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

Assignment 5 - Due date 02/13/24

Jaimie Wargo

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(here)
```

```
## here() starts at C:/Users/jaimi/OneDrive/Documents/Duke/Spring_2024/TSA_Sp24
```

```r
library(readxl)
library(tidyverse)  #load this package so you can 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   1.3.0
## 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
```

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_excel(here("Data",
      "Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx"),
      skip=12, sheet = "Monthly Data", col_names=F)
```

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

```r
#startRow is equivalent to skip on read.table

#Now let's extract the column names from row 11 only
read_col_names <- read_excel(here("Data",
      "Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx"),
      skip=10, n_max=1, sheet = "Monthly Data", col_names=F)
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
```

```
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
```

```
colnames(energy_data) <- read_col_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!

```
columns <-c("Solar Energy Consumption", "Wind Energy Consumption")
energy_data[, columns] <- lapply(columns, function(x) as.numeric(energy_data[[x]]))
```

```
## Warning in FUN(X[[i]], ...): NAs introduced by coercion

## Warning in FUN(X[[i]], ...): NAs introduced by coercion
```

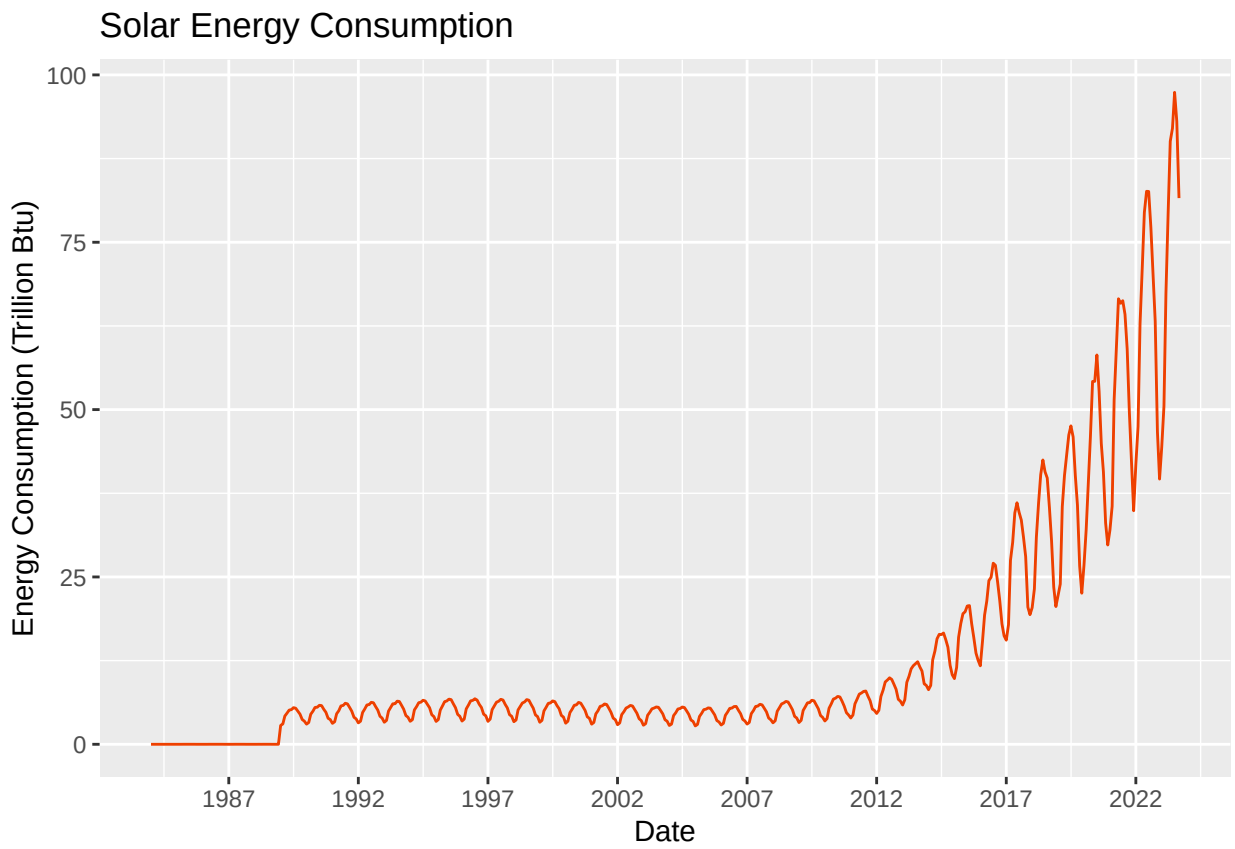
```
solar_wind <- energy_data %>%
  subset(select=c('Month', 'Solar Energy Consumption',
                  'Wind Energy Consumption')) %>%
  drop_na()

