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

Assignment 5 - Due date 02/18/25

Rosie Wu

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.\

```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
```

```
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
```

```
library(here)
```

```
## here() starts at /home/guest/TSA_Sp25
```

```
library(readxl)
library(openxlsx)
```

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
here()
```

```
## [1] "/home/guest/TSA_Sp25"
```

```
#Importing data set without change the original file using read.xlsx
energy_data <- read_excel(path= "./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Sourc
                           skip = 12, sheet="Monthly Data",col_names=FALSE)
```

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

```r
#Now let's extract the column names from row 11
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`
```

```r
#Assign the column names to the data set
colnames(energy_data) <- read_col_names

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
# select columns
energy_df5_selected <- energy_data %>%
  select("Month",
         "Solar Energy Consumption", "Wind Energy Consumption") %>%
  mutate(across(
    c("Solar Energy Consumption", "Wind Energy Consumption"), as.numeric)) %>%
  drop_na()
```

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

```r
# Convert 'Month' to Date format if it's not already
energy_df5_selected$Month <- as.Date(energy_df5_selected$Month, format = "%Y-%m")
```

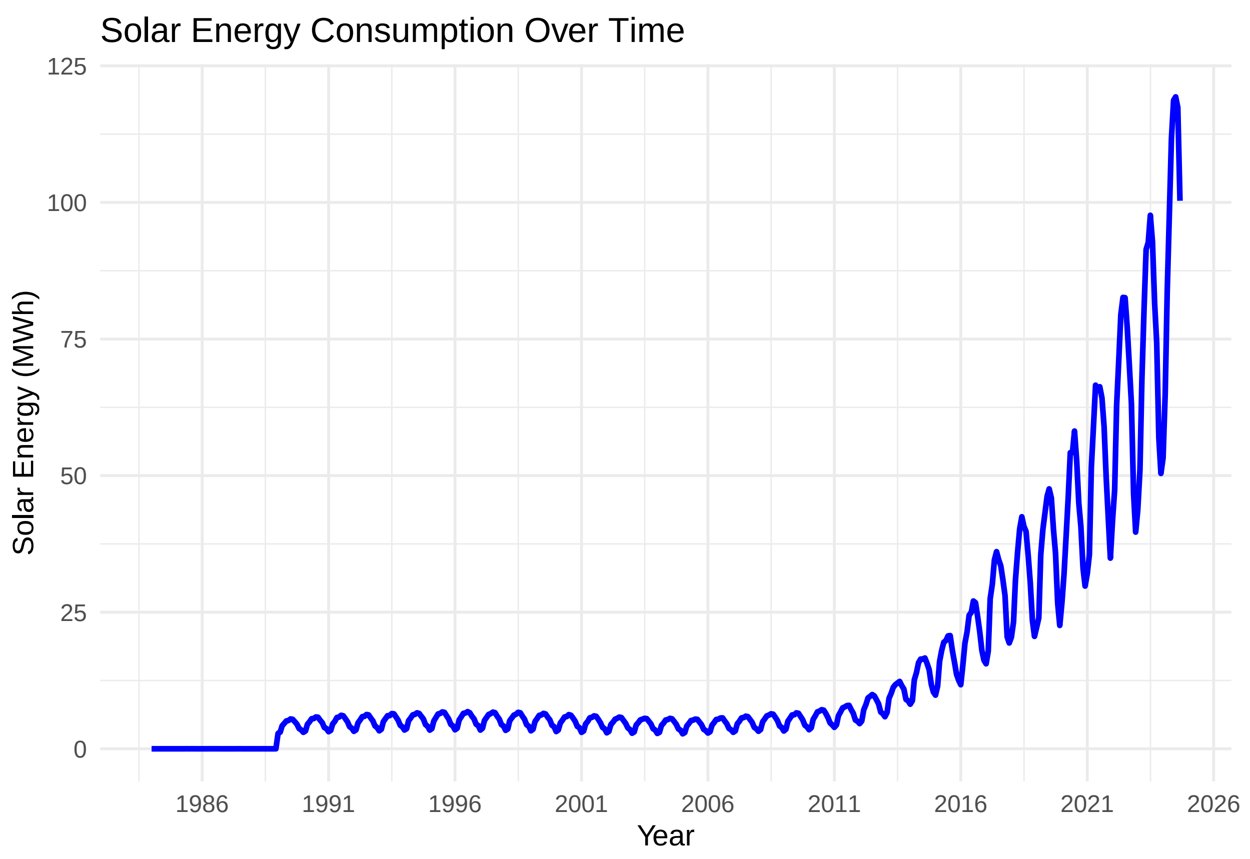
Q2

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")`")

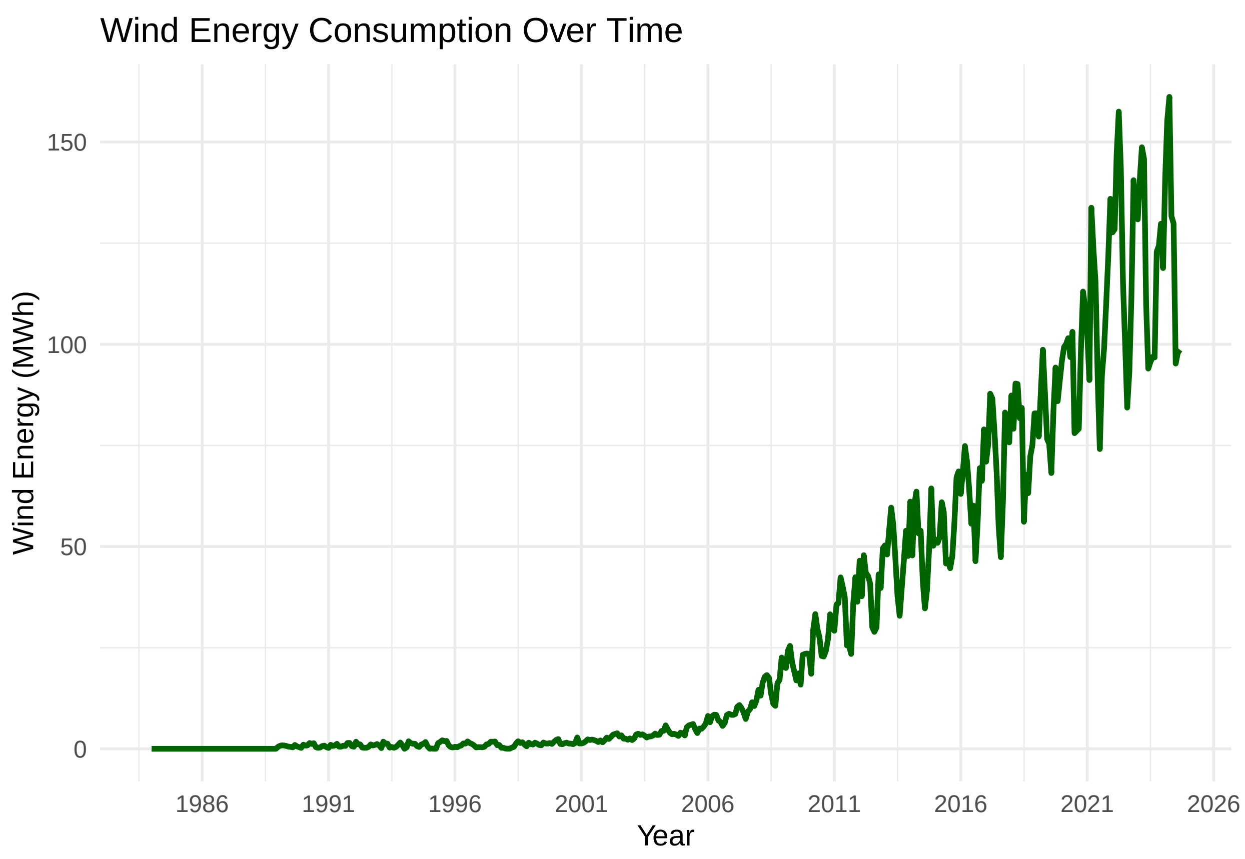
```r
# Extract start year and month
start_year <- as.numeric(format(min(energy_df5_selected$Month), "%Y"))
start_month <- as.numeric(format(min(energy_df5_selected$Month), "%m"))

