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

Assignment 5 - Due date 02/18/25

Jingze Dai

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(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(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
```

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 - using xlsx package
energy_data <- read.xlsx(
  xlsxFile="../Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx",
  colNames=FALSE,startRow = 13,sheet = "Monthly Data")

#Now let's extract the column names from row 11 only
read_col_names <- read.xlsx(
  xlsxFile="../Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx",
  colNames =FALSE, rows = 11,sheet = "Monthly Data")

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

```
##   Month Wood Energy Production Biofuels Production
## 1 26665                129.630       Not Available
## 2 26696                117.194       Not Available
## 3 26724                129.763       Not Available
## 4 26755                125.462       Not Available
## 5 26785                129.624       Not Available
## 6 26816                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
```

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

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
# converting into date format
energy_data$Date <- convertToDate(energy_data$Month)

# filtering out NA data and selecting columns of interest
renewables_data <- energy_data %>%
  filter(`Solar Energy Consumption` != "Not Available") %>%
  filter(`Wind Energy Consumption` != "Not Available") %>%
  select(Date,`Solar Energy Consumption`,`Wind Energy Consumption`)

# converting to numeric format
renewables_data <- renewables_data %>%
  rename(Solar = `Solar Energy Consumption`,
         Wind = `Wind Energy Consumption`) %>%
  mutate(Solar = as.numeric(Solar),
         Wind = as.numeric(Wind))

# creating time series
solar_ts <- ts(renewables_data[,2], start = c(1984,1), frequency = 12)
```

```r
wind_ts <- ts(renewables_data[,3], start = c(1984,1), frequency = 12)

# creating the dataframe
renewables_df <- data.frame(solar_ts, wind_ts, renewables_data$Date)
colnames(renewables_df) <- c("Solar_TS","Wind_TS", "Date")

head(renewables_df)
```

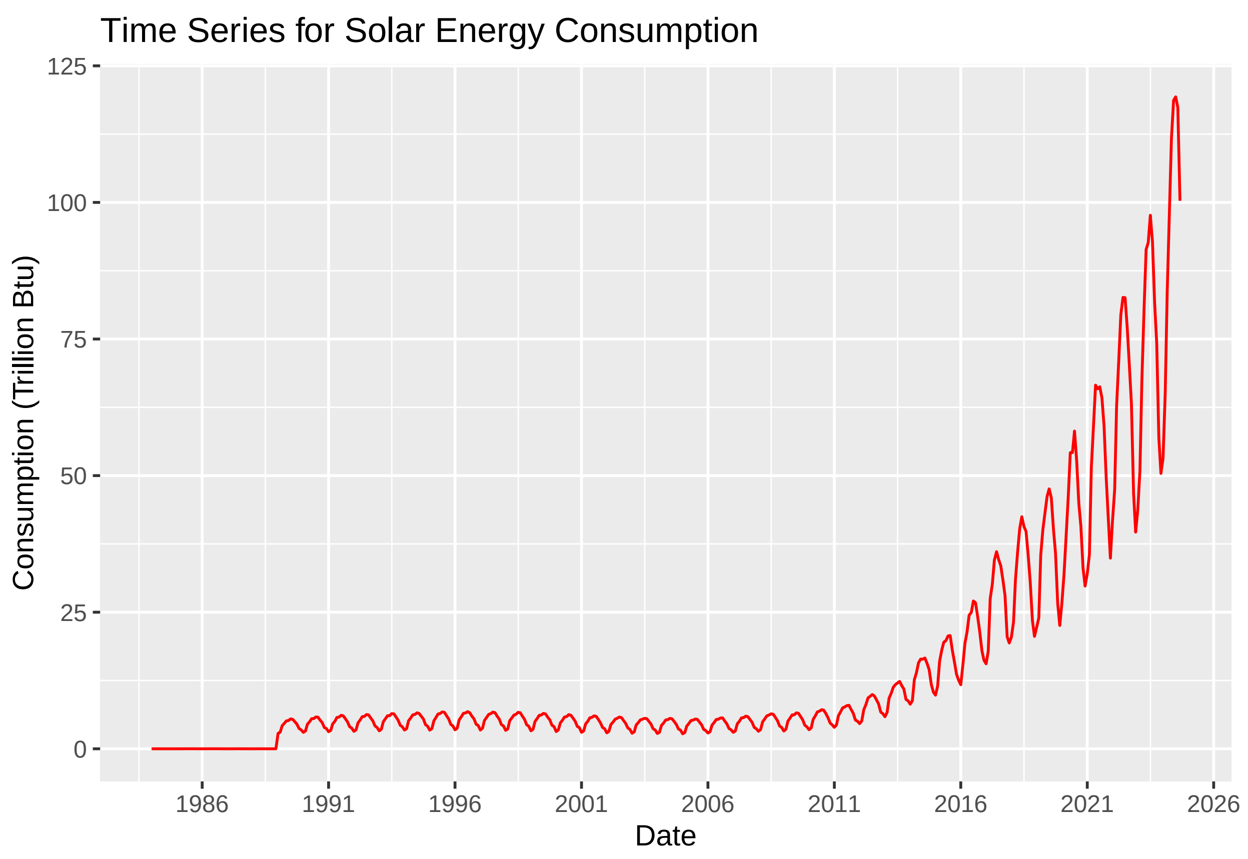
```
##   Solar_TS Wind_TS       Date
## 1    0.000   0.000 1984-01-01
## 2    0.000   0.001 1984-02-01
## 3    0.001   0.001 1984-03-01
## 4    0.001   0.002 1984-04-01
## 5    0.002   0.003 1984-05-01
## 6    0.003   0.002 1984-06-01
```

