ENV 790.30 - Time Series Analysis for Energy Data | Spring 2024

Assignment 5 - Due date 02/13/24

Zhenghao Lin

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_Sp23.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
```

```
library(tidyverse)  #load this package so yon clean the data frame using pipes
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr   1.1.3     v stringr 1.5.0
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.2     v tidyr   1.3.0
## v readr   2.1.4

## -- 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
```

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 - using xlsx package
energy_data <- read.table(file="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source

#Date Conversion
ym_date <- ym(energy_data$Month)

#New data frame
energy_data <- cbind(ym_date, energy_data[2:14])

#Now let's extract the column names from row 11 only
read_col_names <- read.table(file="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Sou

colnames(energy_data)
```

```
##  [1] "ym_date"                           "Wood.Energy.Production"
##  [3] "Biofuels.Production"               "Total.Biomass.Energy.Production"
##  [5] "Total.Renewable.Energy.Production" "Hydroelectric.Power.Consumption"
##  [7] "Geothermal.Energy.Consumption"     "Solar.Energy.Consumption"
##  [9] "Wind.Energy.Consumption"           "Wood.Energy.Consumption"
## [11] "Waste.Energy.Consumption"          "Biofuels.Consumption"
## [13] "Total.Biomass.Energy.Consumption"  "Total.Renewable.Energy.Consumption"
```

```r
nobs=nrow(energy_data)
nvar=ncol(energy_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!

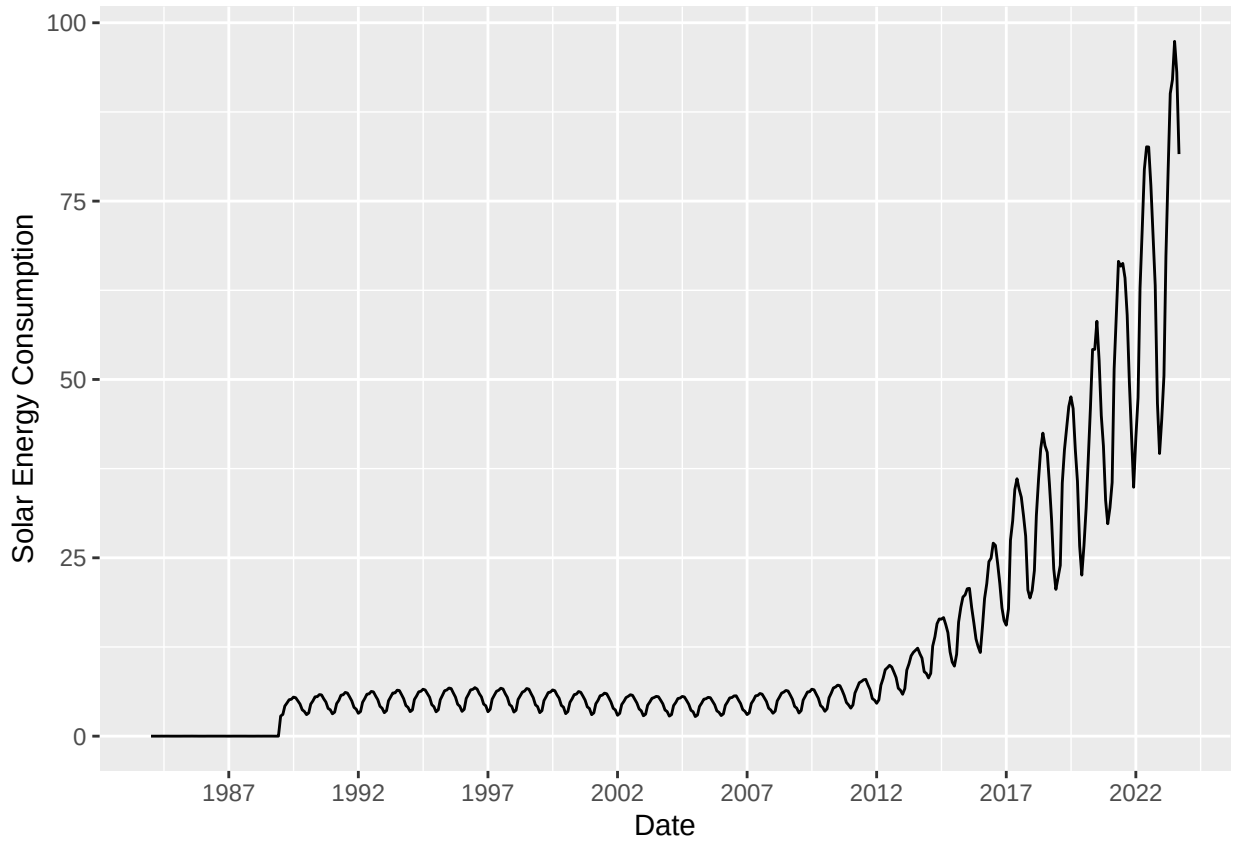
```r
# Filtering and cleaning the data
clean_energy_data <- energy_data %>%
  select(Date = ym_date, Solar = Solar.Energy.Consumption, Wind = Wind.Energy.Consumption) %>%
  filter(Solar != "Not Available", Wind != "Not Available") %>%
  mutate(Solar = as.numeric(as.character(Solar)), Wind = as.numeric(as.character(Wind)))


