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

Assignment 5 - Due date 02/17/26

Trudy

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="./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`
```

```
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_q1 <- energy_data %>%
  select(Month,
         `Solar Energy Consumption`,
         `Wind Energy Consumption`) %>%
  rename(Date = Month) %>%
  mutate(across(-Date, as.numeric)) %>%
  drop_na()
```

```
## Warning: There were 2 warnings in ‘mutate()‘.
## The first warning was:
## i In argument: ‘across(-Date, as.numeric)‘.
## Caused by warning:
## ! NAs introduced by coercion
## i Run ‘dplyr::last_dplyr_warnings()‘ to see the 1 remaining warning.
```

```
glimpse(energy_q1)
```

```
## Rows: 501
## Columns: 3
## $ Date                          <dttm> 1984-01-01, 1984-02-01, 1984-03-01, 1984-0~
## $ ‘Solar Energy Consumption‘ <dbl> 0.000, 0.000, 0.001, 0.001, 0.002, 0.003, 0~
## $ ‘Wind Energy Consumption‘  <dbl> 0.000, 0.001, 0.001, 0.002, 0.003, 0.002, 0~
```

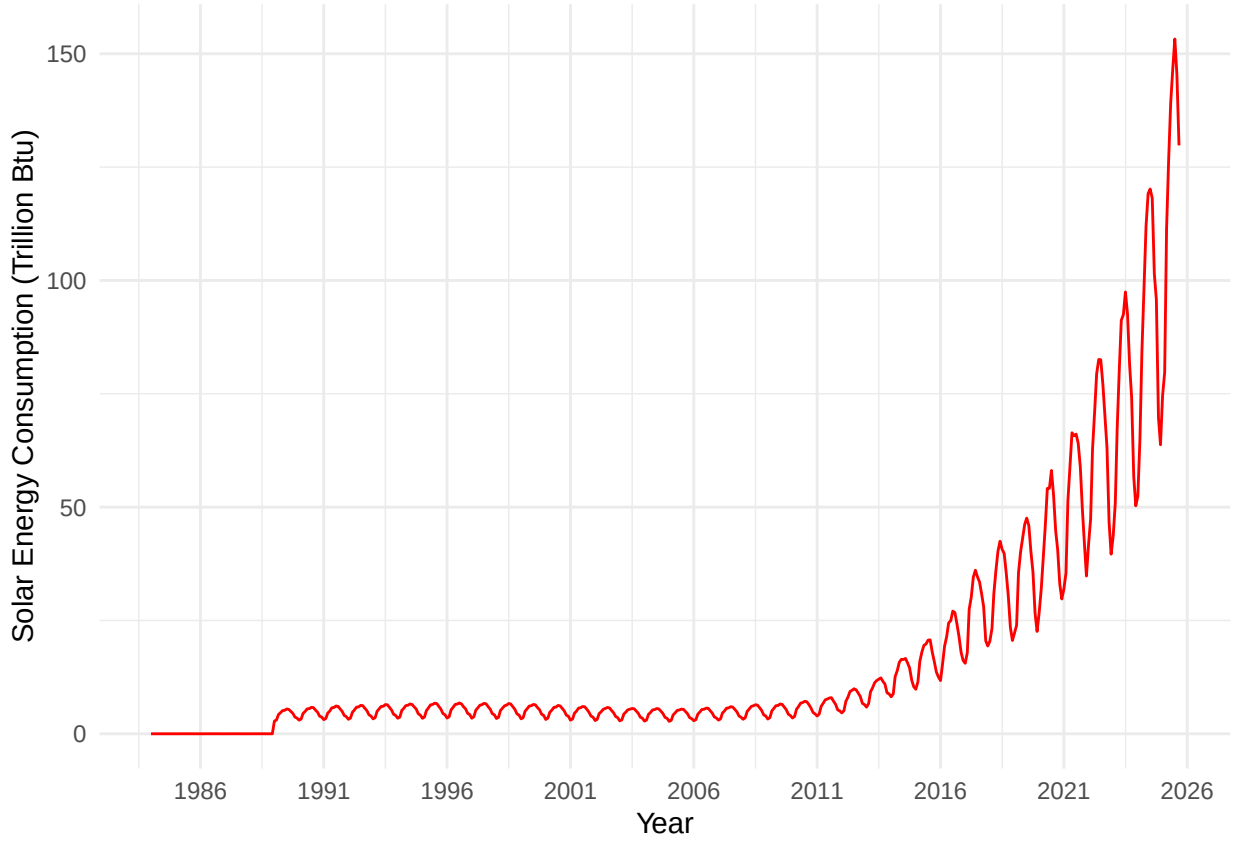
```
summary(energy_q1)
```

