ENV 797 - Time Series Analysis for Energy and Environment Applications | Spring 2026

Assignment 5 - Due date 02/17/26

Aisha Shen

Directions

You should open the .rmd file corresponding to this assignment on RStudio. The file is available on our class repository on Github.

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_Sp26.Rmd"). Then change "Student Name" on line 3 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 Canvas.

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
options(warn = -1)
library(forecast, warn.conflicts = FALSE)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tseries, warn.conflicts = FALSE)
library(ggplot2, warn.conflicts = FALSE)
library(Kendall, warn.conflicts = FALSE)
library(lubridate, warn.conflicts = FALSE)
library(tidyverse, warn.conflicts = FALSE)  #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 1.6.0
## v forcats 1.0.1     v tibble  3.3.1
## v purrr   1.2.1     v tidyr   1.3.2
## v readr   2.1.6
## -- 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(readxl, warn.conflicts = FALSE)
```

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 2025 Monthly Energy Review.

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

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

```
#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_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
colnames(energy_data) <- read_col_names
nobs <- nrow(energy_data)

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

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>, ...
```

Handling Missing Data

Q1

Using the original dataset, create a new data frame that includes only the following variables: **Date, Solar Energy Consumption and Wind Energy Consumption**. Check the class of columns, you will see that they are stored are characters instead of numbers. Because solar generation begins later in the sample, the early observations are recorded as “Not Available”. Convert the data to numeric, the “Not Available” will became NAs.

You may either filter out the “Not Available” rows and then convert the column to numeric or convert first and then remove missing values using drop_na() (or na.omit()). If you are comfortable using pipes for data wrangling, please do so.

Important: Note that we dropping the missing observations instead of interpolating is becasue they only happen in the beginning of the series!

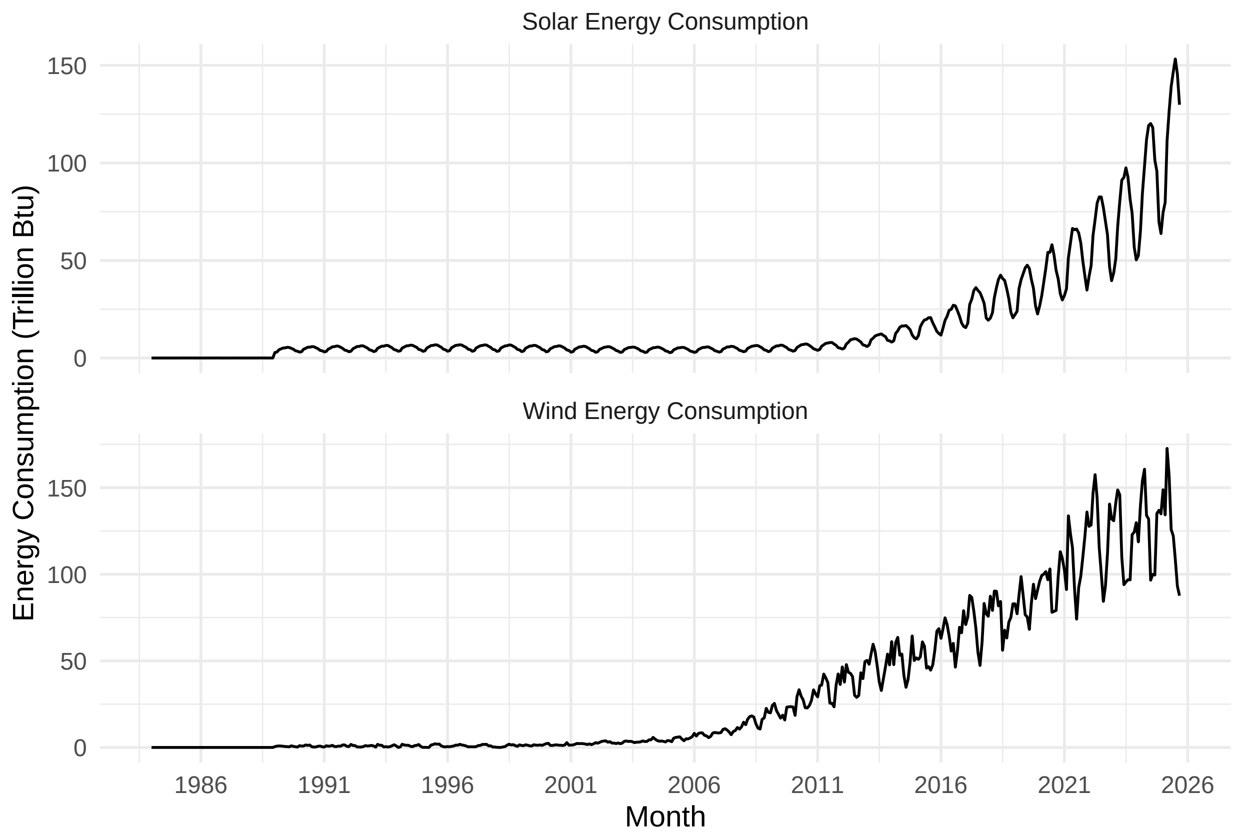
```r
library(dplyr)
timeseries_df <- energy_data %>%
  select(`Month`, `Solar Energy Consumption`, `Wind Energy Consumption`) %>%
  mutate(across(c(`Solar Energy Consumption`, `Wind Energy Consumption`), as.numeric)) %>%
  drop_na()
timeseries_df
```

```
## # A tibble: 501 x 3
##    Month               `Solar Energy Consumption` `Wind Energy Consumption`
##    <dttm>                                   <dbl>                     <dbl>
##  1 1984-01-01 00:00:00                      0                         0
##  2 1984-02-01 00:00:00                      0                         0.001
##  3 1984-03-01 00:00:00                      0.001                     0.001
##  4 1984-04-01 00:00:00                      0.001                     0.002
##  5 1984-05-01 00:00:00                      0.002                     0.003
##  6 1984-06-01 00:00:00                      0.003                     0.002
##  7 1984-07-01 00:00:00                      0.001                     0.002
##  8 1984-08-01 00:00:00                      0.003                     0.001
##  9 1984-09-01 00:00:00                      0.003                     0.002
## 10 1984-10-01 00:00:00                      0.002                     0.003
## # i 491 more rows
```

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")`

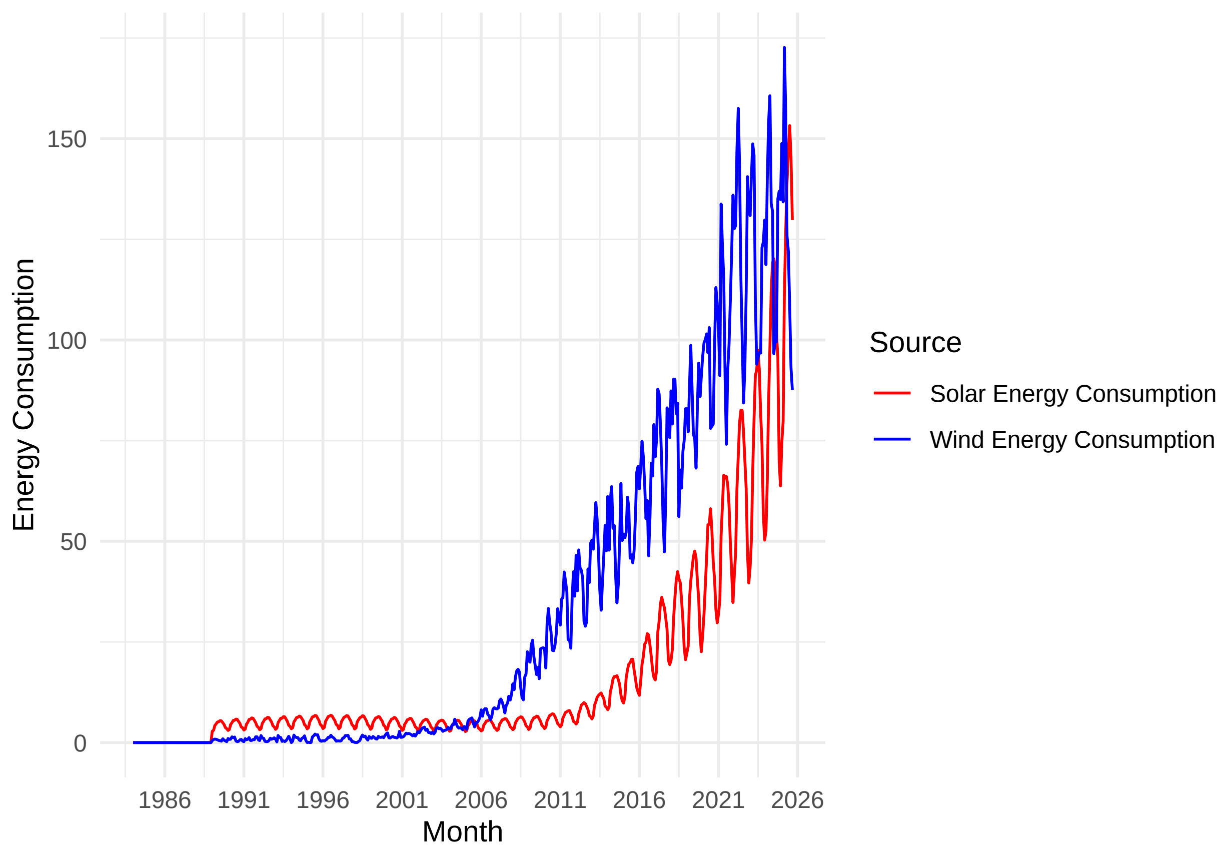
```
library(tidyverse)
library(lubridate)