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")")`

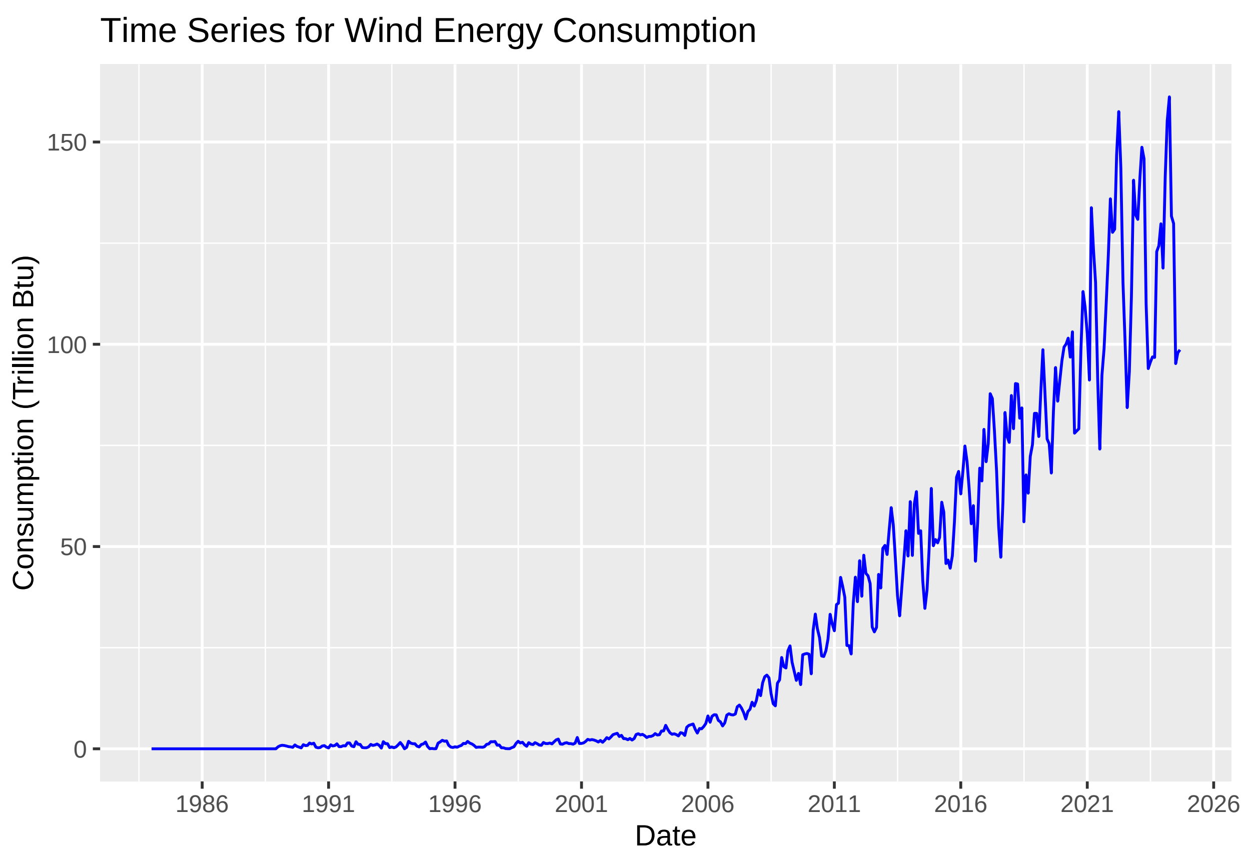
```r
# solar energy consumption
ggplot(data = renewables_df, aes(x=Date, y=Solar_TS)) +
  geom_line(color = "red") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ggtitle("Time Series for Solar Energy Consumption") +
  ylab("Consumption (Trillion Btu)")
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



```r
# wind energy consumption
ggplot(data = renewables_df, aes(x=Date, y=Wind_TS)) +
  geom_line(color = "blue") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ggtitle("Time Series for Wind Energy Consumption") +
  ylab("Consumption (Trillion Btu)")
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

