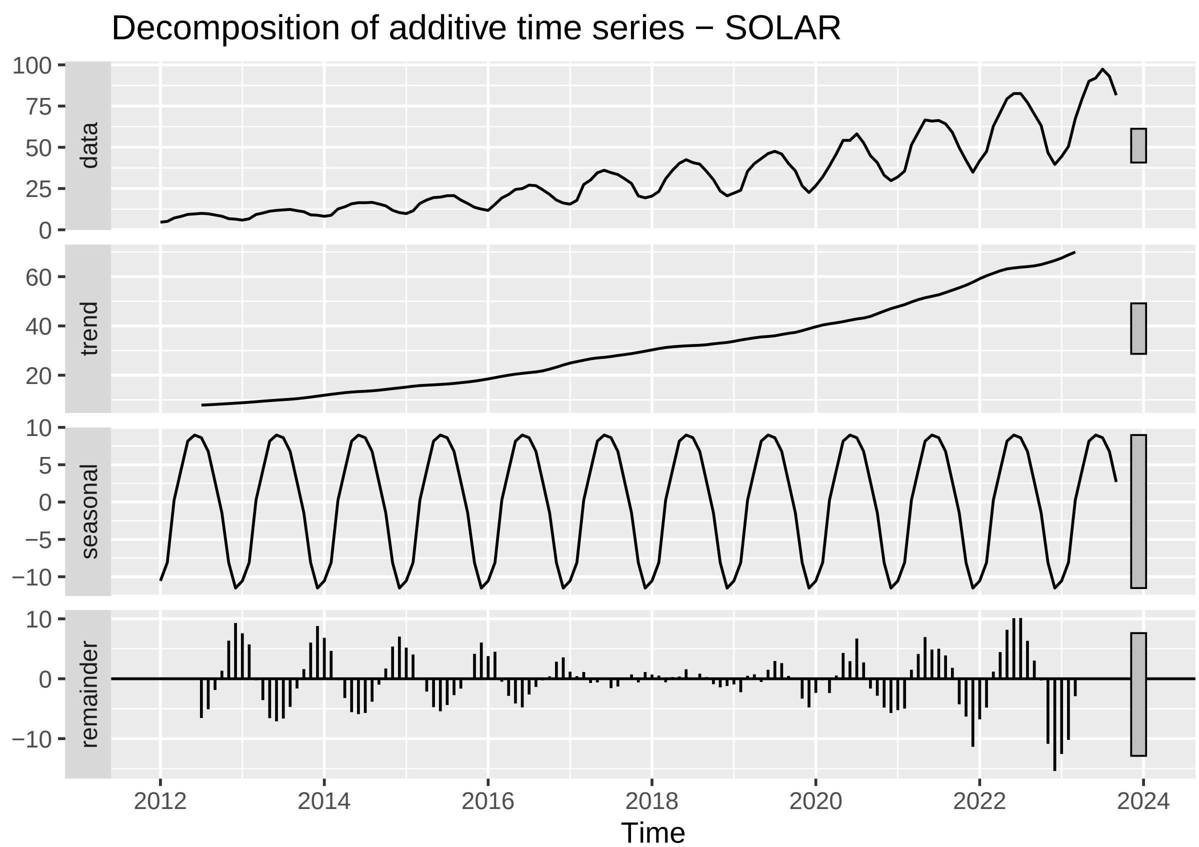
> Answer: I do not think we need ALL the historical data to fit a model. This is because the trends in both solar and wind consumption change quite drastically from the 1990s and early 2000s, going from relatively flat to positively increasing. If we are looking to forecast the next 6 months, we'd likely only need the data from the mid-2000s to present day since that data shows a relatively similar upward sloping trend pattern. And since forecasting depends on knowing what happened in the recent past to predict future patterns, data from before the mid-2000s wouldn't be additive to predicting future consumption patterns.