str(solar_wind)
```

```
## tibble [477 x 3] (S3: tbl_df/tbl/data.frame)
##  $ Month                   : POSIXct[1:477], format: "1984-01-01" "1984-02-01" ...
##  $ Solar Energy Consumption: num [1:477] 0 0 0.001 0.001 0.002 0.003 0.001 0.003 0.003 0.002 ...
##  $ Wind Energy Consumption : num [1:477] 0 0.001 0.001 0.002 0.003 0.002 0.002 0.001 0.002 0.003 ...
```

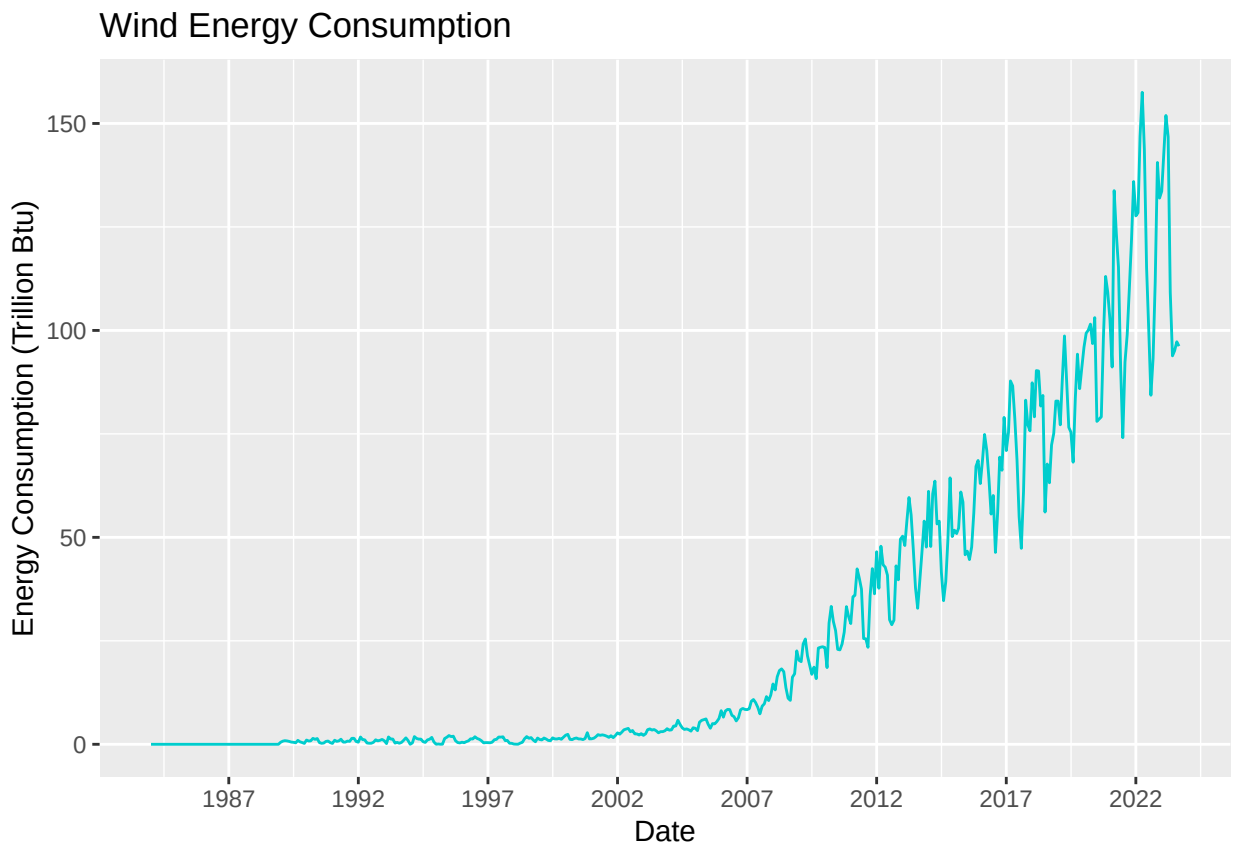
Q2

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")")`

```
nospace_colnames <- c('date', 'solar', 'wind')
colnames(solar_wind) <- nospace_colnames
solar_wind$date <- ymd(solar_wind$date)

solar1 <- ggplot(data=solar_wind, aes(x=date, y=solar))+
  geom_line(color='orangered2')+
  labs(title="Solar Energy Consumption",
       y="Energy Consumption (Trillion Btu)",
       x='Date')+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")

solar1
```



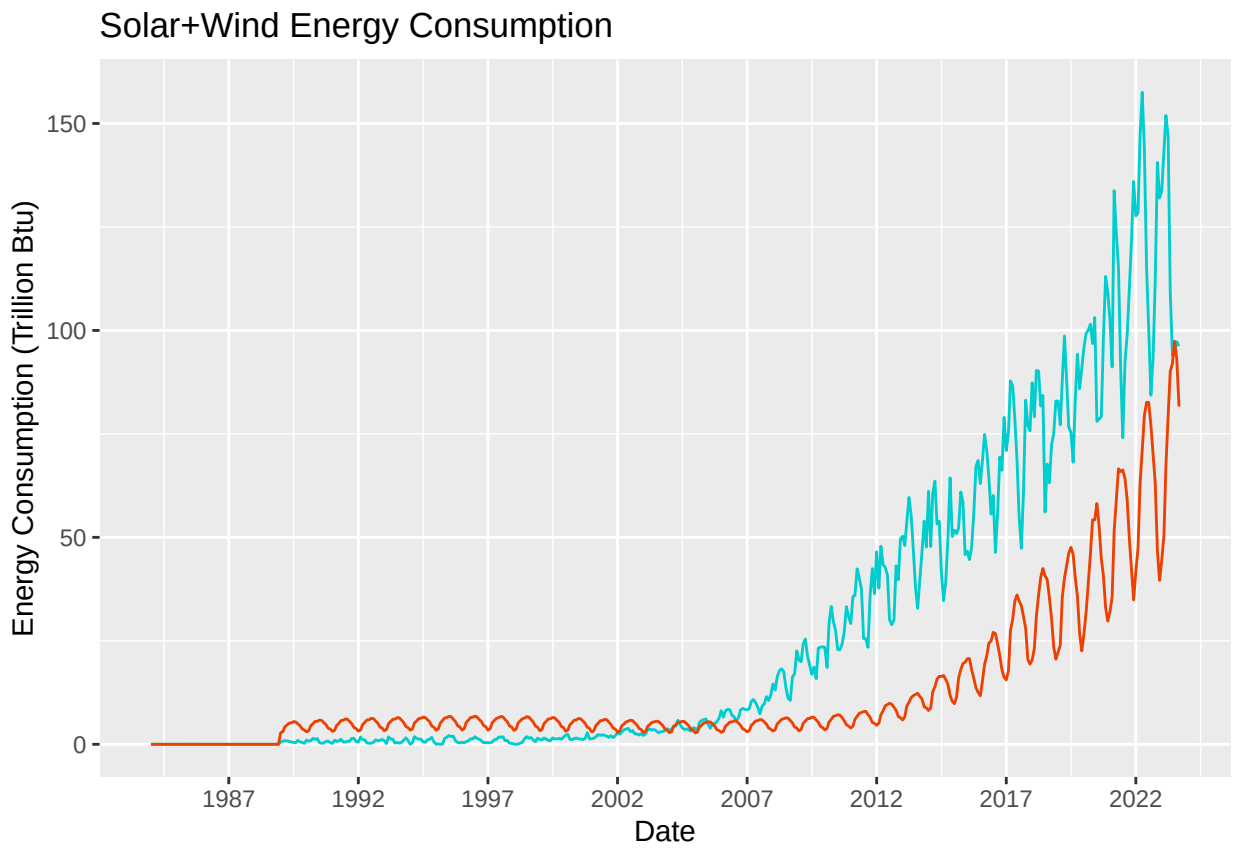

```
wind1 <- ggplot(data=solar_wind, aes(x=date, y=wind))+
  geom_line(color='cyan3')+
  labs(title="Wind Energy Consumption",
       y="Energy Consumption (Trillion Btu)",
       x='Date')+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")

