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

Assignment 5 - Due date 02/18/25

Alex Lopez

## 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)
```

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

```
library(tseries)
library(ggplot2)
library(Kendall)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
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 errors
```

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

## 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.

```r
#importing data set
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`
```

```r
#extract column names
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`
```

```
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)

#set theme
theme_set(
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 12),
          axis.title.x = element_text(size = 10),
          axis.title.y = element_text(size = 10, angle = 90, vjust = 0.5))
  )
```

### 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!

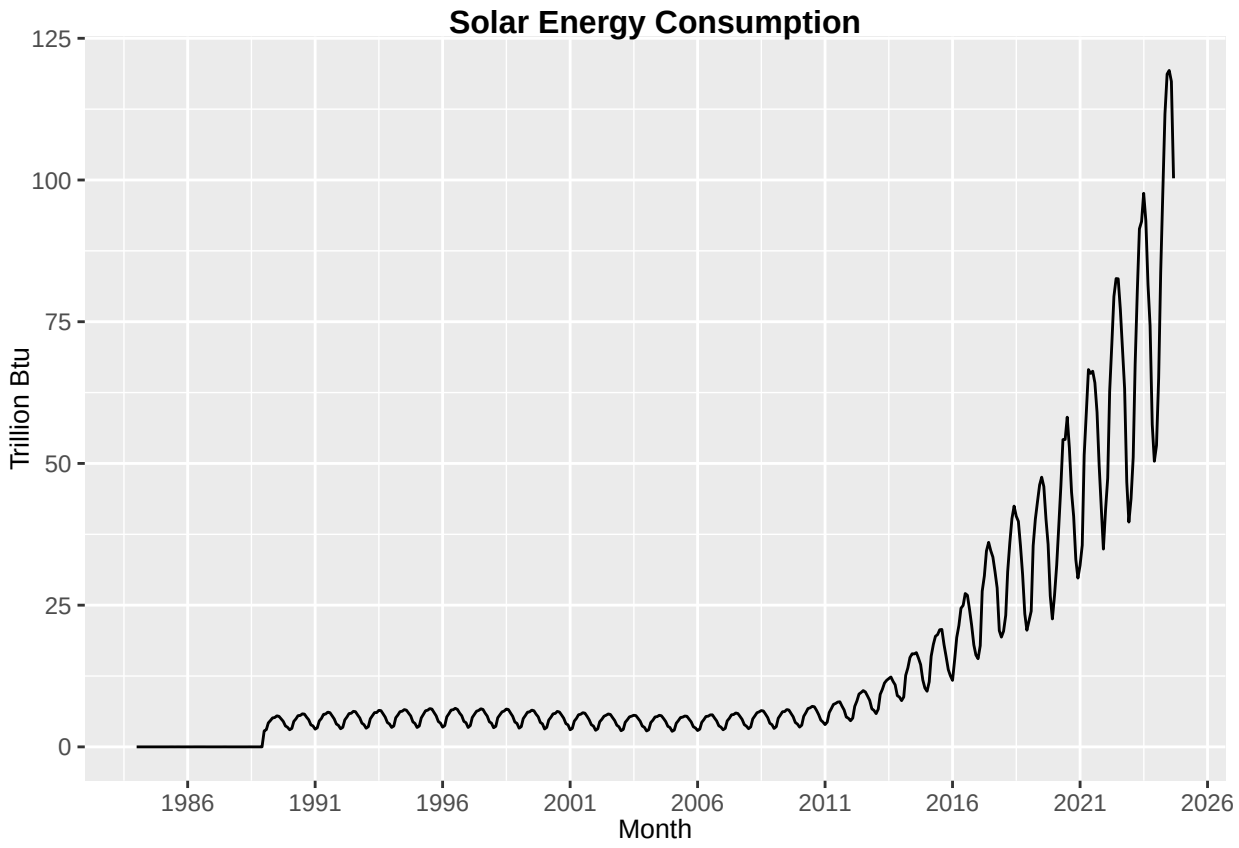
```
#select columns
solarwind_consumption_df <- energy_data %>%
  select(1,8,9) %>%
  mutate(
    `Solar Energy Consumption` = as.numeric(
      ifelse(`Solar Energy Consumption` == "Not Available", NA,
             `Solar Energy Consumption`)),
    `Wind Energy Consumption` = as.numeric(
      ifelse(`Wind Energy Consumption` == "Not Available", NA,
             `Wind Energy Consumption`))) %>%
  drop_na()

#convert Month column to Date type
solarwind_consumption_df$Month <- as.Date(solarwind_consumption_df$Month)

