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

Assignment 5 - Due date 02/13/24

Jenn McNeill

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.\

```
#Install required packages
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)
```

```
## -- 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
```

```
library(dplyr)
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.

```
#Import the dataset
energy_data <- read_excel(path="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
```

```
#Extract the column names from row 11
read_col_names <- read_excel(path="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Sou
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
```

```
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
```

```
#Add column names to the table
colnames(energy_data) <- read_col_names

#Check column 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)
```

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 of interest
df <- energy_data %>%
  select(`Month`, `Solar Energy Consumption`, `Wind Energy Consumption`) %>%
  mutate(`Solar Energy Consumption` = as.numeric(`Solar Energy Consumption`),
         `Wind Energy Consumption` = as.numeric(`Wind Energy Consumption`)) %>%
  drop_na()
```

```
## Warning: There were 2 warnings in `mutate()`.
## The first warning was:
## i In argument: `Solar Energy Consumption = as.numeric(`Solar Energy
##   Consumption`)`.
```

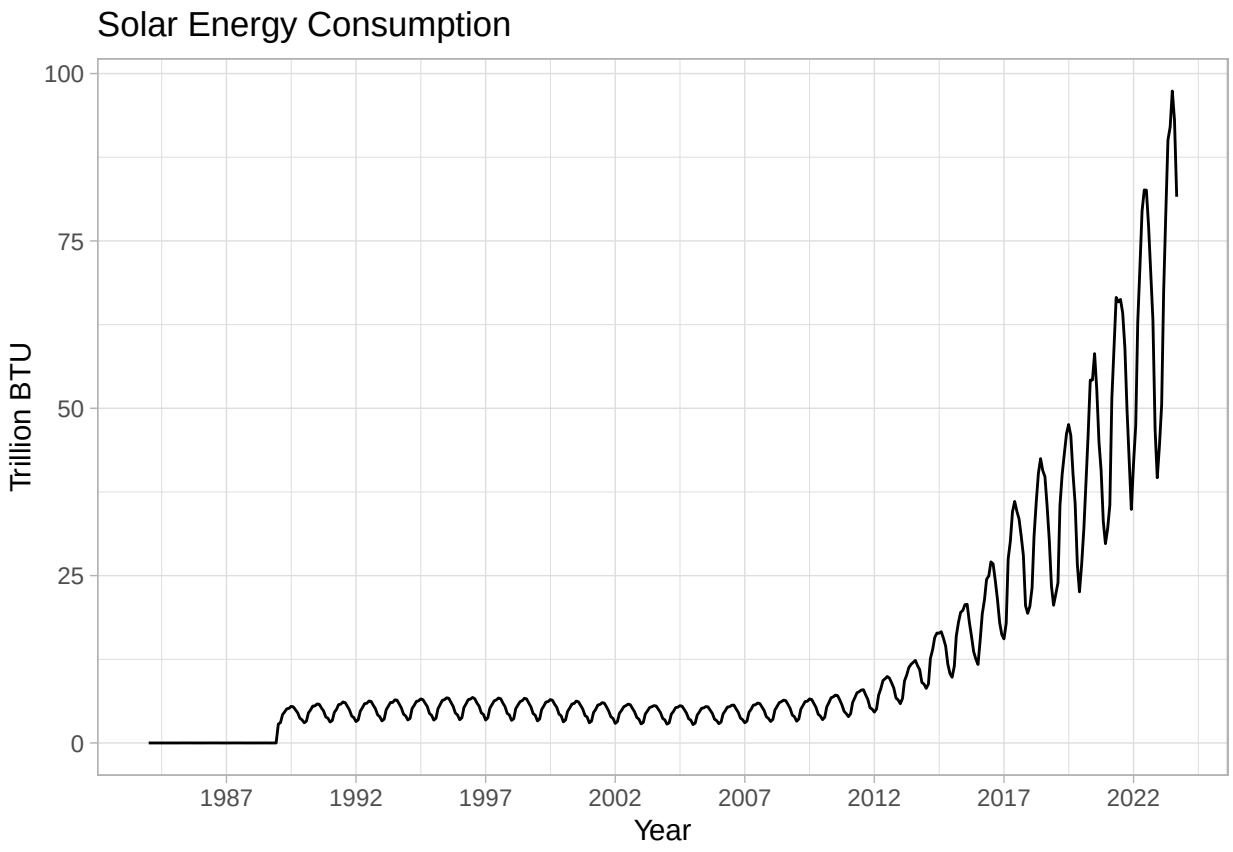
```
## Caused by warning:
## ! NAs introduced by coercion
## i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```
#Change the date to a date object
Date <- ymd(df$Month)

#Replace the date
df <- cbind(Date,df[,2:3])
```