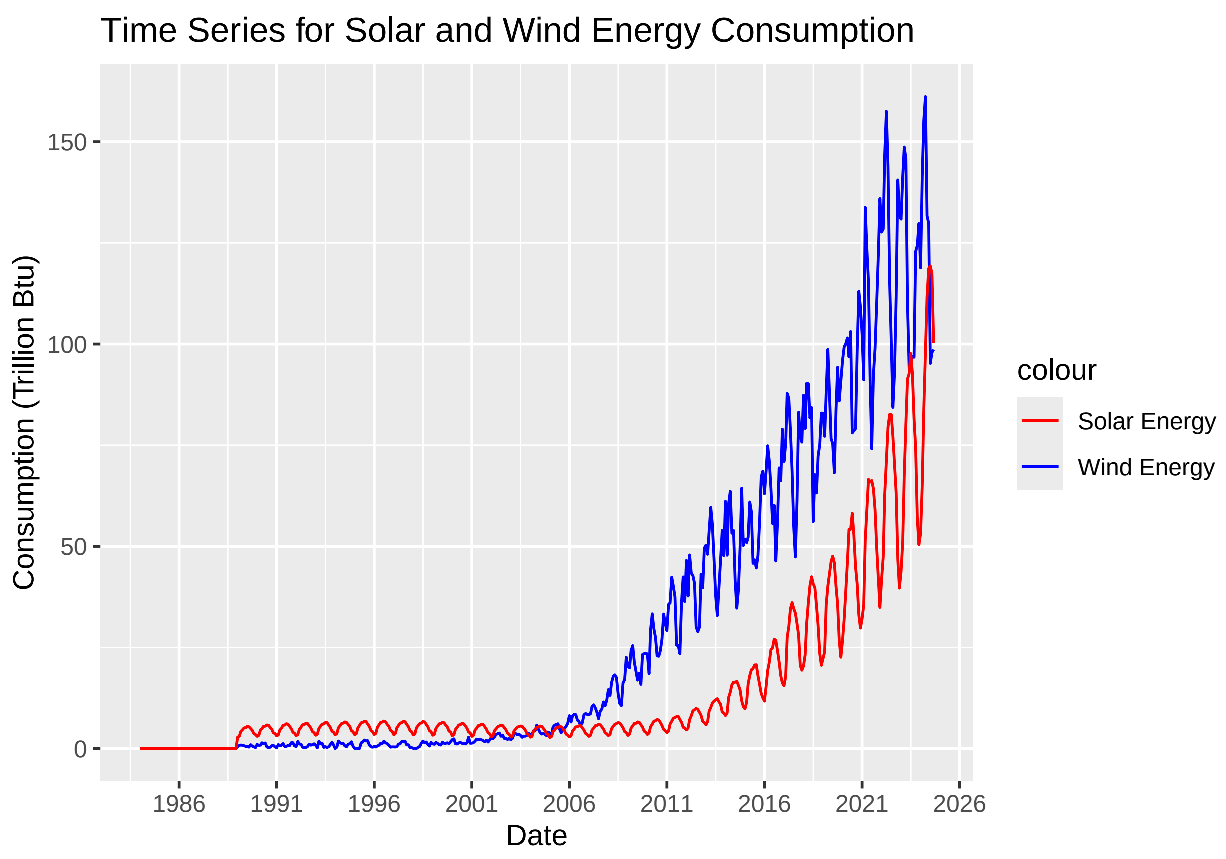


Q3

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.

```
ggplot(data = renewables_df, aes(x=Date)) +
  geom_line(aes(y=Wind_TS,col="Wind Energy")) +
  geom_line(aes(y=Solar_TS,col="Solar Energy")) +
  scale_color_manual(values = c("Wind Energy"="blue","Solar Energy"="red")) +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ggtitle("Time Series for Solar and Wind Energy Consumption") +
  ylab("Consumption (Trillion Btu)")
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



