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

Assignment 5 - Due date 02/17/26

Lauren Shohan

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)
```

```
## 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(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="/home/guest/TSA_Sp26/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="/home/guest/TSA_Sp26/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
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!

```
energy_data1 <- energy_data[,c(1,8,9)]
energy_data1$Month <- ymd(energy_data1$Month)

summary(energy_data1)
```

```
##      Month            Solar Energy Consumption Wind Energy Consumption
##  Min.   :1973-01-01   Length:633               Length:633
##  1st Qu.:1986-03-01   Class :character         Class :character
##  Median :1999-05-01   Mode  :character         Mode  :character
##  Mean   :1999-05-02
##  3rd Qu.:2012-07-01
##  Max.   :2025-09-01
```

```
#converting to numeric
energy_data1$`Solar Energy Consumption` <- as.numeric(energy_data1$`Solar Energy Consumption`)
```

```
## Warning: NAs introduced by coercion
```

```
energy_data1$`Wind Energy Consumption` <- as.numeric(energy_data1$`Wind Energy Consumption`)
```

```
## Warning: NAs introduced by coercion
```

```
summary(energy_data1) #solar has 132 NAs and wind 120 NAs
```

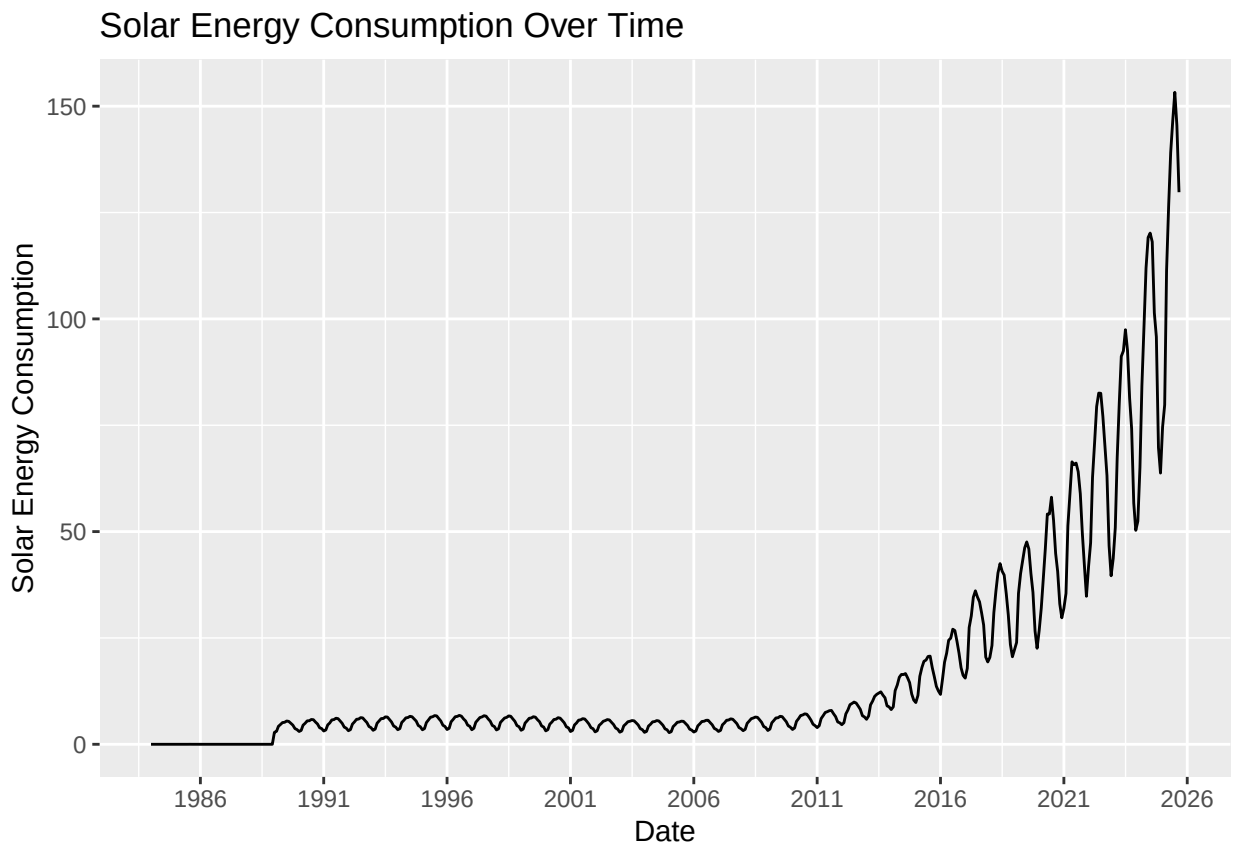
```
##      Month            Solar Energy Consumption Wind Energy Consumption
##  Min.   :1973-01-01   Min.   :  0.000          Min.   :  0.000
##  1st Qu.:1986-03-01   1st Qu.:  3.924          1st Qu.:  0.763
##  Median :1999-05-01   Median :  5.658          Median :  3.729
##  Mean   :1999-05-02   Mean   : 16.623          Mean   : 30.573
##  3rd Qu.:2012-07-01   3rd Qu.: 15.758          3rd Qu.: 53.758
##  Max.   :2025-09-01   Max.   :153.256          Max.   :172.670
##                       NA's   :132              NA's   :120
```

```
energy_data1 <- drop_na(energy_data1)
```