wind1
```



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.

```
sw1 <- ggplot(data=solar_wind)+
  geom_line(aes(x=date, y=wind), color='cyan3')+
  geom_line(aes(x=date, y=solar), color='orangered2')+
  scale_color_manual()+
  labs(title="Solar+Wind Energy Consumption",
       y="Energy Consumption (Trillion Btu)",
       x='Date')+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")

sw1
```



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.

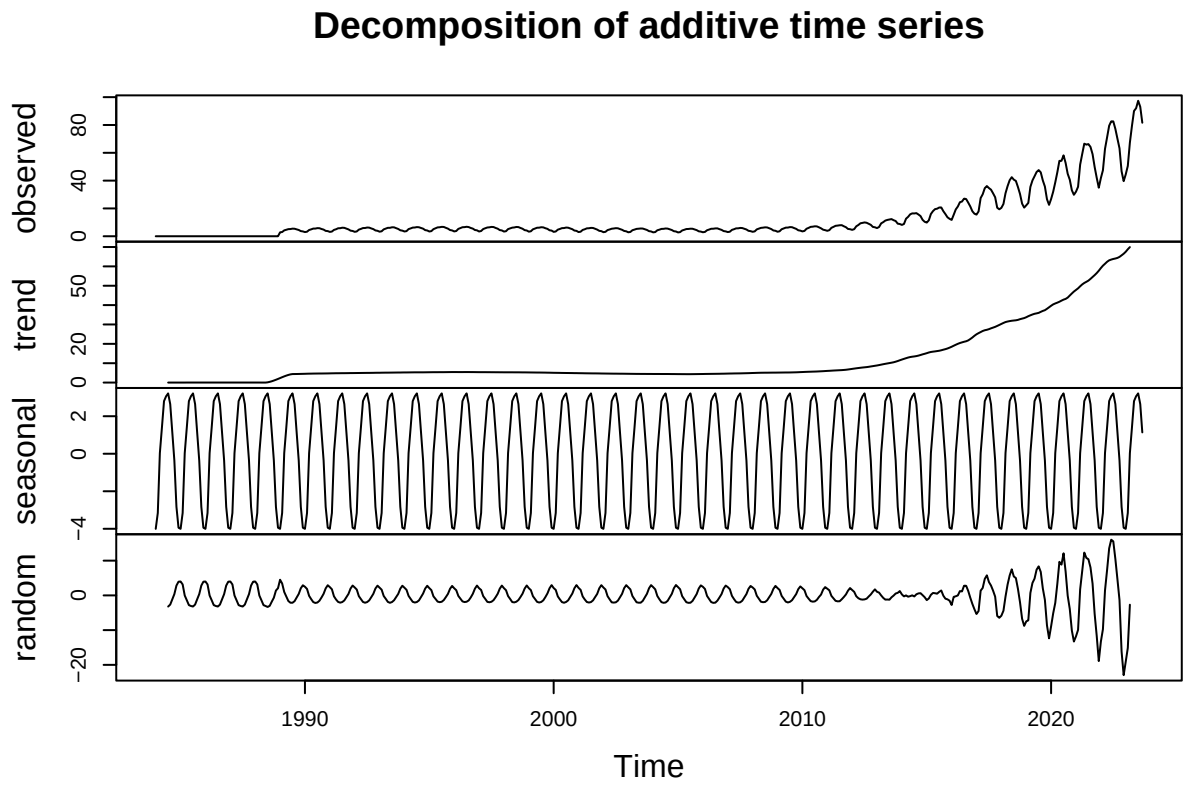
Additional info on `decompose()`.

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.

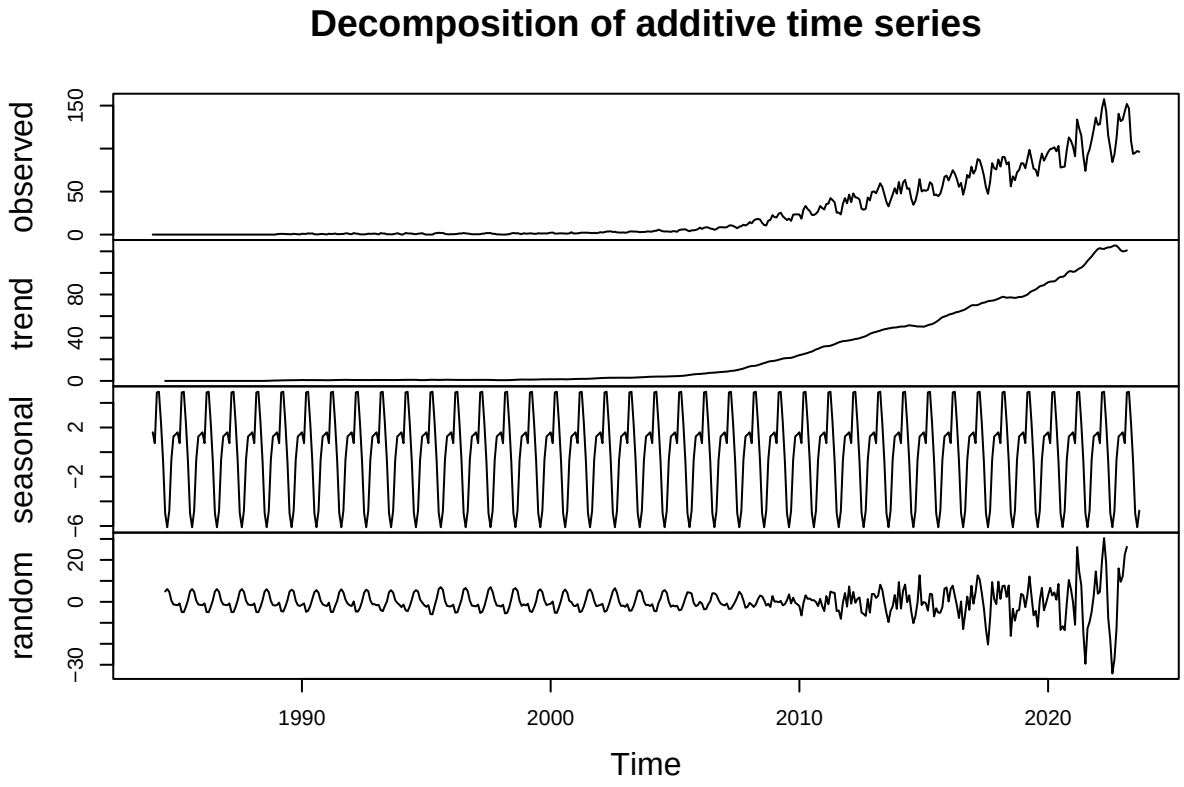
Q4

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_sw <- ts(solar_wind[2:3], start=c(1984,1), frequency=12)
plot(decompose(ts_sw[,1], type="additive")) #Solar
```




```r
plot(decompose(ts_sw[,2], type="additive")) #Wind
```

