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

Assignment 5 - Due date 02/13/24

Faustin Kambale

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.\

```
library(lubridate)
library(ggplot2)
library(forecast)
library(Kendall)
library(tseries)
library(cowplot)
library(dplyr)
library(tidyr)
```

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.

```
raw_data <- read.csv(file
="/Users/faustinkambale/Library/CloudStorage/OneDrive-DukeUniversity/Spring 2024 classes/Time Series 4 
                    header=TRUE,skip=0)
head(raw_data,10)
```

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!

```
#data <- as.data.frame(raw_data[,8:9])
#colnames(data) <- c("solen","winden")

# Replace "Not Available" with NA for consistency
#data[data == "Not Available"] <- NA
#data_clean <- data %>% drop_na()
#data_clean$solen <- as.numeric(data_clean$solen)
#data_clean$winden <- as.numeric(data_clean$winden)
#head(data_clean)
```

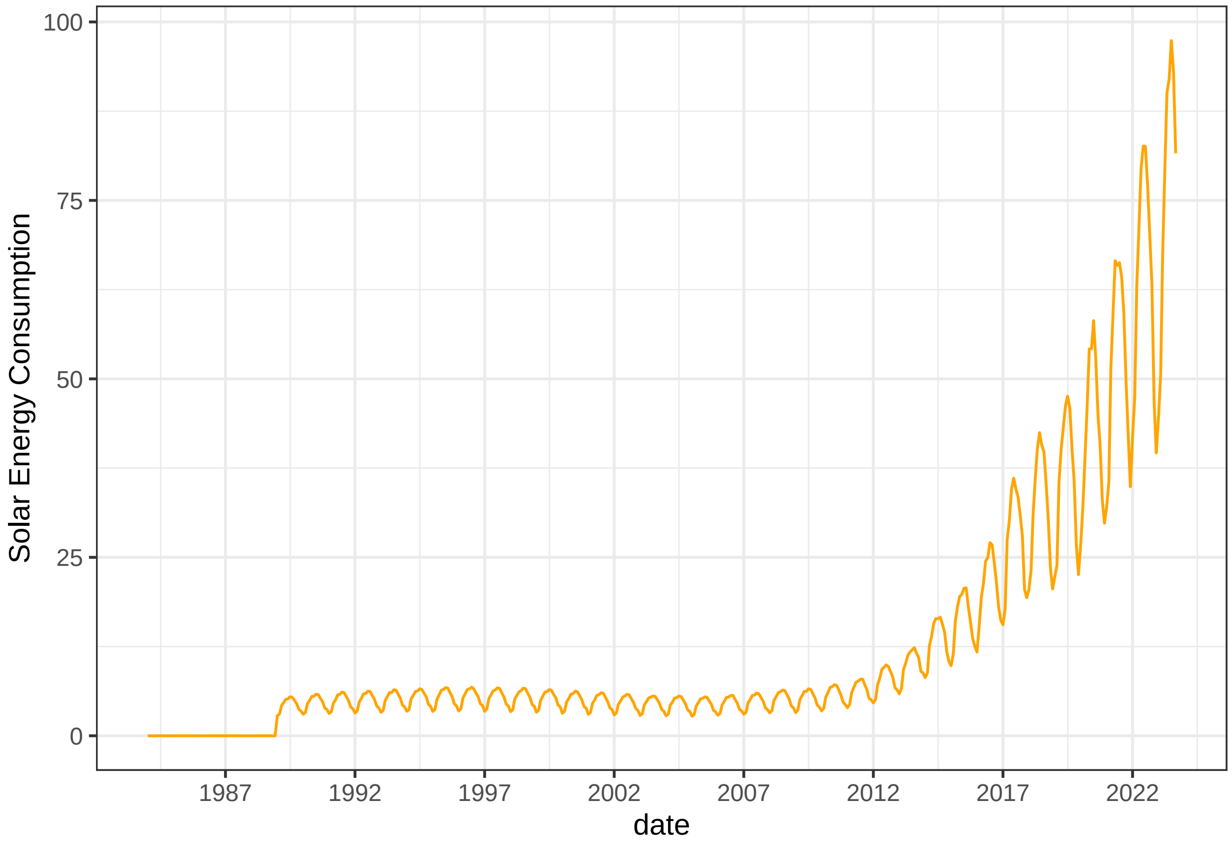
```
#Set variables to work with
data <- as.data.frame(raw_data) %>% select("Month", "Solar.Energy.Consumption","Wind.Energy.Consumption

# Replace "Not Available" with NA for consistency
data[data == "Not Available"] <- NA
data_clean <- data %>% drop_na()
colnames(data_clean) <- c("date", "solen","winden")

#convert dataframe to numeric
data_clean$solen <- as.numeric(data_clean$solen)
data_clean$winden <- as.numeric(data_clean$winden)
data_clean$date <- ym(data_clean$date)
nobs <- nrow(data_clean)
head(data_clean,10)
```

```
##          date solen winden
## 1  1984-01-01 0.000  0.000
## 2  1984-02-01 0.000  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
## 7  1984-07-01 0.001  0.002
## 8  1984-08-01 0.003  0.001
## 9  1984-09-01 0.003  0.002
## 10 1984-10-01 0.002  0.003
```