# Convert to time series (monthly frequency)
solar_ts <- ts(energy_df5_selected$`Solar Energy Consumption`,
               start = c(start_year, start_month),
               frequency = 12)

wind_ts <- ts(energy_df5_selected$`Wind Energy Consumption`,
              start = c(start_year, start_month),
              frequency = 12)

# Convert time series back into a tidy dataframe for ggplot
renewables_df <- tibble(Month = energy_df5_selected$Month,
                        Solar_TS = as.numeric(solar_ts),
                        Wind_TS = as.numeric(wind_ts))

# Plot Solar Energy Consumption
solar_ts_plot <- ggplot(renewables_df, aes(x = Month, y = Solar_TS)) +
  geom_line(color = "blue", size = 1) +
  labs(title = "Solar Energy Consumption Over Time",
       x = "Year", y = "Solar Energy (MWh)") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
print(solar_ts_plot)
```



```r
# Plot Wind Energy Consumption
wind_ts_plot <- ggplot(renewables_df, aes(x = Month, y = Wind_TS)) +
  geom_line(color = "darkgreen", size = 1) +
  labs(title = "Wind Energy Consumption Over Time",
       x = "Year", y = "Wind Energy (MWh)") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  theme_minimal()
print(wind_ts_plot)
```

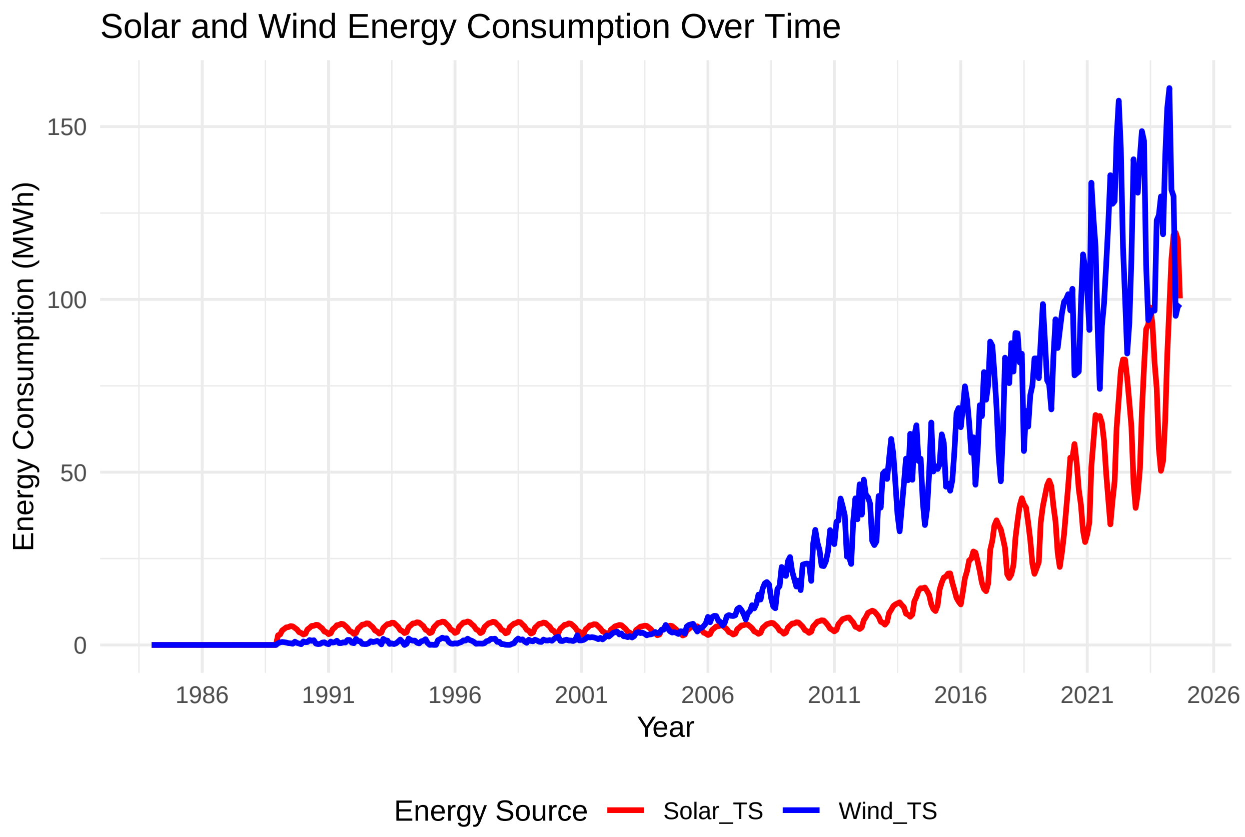


Q3

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.