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.

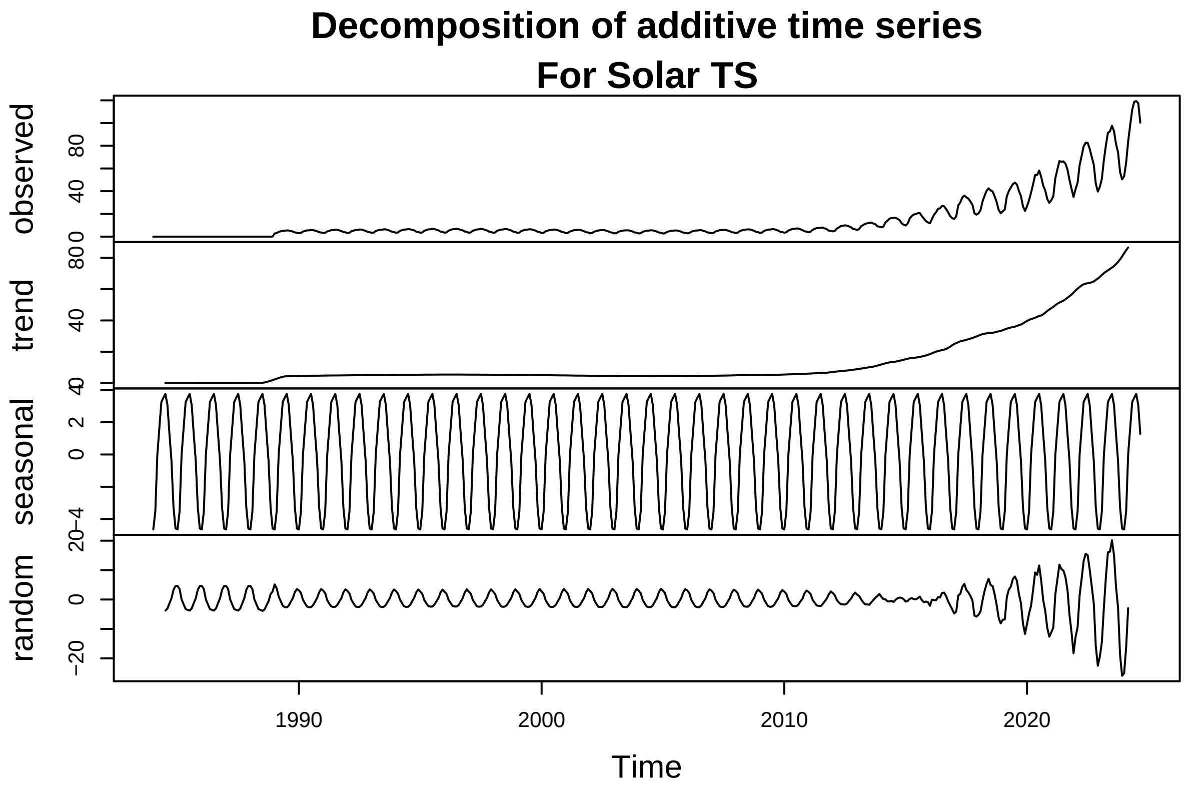
Additional info on `decompose()`.

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

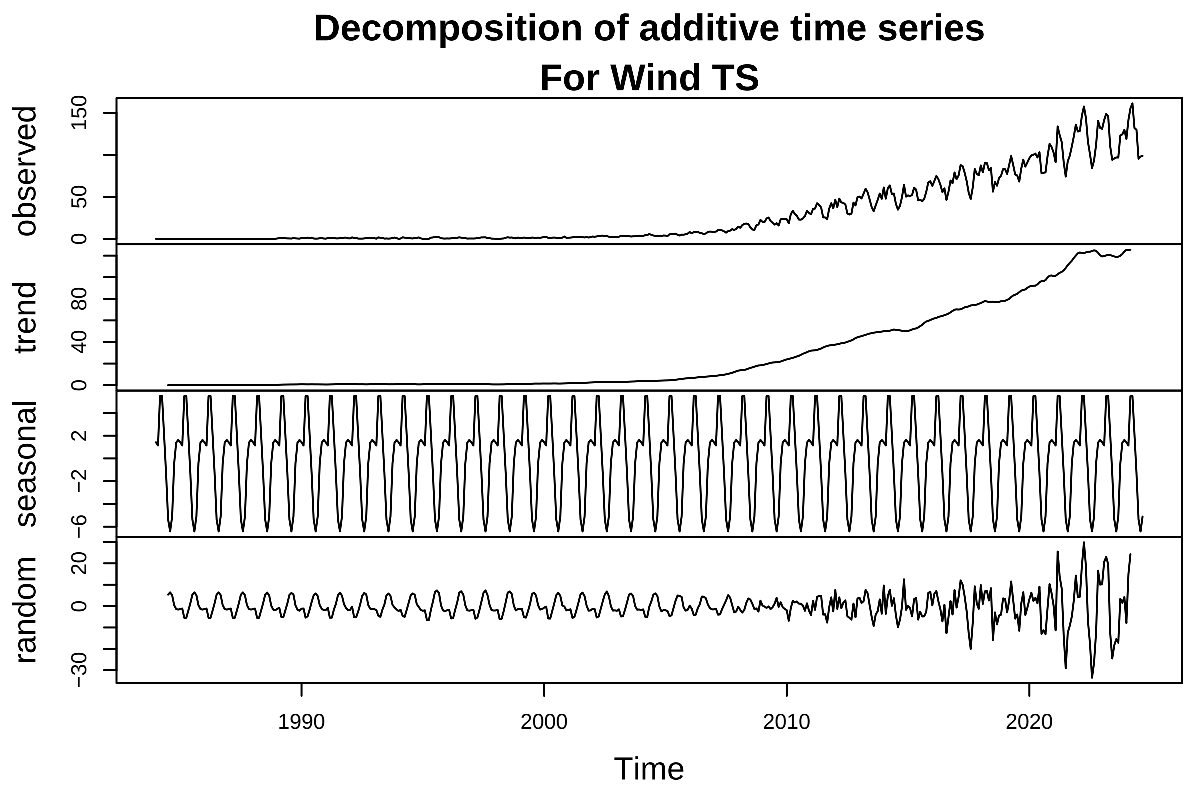
Transform wind and solar series into a time series object and apply the decompose function on them using the additive option, i.e., `decompose(ts_data, type = "additive")`. What can you say about the trend component? What about the random component? Does the random component look random? Or does it appear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

