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

Assignment 5 - Due date 02/13/24

Sai Powar

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.\

```r
#Load/install required package here
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
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
```

```
library(readxl)
library(dplyr)
```

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.

```
#Importing data set

raw_energy_data <- read.table(file="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Sou

nobs=nrow(raw_energy_data)
nvar=ncol(raw_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!

```
#making data frame for two columns
energy_data <- raw_energy_data[,8:9]

#making date column
energy_date <- ym(raw_energy_data[,1])  #function my from package lubridate
head(energy_date)
```

```
## [1] "1973-01-01" "1973-02-01" "1973-03-01" "1973-04-01" "1973-05-01"
## [6] "1973-06-01"
```

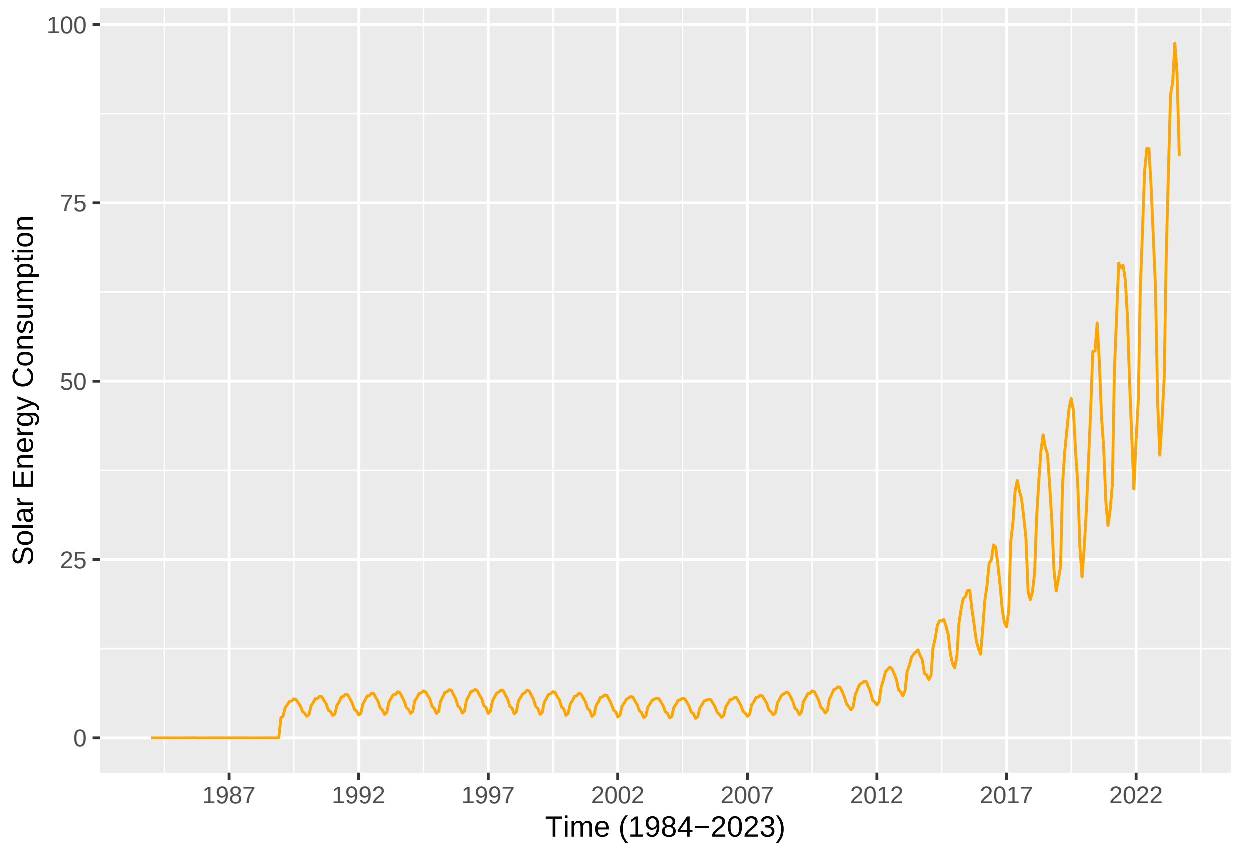
```
energy_data <- cbind(energy_date,energy_data)

energy_data<-energy_data%>%
  filter(energy_data$Solar.Energy.Consumption!='Not Available')
```

