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

Assignment 5 - Due date 02/18/25

Student Name

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(readxl)
theme_set(theme_classic())
library(cowplot)
```

```
##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:lubridate':
##
##     stamp
```

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
data <- read_excel("Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx",
                   skip = 12, col_names = FALSE)
```

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

```r
names <- read_excel("Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx",
                    skip = 10, n_max = 1, col_names = F)
```

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

```r
colnames(data) <- names
head(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>, ...
```

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

Q1

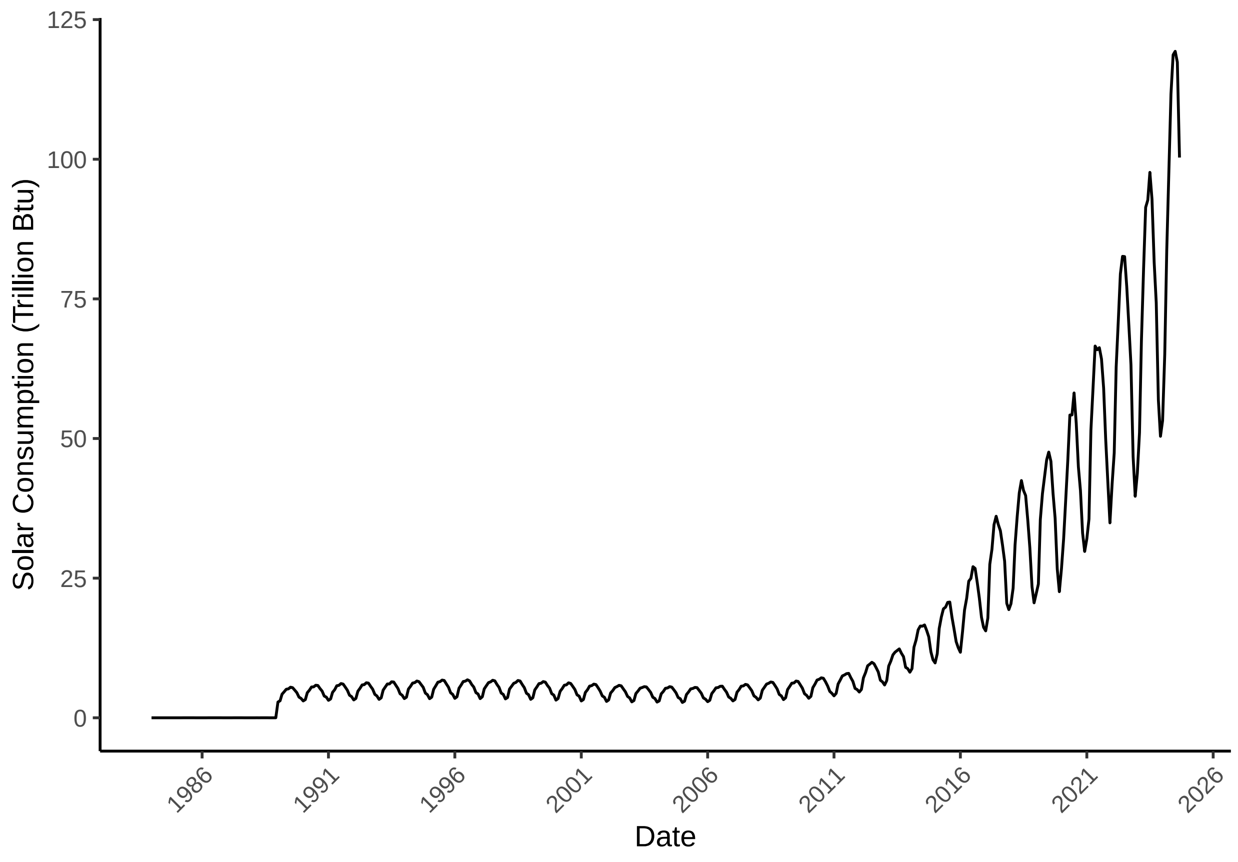
For this assignment you will work only with the following columns: Solar Energy Consumption and Wind Energy Consumption. Create a data frame structure with these two time series only and the Date column. Drop the rows with *Not Available* and convert the columns to numeric. You can use filtering to eliminate 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
data_filtered <- data %>%
  select(Month, `Solar Energy Consumption`, `Wind Energy Consumption`) %>%
  filter(`Solar Energy Consumption` != 'Not Available') %>%
  rename("Date" = 1, "SolarConsumption" = 2, "WindConsumption" = 3) %>%
  mutate_at(c("SolarConsumption", "WindConsumption"), as.numeric) %>%
  mutate_at("Date", ymd)
```

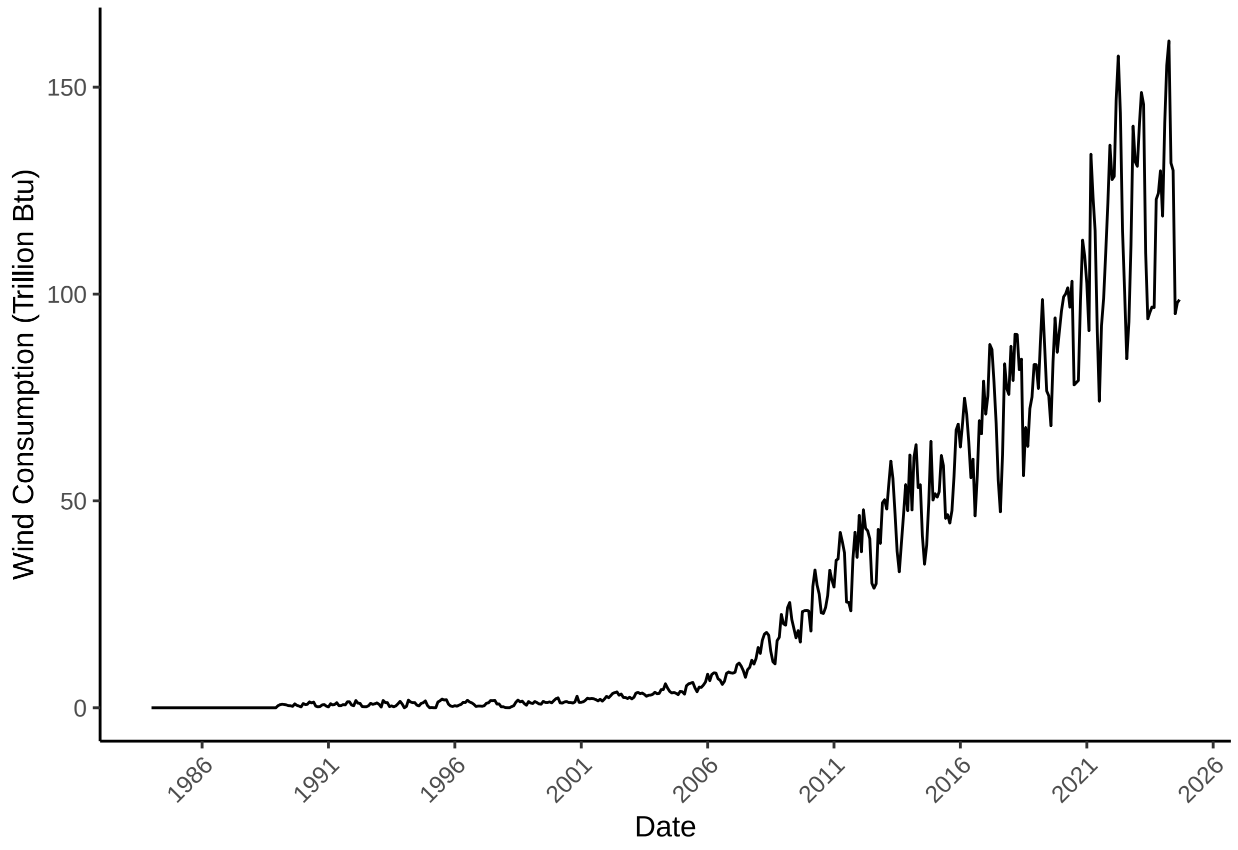
Q2

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
ggplot() +
  geom_line(data = data_filtered, aes(x = Date, y = SolarConsumption)) +
  ylab("Solar Consumption (Trillion Btu)") +
  scale_x_date(date_breaks = "5 year", date_labels = "%Y", guide = guide_axis(angle = 45)) +
  theme_classic()
```