```r
# already transformed to a TS object in previous question
# decomposing both TS objects
solar_decomposed_add <- decompose(solar_ts, type = "additive")
plot(solar_decomposed_add) + title(main = "For Solar TS", line = 1)
```



```
## integer(0)
```

```r
wind_decomposed_add <- decompose(wind_ts, type = "additive")
plot(wind_decomposed_add) + title(main = "For Wind TS", line = 1)
```



```
## integer(0)
```

> Answer:For solar energy consumption time series, the decomposed trend seems like a smooth exponential curve, while the random has a clear oscillation pattern that has constant amplitude from 1984 to 2010, and the amplitude decreased and increased drastically towards 2024.

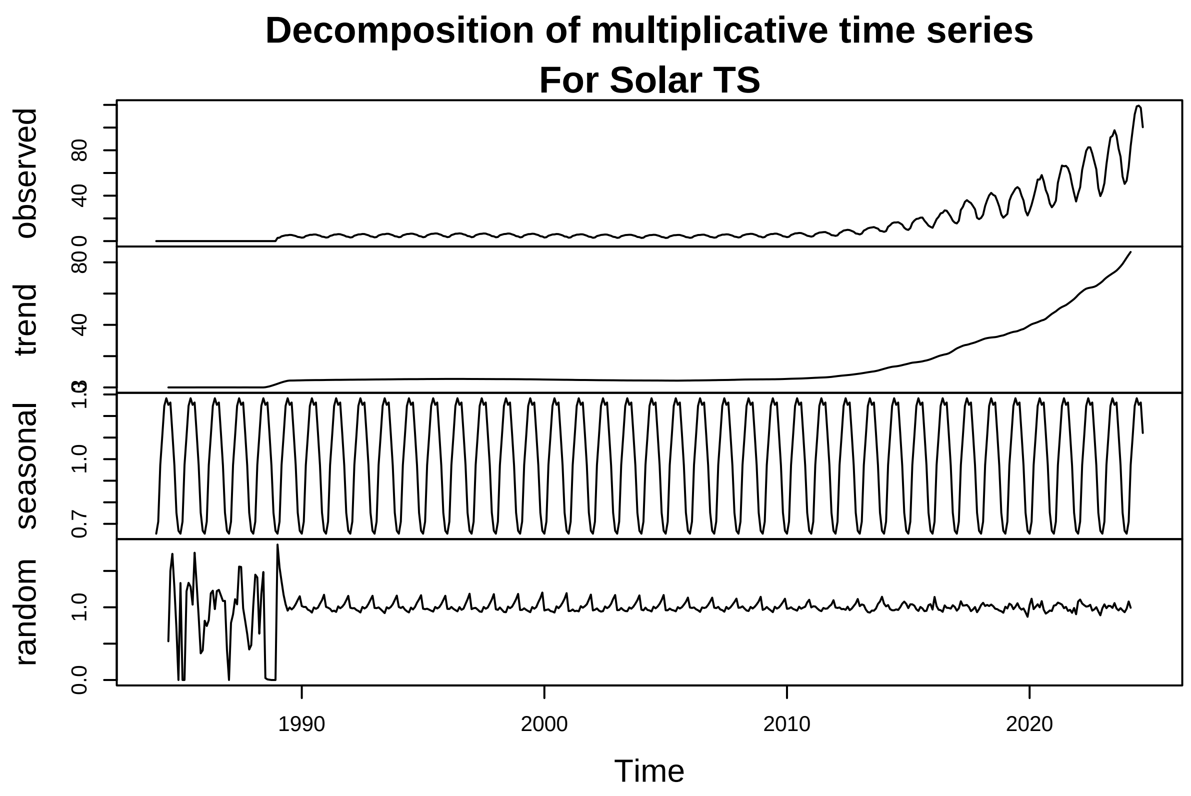
For wind energy consumption time series, the decomposed trend is less smooth but still shows an exponential shape, with the random having a up and down oscillation that increases the amplitude from 2011 to 2024, similar to the solar time series observed above.

These observations indicate that the random component for both time series still have seasonality and that the seasonality changes with the amplitude of the trend.

Q5

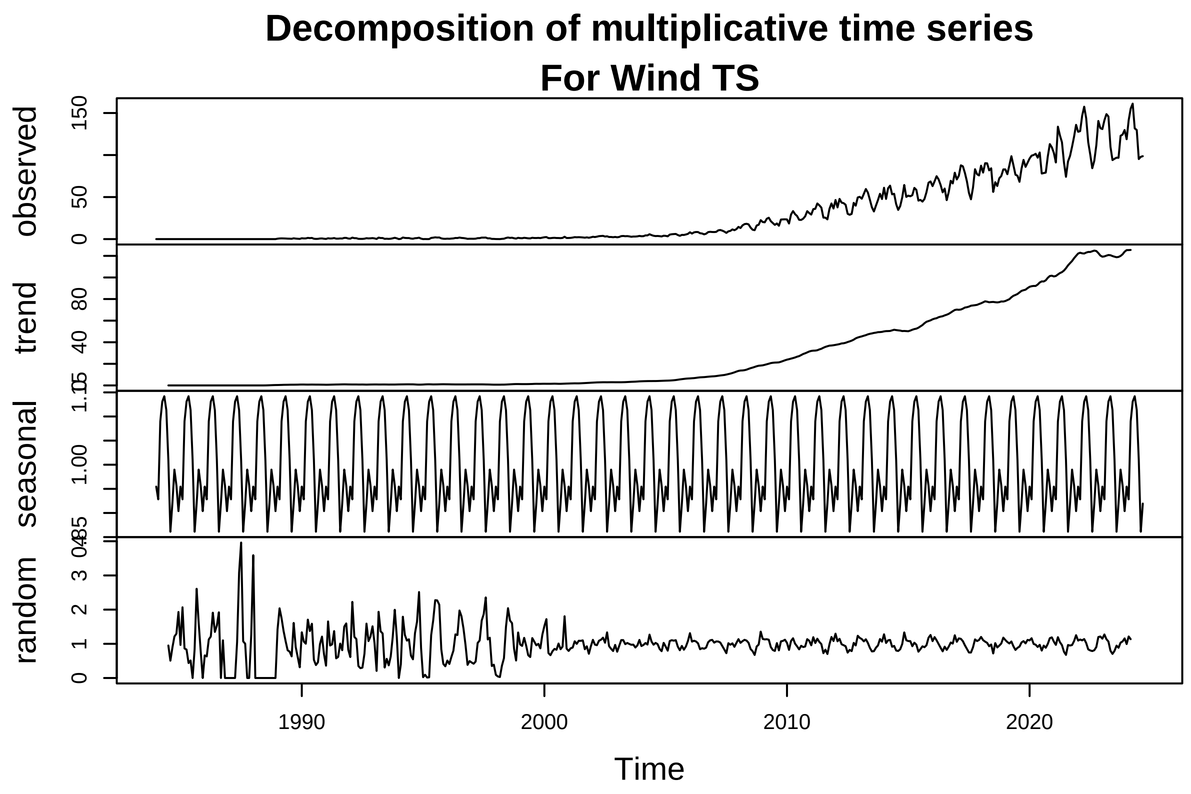
Use the decompose function again but now change the type of the seasonal component from additive to multiplicative. What happened to the random component this time?