```r
#converting to numeric
energy_data$Solar.Energy.Consumption = as.numeric(as.character(energy_data$Solar.Energy.Consumption))
energy_data$Wind.Energy.Consumption = as.numeric(as.character(energy_data$Wind.Energy.Consumption))
```

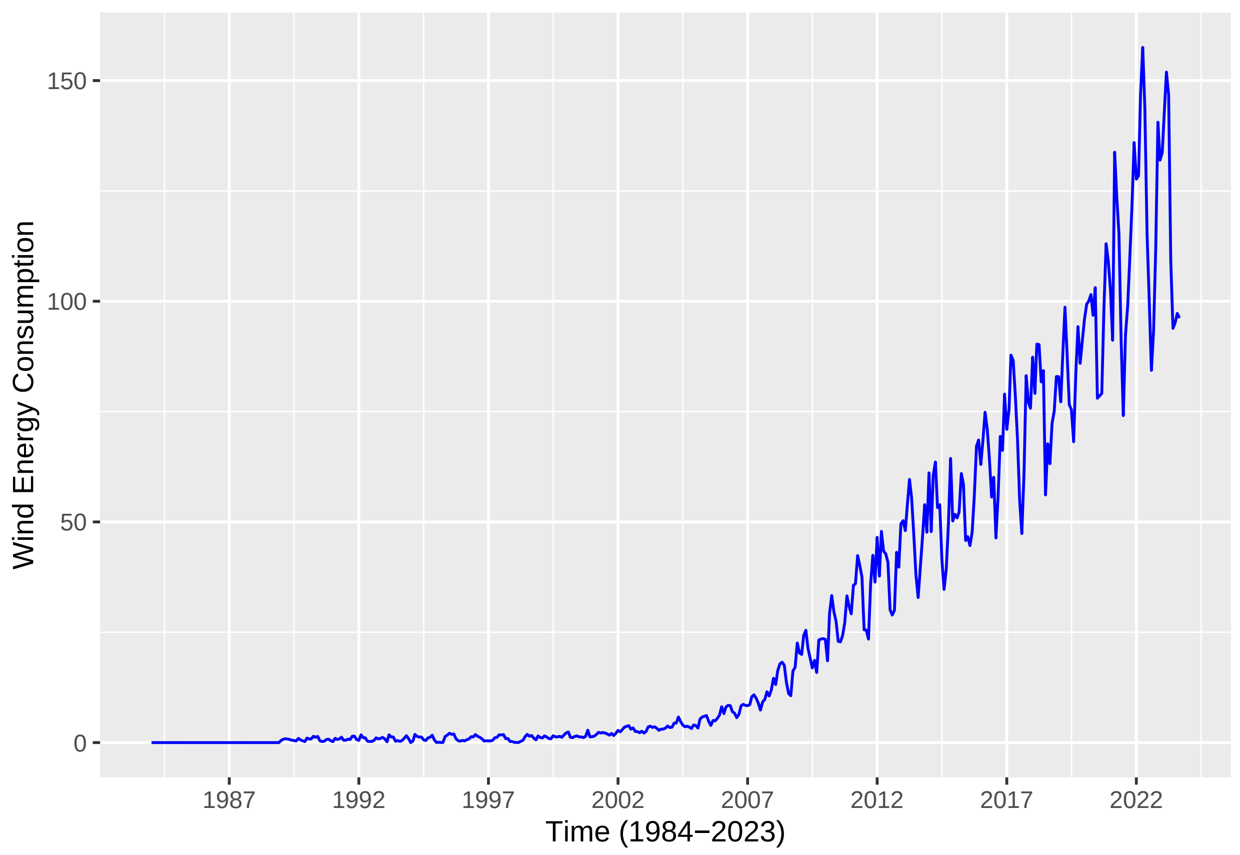
Q2

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
p1 <- ggplot(energy_data, aes(x=energy_date, y=energy_data[,2])) +
        geom_line(color="orange") +
        ylab("Solar Energy Consumption") +
        xlab("Time (1984-2023)")+
        scale_x_date(date_breaks = "5 years", date_labels = "%Y")
p1
```



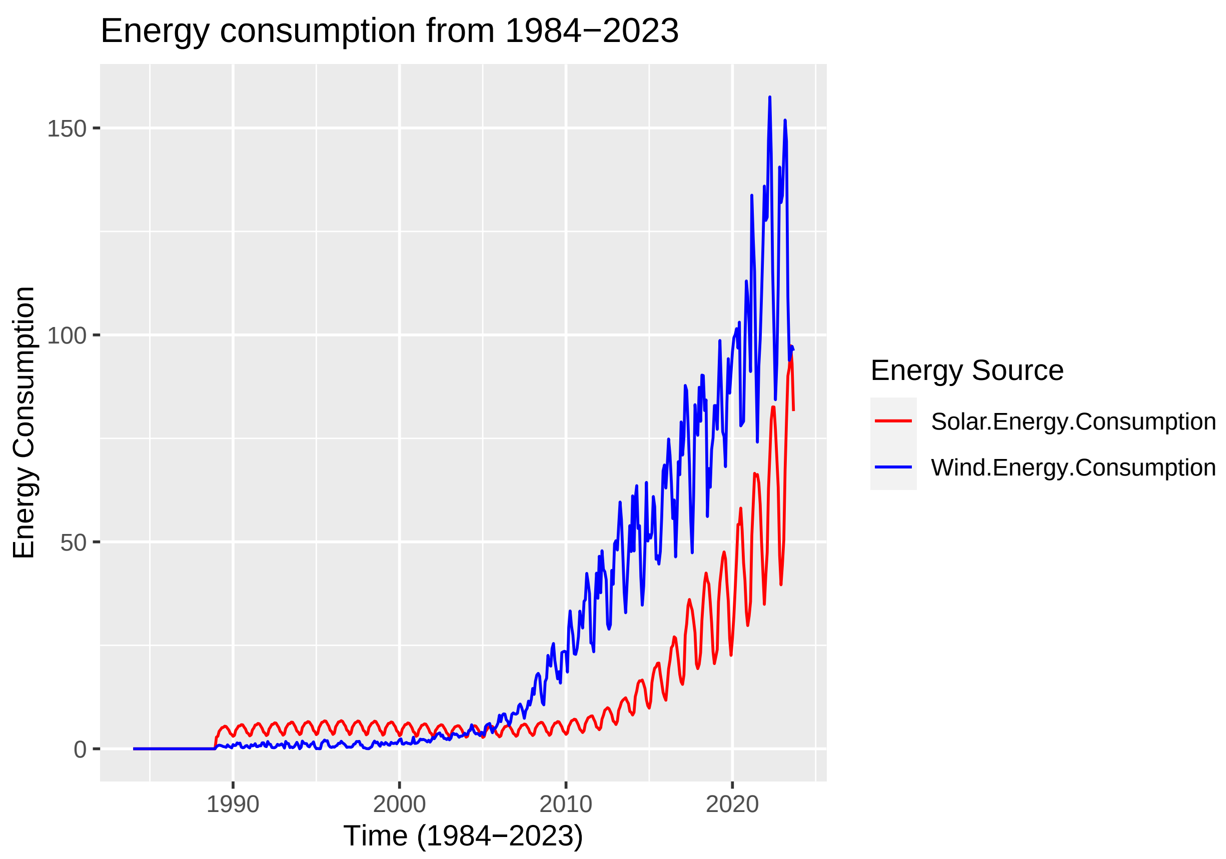
```r
p2 <- ggplot(energy_data, aes(x=energy_date, y=energy_data[,3])) +
        geom_line(color="blue") +
        ylab("Wind Energy Consumption") +
        xlab("Time (1984-2023)")+
        scale_x_date(date_breaks = "5 years", date_labels = "%Y")
p2
```



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.

```
p3 <- energy_data%>%
  ggplot()+
  geom_line(aes(x=energy_date, y=energy_data[,2], color="Solar.Energy.Consumption")) +
            ylab("Energy Consumption") +
            xlab("Time (1984-2023)")+
            geom_line(aes(x=energy_date, y=energy_data[,3], color="Wind.Energy.Consumption"))+
            ggtitle("Energy consumption from 1984-2023")+
            scale_color_manual(name = "Energy Source", values = c("Solar.Energy.Consumption" = "red", "

p3
```



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.