head(solarwind_consumption_df)
```

```
## # A tibble: 6 x 3
##   Month      `Solar Energy Consumption` `Wind Energy Consumption`
##   <date>                          <dbl>                     <dbl>
## 1 1984-01-01                      0                         0
## 2 1984-02-01                      0                         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
```

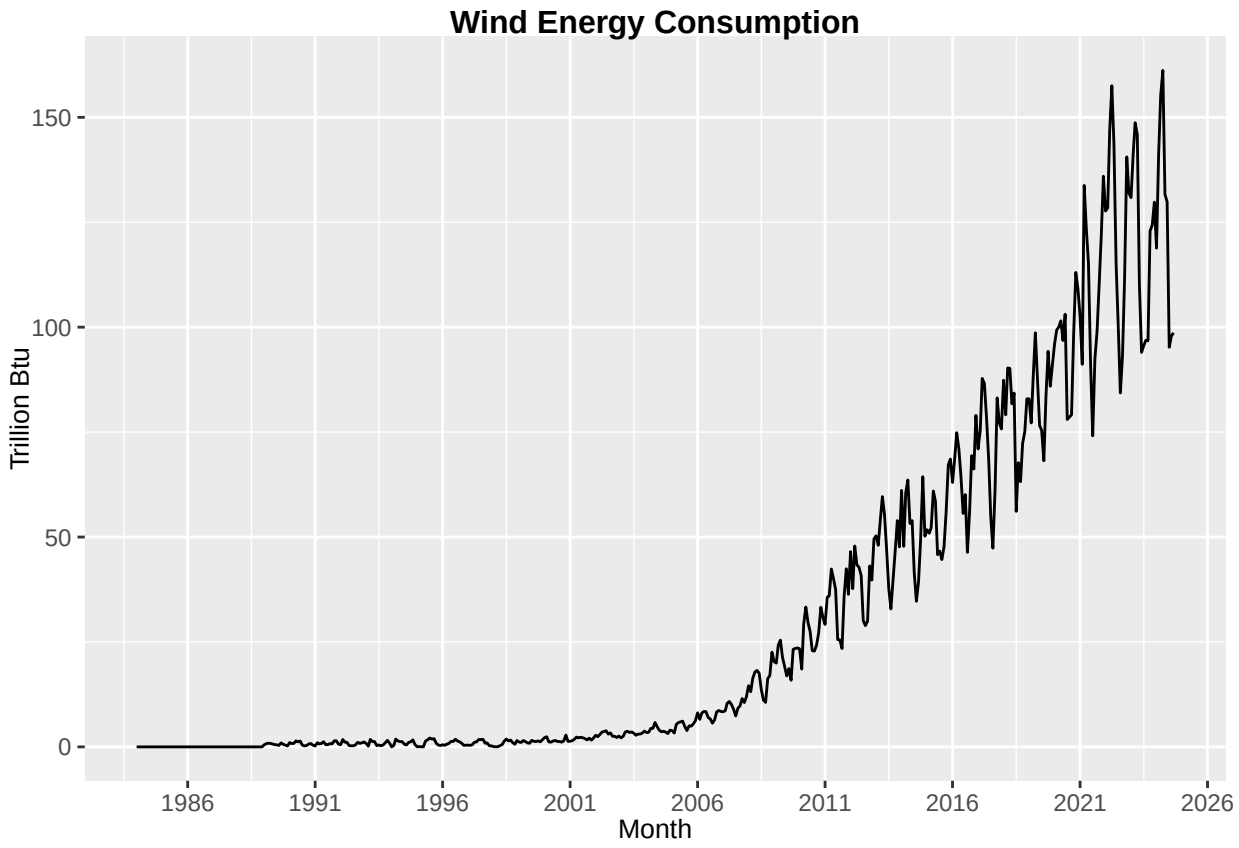
### Q2

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 Energy Consumption graph
ggplot(solarwind_consumption_df, aes(x = Month, y = `Solar Energy Consumption`)) +
  geom_line() +
  labs(y = "Trillion Btu", title = "Solar Energy Consumption") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```



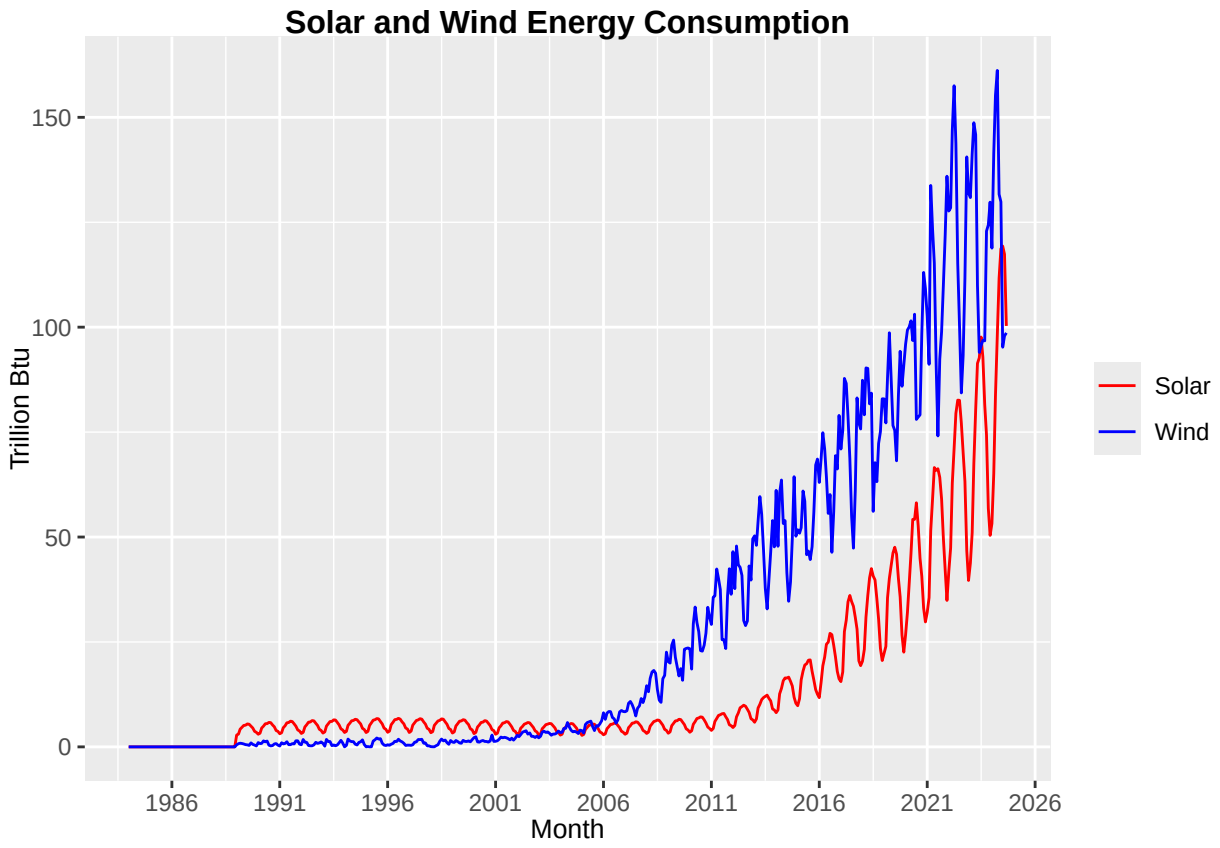

```
#Wind Energy Consumption graph
ggplot(solarwind_consumption_df, aes(x = Month, y = `Wind Energy Consumption`)) +
  geom_line() +
  labs(y = "Trillion Btu", title = "Wind Energy Consumption") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```