Q7

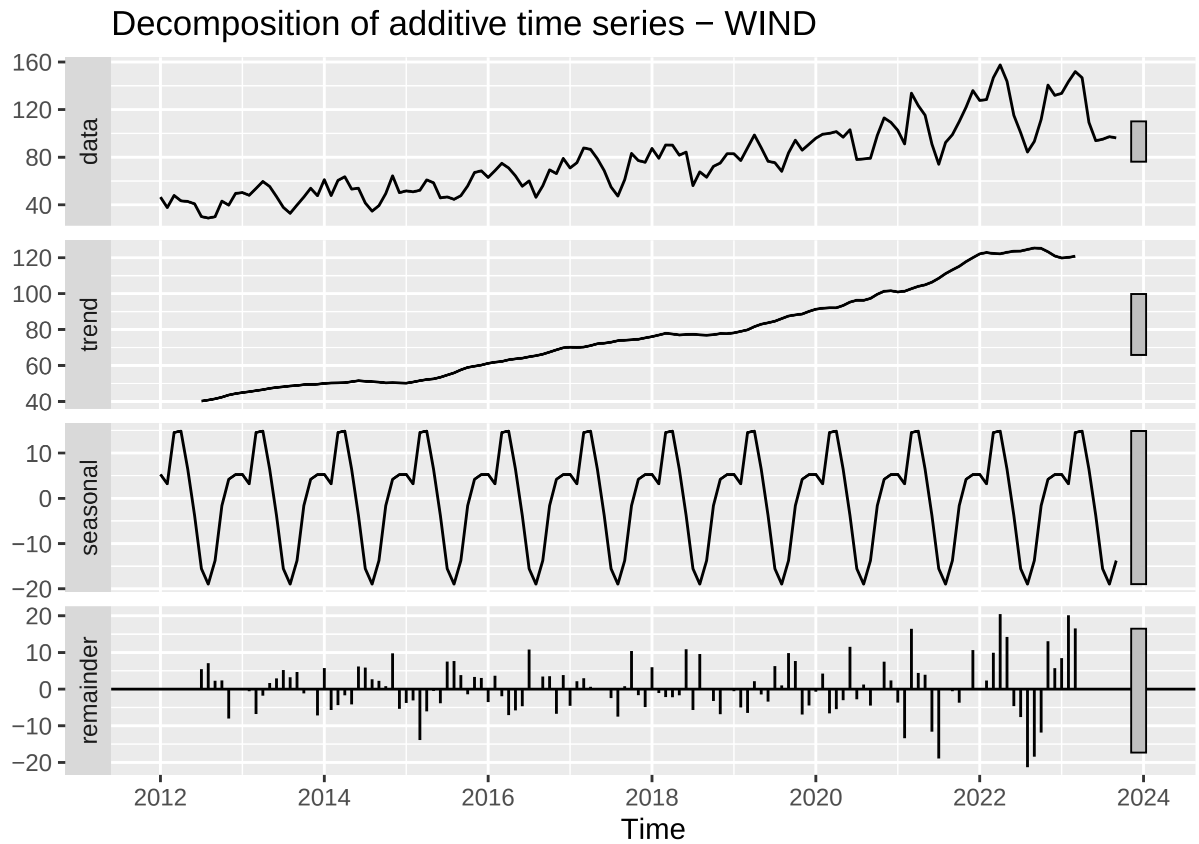
Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the decompose function `type=additive` to this new time series. Comment the results. Does the random component look random? Think about our discussion in class about seasonal components that depends on the level of the series.

```
# filtering data for Jan 2022 onwards
energy_data_sub2 <- energy_data_sub %>%
  filter(year(Date_MY)>=2012)
```

```r
# creating new ts objects for solar and wind
ts_solar2 <- ts(energy_data_sub2[,2],start=c(2012,1),frequency=12)
ts_wind2 <- ts(energy_data_sub2[,3],start=c(2012,1),frequency=12)

# creating new decompositions
decomp_add_solar2 <- decompose(ts_solar2, type = "additive")
decomp_add_wind2 <- decompose(ts_wind2, type = "additive")

# plotting decomposed time series
autoplot(decomp_add_solar2)+
ggtitle("Decomposition of additive time series - SOLAR")
```




```r
autoplot(decomp_add_wind2)+
  ggtitle("Decomposition of additive time series - WIND")
```