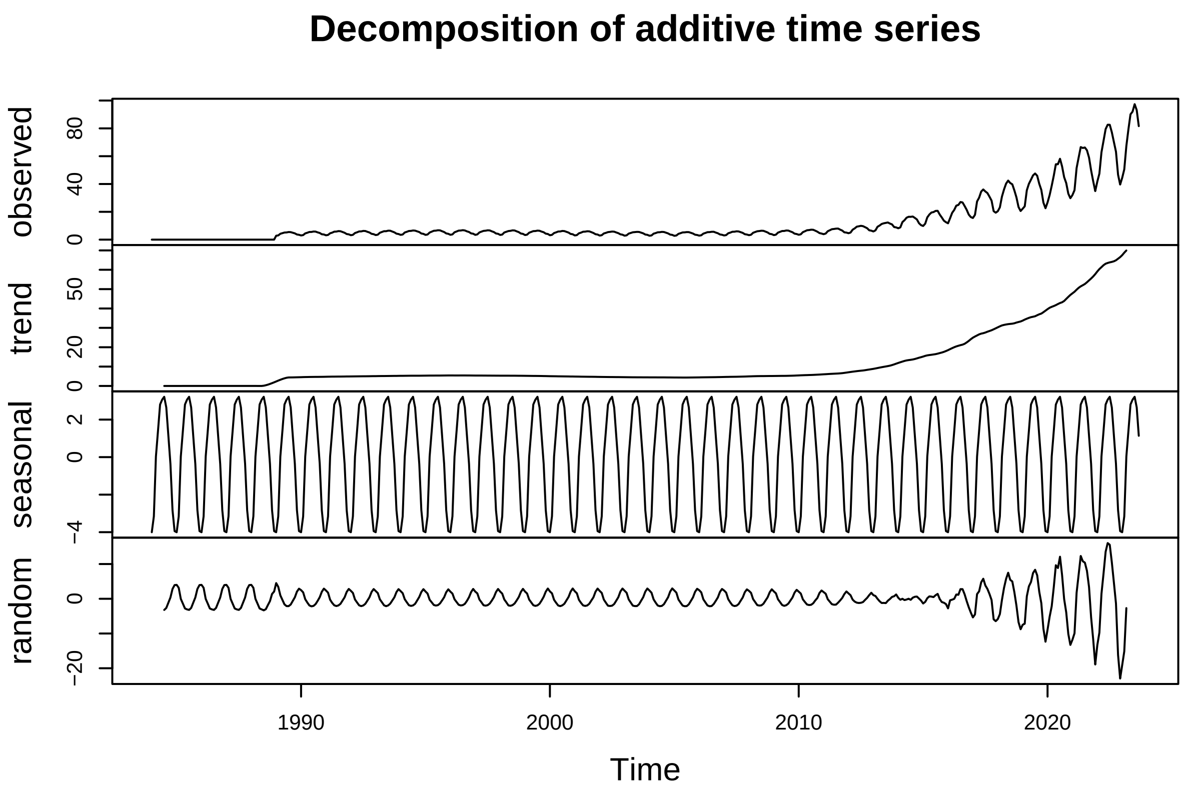
Q4

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?

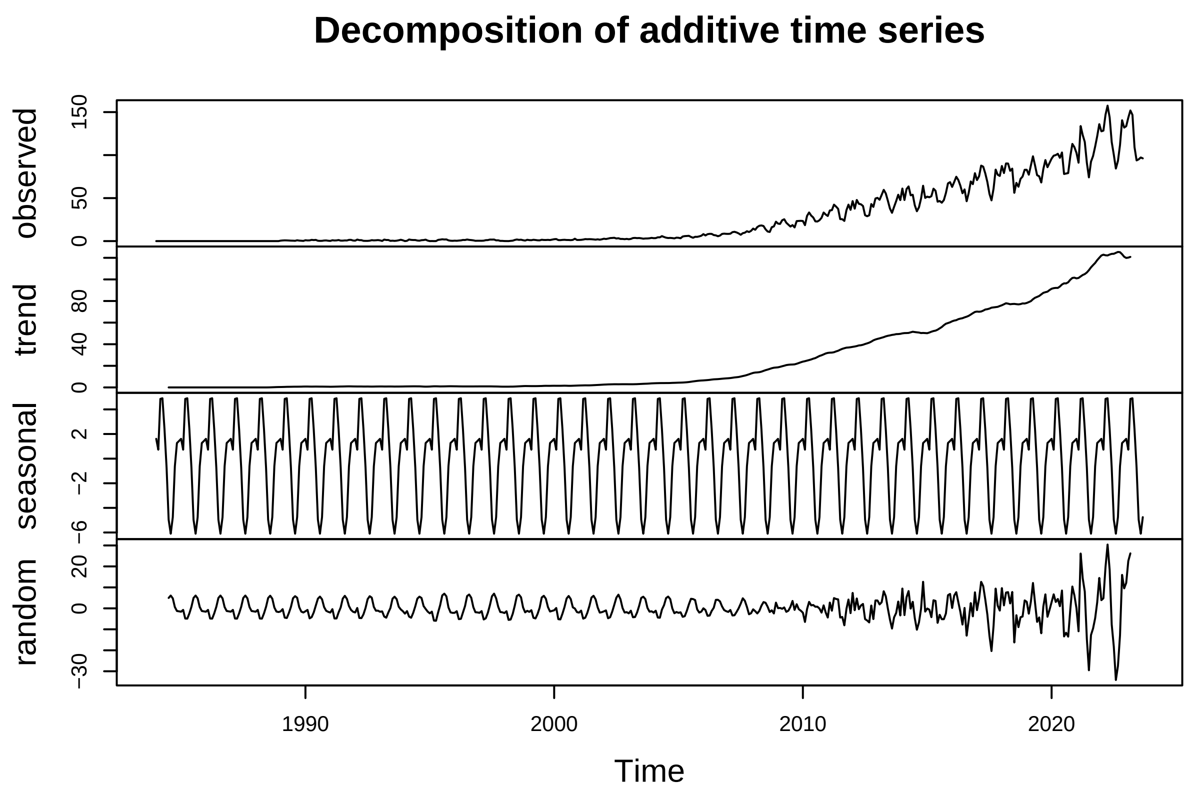
```
ts_re_data <- ts(energy_data[,2:3],start = c(1984,1),frequency=12)
head(ts_re_data)
```

```
##          Solar.Energy.Consumption Wind.Energy.Consumption
## Jan 1984                    0.000                   0.000
## Feb 1984                    0.000                   0.001
## Mar 1984                    0.001                   0.001
## Apr 1984                    0.001                   0.002
## May 1984                    0.002                   0.003
## Jun 1984                    0.003                   0.002
```

```
decompose_solar_add<-decompose(ts_re_data[,1], type="additive")
plot(decompose_solar_add)
```



```
decompose_wind_add<-decompose(ts_re_data[,2], type="additive")
plot(decompose_wind_add)
```