```
##       Date                        Solar Energy Consumption
##  Min.   :1984-01-01 00:00:00.00   Min.   :  0.000
##  1st Qu.:1994-06-01 00:00:00.00   1st Qu.:  3.924
##  Median :2004-11-01 00:00:00.00   Median :  5.658
##  Mean   :2004-10-31 01:26:13.65   Mean   : 16.623
##  3rd Qu.:2015-04-01 00:00:00.00   3rd Qu.: 15.758
##  Max.   :2025-09-01 00:00:00.00   Max.   :153.256
##  Wind Energy Consumption
##  Min.   :  0.000
##  1st Qu.:  0.844
##  Median :  3.999
##  Mean   : 31.305
##  3rd Qu.: 55.001
##  Max.   :172.670
```

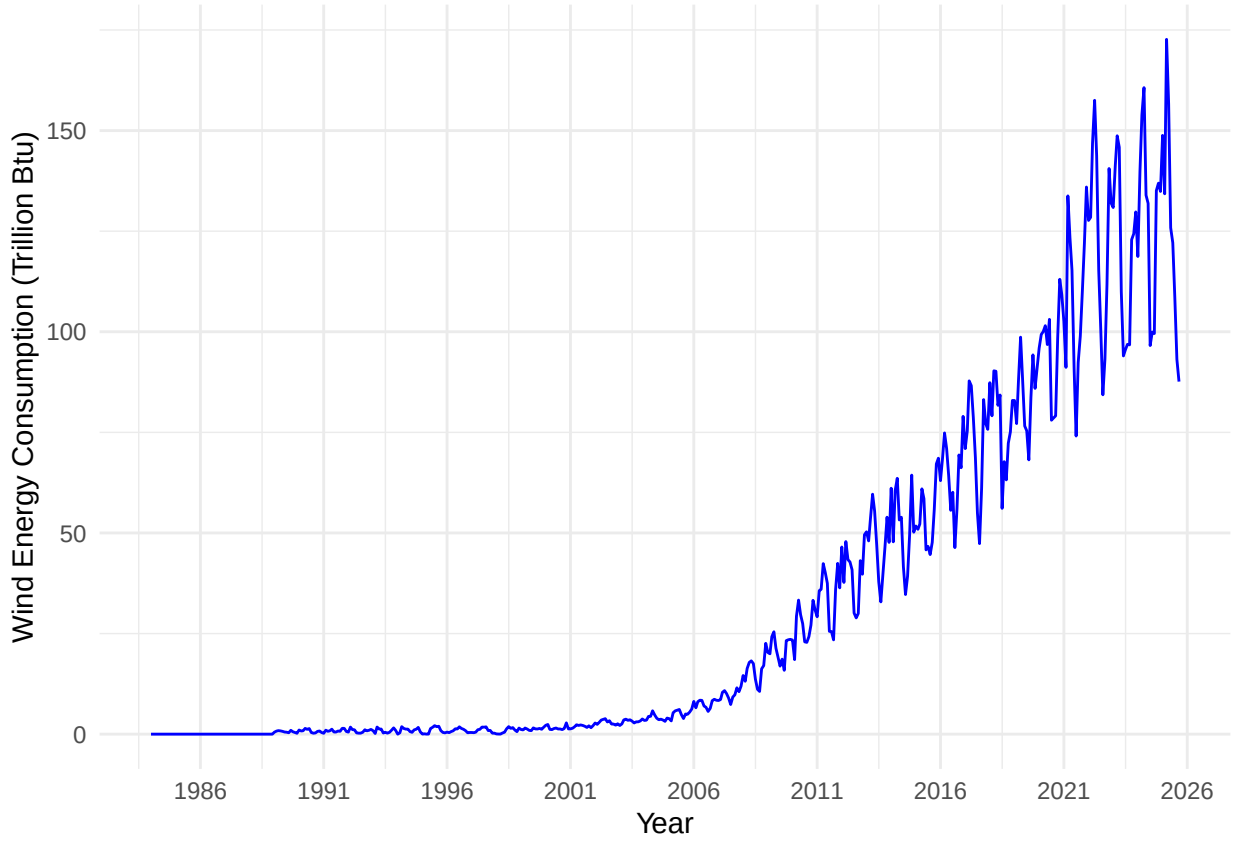
Q2

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
energy_q1 <- energy_q1 %>%
  mutate(Date = as.Date(Date))

#Solar
ggplot(energy_q1, aes(x = Date, y = `Solar Energy Consumption`)) +
  geom_line(color = "red") +
  ylab("Solar Energy Consumption (Trillion Btu)") +
  xlab("Year") +
  scale_x_date(date_breaks = "5 years",
               date_labels = "%Y") +
  theme_minimal()
```



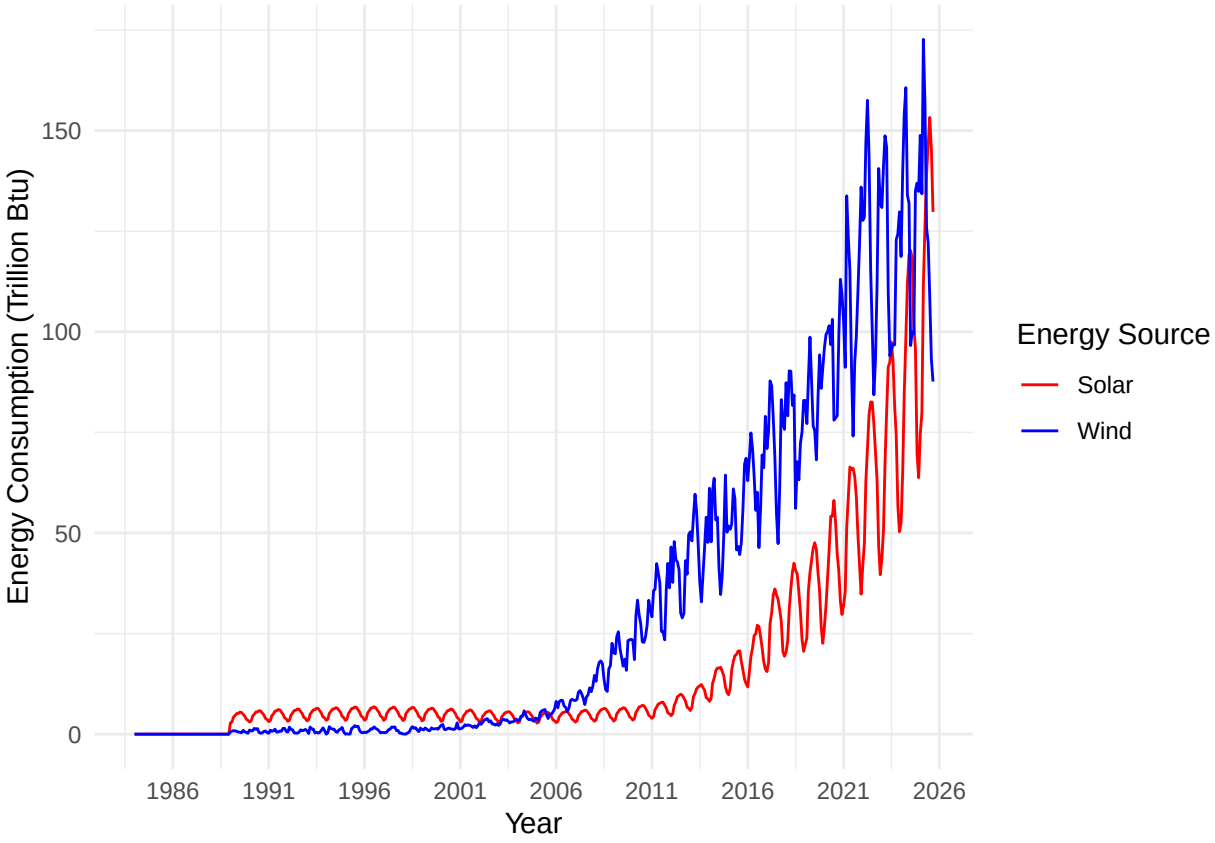

```r
#Wind
ggplot(energy_q1, aes(x = Date, y = `Wind Energy Consumption`)) +
  geom_line(color = "blue") +
  ylab("Wind Energy Consumption (Trillion Btu)") +
  xlab("Year") +
  scale_x_date(date_breaks = "5 years",
               date_labels = "%Y") +
  theme_minimal()
```



