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

Assignment 5 - Due date 02/13/24

Emma Kaufman

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
```

```
#install.packages("xlsx")
library(readxl)
library(ggthemes)
```

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 - using xlsx 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`
```

```
#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
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!

```
#selecting desired columns
solar_and_wind <- 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`),
         `Month` = ymd(`Month`)) %>%
  na.omit()
```

```
## 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.
```

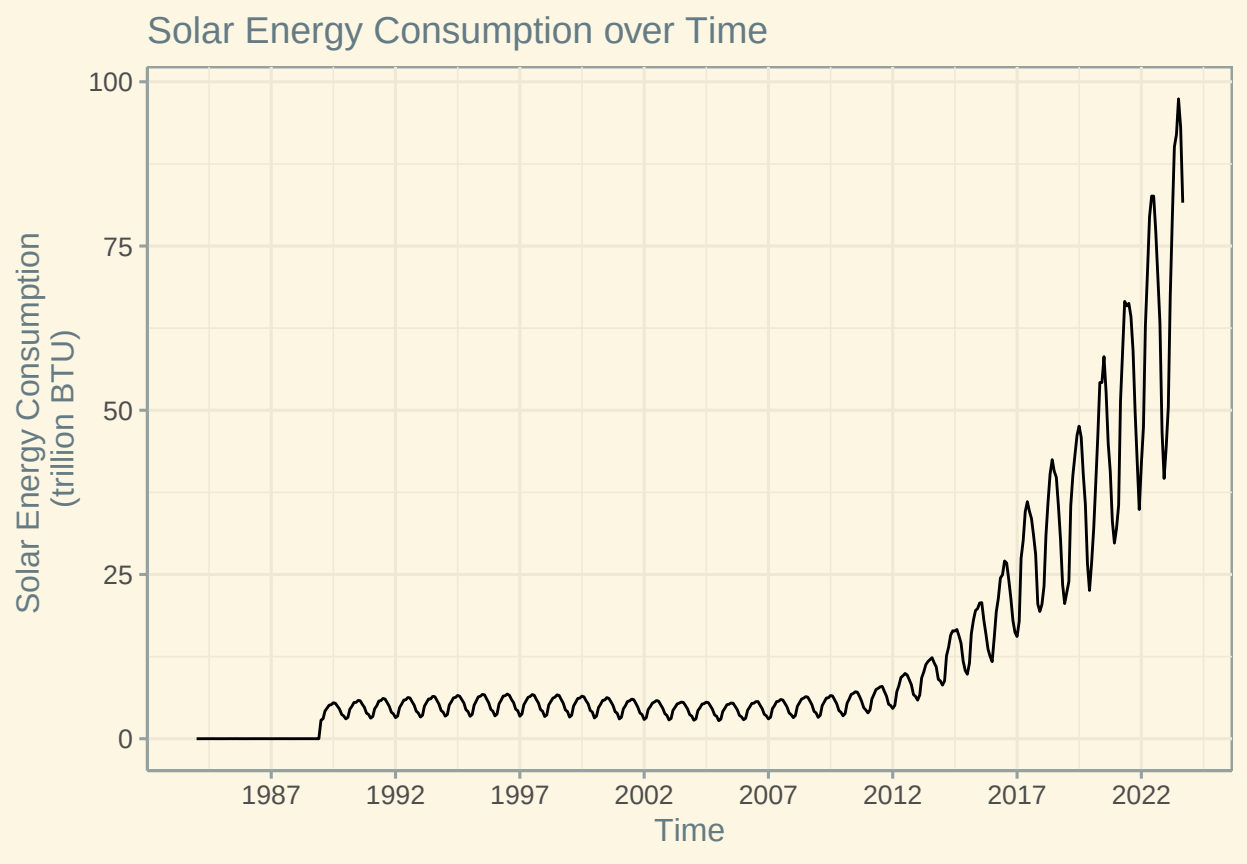
```
#previewing new dataframe
head(solar_and_wind)
```

```
## # A tibble: 6 x 3
##   Month      ‘Solar Energy Consumption‘ ‘Wind Energy Consumption‘
##   <date>                          <dbl>                     <dbl>
## 1 1984-01-01                      0                         0
## 2 1984-02-01                      0                         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
```

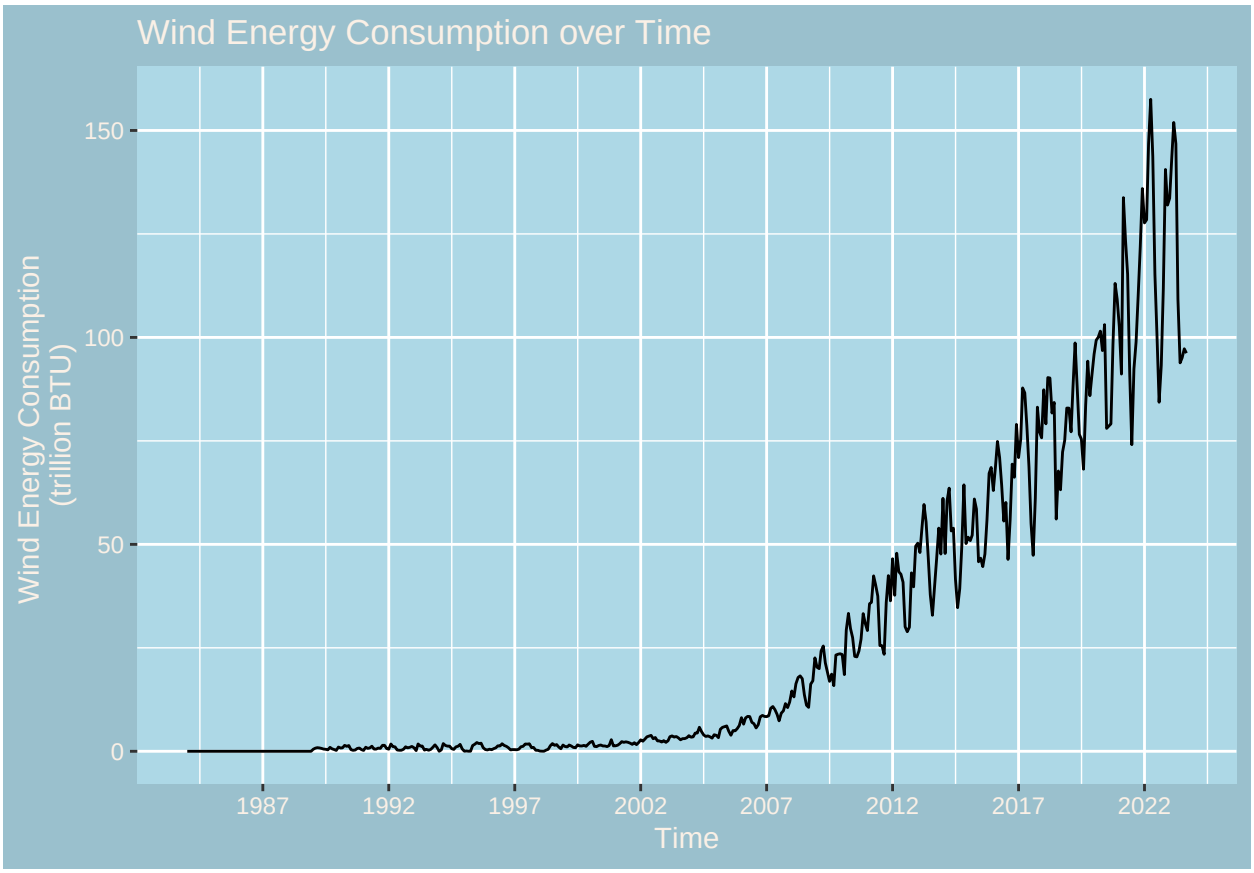
Q2

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")")`

