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

Assignment 5 - Due date 02/17/26

Yeeun Kim

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
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.5.2
```

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

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.5.2
```

```
library(ggplot2)
library(Kendall)
```

```
## Warning: package 'Kendall' was built under R version 4.5.2
```

```r
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 1.5.2
## v forcats 1.0.0     v tibble  3.3.0
## v purrr   1.1.0     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
```

```r
library(readxl)
```

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.

```r
#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`
```

```r
#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!

```
#Select the columns
energy_select <- energy_data %>%
  select(1,8,9)
head(energy_select)
```

```
## # A tibble: 6 x 3
##   Month               `Solar Energy Consumption` `Wind Energy Consumption`
##   <dttm>              <chr>                      <chr>
## 1 1973-01-01 00:00:00 Not Available              Not Available
## 2 1973-02-01 00:00:00 Not Available              Not Available
## 3 1973-03-01 00:00:00 Not Available              Not Available
## 4 1973-04-01 00:00:00 Not Available              Not Available
## 5 1973-05-01 00:00:00 Not Available              Not Available
## 6 1973-06-01 00:00:00 Not Available              Not Available
```

```
#Check the class of columns
class(energy_select[[2]])
```

```
## [1] "character"
```

```
class(energy_select[[3]])
```

```
## [1] "character"
```

```
#Remove NAs and convert to numeric
energy_select <- energy_select %>%
  mutate(`Solar Energy Consumption` = as.numeric(`Solar Energy Consumption`)) %>%
  mutate(`Wind Energy Consumption` = as.numeric(`Wind Energy Consumption`)) %>%
  drop_na()
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Solar Energy Consumption = as.numeric(`Solar Energy
##   Consumption`)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## Warning: There was 1 warning in ‘mutate()‘.
## i In argument: ‘Wind Energy Consumption = as.numeric(‘Wind Energy
##   Consumption‘)‘.
## Caused by warning:
## ! NAs introduced by coercion
```

```
class(energy_select[[2]])
```

```
## [1] "numeric"
```

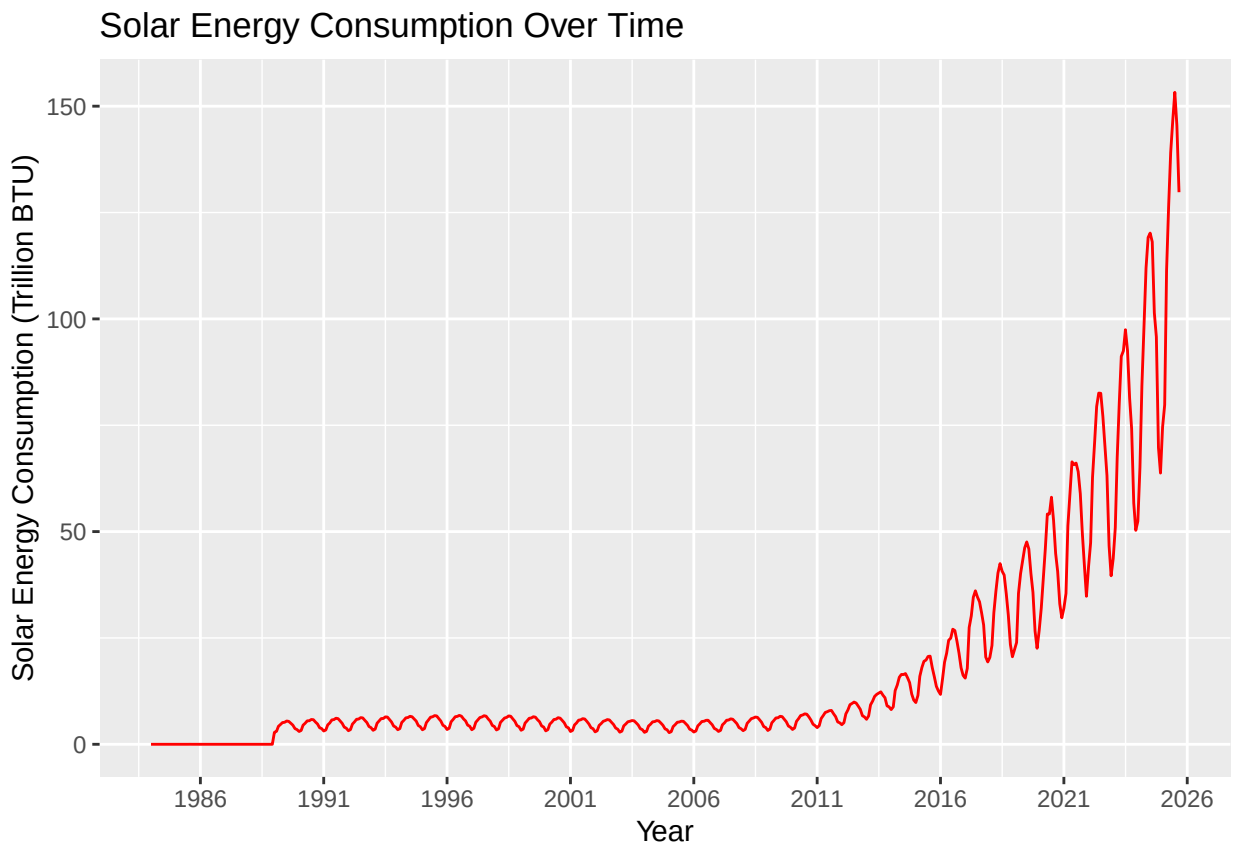
```
class(energy_select[[3]])
```