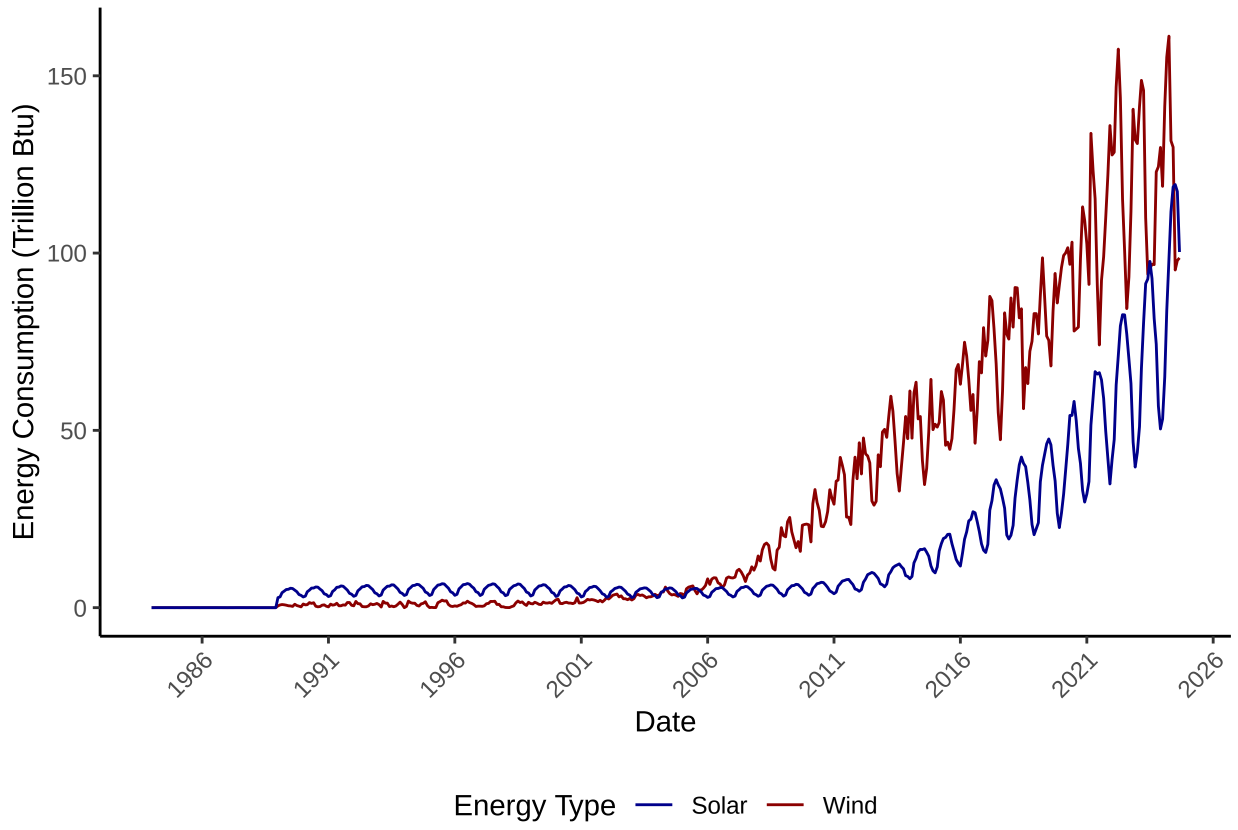
```r
ggplot() +
  geom_line(data = data_filtered, aes(x = Date, y = WindConsumption)) +
  ylab("Wind Consumption (Trillion Btu)") +
  scale_x_date(date_breaks = "5 year", date_labels = "%Y", guide = guide_axis(angle = 45)) +
  theme_classic()
```