head(clean_energy_data)
```

```
##         Date Solar  Wind
## 1 1984-01-01 0.000 0.000
## 2 1984-02-01 0.000 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
```

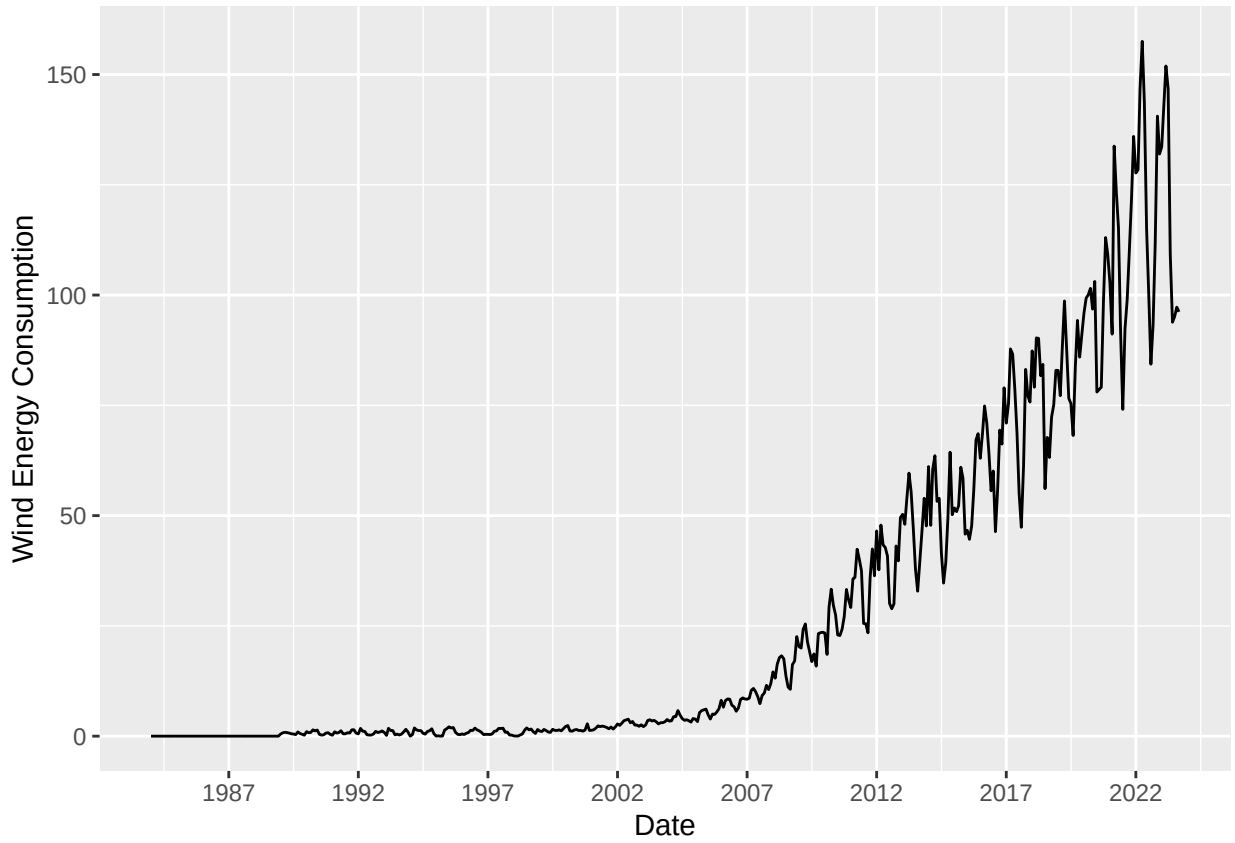
Q2

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")")`

```r
# Plotting Solar Energy Consumption
ggplot(clean_energy_data, aes(x = Date, y = Solar)) +
  geom_line() +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ylab("Solar Energy Consumption")
```



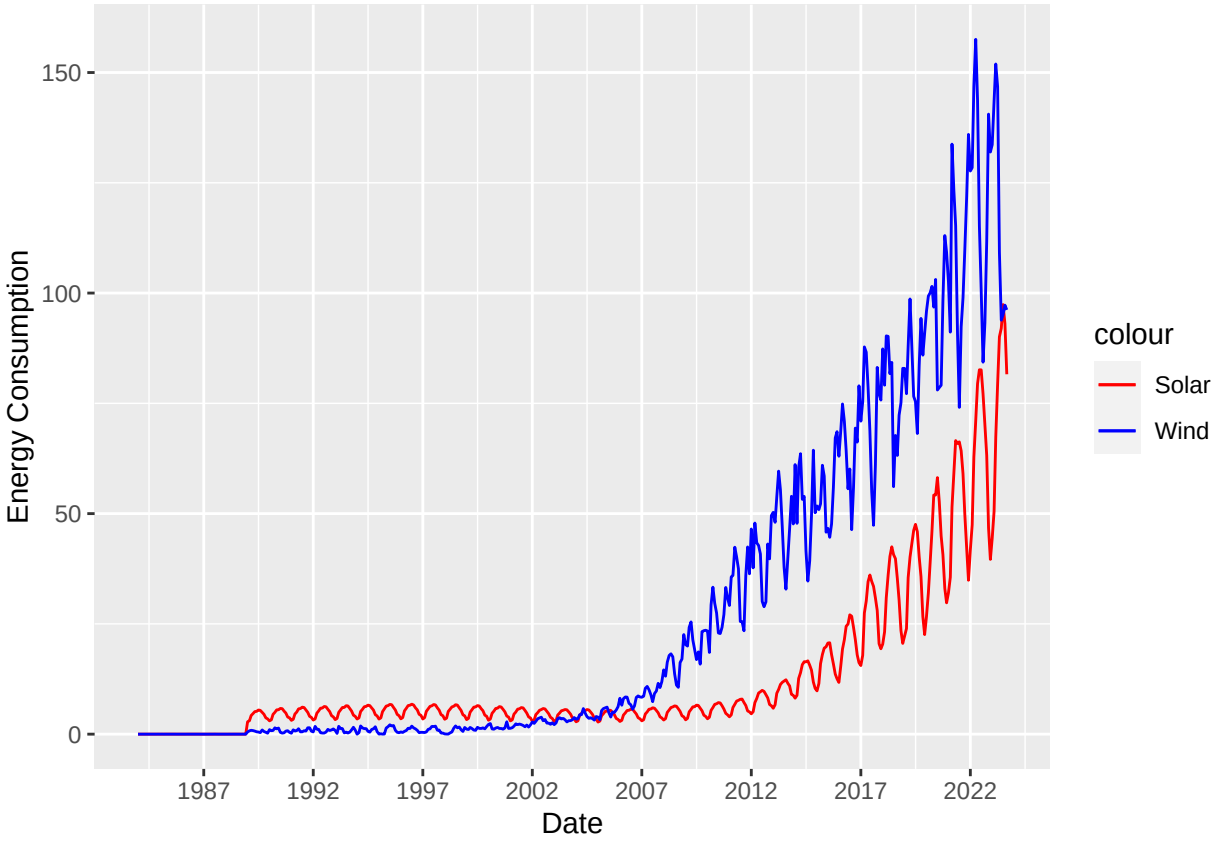

```
# Plotting Wind Energy Consumption
ggplot(clean_energy_data, aes(x = Date, y = Wind)) +
  geom_line() +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  ylab("Wind Energy Consumption")
```



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.