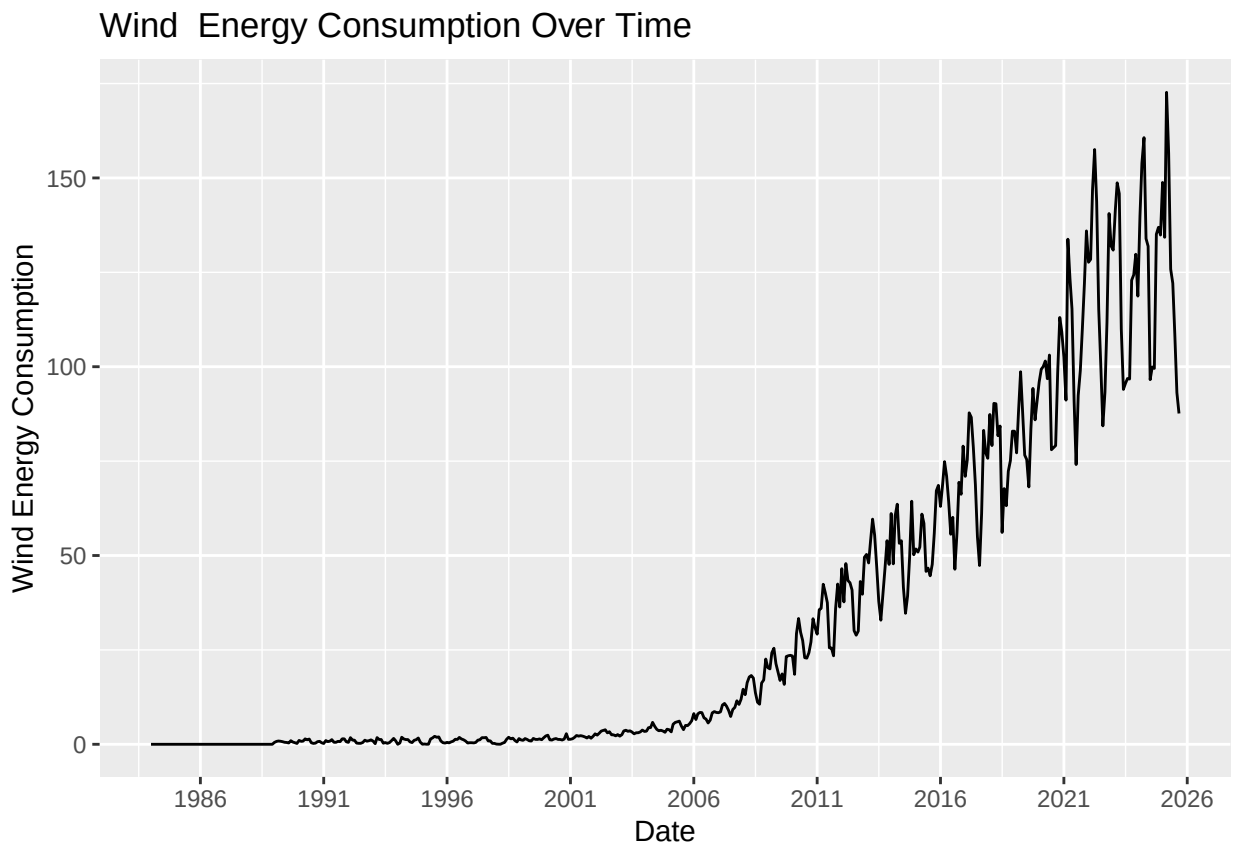
Q2

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")")`

```
ggplot(energy_data1,aes(x = Month, y = `Solar Energy Consumption`)) +
  geom_line() +
  ylab('Solar Energy Consumption') +
  xlab('Date') +
  ggtitle('Solar Energy Consumption Over Time') +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```



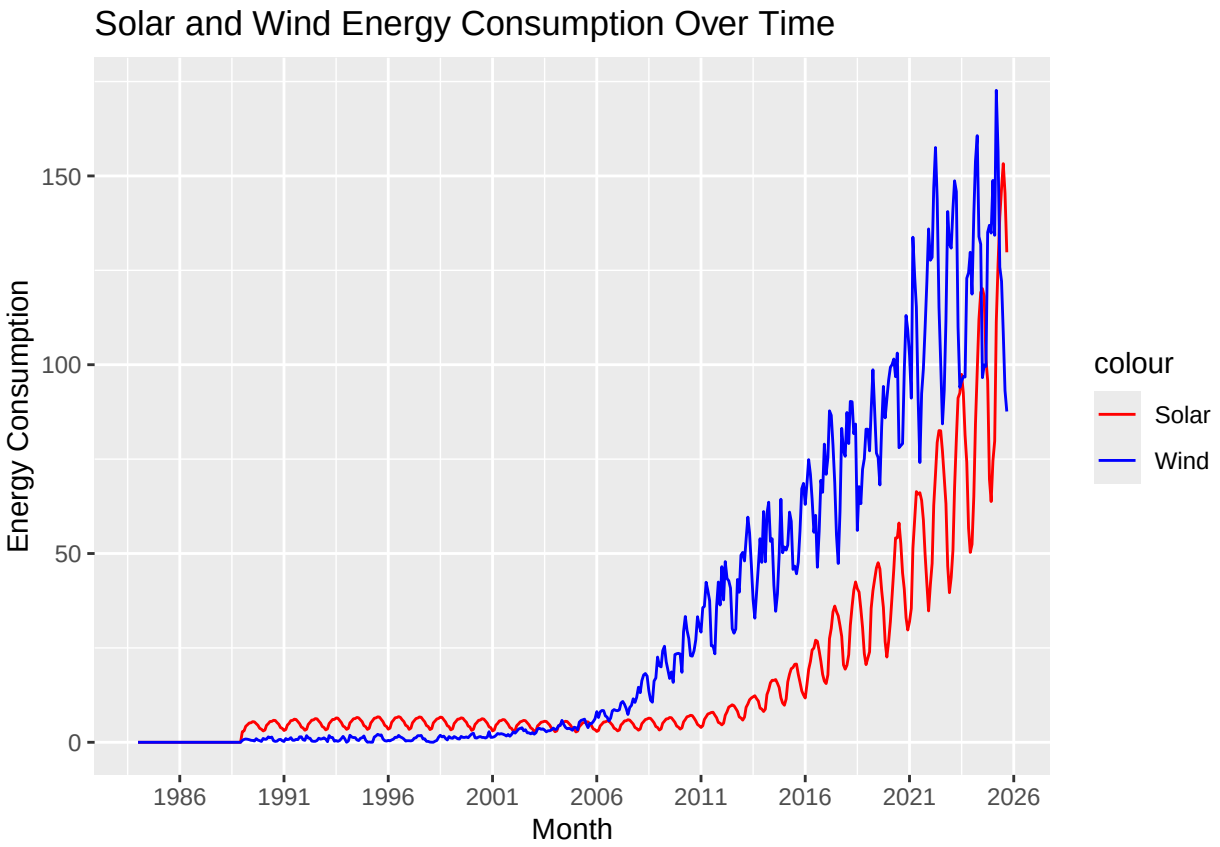

```
ggplot(energy_data1,aes(x = Month, y = `Wind Energy Consumption`)) +
  geom_line() +
  ylab('Wind Energy Consumption') +
  xlab('Date') +
  ggtitle('Wind  Energy Consumption Over Time') +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```



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(energy_data1, aes(x = Month)) +
  geom_line(aes(y=`Solar Energy Consumption`, color = 'Solar'))+
  geom_line(aes(y=`Wind Energy Consumption`, color = 'Wind'))+
  scale_color_manual(values = c('red','blue'))+
  ylab('Energy Consumption')+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  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.