```
Solar_consumption <- ggplot((solar_and_wind),
                            aes(x=Month,
                                y=`Solar Energy Consumption`)) +
  geom_line() +
  labs(x= 'Time',
       y= 'Solar Energy Consumption \n(trillion BTU)',
       title= 'Solar Energy Consumption over Time')+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  theme_solarized()

Solar_consumption
```



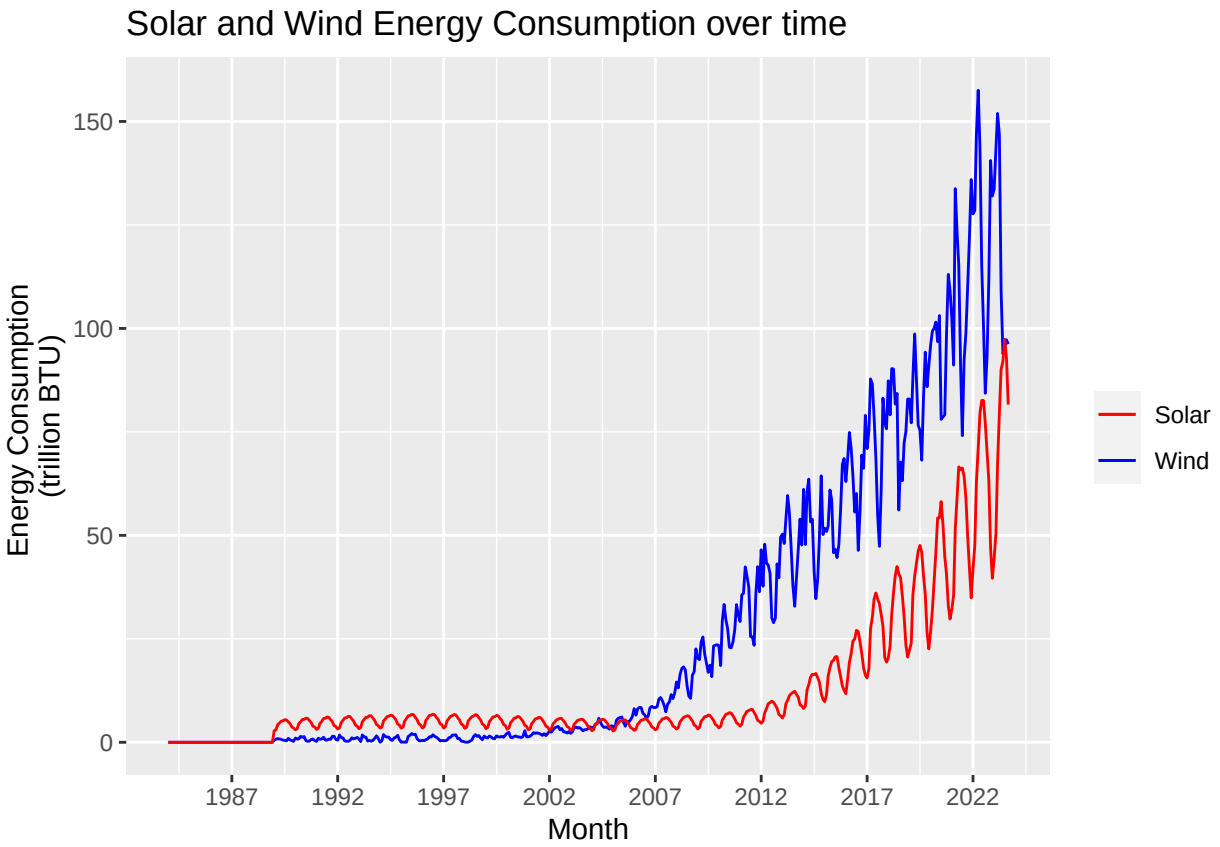

```
wind_theme <- theme_update(
  plot.background = element_rect(fill = "lightblue3", colour = NA),
  panel.background = element_rect(fill = "lightblue", colour = NA),
  axis.text = element_text(colour = "linen"),
  axis.title = element_text(colour = "linen"),
  title=element_text(colour = "linen")
)

Wind_consumption <- ggplot((solar_and_wind),
                           aes(x=Month,
                               y=`Wind Energy Consumption`)) +
  geom_line() +
  labs(x= 'Time',
       y= 'Wind Energy Consumption \n(trillion BTU)',
       title= 'Wind Energy Consumption over Time')+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  theme_set(wind_theme)