```
# Plotting both series in the same graph
ggplot(clean_energy_data) +
  geom_line(aes(x = Date, y = Solar, color = "Solar")) +
  geom_line(aes(x = Date, y = Wind, color = "Wind")) +
  scale_color_manual(values = c("Solar" = "red", "Wind" = "blue")) +
  ylab("Energy Consumption") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```



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.

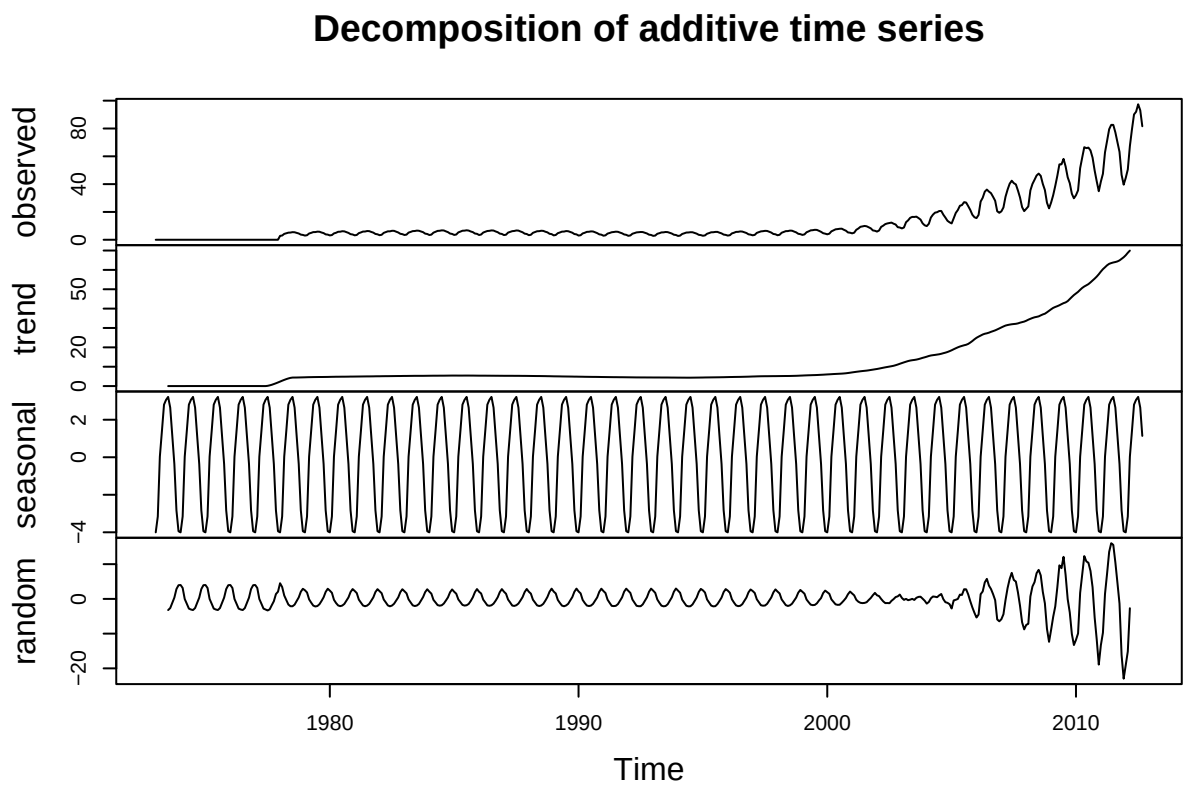
Additional info on `decompose()`.

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.