**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(solarwind_consumption_df) +
  geom_line(aes(x = Month, y = `Solar Energy Consumption`,
                color = "Solar")) +
  geom_line(aes(x = Month, y = `Wind Energy Consumption`,
                color = "Wind")) +
  labs(y = "Trillion Btu",
       title = "Solar and Wind Energy Consumption") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  scale_color_manual(values = c("Solar" = "red", "Wind" = "blue")) +
  guides(color = guide_legend(title = NULL))
```




## 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.

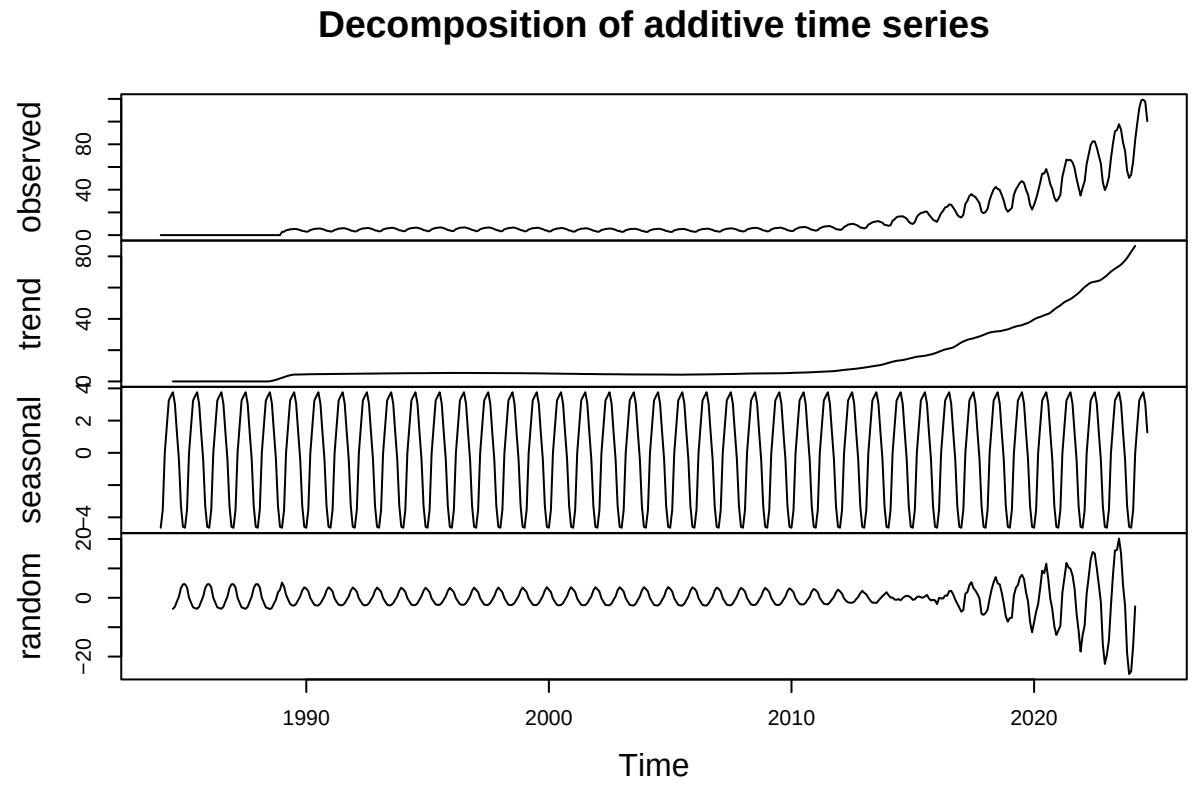
Additional info on `decompose()`.

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.

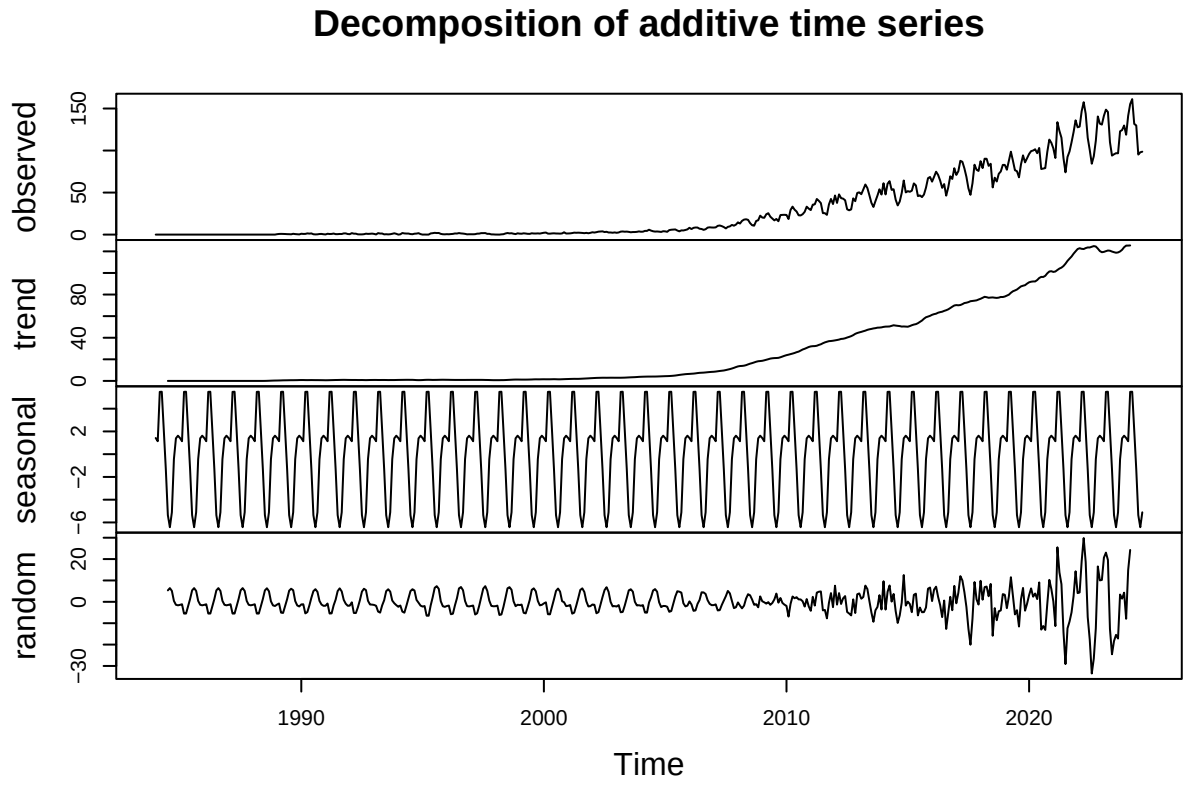
### Q4

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
#transform df to ts object
ts_solarwind_consumption <- ts(solarwind_consumption_df,
                               frequency = 12, start = c(1984,1))

#decompose solar
decompose_solar_add <- decompose(ts_solarwind_consumption[,2], 'additive')
plot(decompose_solar_add)
```




```r
#decompose wind
decompose_wind_add <- decompose(ts_solarwind_consumption[,3], 'additive')
plot(decompose_wind_add)
```