Wind_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 datasets with ggplot()
Wind_and_solar_plot <- ggplot((solar_and_wind),
                              aes(x=Month))+
  geom_line(aes(y=`Wind Energy Consumption`,
                color="Wind"))+
  geom_line(aes(y= `Solar Energy Consumption`,
                color="Solar"))+
  labs(y='Energy Consumption \n(trillion BTU)')+
  scale_x_date(date_breaks = "5 years",
               date_labels = "%Y")+
  scale_colour_manual(values=c("Solar"="red",
                      "Wind"="blue"))+
  labs(color="",
       title='Solar and Wind Energy Consumption over time')

Wind_and_solar_plot
```



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.

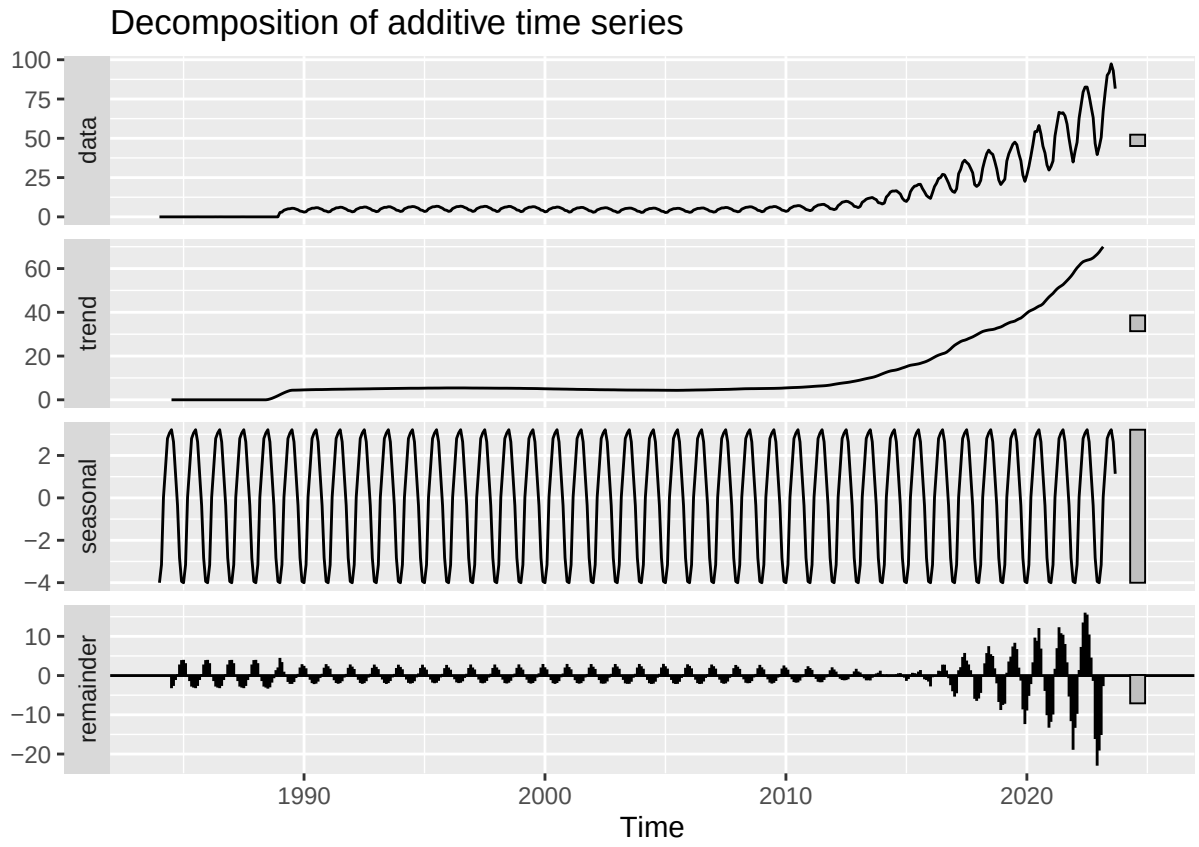
Additional info on `decompose()`.

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.

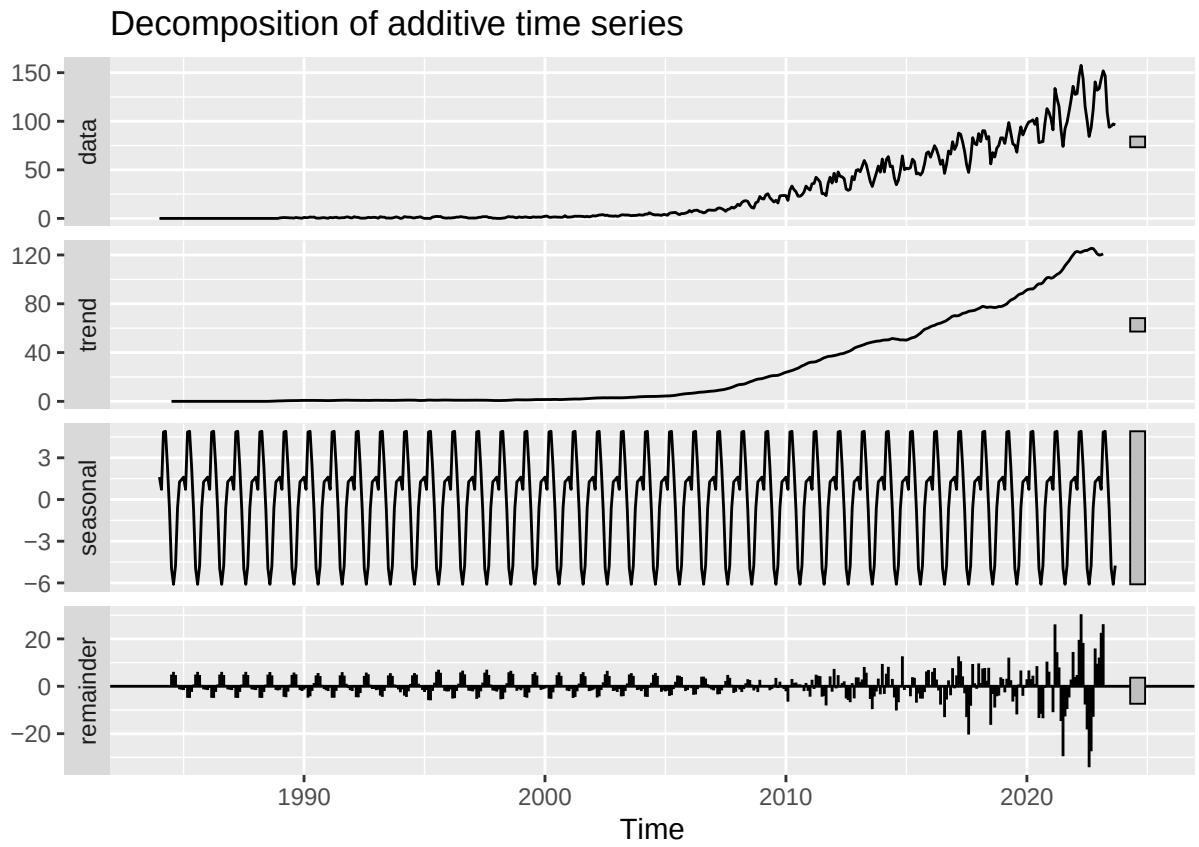
Q4

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
#making solar time series
solar_ts <- ts(solar_and_wind$`Solar Energy Consumption`,
               start = c(1984,1),
               frequency = 12)

#making wind time series
wind_ts <- ts(solar_and_wind$`Wind Energy Consumption`,
              start=c(1984,1),
              frequency = 12)

#decomposition
solar_decomposed <- decompose(solar_ts, type = "additive")
wind_decomposed <- decompose(wind_ts, type = "additive")

#plotting decomposed solar
autoplot(solar_decomposed)
```




```r
#plotting decomposed wind
autoplot(wind_decomposed)
```