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.

```
ggplot(energy_q1, aes(x = Date)) +
  geom_line(aes(y = `Solar Energy Consumption`, color = "Solar")) +
  geom_line(aes(y = `Wind Energy Consumption`, color = "Wind")) +
  scale_color_manual(values = c(
    "Solar" = "red",
    "Wind" = "blue"
  )) +
  labs(color = "Energy Source") +
  ylab("Energy Consumption (Trillion Btu)") +
  xlab("Year") +
  scale_x_date(date_breaks = "5 years",
               date_labels = "%Y") +
  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.

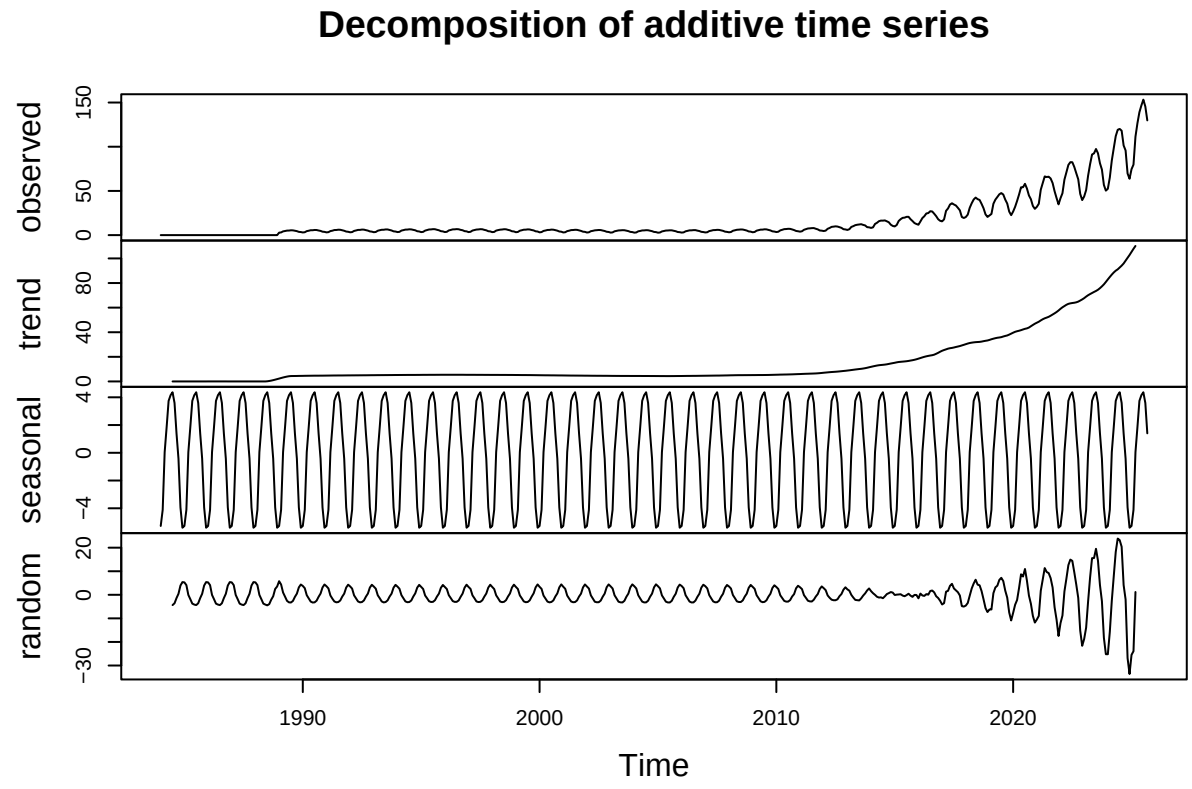
Additional info on `decompose()`.

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.

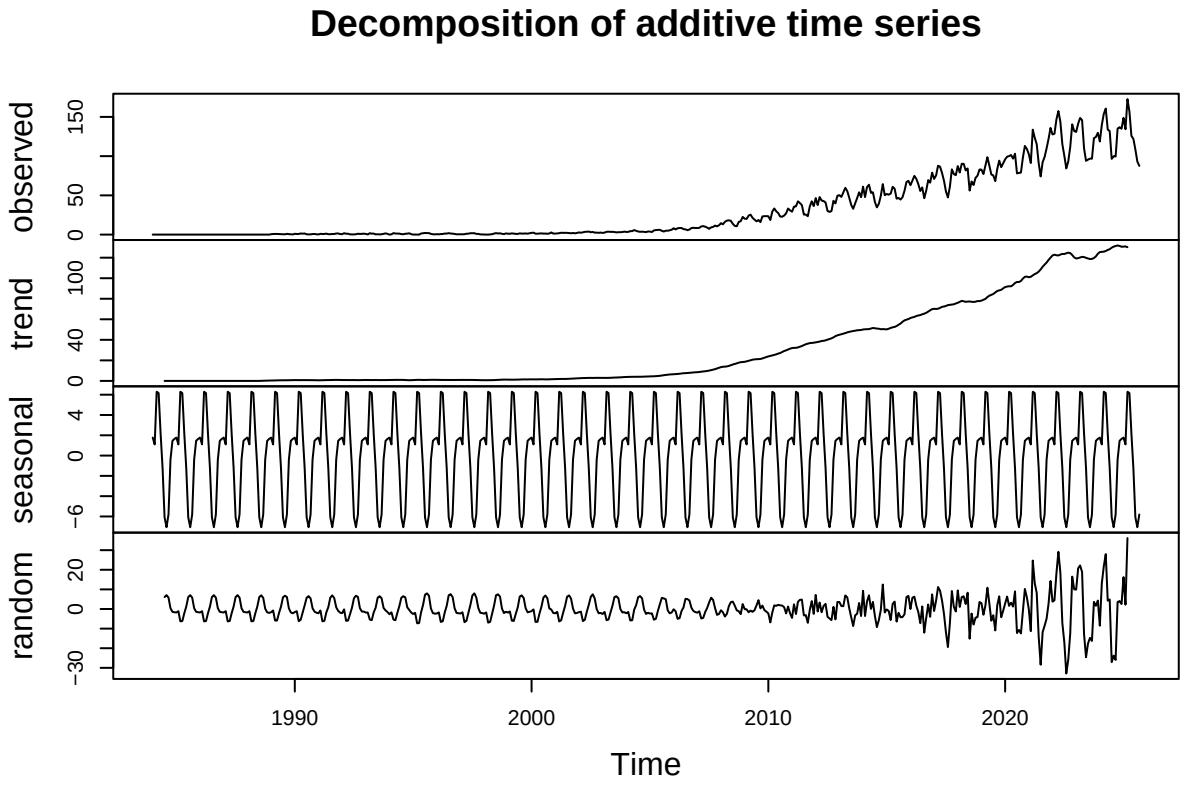
Q4

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
#Solar
solar_ts <- ts(energy_q1$`Solar Energy Consumption`,
               start = c(1984, 1),
               frequency = 12)

solar_decomp <- decompose(solar_ts, type = "additive")
plot(solar_decomp)
```




```r
#Wind
wind_ts <- ts(energy_q1$`Wind Energy Consumption`,
              start = c(1984, 1),
              frequency = 12)

wind_decomp  <- decompose(wind_ts,  type = "additive")
plot(wind_decomp)
```