```r
# Reshape the data into long format for ggplot
renewables_long <- renewables_df %>%
  pivot_longer(cols = c(Solar_TS, Wind_TS),
               names_to = "Energy_Source",
               values_to = "Energy_Consumption")

# Plot both time series in the same graph
combined_plot <- ggplot(renewables_long, aes(x = Month, y = Energy_Consumption, color = Energy_Source))
  geom_line(size = 1) +
  scale_color_manual(values = c("Solar_TS" = "red", "Wind_TS" = "blue")) +
  labs(title = "Solar and Wind Energy Consumption Over Time",
       x = "Year",
       y = "Energy Consumption (MWh)",
       color = "Energy Source") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

```r
# Print the combined plot
print(combined_plot)
```



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.

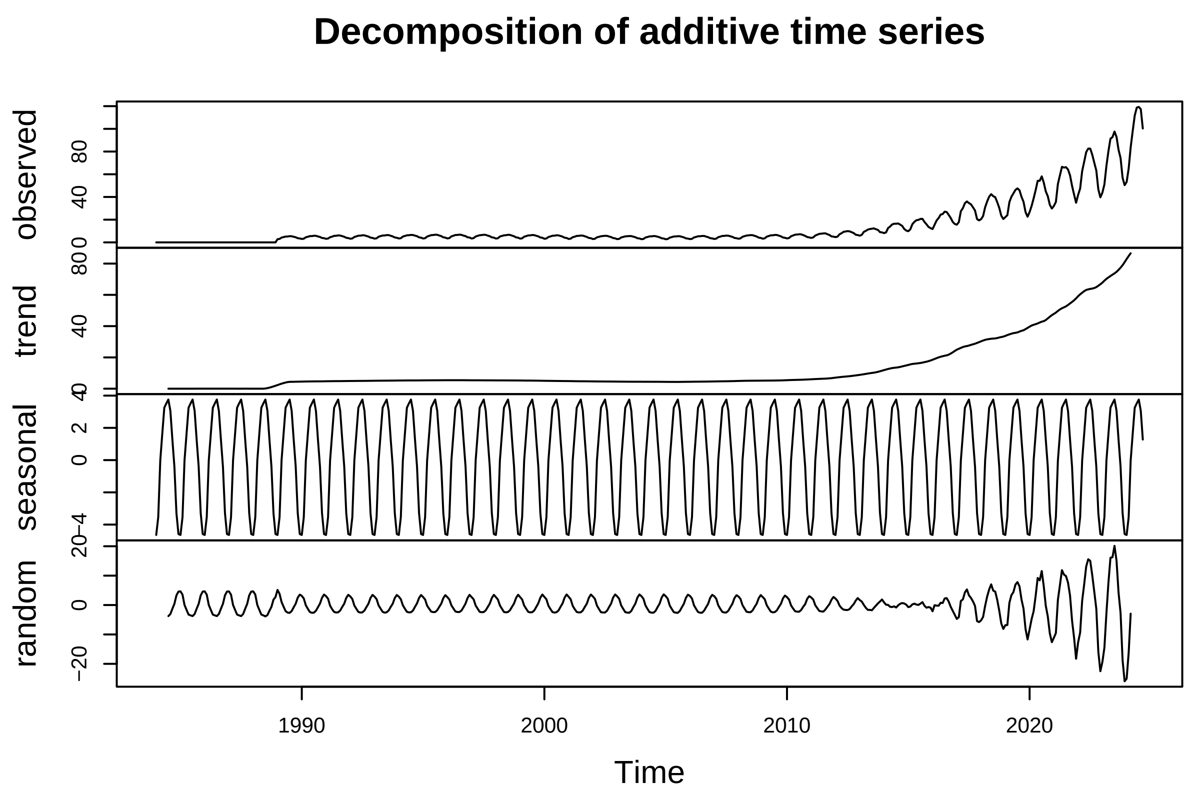
Additional info on `decompose()`.

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?

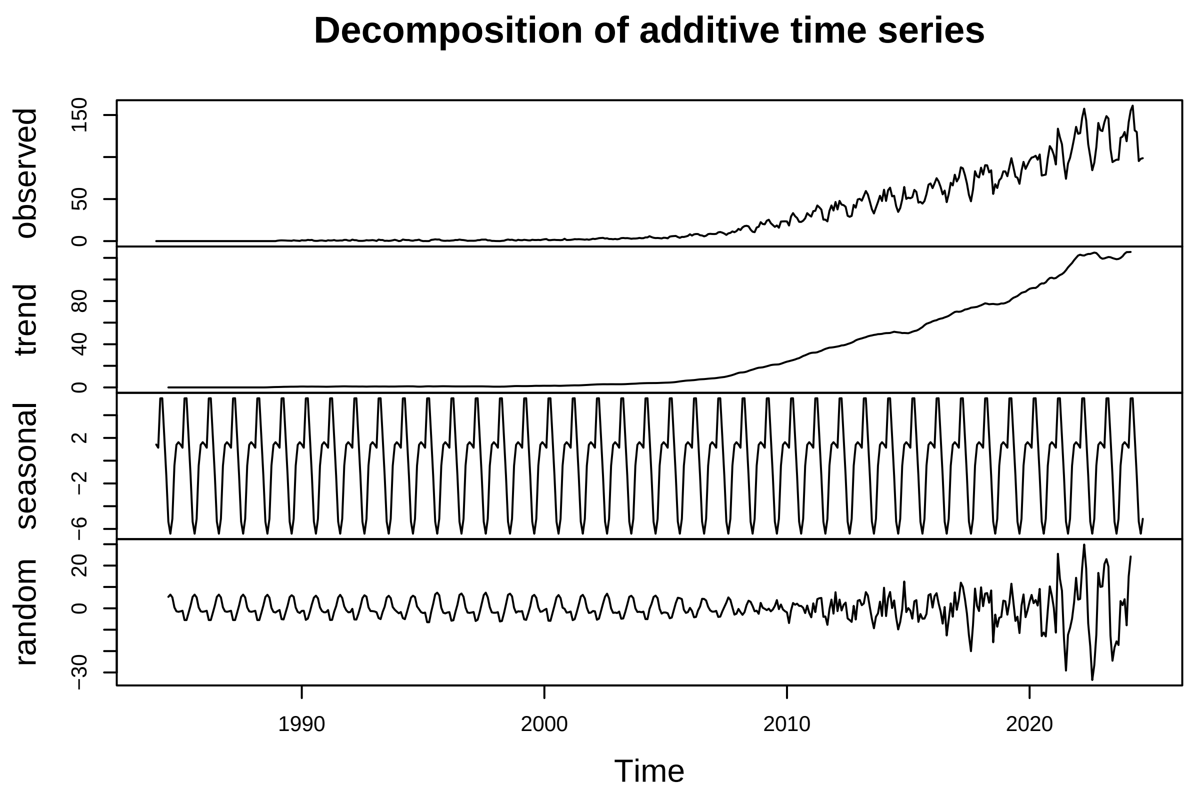
```r
# Decompose the solar and wind time series
solar_decomposed_add <- decompose(solar_ts, type = "additive")
wind_decomposed_add <- decompose(wind_ts, type = "additive")

solar_decomposed_add_plot <- plot(solar_decomposed_add)
```



```r
print(solar_decomposed_add_plot)
```

```
## NULL
```

```r
wind_decomposed_add_plot <- plot(wind_decomposed_add)
```



```r
print(wind_decomposed_add_plot)
```

```
## NULL
```