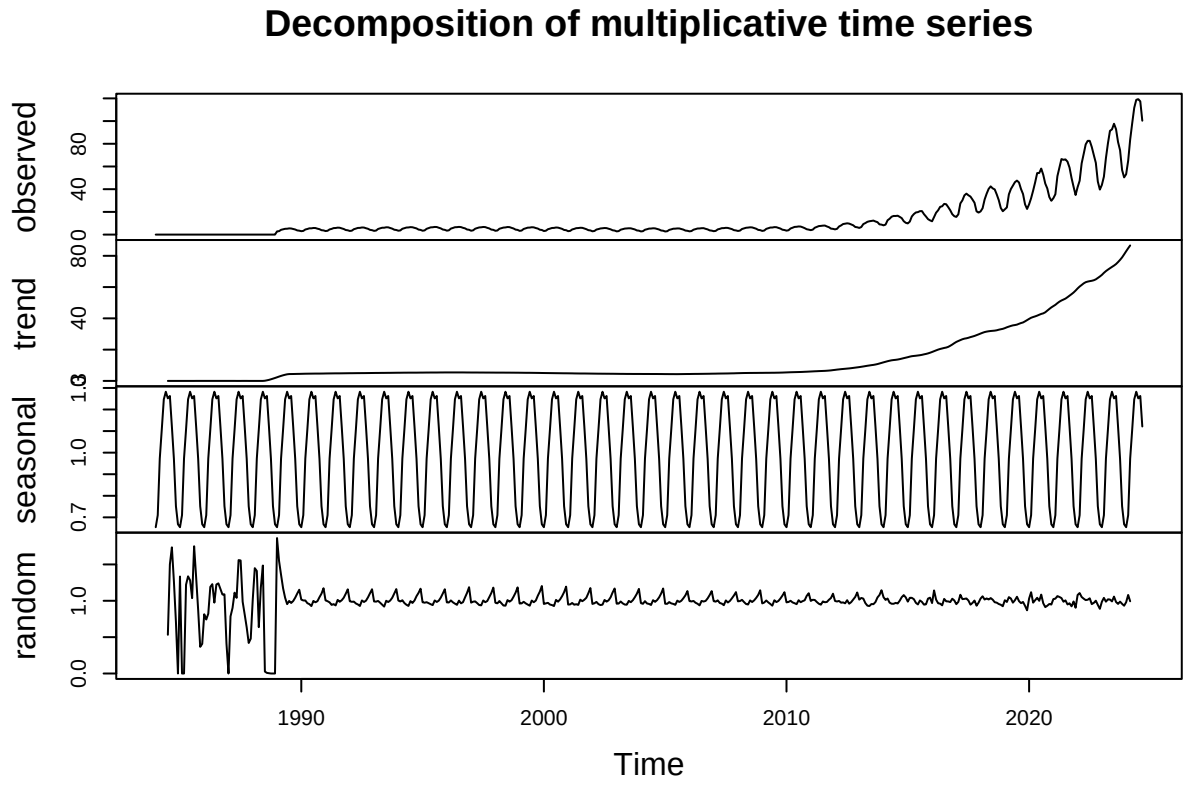



Answer: For solar energy consumption, there seems to be an increasing trend starting in the early 2010s. However, its random component doesn't look very random - it appears to show strong seasonality with regular peaks and lows and regular intervals. For wind energy consumption, there seems to be an increasing trend starting in the late 2000s until the early 2020s, where it starts to flatten and even slightly decrease. Its random component doesn't appear random until after around 2020. Before that, the random component appears to show seasonality.