timeseries_df %>%
  pivot_longer(cols = -Month, names_to = "Source", values_to = "Consumption") %>%
  ggplot(aes(x = Month, y = Consumption)) +
  geom_line() +
  facet_wrap(~ Source, scales = "free_y", ncol = 1) +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ylab("Energy Consumption (Trillion Btu)") +
  theme_minimal()
```



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
plot_data <- timeseries_df %>%
  pivot_longer(cols = -Month, names_to = "Source", values_to = "Consumption")

ggplot(plot_data, aes(x = Month, y = Consumption, color = Source)) +
  geom_line() +
  scale_color_manual(values = c("Solar Energy Consumption" = "red", "Wind Energy Consumption" = "blue")
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ylab("Energy Consumption") +
  theme_minimal()
```



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.

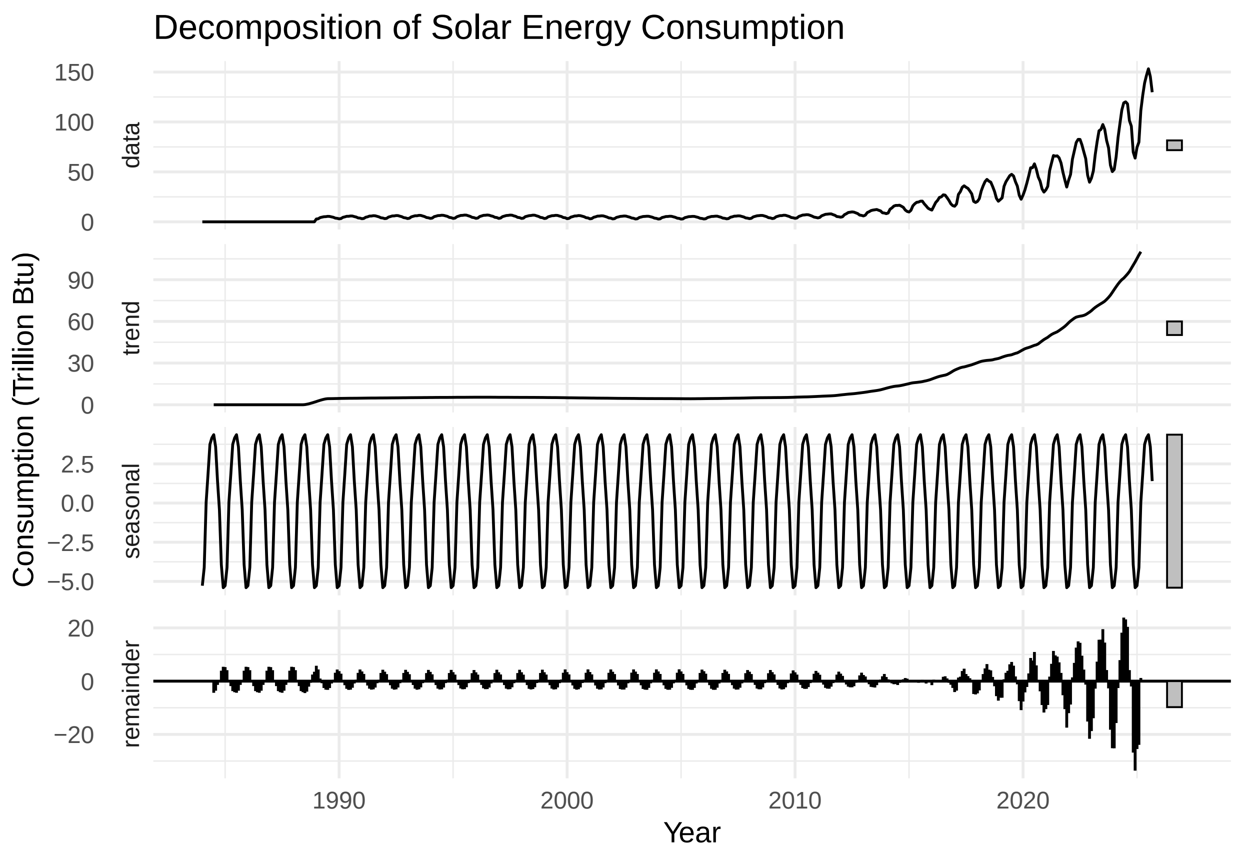
Additional info on `decompose()`.

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.