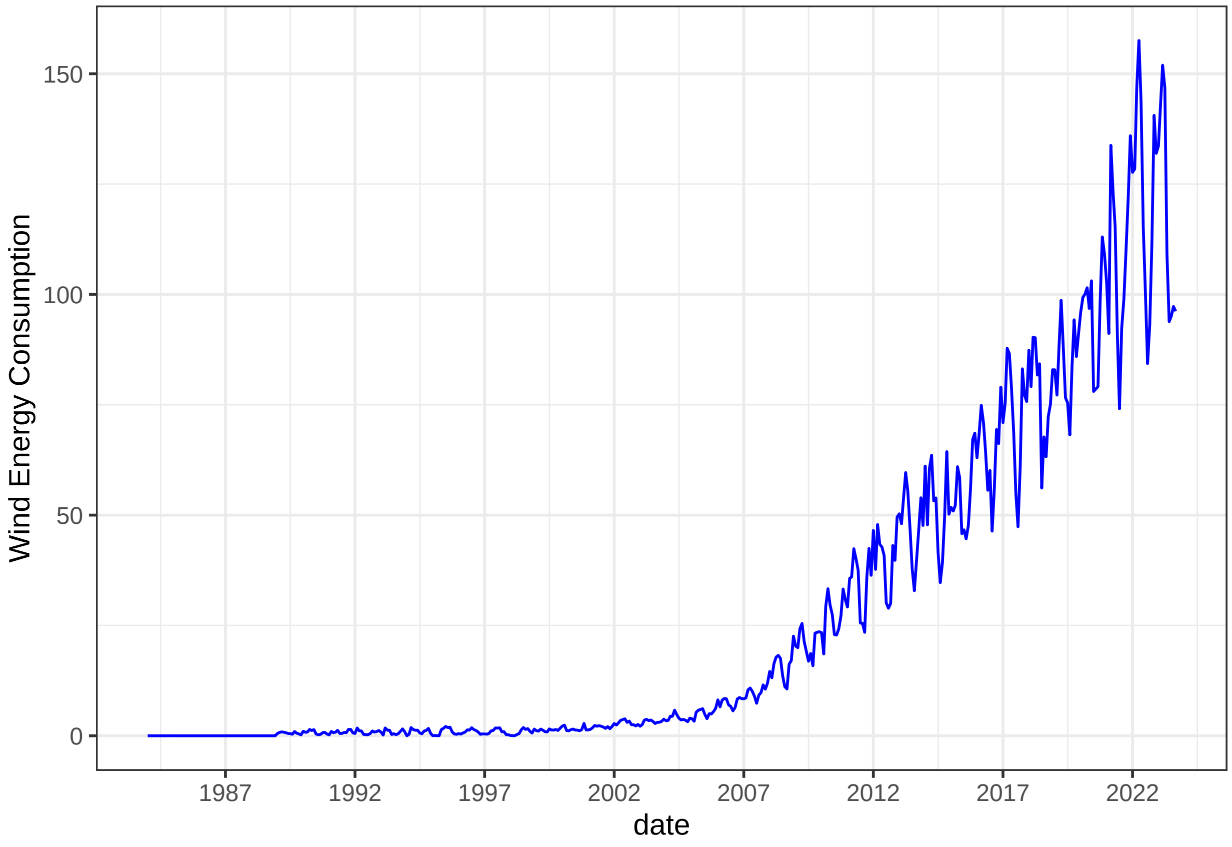
Q2

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")")`

```
# Plot solen series
solenplot <- ggplot(data = data_clean, aes(x = date, y = solen)) +
  geom_line(color = "orange") +
  ylab("Solar Energy Consumption")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  theme_bw()
print(solenplot)
```



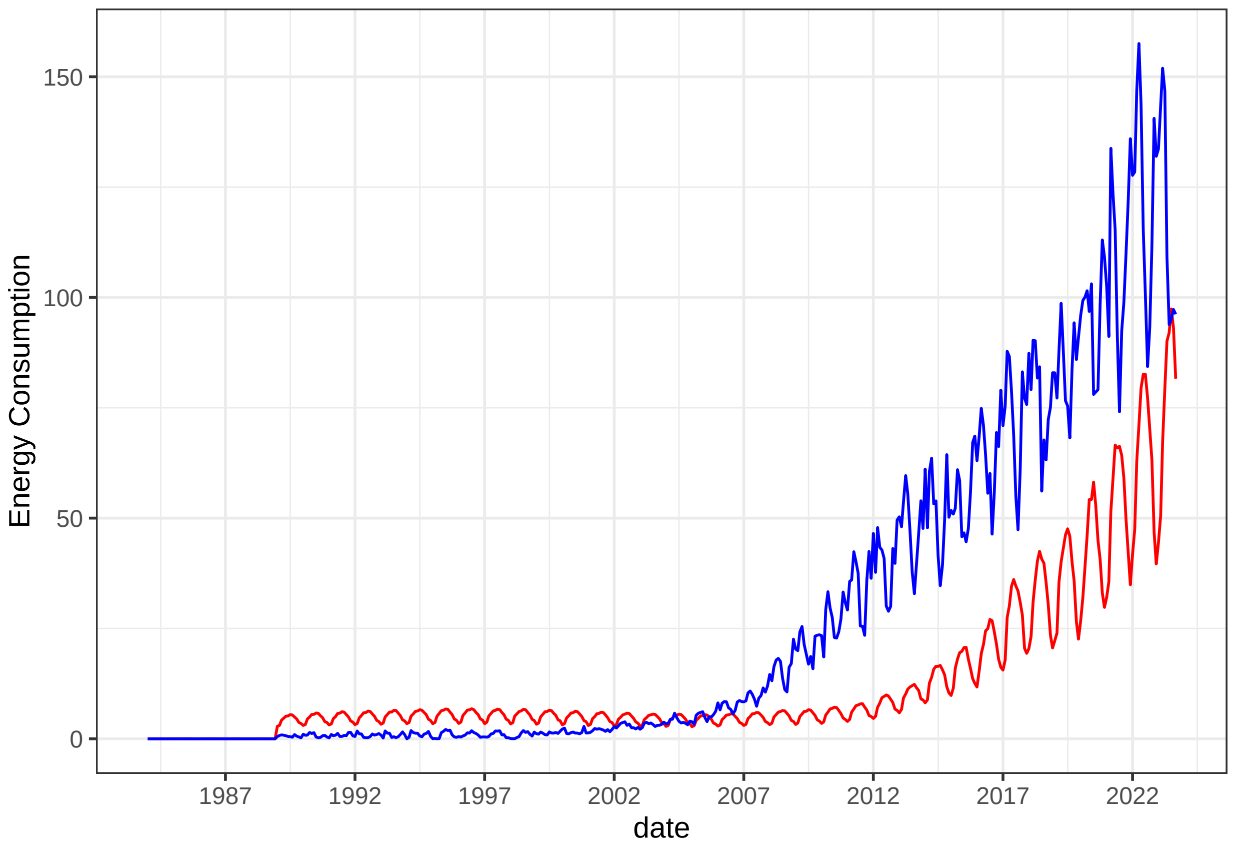
```r
# Plot Wind Energy
windenplot <- ggplot(data = data_clean, aes(x = date, y = winden)) +
  geom_line(color = "blue") +
  ylab("Wind Energy Consumption")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  theme_bw()
  print(windenplot)
```



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.

```
# Creating the plot
print(
  ggplot(data = data_clean, aes(x = date)) +
  geom_line(aes(y = solen), color = "red", linetype = "solid") +
  geom_line(aes(y = winden), color = "blue", linetype = "solid") +
  ylab("Energy Consumption")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  theme_bw()
)
```



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.

