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

Assignment 5 - Due date 02/18/25

Lucy Wang

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

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.

```
#Importing data set - using xlsx package
energy_data <- read_excel(path="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source
                          skip = 12,
                          sheet="Monthly Data",col_names=FALSE)

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

```
read_col_names <- read_excel(path="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Sou
                             skip = 10,n_max = 1,
                             sheet="Monthly Data",col_names=FALSE)

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

```
## # 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_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!

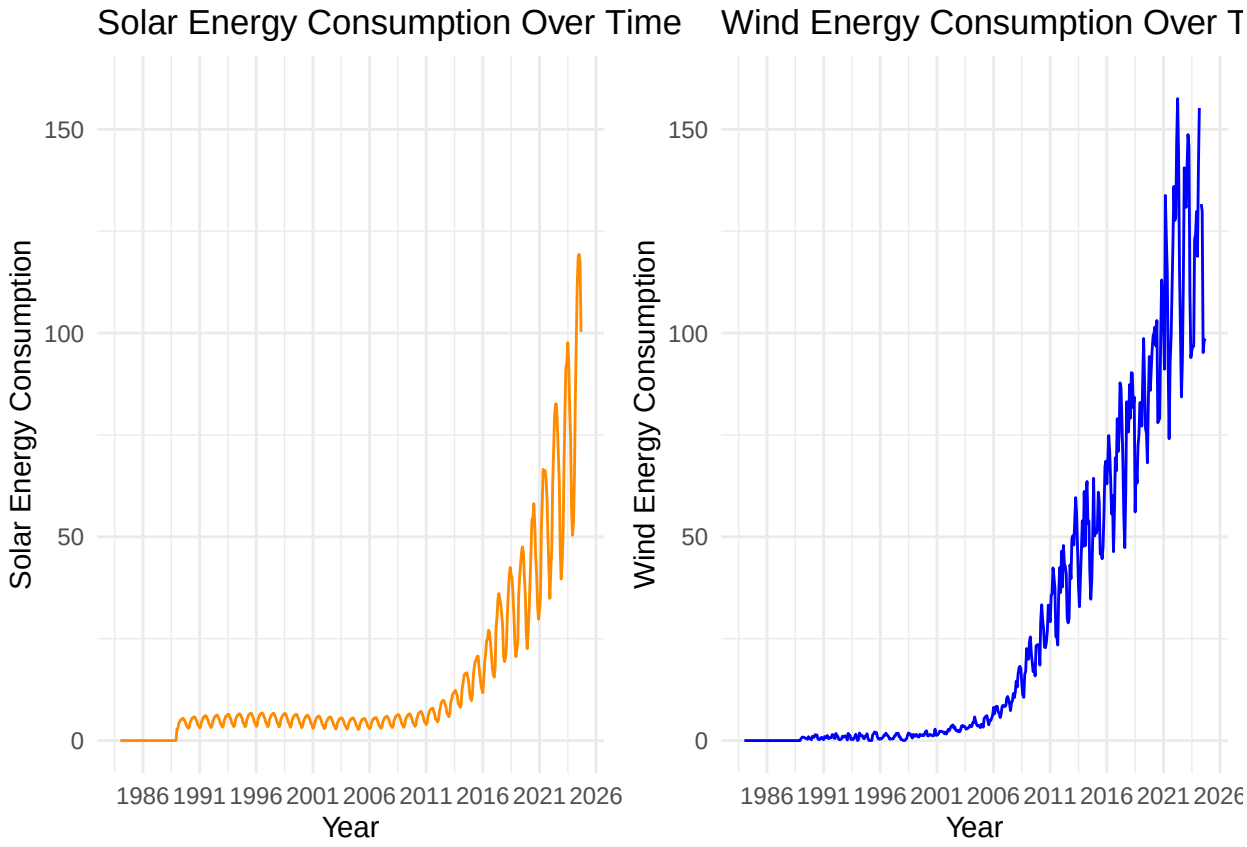
```
solar_wind_df <- as.data.frame(subset(energy_data,
                                      select = c('Month',
                                                 'Solar Energy Consumption',
                                                 'Wind Energy Consumption'))) %>%
  mutate_at(vars('Solar Energy Consumption','Wind Energy Consumption'), as.numeric) %>%
  drop_na('Solar Energy Consumption','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")")`

```
solar_wind_df$Month <- as.Date(solar_wind_df$Month)

# Solar Energy Consumption Plot
solar_plot <- ggplot(solar_wind_df, aes(x = Month, y = `Solar Energy Consumption`)) +
  geom_line(color = "darkorange") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
```

```
  xlab("Year") +
  ylab("Solar Energy Consumption") +
  ylim(0,160)+
  ggtitle("Solar Energy Consumption Over Time") +
  theme_minimal()

# Wind Energy Consumption Plot
wind_plot <- ggplot(solar_wind_df, aes(x = Month, y = `Wind Energy Consumption`)) +
  geom_line(color = "blue") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  xlab("Year") +
  ylab("Wind Energy Consumption") +
  ylim(0,160)+
  ggtitle("Wind Energy Consumption Over Time") +
  theme_minimal()

plot_grid(solar_plot, wind_plot, ncol = 2)
```