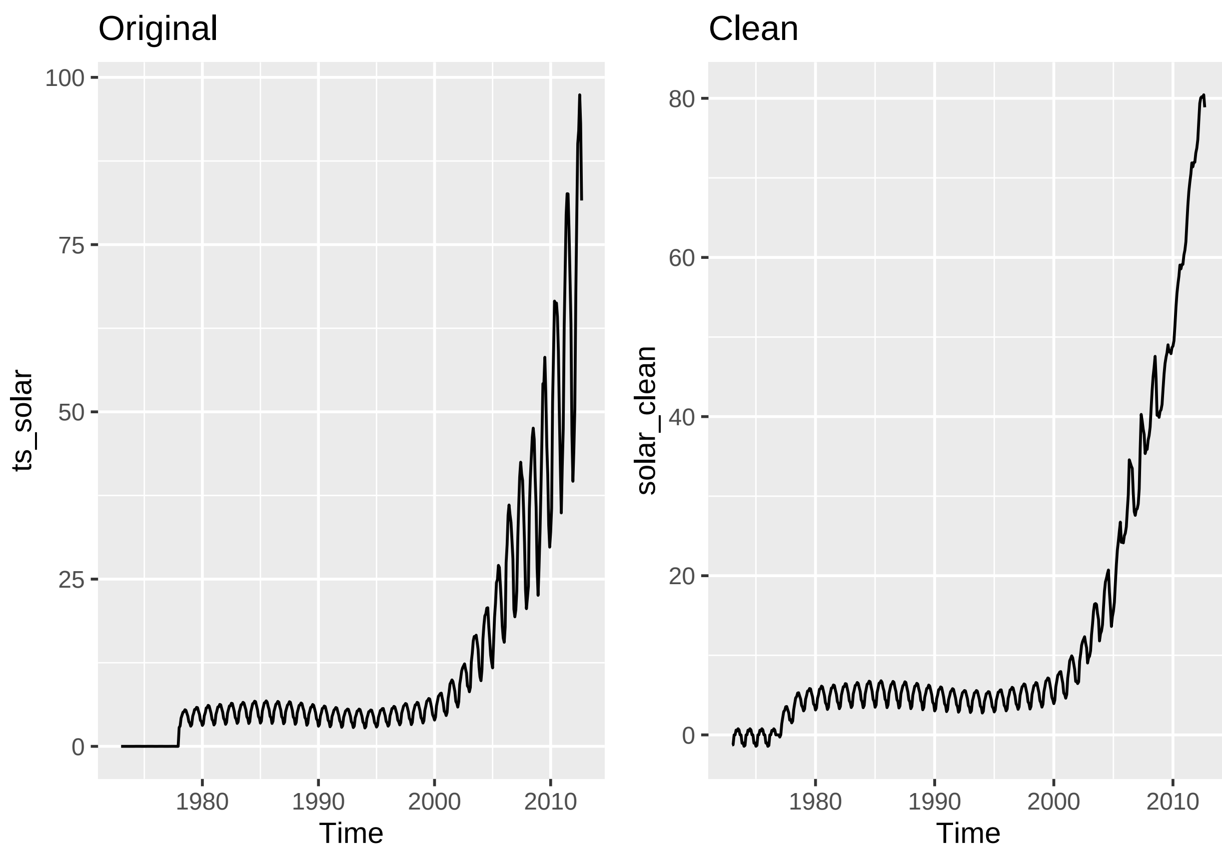
SOLAR: The trend is increasing. The random component still appears to be influenced by seasonality. This may be because seasonality is very prominent in this series. WIND: Similarly, there is an increasing trend. The random component appears to be be more "random" compared to the decompositions in the previous questions.