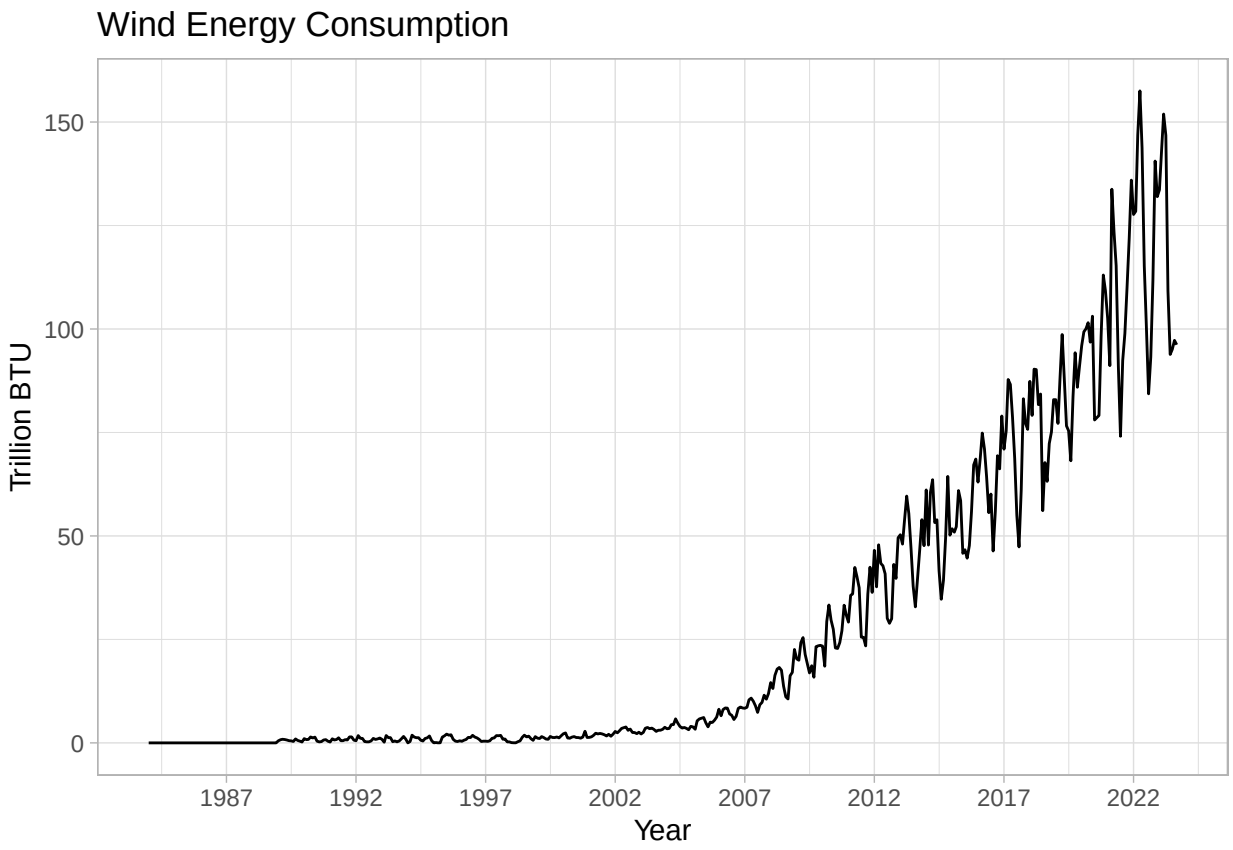
Q2

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(df, aes(x = Date, y = `Solar Energy Consumption`))+
  geom_line()+
  ggtitle("Solar Energy Consumption")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  xlab("Year")+
  ylab("Trillion BTU")+
  theme_light()
```



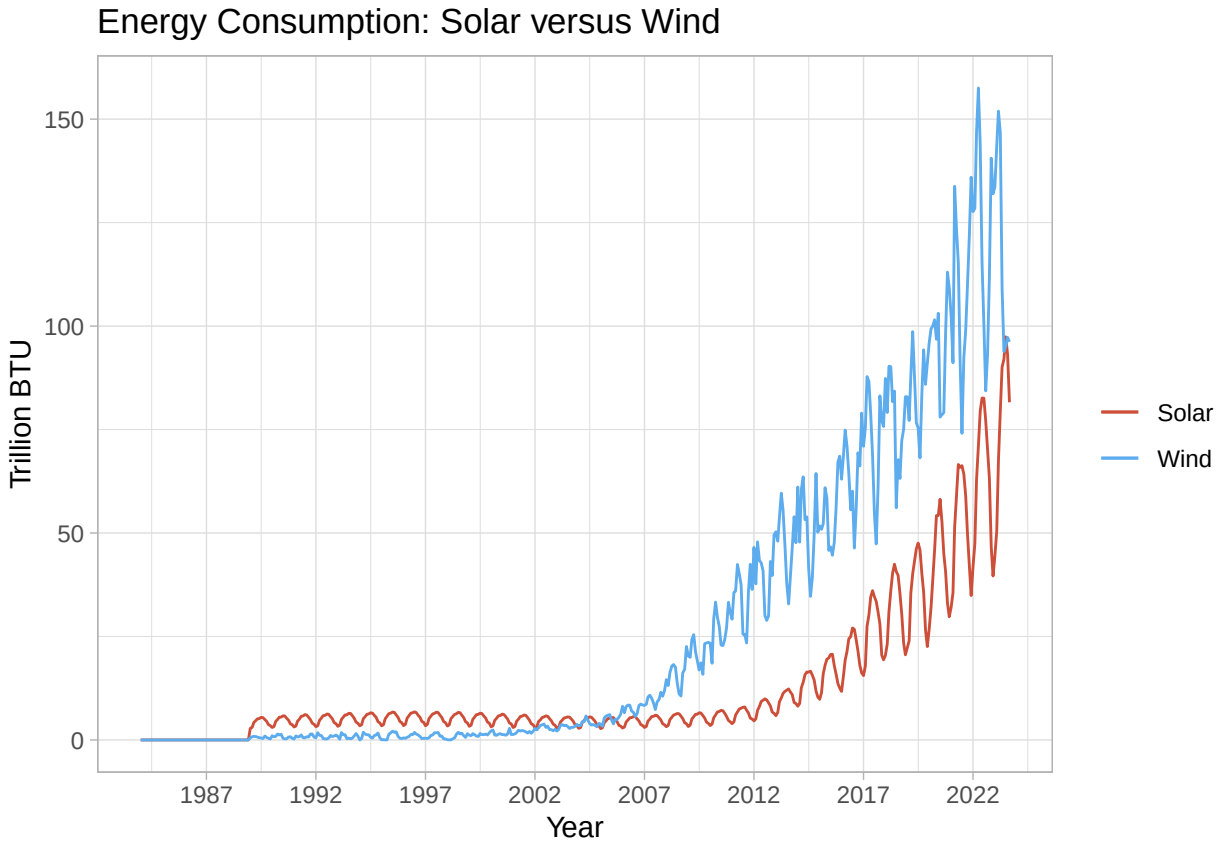
```r
ggplot(df, aes(x = Date, y = `Wind Energy Consumption`))+
  geom_line()+
  ggtitle("Wind Energy Consumption")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  xlab("Year")+
  ylab("Trillion BTU")+
  theme_light()
```



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.