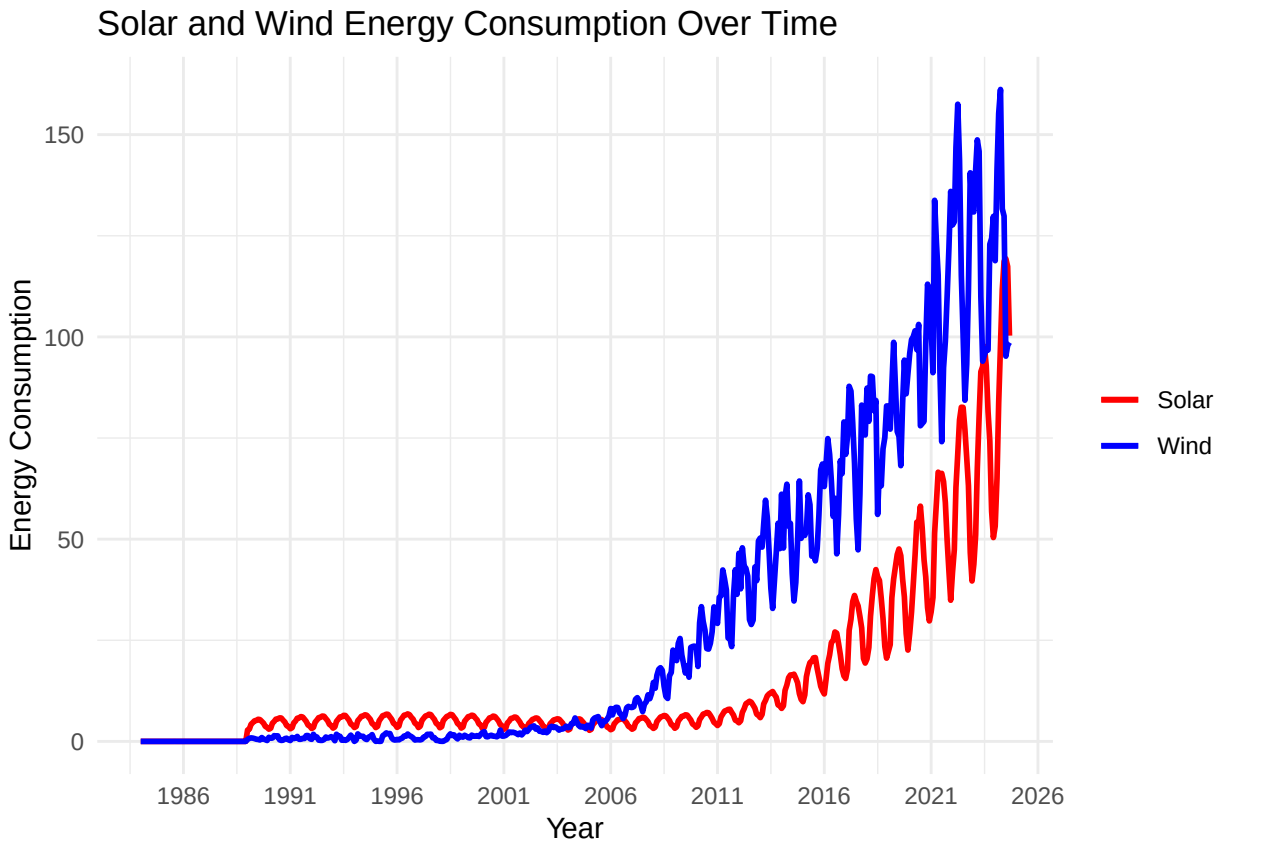


Q3

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.

```
solar_wind_plot <- ggplot(solar_wind_df, aes(x=Month))+
  geom_line(aes(y = `Solar Energy Consumption`, color = "Solar"), size = 1) +
```

```
  geom_line(aes(y = `Wind Energy Consumption`, color = "Wind"), size = 1) +
  scale_color_manual(values = c("Solar" = "red", "Wind" = "blue")) +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  xlab("Year") +
  ylab("Energy Consumption") +
  ggtitle("Solar and Wind Energy Consumption Over Time") +
  theme_minimal() +
  theme(legend.title = element_blank())

solar_wind_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.

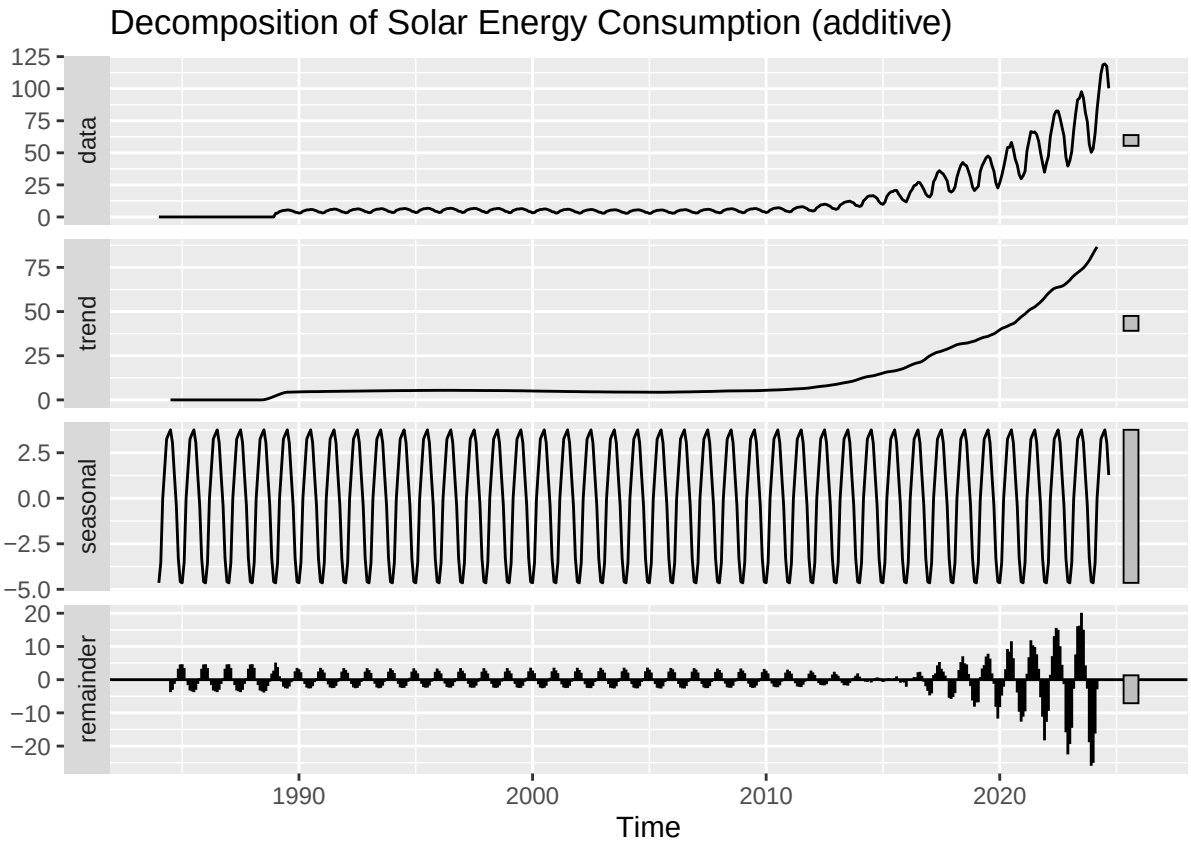
Additional info on `decompose()`.