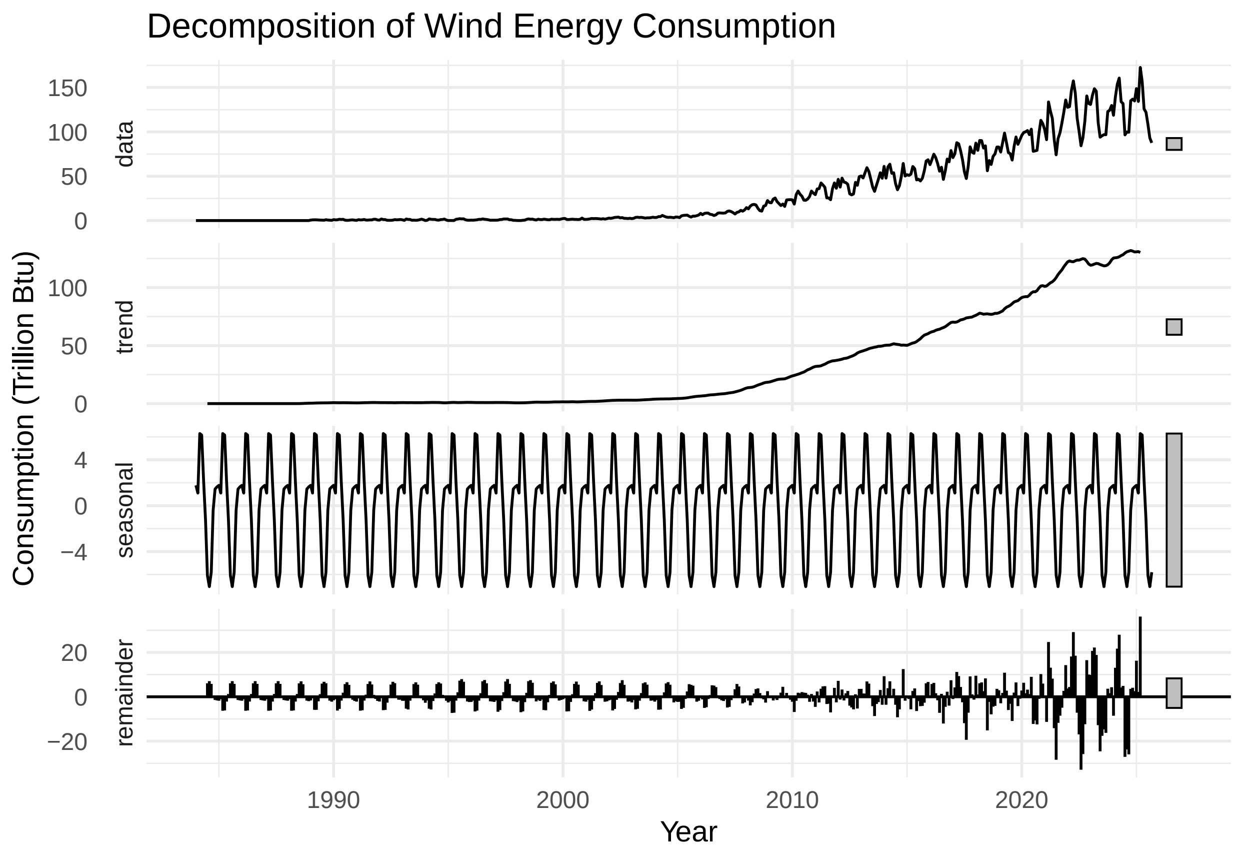
Q4

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
start_date <- c(year(min(timeseries_df$Month)), month(min(timeseries_df$Month)))

solar_ts <- ts(timeseries_df$`Solar Energy Consumption`, start = start_date, frequency = 12)
wind_ts <- ts(timeseries_df$`Wind Energy Consumption`, start = start_date, frequency = 12)
solar_decomp <- decompose(solar_ts, type = "additive")
wind_decomp <- decompose(wind_ts, type = "additive")
autoplot(solar_decomp) +
  labs(title = "Decomposition of Solar Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



```r
autoplot(wind_decomp) +
  labs(title = "Decomposition of Wind Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```

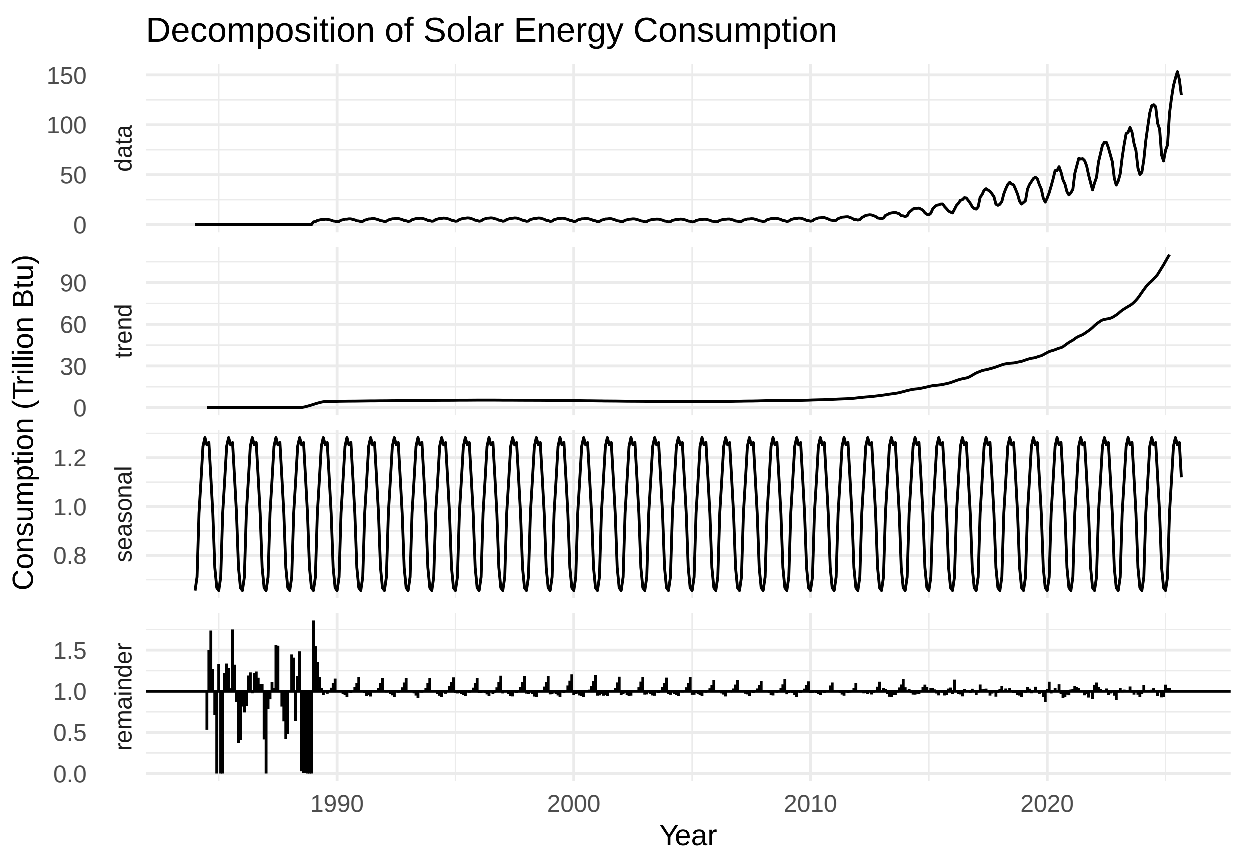


Answer: The trend component for both wind and solar appears to show explosive growth, with a flat line up until the mid-late 2000's and then a steep increase after that. In both settings the random component appears to show a range-bound behaviour initially having values bounded in a small range with this range increasing continually after 2010. It does not truly appear to be random and looks like it has some periodicity and shows some seasonality. Finally, the seasonal component clearly shows the annual periodic patterns.

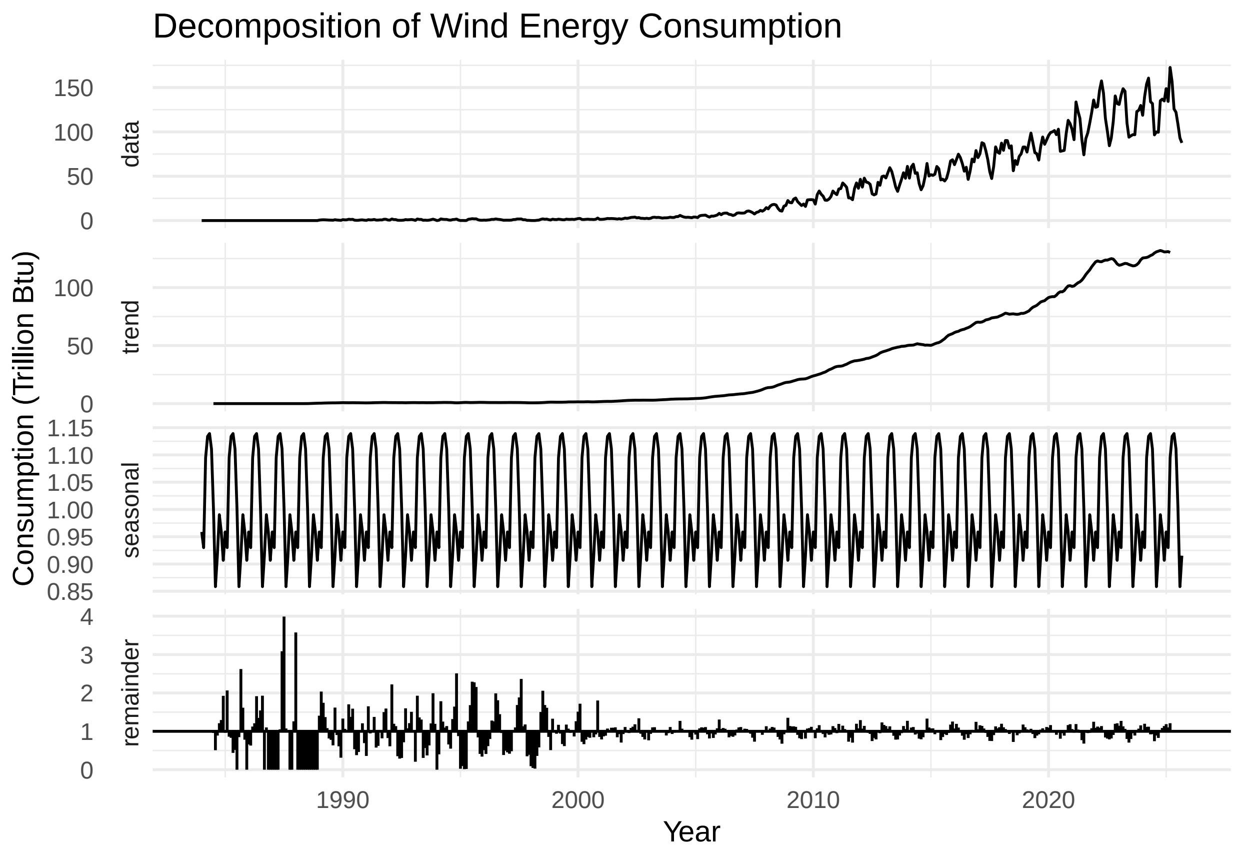
Q5

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_decomp_mult <- decompose(solar_ts, type = "multiplicative")
wind_decomp_mult <- decompose(wind_ts, type = "multiplicative")
autoplot(solar_decomp_mult) +
  labs(title = "Decomposition of Solar Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