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() +
  geom_line(data = data_filtered, aes(x = Date, y = WindConsumption, color = 'Wind') ) +
  geom_line(data = data_filtered, aes(x = Date, y = SolarConsumption, color = 'Solar')) +
  ylab("Energy Consumption (Trillion Btu)") +
  scale_x_date(date_breaks = "5 year", date_labels = "%Y", guide = guide_axis(angle = 45)) +
  scale_color_manual(name = "Energy Type", values = c("darkblue", "darkred")) +
  theme(legend.position = 'bottom')
```



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()`.

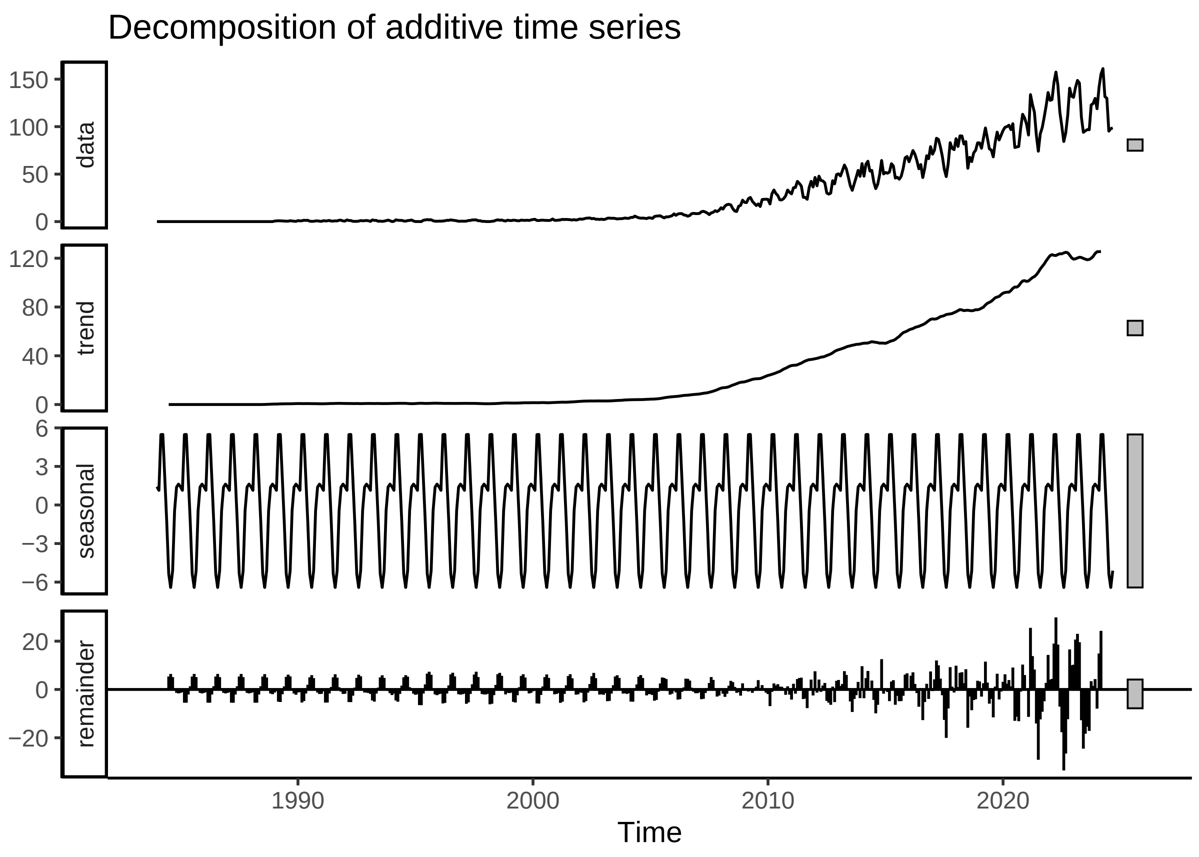
1) You have two options: alternative and multiplicative. Multiplicative models exhibit a change in frequency over time.
2) The trend is not a straight line because it uses a moving average method to detect trend.
3) The seasonal component of the time series is found by subtracting the trend component 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

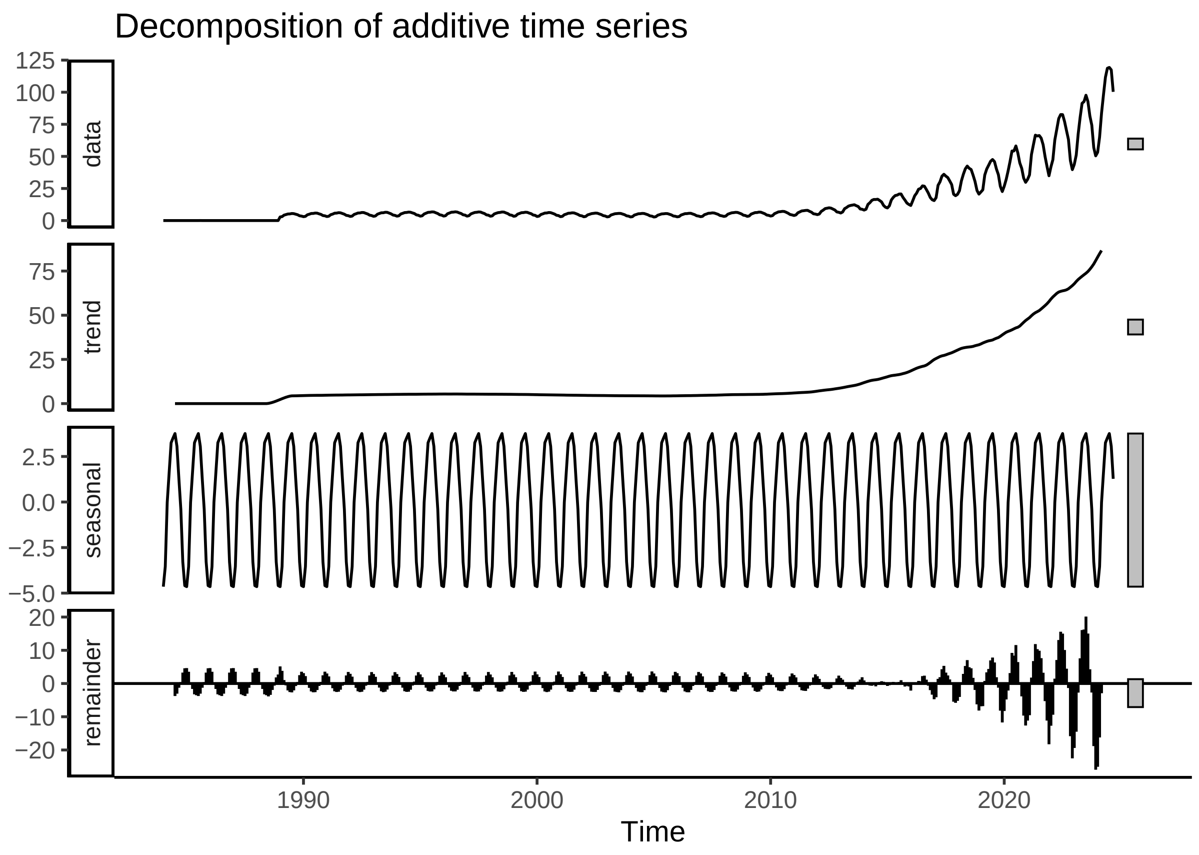
Transform wind and solar series into a time series object and apply the decompose function on them using the additive option, i.e., `decompose(ts_data, type = "additive")`. What can you say about the trend component? What about the random component? Does the random component look random? Or does it appear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