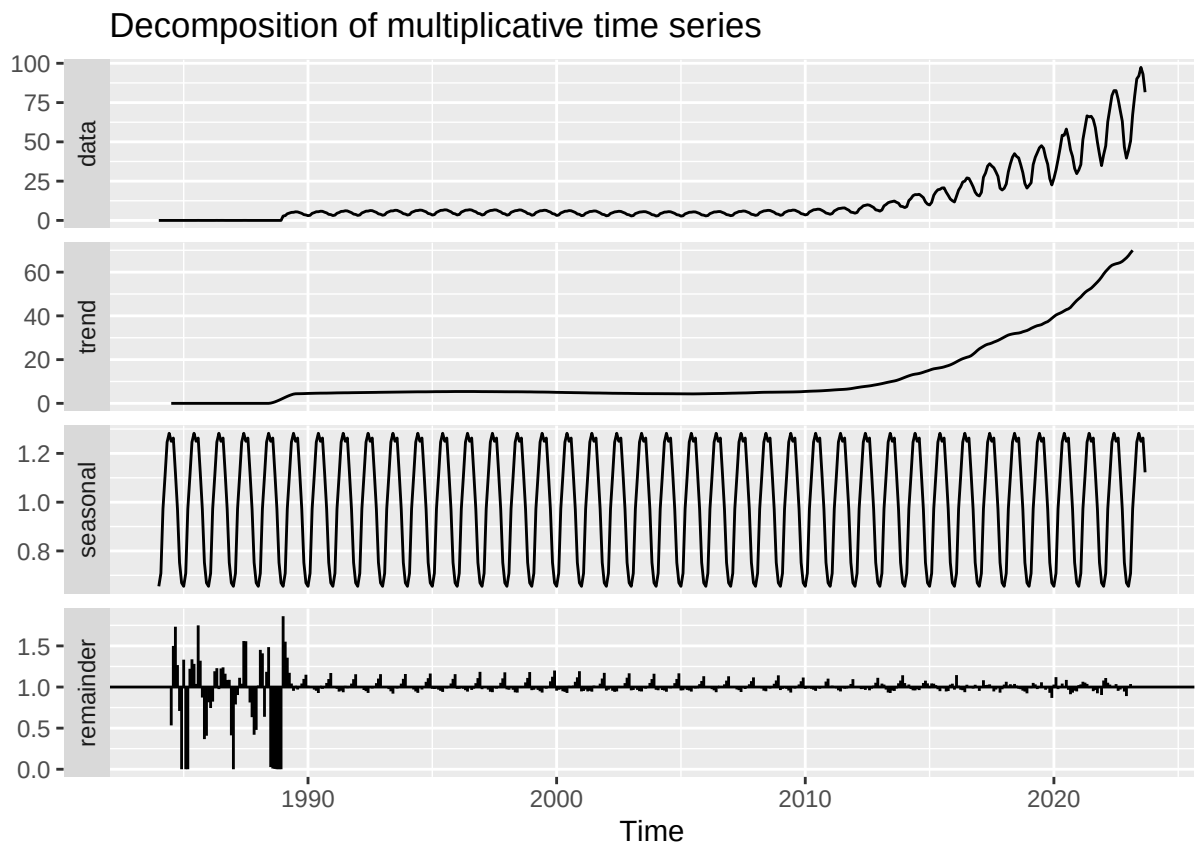


> The trend component for both the wind and the solar data seems to be exponential in nature, increasing dramatically around 2005-2010. For random component doesn't look random for either dataset, it's periodic for both and looks like it still contains a seasonal component.