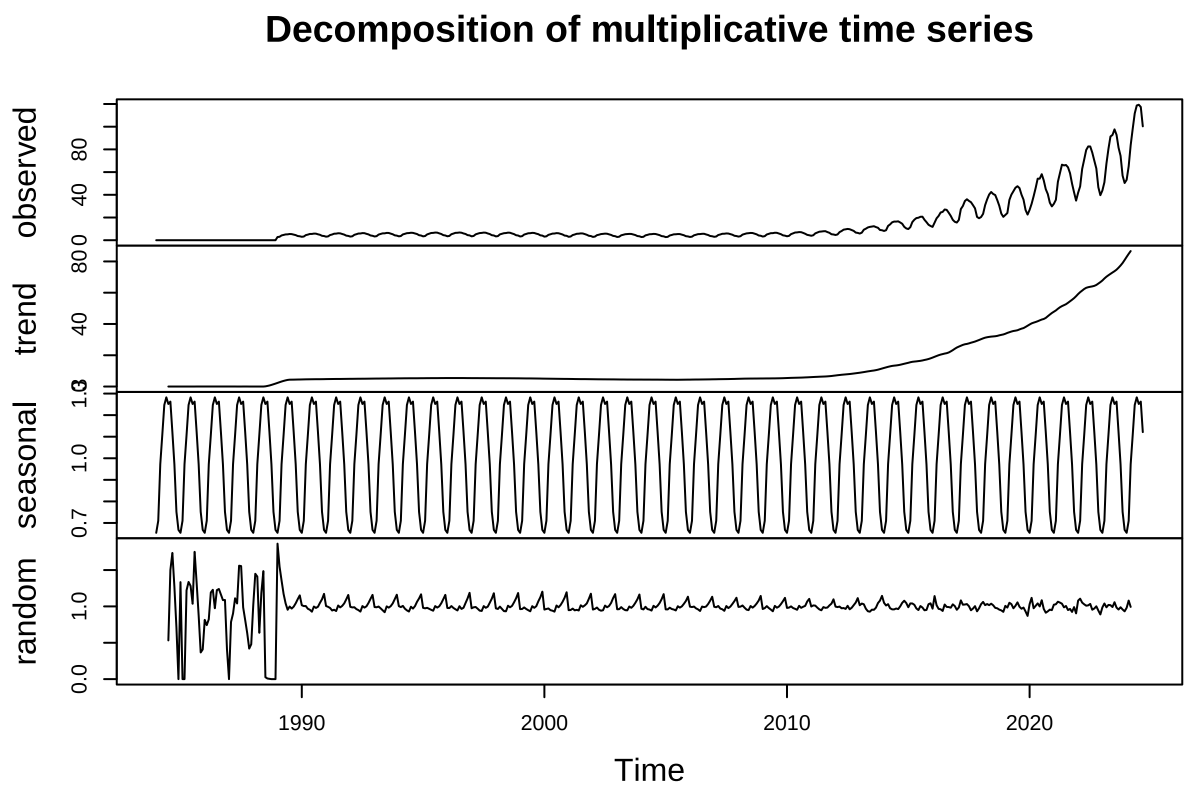
> Answer: The trends for both solar time series and wind time series are increasing overtime. In terms of the Random Component, it represents the residuals after removing the trend and seasonal components. Ideally, the random component should look like random noise (no discernible pattern). However, the decomposed solar plot displays some repeating patterns/ fluctuations, and the fluctuations have become more dramatic around year 2020. This suggest that the decomposition may not have fully captured the seasonal or trend components. The wind decomposed plot shows a little more random pattern after year 2010. Both solar and wind series plots display repeating patterns within the yearly/ monthly cycles. For energy consumption data, seasonality might reflect higher consumption in certain months (e.g., higher solar energy in summer or higher wind energy in winter).

Q5

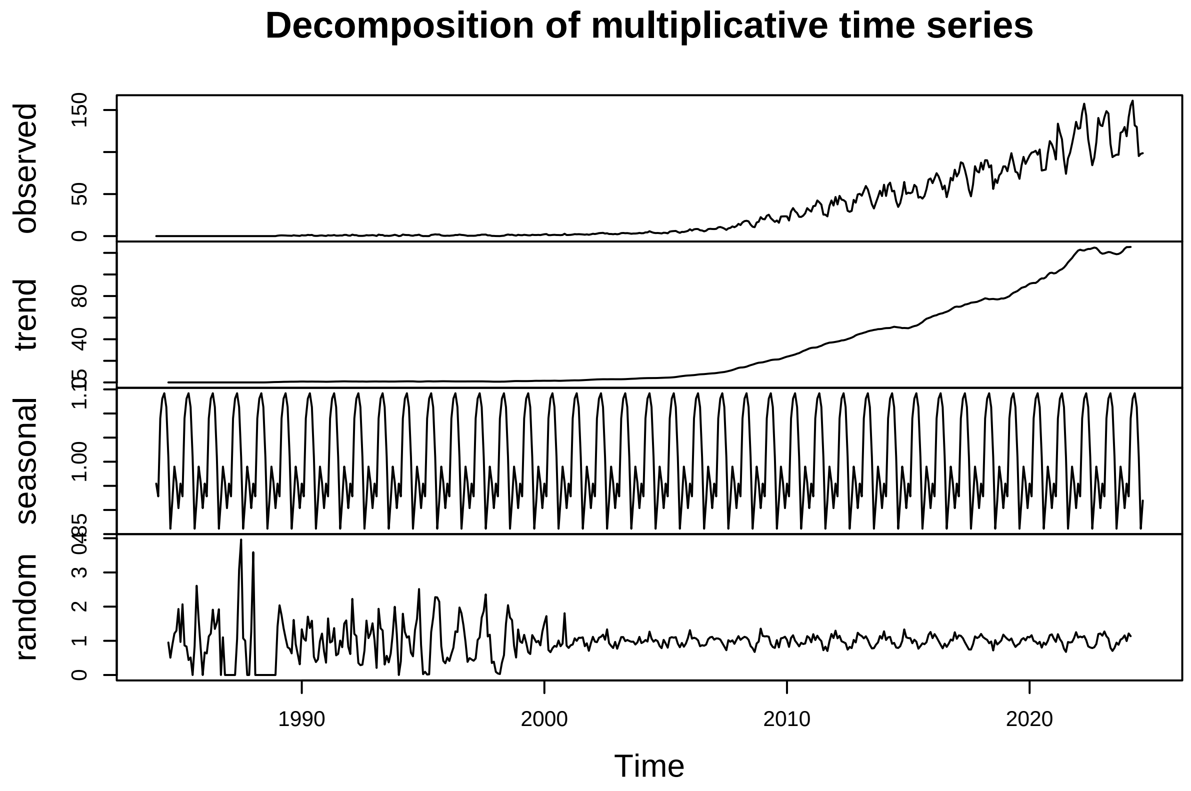
Use the decompose function again but now change the type of the seasonal component from additive to multiplicative. What happened to the random component this time?

```r
# Decompose solar and wind time series using multiplicative decomposition
solar_decomposed_multiplicative <- decompose(solar_ts, type = "multiplicative")
wind_decomposed_multiplicative <- decompose(wind_ts, type = "multiplicative")
```

```r
# Plot multiplicative plots
solar_decomposed_multi_plot <- plot(solar_decomposed_multiplicative)
```



```r
wind_decomposed_multi_plot <- plot(wind_decomposed_multiplicative)
```