```
## [1] "numeric"
```

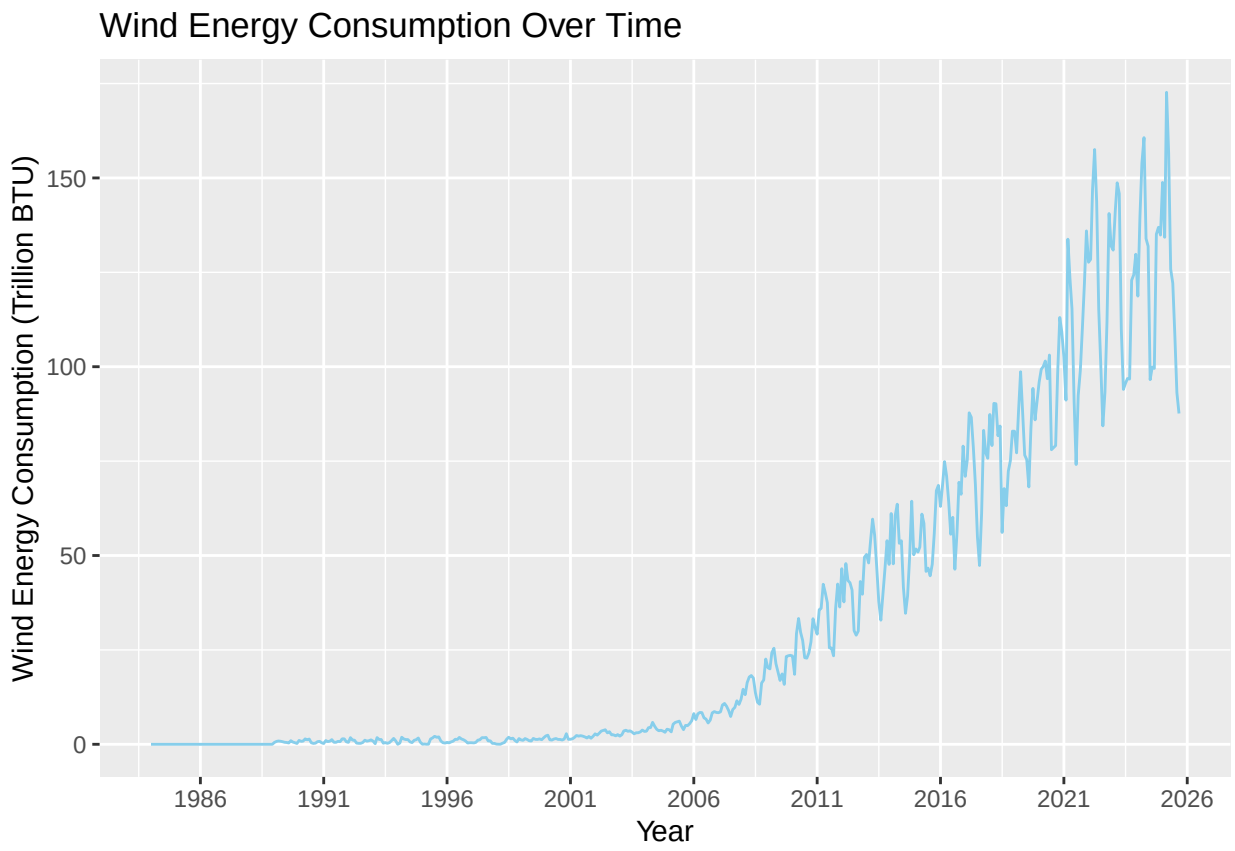
Q2

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
ggplot(energy_select, aes(x = Month, y = `Solar Energy Consumption`)) +
  geom_line(color = "red") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ylab("Solar Energy Consumption (Trillion BTU)") +
  xlab("Year") +
  ggtitle("Solar Energy Consumption Over Time")
```



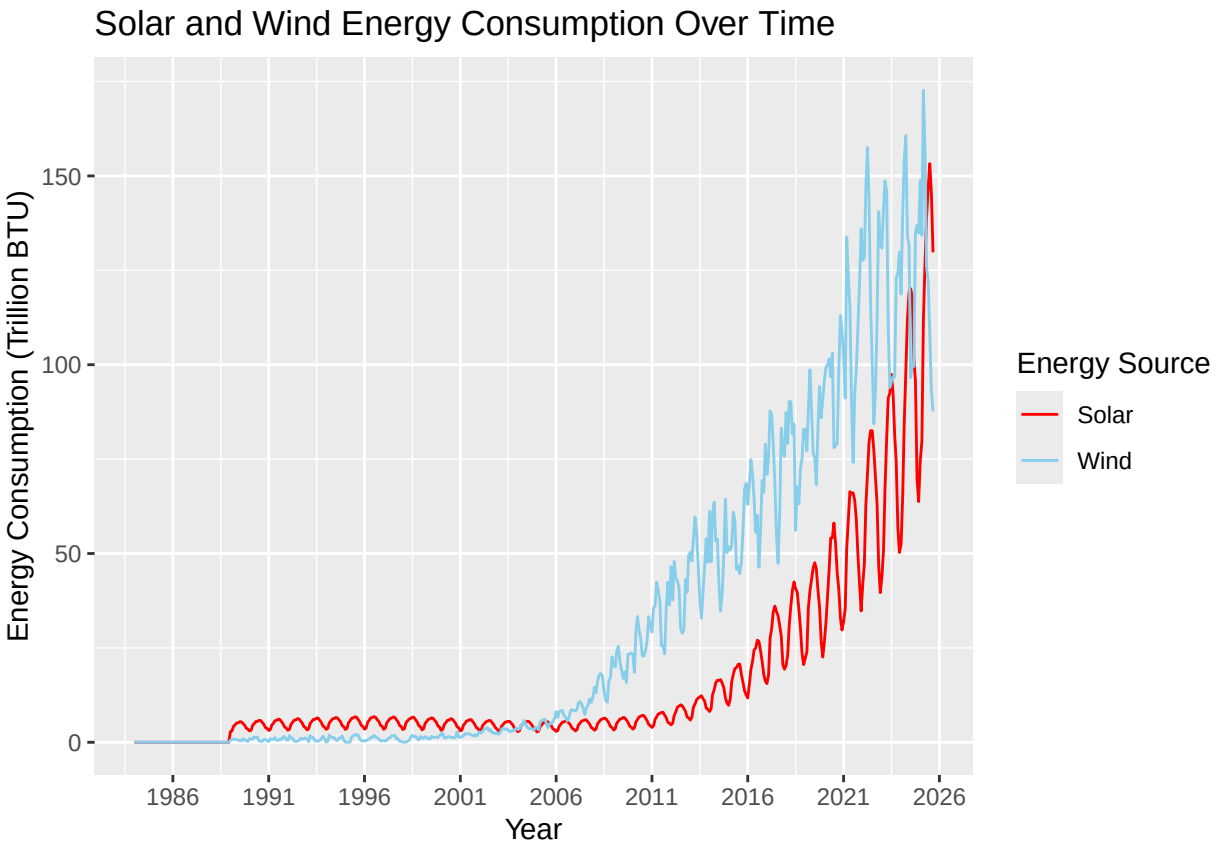

```
# Wind Energy Consumption
ggplot(energy_select, aes(x = Month, y = `Wind Energy Consumption`)) +
  geom_line(color = "skyblue") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ylab("Wind Energy Consumption (Trillion BTU)") +
  xlab("Year") +
  ggtitle("Wind Energy Consumption Over Time")
```