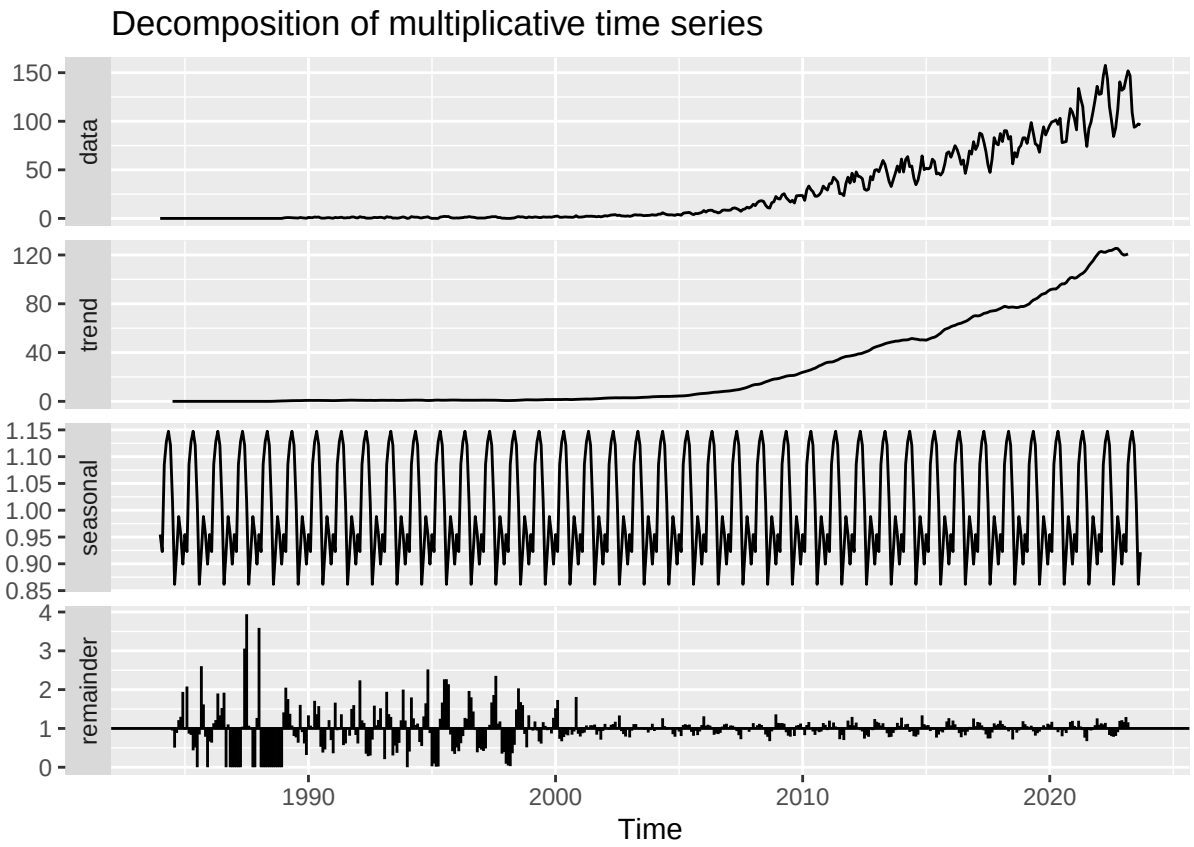
Q5

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?

```
#decomposition
solar_decomposed2 <- decompose(solar_ts, type = "multiplicative")
wind_decomposed2 <- decompose(wind_ts, type = "multiplicative")

#plotting decomposed
autoplot(solar_decomposed2)
```




```
autoplot(wind_decomposed2)
```



> The random component is more variable now (especially comparing before 1990 to after 1990 for solar and before 2000 to after 2000 for wind). There is still some repetitive pattern seen in the random component though.

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.

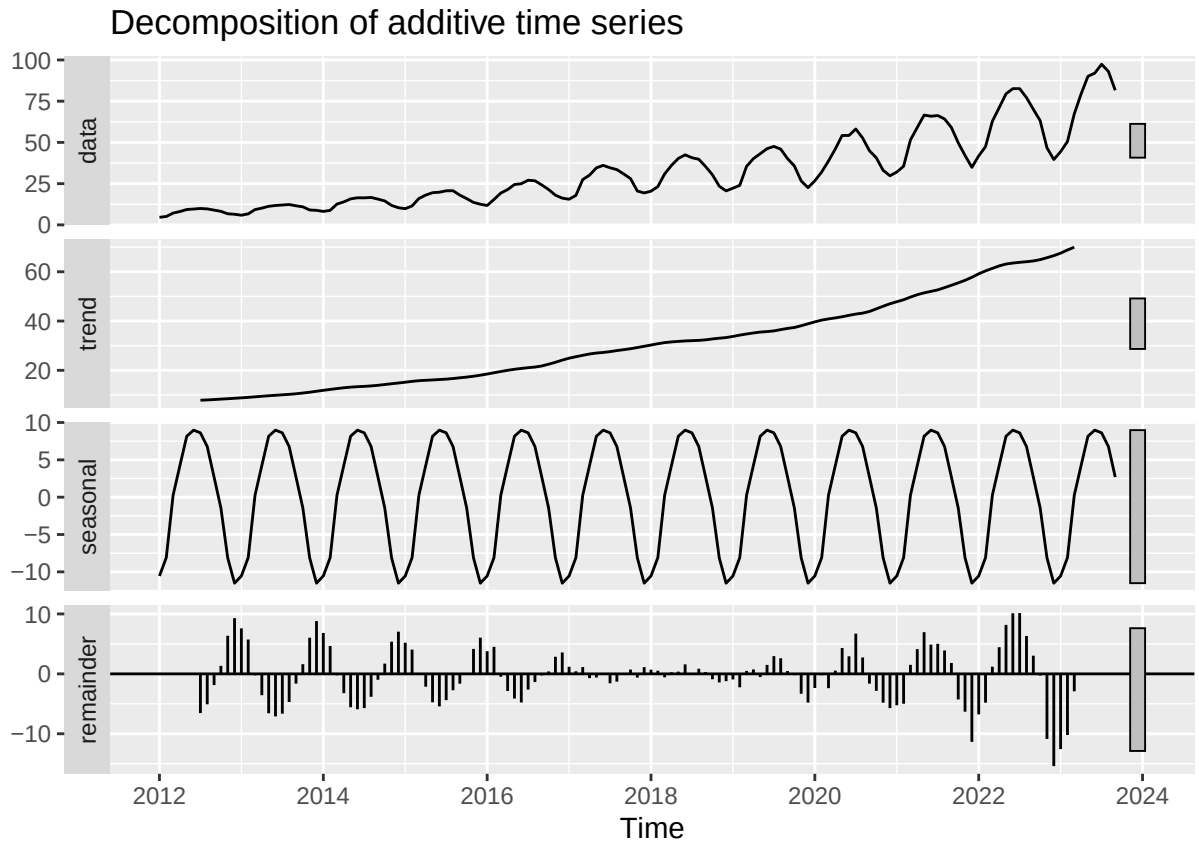
> Answer: No, I don't think we need all of the historical data. There is a clear period of stagnancy at the beginning of this dataset for both wind and solar consumption. I don't think that stagnant period represents how solar and wind are being used at this point in time, so it isn't helpful for forecasting the next 6 months of solar/wind consumption.