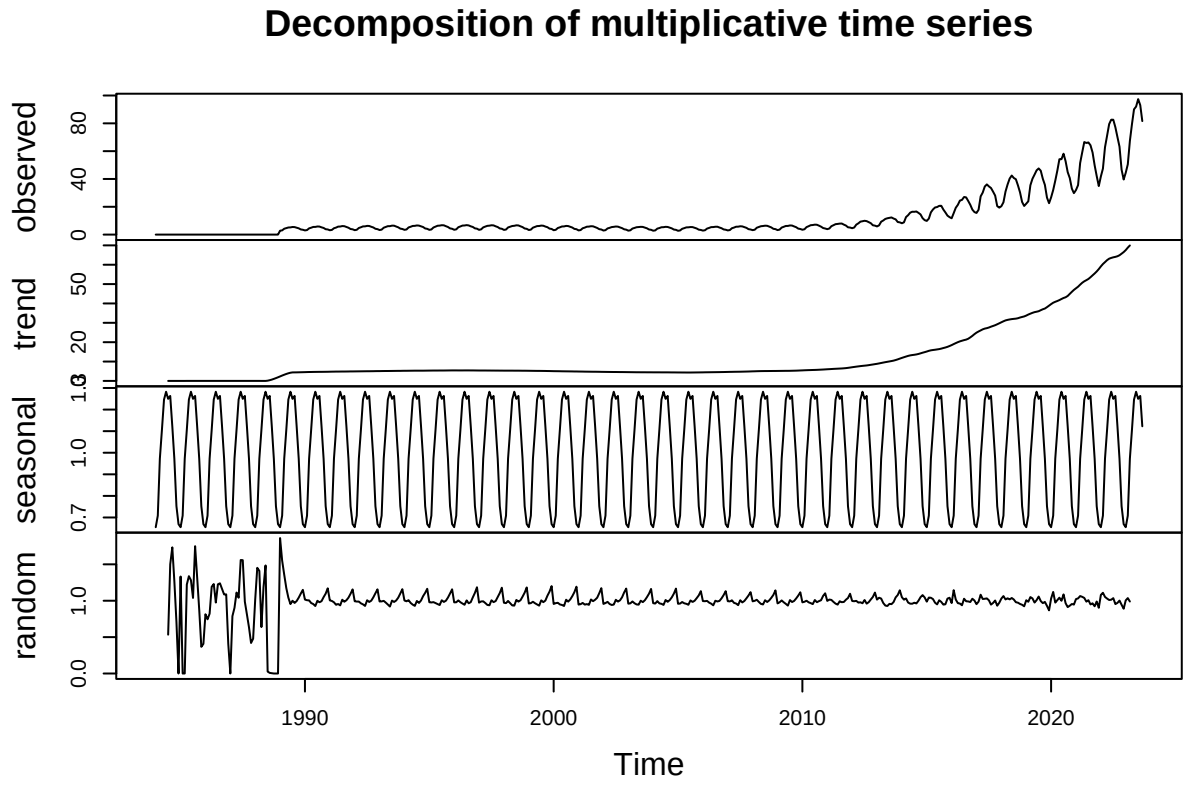


The trend component appears to steadily increase for both time series. Additionally, we can note that the random component appears to have seasonality in both time series, as it fluctuates in a wave pattern. This may indicate that a multiplicative decomposition is better for these series.