```r
ts_wind <- ts(data_filtered$WindConsumption, start = c('1984','1','1'), frequency = 12)
ts_solar <- ts(data_filtered$SolarConsumption, start = c('1984','1','1'), frequency = 12)

ts_wind_decompose <- decompose(ts_wind, type = "additive")
ts_solar_decompose <- decompose(ts_solar, type = "additive")
autoplot(ts_wind_decompose)
```



```r
autoplot(ts_solar_decompose)
```

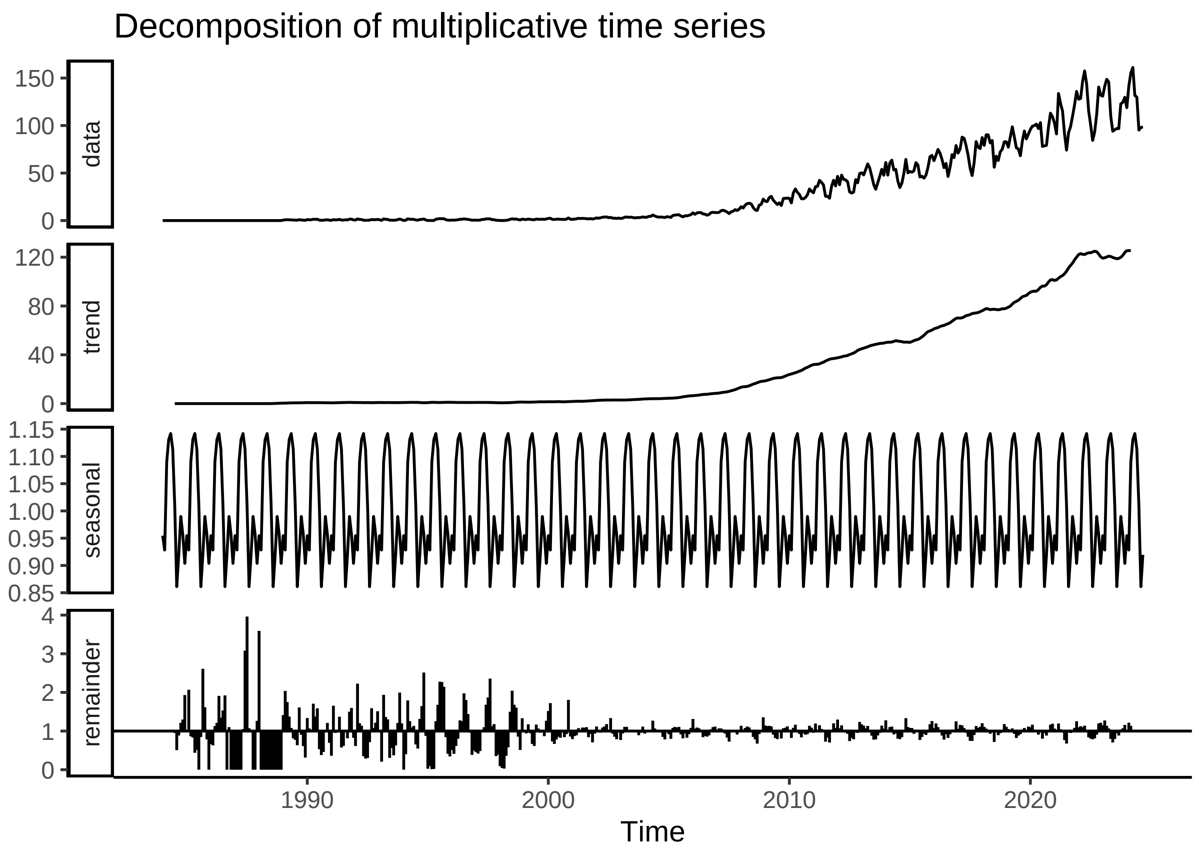


Wind: The trend component is small until the late 2000s, after which it turns into an increasing trend. There appears to be a significant seasonal component. The random component does not look random, and appears to have both seasonality and trend. Solar: The trend component is small until the mid 2010s, and increases after that. There is apparent seasonality, which along with trend is also found in the random component. The reason for these effects still appearing in the random component is likely because there was a level shift of wind and solar consumption.

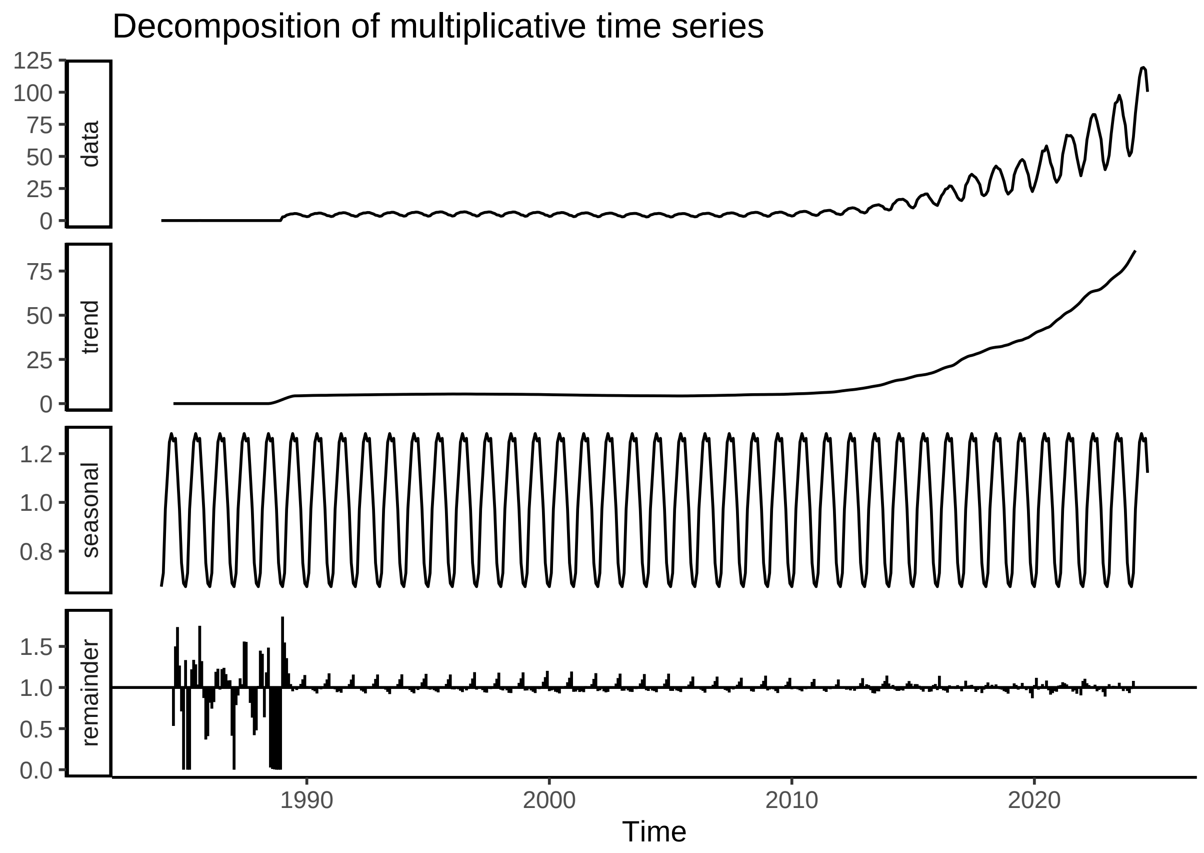
Q5

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
ts_wind_decompose <- decompose(ts_wind, type = "multiplicative")
ts_solar_decompose <- decompose(ts_solar, type = "multiplicative")
autoplot(ts_wind_decompose)
```



```r
autoplot(ts_solar_decompose)
```