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.

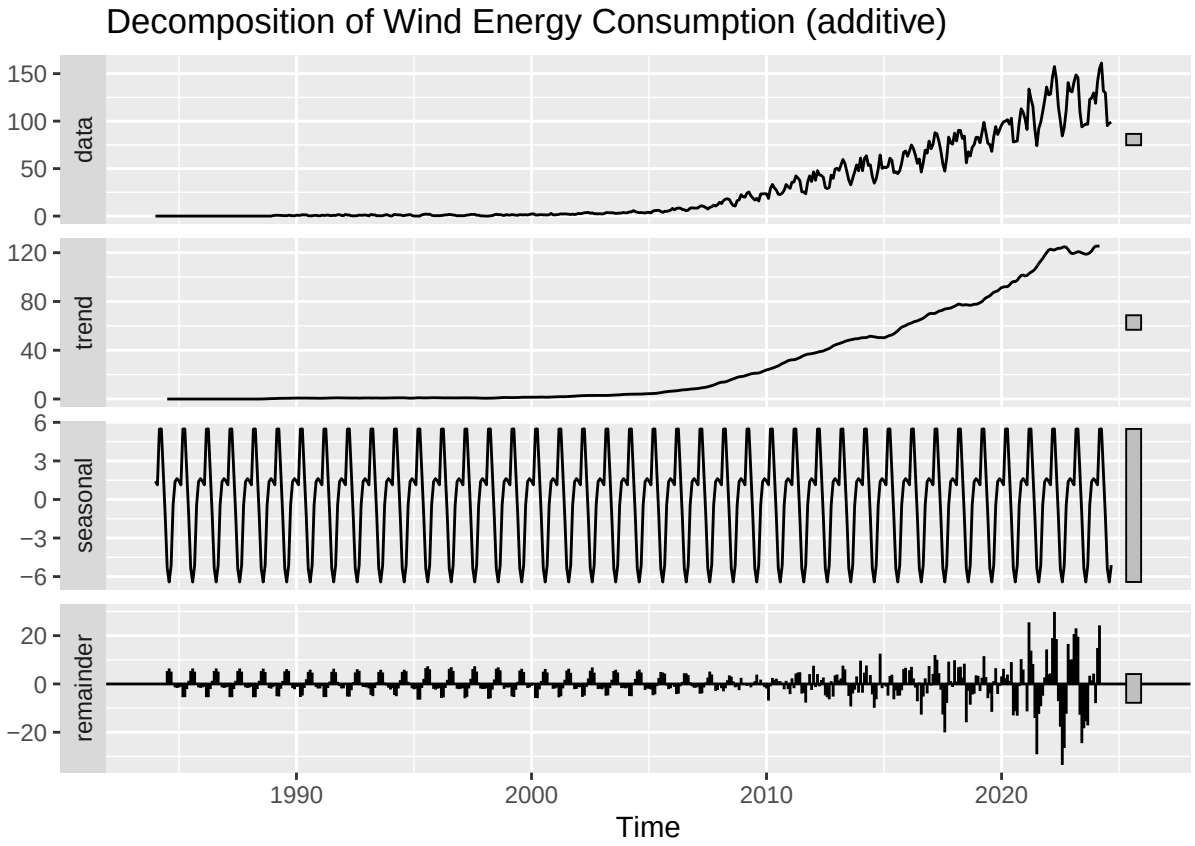
Q4

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?

```
# Convert to time series objects
solar_ts <- ts(solar_wind_df$`Solar Energy Consumption`, start=c(1984,1), frequency = 12)
wind_ts <- ts(solar_wind_df$`Wind Energy Consumption`, start=c(1984,1), frequency = 12)

# Apply decomposition
solar_decomp_a <- decompose(solar_ts, type = "additive")
wind_decomp_a <- decompose(wind_ts, type = "additive")

# Plot the decomposed components
solar_decomp_a_plot <- autoplot(solar_decomp_a) +
  ggtitle("Decomposition of Solar Energy Consumption (additive)")
wind_decomp_a_plot <- autoplot(wind_decomp_a) +
  ggtitle("Decomposition of Wind Energy Consumption (additive)")