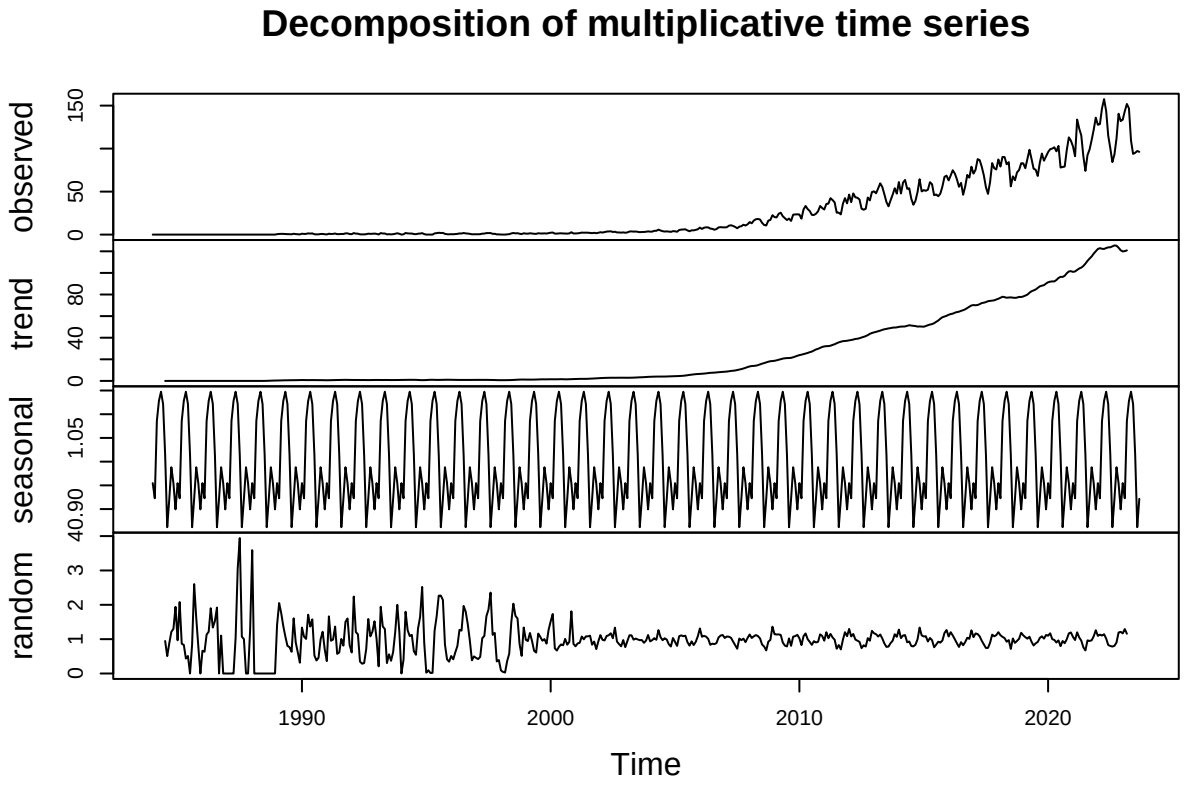
Q5

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?

```
plot(decompose(ts_sw[,1], type="multiplicative")) #Solar
```




```
plot(decompose(ts_sw[,2], type="multiplicative")) #Wind
```



The random component looks much more random with the multiplicative decomposition, with more randomness in the wind. The solar has some minor seasonality remaining, but still looks much better with the multiplicative decomposition.

Q6

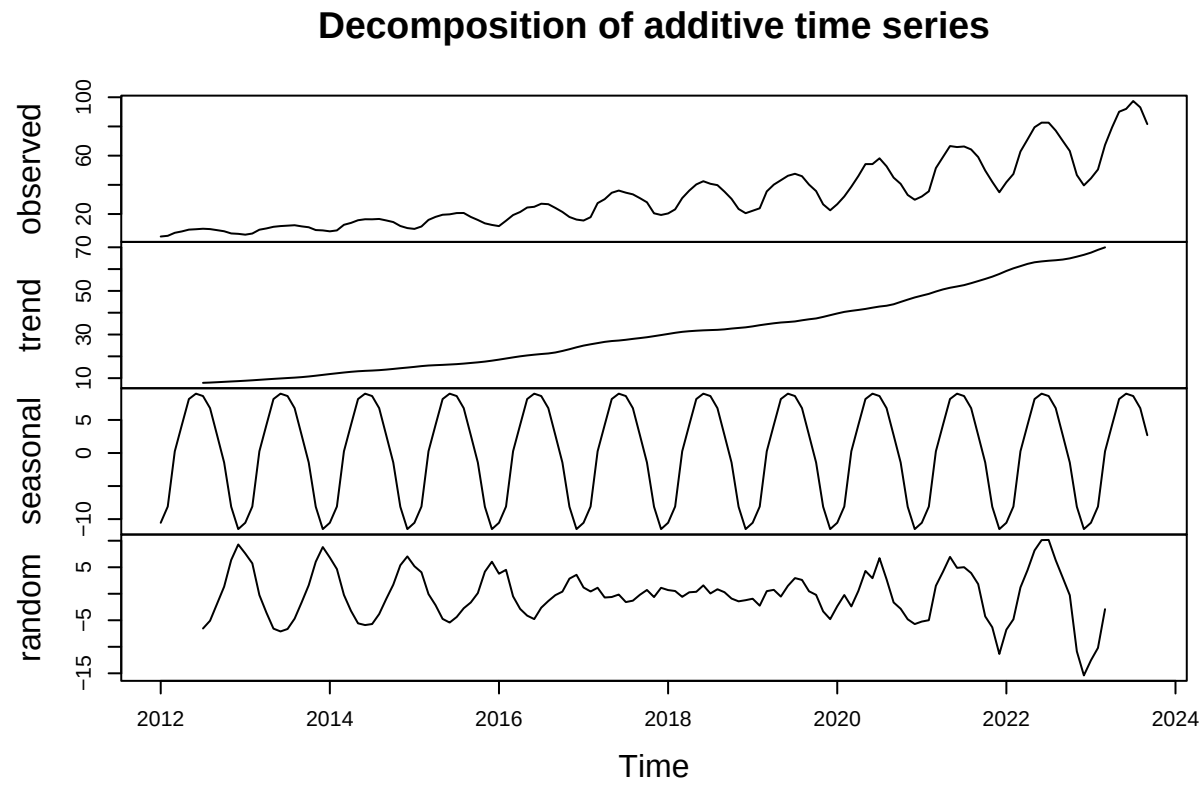
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: It's pretty evident that the solar and wind energy consumption did not truly begin until the mid 2000s. It may be beneficial not to include this data as to more accurately capture the trend, and there is not data there that would be helpful in predicting energy consumption for the next 6 months since there was none.