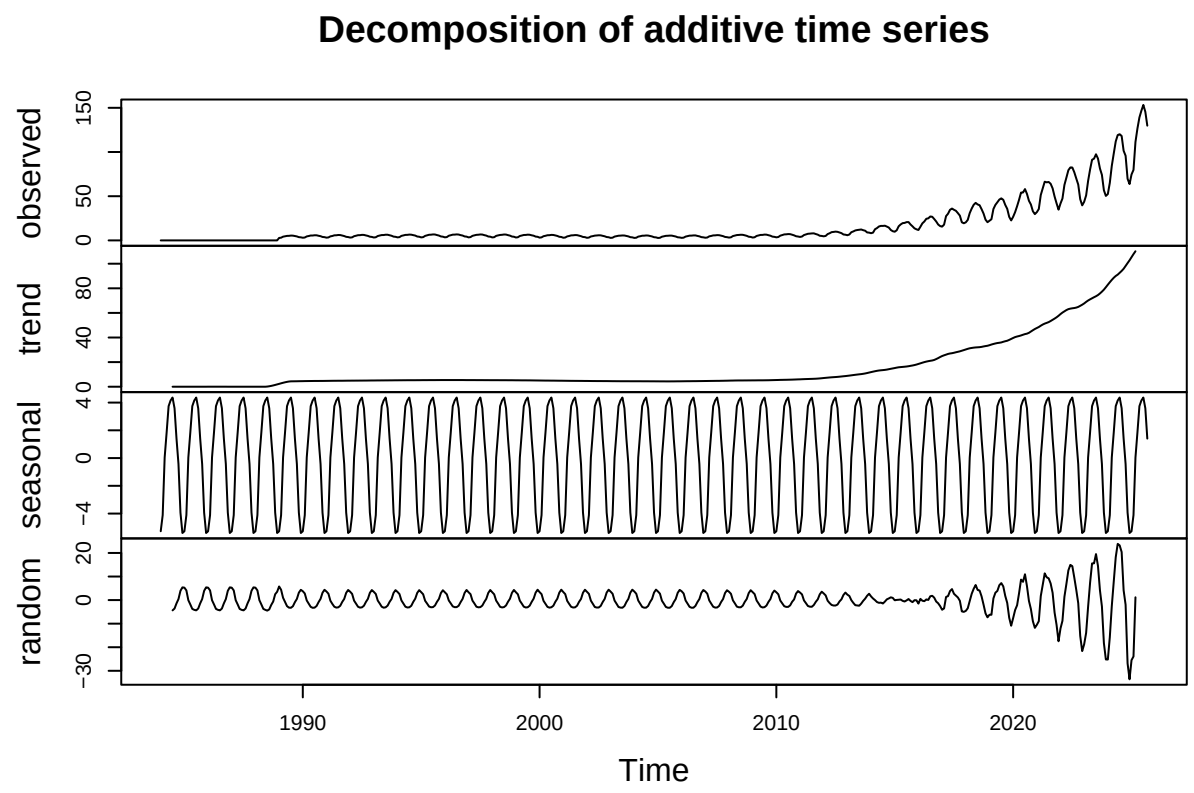
Additional info on `decompose()`.

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.