Answer: In question 4, the random component did not display as much of a randomness, and degree of fluctuations were more dramatic after year 2010. For here, both of the multiplicative models show more dramatic fluctuations before year 2000. In addition, there is overall more randomness displayed in the random component of the multiplicative model.

Q6

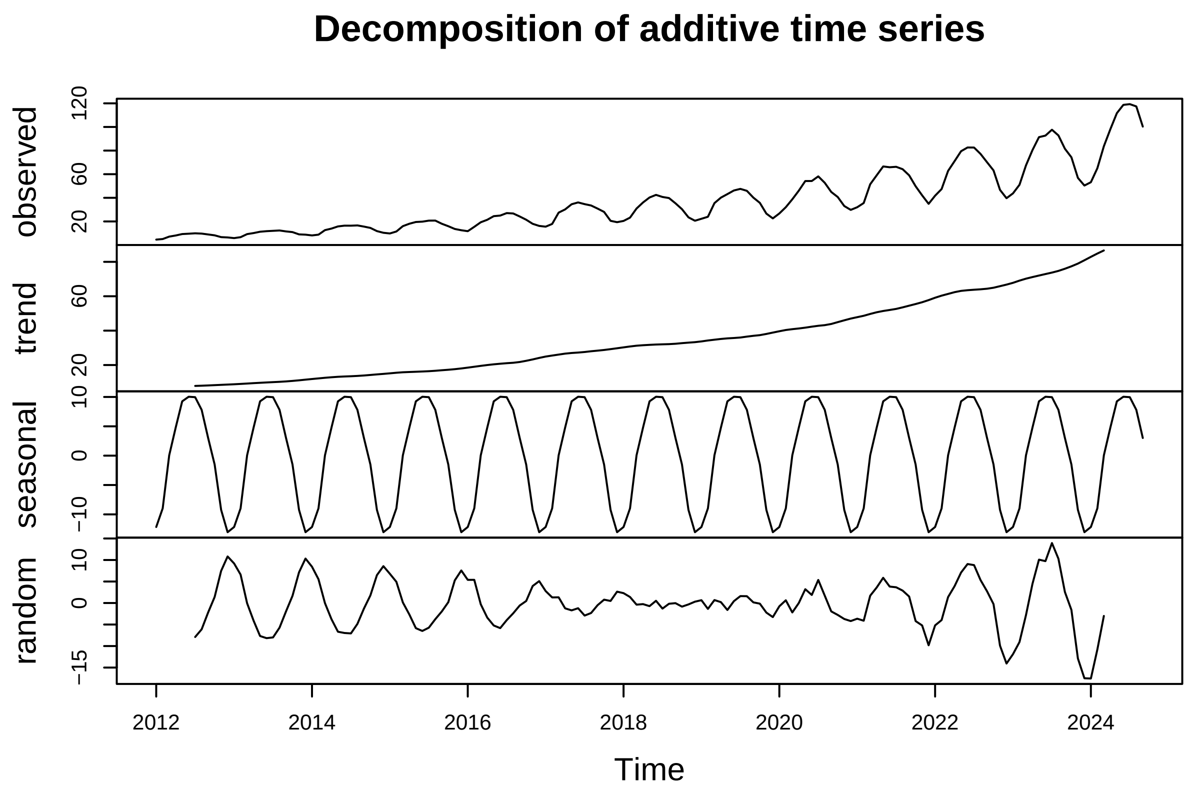
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

Answer: According to the solar ts trend and its random component in the decomposed plots, the data before year 1990 and after 2020 display different and more random patterns. Meanwhile, the wind series data before 2000 and after 2020 display more randomness and less consistent patterns compared to other time periods. Therefore, we might not want to include all historical data for consistency of forecasting. In addition, if historical data is sparse or inconsistent, it may not provide enough information to improve the model's performance. Some more information to help the forecasting process could be assessing significant structural changes in the energy market, such as technologica advancements, policy changes, or shifts in the energy demand.

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.