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(energy_select, aes(x = Month)) +
  geom_line(aes(y = `Solar Energy Consumption`, color = "Solar")) +
  geom_line(aes(y = `Wind Energy Consumption`, color = "Wind")) +
  scale_color_manual(
    name = "Energy Source",
    values = c("Solar" = "red", "Wind" = "skyblue")
  ) +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ylab("Energy Consumption (Trillion BTU)") +
  xlab("Year") +
  ggtitle("Solar and Wind Energy Consumption Over Time")
```



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()`.

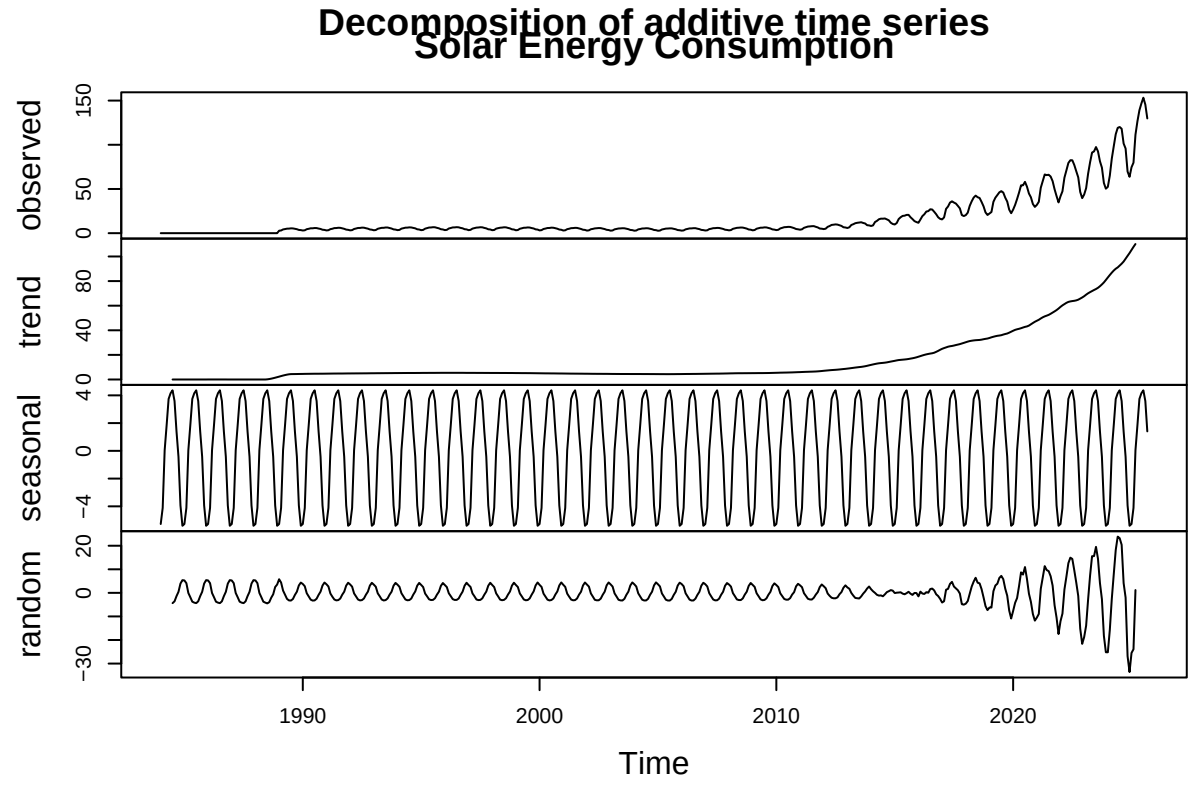
1) You have two options: alternative and multiplicative. Multiplicative models exhibit a change in frequency over time.
2) The trend is not a straight line because it uses a moving average method to detect trend.
3) The seasonal component of the time series is found by subtracting the trend component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

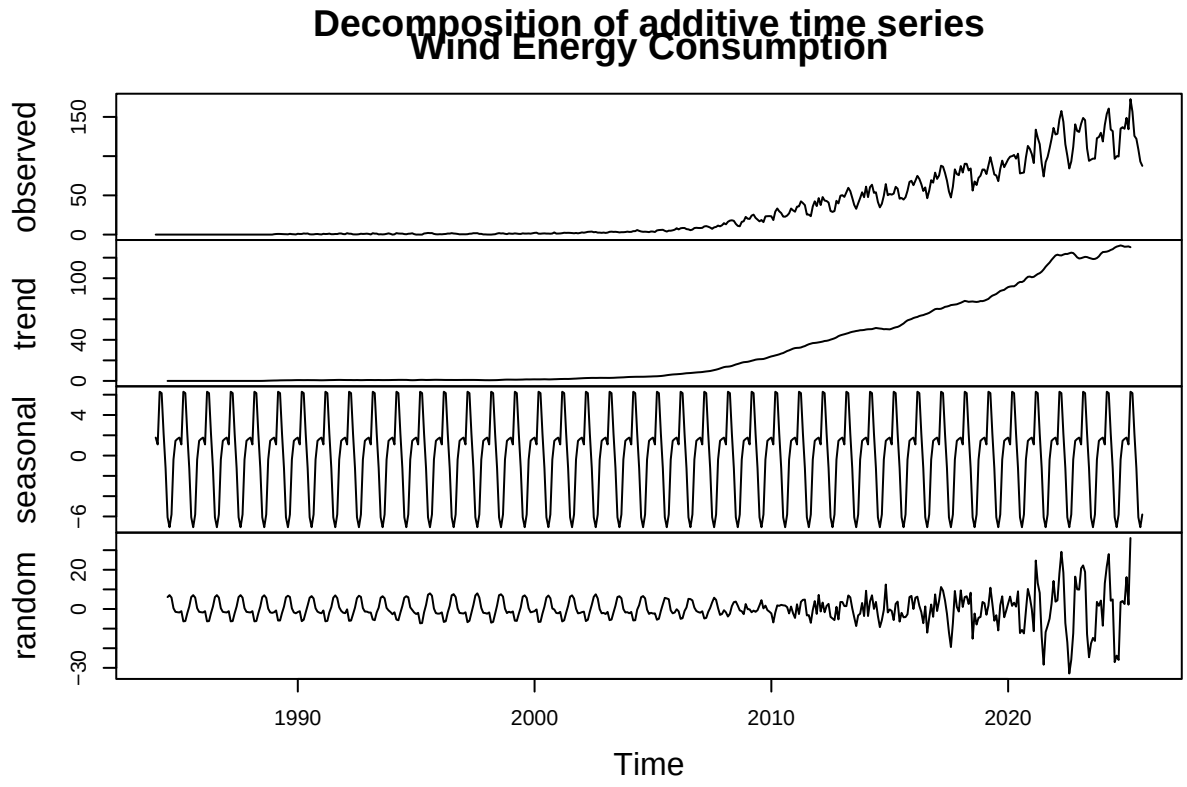
Transform wind and solar series into a time series object and apply the decompose function on them using the additive option, i.e., `decompose(ts_data, type = "additive")`. What can you say about the trend component? What about the random component? Does the random component look random? Or does it appear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

For both solar and wind energy consumption, the trend is increasing, showing an upward pattern. The random components of both fluctuate around zero, meaning seasonality has been removed. However, the random components increase over time, after around 2015 for solar and around 2010 for wind.