```r
ggplot(df, 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" = "tomato3", "Wind" = "steelblue2"))+
  labs(color = "")+
  xlab("Year")+
  ylab("Trillion BTU")+
  ggtitle("Energy Consumption: Solar versus Wind")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  theme_light()
```



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.

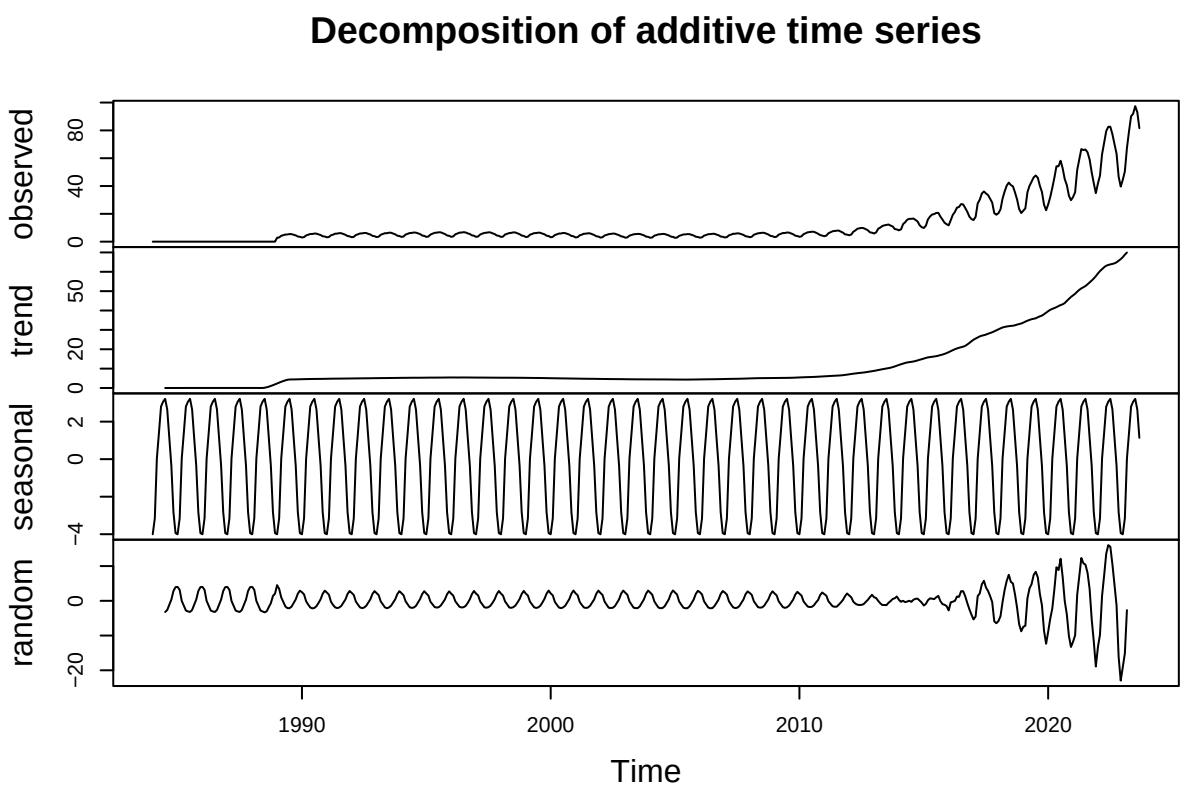
Additional info on `decompose()`.

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.

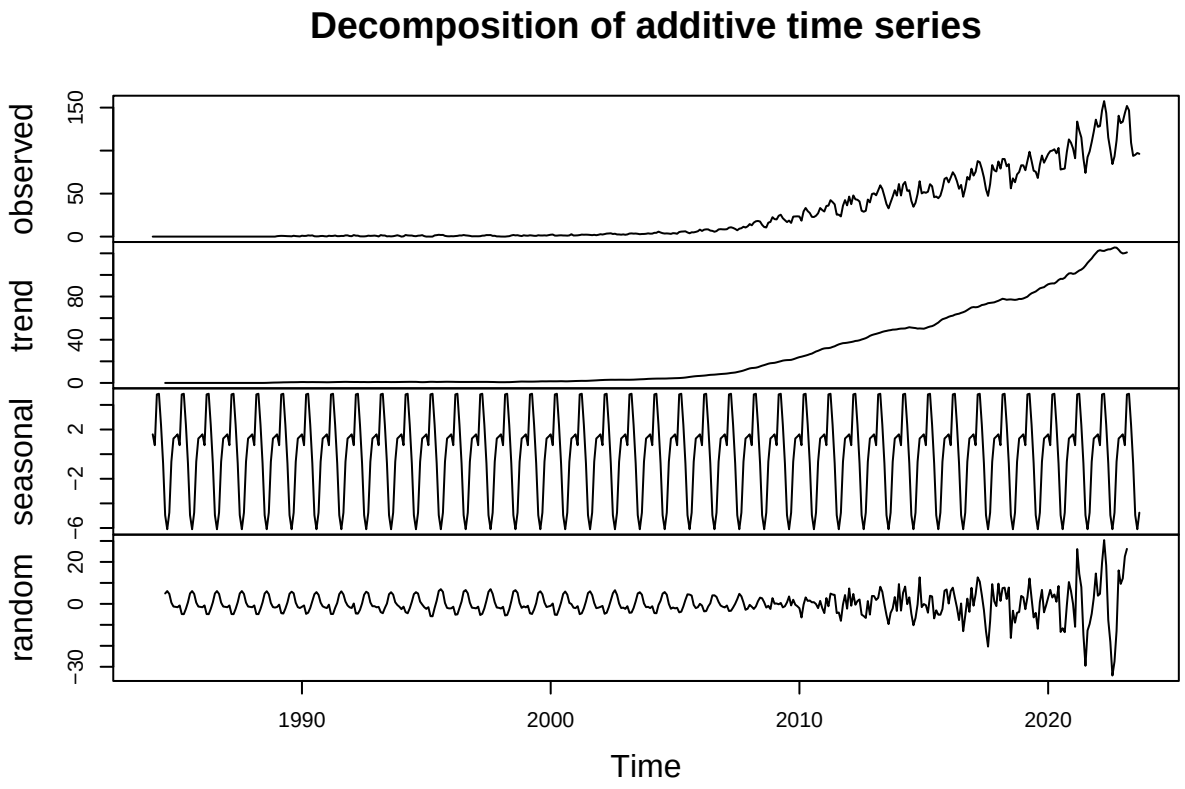
Q4

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
ts_solar <- ts(df[,2],start=c(1984,1), frequency=12)
ts_wind <- ts(df[,3],start=c(1984,1), frequency=12)

decompose_solar_add <- decompose(ts_solar, type = "additive")
decompose_wind_add <- decompose(ts_wind, type="additive")

plot(decompose_solar_add)
```




```r
plot(decompose_wind_add)
```