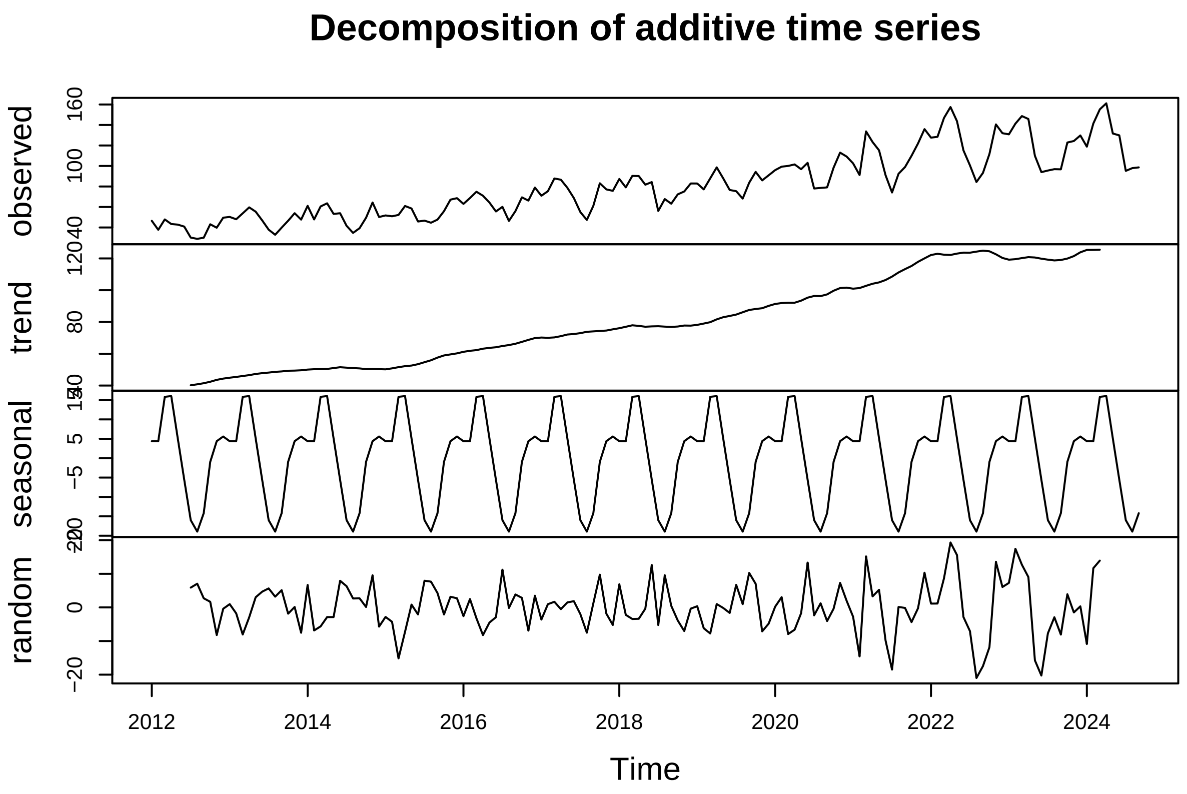
```r
# Filter data to start from January 2012
energy_df_2012 <- energy_df5_selected %>%
  filter(year(Month) >= 2012)

# Extract start year and month
start_year <- 2012
start_month <- as.numeric(format(min(energy_df_2012$Month), "%m"))

# Convert to time series (monthly)
solar_ts_2012 <- ts(energy_df_2012$`Solar Energy Consumption`,
                    start = c(start_year, start_month),
                    frequency = 12)

wind_ts_2012 <- ts(energy_df_2012$`Wind Energy Consumption`,
                   start = c(start_year, start_month),
                   frequency = 12)

# Apply additive decomposition
solar_decomp_2012 <- decompose(solar_ts_2012, type = "additive")
wind_decomp_2012 <- decompose(wind_ts_2012, type = "additive")

solar_2012_plot <- plot(solar_decomp_2012)
```



```
print(solar_2012_plot)
```

```
## NULL
```

```
wind_2012_plot <- plot(wind_decomp_2012)
```



```
print(wind_2012_plot)
```

```
## NULL
```

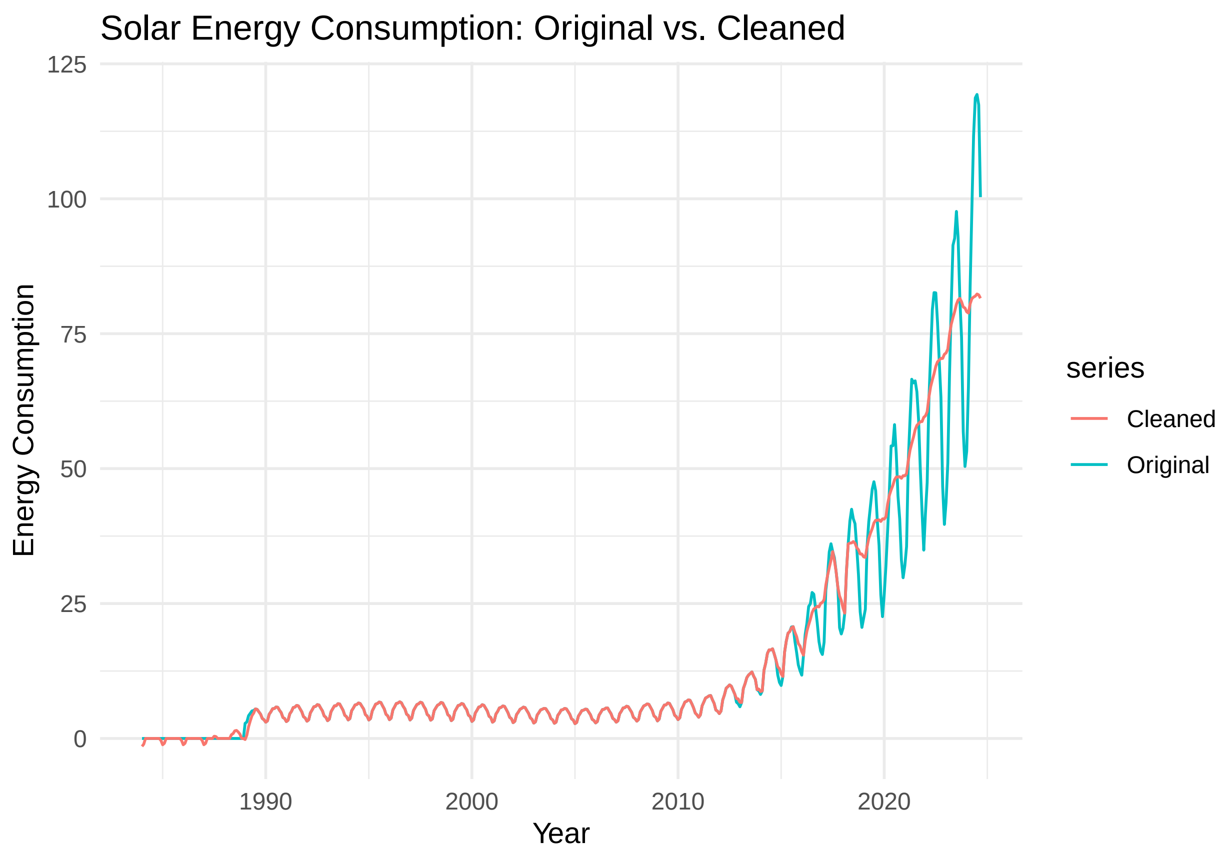
Answer: The random component of the 2012 onward solar series appears to be more random than the original series, with slightly more seasonal patterns from year 2012-2016. Meanwhile, the random component of the wind series appears to be random from year 2012 to 2024. In an additive model, the seasonal effect remains constant over time, regardless of the overall level of the series, as shown in the seasonal component in the decomposed series from year 2012 onward.