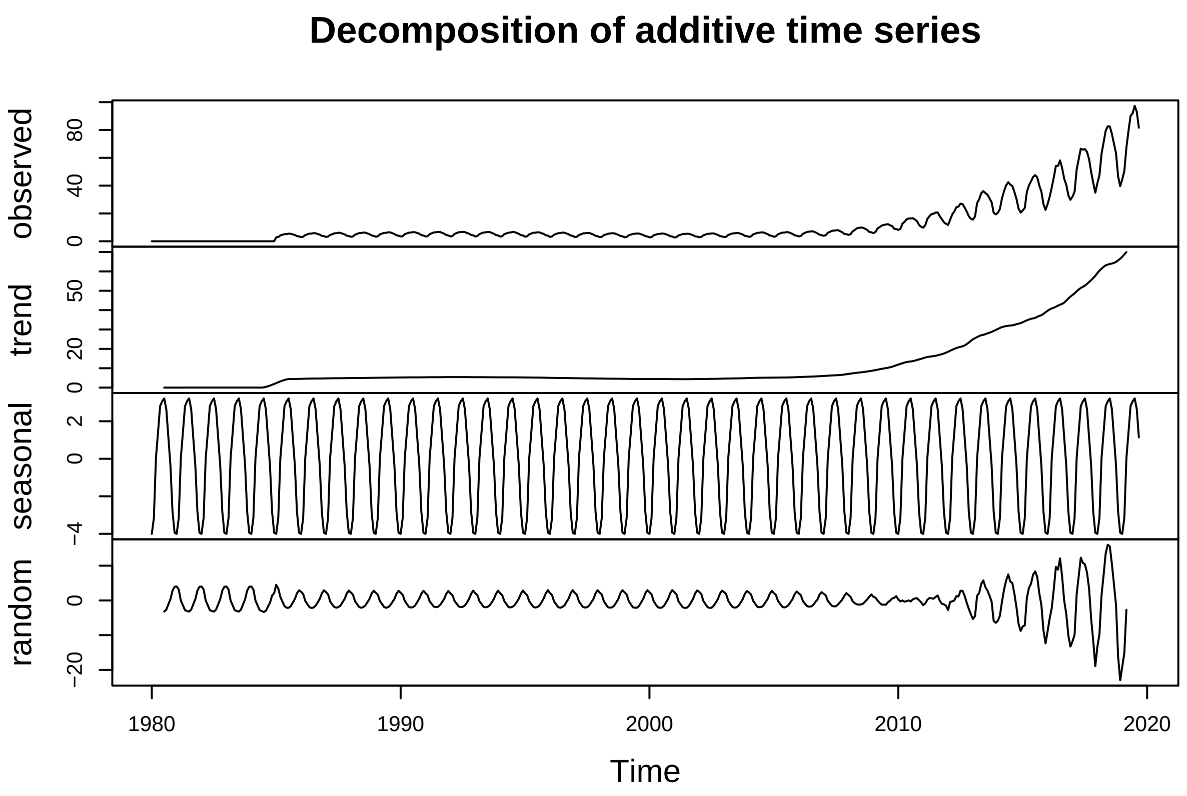
Additional info on `decompose()`.

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.