solar_decomp_a_plot
```




```
wind_decomp_a_plot
```

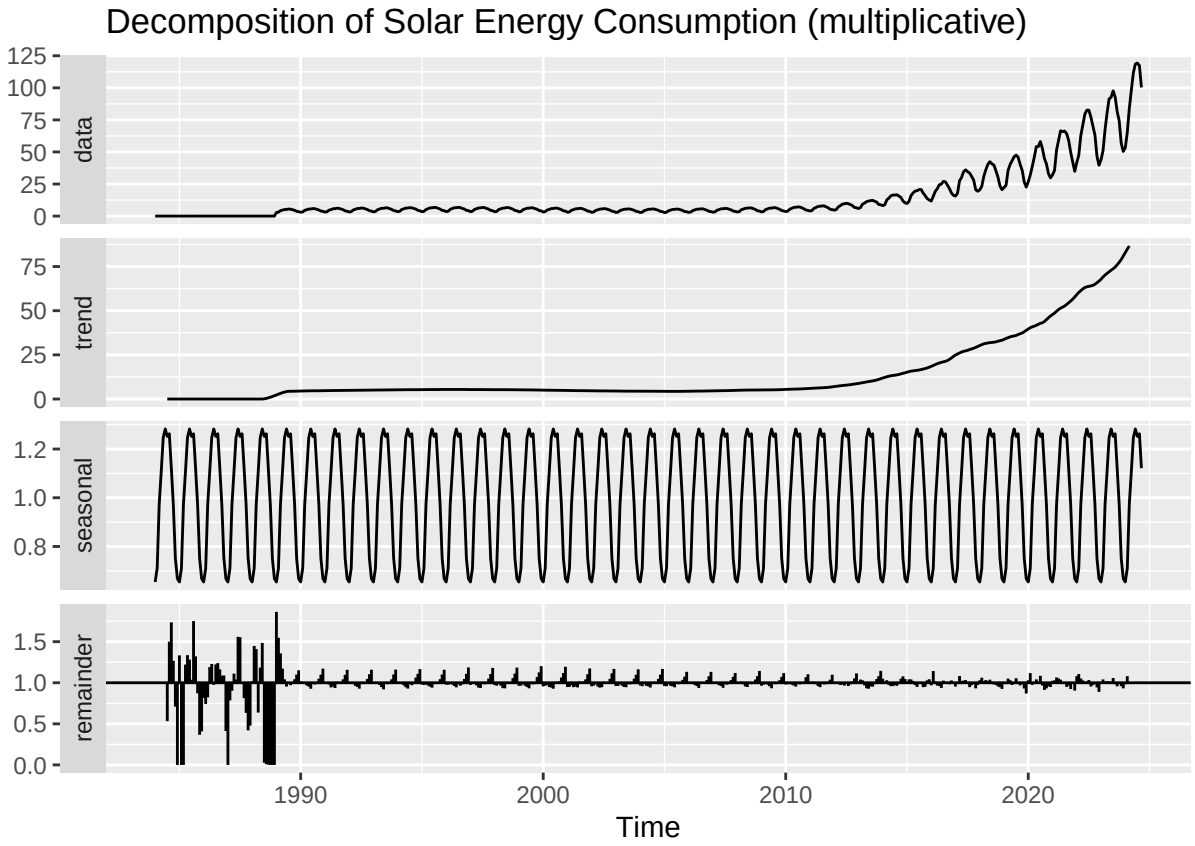


Trend Component: Both wind and solar shows a long-term upward trend. While solar starts to experience a stronger growth from 2010, the growth of wind starts earlier from around 2005. The increasing trend of solar is smoother than wind, as wind consumption fluctuates slightly in recent years. **Random Component:** Wind's and solar's random component does not look as random, especially for solar. The remainder component shows notable rotating spikes and increasing variance, particularly in recent years. The pattern is especially observable for solar, indicating that some seasonal component might still exist. **Seasonal Component:** The seasonal component is strong for both solar and wind. Wind experience different levels of consumption throughout the seasons due to seasonal wind conditions. Solar experience more of a symmetric seasonal pattern where peaks in some months than others.

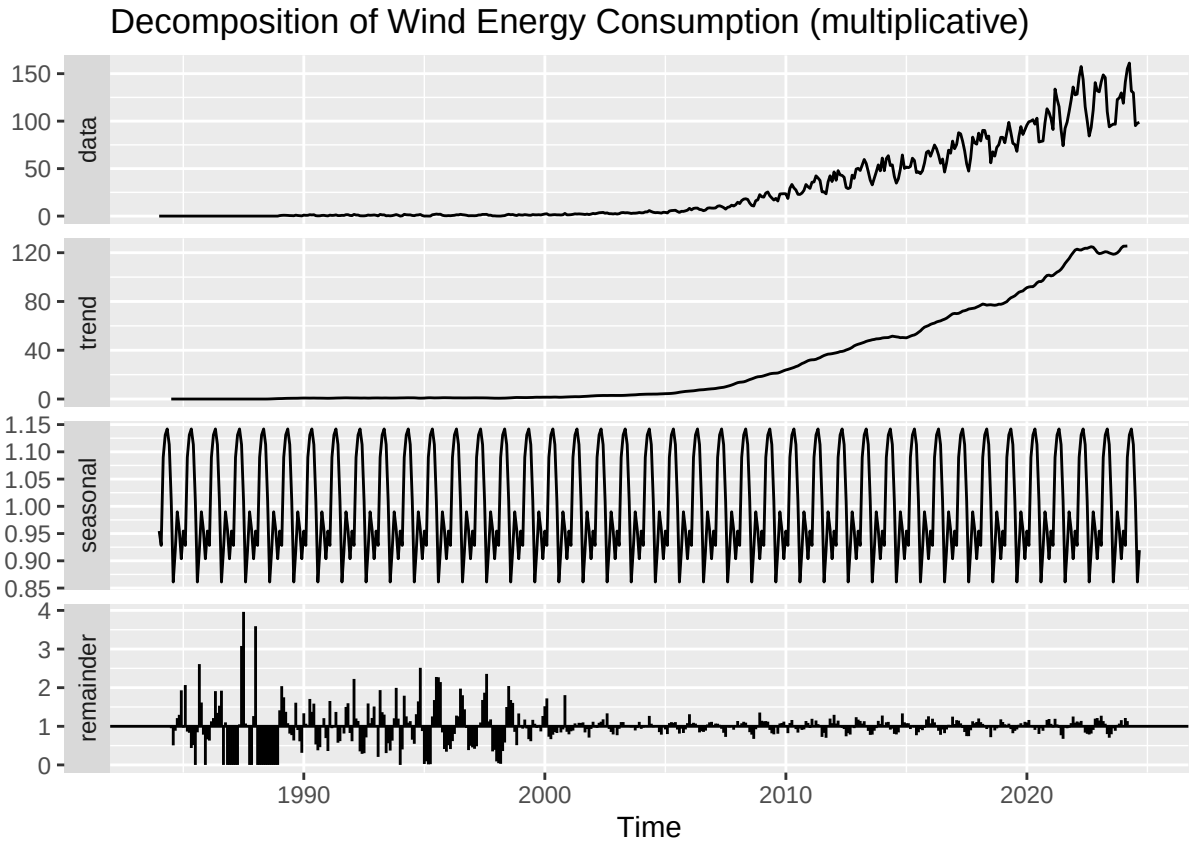
Q5

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?