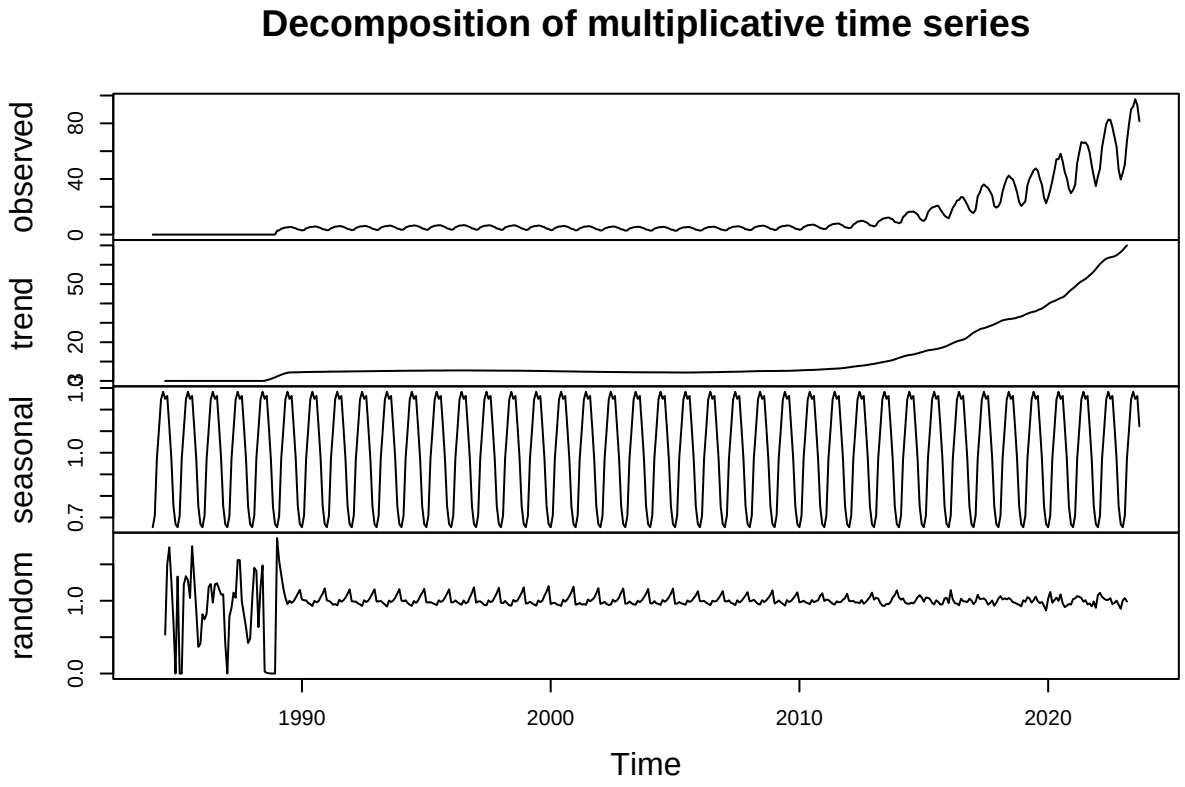


Answer: Both series show trend components that have a steep trend upwards starting around 2010. The random components for both series show more noise as the trend starts creeping upwards. The random components seem to have a time dependency and some seasonality because they follow a somewhat predictable up and down pattern.