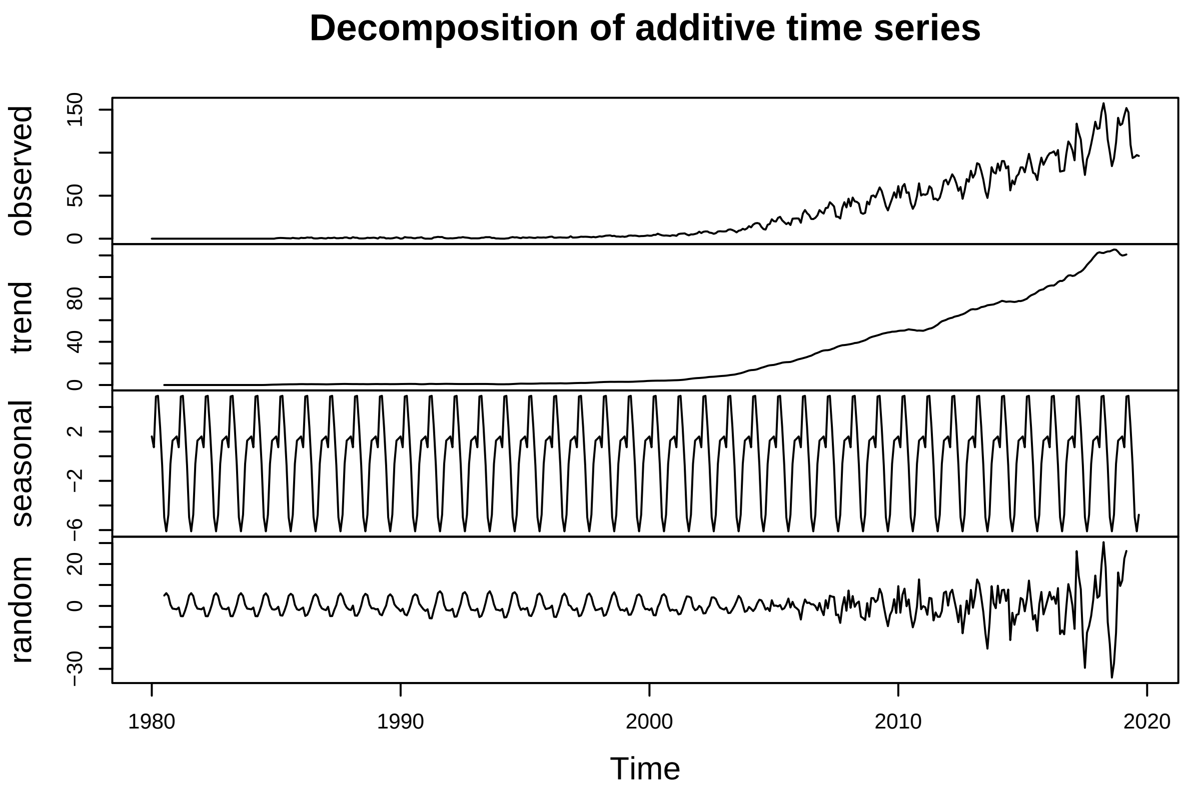
Q4

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?

```
#convert in time series
tsdata_clean <- ts(data_clean[,2:3], start=c(1980,1),frequency=12)
#Decomposing with additive function
sol_decom <- decompose(tsdata_clean[,1], type = "additive")
plot(sol_decom) #Red flag : seasonality and increase in magnitude in random plot. meaning the additive
```



```r
win_decom <- decompose(tsdata_clean[,2], type = "additive")
plot(win_decom) #no stable seasonality in random, more variability and higher magnitude.
```

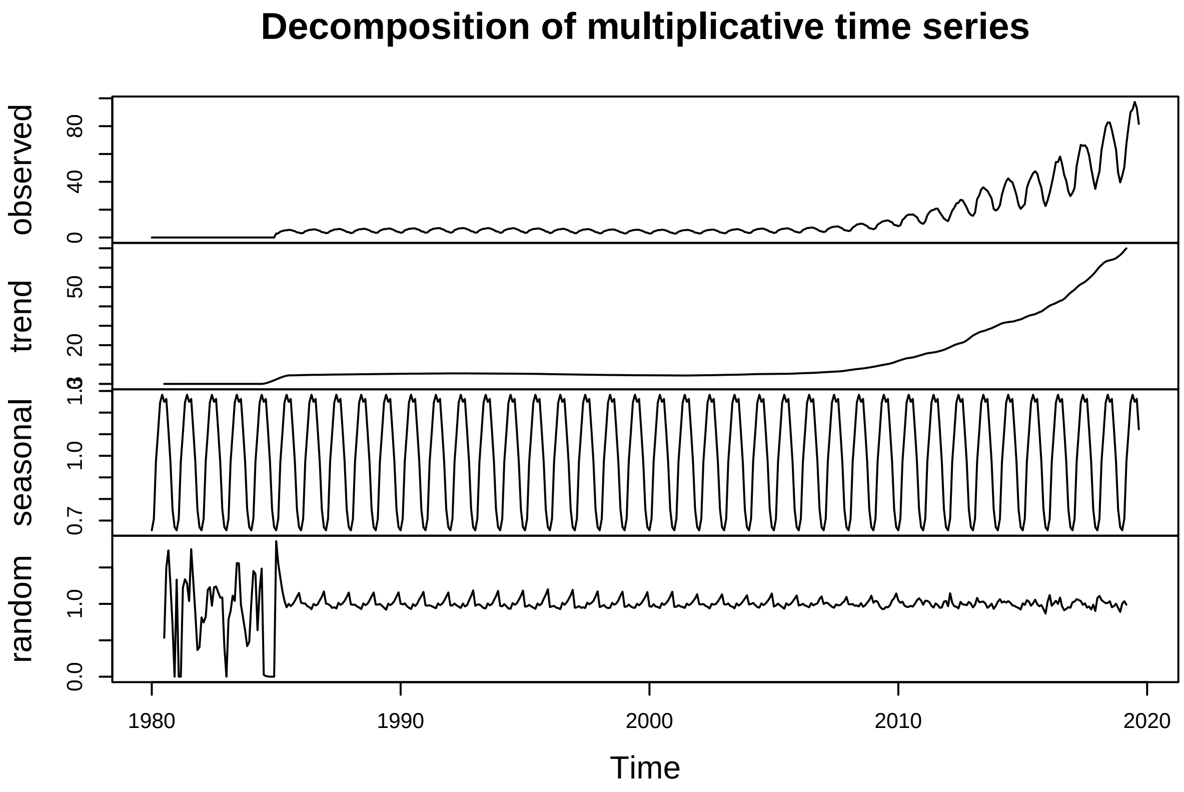


From this graph, we can observe *a straight upward trend line* over time for the Solar Energy Consumption series and *an upward increasing trend line* for the Wind Energy Consumption series. So, both series display a linear increasing trend over time. From the random part of the plot, the Solar Energy series displays some noise in the data. We can notice a seasonality trend in these series (solar and Wind). However, the Wind series displays less nois than the Solar.