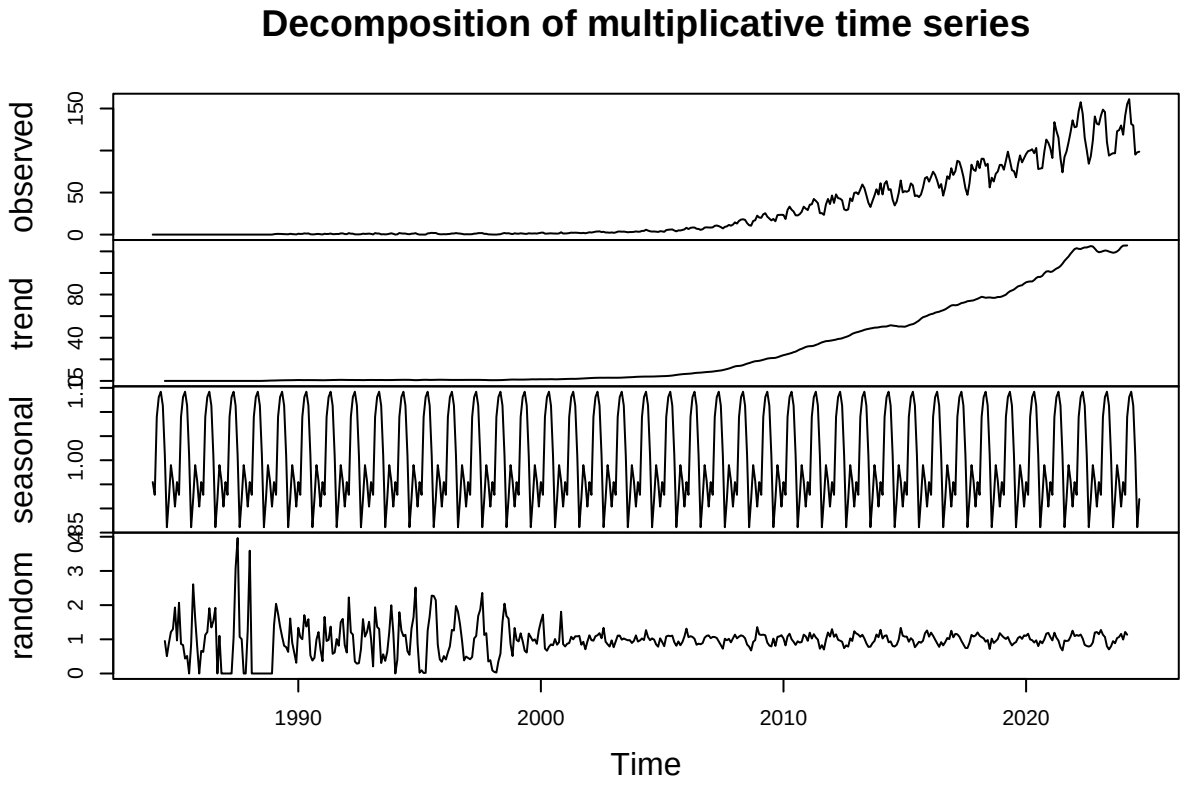
### Q5

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?

```
#decompose solar (multiplicative)
decompose_solar_multi <- decompose(ts_solarwind_consumption[,2], 'multiplicative')
plot(decompose_solar_multi)
```




```r
#decompose wind (multiplicative)
decompose_wind_multi <- decompose(ts_solarwind_consumption[,3], 'multiplicative')
plot(decompose_wind_multi)
```




Answer: The random component for the solar energy consumption series shows the most randomness before 1990. After that, there isn't much randomness until after the mid-2010s, but that is arguable, because there seemes to be a steady range of values throughout despite the slight randomness. The random component for the wind energy consumption series appears the most random before around 2000. After that, there isn't much randomness, but rather a general pattern of peaks and lows that suggests seasonality.

**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.

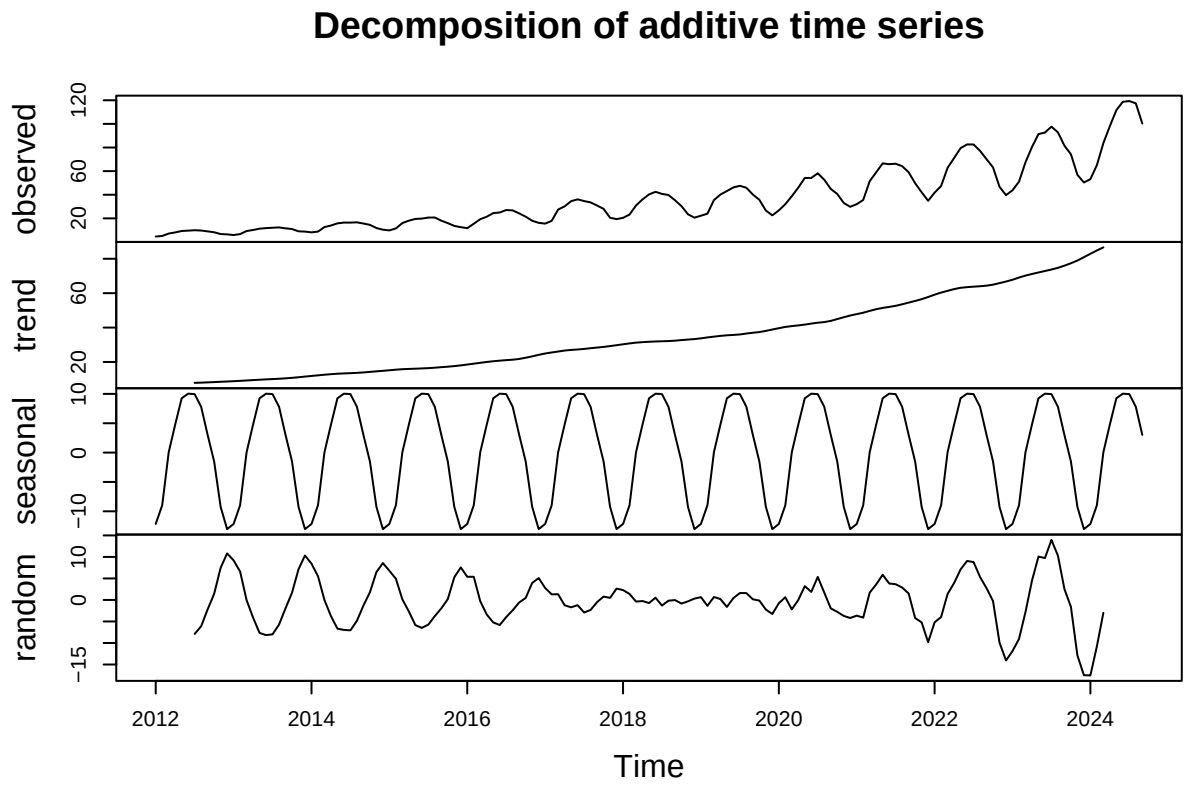
Answer: Because solar and wind energy consumption levels were so low during the 90s and early 2000s in comparison to now, mainly due to the fact that at the time these were still emerging technologies, the data from these periods do not reflect current trends or consumption patterns. Therefore, the outdated information from these years is likely not needed to forecast the next six months of solar and /or wind consumption.