```
# Apply decomposition
solar_decomp_m <- decompose(solar_ts, type = "multiplicative")
wind_decomp_m <- decompose(wind_ts, type = "multiplicative")

# Plot the decomposed components
solar_decomp_m_plot <- autoplot(solar_decomp_m) +
  ggtitle("Decomposition of Solar Energy Consumption (multiplicative)")
wind_decomp_m_plot <- autoplot(wind_decomp_m) +
  ggtitle("Decomposition of Wind Energy Consumption (multiplicative)")

solar_decomp_m_plot
```




```
wind_decomp_m_plot
```



Comparing with the output in the additive model, the random component looks more random with reduced periodic pattern in the multiplicative model. Though there are spikes in early years, the random component in recent years seem to have stablized.

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.

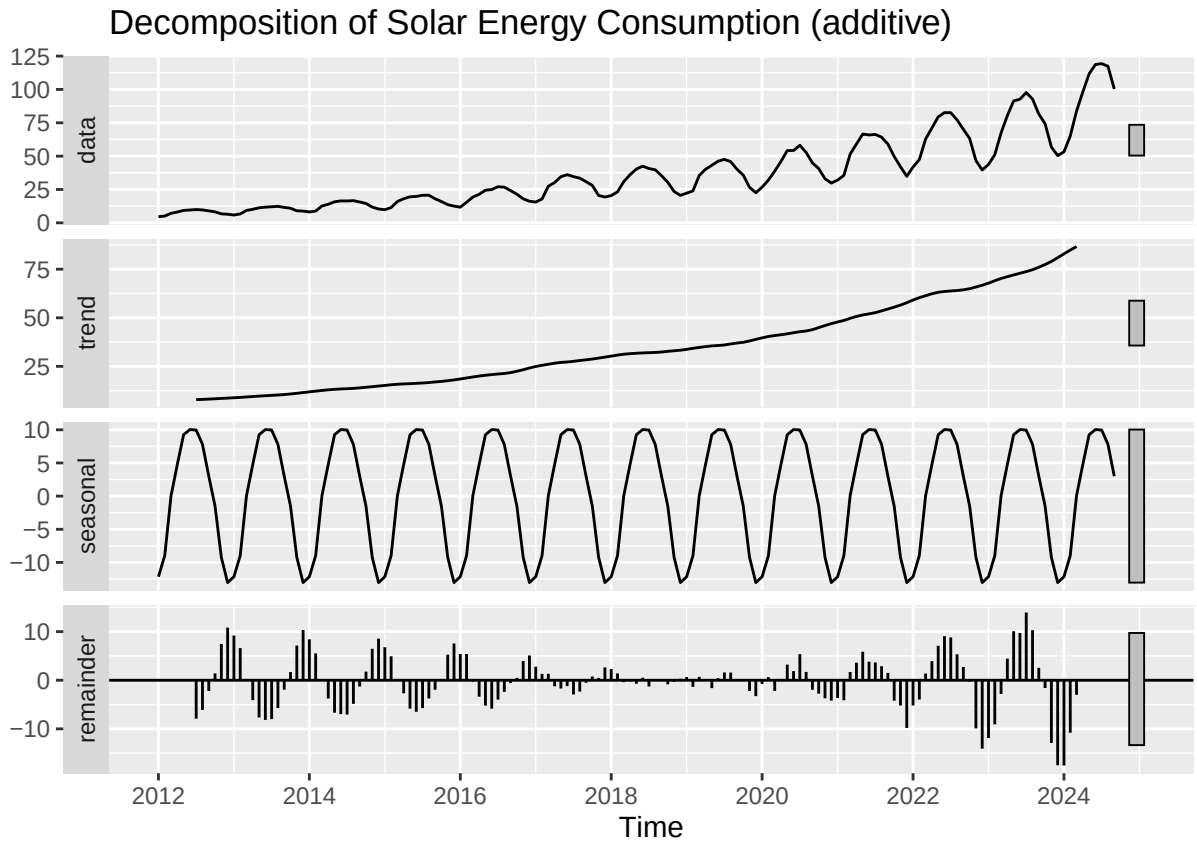
> Answer: We do not need all the historical data. Especially if to forecast the next six month consumption, the data 30 years ago or even years ago might not be relevant. For example, wind consumption in recent years fluctuates while it was increasing more firmly years ago. Therefore, you don't need the upward trend several years ago to forecast future six months' consumption that is more likely to follow the current fluctuating trend. However, the overall trend (increasing/decreasing) in recent periods and seasonal patterns observed in past years are important for forecasting. The observed seasonal vairation in the recent years would be useful to predict the consumption in the next six months.

Q7

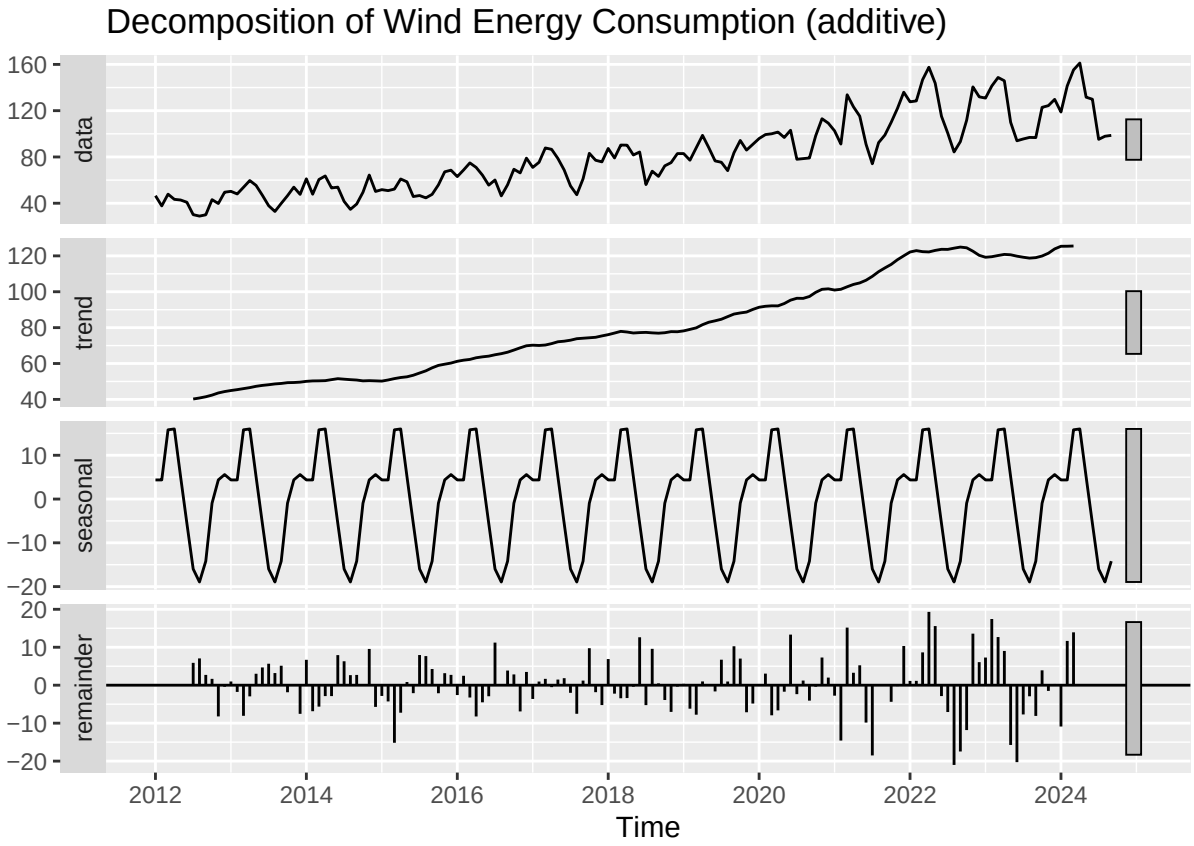
Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the decompose function `type=additive` to this new time series. Comment the results. Does the random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