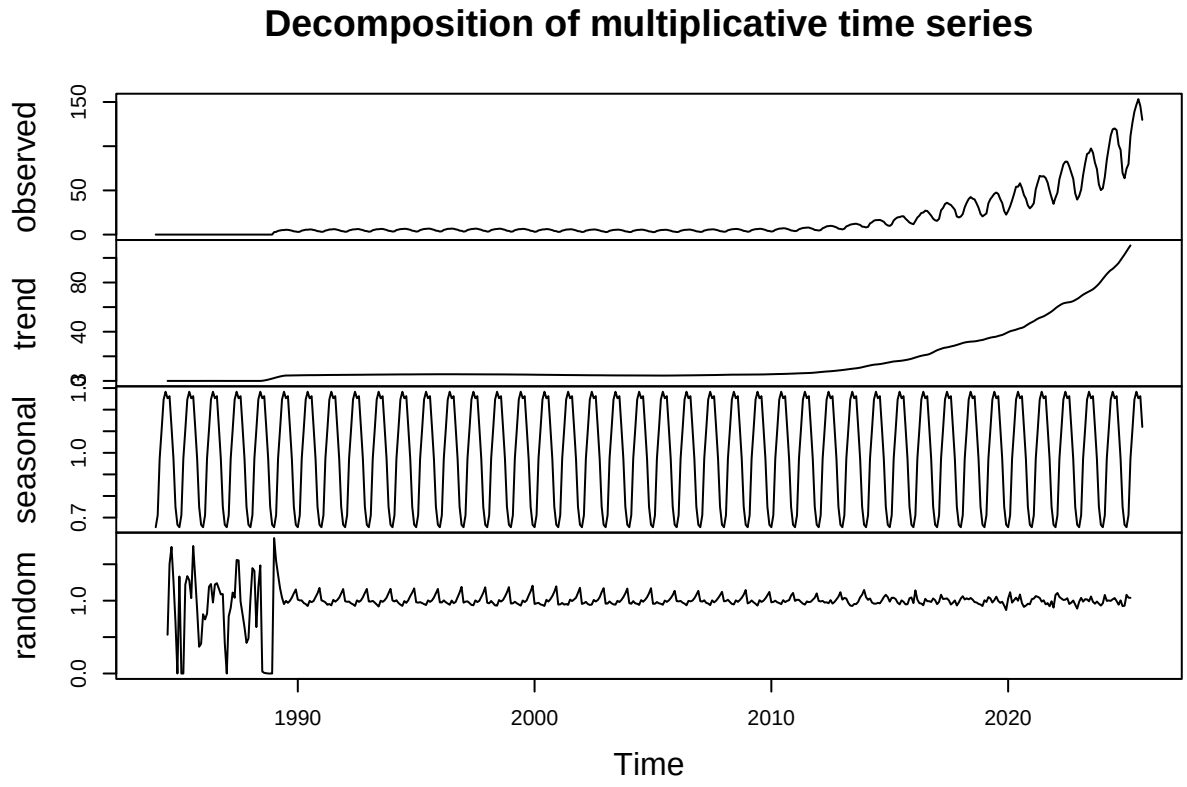


> Answer: For solar, the trend component shows a strong upward increase, especially after 2010, with rapid growth in recent years. The seasonal component displays a clear and stable 12-month cycle. However, the random component does not appear purely random, as its variance increases over time. This suggests that the additive model may not fully capture the structure of the series. For wind, the trend component increases steadily and accelerates after the mid-2000s. The seasonal pattern is regular and consistent across years. However, the random component shows increasing variability in later periods, indicating heteroskedasticity. This suggests that a multiplicative model may be more appropriate.