```
# TS objects
ts_solar <- ts(energy_select$`Solar Energy Consumption`,
               start = c(1984,1), frequency = 12)

ts_wind <- ts(energy_select$`Wind Energy Consumption`,
              start = c(1984,1), frequency = 12)

# Decompose
decomp_solar <- decompose(ts_solar, type = "additive")
decomp_wind <- decompose(ts_wind, type = "additive")

# Plot
plot(decomp_solar)
title(main = "Solar Energy Consumption")
```




```
plot(decomp_wind)
title(main = "Wind Energy Consumption")
```

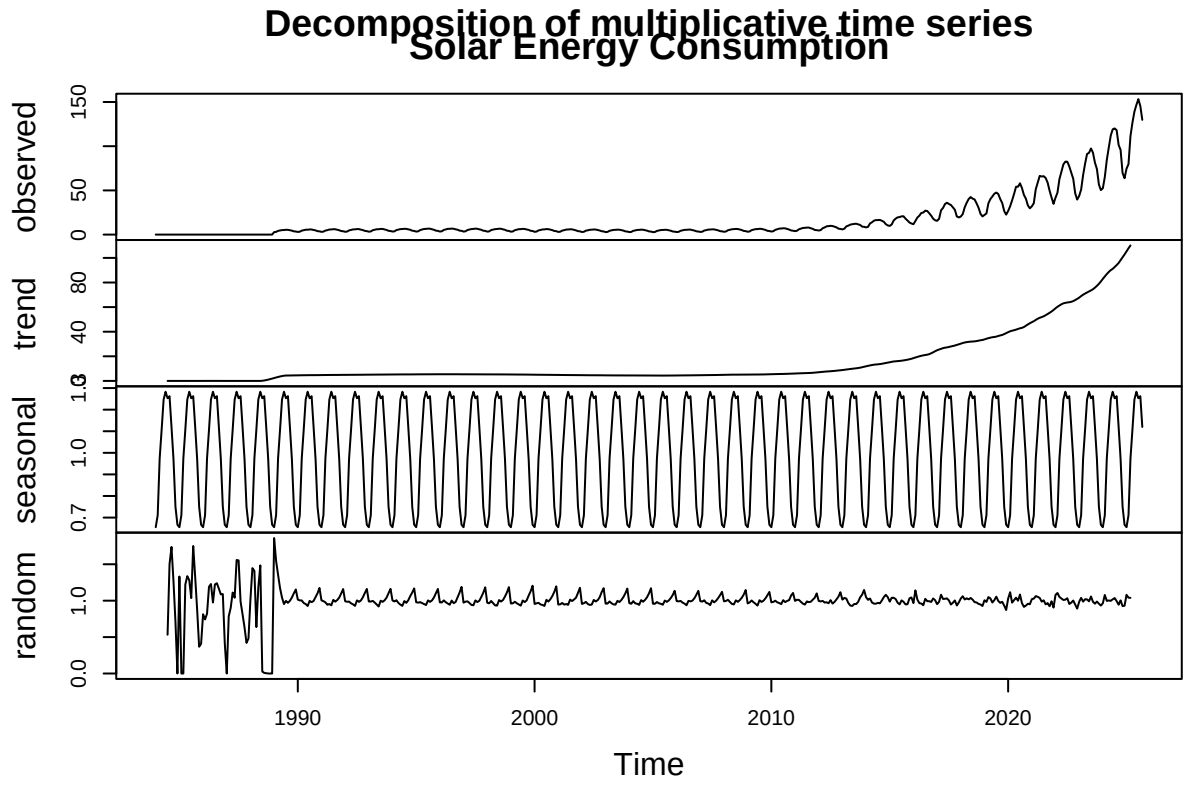


Q5

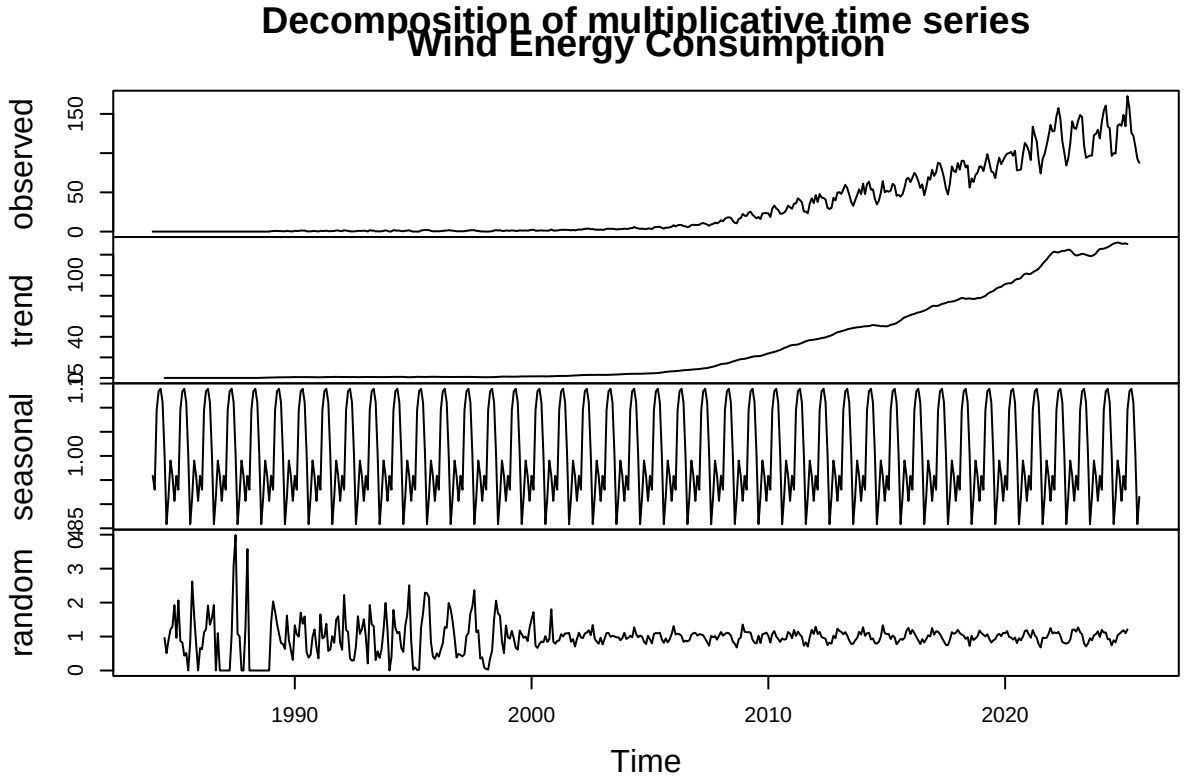
Use the decompose function again but now change the type of the seasonal component from additive to multiplicative. What happened to the random component this time?

The random component fluctuates around 1 and appears more stable over time compared to the additive model. This is because the random component is calculated as a ratio rather than an absolute deviation. Therefore, it seems like variability in solar and wind energy consumption increase proportionally over time.