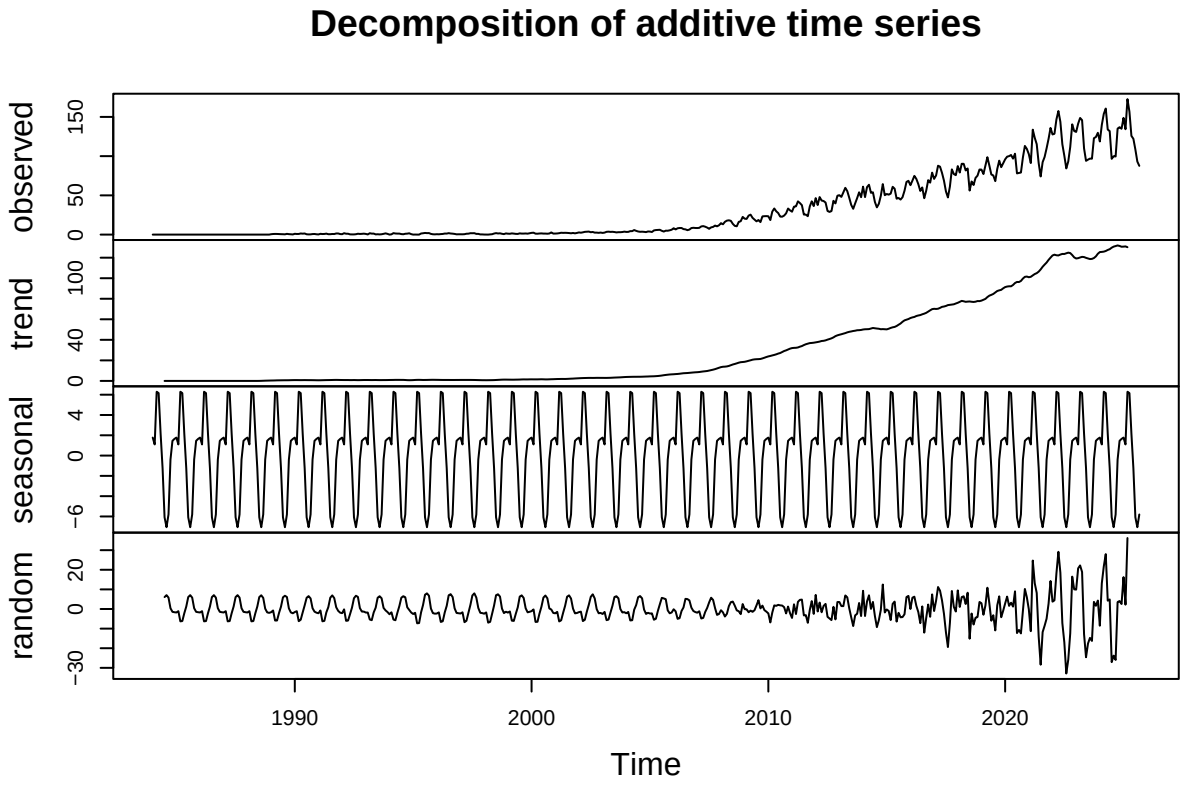
Q4

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?

```r
#turn into ts
solar_ts <- ts(energy_data1$`Solar Energy Consumption`,
               start=c(1984,1),
               frequency = 12)

wind_ts <- ts(energy_data1$`Wind Energy Consumption`,
               start=c(1984,1),
               frequency = 12)

solar_decomp_add <- decompose(solar_ts, type = 'additive')
wind_decomp_add  <- decompose(wind_ts,  type = 'additive')

plot(solar_decomp_add)
```




```r
plot(wind_decomp_add)
```

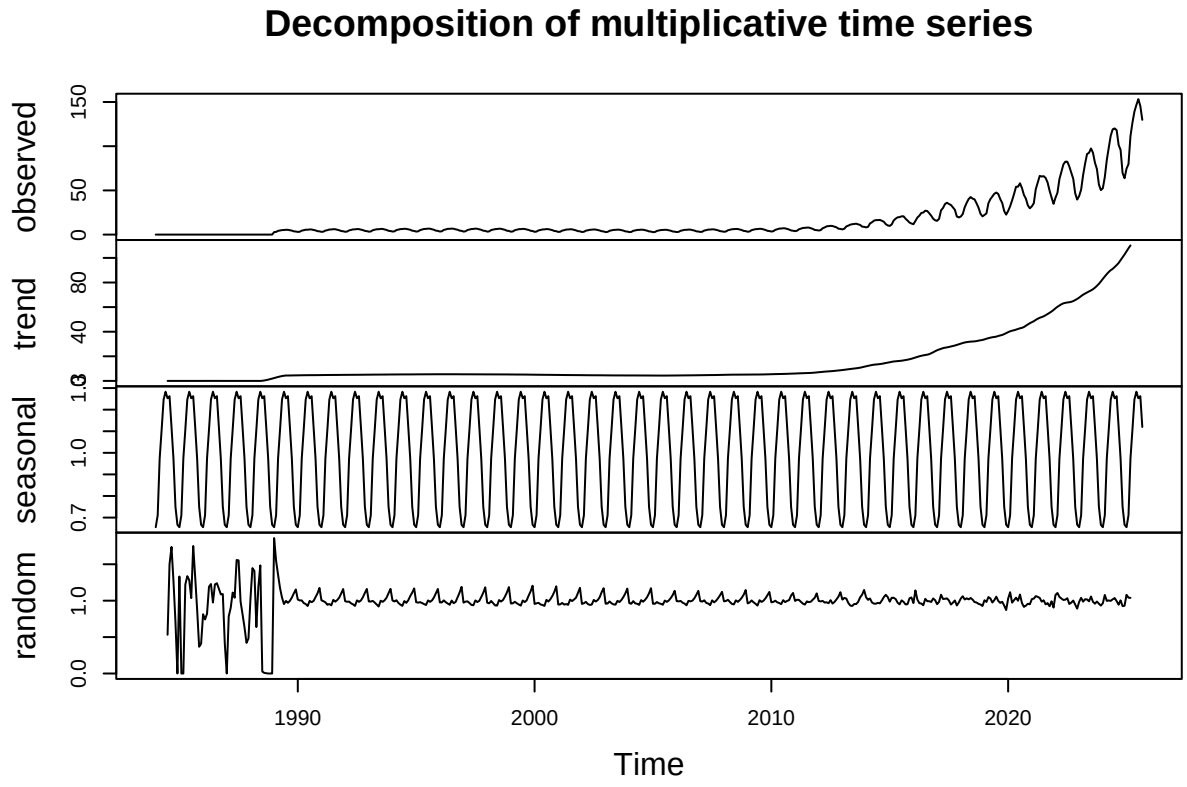


Answer: For both the trend component was non-existent until later in the series. For solar, it picked up in 2015 while wind picked up in 2010. After these time points, ther is a clear strong upward trend that accelerates at these time points. This could mean there were strong technology improvements and policy incentives to push for more consumption. The random component does not look completely random, in both series (pre the 2015 and 2010 event), it looked small and stable since the consumption of both were very minimal. But after these two periods, it turned into larger oscillations with spikes. Though, the wind one looks much more random and jagged while the solar has smooth up and down spikes that are not as random, and are more structured. These showcase the additive may not be a great fit.