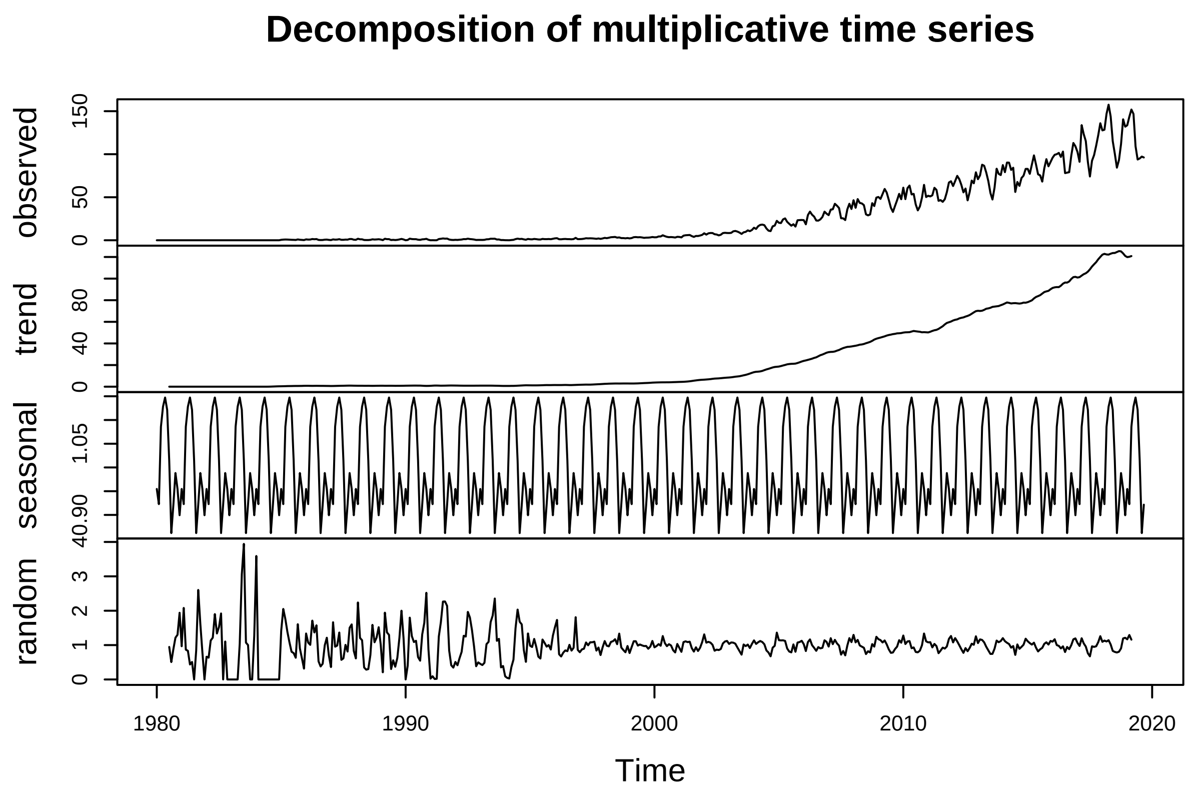
Q5

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 with multiplicative function
sol_decom1 <- decompose(tsdata_clean[,1], type = "multiplicative")
plot(sol_decom1) #no magniture but still seasonality going on. they are not centered to zero here also
```



```
win_decom1 <- decompose(tsdata_clean[,2], type = "multiplicative")
plot(win_decom1)# still suspect seasonality. they are not centered to zero here also.
```



In this specific example, the additive function displays a similar trend as the multiplicative function except that, for the Wind series, the noise (random) seems to be stable than Solar's.

Q6

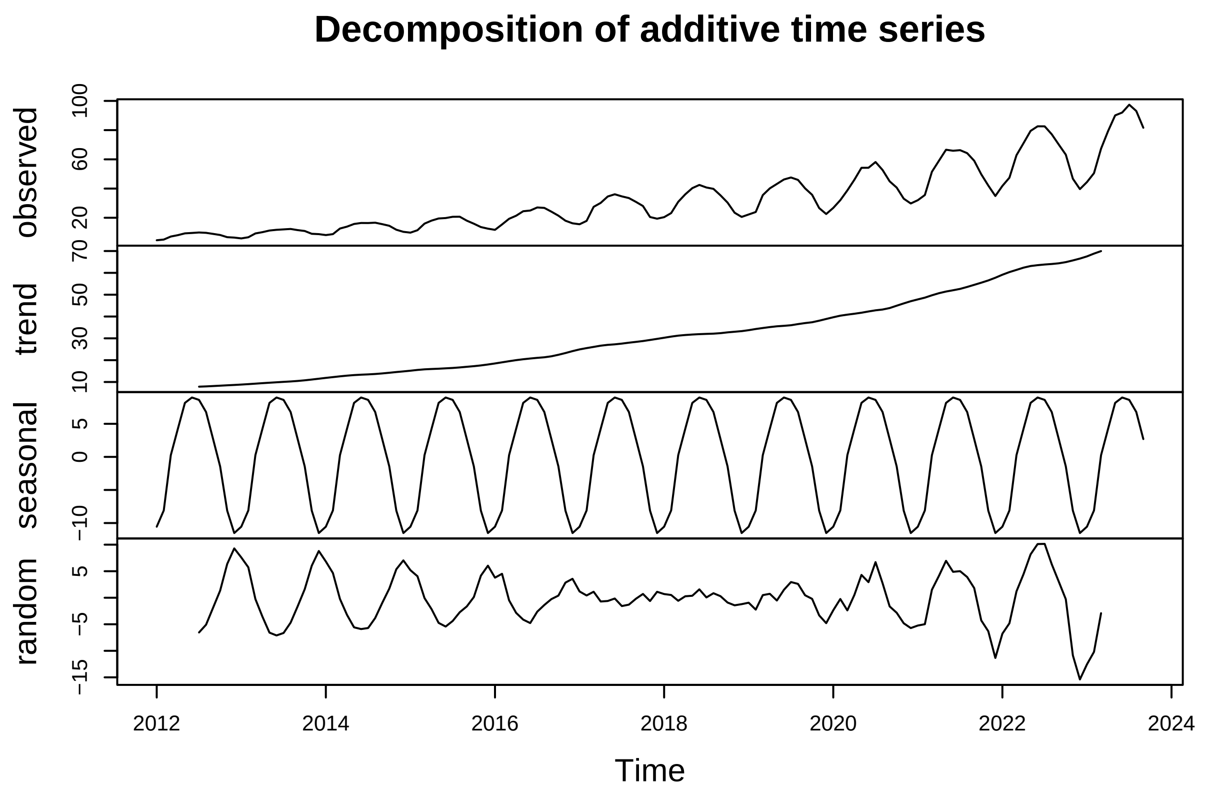
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: From the trend observation, we can see that the trend is visible moslty from the 2000s. Before, there is a flat pattern in the series.

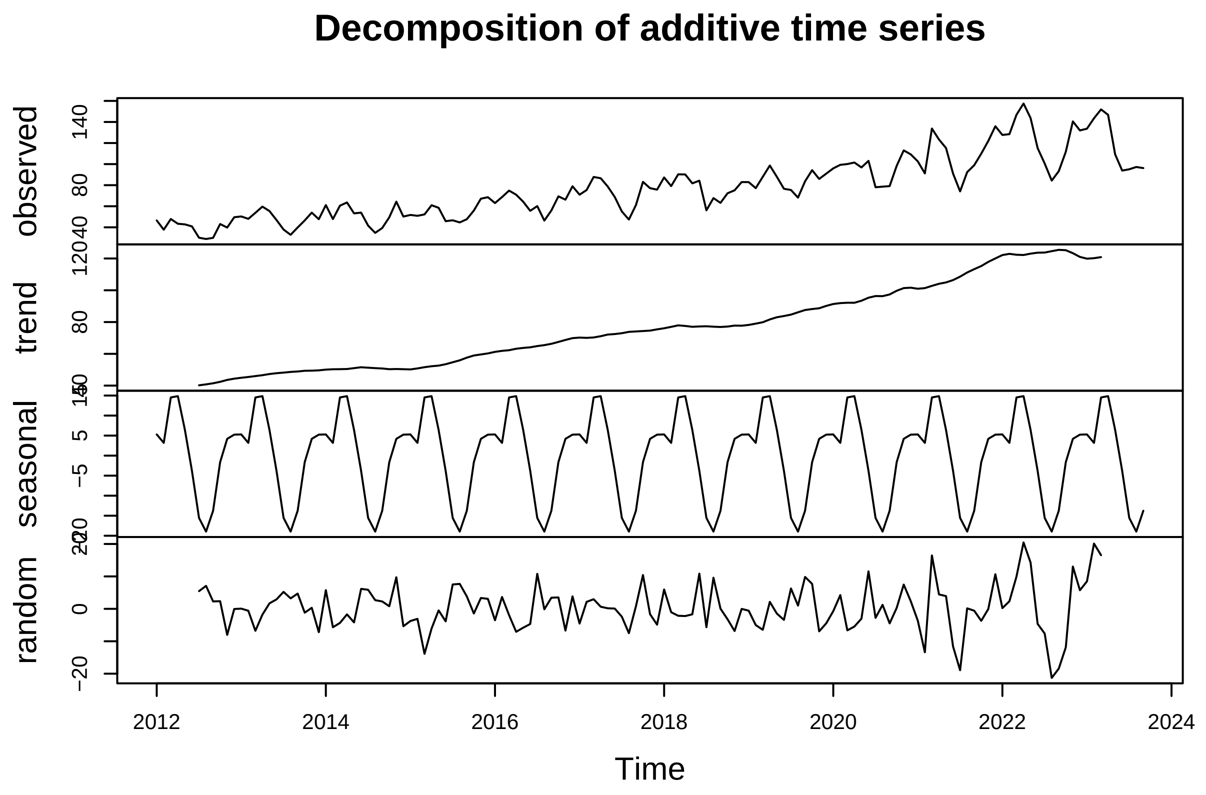
Q7

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
#New time series
filtered_data <- filter(data_clean, year(date) >= 2012)
filtered_data <- ts(filtered_data[,2:3], start= c(2012,1), frequency=12)

#Decomposing by additive function with new ts
sol_decomp <- decompose(filtered_data[,1], type = "additive")
plot(sol_decomp) #here components are now centered arround zero this time.
```



```
win_decomp <- decompose(filtered_data[,2], type = "additive")
plot(win_decomp) #random component but not yet seasonal related.
```