```
# Decompose
decomp_solar <- decompose(ts_solar, type = "multiplicative")
decomp_wind <- decompose(ts_wind, type = "multiplicative")

# Plot
plot(decomp_solar)
title(main = "Solar Energy Consumption")
```




```
plot(decomp_wind)
title(main = "Wind Energy Consumption")
```



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 think I don't need all the historical data. This is because both solar and wind energy consumption show very low observation until 2010. To forecast the next six months consumption, the recent data might be more helpful.

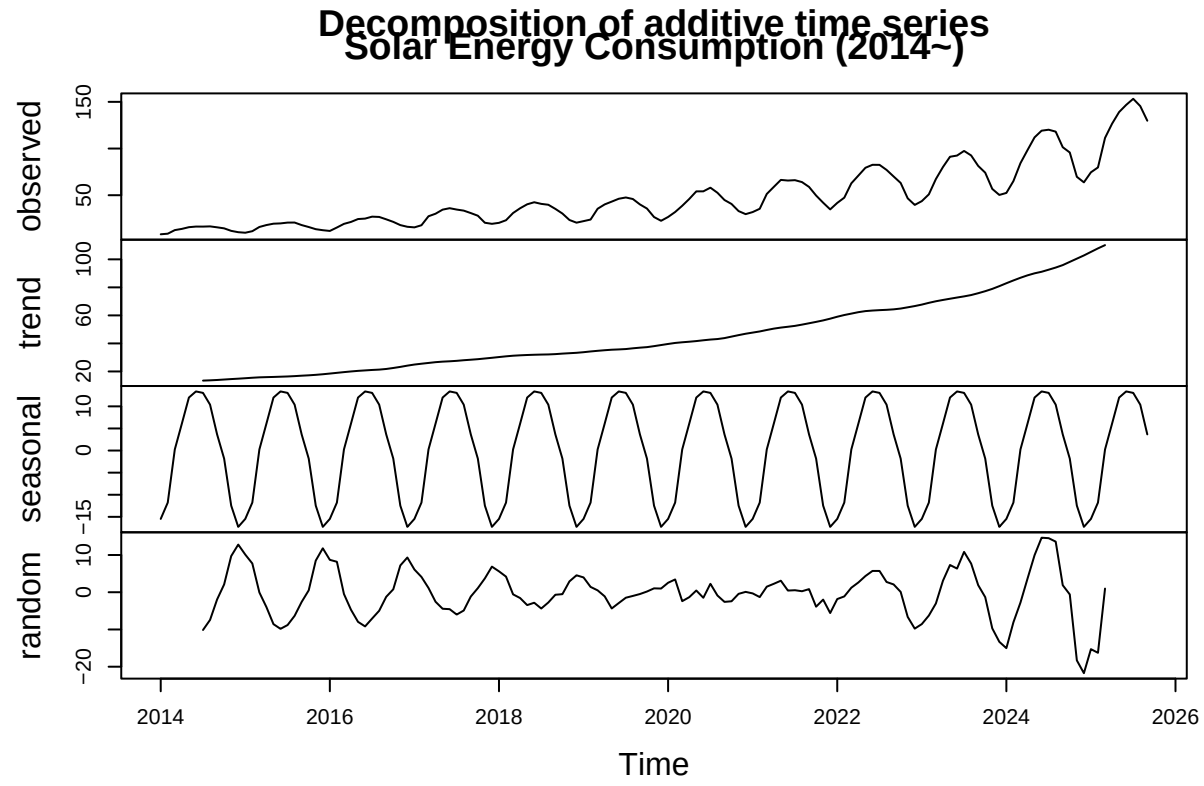
Q7

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?

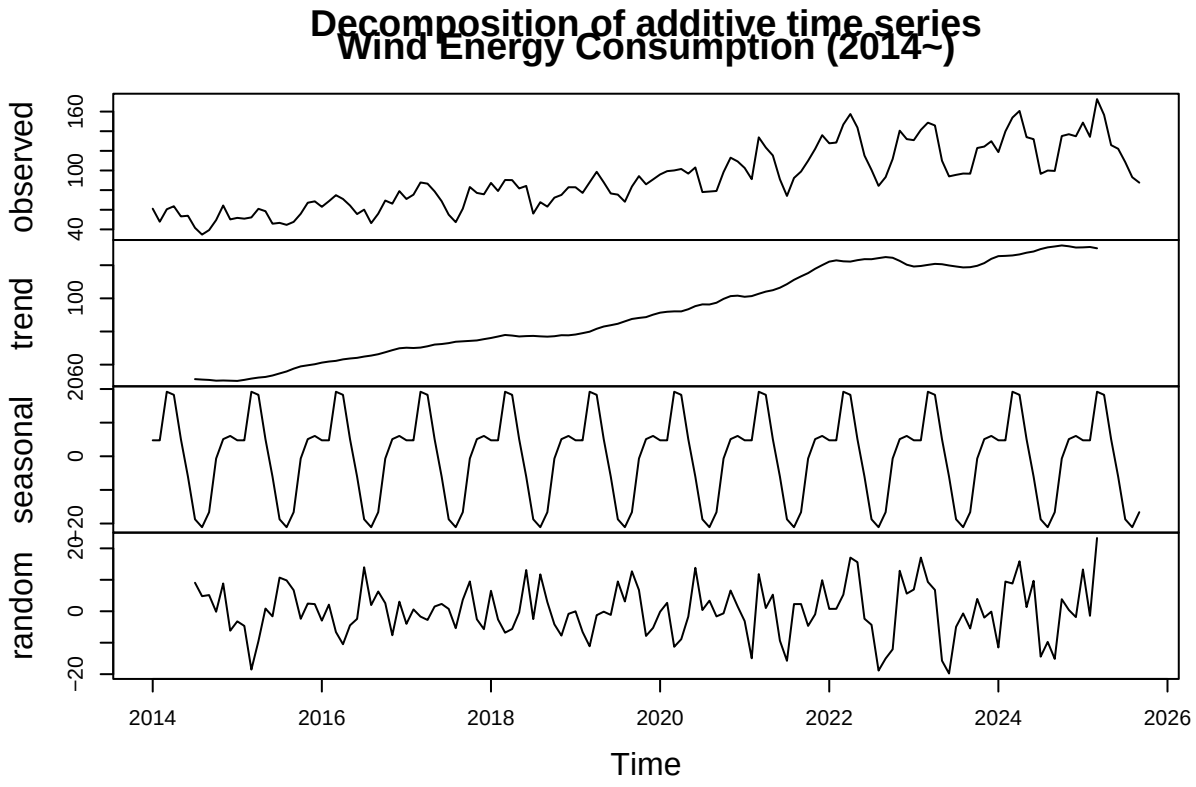
Both energy consumption show upward trend and clear seasonality. Both random components fluctuate around zero, but variance increase after 2022. It indicates both random compenents are not perfectly random.