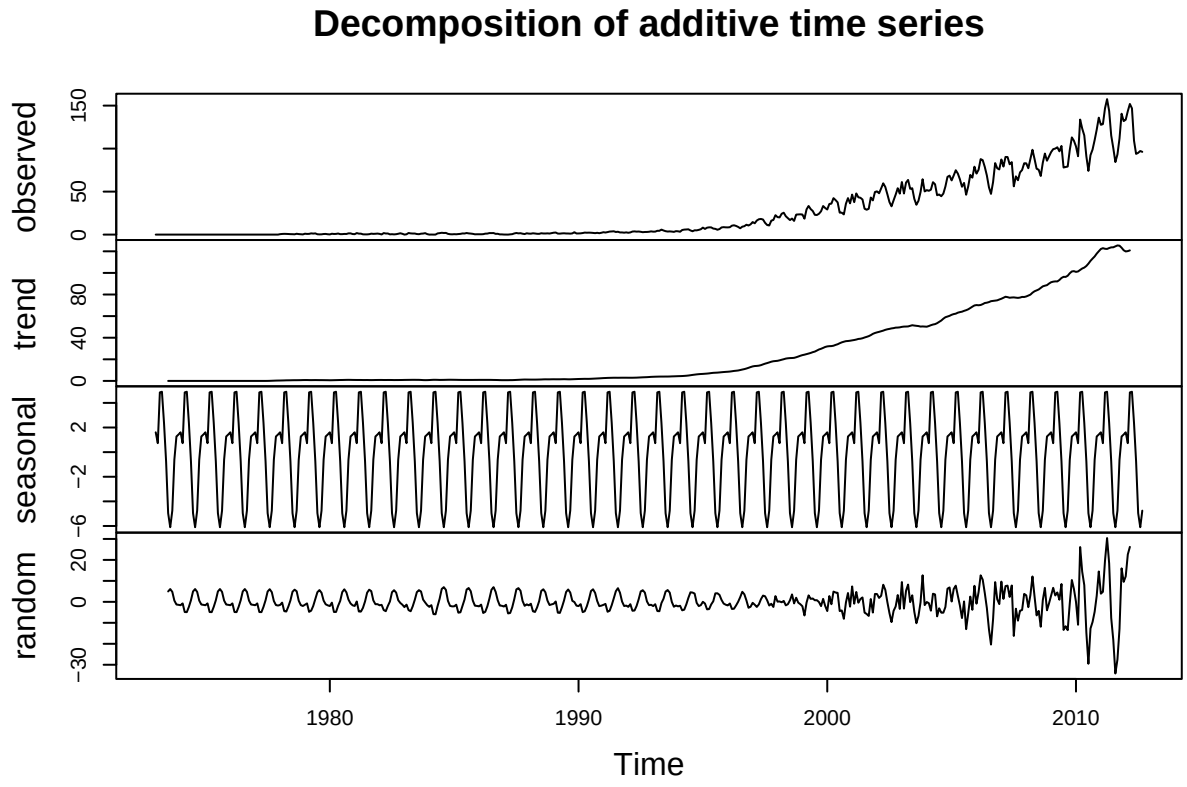
Q4

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
# Transforming into time series objects
ts_solar <- ts(clean_energy_data$Solar, start = c(1973, 1), frequency = 12)
ts_wind <- ts(clean_energy_data$Wind, start = c(1973, 1), frequency = 12)

# Decomposing the time series
decomposed_solar <- decompose(ts_solar, type = "additive")
decomposed_wind <- decompose(ts_wind, type = "additive")

# Plotting the decomposed series
plot(decomposed_solar)
```




```r
plot(decomposed_wind)
```

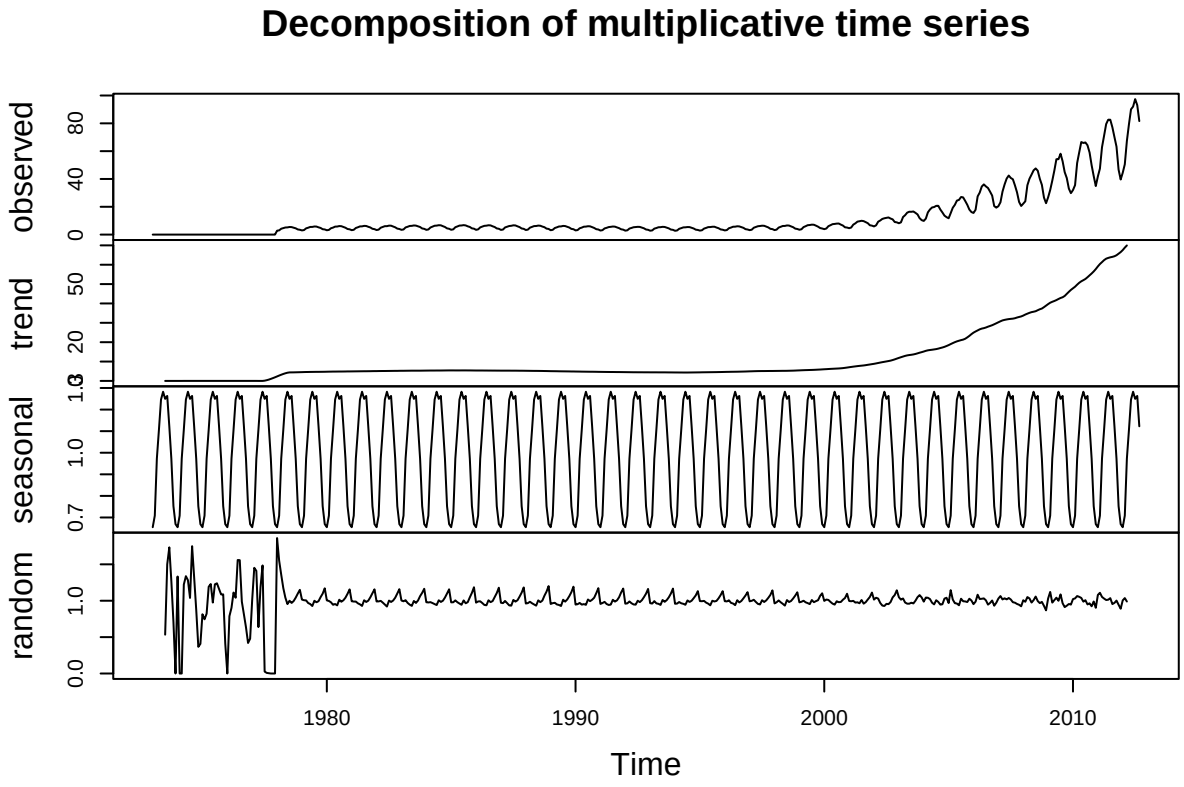