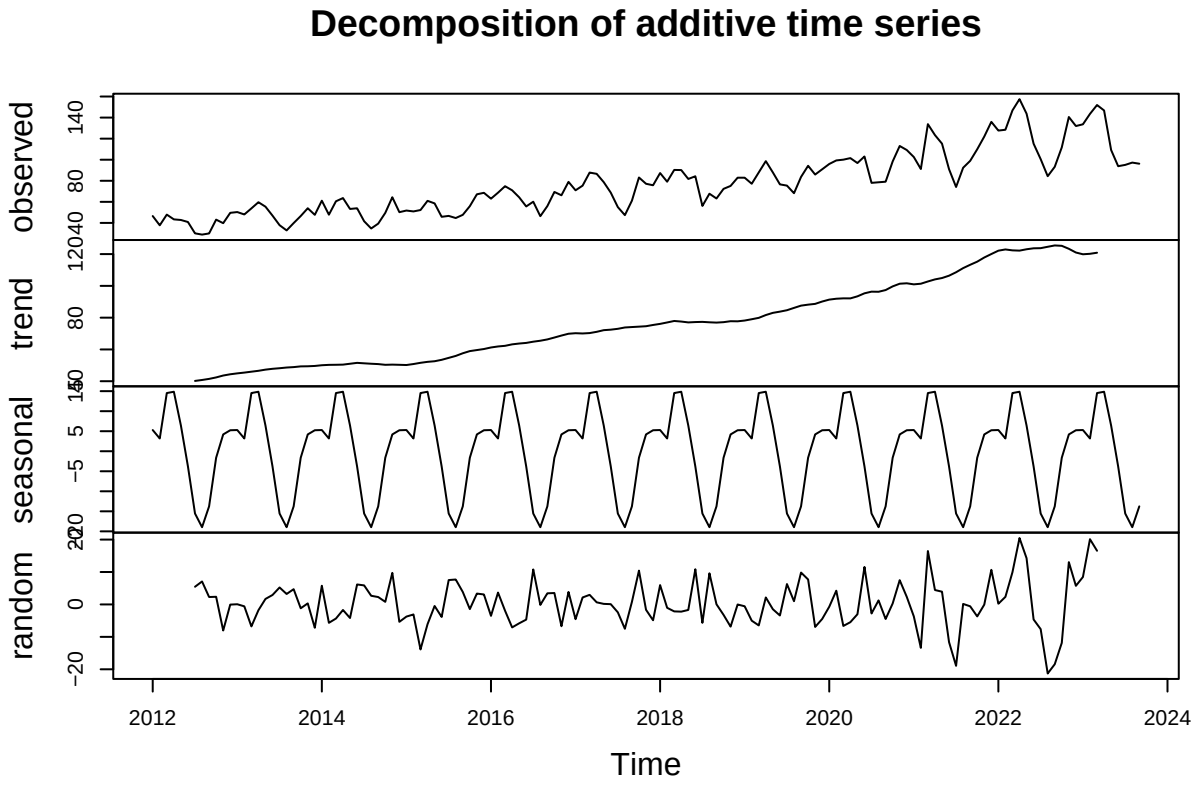
Q7

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.

```r
solar_wind_2012 <- solar_wind %>%
  filter(year(date) >= 2012 )

ts_sw12 <- ts(solar_wind_2012[,2:3], start=c(2012,1), frequency=12)

plot(decompose(ts_sw12[,1], type="additive")) #Solar
```




```r
plot(decompose(ts_sw12[,2], type="additive")) #Wind
```



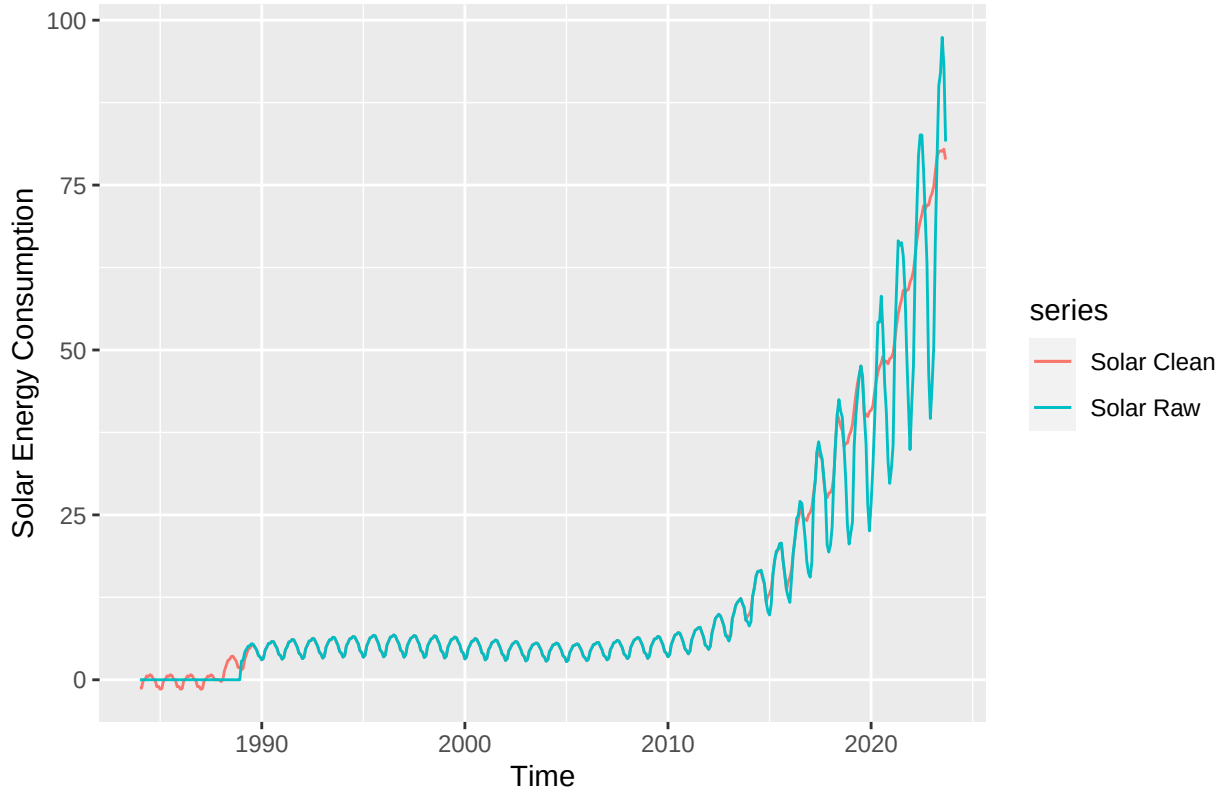
Answer: The random variable does look random for the wind series, but less so for the solar series. It seems that the solar may need the multiplicative decomposition, as the higher levels are showing a pattern in the random plot.