```r
solar_decomposed_mul <- decompose(solar_ts, type = "multiplicative")
plot(solar_decomposed_mul) + title(main = "For Solar TS", line = 1)
```



```
## integer(0)
```

```r
wind_decomposed_mul <- decompose(wind_ts, type = "multiplicative")
plot(wind_decomposed_mul) + title(main = "For Wind TS", line = 1)
```



```
## integer(0)
```

> Answer:The random component for both time series looks more random (i.e. less seasonality pattern) with the multiplicative model. However, the solar time series' random component still shows subtle seasonality from 1990 to 2009. The wind time series' one appears to be more random.

These observations suggest that the multiplicative model works better than the additive model in removing the trend from the original time series and to decompose them further to seasonality and random parts.

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.

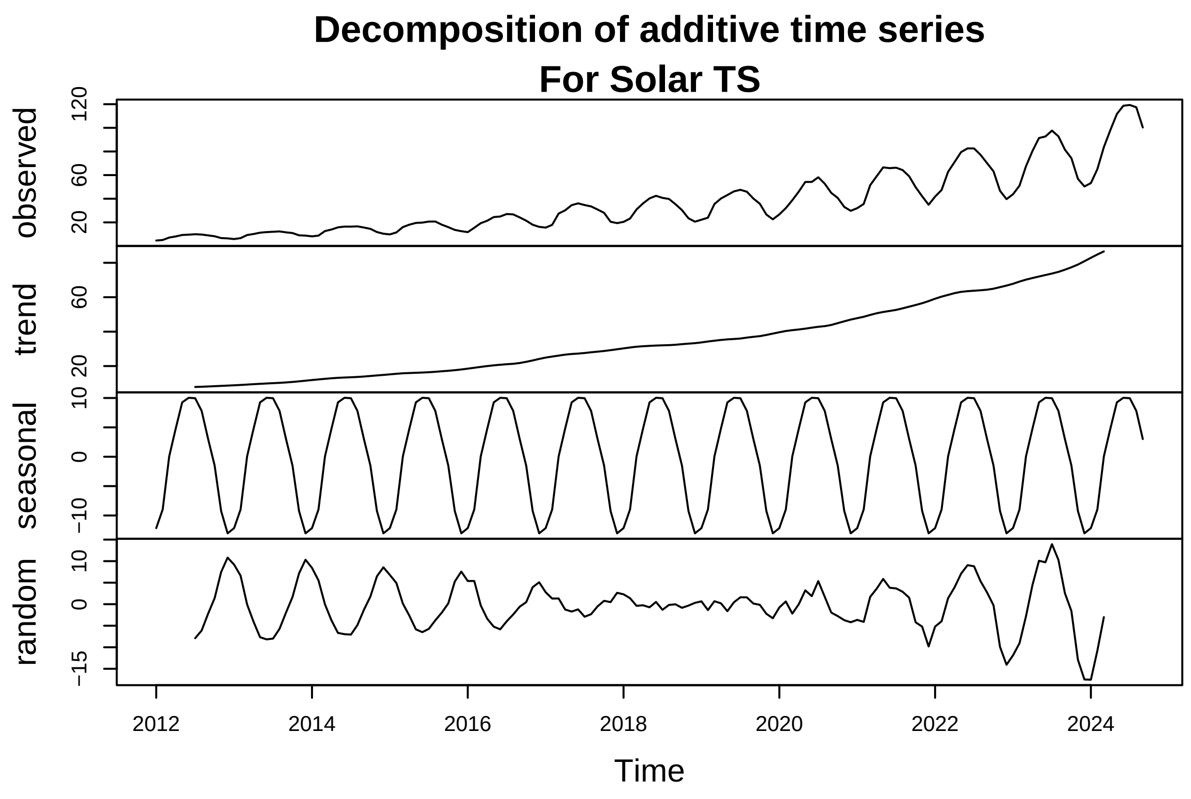
> Answer: The short answer is no, not all historical data is needed. This is because at the early part of the time series (80s and 90s) the magnitude is too small as compared to later time periods (2020s), and the overall trend will not change of the earlier data is truncated. Meanwhile, data from the early 20s is important to predict the next six months of solar and wind consumption, since they are temporally close to the future. Therefore, it is not necessary to have all historical data before fitting into a time series model. To preditct the energy consumption for the next six months, finding the recent trend with recent data (in early 20s) is more important.

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.

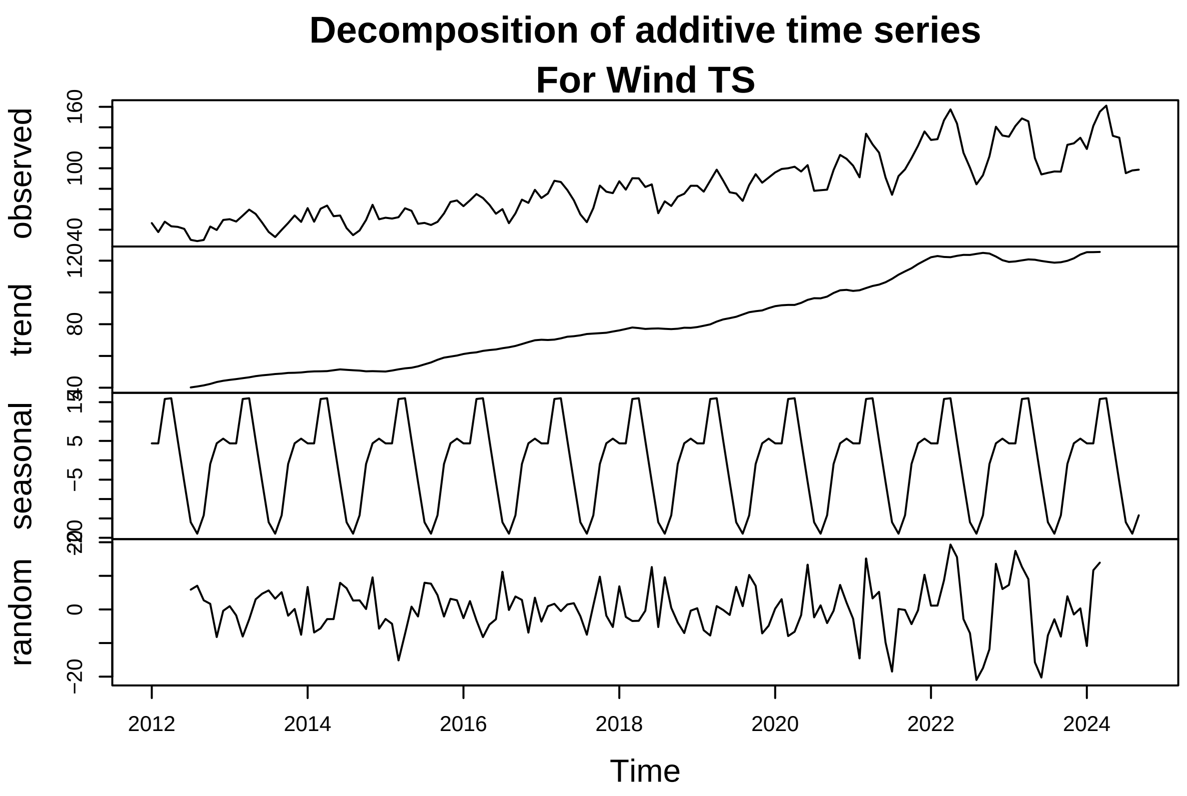
```r
# filtering date
renewables_present <- filter(renewables_data, year(Date) >= 2012)
solar_present_ts <- ts(
  renewables_present$Solar, start = c(2012,1), frequency = 12)
wind_present_ts <- ts(
  renewables_present$Wind, start = c(2012,1), frequency = 12)