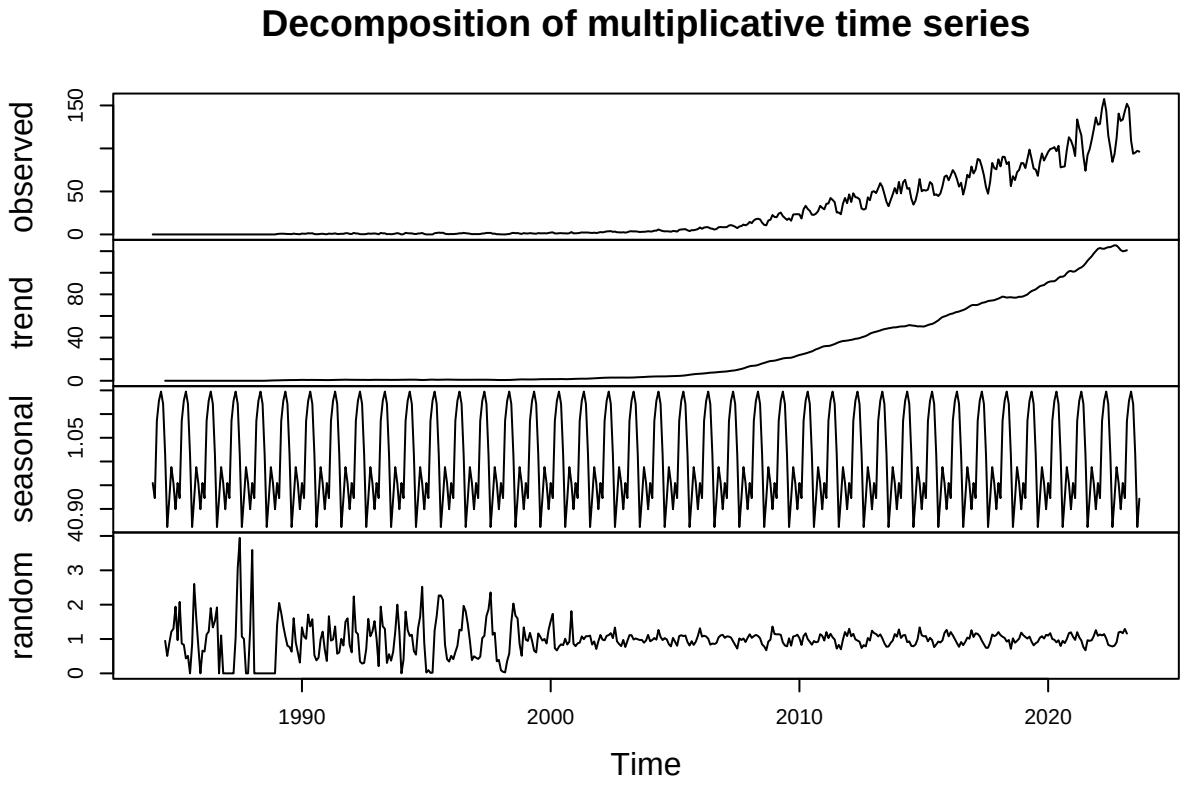
Q5

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_mult <- decompose(ts_solar, type = "multiplicative")
decompose_wind_mult <- decompose(ts_wind, type="multiplicative")

plot(decompose_solar_mult)
```




```
plot(decompose_wind_mult)
```



Answer: The random component for both time series appears much more random using the multiplicative method. The solar time series still has a bit of a predictable, seasonal pattern with

the multiplicative method, but not as severe as with the additive method. The wind time series using the multiplicative method appears truly random.

Q6