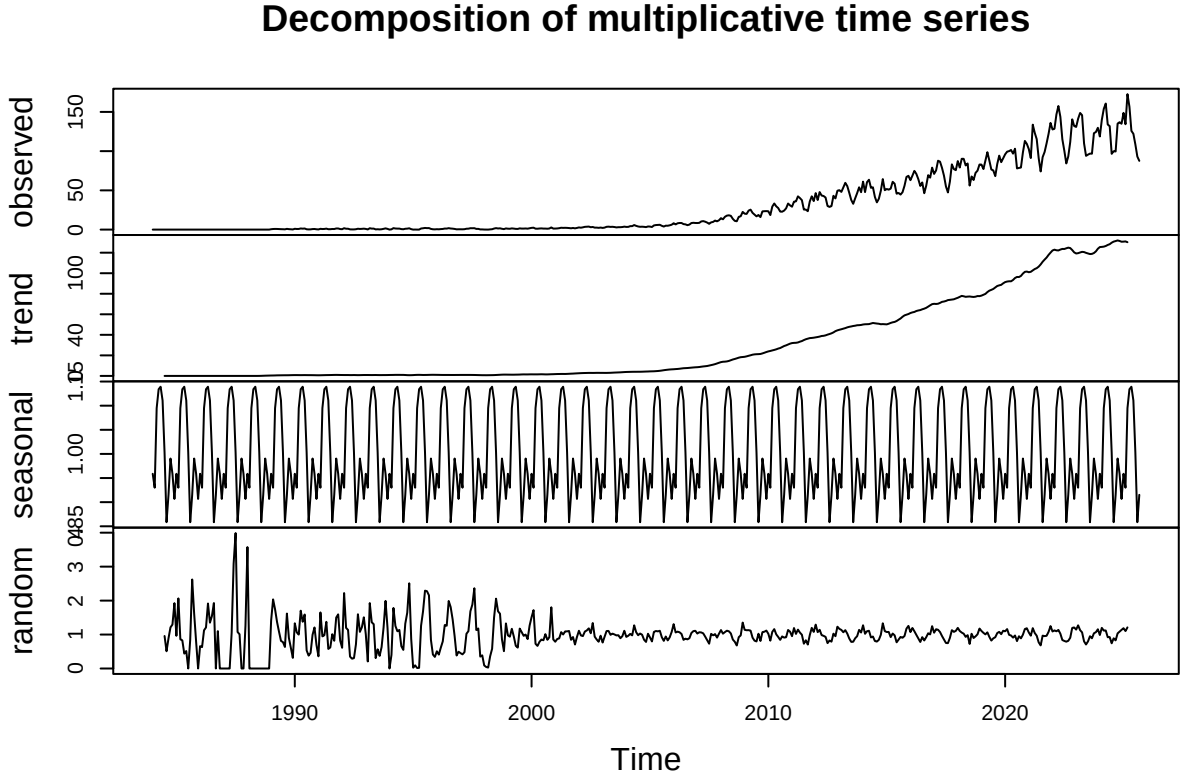
Q5

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
solar_decomp_mult <- decompose(solar_ts, type = "multiplicative")
plot(solar_decomp_mult)
```




```
#Wind
wind_decomp_mult  <- decompose(wind_ts,  type = "multiplicative")
plot(wind_decomp_mult)
```



> Answer: Under the multiplicative decomposition, the random component becomes more stable for both solar and wind series. The increasing variance observed under the additive model is largely reduced. The residual fluctuations now appear more constant over time and closer to white noise. This indicates that the multiplicative model provides a better fit, as seasonal variations increase proportionally with the level of the series.

Q6

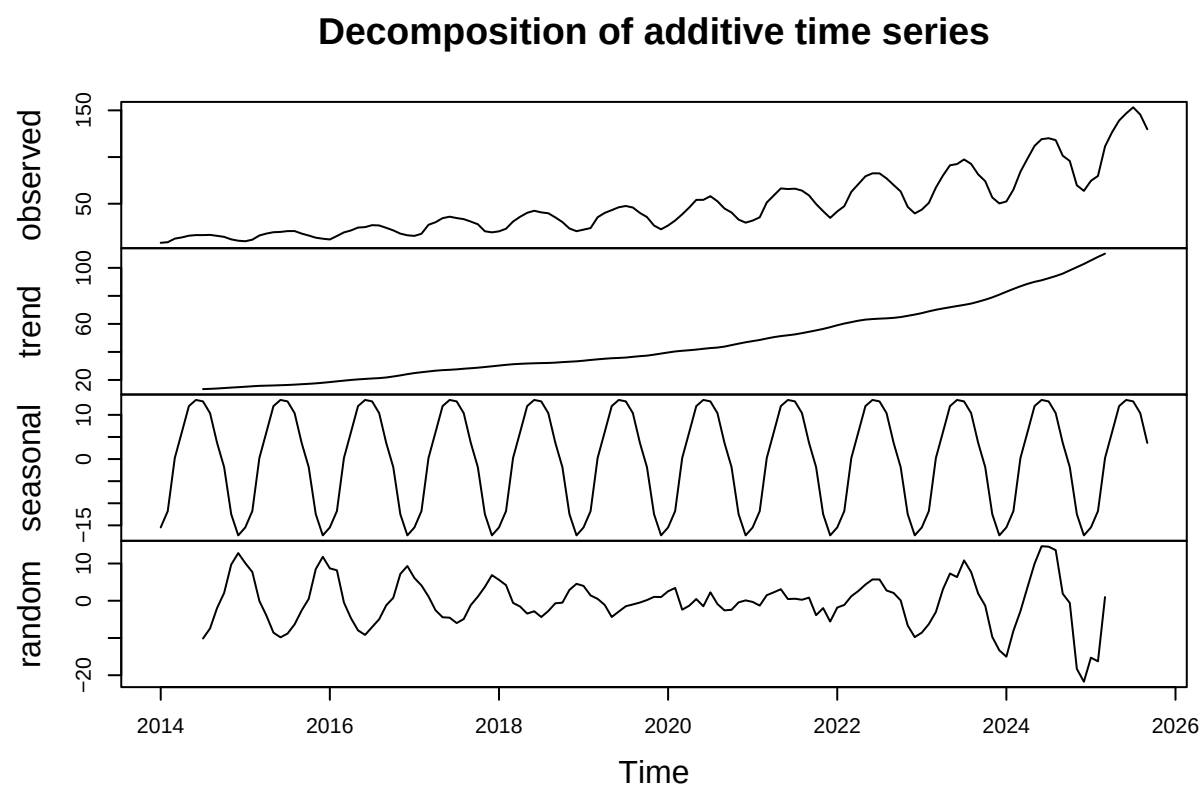
When fitting a model to this data, do you think you need all the historical data? Think about the data from 80s, 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: For forecasting the next six months, we likely do not need to use all historical data from the 1980s and 1990s. Those early periods reflect very low levels and a different growth regime, which may not be relevant for short-term forecasting today. What is most important for near-term prediction is the recent trend and the seasonal pattern. As long as we include enough recent data to estimate seasonality reliably, using a more recent window (such as the last several years) is generally sufficient and may even improve model performance by focusing on the current structure of the series.

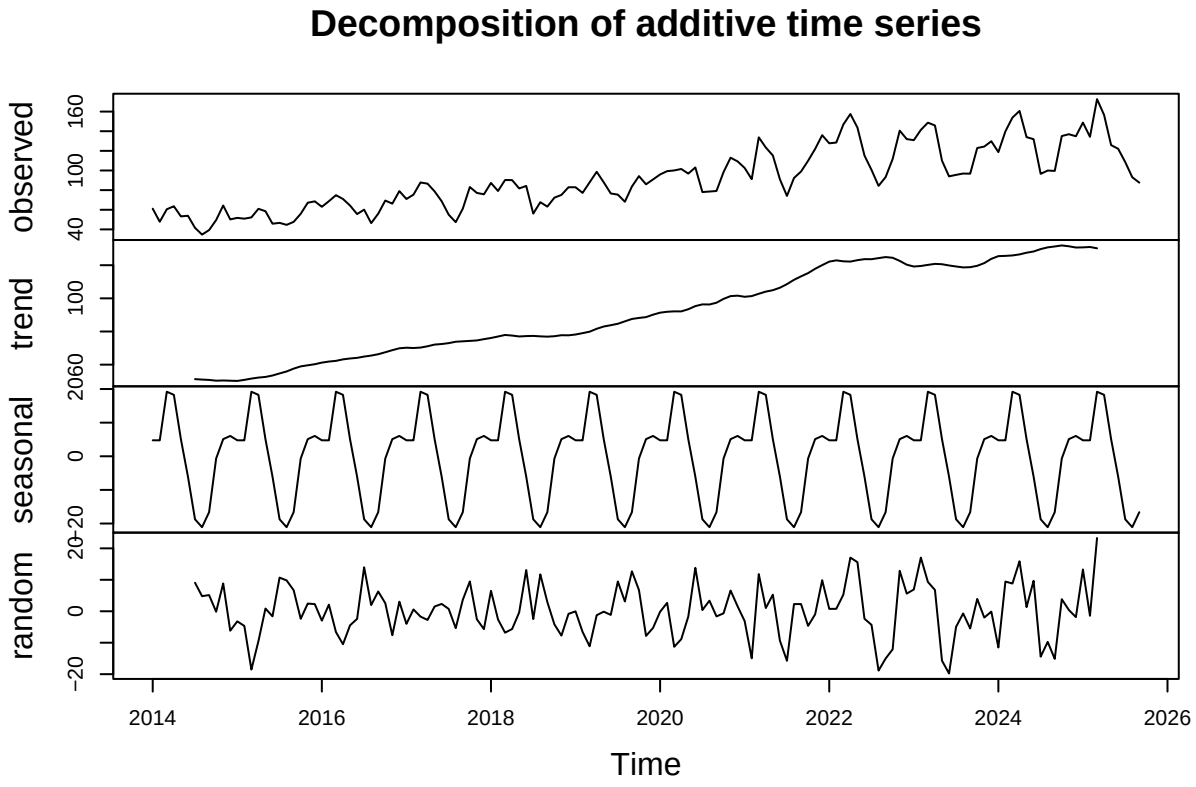
Q7

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_recent <- energy_q1 %>%
  filter(year(Date) >= 2014)

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

solar_decomp_recent <- decompose(solar_ts_recent, type = "additive")
plot(solar_decomp_recent)
```




```r
#Wind
wind_ts_recent <- ts(
  energy_recent$`Wind Energy Consumption`,
  start = c(2014, 1),
  frequency = 12
)

wind_decomp_recent  <- decompose(wind_ts_recent,  type = "additive")
plot(wind_decomp_recent)
```