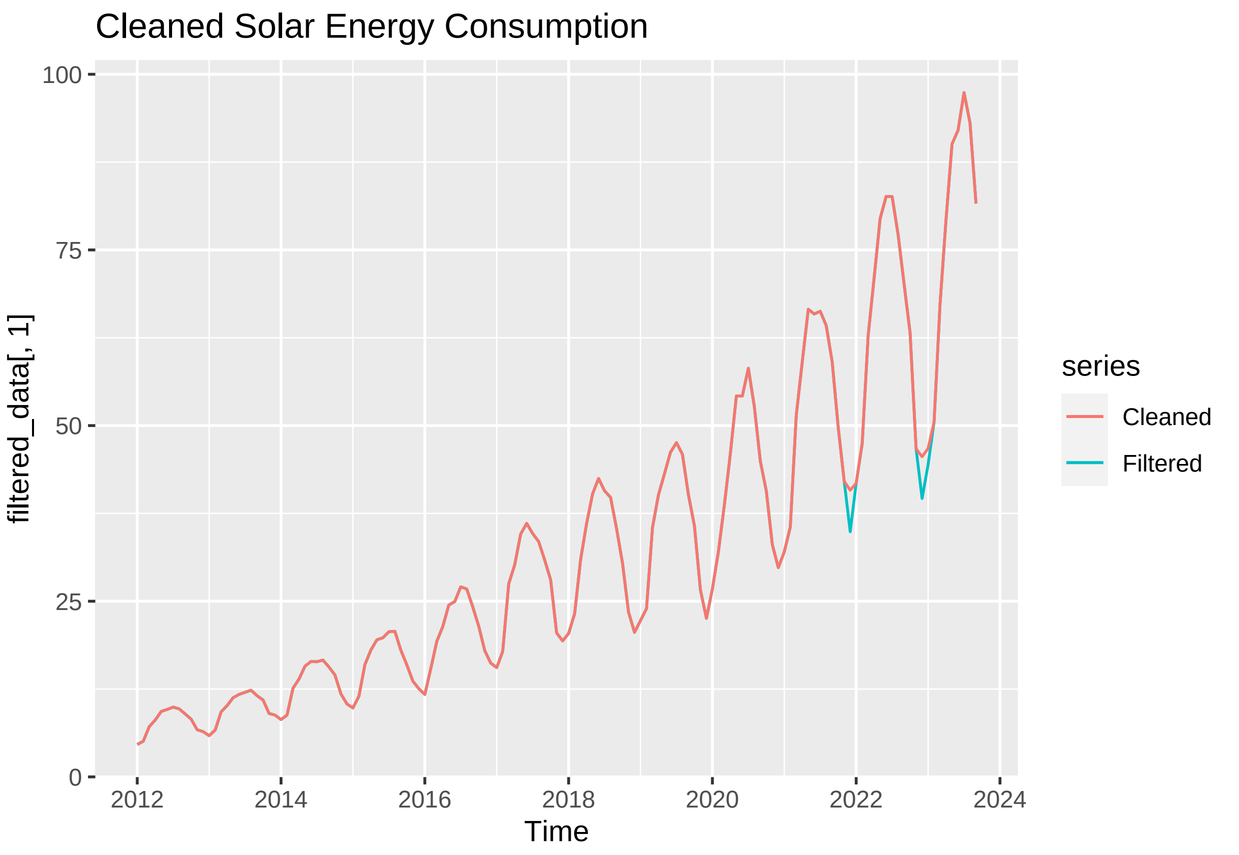
Answer: Compared to the previous plot, the filtered plot displays clearly the random component of the series. This plot shows a random pattern for the Wind series with a flat trend in the middle (between 2017 and 2021). Additionnaly, the Solar are now centered arround 0.