Identify and Remove outliers

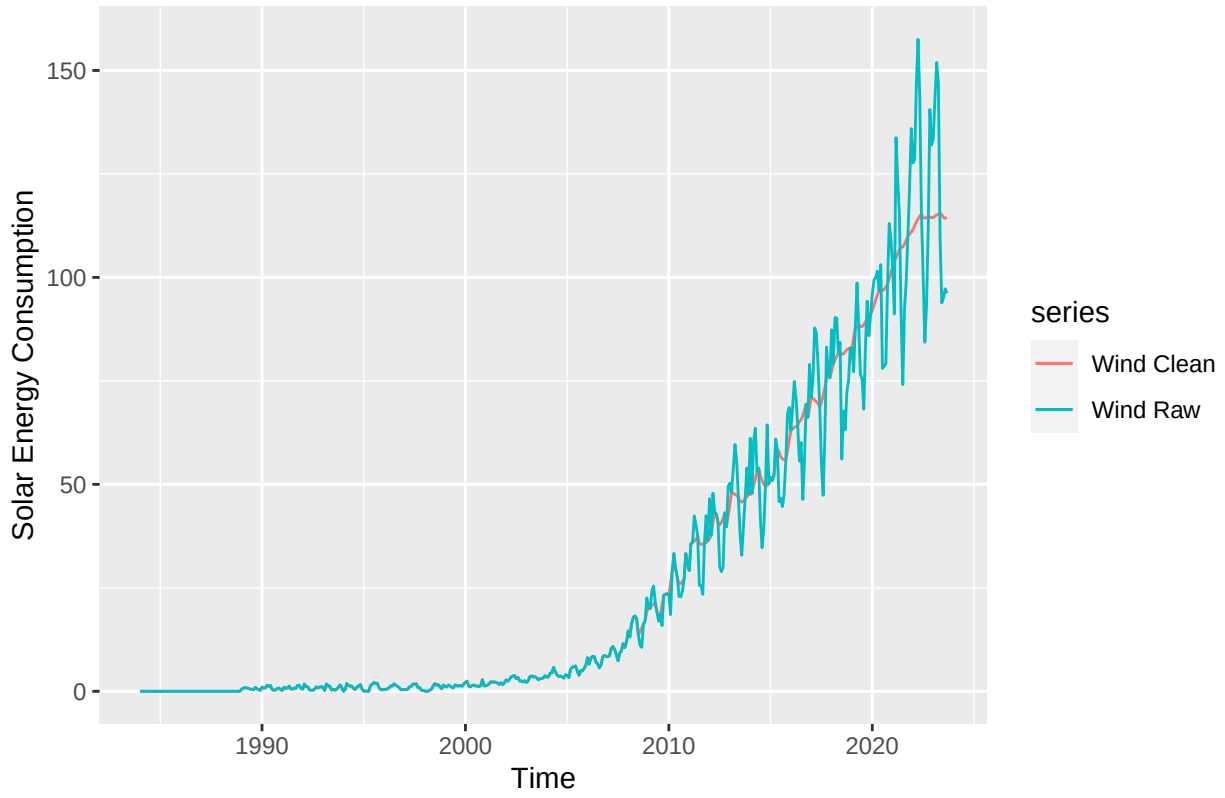
Q8

Apply the `tsclean()` to both series from Q7 (I think this was a typo since Q9 asks for this– I will use the original ts data without cropping for 2012). Did the function removed any outliers from the series? Hint: Use `autoplot()` to check if there is difference between cleaned series and original series.

```
full_clean_solar <- tsclean(ts_sw[,1])
full_clean_wind <- tsclean(ts_sw[,2])

autoplot(full_clean_solar, series="Solar Clean") +
  autolayer(ts_sw[,1], series="Solar Raw") +
  ylab("Solar Energy Consumption")
```




```
autoplot(full_clean_wind, series="Wind Clean") +
  autolayer(ts_sw[,2], series="Wind Raw") +
  ylab("Solar Energy Consumption")
```