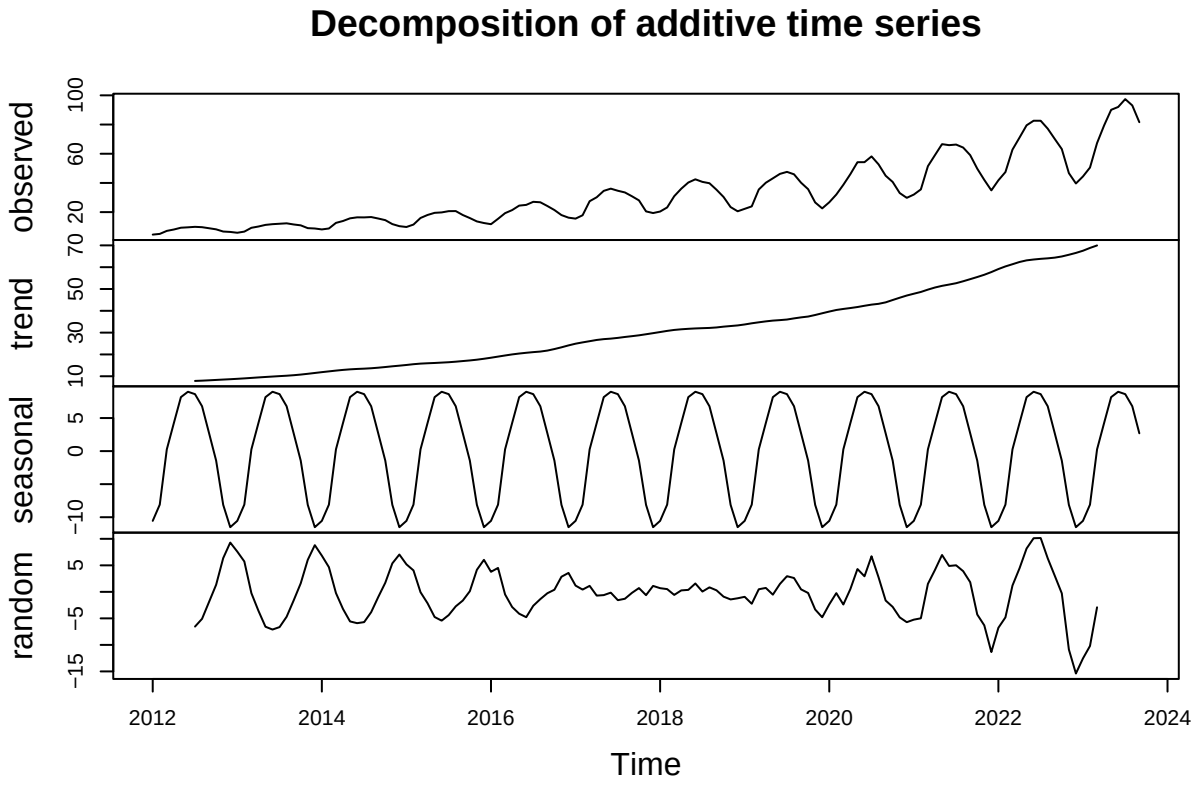
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

Answer: Both of these time series show very low solar and wind energy consumption in the 1990s and early 2000s. In my opinion, this data is not a good forecast of what is to come in the later years when the renewable energy industry begins to increase in popularity. Looking at the time series plotted together above, I think it would be a conservative approach to start using data from 2000 as a forecast for the rest of the series and omitting the data from earlier on.

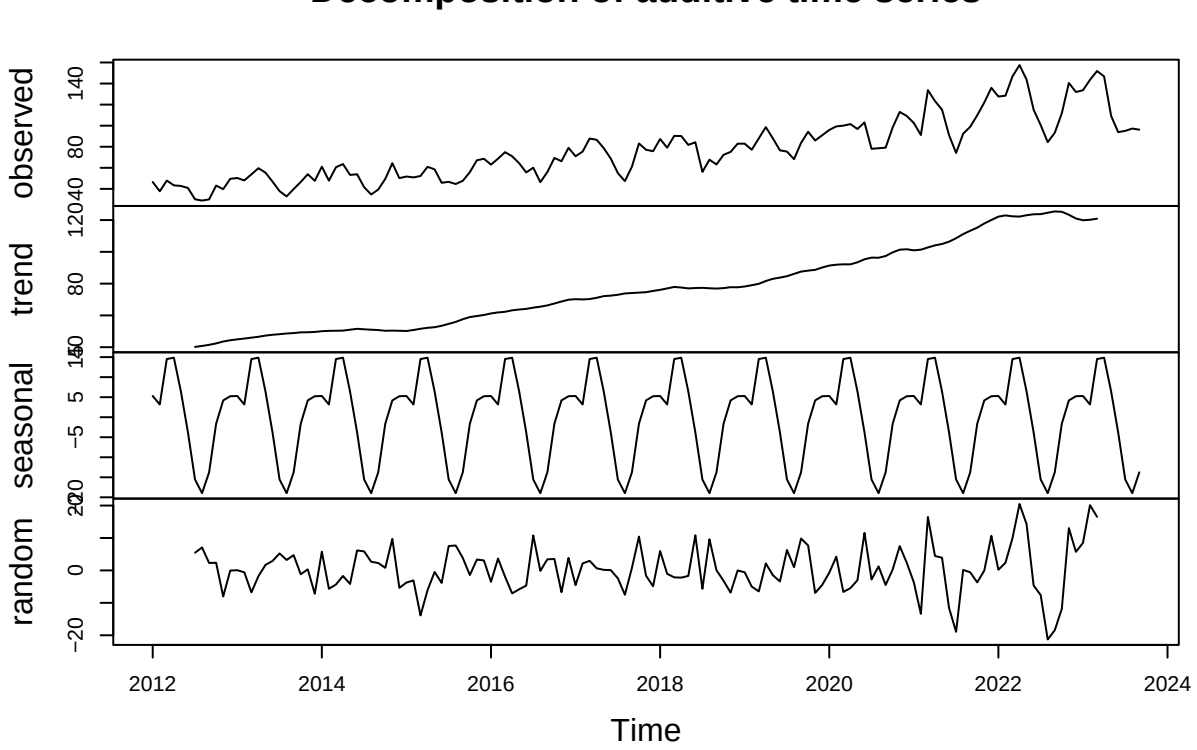
Q7

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.