Identify and Remove outliers

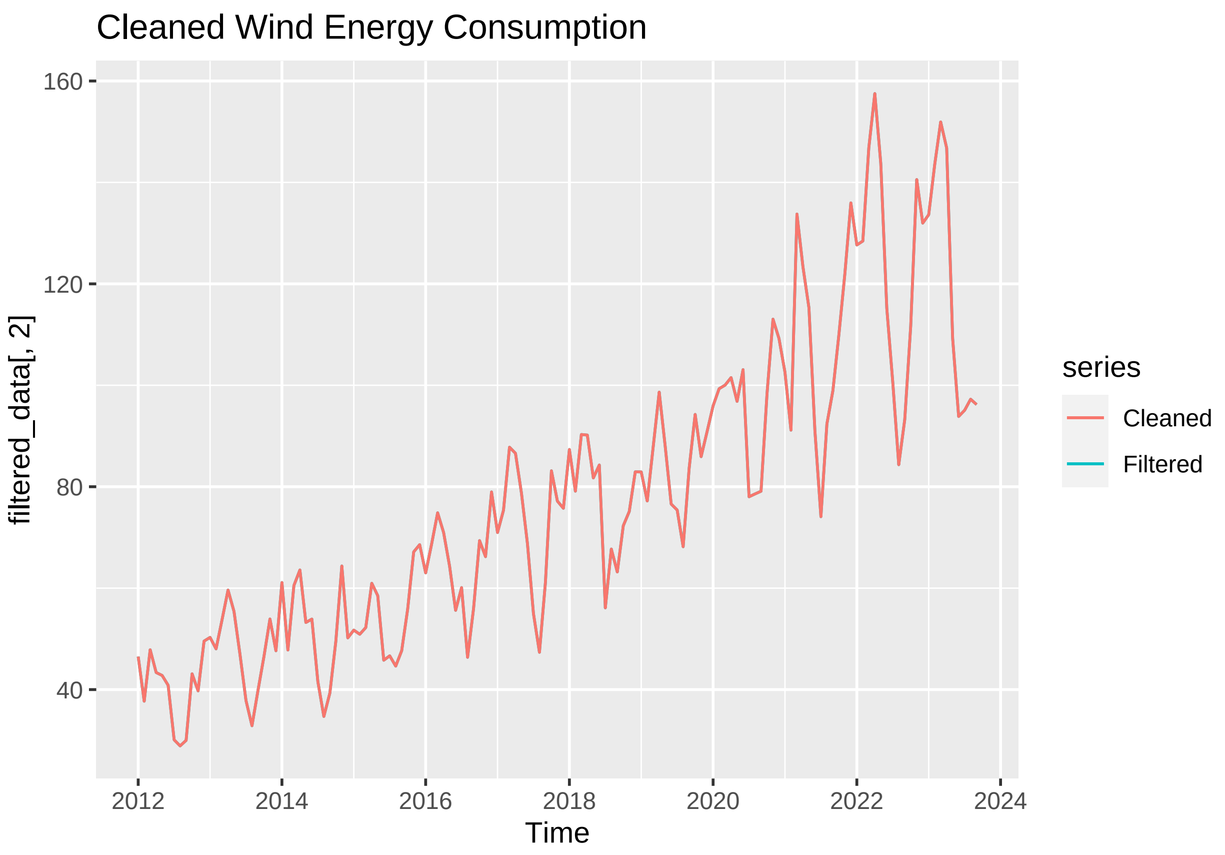
Q8

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.

```
#cleaning Salar Energy data
tscleansol <- ts(sol_decomp, start = c(2012,1), frequency = 12)
clean_solen <- tsclean(filtered_data[,1])
autoplot(filtered_data[, 1], series="Filtered", main = "Cleaned Solar Energy Consumption")+
  autolayer(clean_solen, series="Cleaned")
```



```
#cleaning Wind Energy Data
clean_winden <- tsclean(filtered_data[,2])
autoplot(filtered_data[,2], series="Filtered", main = "Cleaned Wind Energy Consumption")+
  autolayer(clean_winden, series="Cleaned")
```