```
# Filter data starts on January 2012
filtered_df <- solar_wind_df %>%
```

```r
  filter(year(Month) >= 2012)

# Convert to time series objects
solar_ts_2012 <- ts(filtered_df$`Solar Energy Consumption`, start=c(2012,1), frequency=12)
wind_ts_2012 <- ts(filtered_df$`Wind Energy Consumption`, start=c(2012,1), frequency=12)

# Apply decomposition
solar_decomp_a_2012 <- decompose(solar_ts_2012, type = "additive")
wind_decomp_a_2012 <- decompose(wind_ts_2012, type = "additive")

# Plot the decomposed components
solar_decomp_m_plot <- autoplot(solar_decomp_a_2012) +
  ggtitle("Decomposition of Solar Energy Consumption (additive)")
wind_decomp_m_plot <- autoplot(wind_decomp_a_2012) +
  ggtitle("Decomposition of Wind Energy Consumption (additive)")

solar_decomp_m_plot
```




```r
wind_decomp_m_plot
```



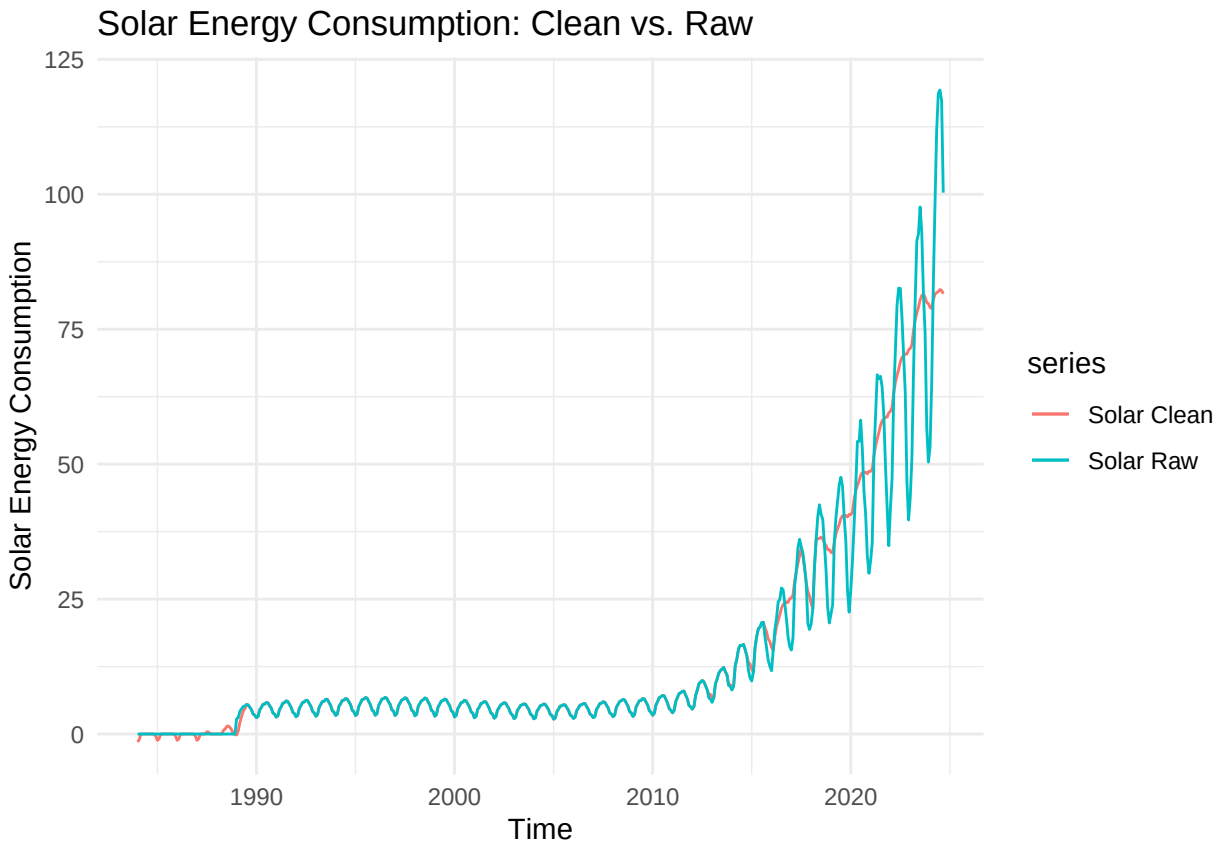
Answer: Removing older data improved the model suitablity. The trend component shows strong upward trend for both, magnifying the trend after 2012 where both begins to grow (though wind trends to fluctuate in recent years). The random component looks more random after filtering the timeframe, with weaker spikes or variations in recent years. In the early years, solar and wind consumtion growth might be driven by some structural change that fades in the recent decade. Therefore, removing old data improves the model's ability to capture the seasonal variability.

Identify and Remove outliers

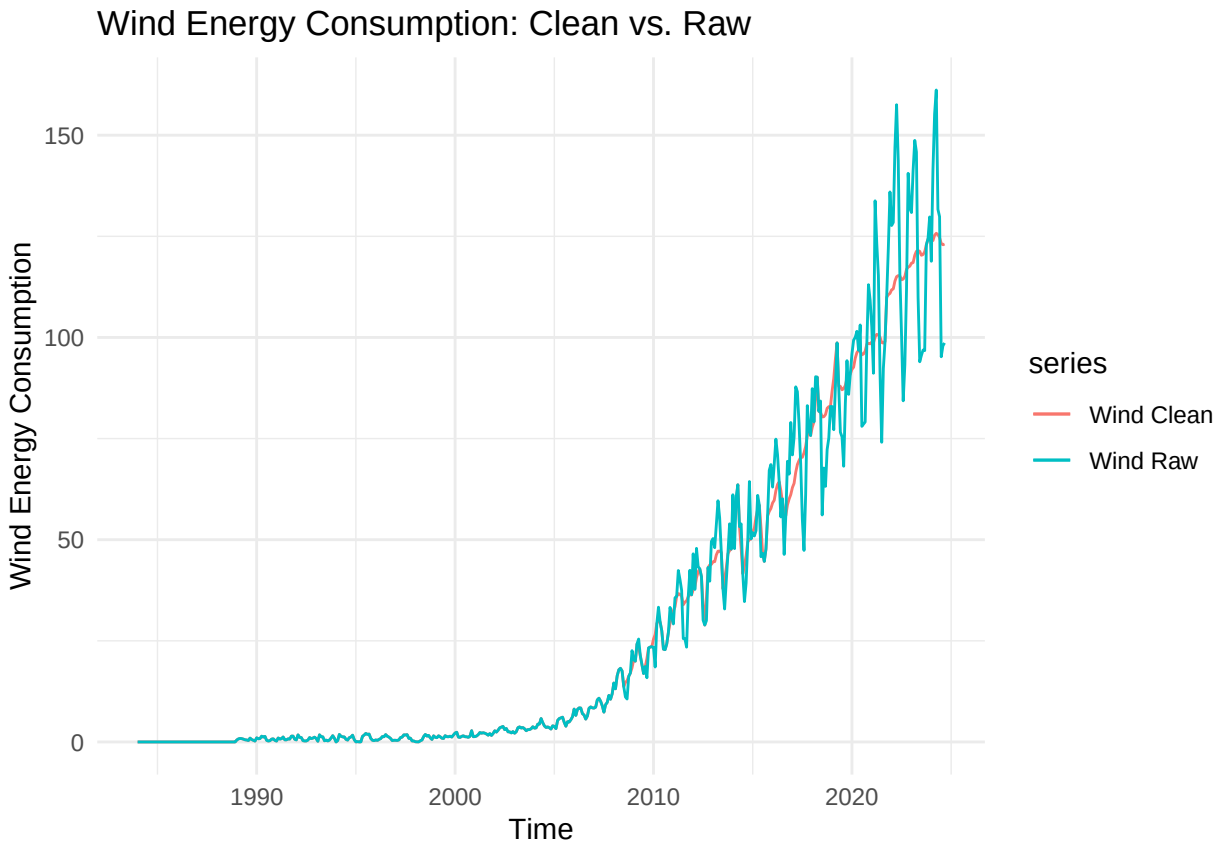
Q8

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.