```
df_2012 <- df %>%
  filter(year(Date) >= 2012)

ts_solar_2012 <- ts(df_2012[,2],start=c(2012,1), frequency=12)
ts_wind_2012 <- ts(df_2012[,3],start=c(2012,1), frequency=12)

decompose_solar_add_2012 <- decompose(ts_solar_2012, type = "additive")
decompose_wind_add_2012 <- decompose(ts_wind_2012, type="additive")

plot(decompose_solar_add_2012)
```




```
plot(decompose_wind_add_2012)
```



Answer: The random component for the wind series starting at 2012 looks very random. The

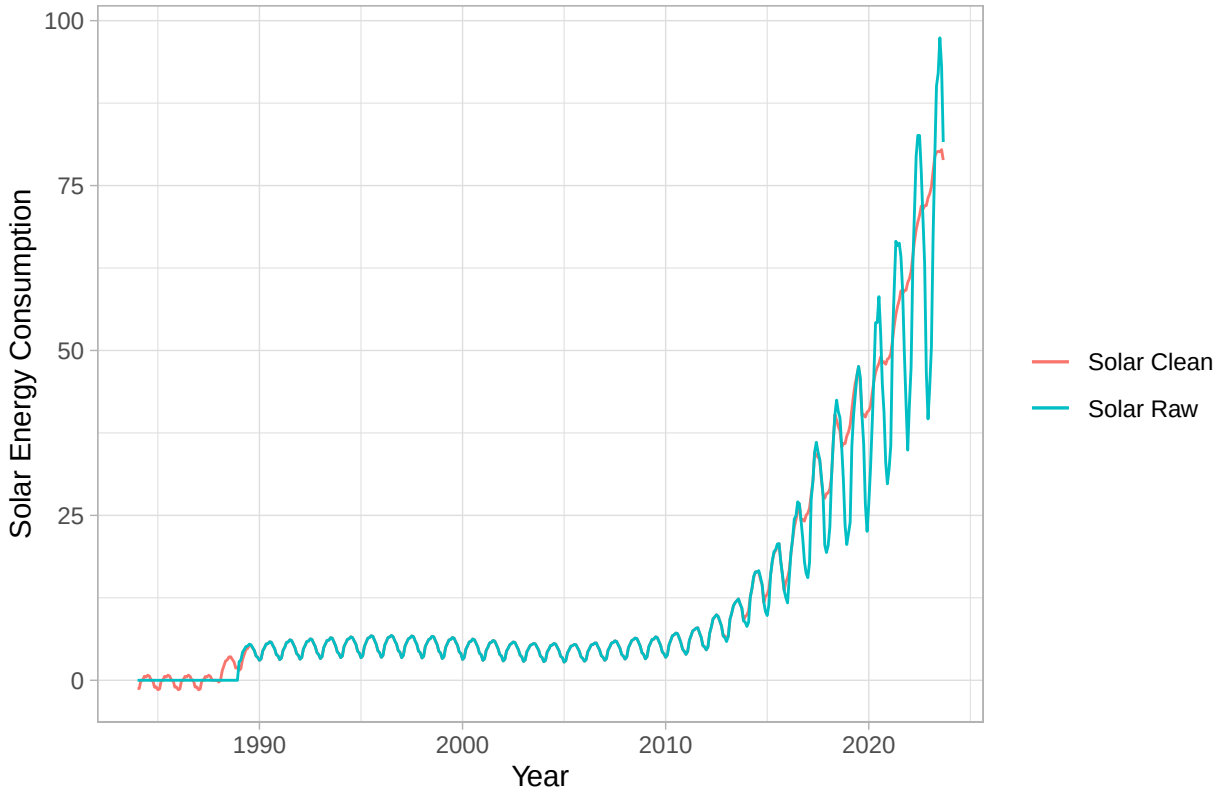
random component for the solar series starting at 2012 has regions of randomness but other regions where the randomness appears to be seasonal.

Identify and Remove outliers

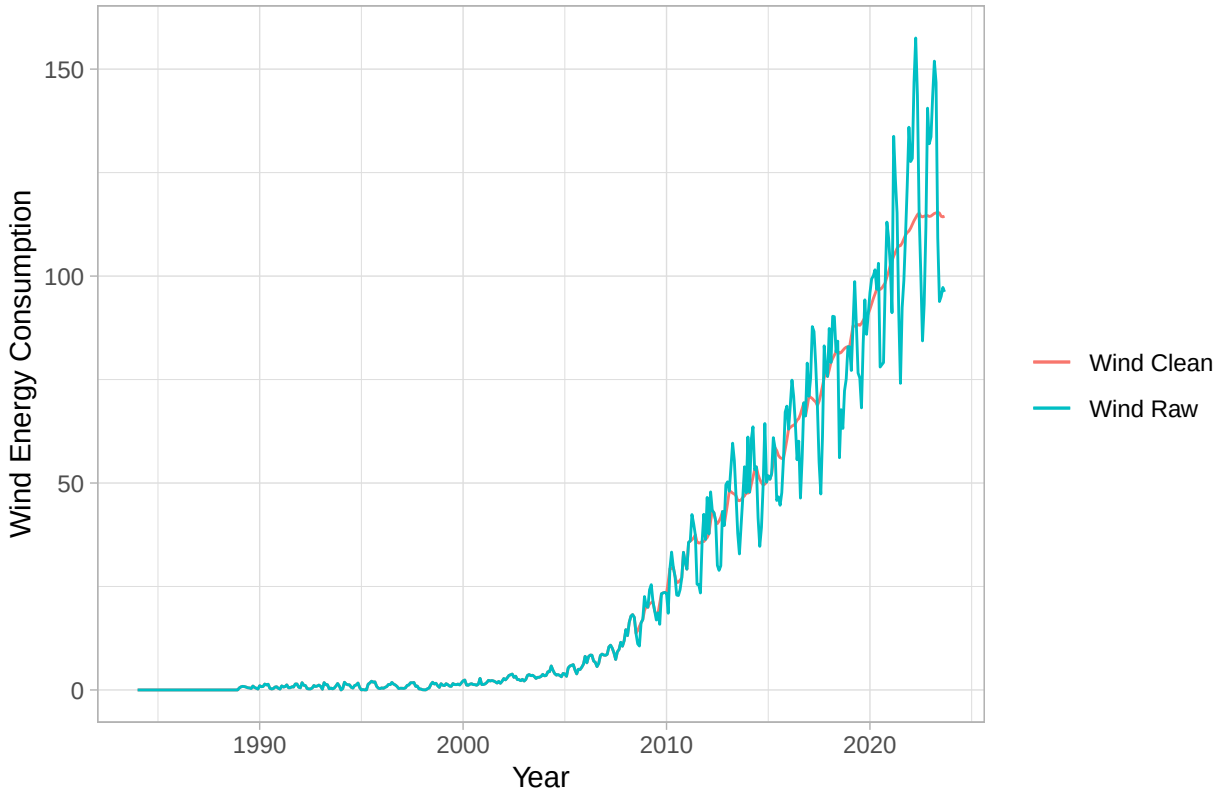
Q8

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.

```
clean_solar <- tsclean(ts_solar)
clean_wind <- tsclean(ts_wind)

autoplot(clean_solar, series="Solar Clean")+
  autolayer(ts_solar, series="Solar Raw")+
  xlab("Year")+
  ylab("Solar Energy Consumption")+
  labs(color = "")+
  theme_light()
```