```r
autoplot(wind_decomp_mult) +
  labs(title = "Decomposition of Wind Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



Answer: The trend component for both wind and solar appears to still show explosive growth, with a flat line up until the late 2000's and then a steep increase after that. In both settings, now the random component appears to show noisy behaviour initially. However, it doesn not appear to be truly random and has some periodicity to it. In later years, post mid 2000, the random component becomes much smaller but retains the periodicity. Finally, the seasonal component clearly shows the annual periodic patterns.

Q6

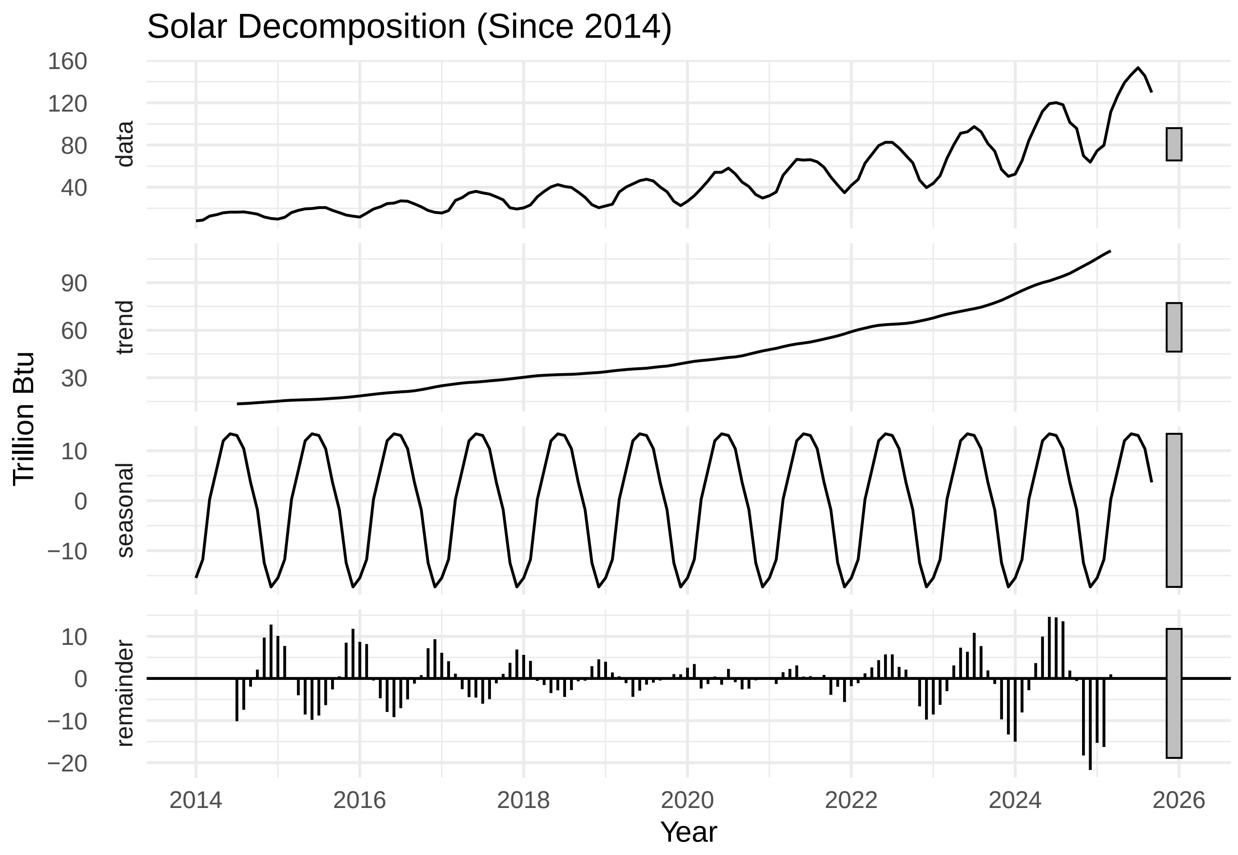
When fitting a model to this data, do you think you need all the historical data? Think about the data from 80s, 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

Answer:I dont think all the historic data is useful as the initial flat line period has no predictive power over the early 2020's. Since the trendline seen above is based on a moving average, ideally we should be able to take a sliding window over the last year or two years to predict the trend for forecasting, and add seasonal variance based on recent data.