Q7

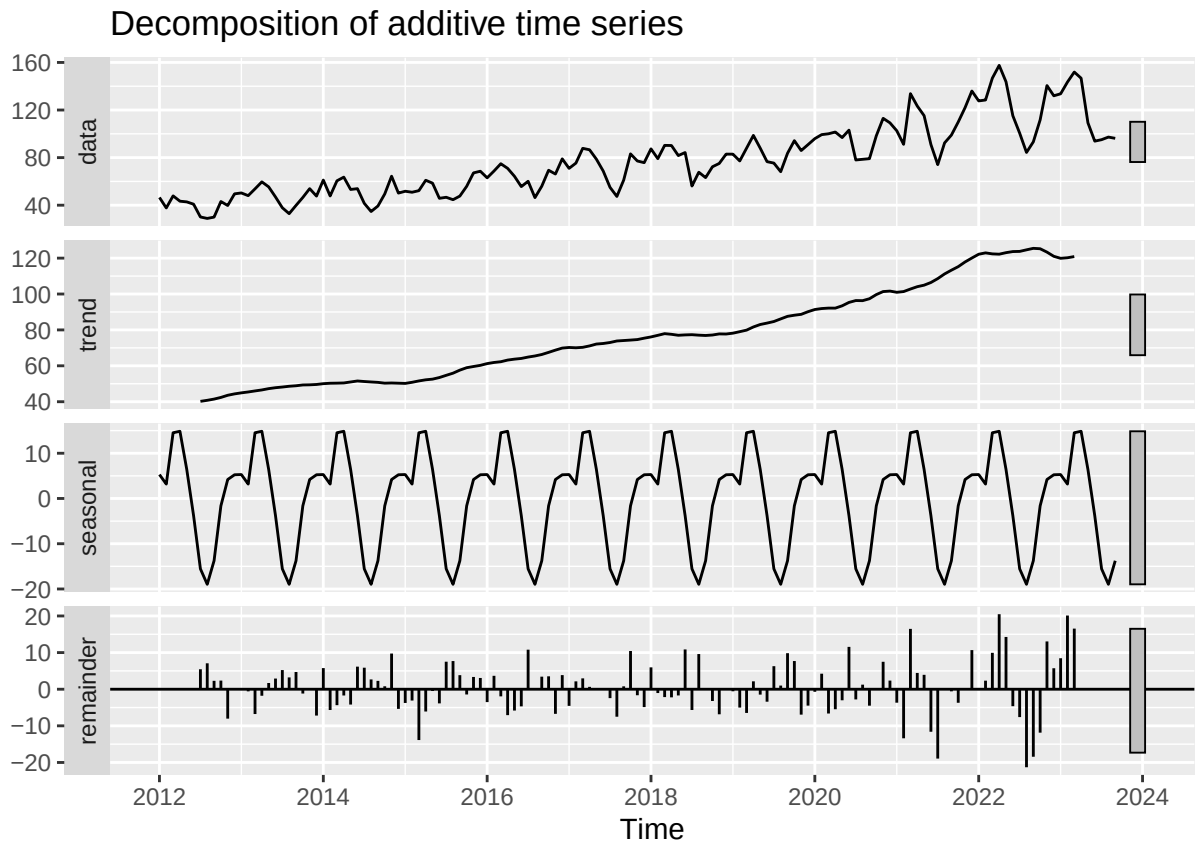
Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the decompose function `type=additive` to this new time series. Comment the results. Does the random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

```
#filtering for more recent years after 2011
recent_energy <- filter(solar_and_wind,
```

```
                        year(Month)>= 2012)

#making new ts objects
recent_wind_ts<- ts(recent_energy$`Wind Energy Consumption`,
               start=c(2012,1),
               frequency = 12)

recent_solar_ts<- ts(recent_energy$`Solar Energy Consumption`,
               start=c(2012,1),
               frequency = 12)

#decomposition
recent_solar_decomposed <- decompose(recent_solar_ts, type = "additive")
recent_wind_decomposed <- decompose(recent_wind_ts, type = "additive")

#plotting decomposed
autoplot(recent_solar_decomposed)
```




```
autoplot(recent_wind_decomposed)
```



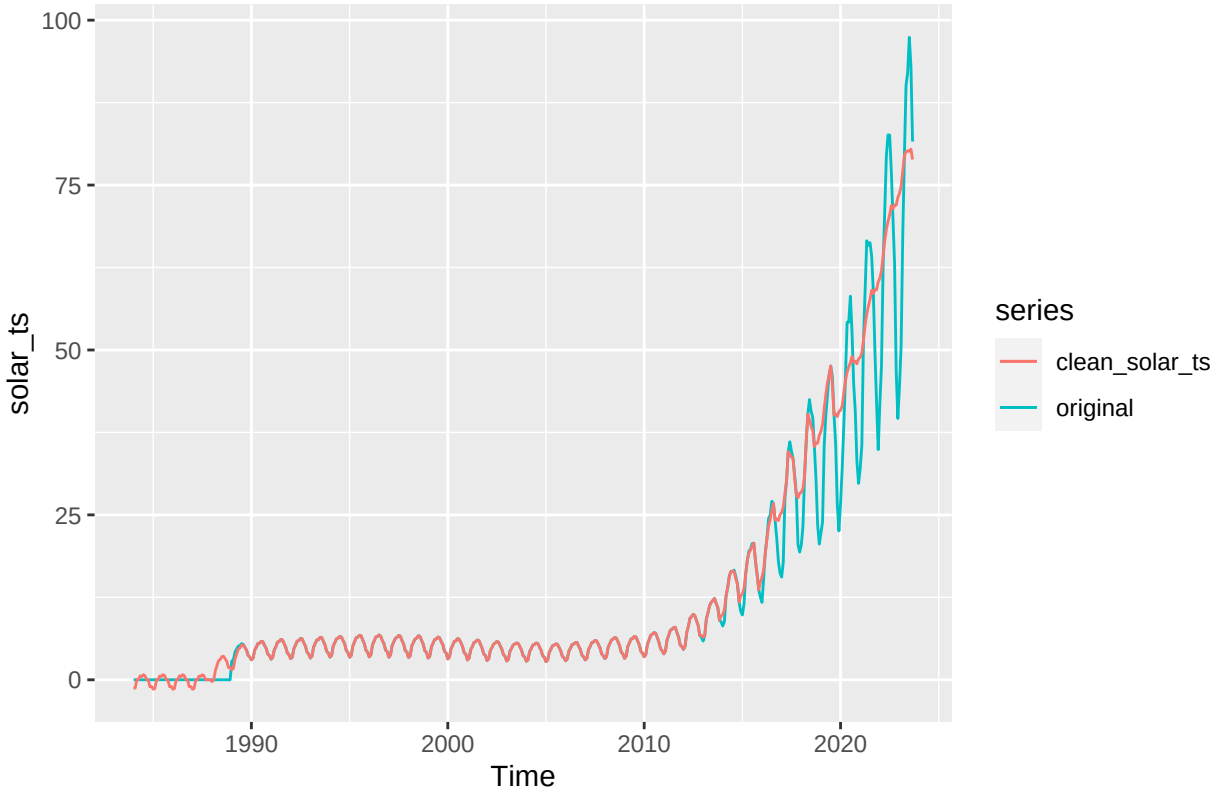
Answer: The random components still don't look random. They seem to follow a periodic trend that looks seasonal. This could be because we are looking at an average when subtracting seasonality. This means that any changes in seasonality over time aren't being well captured (for example if shifts in seasons are changing earlier or later).