Identify and Remove outliers

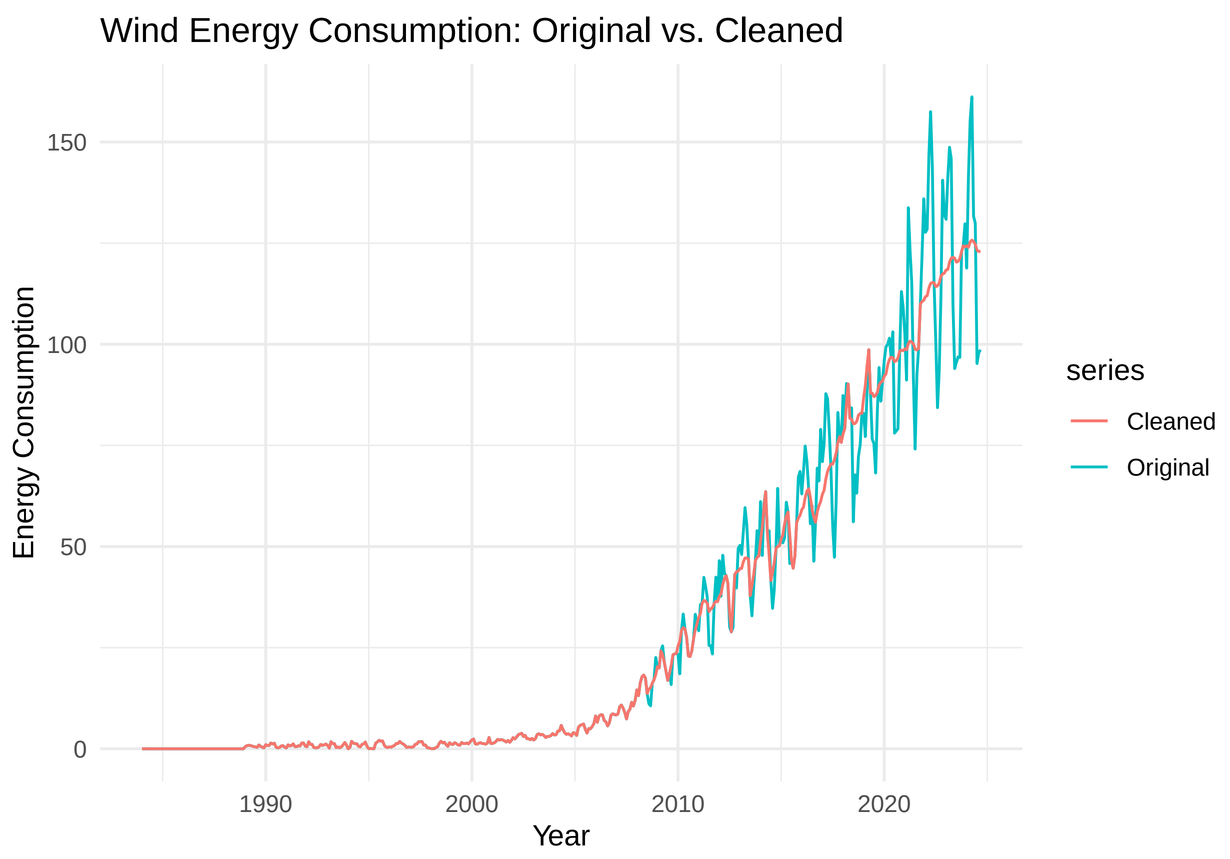
Q8

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.

```r
# tsclean() to remove outliers
solar_ts_clean <- tsclean(solar_ts)
wind_ts_clean <- tsclean(wind_ts)

# Plot Solar Energy Consumption: Original vs. Cleaned
solar_comparison <- autoplot(solar_ts, series = "Original") +
  autolayer(solar_ts_clean, series = "Cleaned") +
  labs(title = "Solar Energy Consumption: Original vs. Cleaned",
       x = "Year", y = "Energy Consumption") +
  theme_minimal()

# Plot Wind Energy Consumption: Original vs. Cleaned
wind_comparison <- autoplot(wind_ts, series = "Original") +
  autolayer(wind_ts_clean, series = "Cleaned") +
  labs(title = "Wind Energy Consumption: Original vs. Cleaned",
       x = "Year", y = "Energy Consumption") +
  theme_minimal()

print(solar_comparison)
```



```r
print(wind_comparison)
```