The random component for wind and solar appears much more 'random', with smaller seasonal influences. The beginning of the time series has large random swings, which is most likely from the small amount of renewable that can not be described by the general trend or seasonal component.

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.

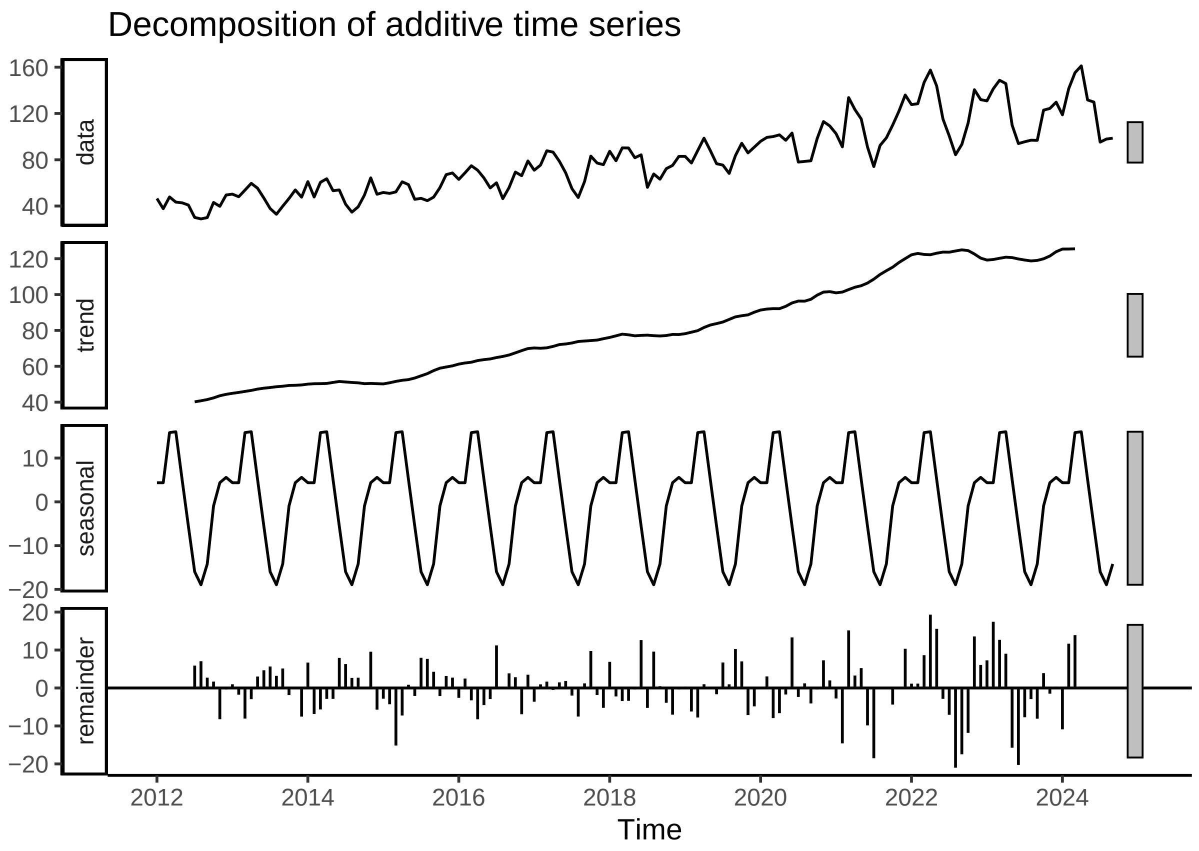
> Answer: We do not need the historical data before the widespread adaption of wind and solar. This data is not indicitave of trends in solar and wind consumption now, since there has been a level shift in the consumption of wind and solar.