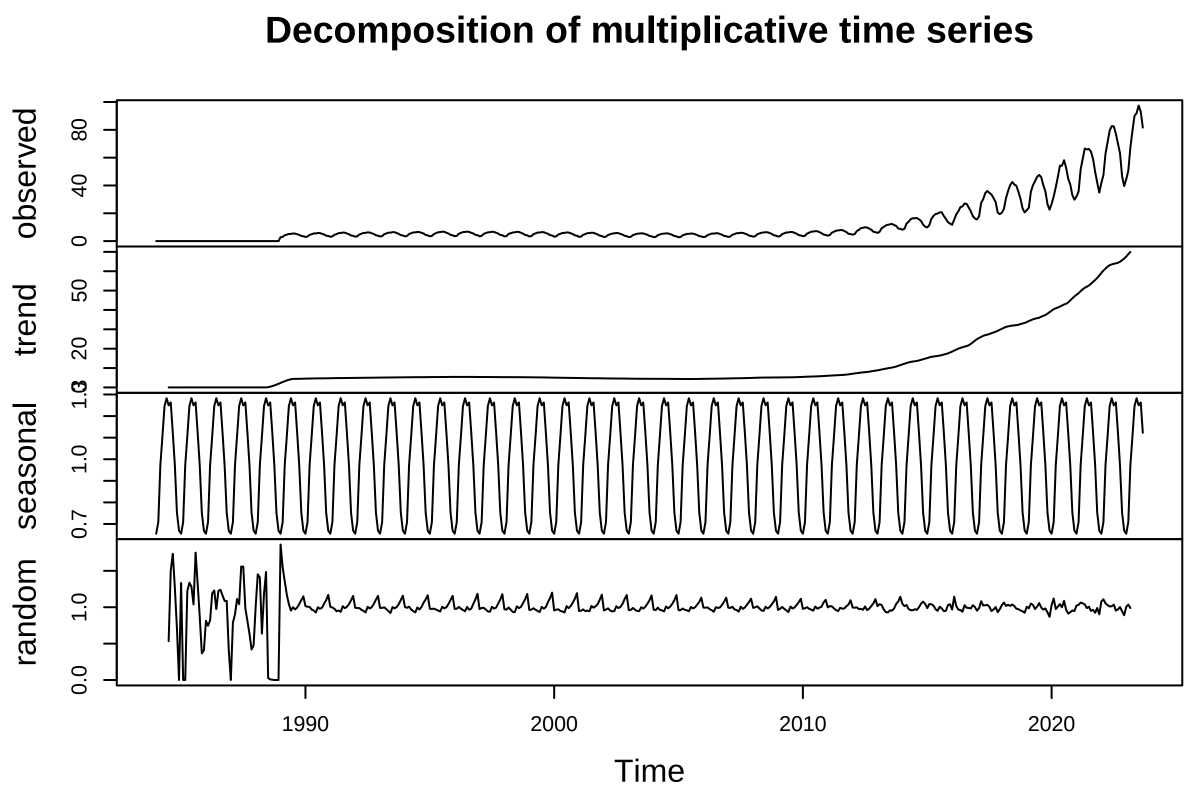


Answer: The trend component is continuous throughout time in terms of an increasing pattern. The random component still seems to have some seasonality and does not appear to be completely random because of the equally spaced patterns.