# decomposing time series
solar_present_decomposed <- decompose(solar_present_ts, type = "additive")
plot(solar_present_decomposed) + title(main = "For Solar TS", line = 1)
```



```
## integer(0)
```

```r
wind_present_decomposed <- decompose(wind_present_ts, type = "additive")
plot(wind_present_decomposed) + title(main = "For Wind TS", line = 1)
```



```
## integer(0)
```

> Answer: For solar series, the random component does not look random, especially from 2012 to 2016, and from 2021 to 2024. The trend and seasonal components remain similar to the full dataset. For wind series, the random component looks random but still demonstrates a little bit of seasonality, while its trend looking like a linear one instead of an exponential one. The reason why the random components for both series still shows seasonality is because the time series is not a linear model, it is a product of the trend and the seasonality. Therefore, the seasonality changes with levels of the trend. In the decomposition process, additive model was selected, thus the unpresented change in seasonality is left in the random component, leading to their seasonality patterns.

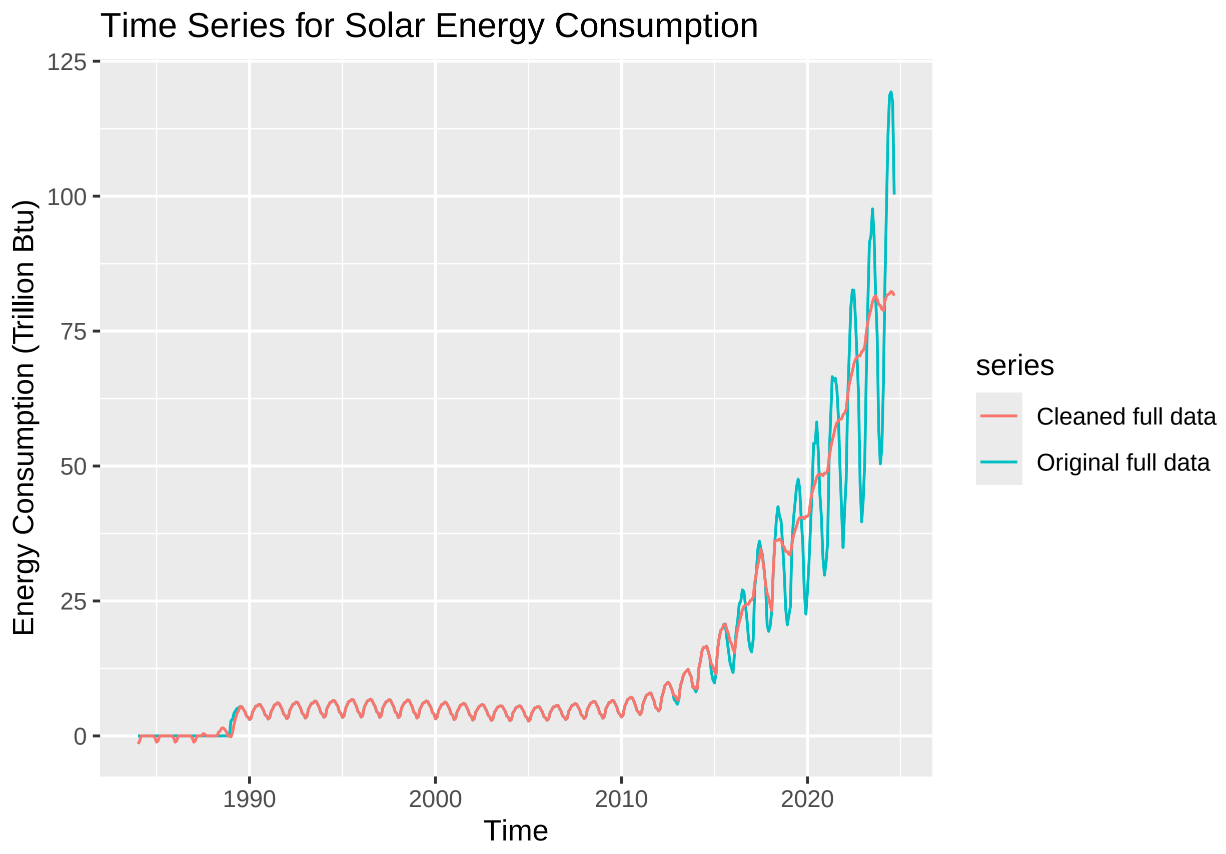
Identify and Remove outliers

Q8

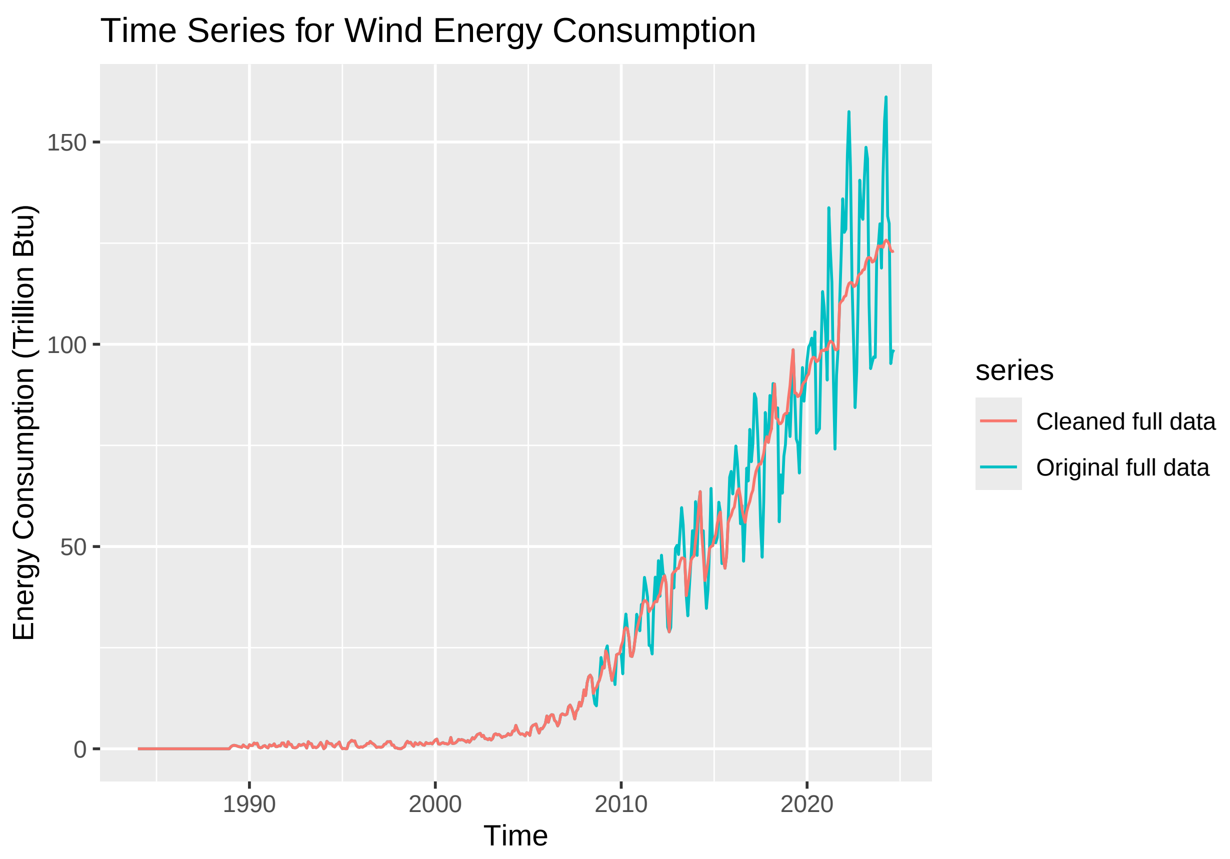
Apply the tsclean() to both time series object you created on Q4. Did the function removed any outliers from the series? Hint: Use autoplot() to check if there is difference between cleaned series and original series.

```r
solar_cleaned <- tsclean(solar_ts)
wind_cleaned <- tsclean(wind_ts)

# plot for solar
autoplot(solar_ts, series = "Original full data") +
  autolayer(solar_cleaned, series = "Cleaned full data")+
  ylab("Energy Consumption (Trillion Btu)") +
  ggtitle("Time Series for Solar Energy Consumption")
```