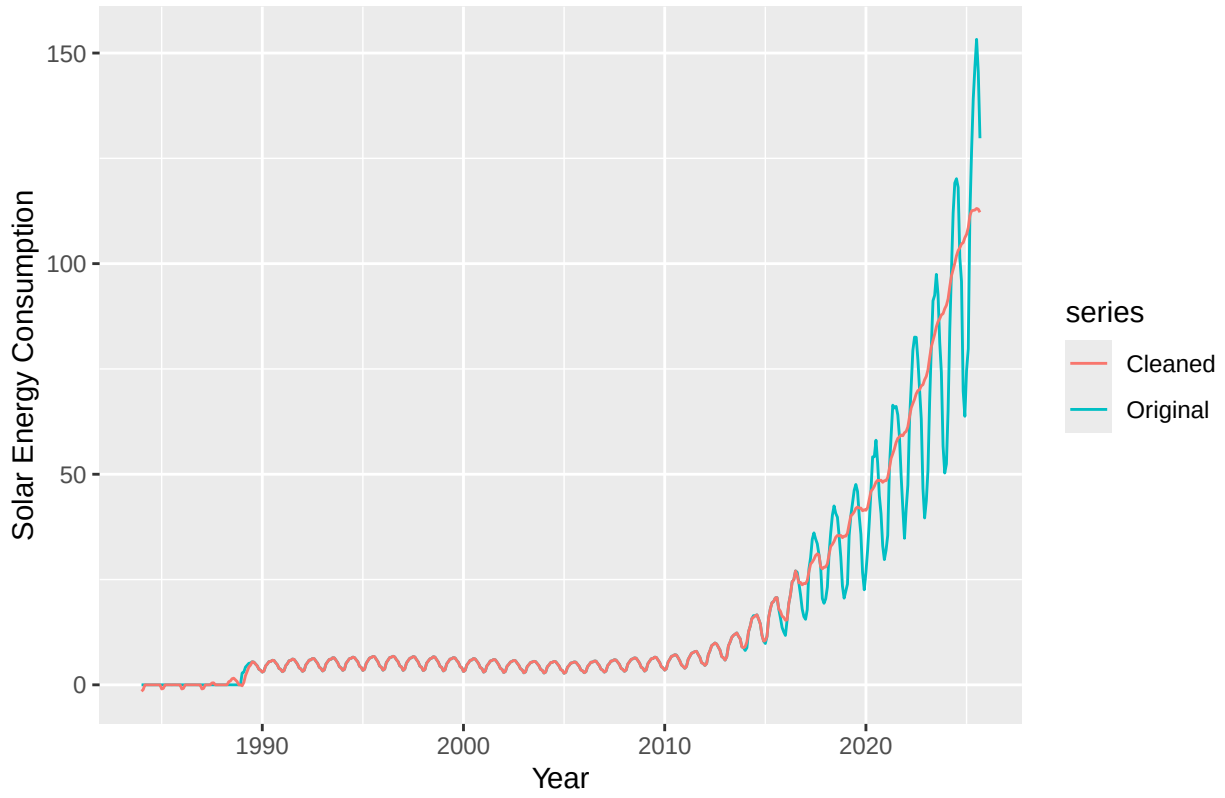
Answer: After restricting the data to start in 2014, the additive decomposition appears more stable for both solar and wind series. The random components show less heteroskedasticity compared to the full-sample decomposition and fluctuate more evenly around zero. Although some variability remains, the residuals appear more random and less structured than before. This suggests that removing the earlier regime shifts improves the fit of the additive model for the recent period.