```
# Filter data from 2014
energy_2014 <- energy_select %>%
  filter(year(Month) >= 2014)
```

```r
# TS objects
ts_solar_2014 <- ts(energy_2014$`Solar Energy Consumption`,
                    start = c(2014, 1), frequency = 12)

ts_wind_2014 <- ts(energy_2014$`Wind Energy Consumption`,
                   start = c(2014, 1), frequency = 12)

# Decompose
decomp_solar_2014 <- decompose(ts_solar_2014, type = "additive")
decomp_wind_2014 <- decompose(ts_wind_2014, type = "additive")

# Plot
plot(decomp_solar_2014)
title("Solar Energy Consumption (2014~)")
```




```r
plot(decomp_wind_2014)
title("Wind Energy Consumption (2014~)")
```



Answer:

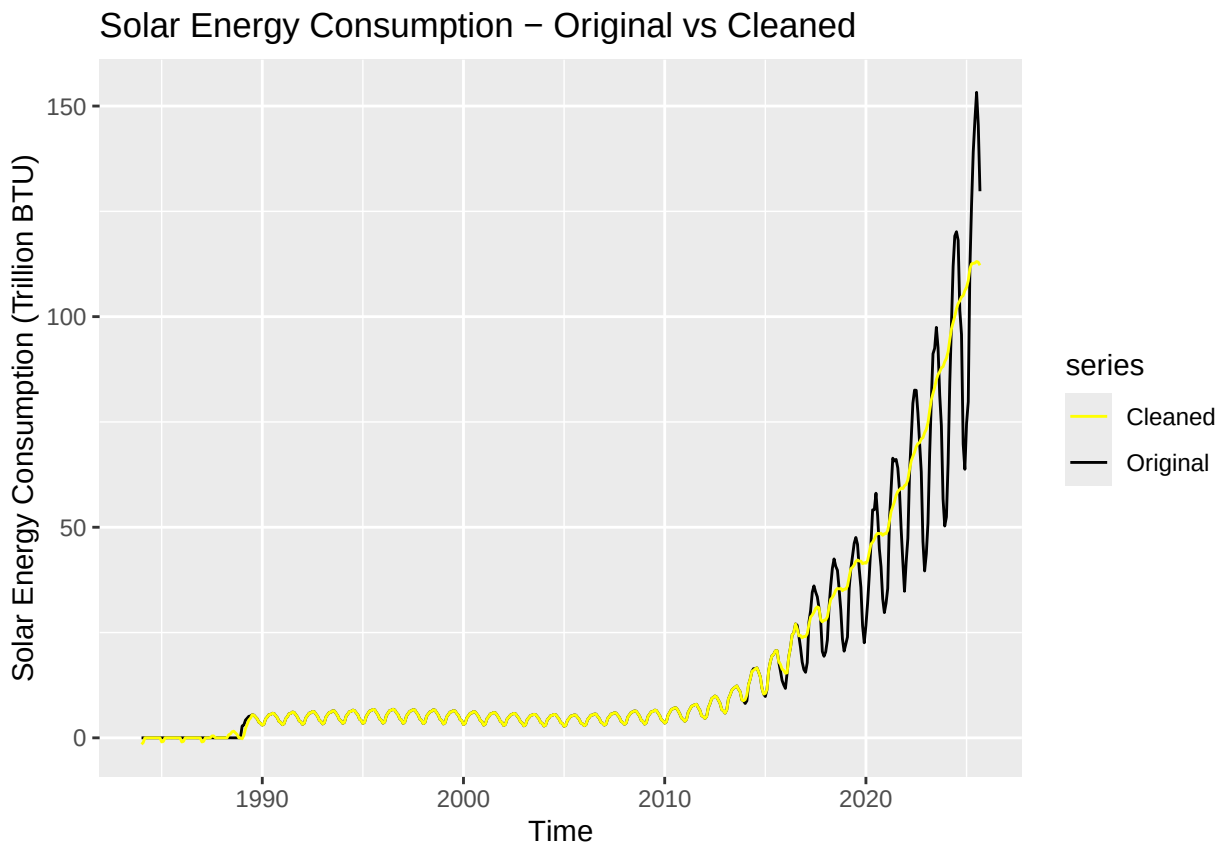
Identify and Remove outliers

Q8

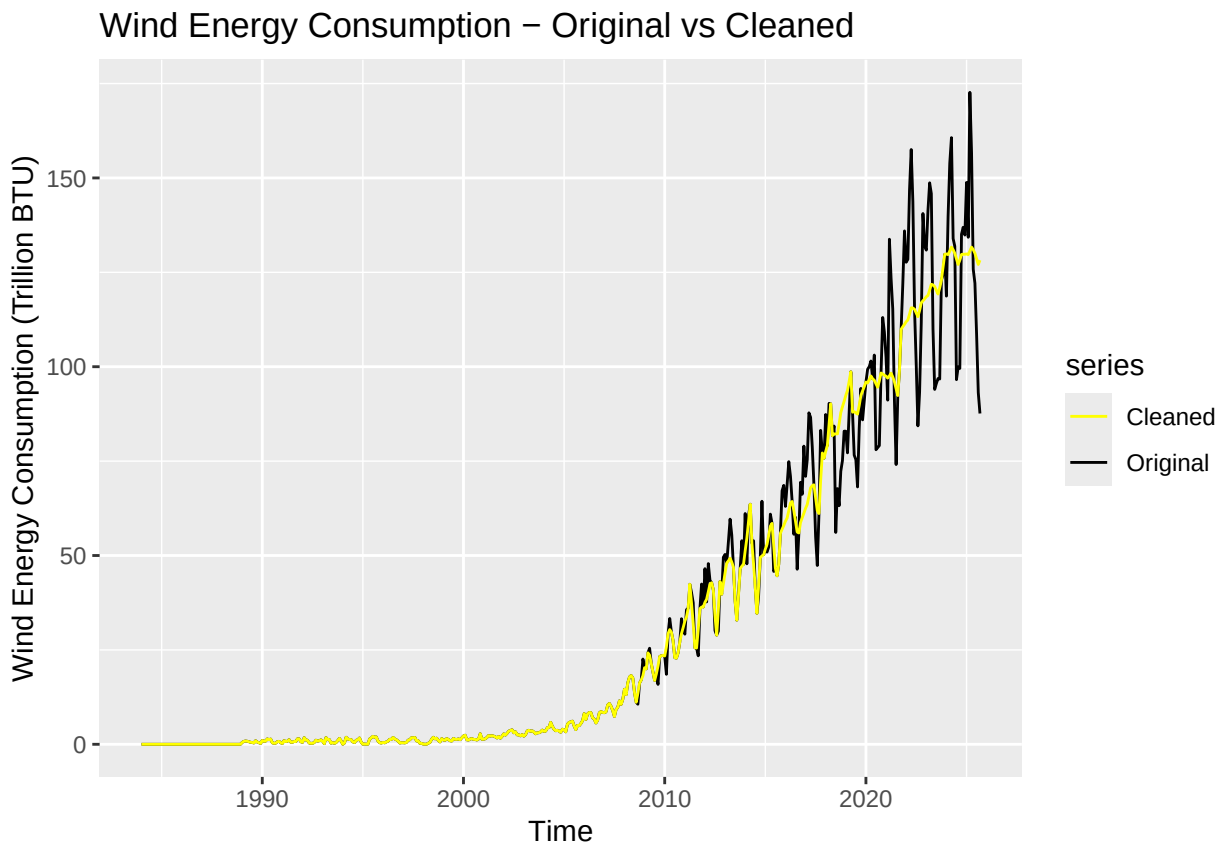