```
autoplot(clean_wind, series="Wind Clean")+
  autolayer(ts_wind, series="Wind Raw")+
  xlab("Year")+
  ylab("Wind Energy Consumption")+
  labs(color = "")+
  theme_light()
```

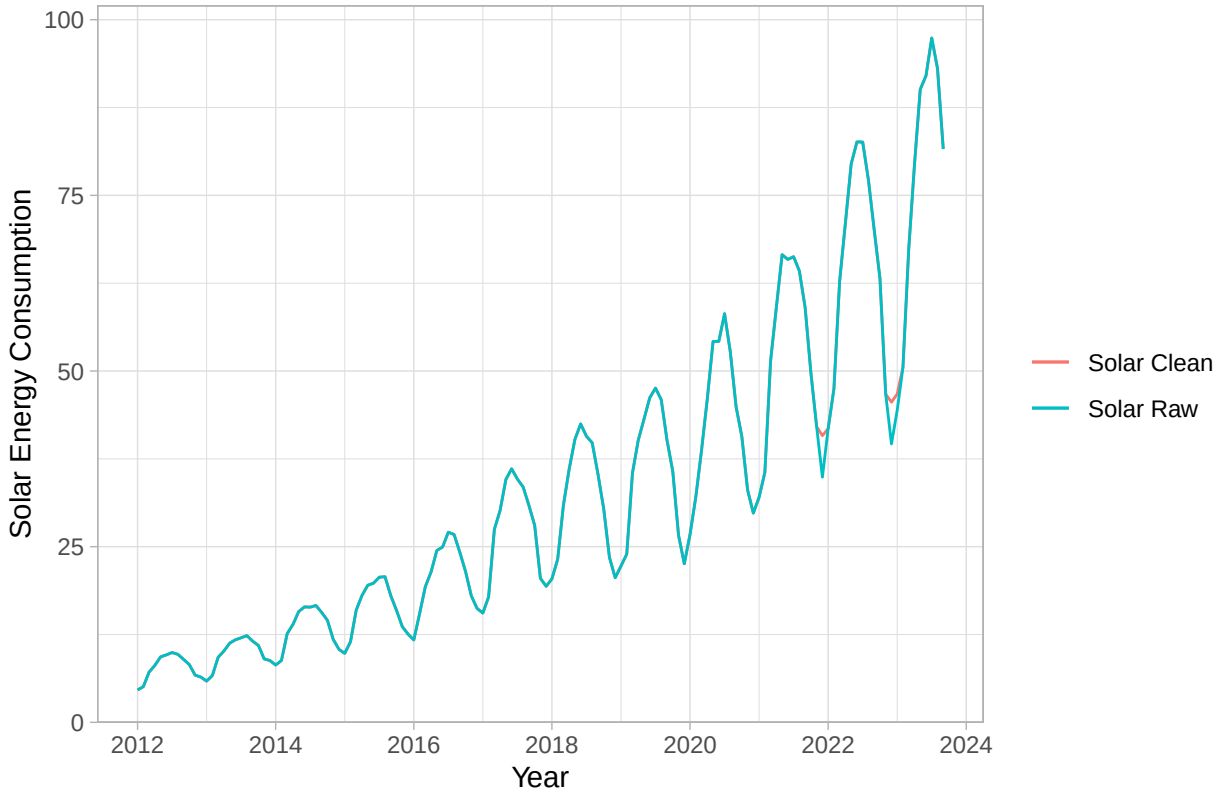


Answer: The tsclean function removed a lot of outliers from both time series. The clean time series lines show much less intense fluctuation between the high and low values in the data from years 2010 and on.

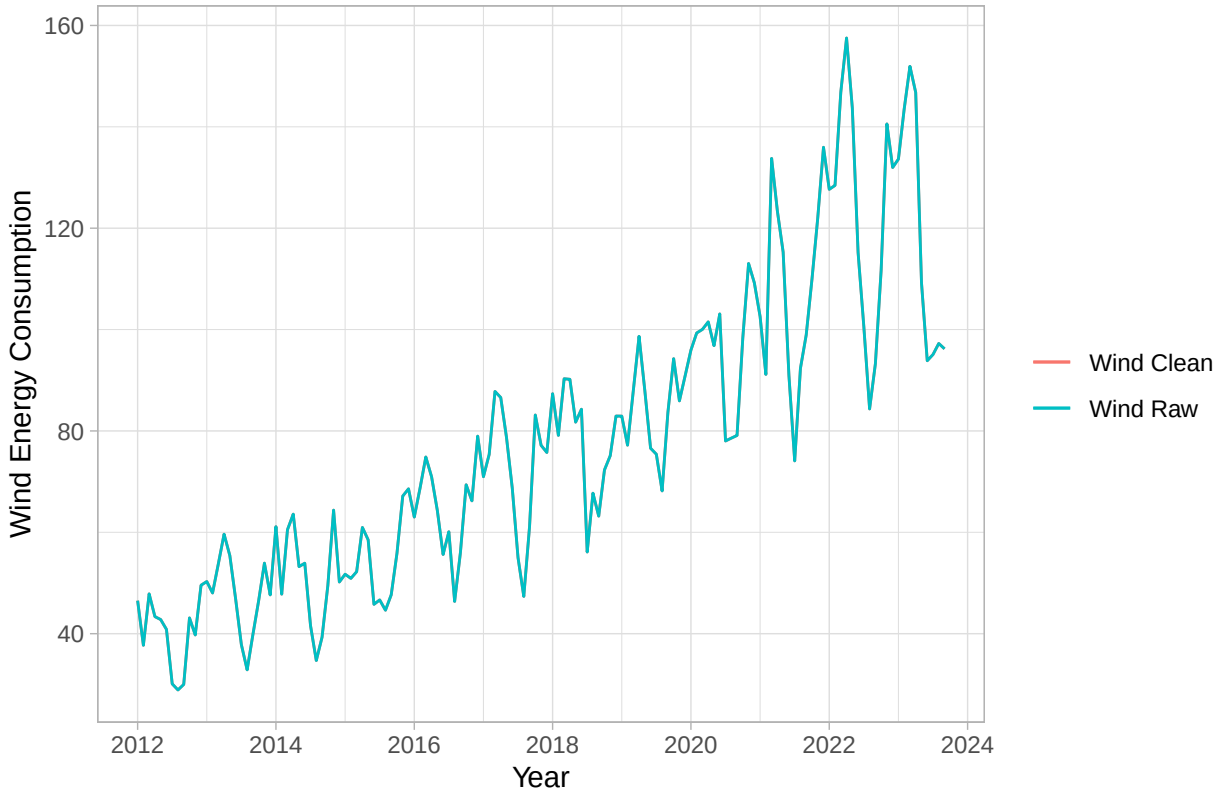
Q9

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_2012 <- tsclean(ts_solar_2012)
clean_wind_2012 <- tsclean(ts_wind_2012)

autoplot(clean_solar_2012, series="Solar Clean")+
  autolayer(ts_solar_2012, series="Solar Raw")+
  xlab("Year")+
  ylab("Solar Energy Consumption")+
  labs(color = "")+
  theme_light()
```




```
autoplot(clean_wind_2012, series="Wind Clean")+
  autolayer(ts_wind_2012, series="Wind Raw")+
  xlab("Year")+
  ylab("Wind Energy Consumption")+
  labs(color = "")+
  theme_light()
```



Answer: Using only the data from 2012 and onwards, the tsclean function barely removes any data. This is because the function knows that the large variations in data are truly occuring and they do not seem so drastic because the earlier data where the numbers weren't fluctuating as much are not considered. High and low values are not as far away from the mean when only 2012 and onward is considered.