**Q7**

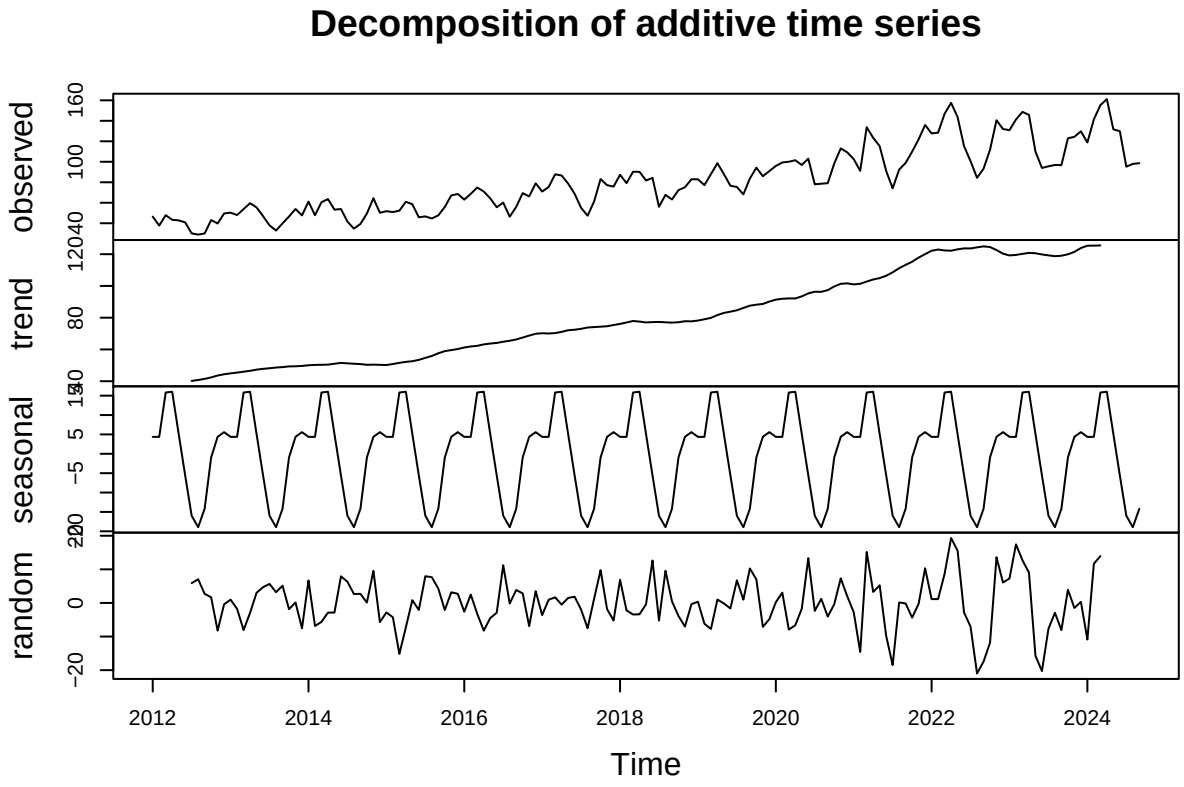
Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the

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 the data to start from January 2012
solarwind_consumption_filtered <- solarwind_consumption_df %>%
  filter(year(Month) >= 2012)

#transform new df to ts object
ts_solarwind_consumption_filtered <- ts(solarwind_consumption_filtered,
                                frequency = 12, start = c(2012,1))

#decompose solar filtered (additive)
decompose_solar_filtered_add <- decompose(ts_solarwind_consumption_filtered[,2], 'additive')