Identify and Remove outliers

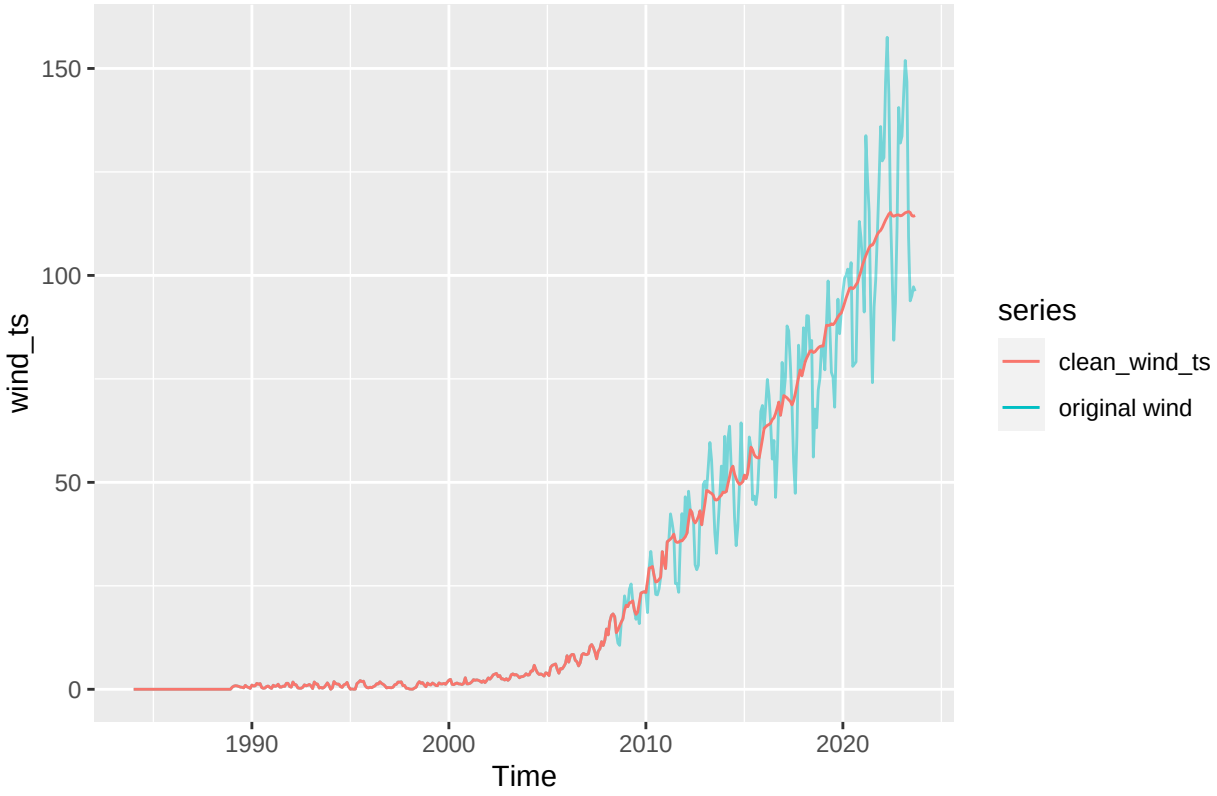
Q8

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.

```
#[use the entire dataset here!!]
#applying tsclean()
clean_wind_ts <- tsclean(wind_ts)
clean_solar_ts<- tsclean(solar_ts)

#checking the difference between the two series
autoplot(solar_ts, series="original")+
  autolayer(clean_solar_ts)
```




```
autoplot(wind_ts, series="original wind",alpha=0.5)+
  autolayer(clean_wind_ts)
```