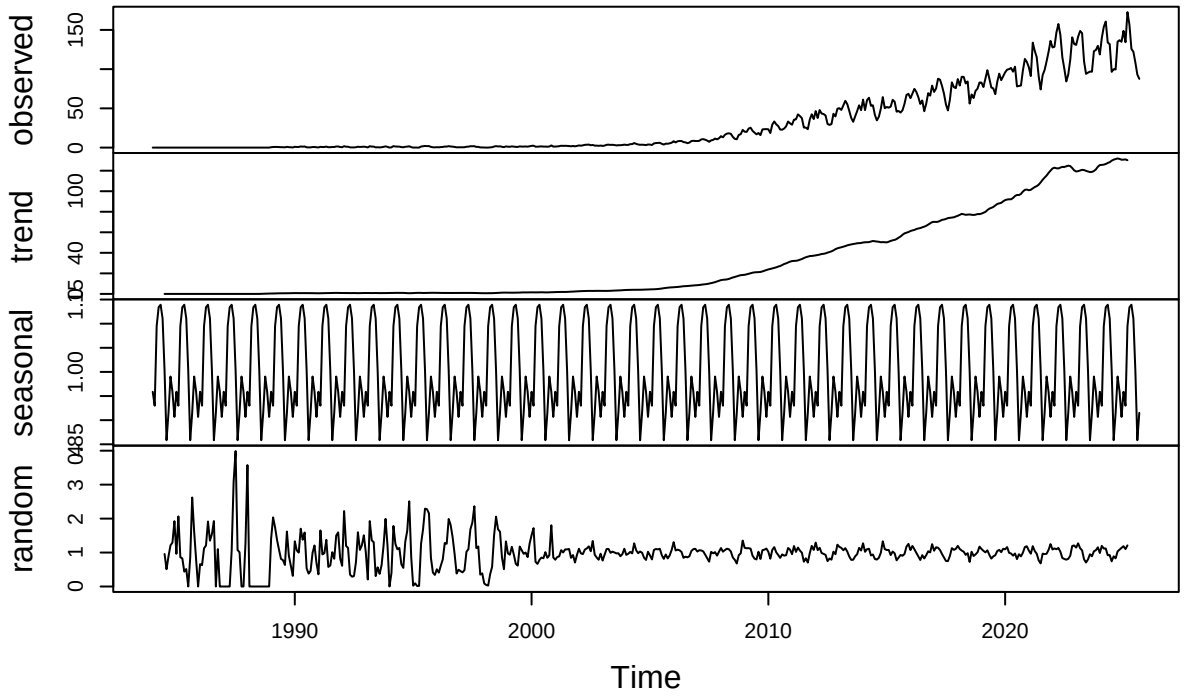
Q5

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?

```
solar_decomp_mult <- decompose(solar_ts, type = 'multiplicative')
wind_decomp_mult <- decompose(wind_ts, type = 'multiplicative')

plot(solar_decomp_mult)
```




```
plot(wind_decomp_mult)
```



>Answer: The random component changed drastically. Once consumption begain for both series the random component relatively smoothed out and became steady with small oscillations. Solar looks much smoother than wind, but both show very small movement and constant steady random component, suggesting the multiplicative is a good fit.

Q6

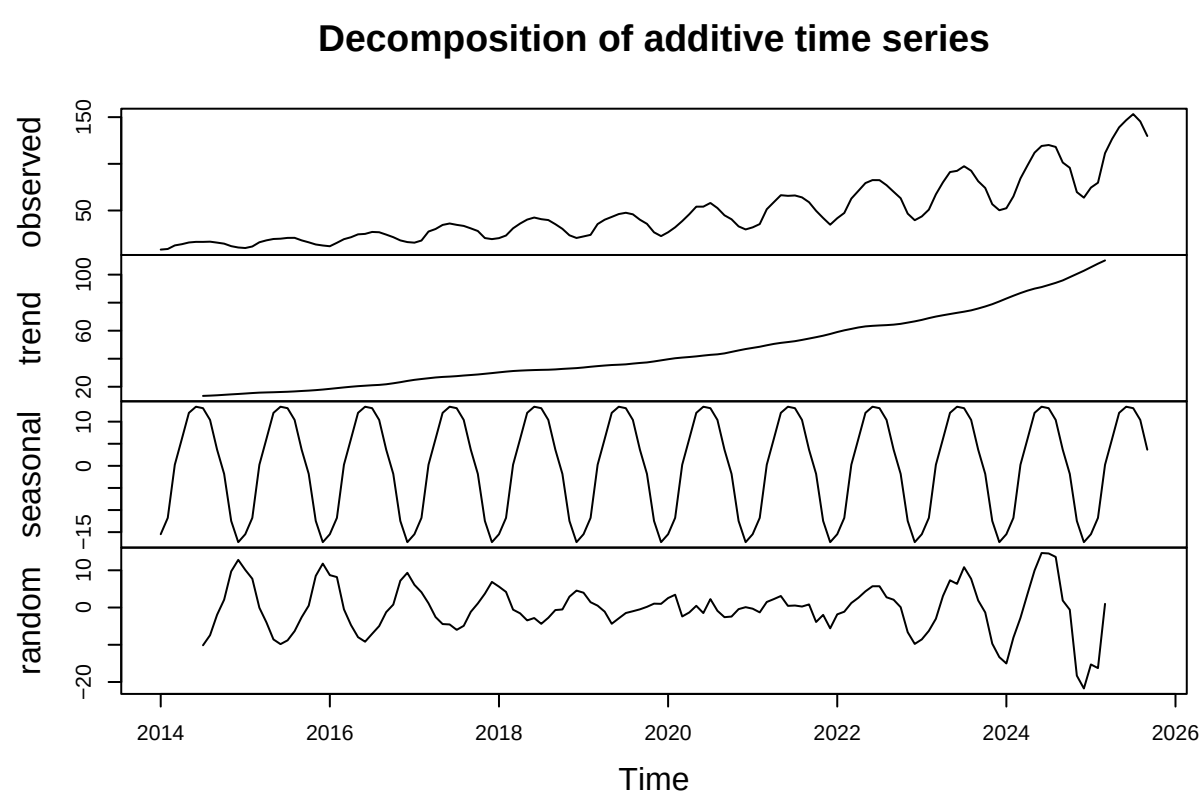
When fitting a model to this data, do you think you need all the historical data? Think about the data from 80s, 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: No, I do not think we need this data since the landscape for solar and wind has changed so drastically since then. There was basically no consumption of either pre the turn of the century. Even the early 20s isn't a good representation of the landscape today since tech improvements and policy incentives have occured making it cheaper and more cost effective. Data from 80s,90s, and early 20s have low and flat consumption that don't represent todays trend.