Q7

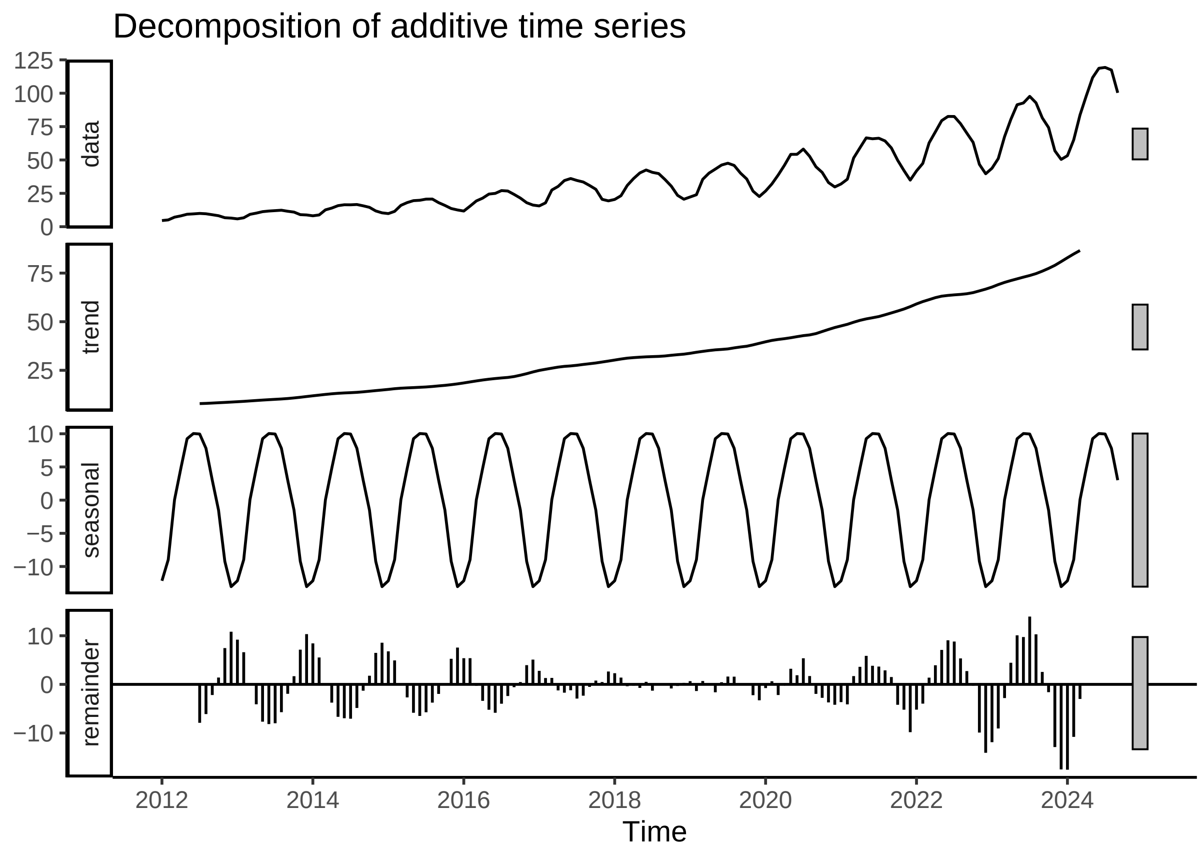
Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the decompose function `type=additive` to this new time series. Comment the results. Does the random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

```r
data_2012 <- filter(data_filtered, year(Date) >= 2012)
ts_wind_2012 <- ts(data_2012$WindConsumption, start = c("2012", "01", "01"), frequency = 12)
ts_solar_2012 <- ts(data_2012$SolarConsumption, start = c("2012", "01", "01"), frequency = 12)
```

```r
ts_wind_decompose_2012 <- decompose(ts_wind_2012, type = "additive")
ts_solar_decompose_2012 <- decompose(ts_solar_2012, type = "additive")
autoplot(ts_wind_decompose_2012)
```



```r
autoplot(ts_solar_decompose_2012)
```



Answer: The random components from wind look more random, but still increaes as the trend component grows larger. The random components from solar look like they have somme relation to seasonal compoonents, with a strange section in the middle that looks to behand differently from the rest of the data.