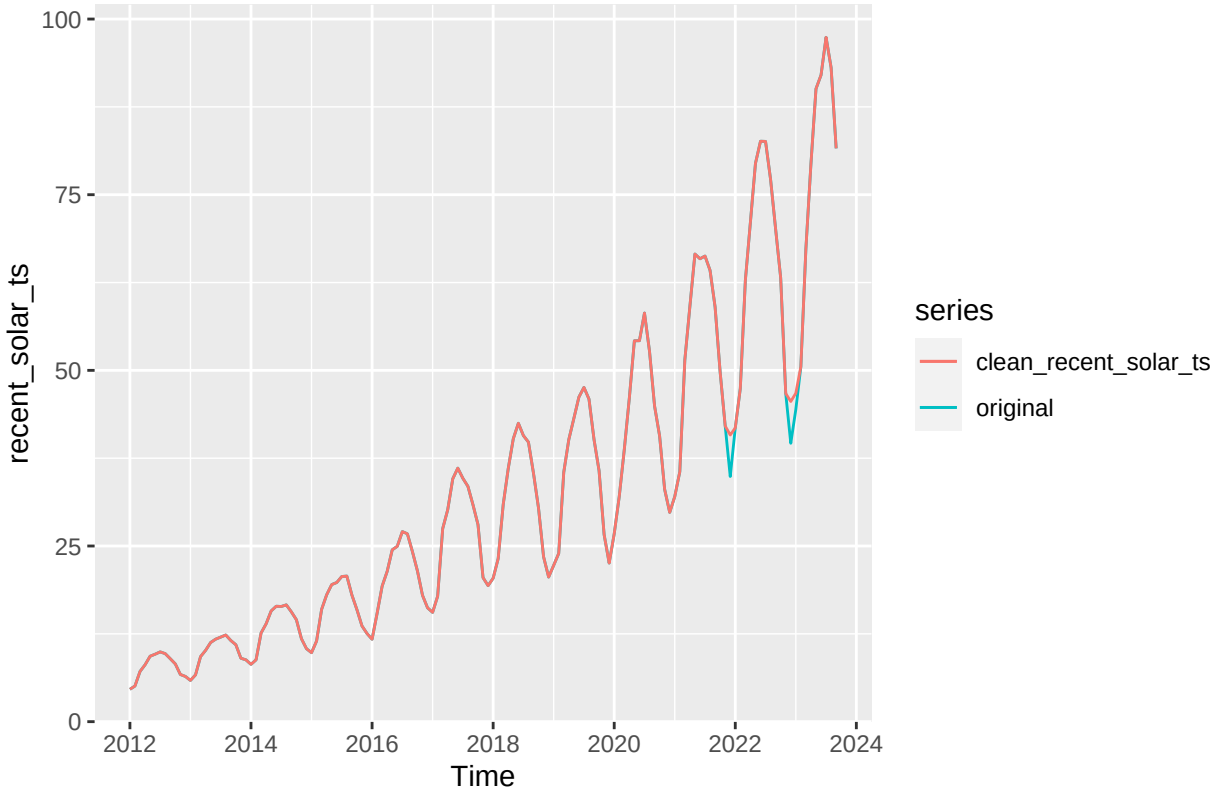


The function removed a lot of data from the series and identified most the variability we see towards the end of the dataset as outliers. It looks like it removed more than just outliers after around 2010. Indicating we might not want to use the earlier years of this dataset becuase it impacts the mean.

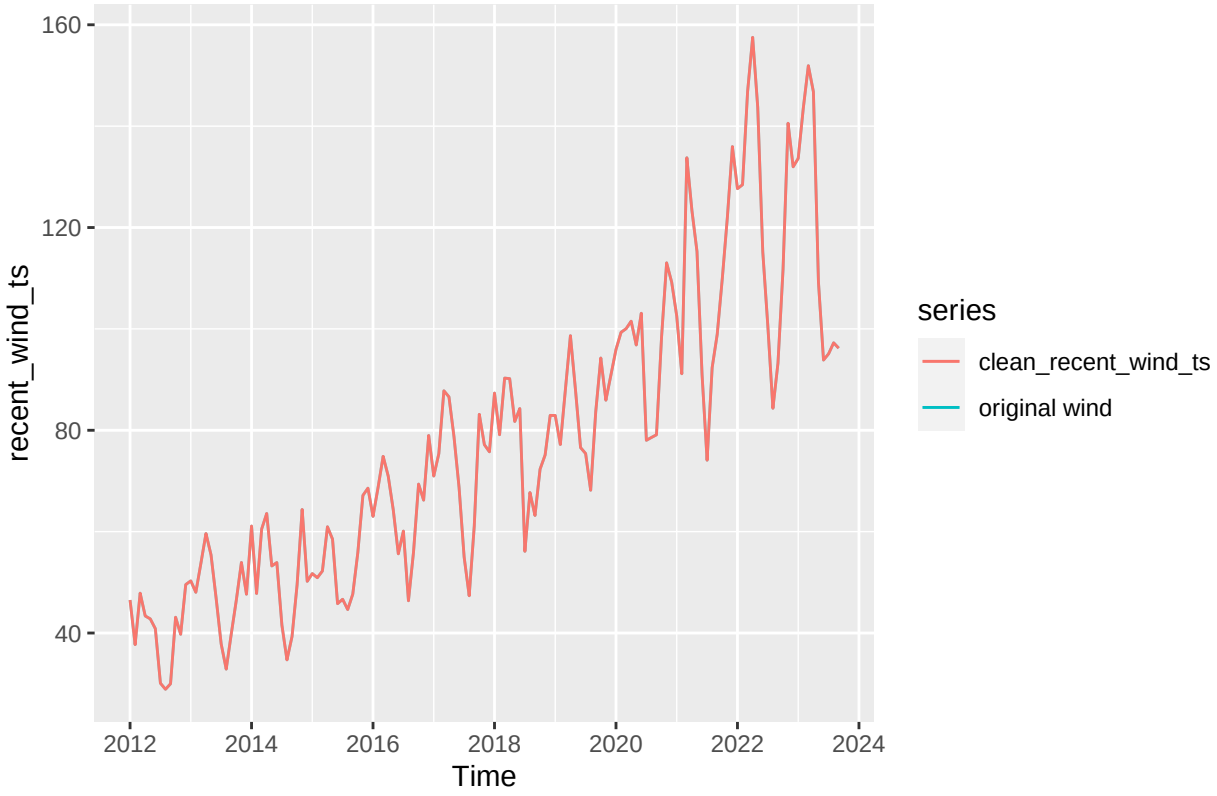
Q9

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?

```
#use just the recent data here!!
#applying tsclean()
clean_recent_wind_ts <- tsclean(recent_wind_ts)
clean_recent_solar_ts<- tsclean(recent_solar_ts)

#checking the differnce between the two series
autoplot(recent_solar_ts, series="original")+
  autolayer(clean_recent_solar_ts)
```




```
autoplot(recent_wind_ts, series="original wind",alpha=0.5)+
  autolayer(clean_recent_wind_ts)
```



Answer: The function did remove a few outliers from the solar series, it did not appear to remove any outliers from the wind series when just looking at more recent years.