Identify and Remove outliers

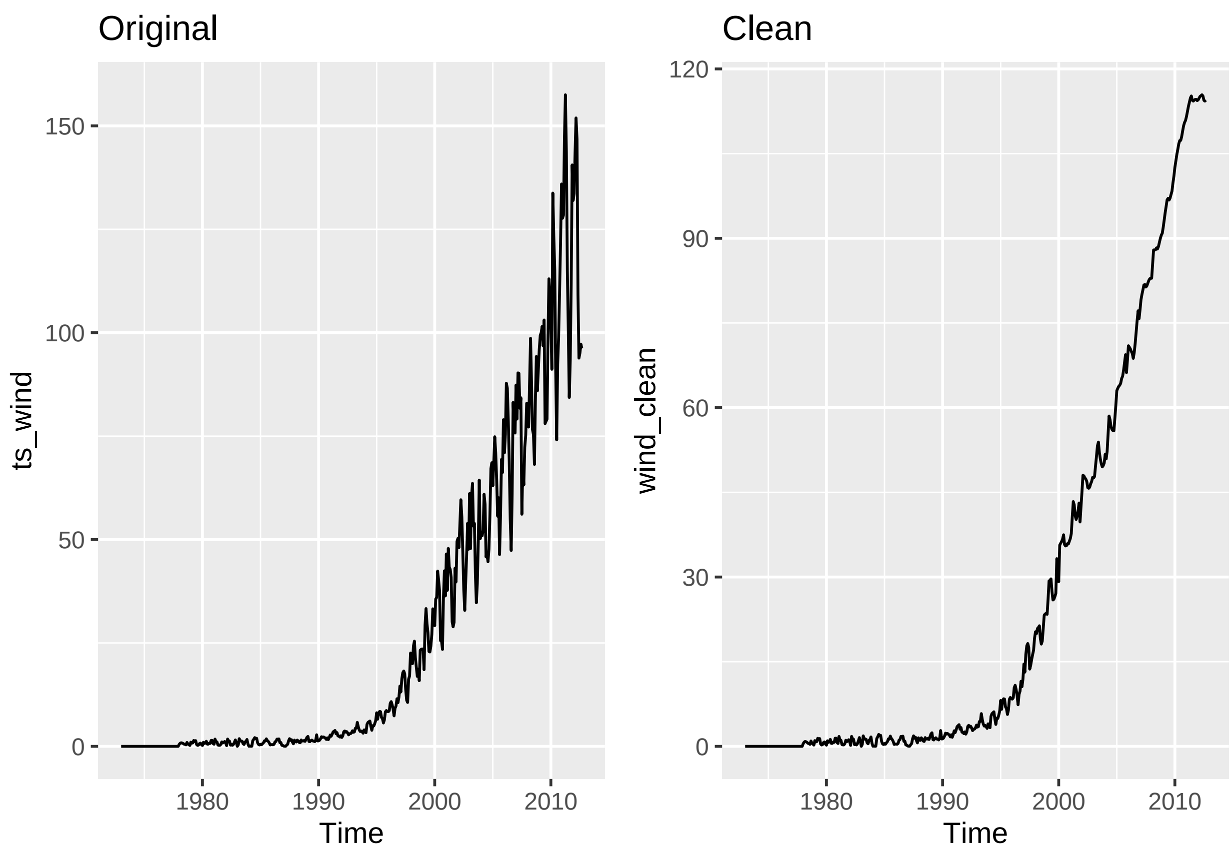
Q8

Apply the `tsclean()` to both series from 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.

```
# removing outliers with tsclean()
solar_clean <- tsclean(ts_solar)
wind_clean <- tsclean(ts_wind)

# plotting "cleaned" series vs. original series
plot_grid(autoplot(ts_solar)+
            ggtitle("Original"),
          autoplot(solar_clean)+
            ggtitle("Clean"))
```




```
plot_grid(autoplot(ts_wind)+
            ggtitle("Original"),
          autoplot(wind_clean)+
            ggtitle("Clean"))
```