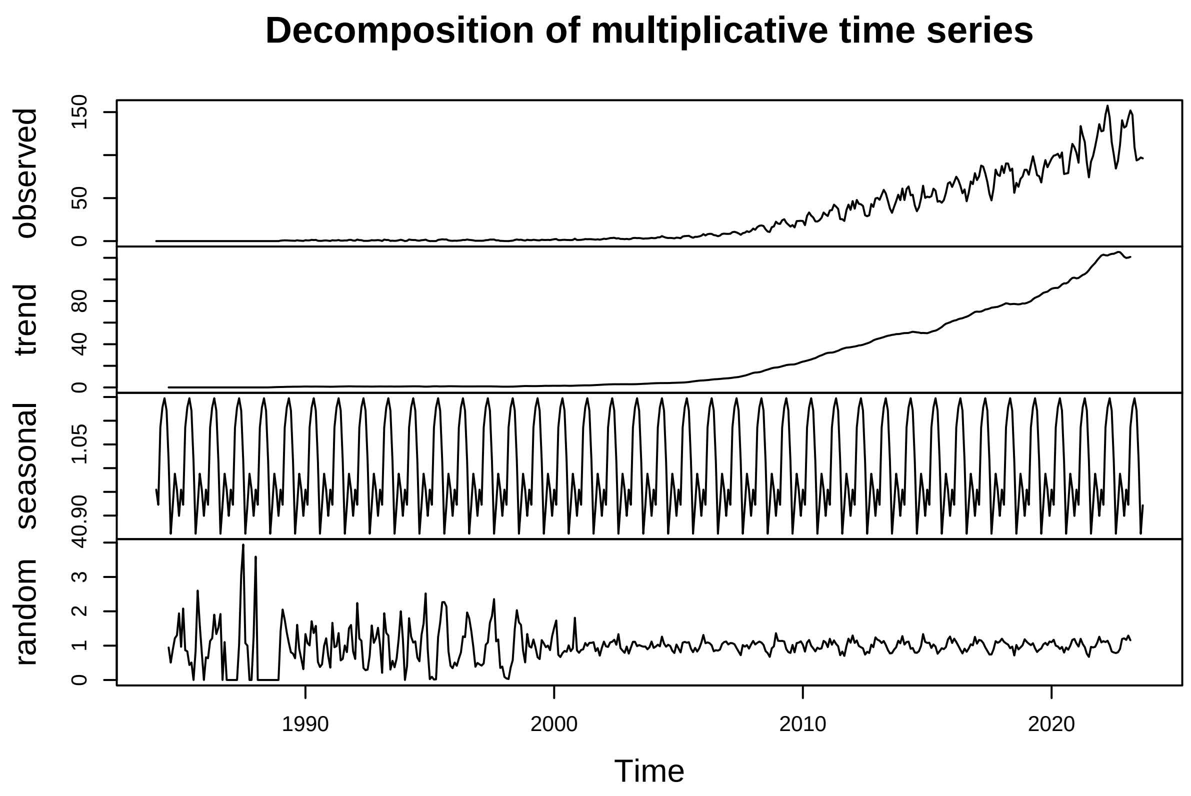
Q5

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?

```r
decompose_solar_mul<-decompose(ts_re_data[,1], type="multiplicative")
plot(decompose_solar_mul)
```



```
decompose_wind_mul<-decompose(ts_re_data[,2], type="multiplicative")
plot(decompose_wind_mul)
```



Answer: For solar, the random component changed completely and seems like the seasonal component has been removed as much as possible. For wind, there is still some seasonality left in the random component in the 2010 to 2020 period.

Q6

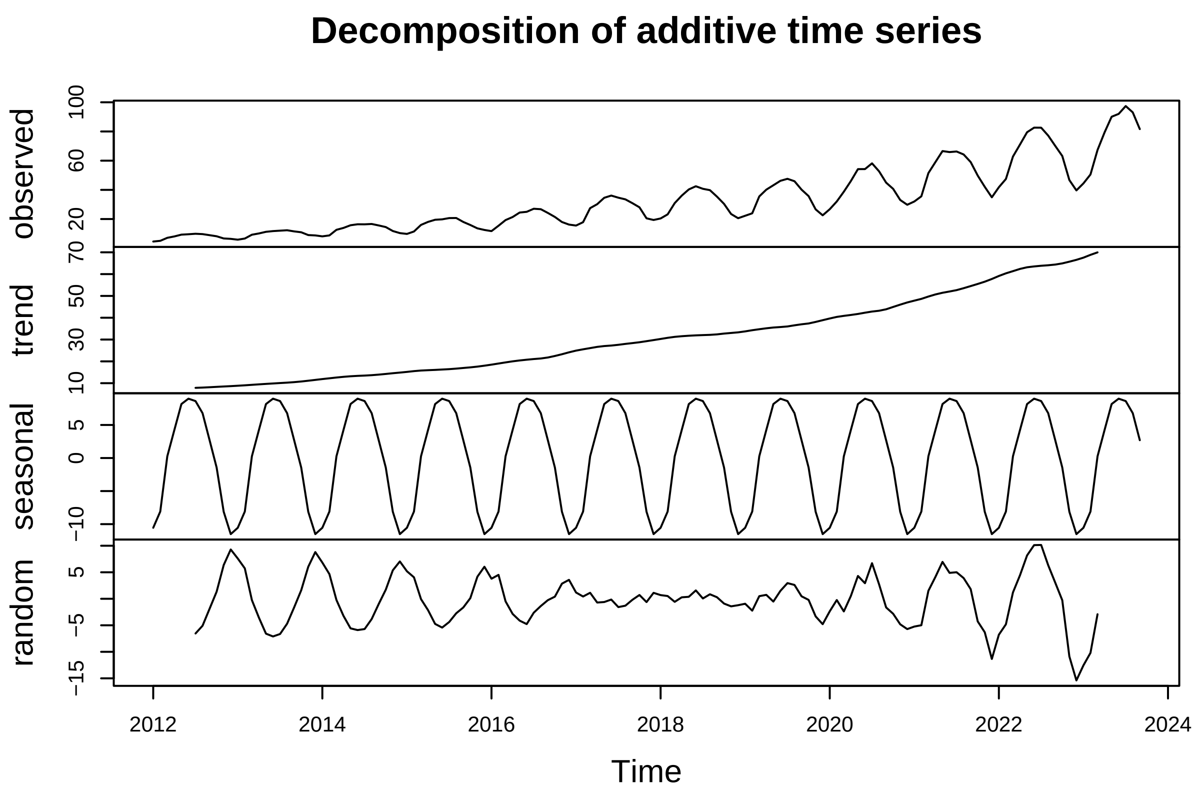
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

Answer: For solar consumption, we may not need all the historical data to forecast for the next six months. There was a significant drop in production around 2012-13 and then the production has been steadily increasing with no outright event/charactertistic. Since we are only projecting for six months and the seasonal component has remained constant throughout, I think using historical data from 2012-13 onwards should suffice. For wind consumption, there was a drop in production around 2019 that followed a downward and slightly picked up in 2023. In the observed series (and the random component), there is a lot of variability post 2020 that we need to take into account for the next six months. Here as well, I don't think data before 2008-2009 is necesssary for 6 month projections. The consumption has reached a new level after that year and hasn't gone below it or had that type of pattern post 2008. So, I think for wind consumptiiion using data from 2008-09 onwards should suffice.