```
# plot for wind
autoplot(wind_ts, series = "Original full data") +
  autolayer(wind_cleaned, series = "Cleaned full data")+
  ylab("Energy Consumption (Trillion Btu)") +
  ggtitle("Time Series for Wind Energy Consumption")
```



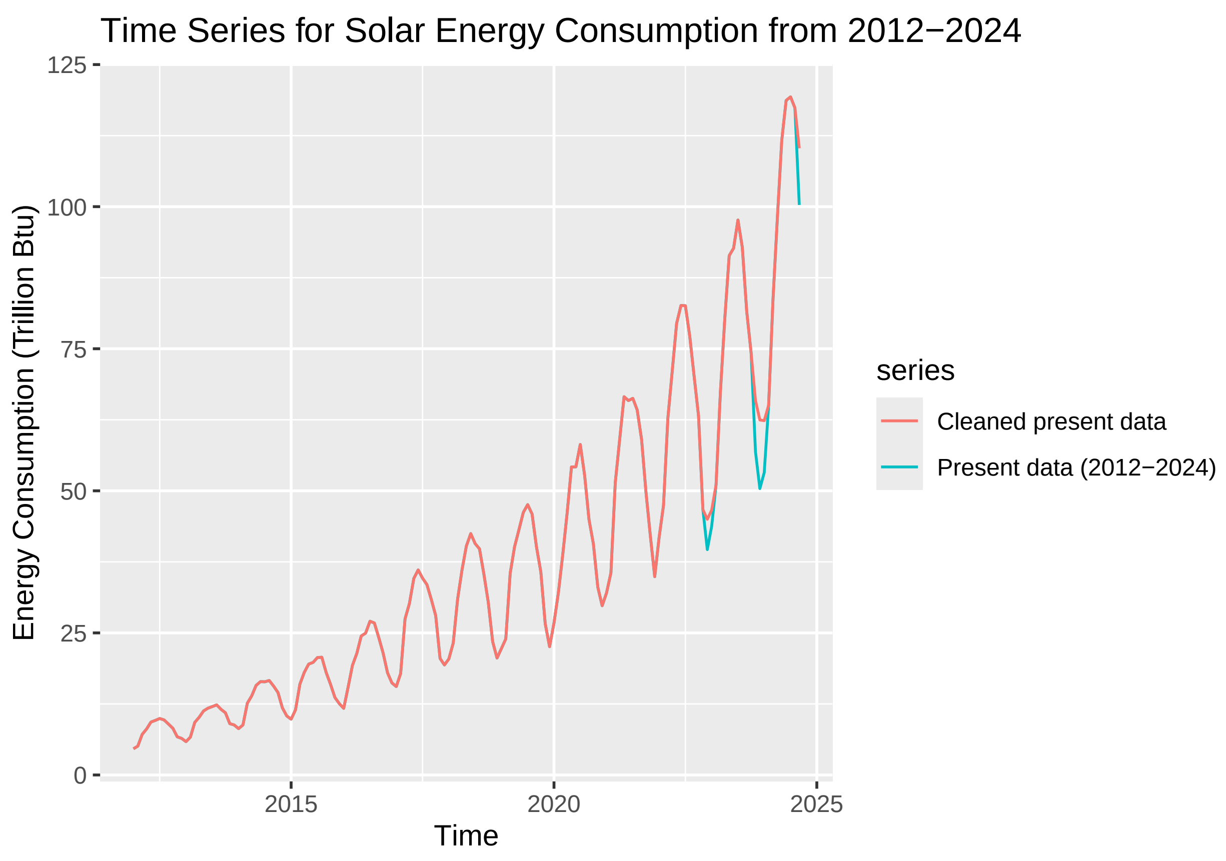
Answer: It seems like the tsclean function treated the large oscillations after the year 2015 as outliers and removed them for a smoother series.

Q9

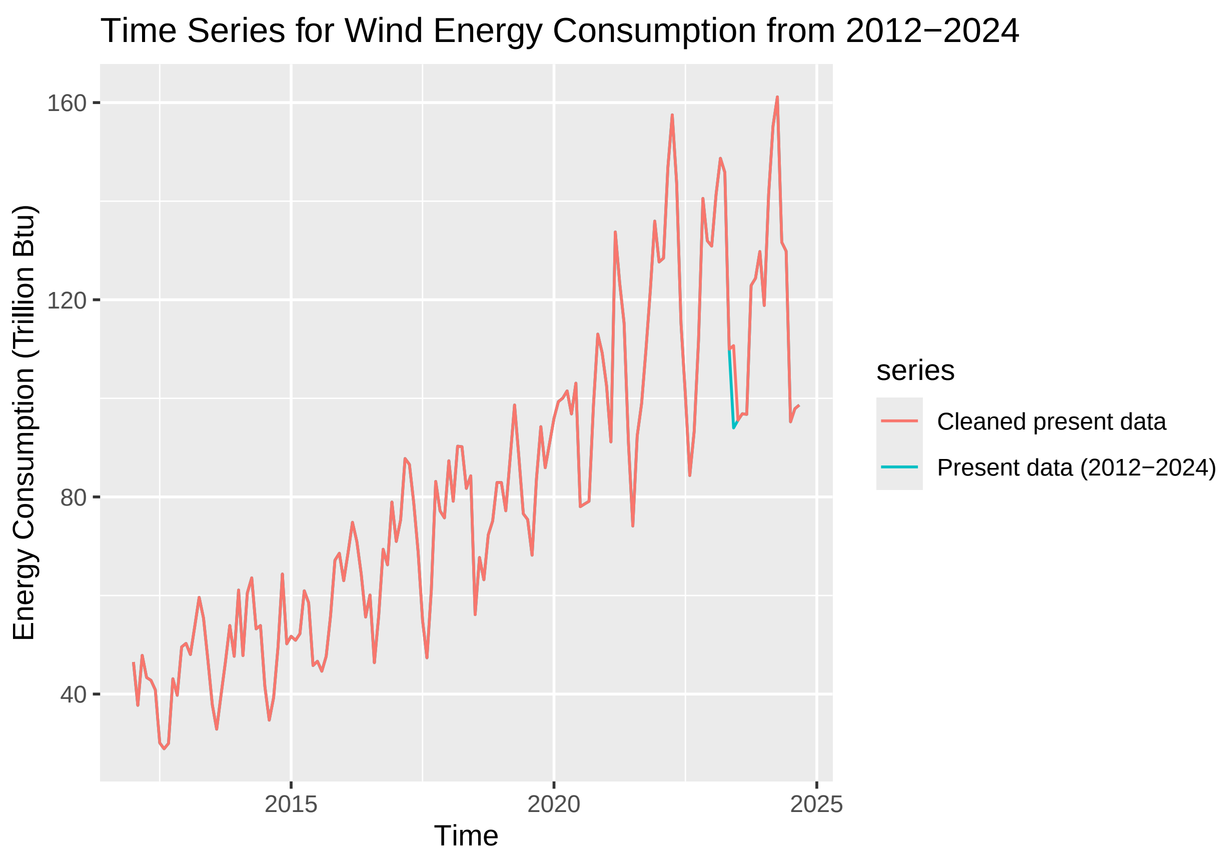
Redo number Q8 but now with the time series you created on Q7, i.e., the series starting in 2012. Using what autoplot() again what happened now? Did the function removed any outliers from the series?

```r
solar_present_cleaned <- tsclean(solar_present_ts)
wind_present_cleaned <- tsclean(wind_present_ts)

# plot for solar
autoplot(solar_present_ts, series = "Present data (2012-2024)") +
  autolayer(solar_present_cleaned, series = "Cleaned present data")+
  ylab("Energy Consumption (Trillion Btu)") +
  ggtitle("Time Series for Solar Energy Consumption from 2012-2024")
```



```r
# plot for wind
autoplot(wind_present_ts, series = "Present data (2012-2024)") +
  autolayer(wind_present_cleaned, series = "Cleaned present data")+
  ylab("Energy Consumption (Trillion Btu)") +
  ggtitle("Time Series for Wind Energy Consumption from 2012-2024")
```



Answer:The tsclean function only removed or changed a very limited amount of data. This indicates that by using recent data it is easier to model recent trends and can achieve a better result.