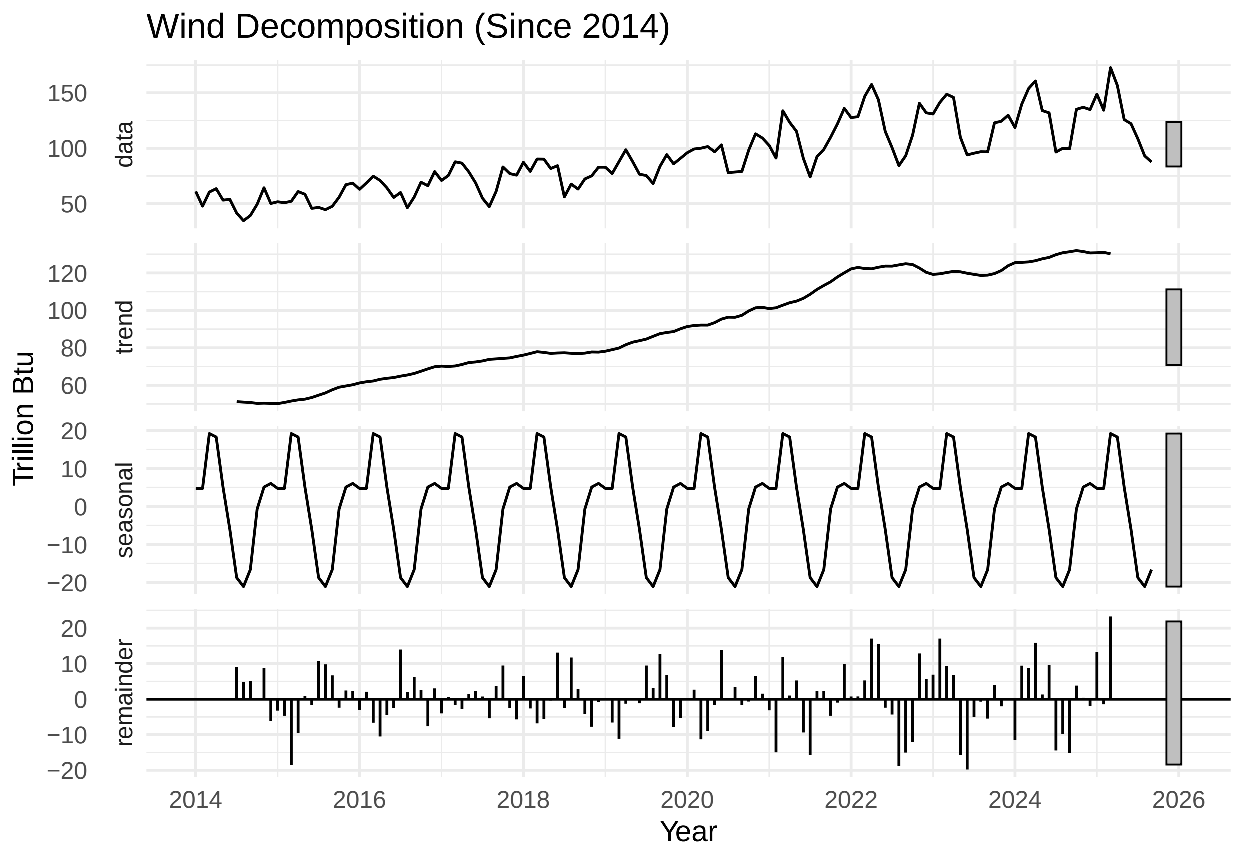
Q7

Create a new time series object where historical data starts on January 2014. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2014 )`. Apply the decompose function `type=additive` to this new time series. Comment on the results. Does the random component look random?

```r
timeseries_df_recent<-timeseries_df %>%
  filter(year(Month) >= 2014)

solar_ts_2014 <- ts(timeseries_df_recent$`Solar Energy Consumption`,
                    start = c(2014, 1), frequency = 12)
wind_ts_2014 <- ts(timeseries_df_recent$`Wind Energy Consumption`,
                   start = c(2014, 1), frequency = 12)

solar_decomp_2014 <- decompose(solar_ts_2014, type = "additive")
wind_decomp_2014 <- decompose(wind_ts_2014, type = "additive")

autoplot(solar_decomp_2014) +
  labs(title = "Solar Decomposition (Since 2014)", x = "Year", y = "Trillion Btu") +
  theme_minimal()
```



```r
autoplot(wind_decomp_2014) +
  labs(title = "Wind Decomposition (Since 2014)", x = "Year", y = "Trillion Btu") +
  theme_minimal()
```



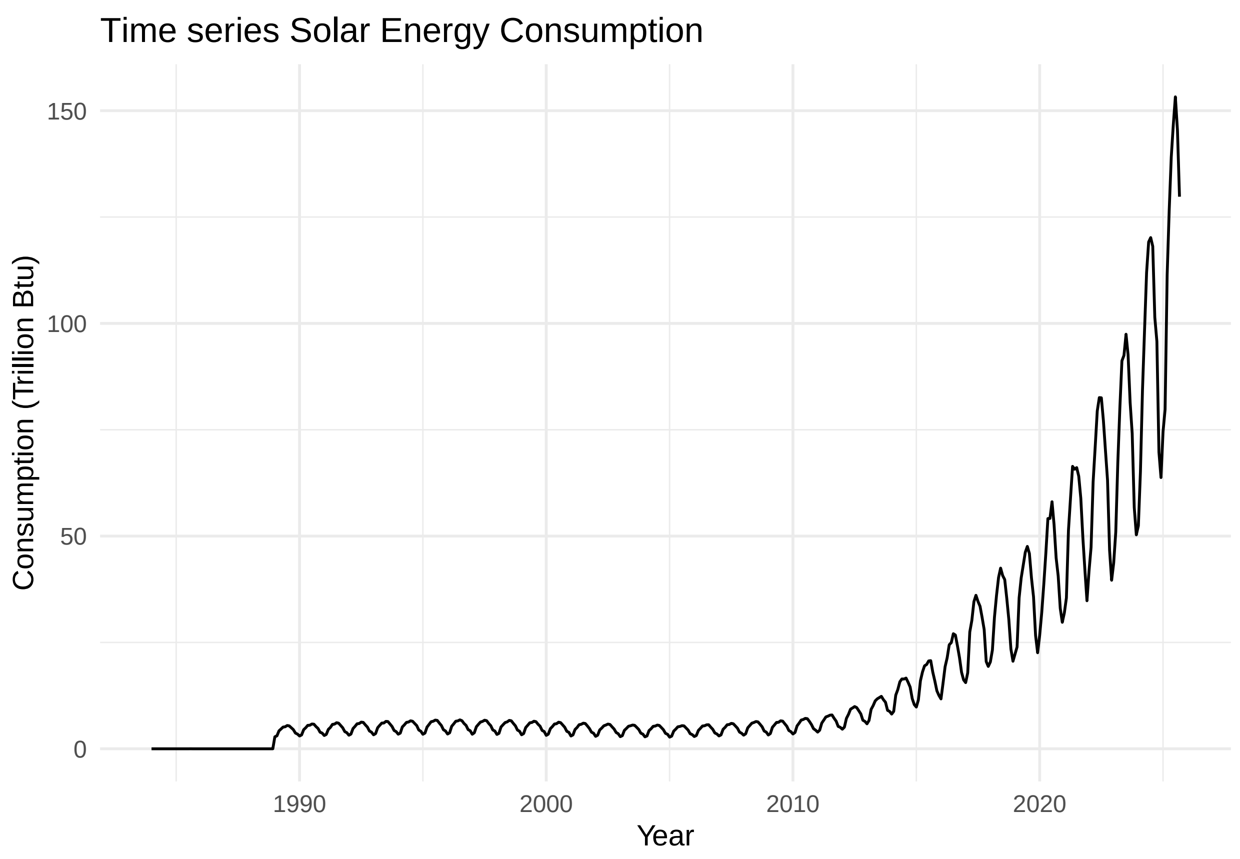
Answer: For Solar data, the random component does not look random and shows a clear periodic trend. Initially, between 2014 and 2020, the periodicity is captured by oscillatory behaviour with decreasing amplitude. However, 2020 onwards the amplitude increases. For wind data, the random component looks more random though it still shows signs of some periodicity.