Identify and Remove outliers

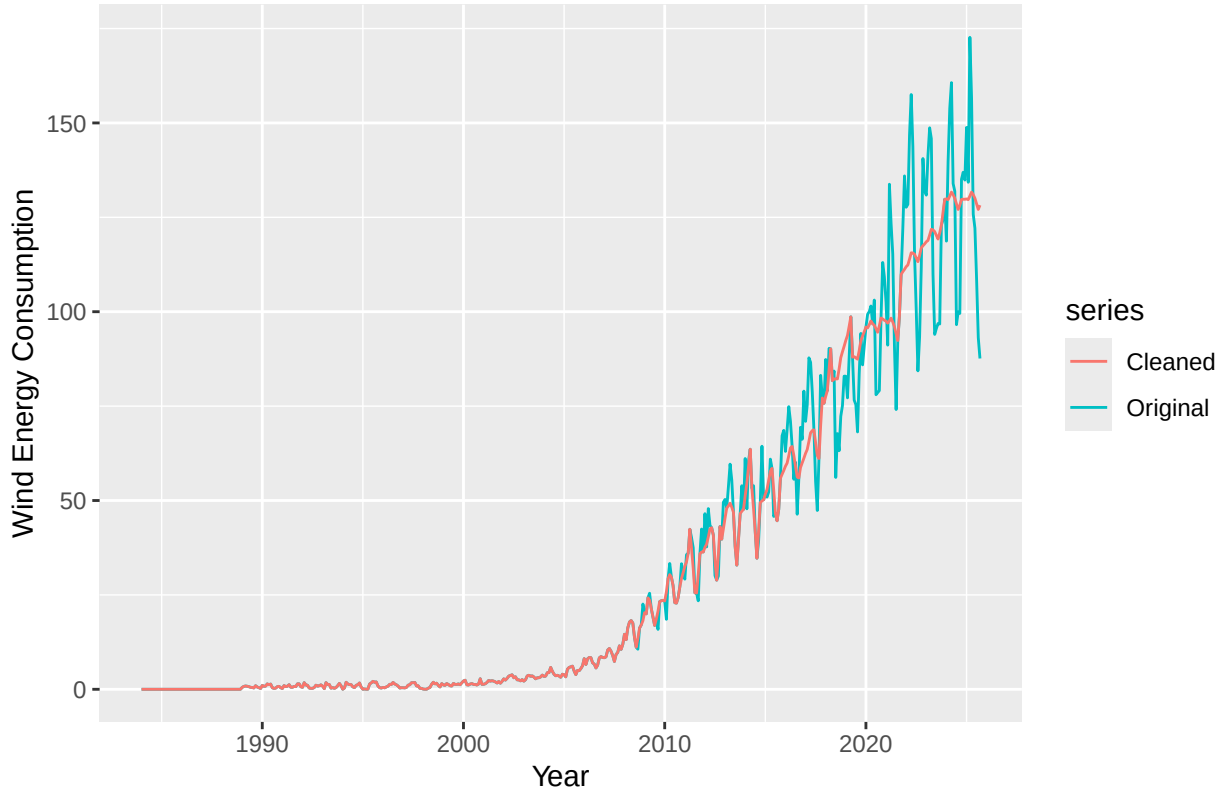
Q8

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.

```
#tsclean() on the two ts objects from Q4
solar_clean <- tsclean(solar_ts)
wind_clean  <- tsclean(wind_ts)

#Compare original vs cleaned
autoplot(solar_ts, series = "Original") +
  autolayer(solar_clean, series = "Cleaned") +
  ylab("Solar Energy Consumption") +
  xlab("Year")
```




```
autoplot(wind_ts, series = "Original") +
  autolayer(wind_clean, series = "Cleaned") +
  ylab("Wind Energy Consumption") +
  xlab("Year")
```