```
# Apply tsclean() to both series form Q7
clean_solar_ts <- tsclean(solar_ts)
clean_wind_ts <- tsclean(wind_ts)

# Plot clean and original ts
autoplot(clean_solar_ts, series="Solar Clean")+
  autolayer(solar_ts, series="Solar Raw") +
  ylab('Solar Energy Consumption' )+
  ggtitle('Solar Energy Consumption: Clean vs. Raw')+
  theme_minimal()
```




```
autoplot(clean_wind_ts, series="Wind Clean") +
  autolayer(wind_ts, series="Wind Raw") +
  ylab("Wind Energy Consumption")+
  ggtitle('Wind Energy Consumption: Clean vs. Raw')+
  theme_minimal()
```

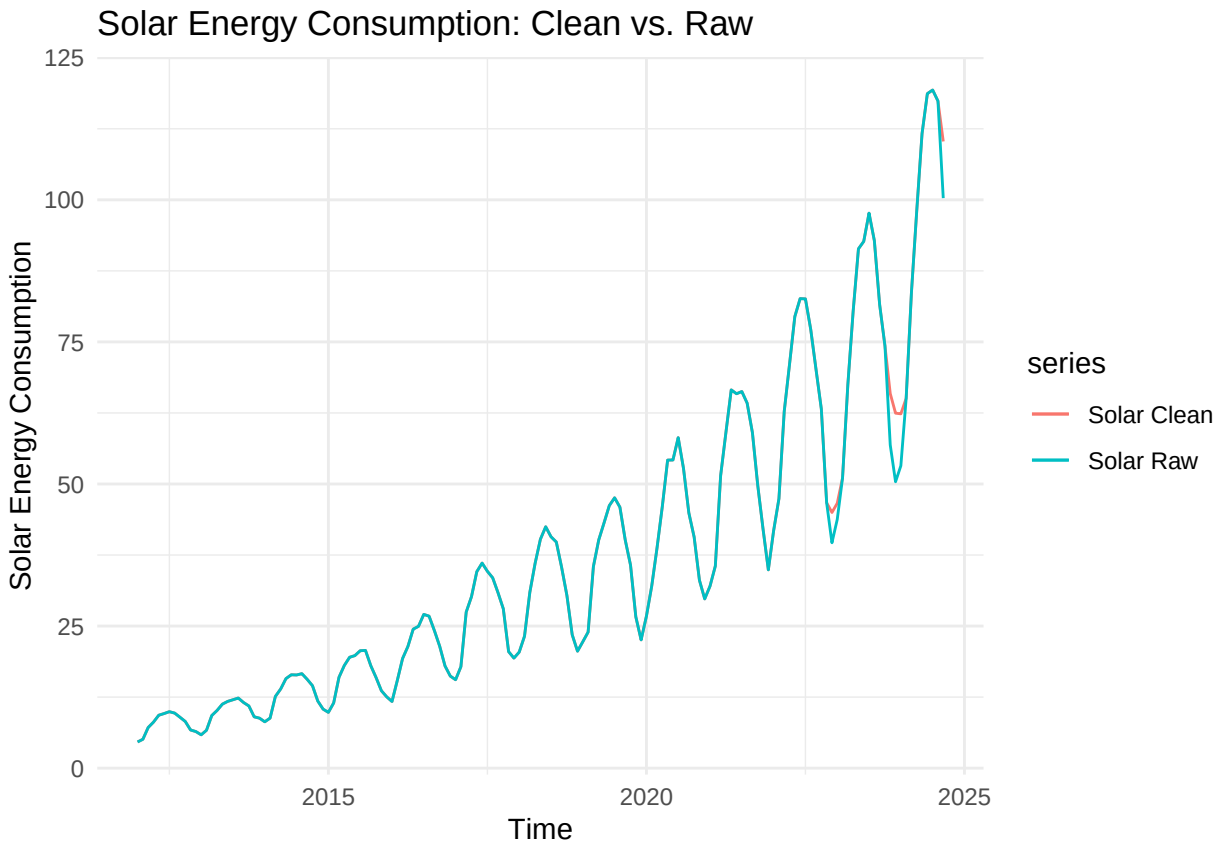


Answer: The function smoothed both series after solar/wind starts to increase at a faster pace. A great number of outliers are removed. It also removed added seasonality for early years in the 1980s. Therefore, it tends to follow the moderate trend and seasonality during 1990 to 2000s.