Identify and Remove outliers

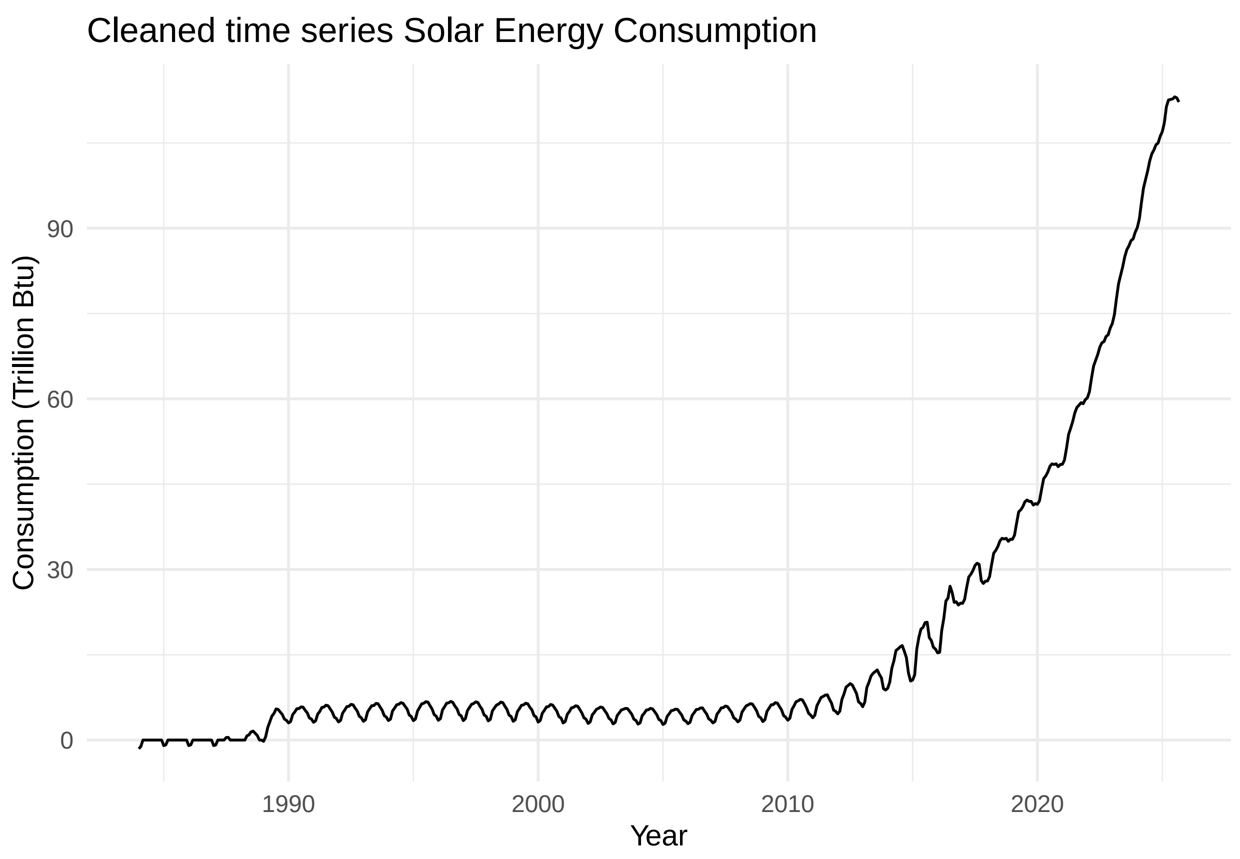
Q8

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.

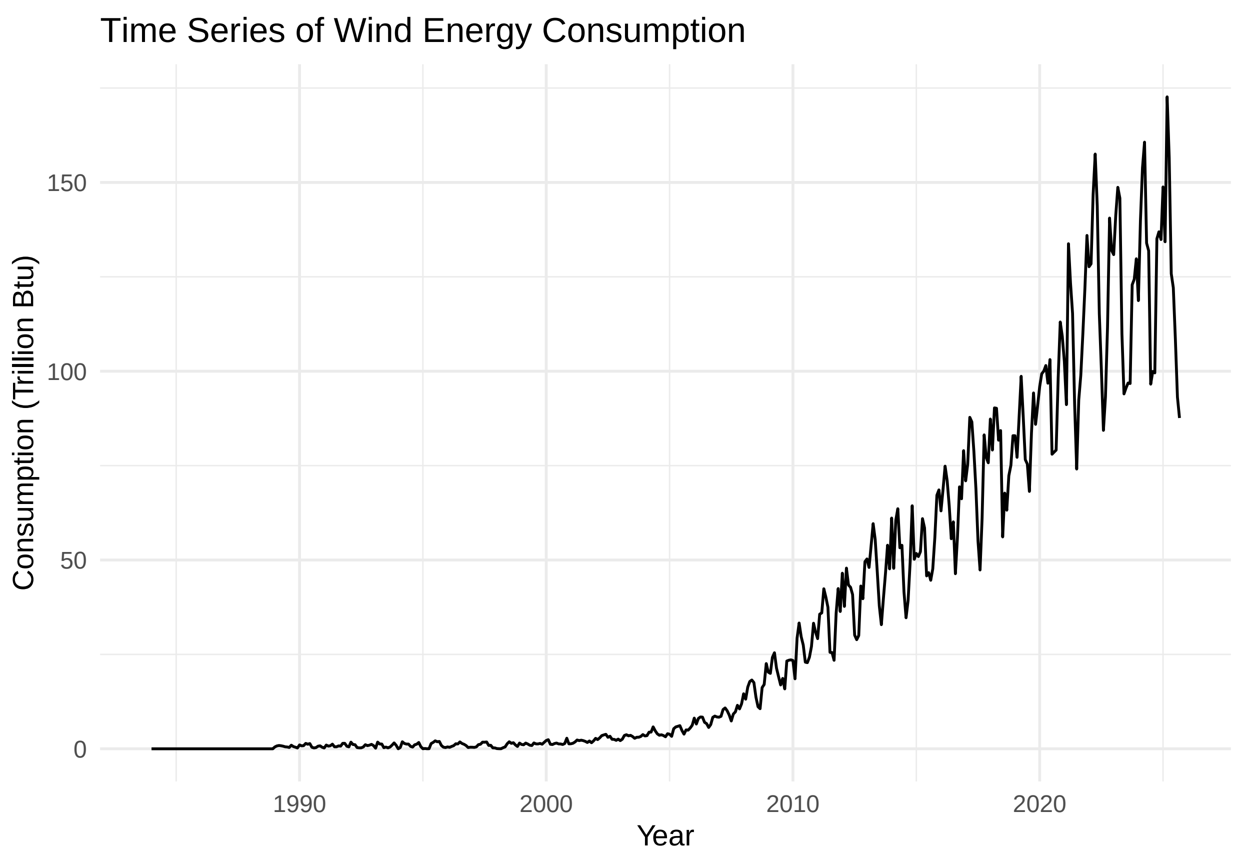
```
clean_solar_ts<-tsclean(solar_ts)
clean_wind_ts<-tsclean(wind_ts)

autoplot(solar_ts) +
  labs(title = "Time series Solar Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



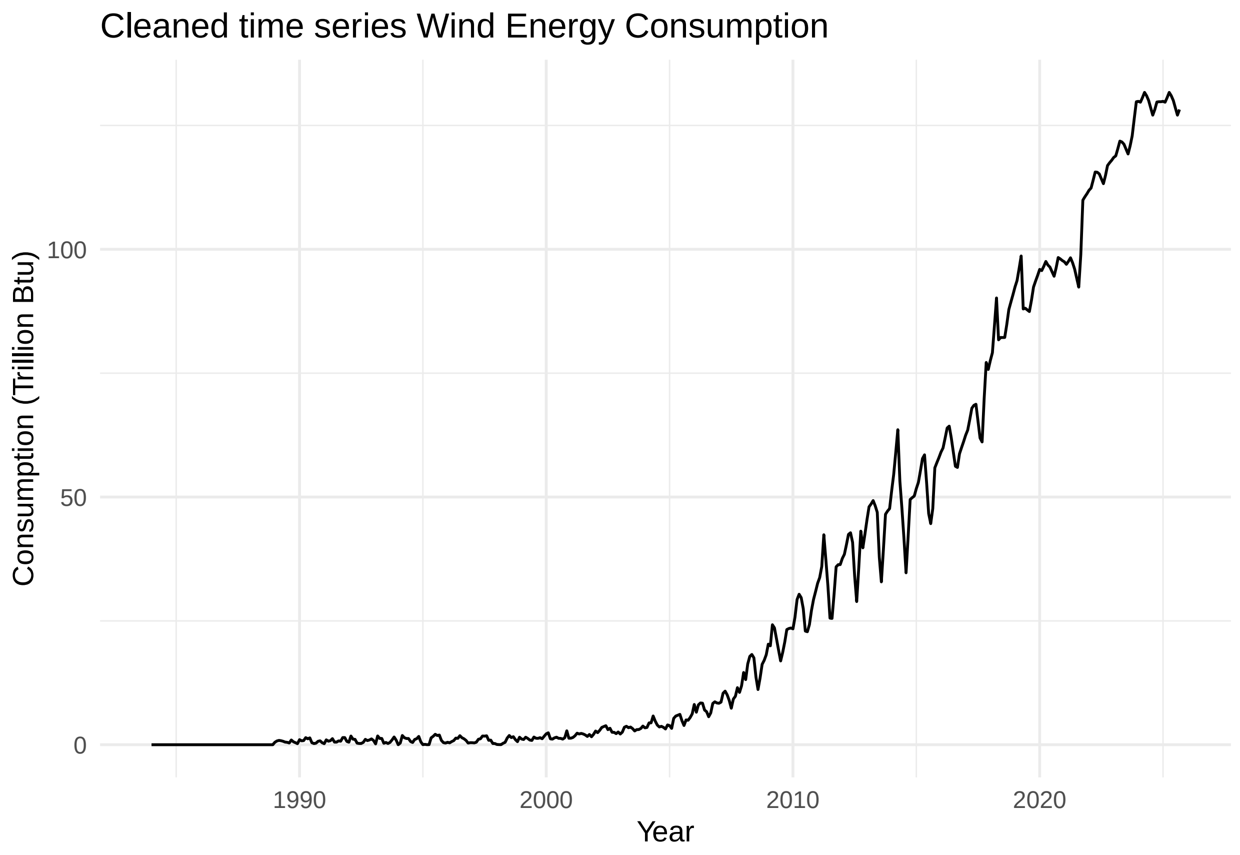
```r
autoplot(clean_solar_ts) +
  labs(title = "Cleaned time series Solar Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



```r
autoplot(wind_ts) +
  labs(title = "Time Series of Wind Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