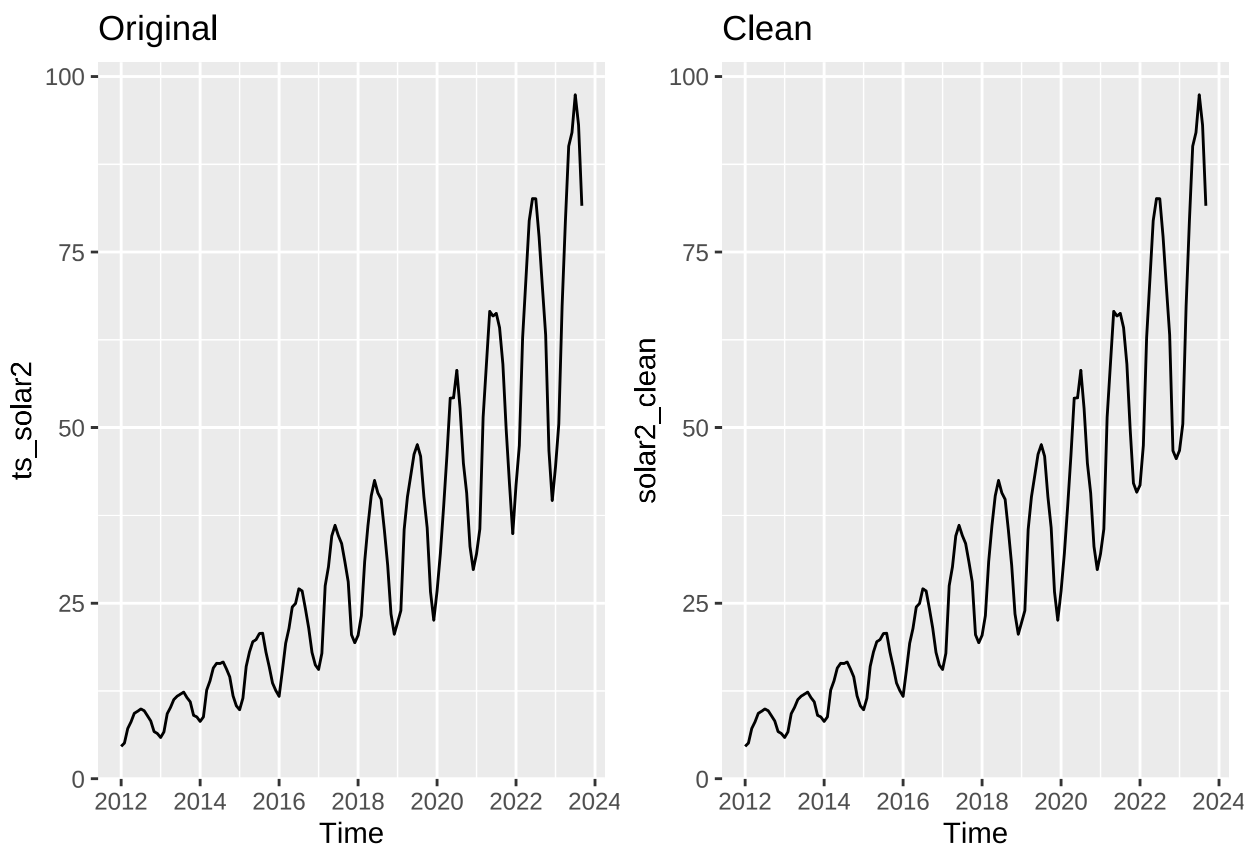


Yes, it appears that the ts_clean pacakge removed outliers from both the solar and wind time series. For both series we see less variation in applitudes when looking at the clean series compared to the original series.