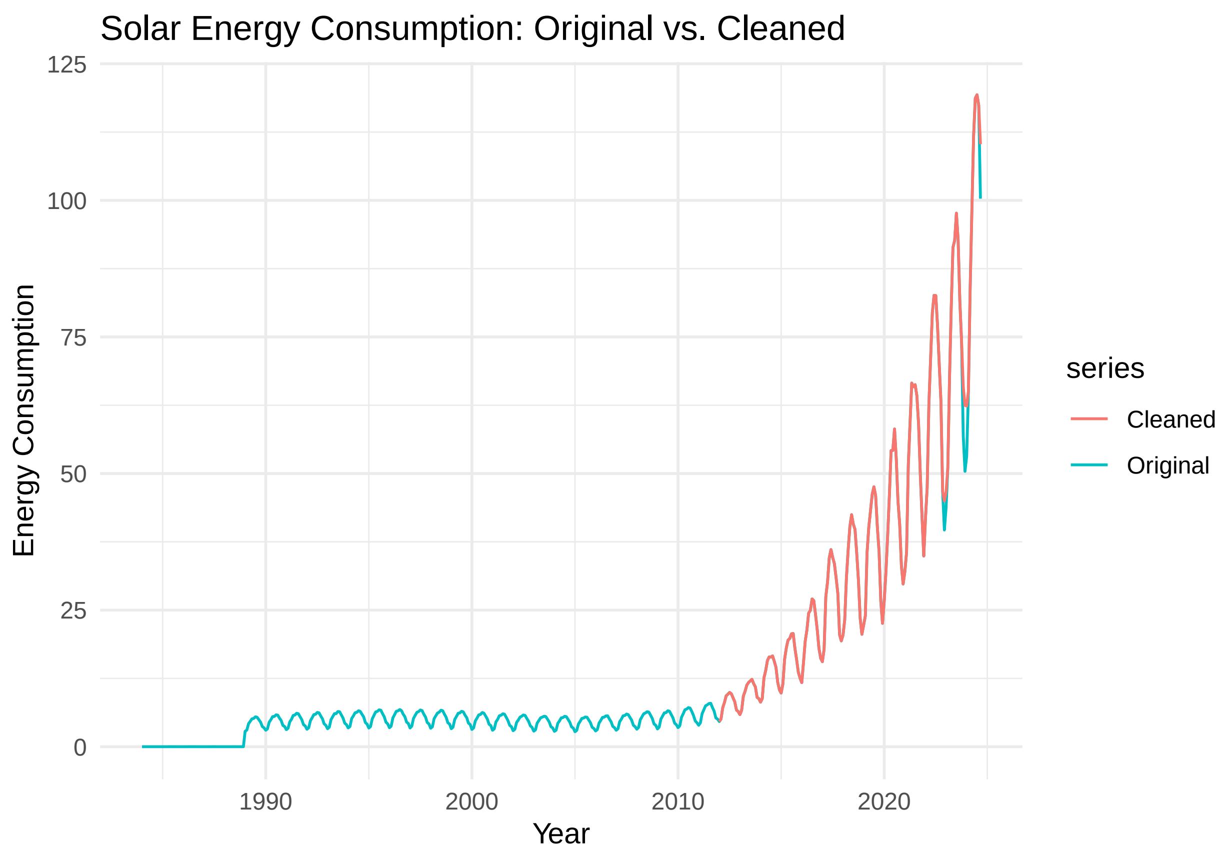


Answer: In terms of the solar energy consumption series, the cleaned series shows more dramatic yet almost repeating/ consistent fluctuations in the for roughly year 2016 onward. In terms of the wind energy consumption series, the cleaned series shows more dramatic but random fluctuations for roughly year 2010 onward.

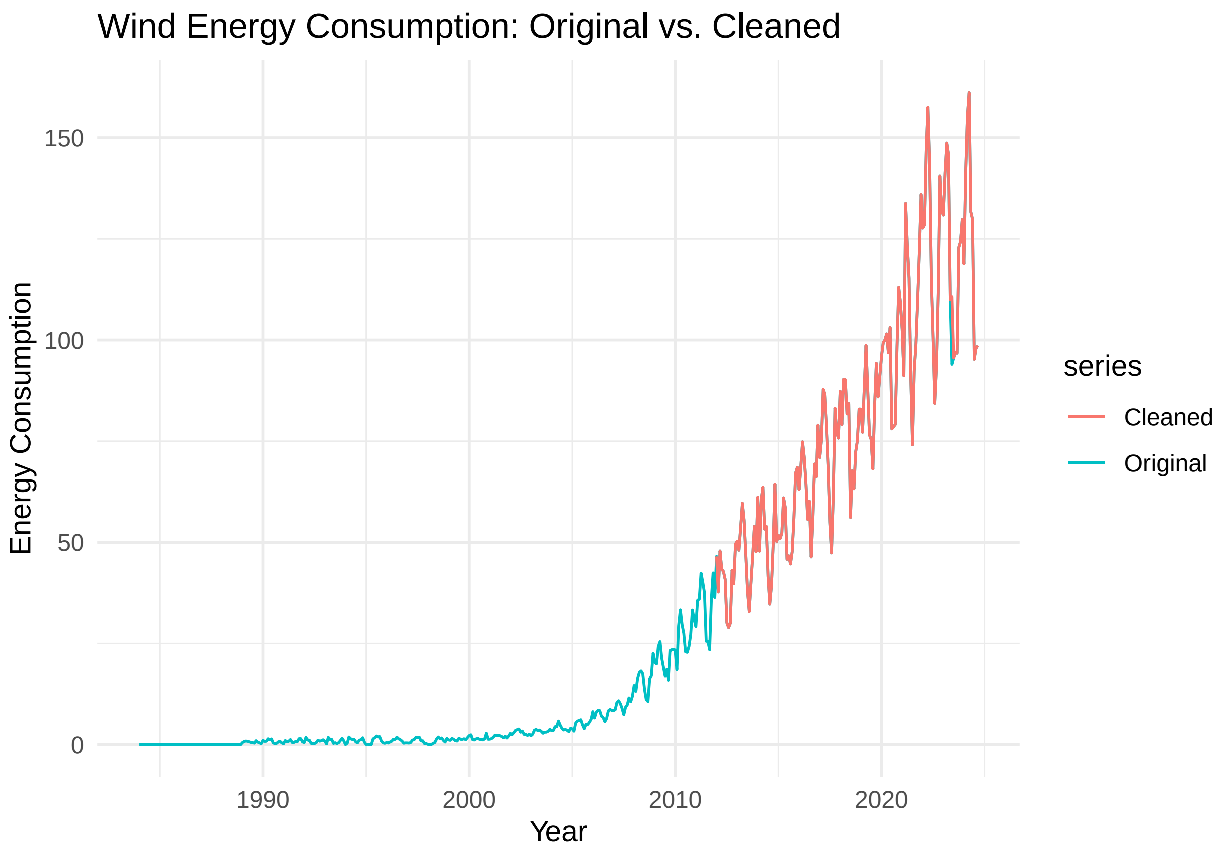
Q9

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
# tsclean() to remove outliers
solar_ts_2012_clean <- tsclean(solar_ts_2012)
wind_ts_2012_clean <- tsclean(wind_ts_2012)

# Plot Solar Energy Consumption: Original vs. Cleaned
solar_comparison2012 <- autoplot(solar_ts, series = "Original") +
  autolayer(solar_ts_2012_clean, series = "Cleaned") +
  labs(title = "Solar Energy Consumption: Original vs. Cleaned",
       x = "Year", y = "Energy Consumption") +
  theme_minimal()

# Plot Wind Energy Consumption: Original vs. Cleaned
wind_comparison2012 <- autoplot(wind_ts, series = "Original") +
  autolayer(wind_ts_2012_clean, series = "Cleaned") +
  labs(title = "Wind Energy Consumption: Original vs. Cleaned",
       x = "Year", y = "Energy Consumption") +
  theme_minimal()

print(solar_comparison2012)
```



```
print(wind_comparison2012)
```



Answer: According to the comparison plots between the cleaned versus the original Solar Energy Consumption and Wind Energy Consumption for year 2012 onward, there did not display much difference between cleaned and original series for both solar and wind. The function removed outliers significantly.