```r
autoplot(clean_wind_ts) +
  labs(title = "Cleaned time series Wind Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```

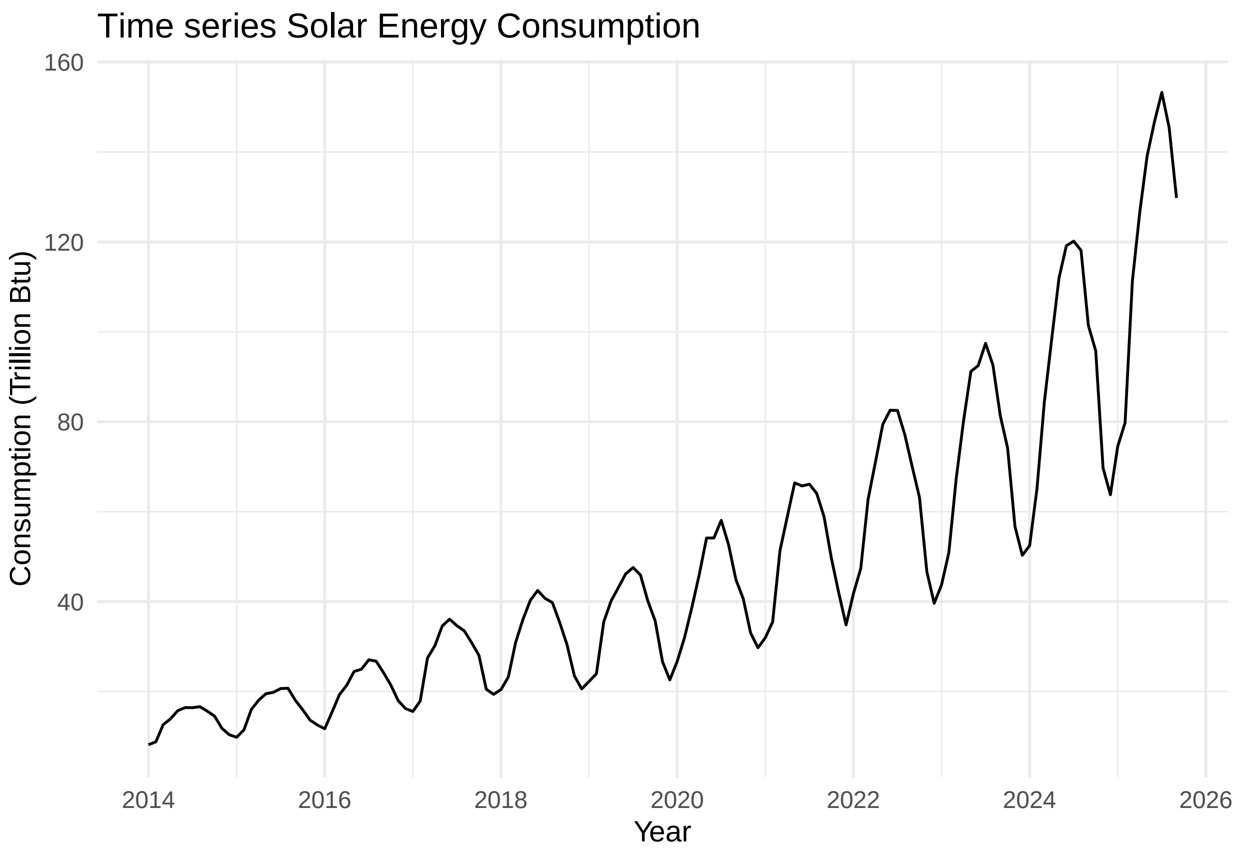


> Answer: The function removes a lot of outliers for both solar and wind datasets. Based on looking at the documentation for the tsclean() function and the explanation here (https://robjhyndman.com/hyndsight/tsoutliers/), the outlier detection process for tsclean constructs some robust distribution to remove statistical outliers. Since this is the entire dataset, the strength of seasonality is not as strongly captured as the super smoothed trend is much stronger across longer time spans.

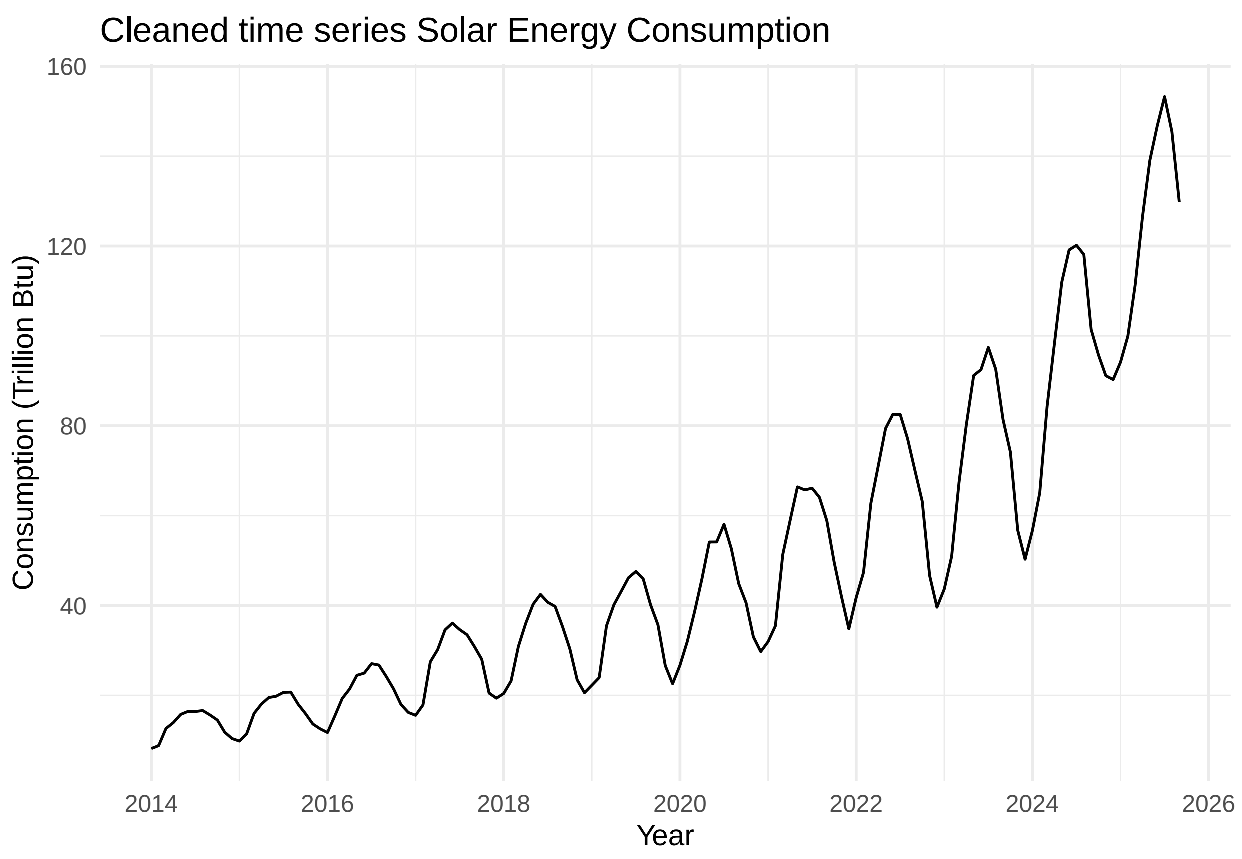
Q9

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?