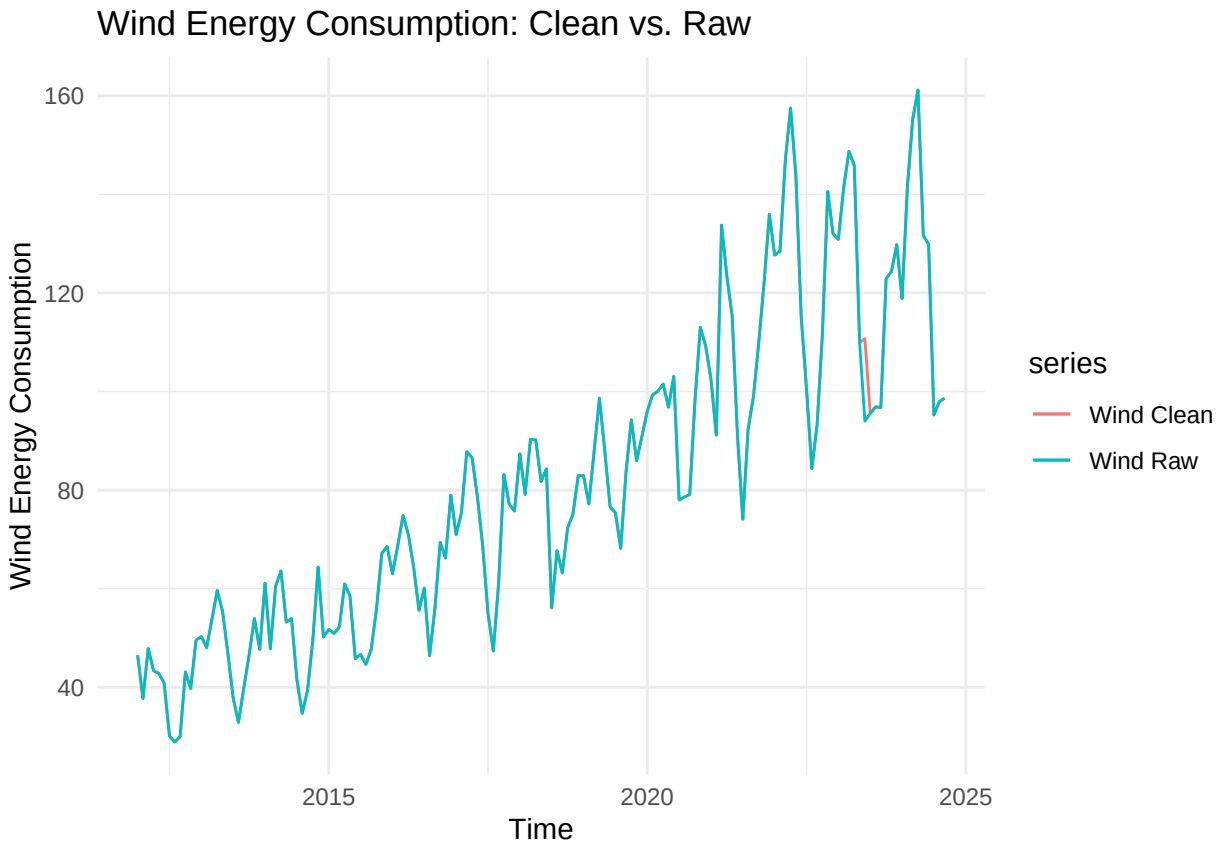
Q9

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?

```
# Apply tsclean() to both series form Q7
clean_solar_ts_2012 <- tsclean(solar_ts_2012)
clean_wind_ts_2012 <- tsclean(wind_ts_2012)

# Plot clean and original ts
autoplot(clean_solar_ts_2012, series="Solar Clean")+
  autolayer(solar_ts_2012, series="Solar Raw") +
  ylab('Solar Energy Consumption' )+
  ggtitle('Solar Energy Consumption: Clean vs. Raw')+
  theme_minimal()
```




```
autoplot(clean_wind_ts_2012, series="Wind Clean") +
  autolayer(wind_ts_2012, series="Wind Raw") +
  ylab("Wind Energy Consumption")+
  ggtitle('Wind Energy Consumption: Clean vs. Raw')+
  theme_minimal()
```



Answer: The function removed three points for solar and only one point for wind. Focusing on the years after 2012 allows the function to follow the steep trend in the recent decade rather than being influenced by early data.