Apply the `tsclean()` to both time series object you created on Q4. Did the function removed any outliers from the series? Hint: Use `autoplot()` to check if there is difference between cleaned series and original series.

The function removed outliers in recent years for both solar and wind energy consumption.

```
# Apply tsclean()
clean_ts_solar <- tsclean(ts_solar)
clean_ts_wind <- tsclean(ts_wind)

# Compare original vs cleaned using autoplot()
autoplot(ts_solar, series = "Original") +
  autolayer(clean_ts_solar, series = "Cleaned") +
  scale_color_manual(values = c("Original" = "black", "Cleaned" = "yellow")) +
  ylab("Solar Energy Consumption (Trillion BTU)") +
  ggtitle("Solar Energy Consumption - Original vs Cleaned")
```



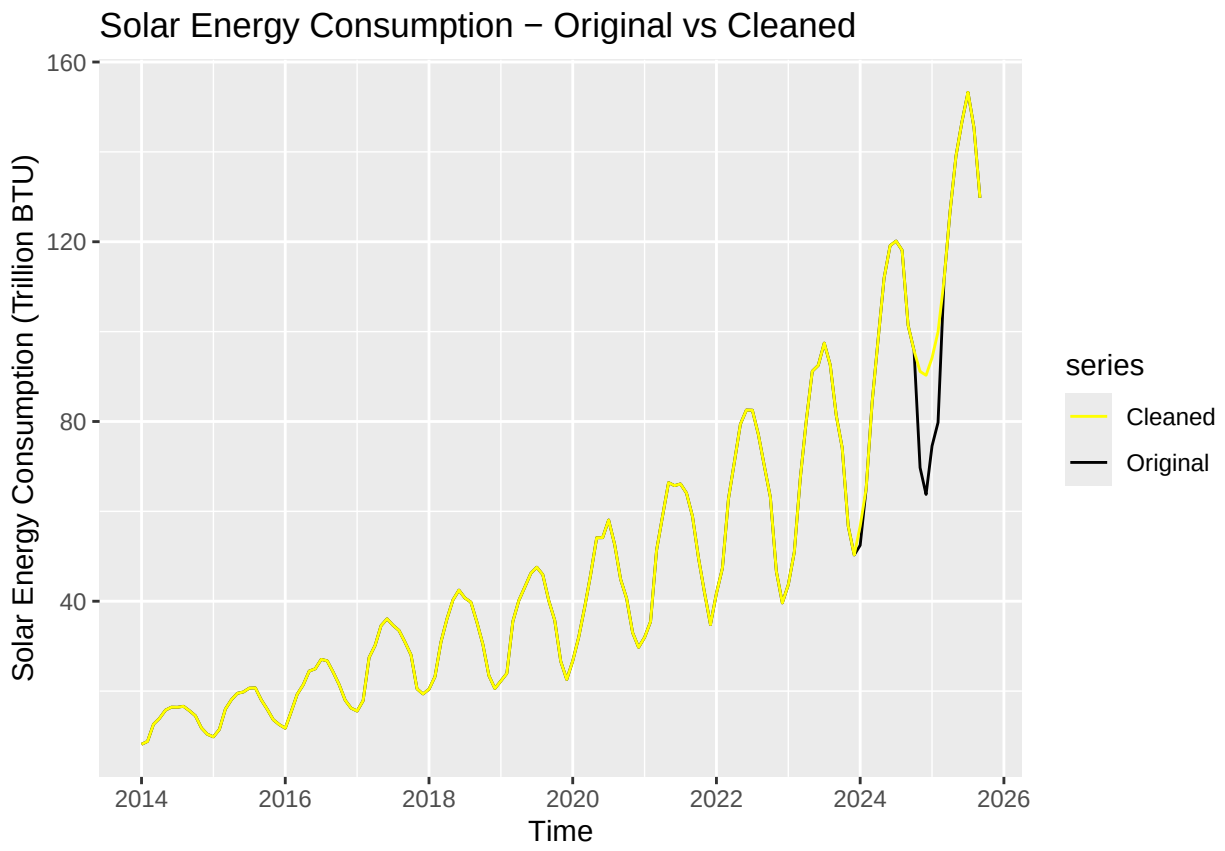