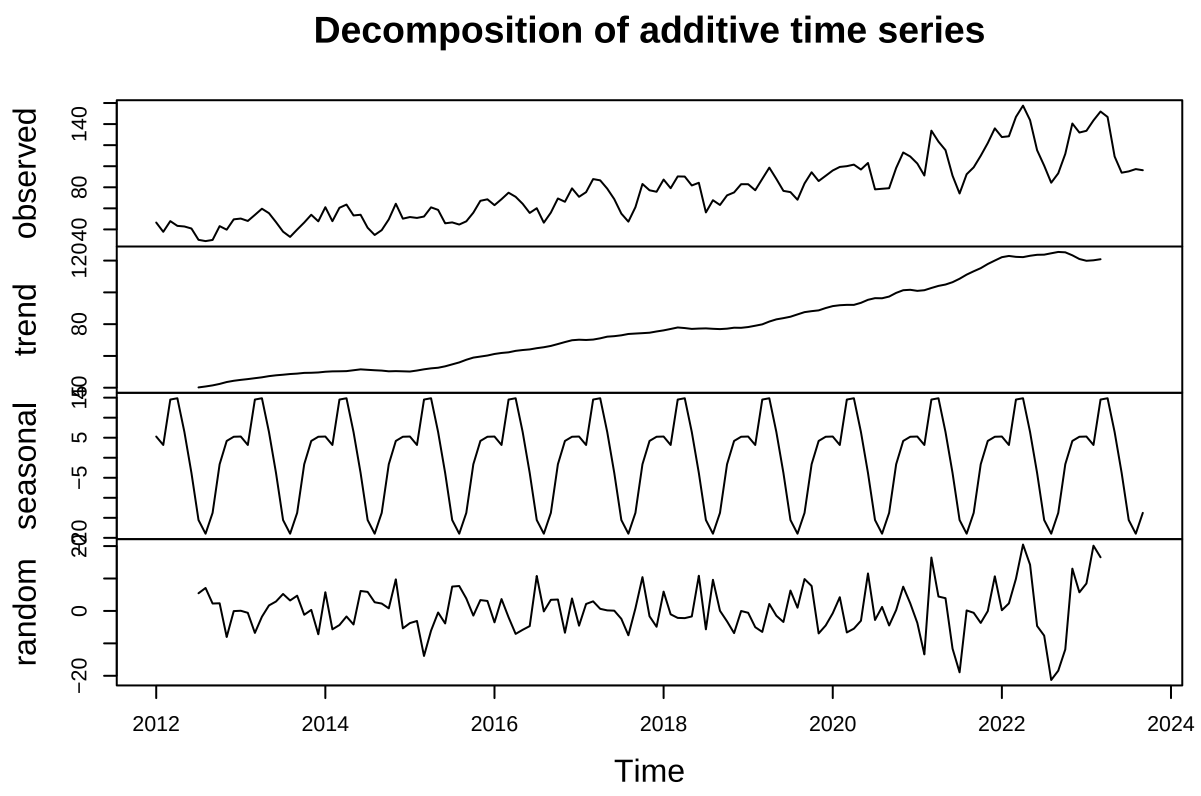
Q7

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.

```
energy_data_2012 <- energy_data%>%
  filter(year(energy_date)>=2012)

ts_re_data_2012 <- ts(energy_data_2012[,2:3],start = c(2012,1),frequency=12)

decompose_solar_2012<-decompose(ts_re_data_2012[,1], type="additive")
plot(decompose_solar_2012)
```



```
decompose_wind_2012<-decompose(ts_re_data_2012[,2], type="additive")
plot(decompose_wind_2012)
```



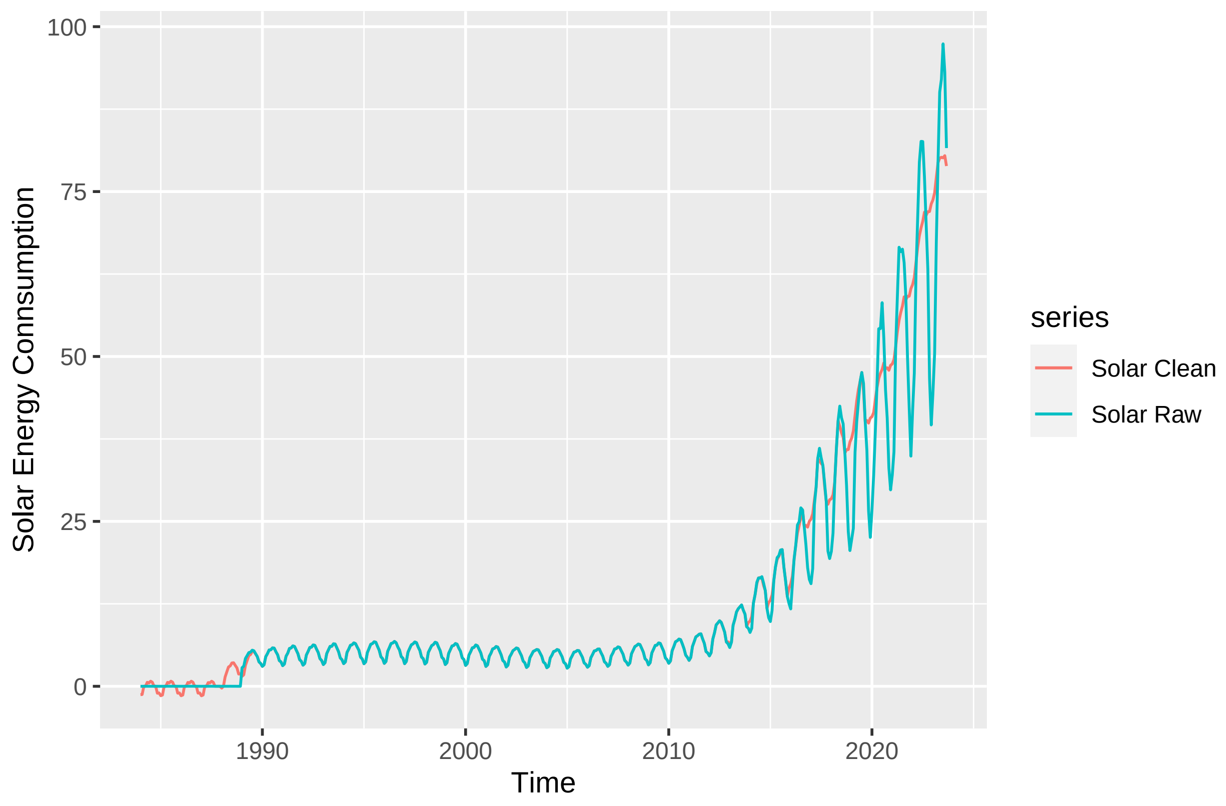
Answer: The random component for the solar consumption series doesn't necessarily look random. The magnitude of the random component wave is increasing with time, especially from 2017. For wind consumption, the random component looks more "random" than solar in the 2020-2023 period. Before that, it looks there is some seasonality remaining that could be dependent on the level of the series.