plot(decompose_solar_filtered_add)
```




```
#decompose wind filtered (additive)
decompose_wind_filtered_add <- decompose(ts_solarwind_consumption_filtered[,3], 'additive')
plot(decompose_wind_filtered_add)
```



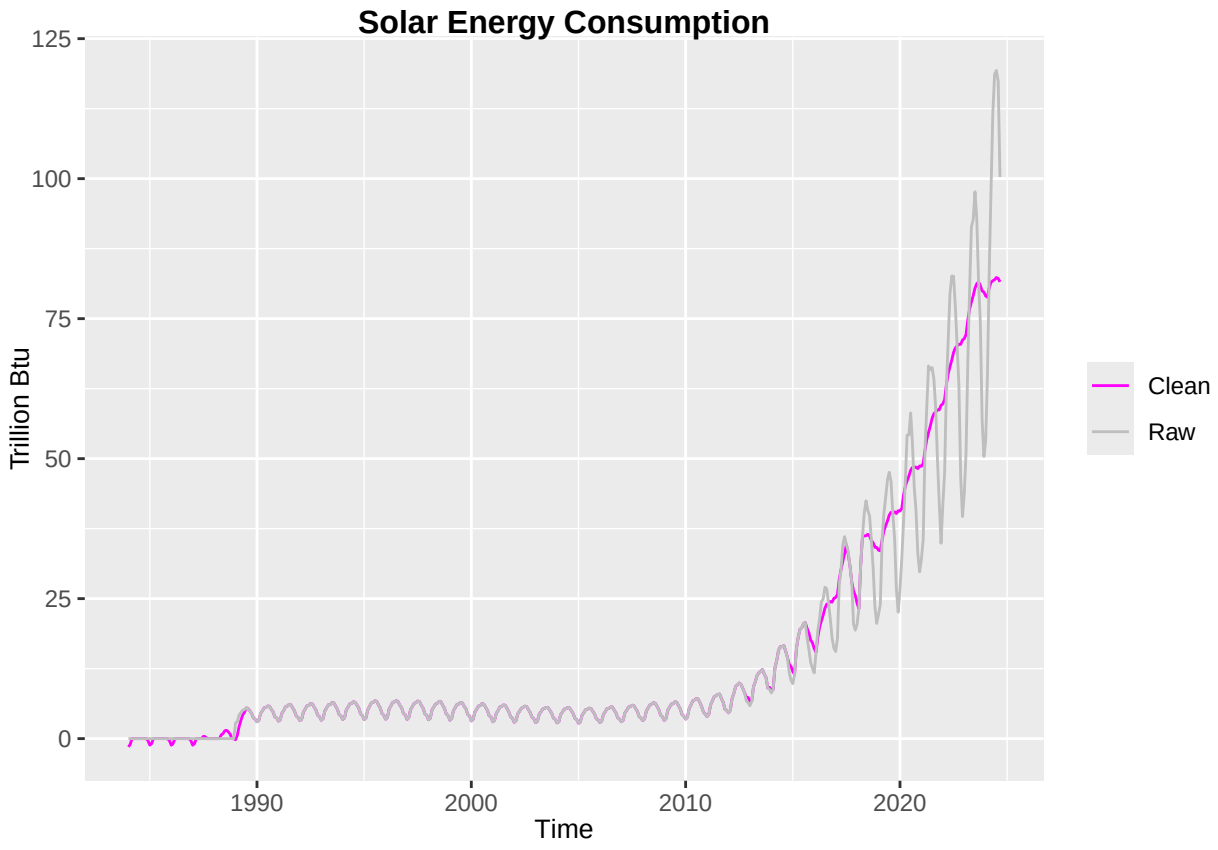

Answer: For the solar energy consumption series, the random component still doesn't appear very random. For the wind energy consumption series, the random component now looks much more random throughout all the time periods.

## Identify and Remove outliers

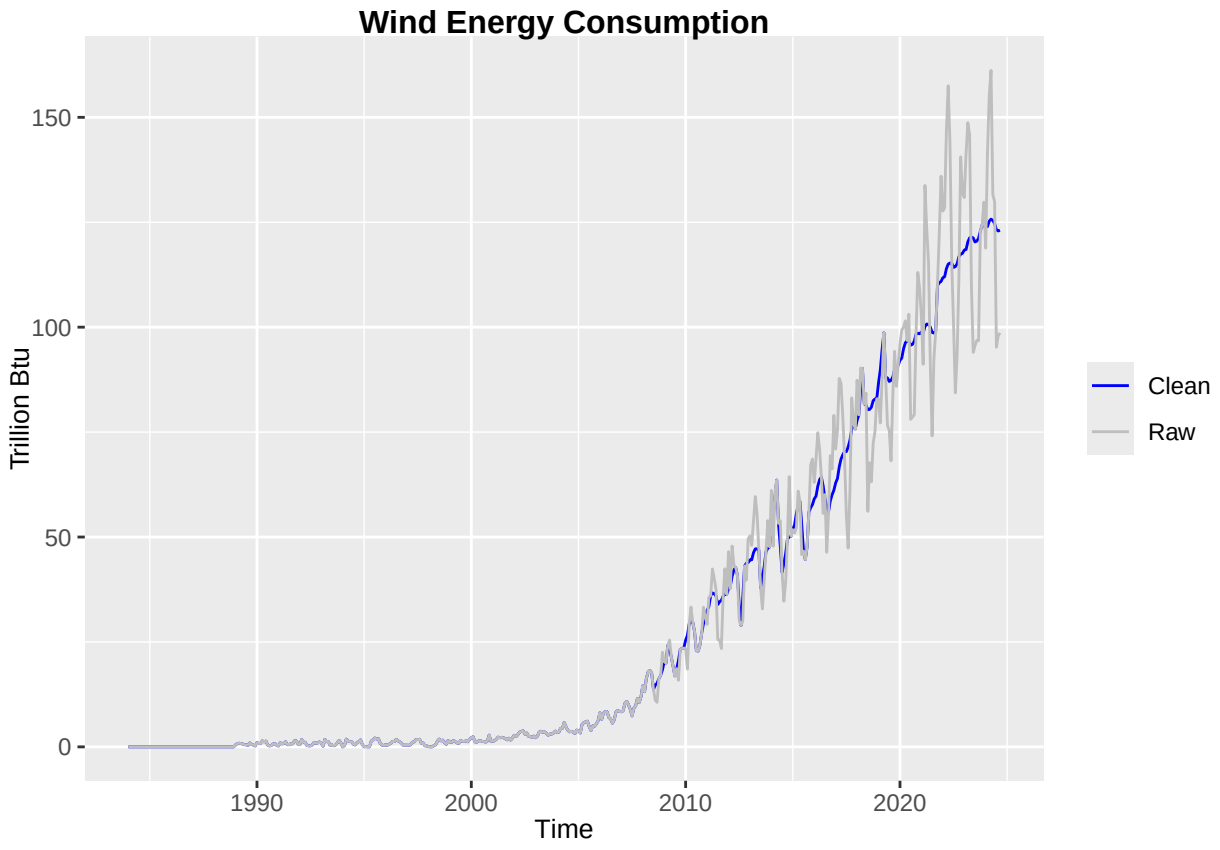
### Q8

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.