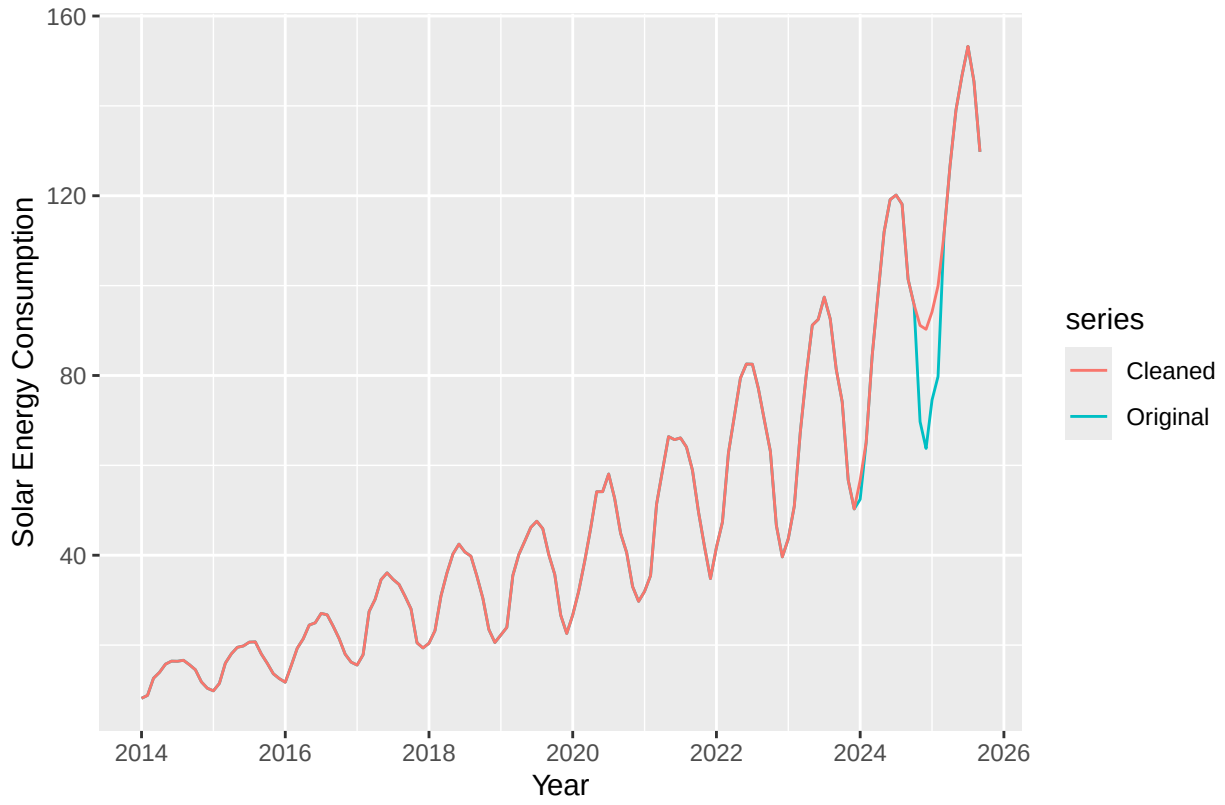


> Answer: The tsclean() function removed some outliers in both series, especially in recent years when values became large. The cleaned series are smoother than the original ones, but the overall trend and seasonality remain unchanged. The effect is more noticeable for wind than for solar.

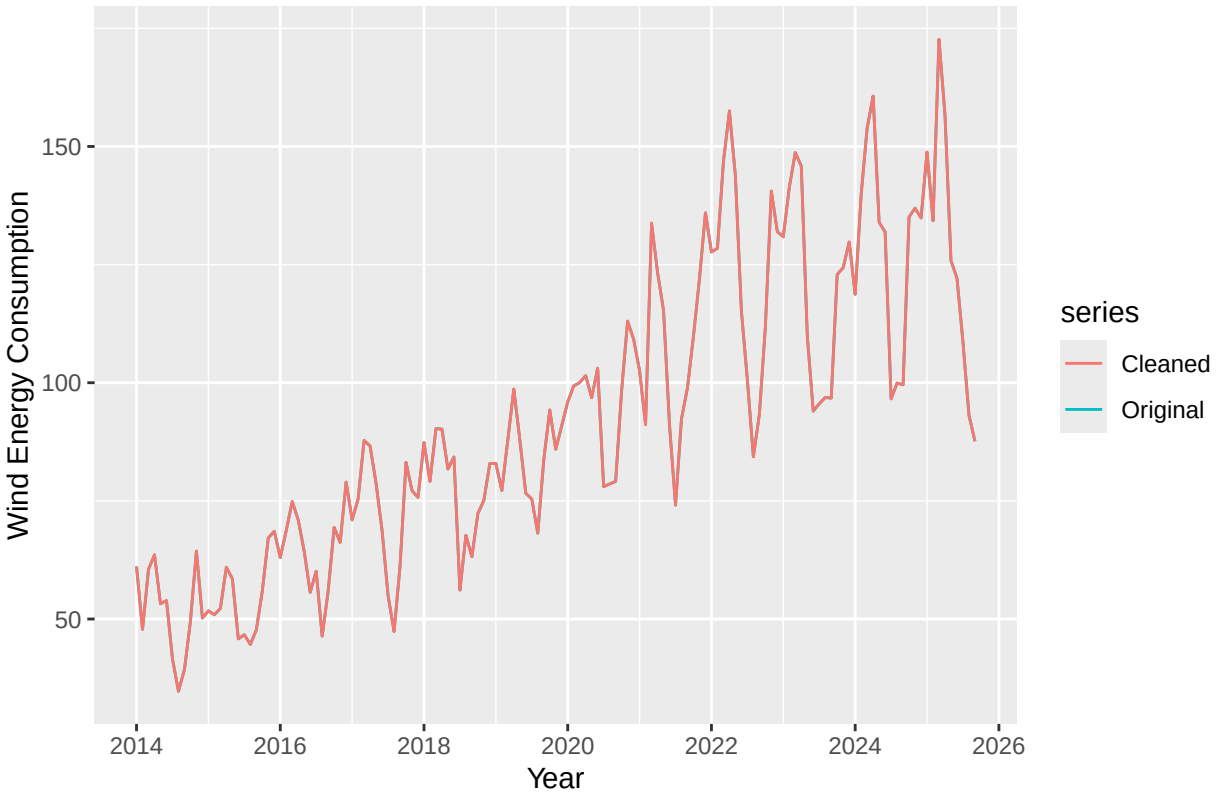
Q9

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?

```
#tsclean() on the 2014 later time series from Q7
solar_clean_recent <- tsclean(solar_ts_recent)
wind_clean_recent  <- tsclean(wind_ts_recent)

#Compare original vs cleaned
autoplot(solar_ts_recent, series = "Original") +
  autolayer(solar_clean_recent, series = "Cleaned") +
  ylab("Solar Energy Consumption") +
  xlab("Year")
```




```
autoplot(wind_ts_recent, series = "Original") +
  autolayer(wind_clean_recent, series = "Cleaned") +
  ylab("Wind Energy Consumption") +
  xlab("Year")
```



Answer: When applying tsclean() to the time series starting in 2014, almost no visible changes occurred for either solar or wind. The cleaned series nearly overlaps with the original series in both cases. Only very small adjustments may have been made to a few extreme observations, but no major outliers were removed. This suggests that after restricting the sample to post-2014 data, the series does not contain strong isolated outliers according to the algorithm.