Identify and Remove outliers

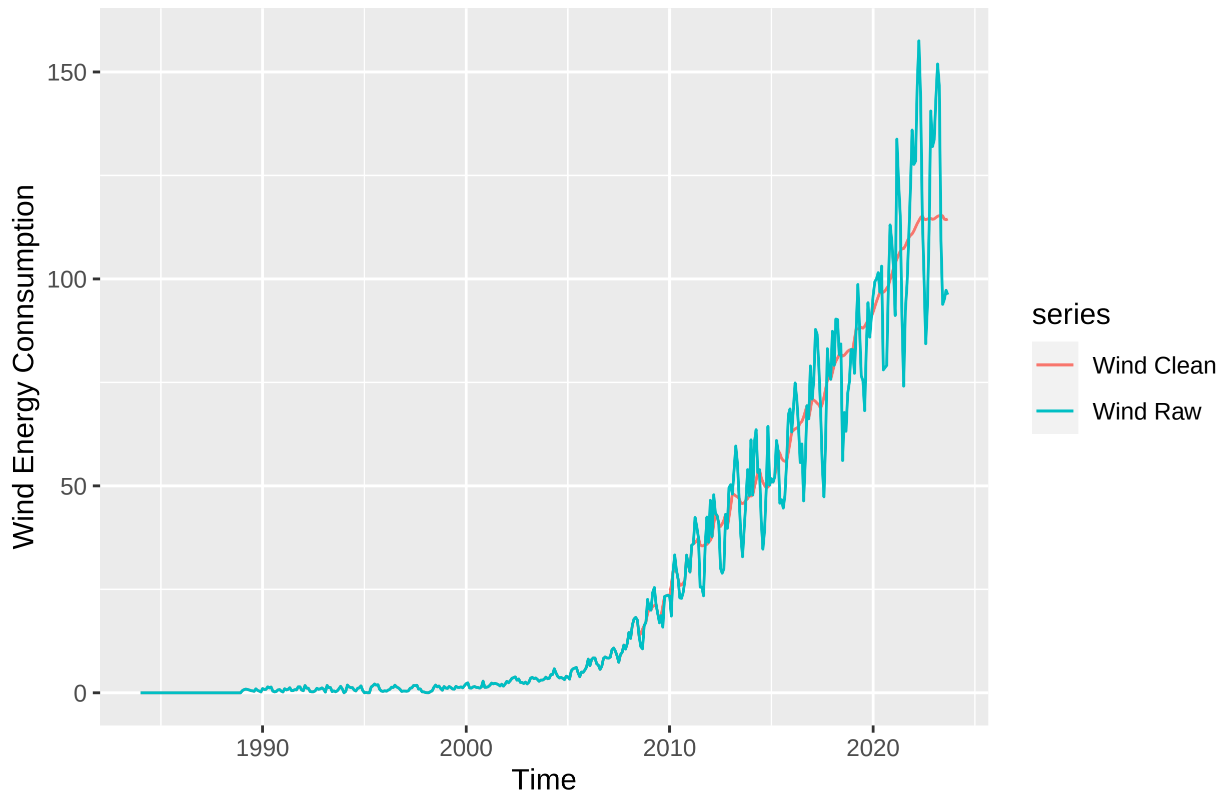
Q8

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.

```r
clean_solar_data <- tsclean(ts_re_data[,1])

p4<- autoplot(clean_solar_data, series="Solar Clean") +
  autolayer(ts_re_data[,1], series="Solar Raw") +
  ylab("Solar Energy Connsumption")
p4
```



```
clean_wind_data <- tsclean(ts_re_data[,2])

p5<- autoplot(clean_wind_data, series="Wind Clean") +
  autolayer(ts_re_data[,2], series="Wind Raw") +
  ylab("Wind Energy Connsumption")
p5
```