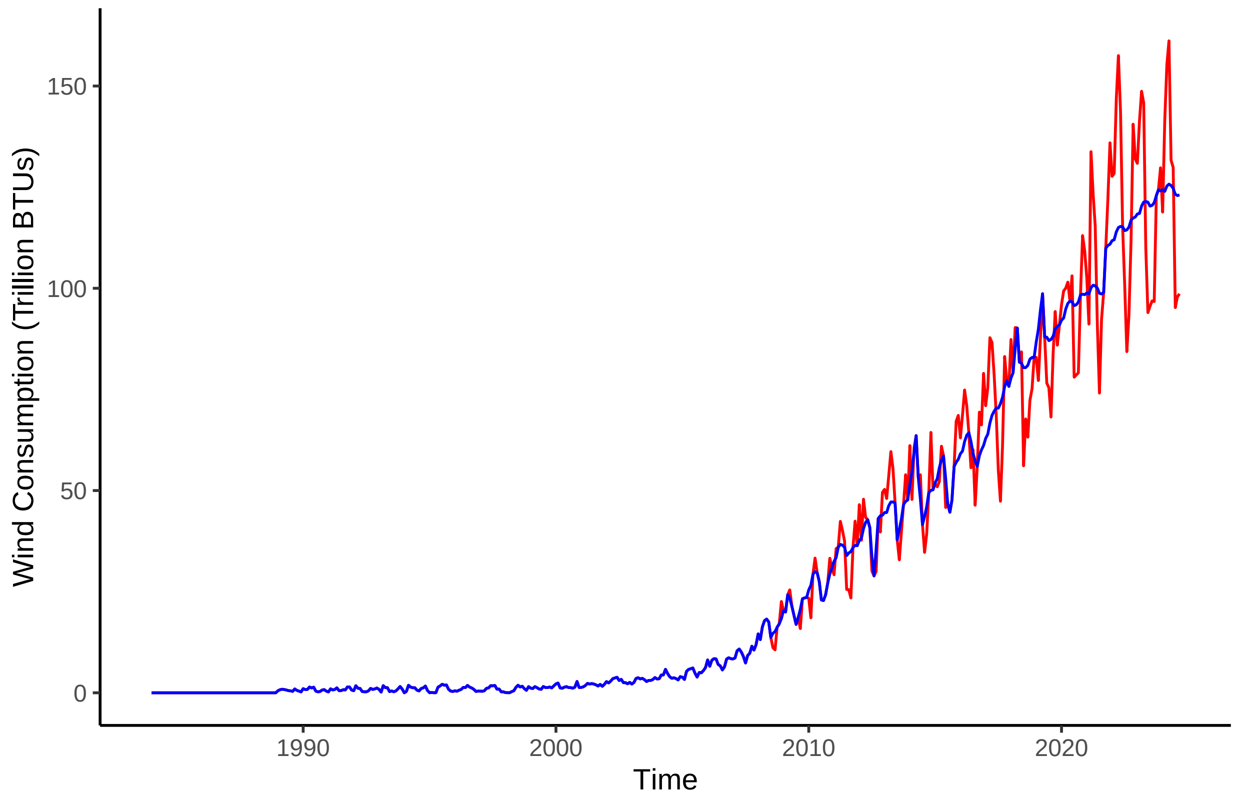
Identify and Remove outliers

Q8

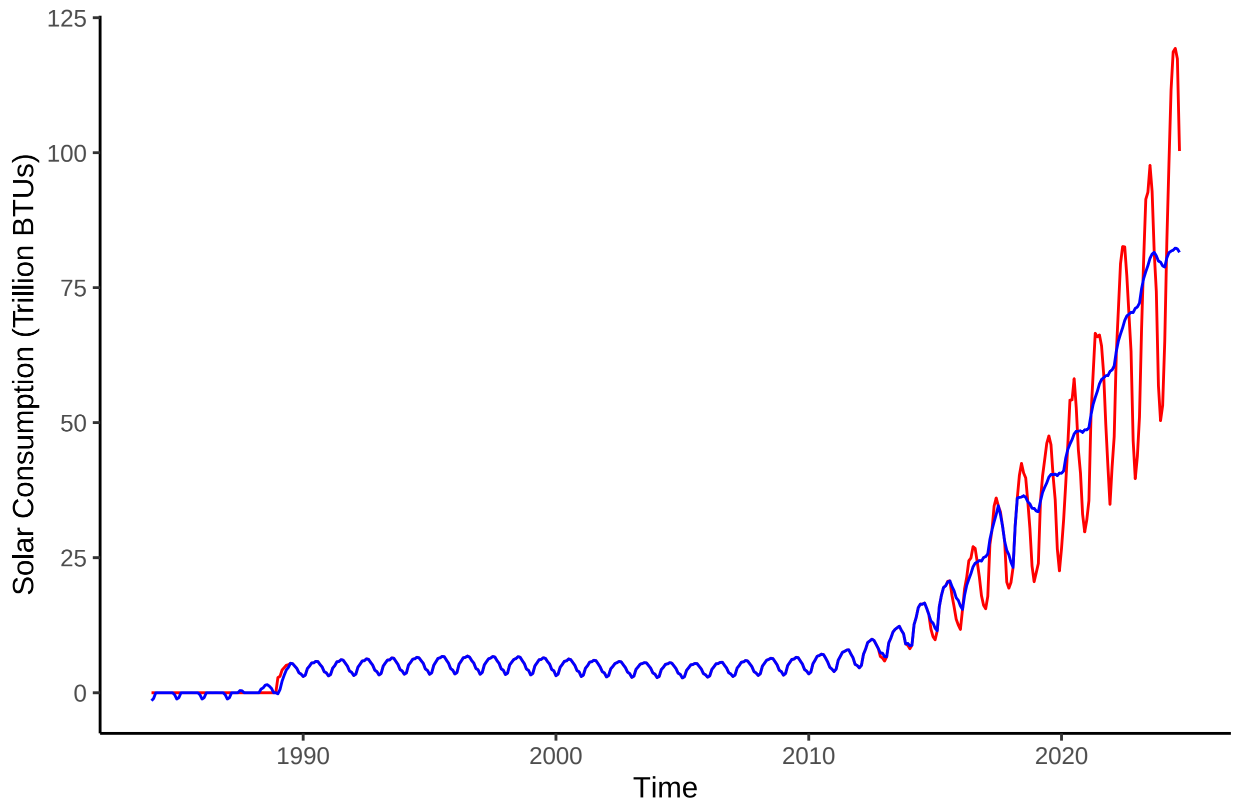
Apply the `tsclean()` to both series from Q7. Did the function removed any outliers from the series? Hint: Use `autoplot()` to check if there is difference between cleaned series and original series.

```r
ts_wind_clean <- tsclean(ts_wind)
ts_solar_clean <- tsclean(ts_solar)

autoplot(ts_wind, color = 'red') +
  autolayer(ts_wind_clean, color = 'blue')  +
  ylab('Wind Consumption (Trillion BTUs)')
```



```r
autoplot(ts_solar, color = 'red') +
  autolayer(ts_solar_clean, color = 'blue')  +
  ylab('Solar Consumption (Trillion BTUs)')
```