The decomposed time series plots reveal a distinct long-term increasing trend, indicating a steady rise in the observed values, suggesting growing solar and wind energy production over time. The trends are steep in recent years, suggesting an acceleration. However, the random components of both plots show variability that hints at underlying patterns, with bursts of higher variability and not entirely random fluctuations, suggesting that the seasonal decomposition may not have captured all the seasonality in the data.

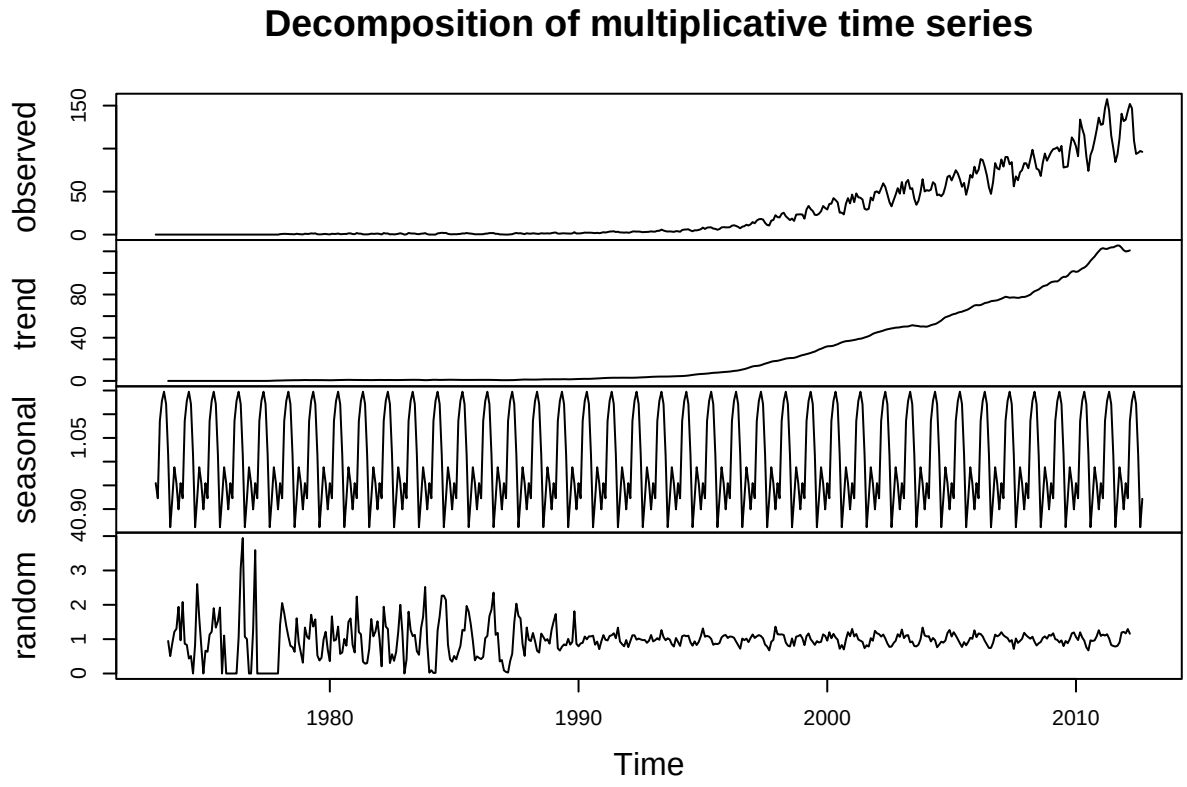
Q5

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?