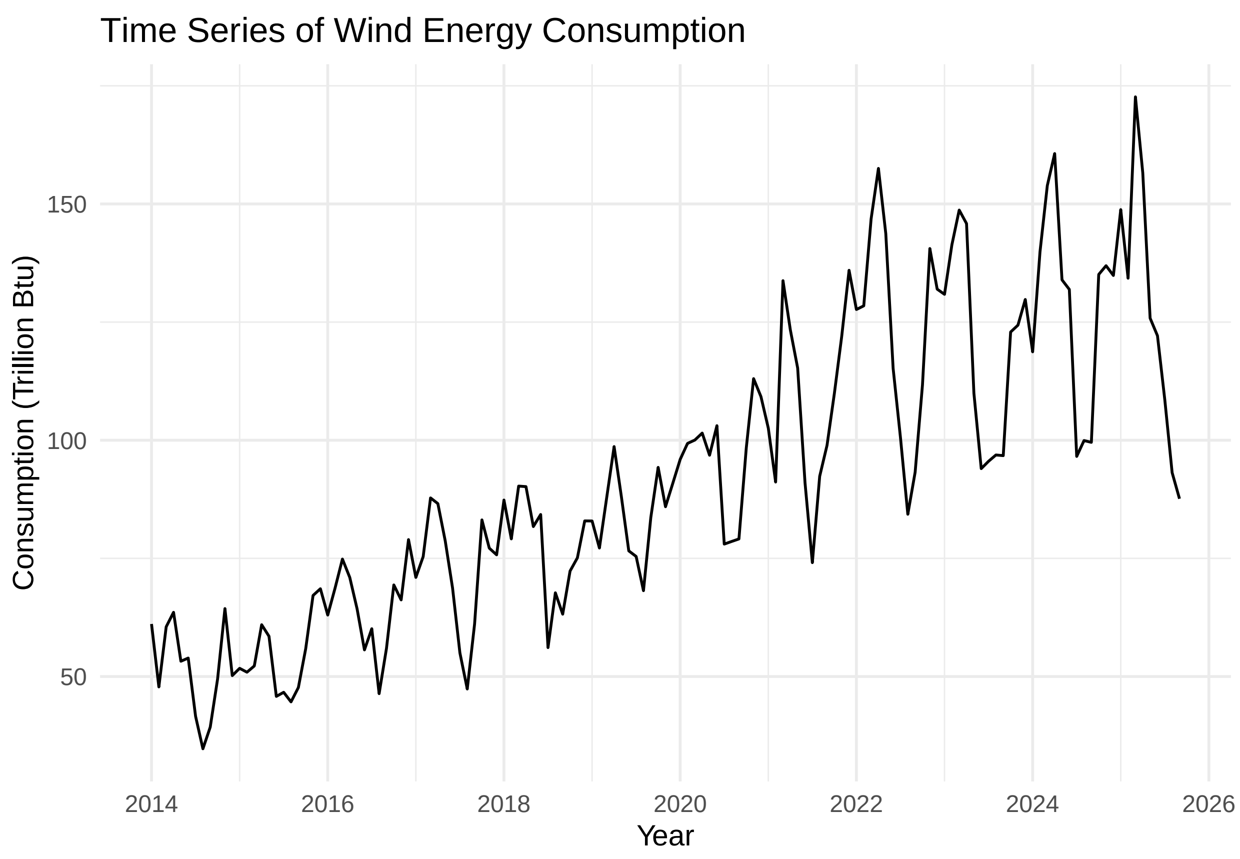
```r
clean_solar_ts_2014<-tsclean(solar_ts_2014)
clean_wind_ts_2014<-tsclean(wind_ts_2014)

autoplot(solar_ts_2014) +
  labs(title = "Time series Solar Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



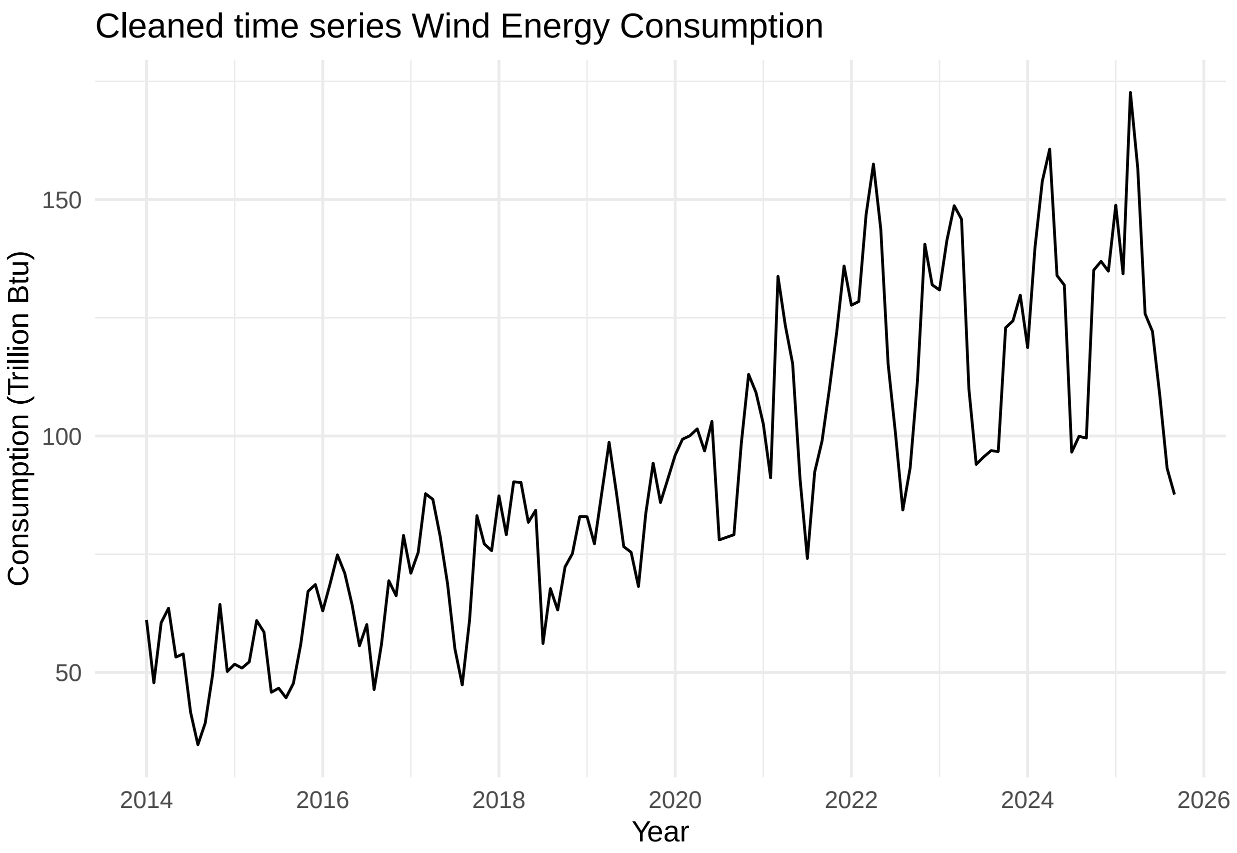
```r
autoplot(clean_solar_ts_2014) +
  labs(title = "Cleaned time series Solar Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



```
autoplot(wind_ts_2014) +
  labs(title = "Time Series of Wind Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



```
autoplot(clean_wind_ts_2014) +
  labs(title = "Cleaned time series Wind Energy Consumption",
       x = "Year",
       y = "Consumption (Trillion Btu)") +
  theme_minimal()
```



Answer: For the solar energy consumption data, some outliers were removed in 2025 as the consumption levels in the original data showed a steeper decline than should be expected. For the wind energy consumption data, no outliers were removed, suggesting that this new series might be better suited to any forecasting we want to do for the next few months. We can see that these series remain largely unchanged as for this period the trend component is a simple function (as opposed to over the longer horizon before) and seasonality is strongly captured.