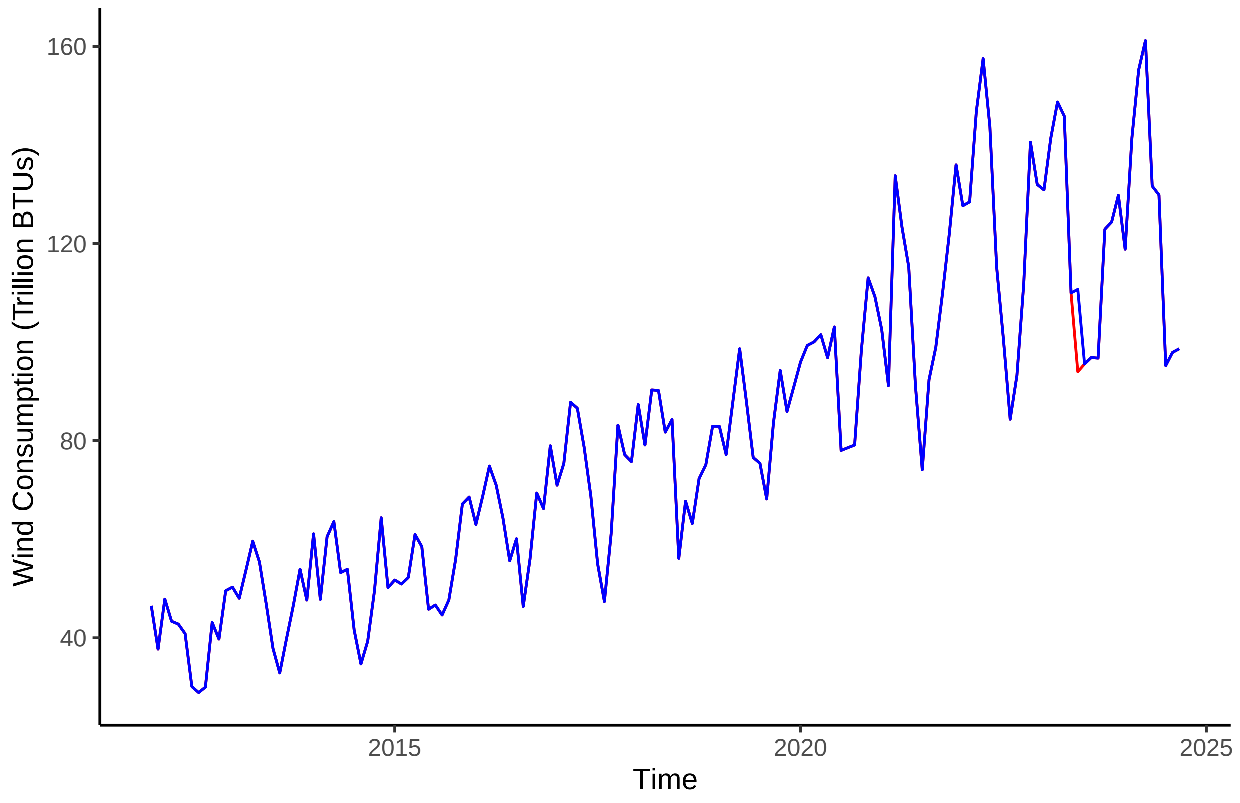
tsclean did remove outliers, but not from the beginning of the time series. Instead, it removed outliers from the end of the time series, when solar and wind consumption are increasing at faster rates.

Q9

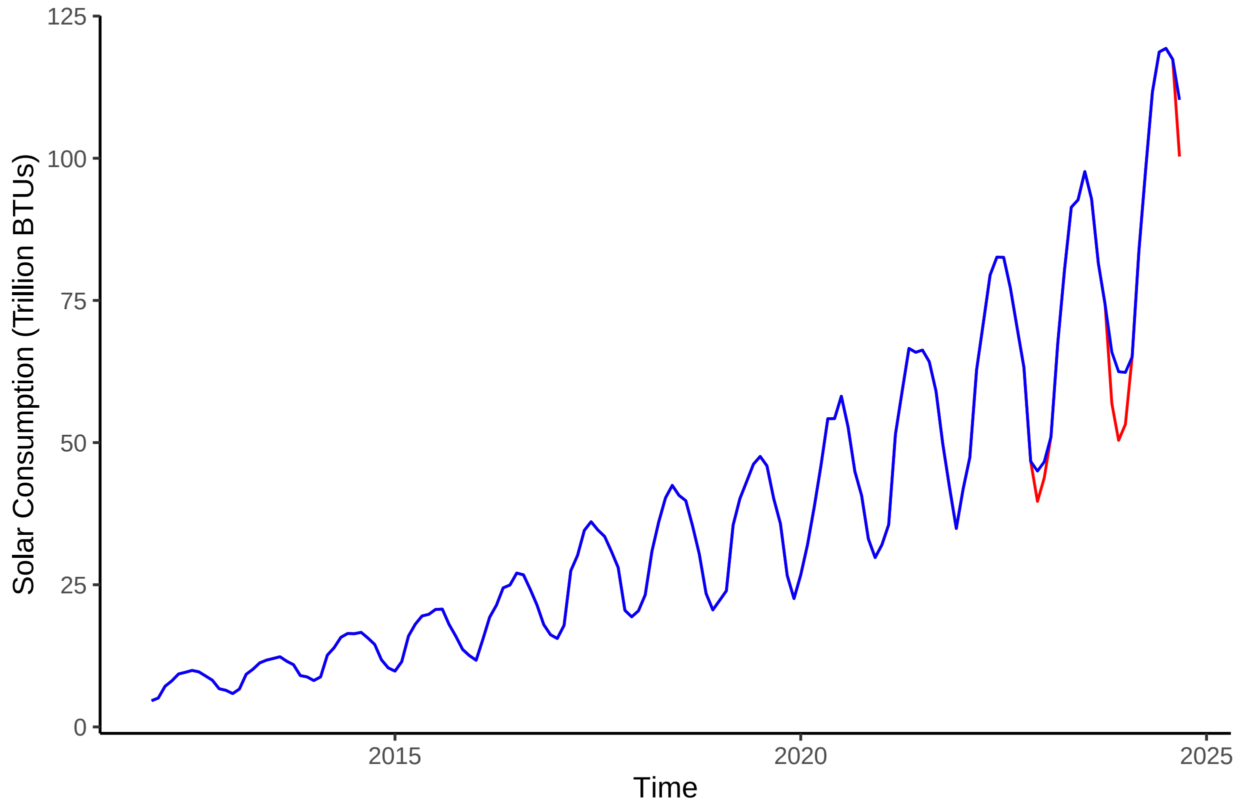
Redo number Q8 but now with the time series you created on Q7, i.e., the series starting in 2014. Using what `autoplot()` again what happened now?Did the function removed any outliers from the series?

```
ts_wind_2012_clean <- tsclean(ts_wind_2012)
ts_solar_2012_clean <- tsclean(ts_solar_2012)

autoplot(ts_wind_2012, color = 'red') +
  autolayer(ts_wind_2012_clean, color = 'blue')  +
  ylab('Wind Consumption (Trillion BTUs)')
```



```r
autoplot(ts_solar_2012, color = 'red') +
  autolayer(ts_solar_2012_clean, color = 'blue')  +
  ylab('Solar Consumption (Trillion BTUs)')
```



Answer: One outlier was removed from the wind time series, and three were removed from the solar time series. Interestingly, the three removed from the solar time series were at the end of the last three years.