```
# Decomposing with multiplicative option
decomposed_solar_mult <- decompose(ts_solar, type = "multiplicative")
decomposed_wind_mult <- decompose(ts_wind, type = "multiplicative")

# Plotting the decomposed series
plot(decomposed_solar_mult)
```




```
plot(decomposed_wind_mult)
```



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.

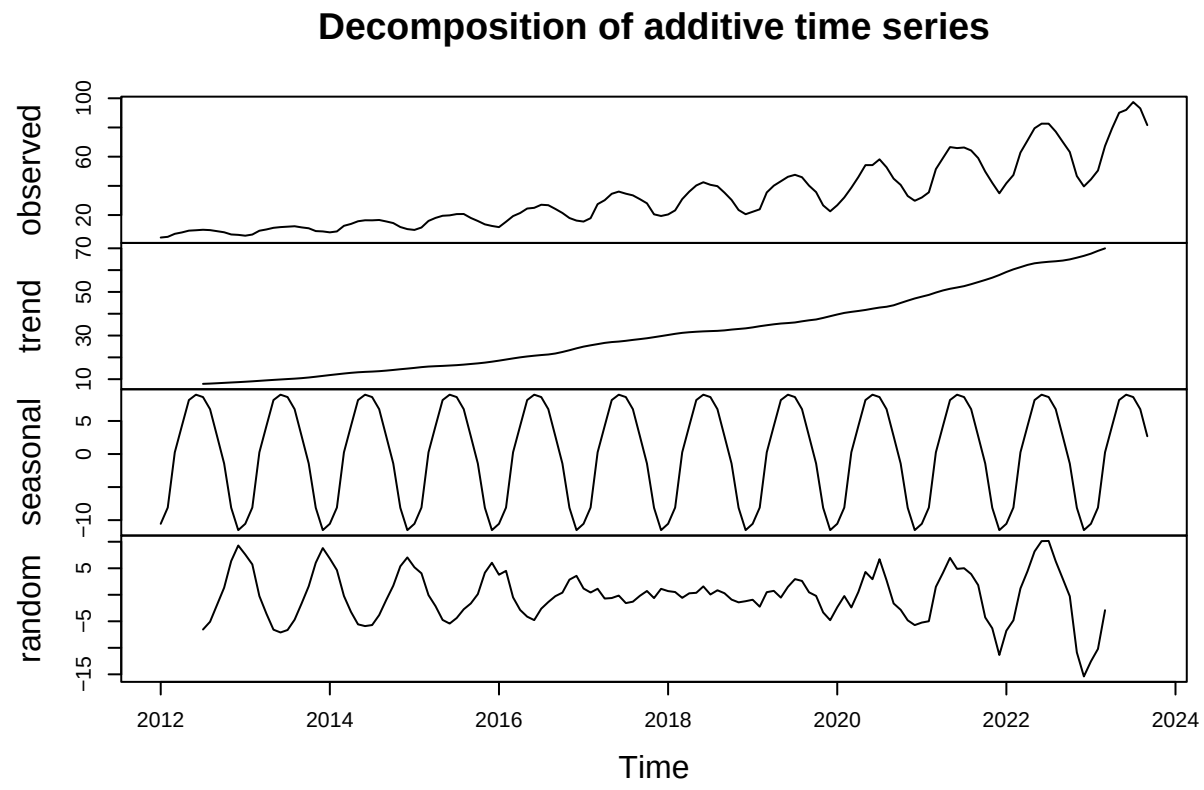
> Answer: Historical data from the 90s and early 2000s may not be as relevant for forecasting the next six months of Solar and/or Wind consumption due to significant changes in technology, policy, and market conditions over time. It might be more effective to focus on more recent data to capture the current trends and patterns.

Q7

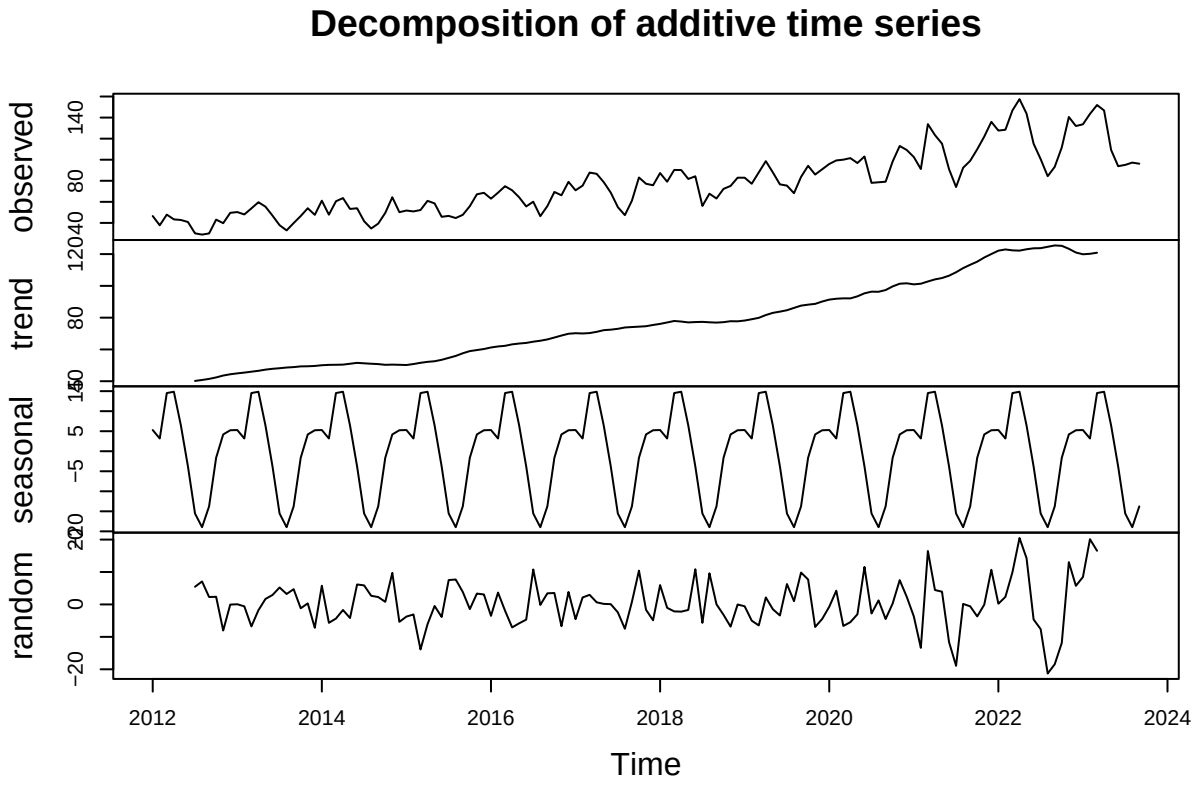
Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the decompose function `type=additive` to this new time series. Comment the results. Does the random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