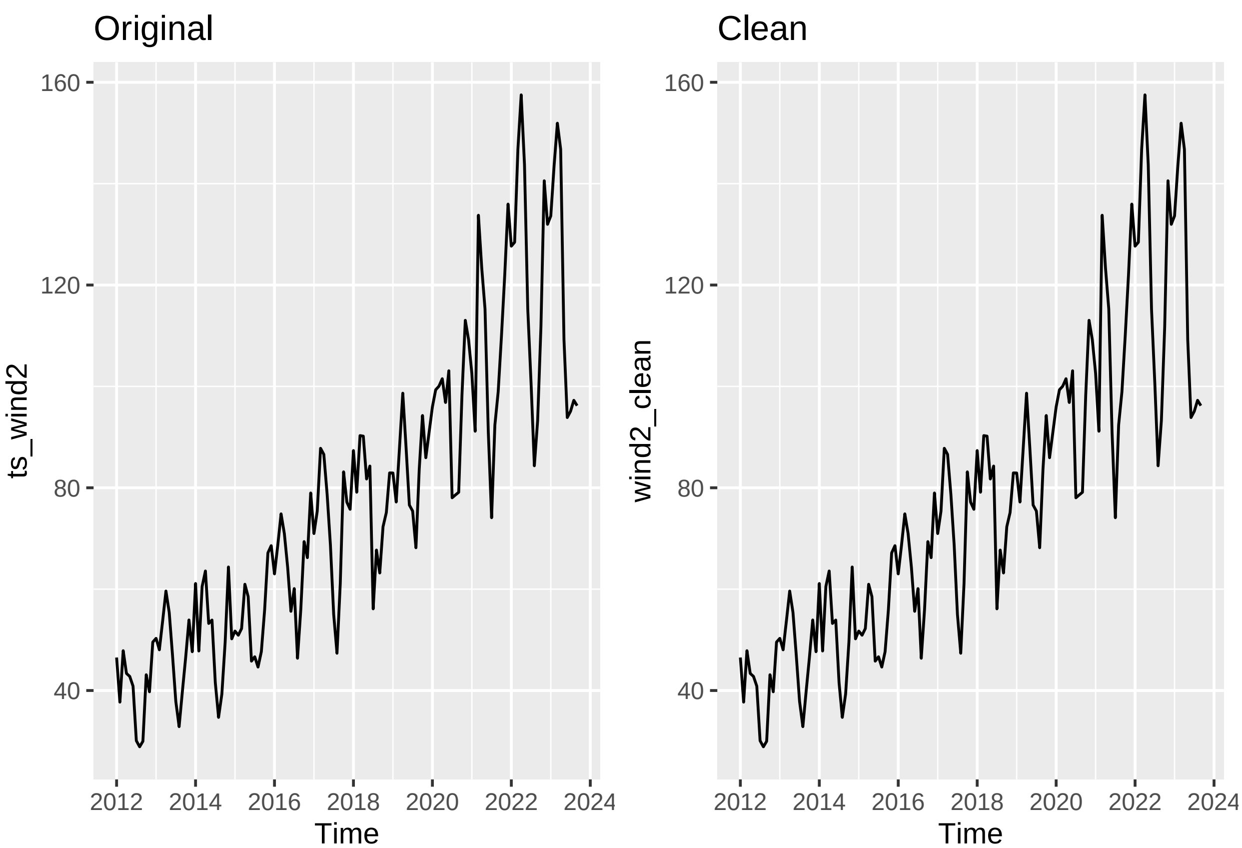
Q9

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

```
# removing outliers with tsclean()
solar2_clean <- tsclean(ts_solar2)
wind2_clean <- tsclean(ts_wind2)

# plotting "cleaned" series vs. "original" series
plot_grid(autoplot(ts_solar2)+
            ggtitle("Original"),
          autoplot(solar2_clean)+
            ggtitle("Clean"))
```




```r
plot_grid(autoplot(ts_wind2)+
            ggtitle("Original"),
          autoplot(wind2_clean)+
            ggtitle("Clean"))
```



Answer: Using the time series from Q7 it does not appear that the ts_clean pacakge removed any outliers.