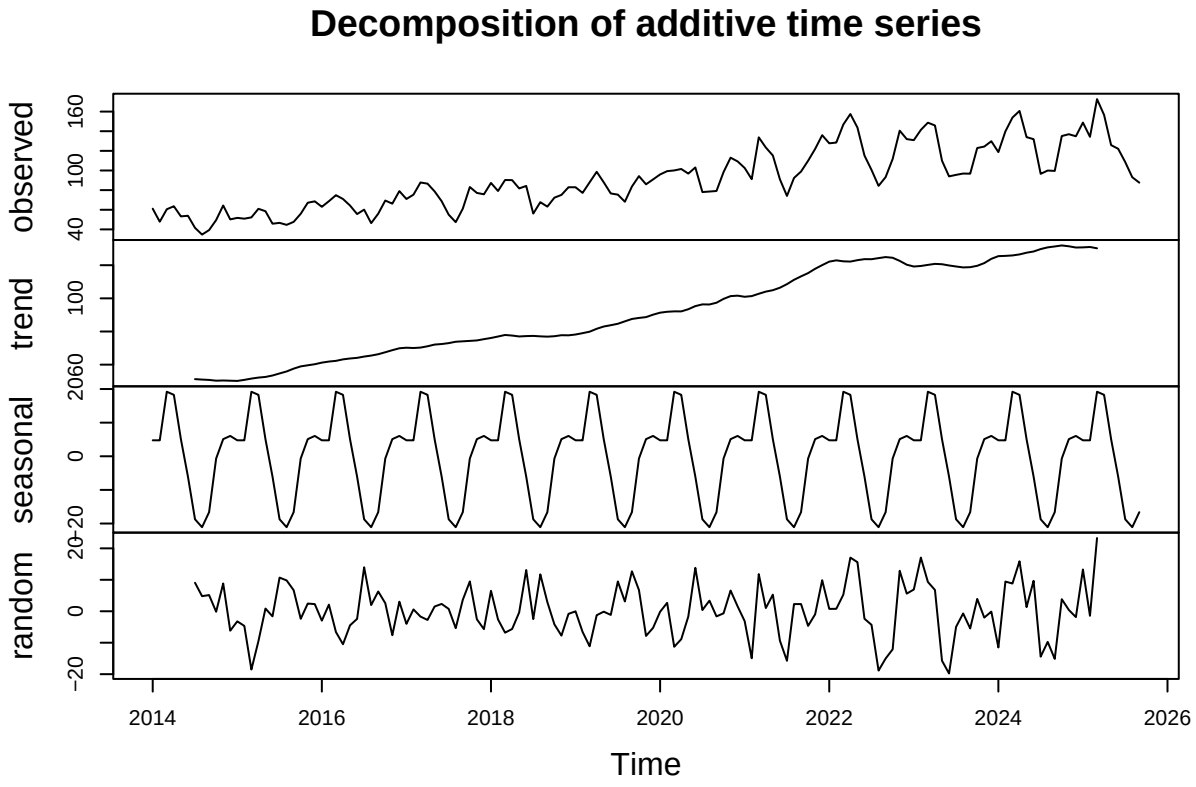
Q7

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
energy_data_2014 <- filter(energy_data1, year(Month) >= 2014)

#ts
solar14_ts <- ts(energy_data_2014$`Solar Energy Consumption`,
                 start=c(2014,1),
                 frequency = 12)

wind14_ts <- ts(energy_data_2014$`Wind Energy Consumption`,
                 start=c(2014,1),
                 frequency = 12)

#add
solar14_decomp_add <- decompose(solar14_ts, type = 'additive')
wind14_decomp_add  <- decompose(wind14_ts,  type = 'additive')