```
# Filtering data from January 2012 onwards
recent_energy_data <- clean_energy_data %>% filter(year(Date) >= 2012)

# Transforming into time series objects
ts_solar_recent <- ts(recent_energy_data$Solar, start = c(2012, 1), frequency = 12)
```

```r
ts_wind_recent <- ts(recent_energy_data$Wind, start = c(2012, 1), frequency = 12)

# Decomposing the time series
decomposed_solar_recent <- decompose(ts_solar_recent, type = "additive")
decomposed_wind_recent <- decompose(ts_wind_recent, type = "additive")

# Plotting the decomposed series
plot(decomposed_solar_recent)
```




```r
plot(decomposed_wind_recent)
```



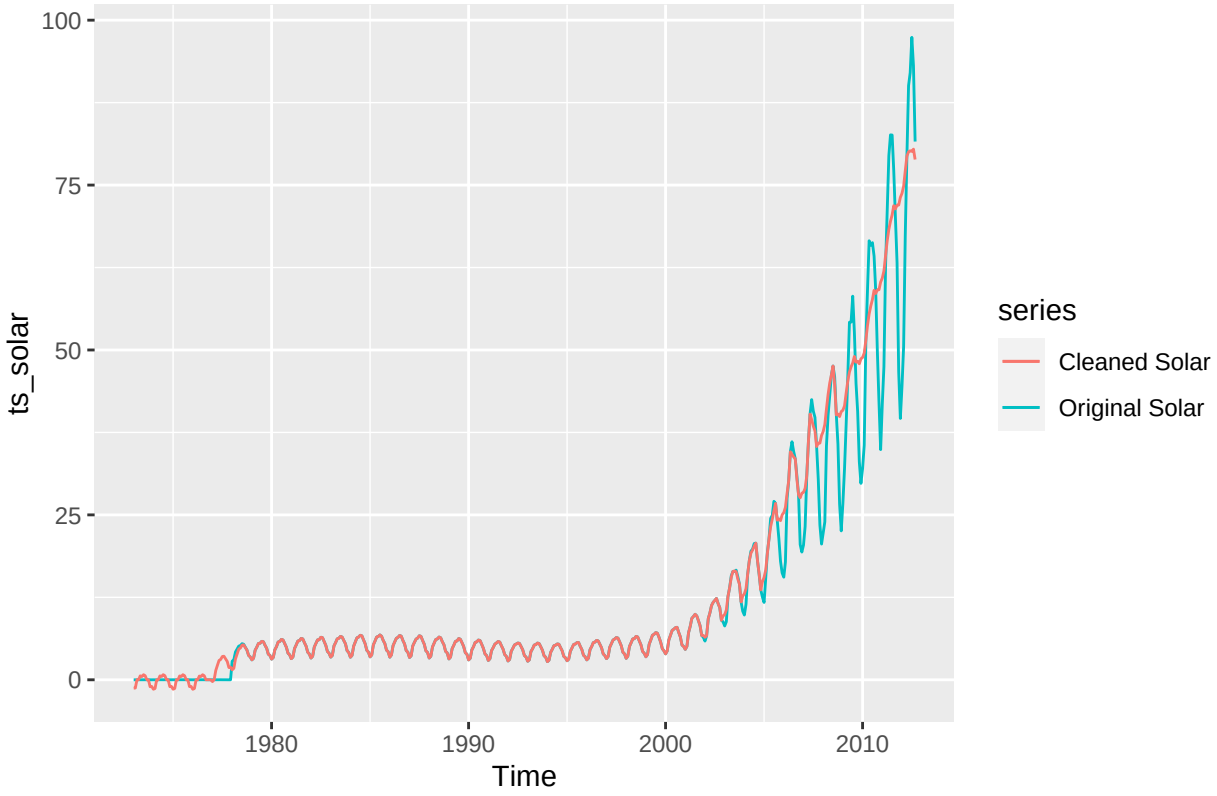
Answer: The observed data shows clear seasonality with a fluctuating pattern that repeats every year. The trend component indicates a gradual increase over time, suggesting a long-term rise in the underlying level of the data series. The random component show some patterns and do not appear entirely random. There are periods where the variability in the random component is larger, which could be tied to the level of the series itself. This is indicative of the potential inadequacy of an additive model for series with changing variance over time. In such cases, a multiplicative model might be more appropriate because it allows the seasonal component to vary in proportion to the level of the series.

Identify and Remove outliers

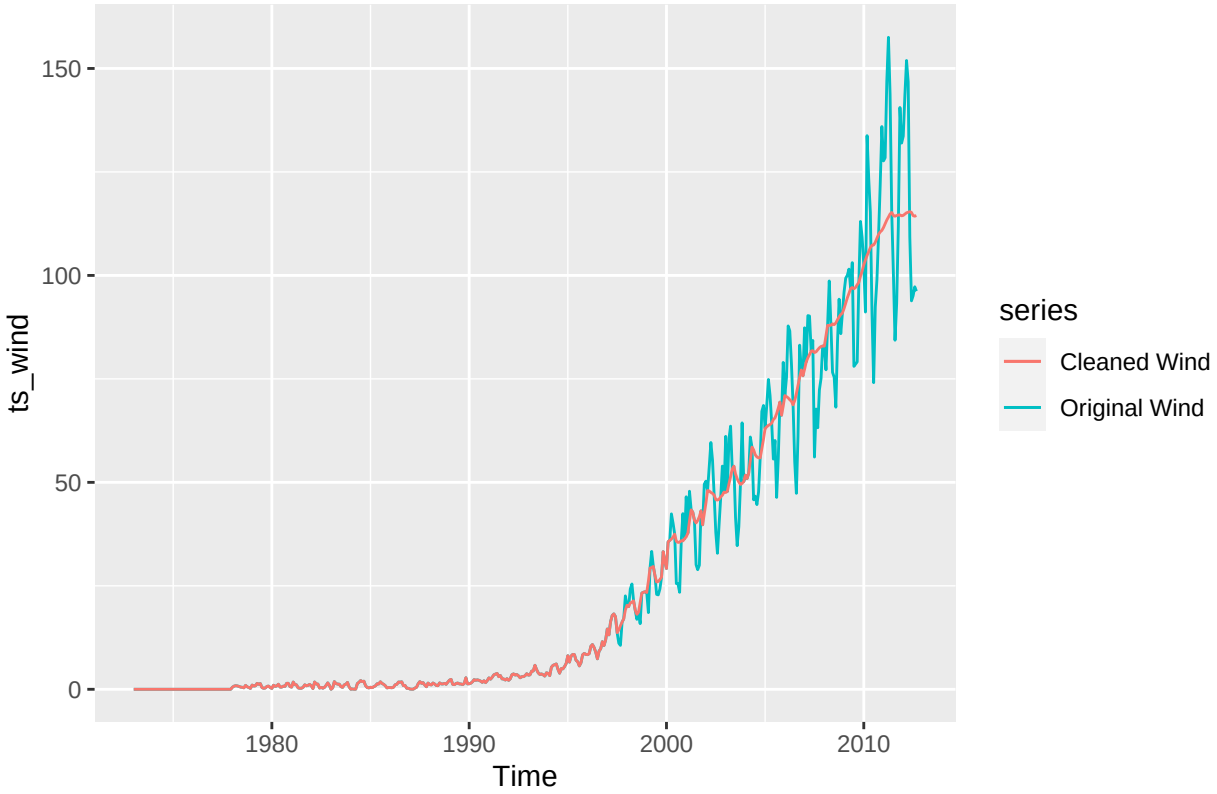
Q8

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.

```
# Applying tsclean to the series
clean_solar <- tsclean(ts_solar)
clean_wind <- tsclean(ts_wind)

# Plotting to check for differences
autoplot(ts_solar, series = "Original Solar") +
  autolayer(clean_solar, series = "Cleaned Solar")
```