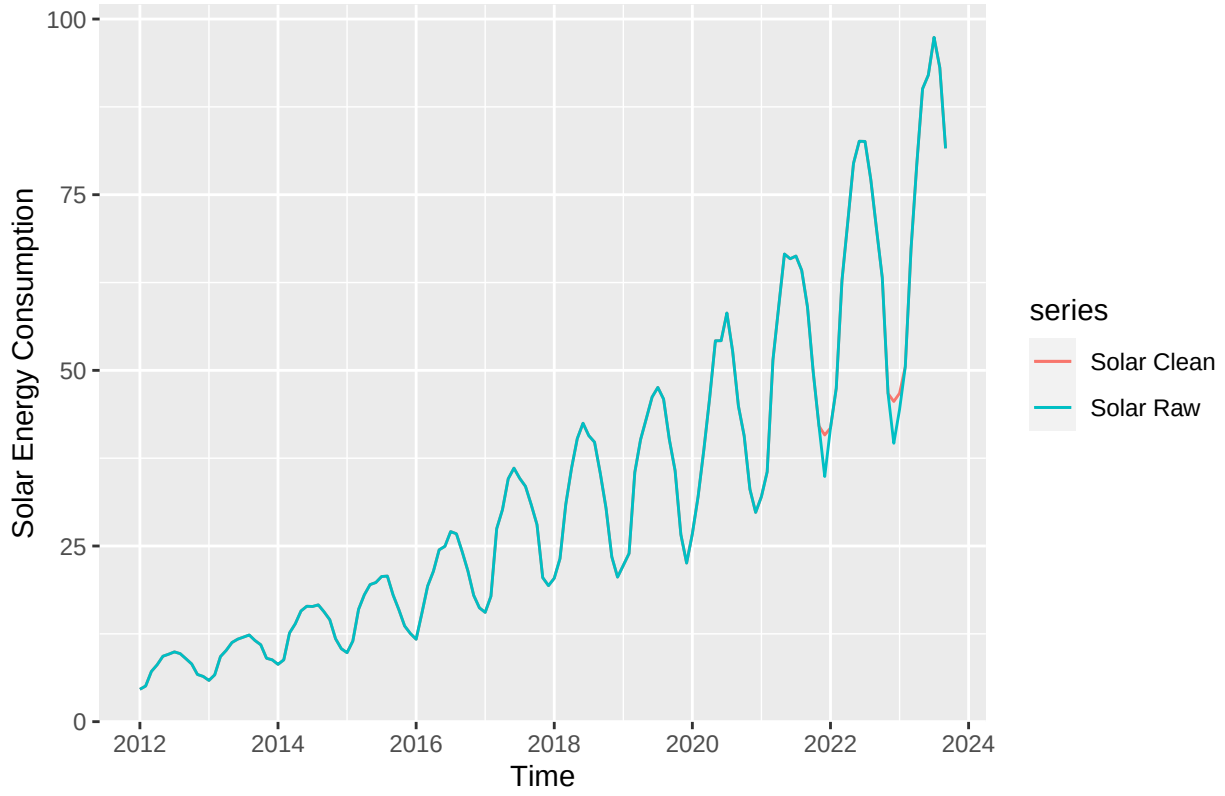


There were many outliers removed from the raw data as shown in the fairly flat trend line starting around 2015, omitting the high peaks and low valleys of this series. The wind series also does this starting around 2010, where the variability is removed.

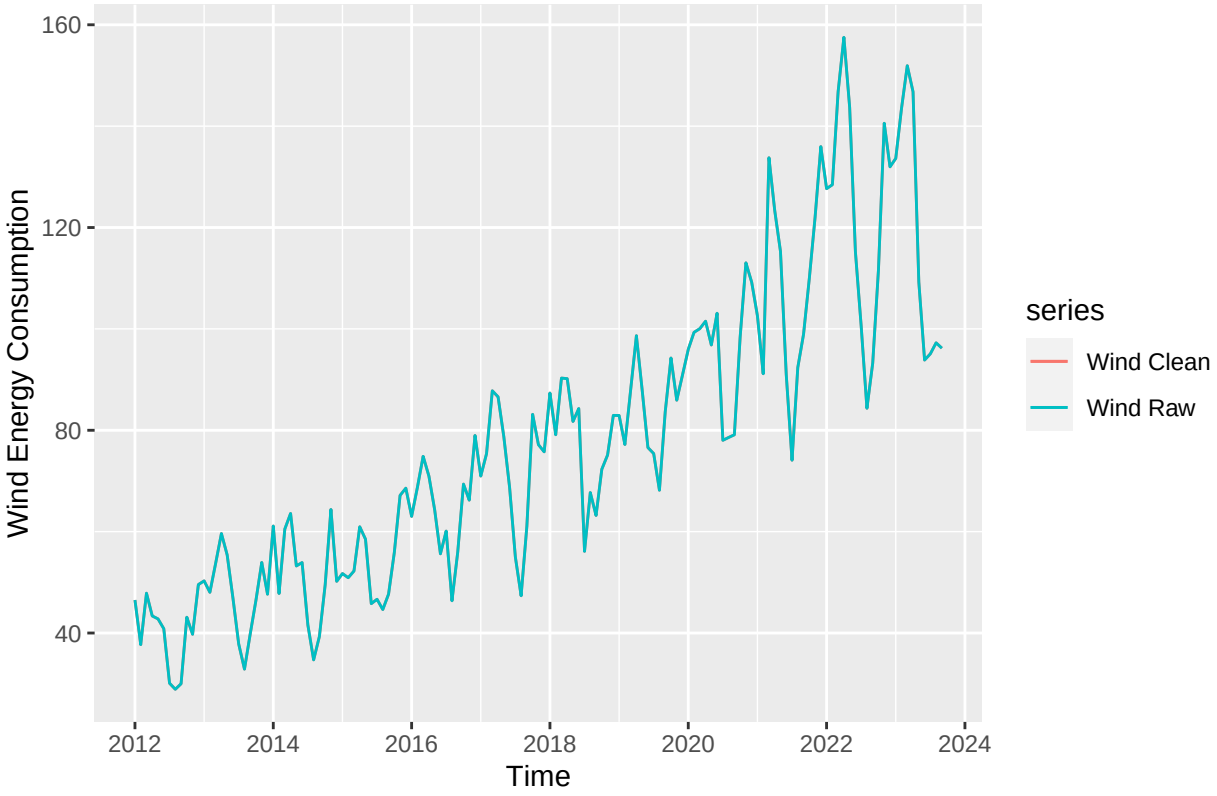
Q9

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?

```
clean_solar <- tsclean(ts_sw12[,1])
clean_wind <- tsclean(ts_sw12[,2])

autoplot(clean_solar, series="Solar Clean") +
  autolayer(ts_sw12[,1], series="Solar Raw") +
  ylab("Solar Energy Consumption")
```




```
autoplot(clean_wind, series="Wind Clean") +
  autolayer(ts_sw12[,2], series="Wind Raw") +
  ylab("Wind Energy Consumption")
```



Answer: The solar series removed 2 outliers, the 2 dips in the winters before 2022 and 2023. I would not recommend removing these points because they seem related to seasonality and would be useful in predicting dips in future years. The wind series does not have any outliers removed. These series have far fewer adjustments from the raw data than the full time series.