plot(solar14_decomp_add)
```




```r
plot(wind14_decomp_add)
```



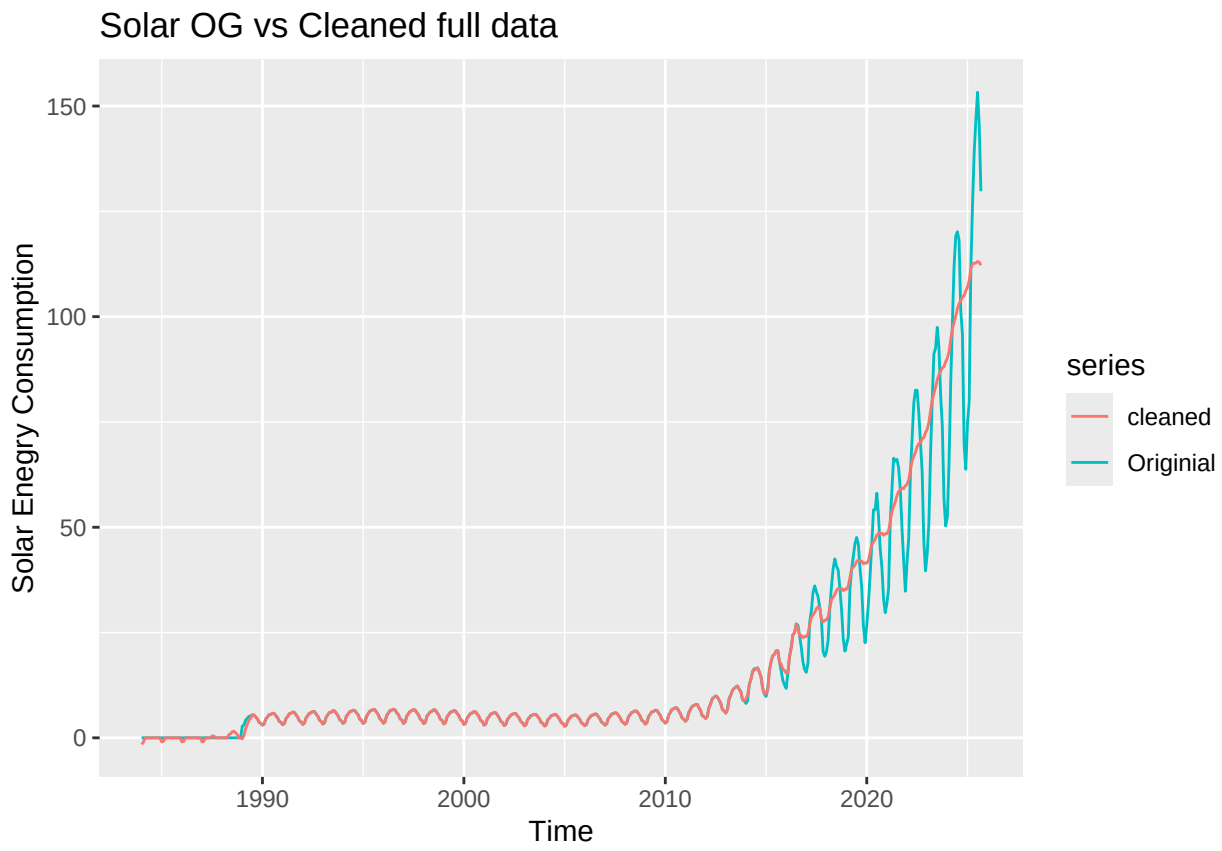
Answer: It looks better than the previous full data series, but is still not fully random, with solar having a spike at the end in 2025 and the smaller oscillating period between 2020 and 2022. There is still structure to this and not completley random. Wind however looks more random than solar with residuals osciallating around zero with a repeating pattern and no intense spikes.

Identify and Remove outliers

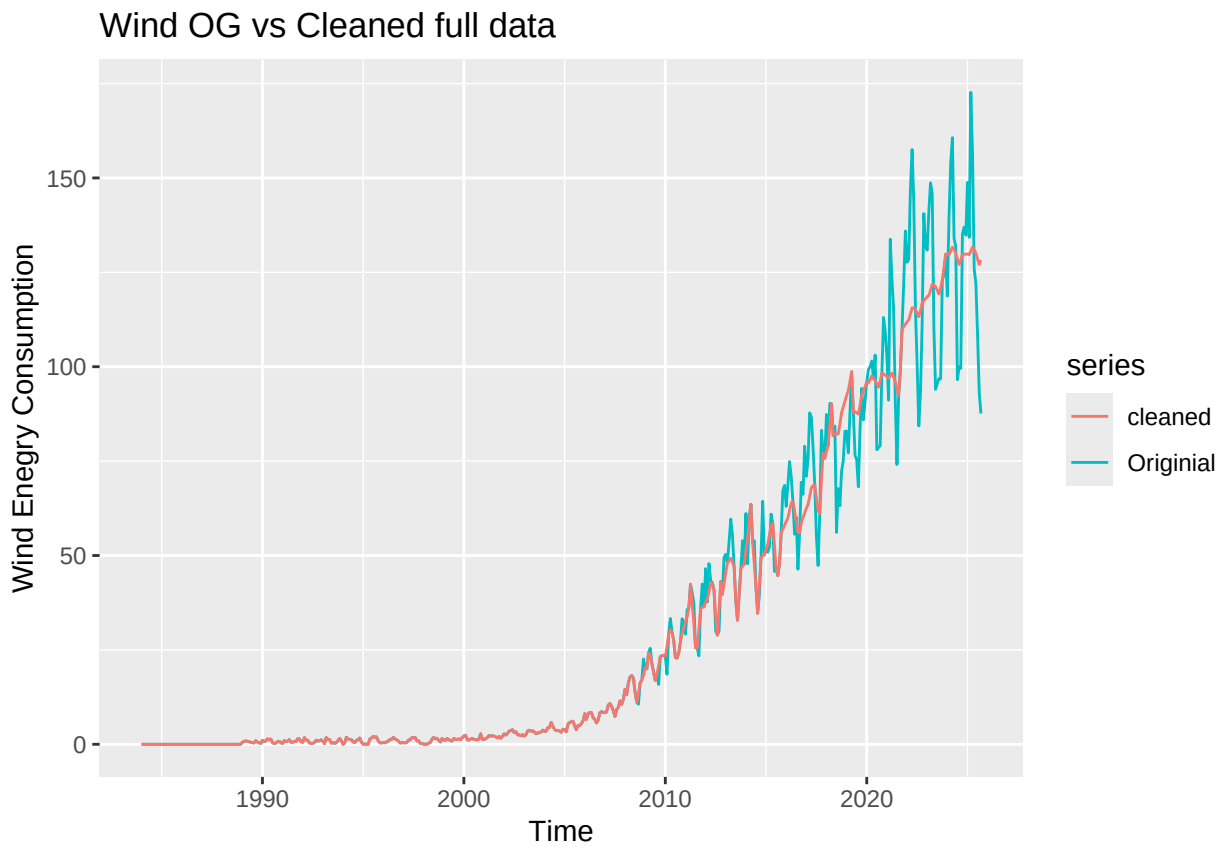
Q8

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.

```
solar_clean <- tsclean(solar_ts)
wind_clean <- tsclean(wind_ts)

autoplot(solar_ts, series = 'Originial') +
  autolayer(solar_clean, series = 'cleaned') +
  ggtitle('Solar OG vs Cleaned full data') +
  ylab('Solar Enegry Consumption')
```