```
autoplot(ts_wind, series = "Original Wind") +
  autolayer(clean_wind, series = "Cleaned Wind")
```

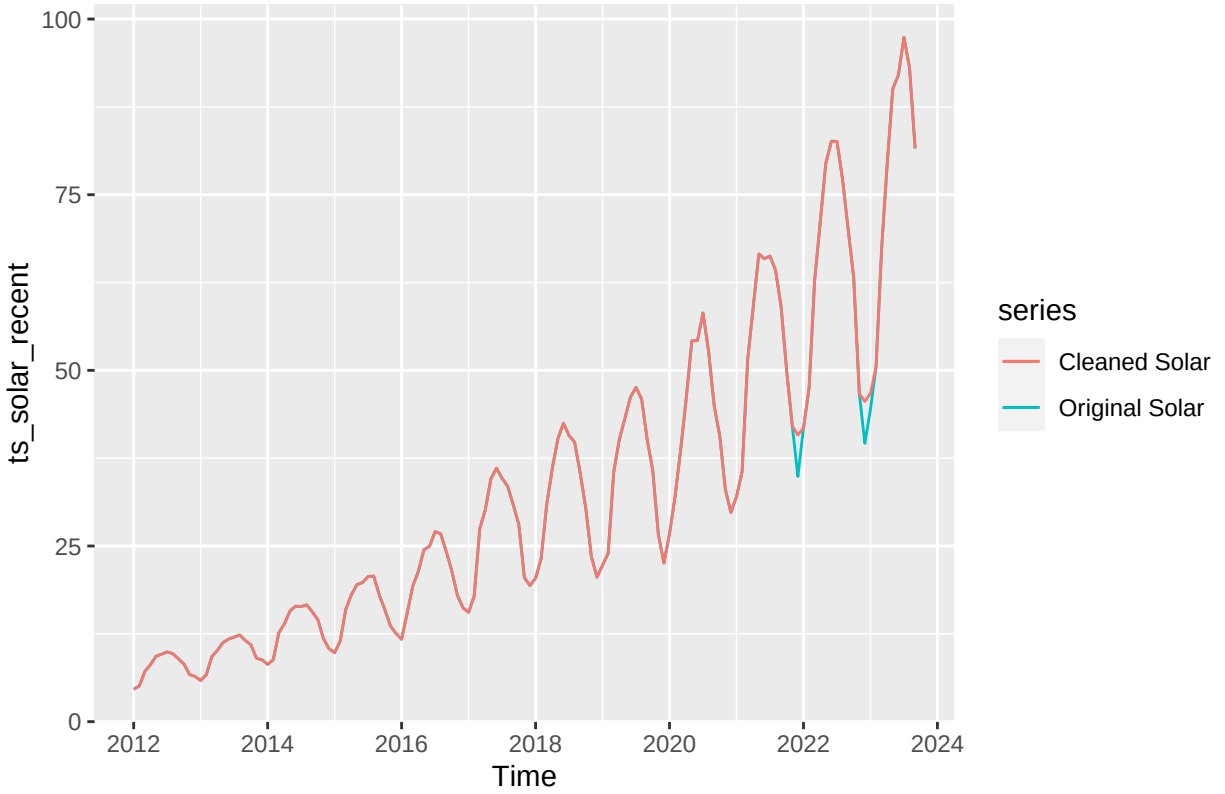


Answer: Both cleaned graphs removed many outliers. Most of the outliers started to appear around 2003, and are becoming more and more since then.

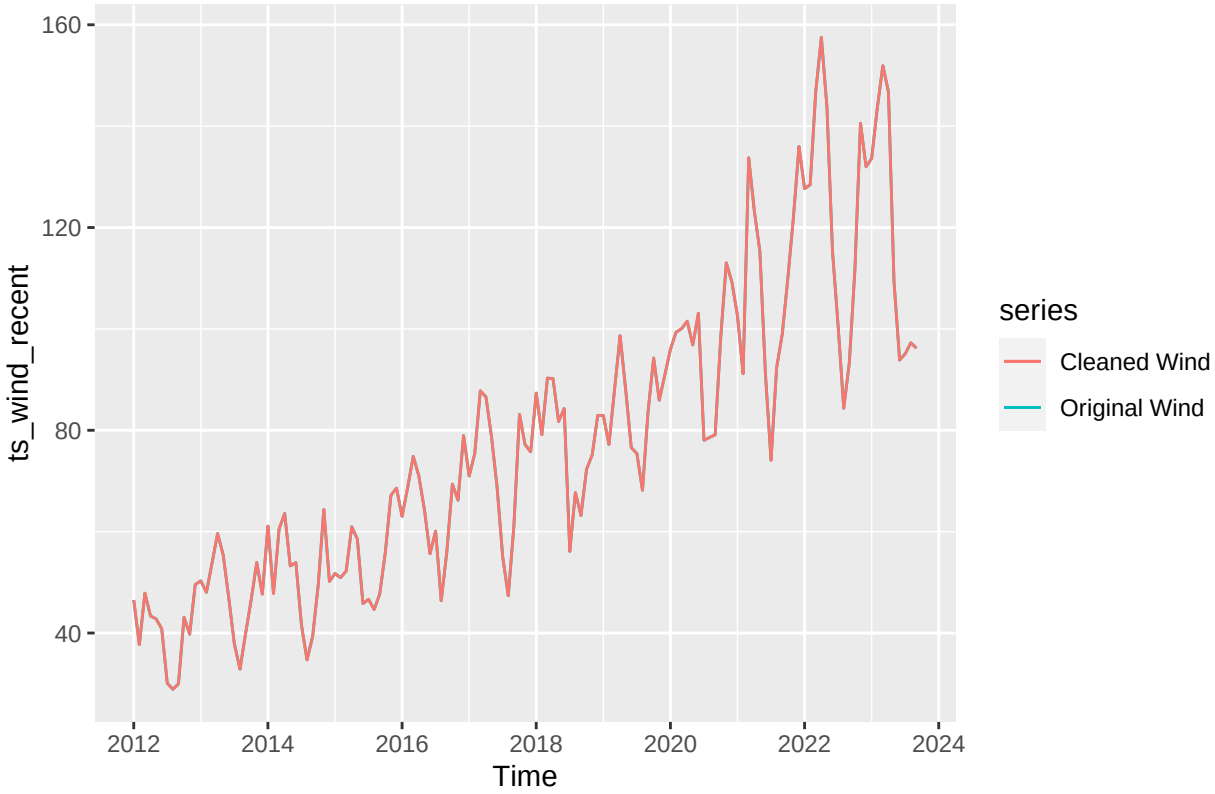
Q9

Redo number Q8 but now with the time series you created on Q7, i.e., the series starting in 2012. Using what `autoplot()` again what happened now?Did the function removed any outliers from the series?

```
# Applying tsclean to the series starting in 2012
# Applying tsclean to the series
clean_solar_recent <- tsclean(ts_solar_recent)
clean_wind_recent <- tsclean(ts_wind_recent)

# Plotting to check for differences
autoplot(ts_solar_recent, series = "Original Solar") +
  autolayer(clean_solar_recent, series = "Cleaned Solar")
```




```
autoplot(ts_wind_recent, series = "Original Wind") +
  autolayer(clean_wind_recent, series = "Cleaned Wind")
```



Answer: Overall both solar and wind have far less outliers after cleaning. For the cleaned solar one, there are still a few outliers clearly identified, but the original and the cleaned series of the wind data seem to be very similar and there are no obvious outliers identified.