```
clean_solar_consumption <- tsclean(ts_solarwind_consumption[,2])
clean_wind_consumption <- tsclean(ts_solarwind_consumption[,3])

#solar original vs clean
autoplot(clean_solar_consumption, series ="Clean") +
  autolayer(ts_solarwind_consumption[,2], series ="Raw") +
  ggtitle('Solar Energy Consumption') +
  ylab("Trillion Btu") +
  scale_color_manual(values = c("Clean" = "magenta", "Raw" = "gray")) +
  guides(color = guide_legend(title = NULL))
```




```
#wind original vs clean
autoplot(clean_wind_consumption, series ="Clean") +
  autolayer(ts_solarwind_consumption[,3], series ="Raw") +
  ggtitle('Wind Energy Consumption') +
  ylab("Trillion Btu") +
  scale_color_manual(values = c("Clean" = "blue", "Raw" = "gray")) +
  guides(color = guide_legend(title = NULL))
```

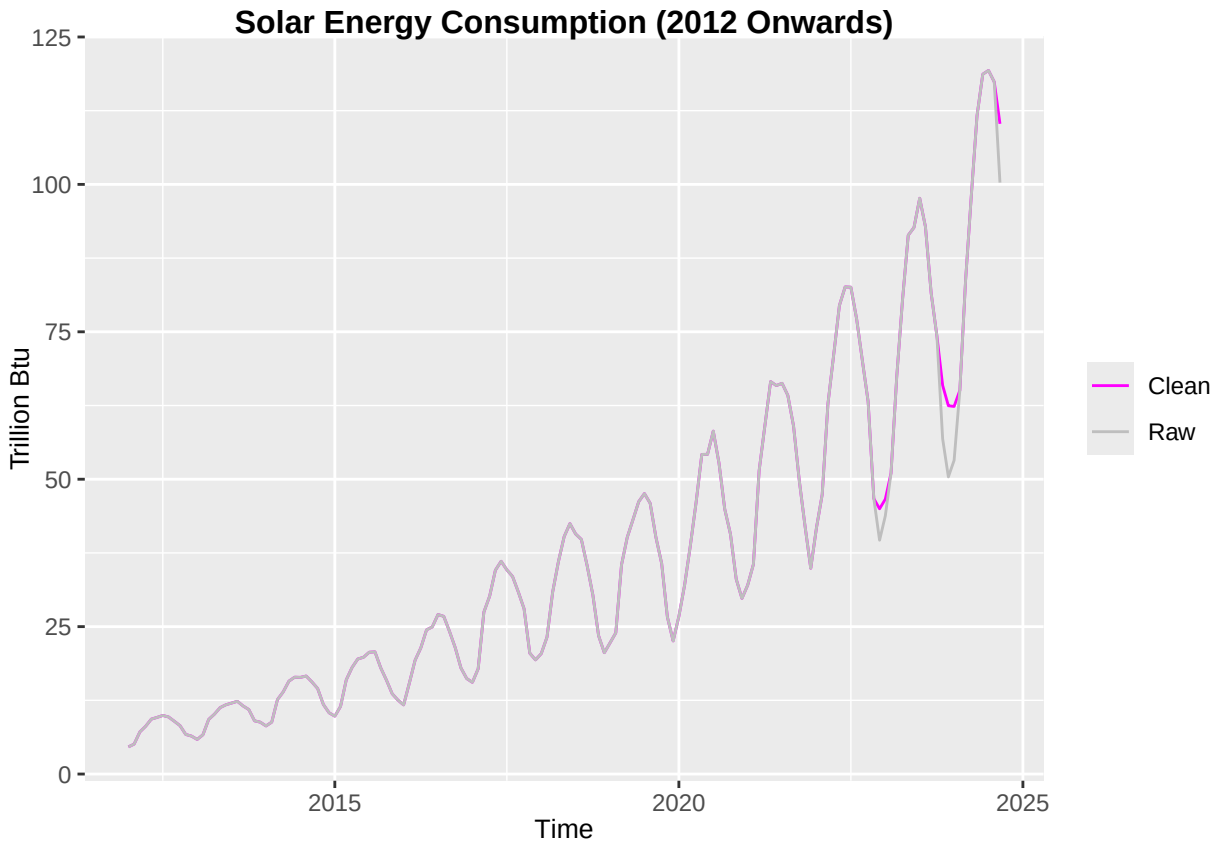



Answer: The function removed many outliers from both time series.

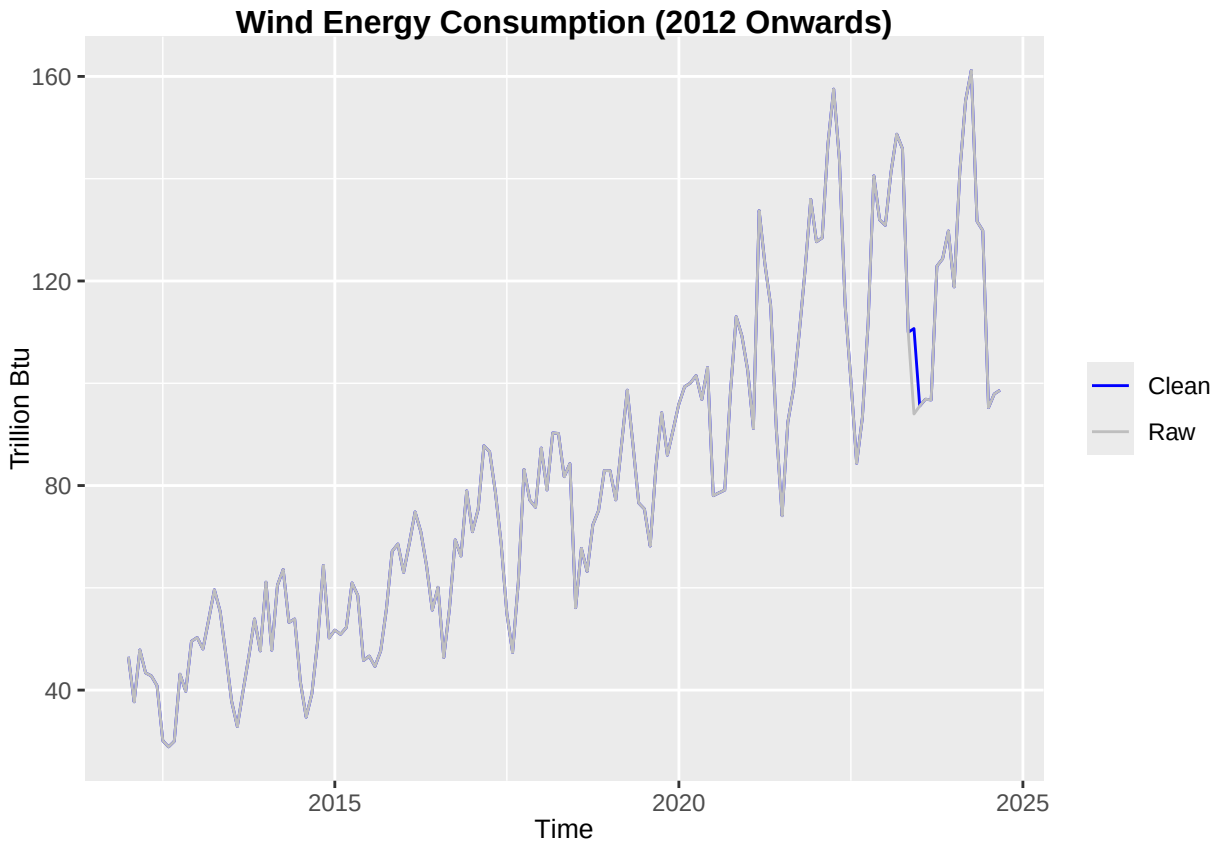
**Q9**

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?

```
clean_solar_consumption_filtered <- tsclean(ts_solarwind_consumption_filtered[,2])
clean_wind_consumption_filtered <- tsclean(ts_solarwind_consumption_filtered[,3])

#solar original vs clean (filtered)
autoplot(clean_solar_consumption_filtered, series ="Clean") +
  autolayer(ts_solarwind_consumption_filtered[,2], series ="Raw") +
  ggtitle('Solar Energy Consumption (2012 Onwards)') +
  ylab("Trillion Btu") +
  scale_color_manual(values = c("Clean" = "magenta", "Raw" = "gray")) +
  guides(color = guide_legend(title = NULL))
```




```
#wind original vs clean (filtered)
autoplot(clean_wind_consumption_filtered, series ="Clean") +
  autolayer(ts_solarwind_consumption_filtered[,3], series ="Raw") +
  ggtitle('Wind Energy Consumption (2012 Onwards)') +
  ylab("Trillion Btu") +
  scale_color_manual(values = c("Clean" = "blue", "Raw" = "gray")) +
  guides(color = guide_legend(title = NULL))
```




Answer: Using the data from 2012 onwards, it looks like the function removed three outliers from the solar energy consumption series and one outlier from the wind energy consumption series, a big difference compared to using the function for the data starting from 1984.