```
autoplot(wind_ts, series = 'Originial') +
  autolayer(wind_clean, series = 'cleaned') +
  ggtitle('Wind OG vs Cleaned full data') +
  ylab('Wind Enegry Consumption')
```

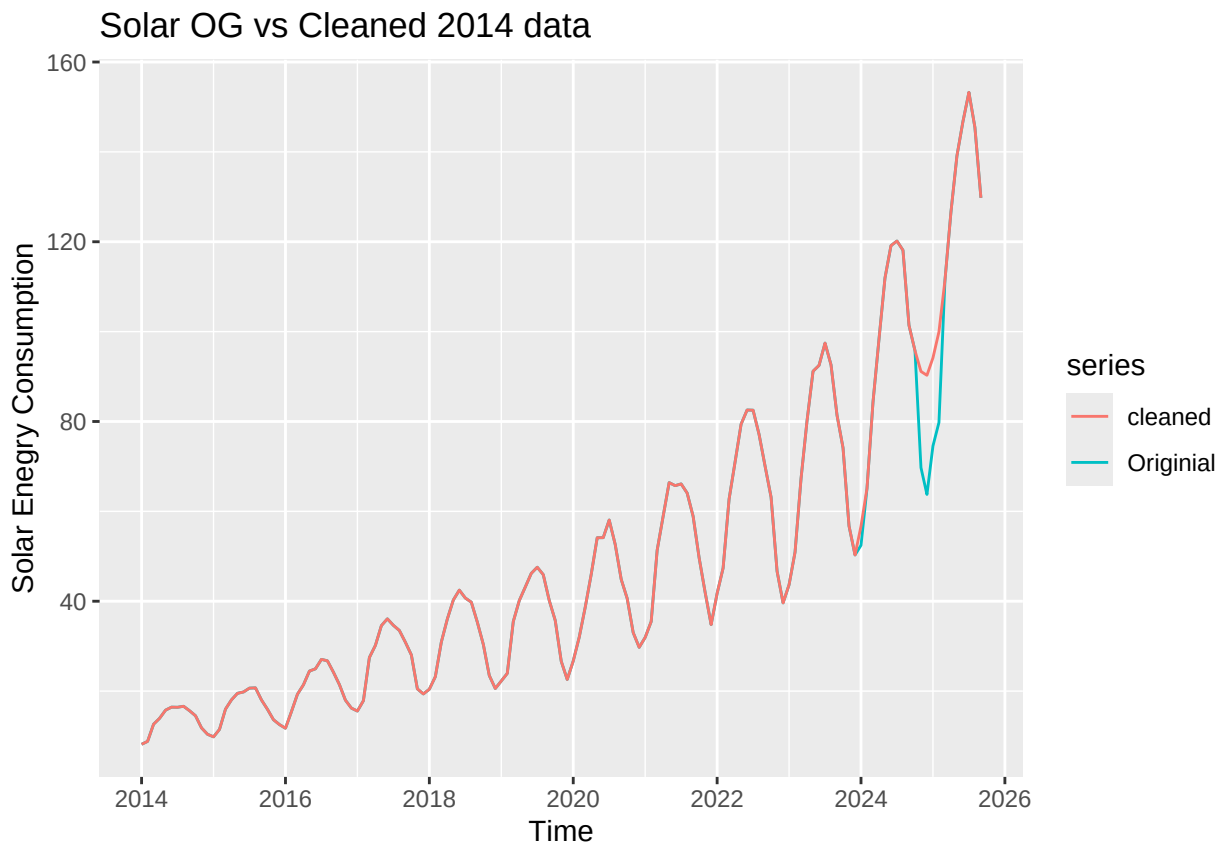


Answer: Yes it removed outliers from the series, you can see it smoothed out the trend for both time series, you can harshly see it removed a lot of varying point after 2010 for Wind to make it a smoother line, as for solar it began a little later around 2015 where the dips and curves began to go further apart.

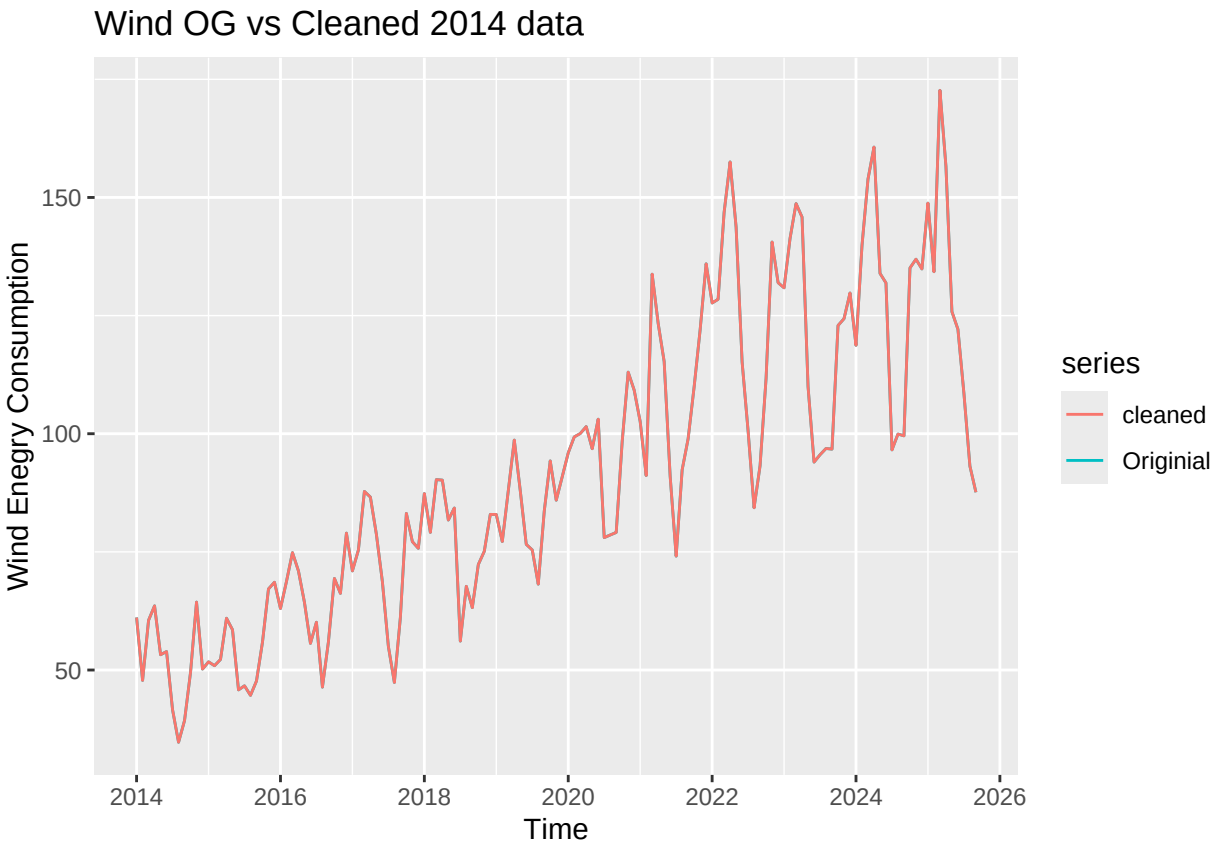
Q9

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?

```
solar14_clean <- tsclean(solar14_ts)
wind14_clean <- tsclean(wind14_ts)

autoplot(solar14_ts, series = 'Originial') +
  autolayer(solar14_clean, series = 'cleaned') +
  ggtitle('Solar OG vs Cleaned 2014 data') +
  ylab('Solar Enegry Consumption')
```




```
autoplot(wind14_ts, series = 'Originial') +
  autolayer(wind14_clean, series = 'cleaned') +
  ggtitle('Wind OG vs Cleaned 2014 data') +
  ylab('Wind Enegry Consumption')
```



Answer: This cleaned data set looks very different, for solar it removed a single outlier dip in 2025 while in wind nothing was removed. So, yes it removed one single outlier from solar while it removed nothing from wind.