```
autoplot(ts_wind, series = "Original") +
  autolayer(clean_ts_wind, series = "Cleaned") +
  scale_color_manual(values = c("Original" = "black", "Cleaned" = "yellow")) +
  ylab("Wind Energy Consumption (Trillion BTU)") +
  ggtitle("Wind Energy Consumption - Original vs Cleaned")
```



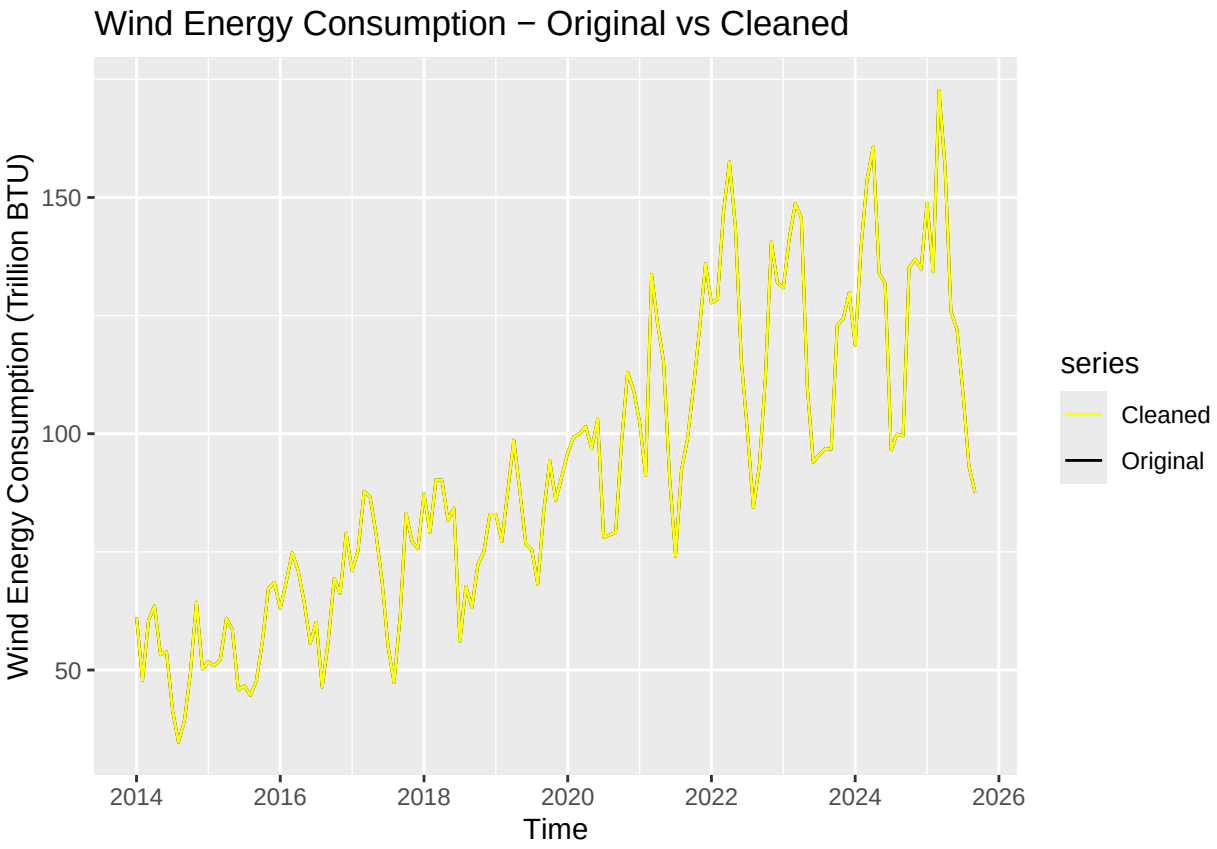
Q9

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_ts_solar_2014 <- tsclean(ts_solar_2014)
clean_ts_wind_2014 <- tsclean(ts_wind_2014)

# Compare original vs cleaned using autoplot()
autoplot(ts_solar_2014, series = "Original") +
  autolayer(clean_ts_solar_2014, series = "Cleaned") +
  scale_color_manual(values = c("Original" = "black", "Cleaned" = "yellow")) +
  ylab("Solar Energy Consumption (Trillion BTU)") +
  ggtitle("Solar Energy Consumption - Original vs Cleaned")
```




```
autoplot(ts_wind_2014, series = "Original") +
  autolayer(clean_ts_wind_2014, series = "Cleaned") +
  scale_color_manual(values = c("Original" = "black", "Cleaned" = "yellow")) +
  ylab("Wind Energy Consumption (Trillion BTU)") +
  ggtitle("Wind Energy Consumption - Original vs Cleaned")
```



Answer:Only minor removal seems to have been made in the solar series around 2024–2025, but overall, there are no outlier removal in both solar and wind energy consumption.