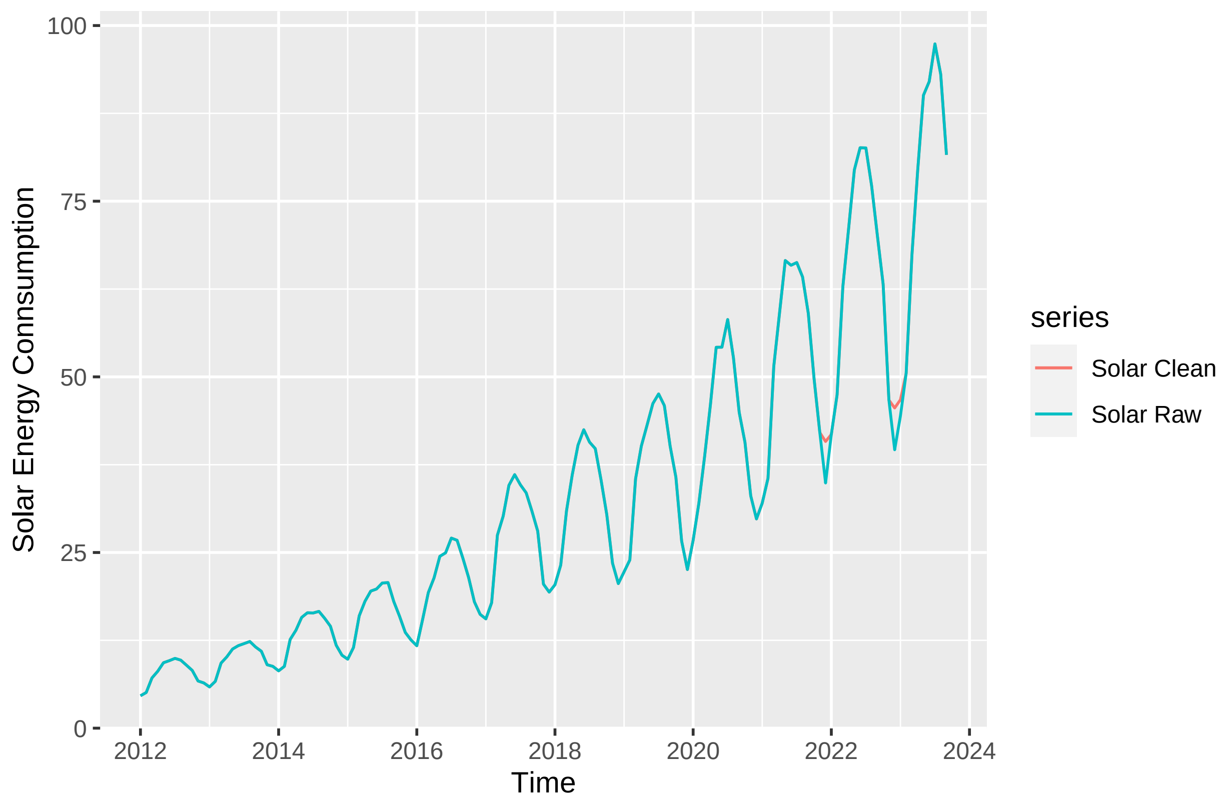


Answer: Cleaning the data got rid of a lot of the variation in the data from 2010 onwards, which we probably don't want to remove because this is needed for future forecasting given the way the energy market functions now.

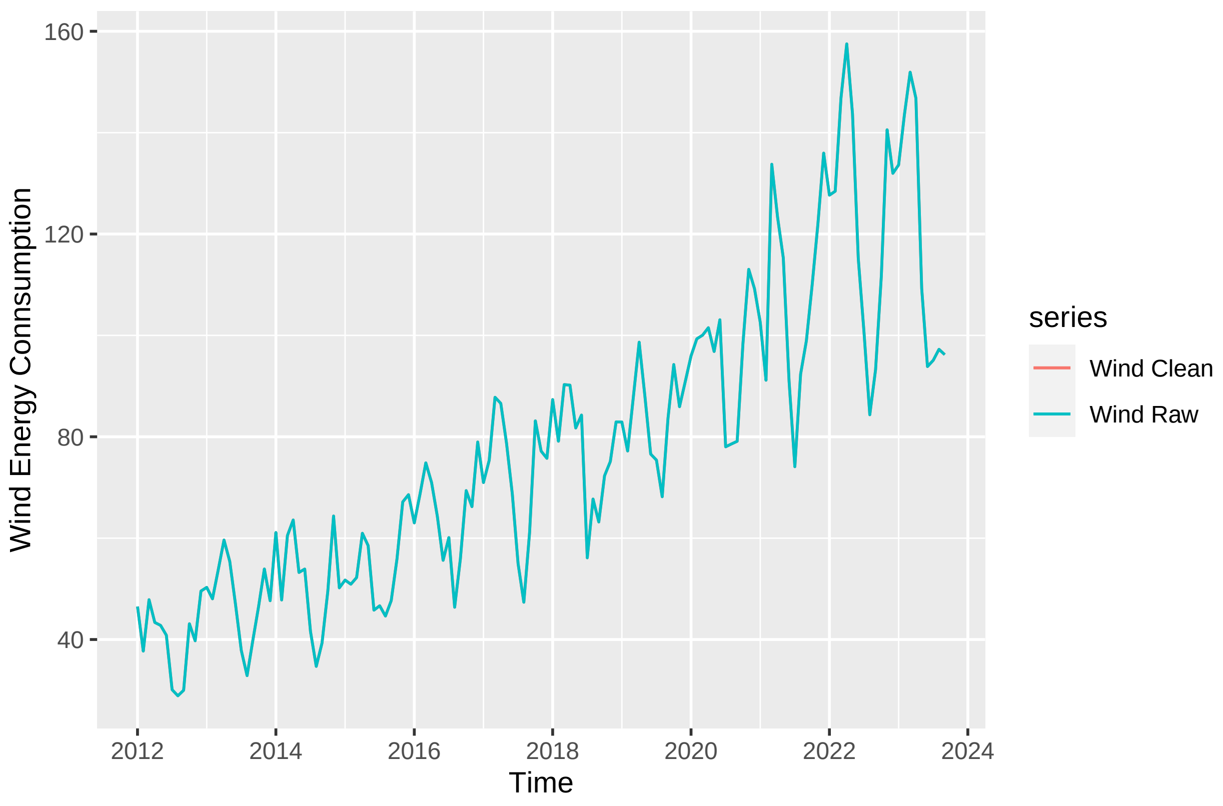
Q9

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?

```r
clean_solar_2012 <- tsclean(ts_re_data_2012[,1])

p6<- autoplot(clean_solar_2012, series="Solar Clean") +
  autolayer(ts_re_data_2012[,1], series="Solar Raw") +
  ylab("Solar Energy Connsumption")
p6
```



```
clean_wind_2012 <- tsclean(ts_re_data_2012[,2])
p7<- autoplot(clean_wind_2012, series="Wind Clean") +
  autolayer(ts_re_data_2012[,2], series="Wind Raw") +
  ylab("Wind Energy Connsumption")
p7
```



Answer: There are no changes in the cleaned vs. raw series for both solar and wind consumption.