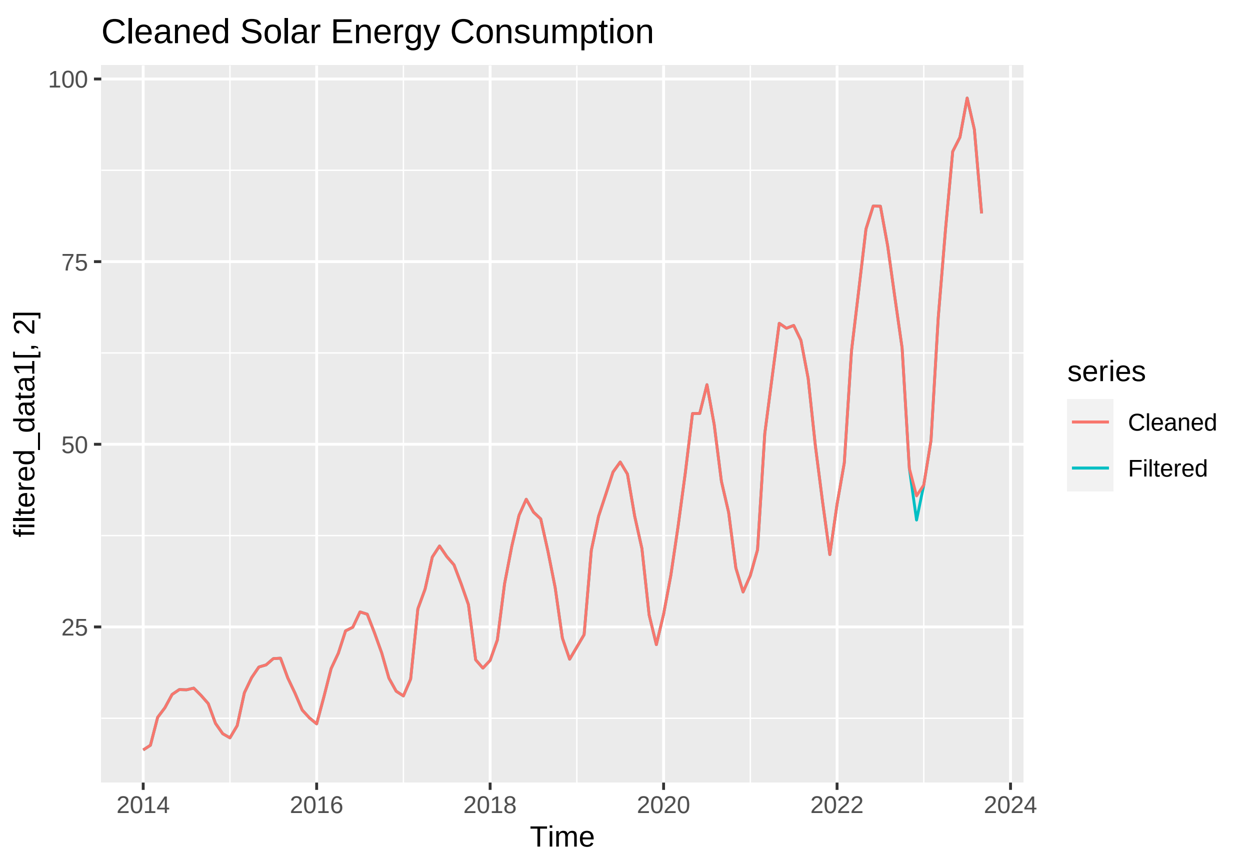


The function was effective on the Solar energy series by detecting and removing some outliers but did not detect any outlier from the Wind energy series.

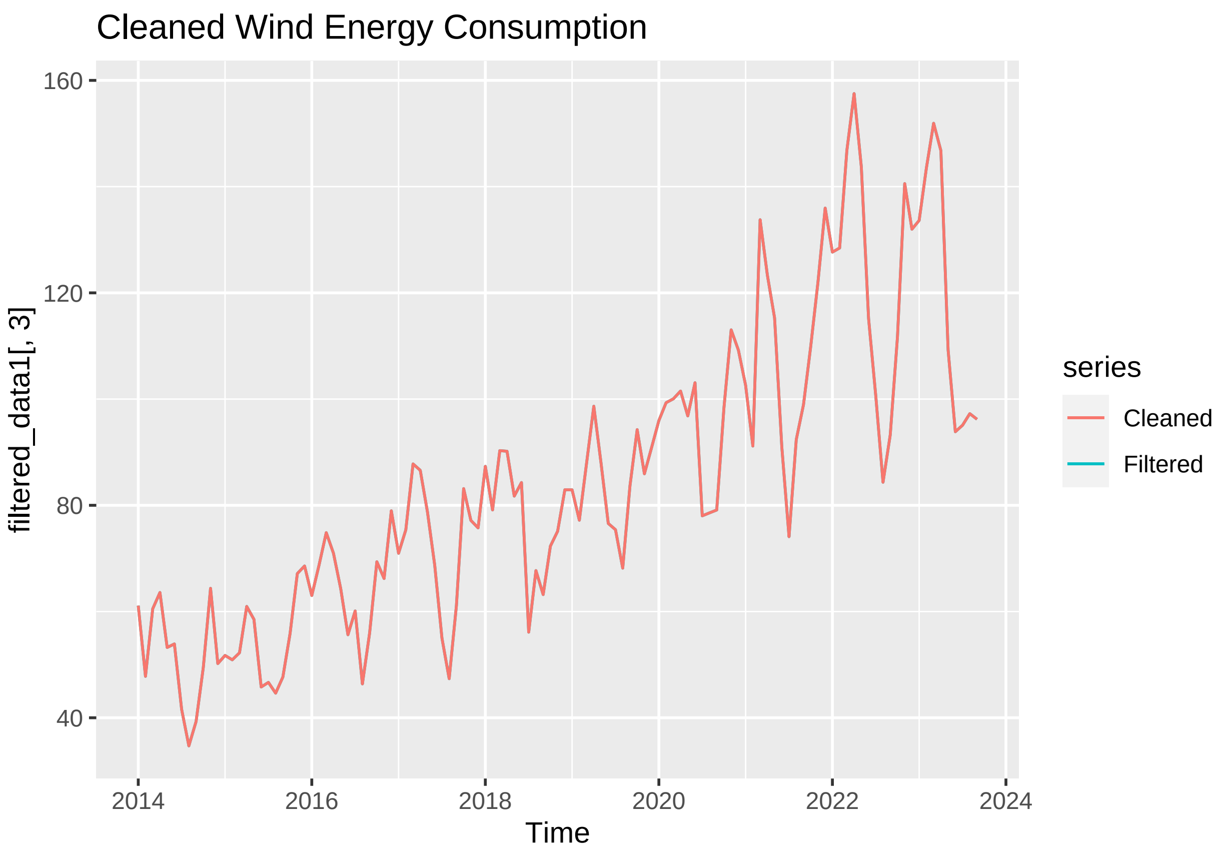
Q9

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

```
#New time series
filtered_data1 <- filter(data_clean, year(date) >= 2014)
filtered_data1 <- ts(filtered_data1, start= c(2014,1), frequency=12)
#cleaning Salar Energy data
clean_solen <- tsclean(filtered_data1[,2])
autoplot(filtered_data1[, 2], series="Filtered", main = "Cleaned Solar Energy Consumption")+
  autolayer(clean_solen, series="Cleaned")
```



```
#cleaning Wind Energy Data
clean_winden <- tsclean(filtered_data1[,3])
autoplot(filtered_data1[,3], series="Filtered", main = "Cleaned Wind Energy Consumption")+
  autolayer(clean_winden, series="Cleaned")#data don't find any outlier to remove.
```



Answer: same observation as on Q8.