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

Assignment 5 - Due date 02/13/24

Cara Kuuskvere

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   1.1.3     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 error
```

```
library(readr)
```

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
library(readxl)
energy_data_raw <- read_excel("./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`
```

```
#Now let's extract the column names from row 11 only
read_col_names <- read_excel("./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.x
```

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

```
colnames(energy_data_raw) <- read_col_names
head(energy_data_raw)
```

```
## # 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_raw)
nvar=ncol(energy_data_raw)
```

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!

```
#creating date object
library(lubridate)
energy_data_raw$Month <- ymd(energy_data_raw$Month)

energy_data <- as.data.frame(c(energy_data_raw[,1],energy_data_raw[,8],energy_data_raw[,9]))

energy_data$Solar.Energy.Consumption <- as.numeric(energy_data$Solar.Energy.Consumption)
```

```
## Warning: NAs introduced by coercion
```

```
energy_data$Wind.Energy.Consumption <- as.numeric(energy_data$Wind.Energy.Consumption)
```

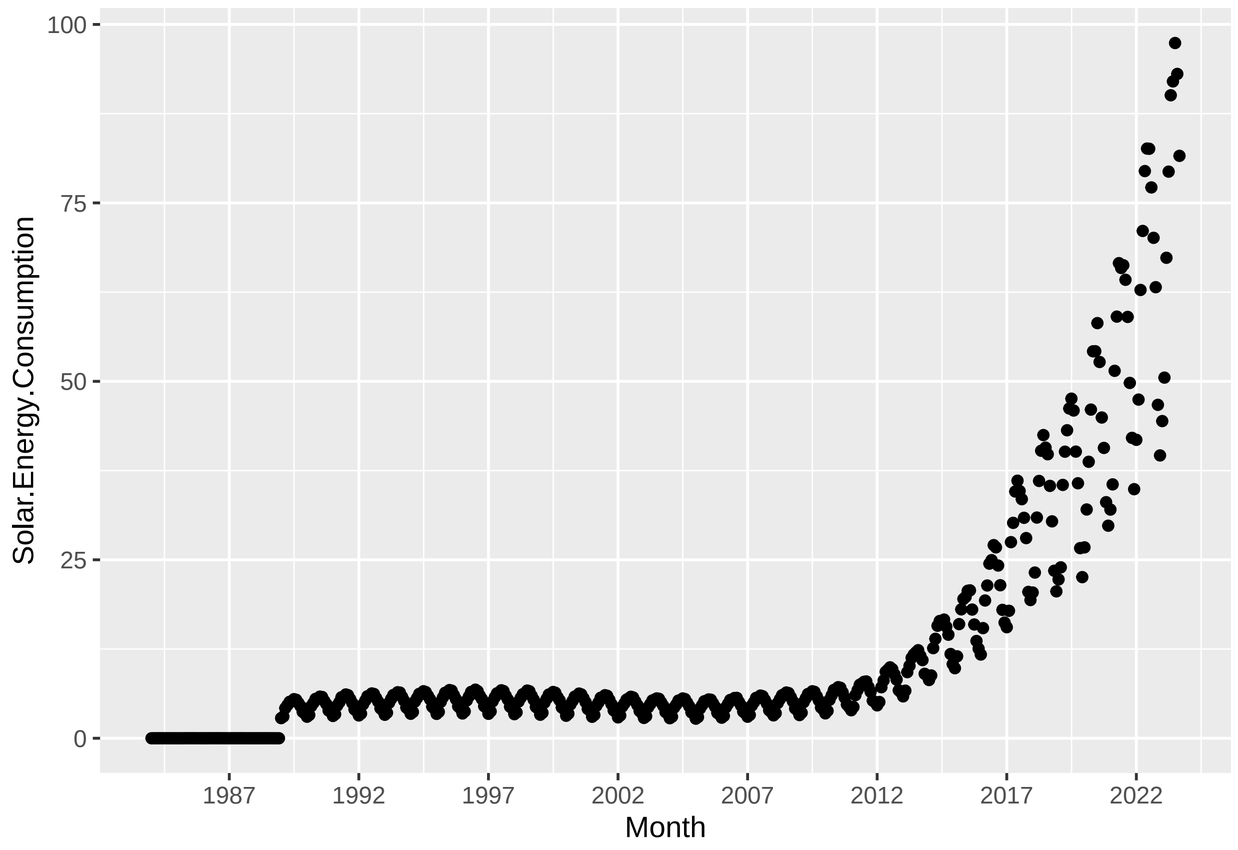
```
## Warning: NAs introduced by coercion
```

```
energy_data <- drop_na(energy_data)

solar_ts <- ts(energy_data$Solar.Energy.Consumption,frequency = 12,start=1984)
wind_ts <- ts(energy_data$Wind.Energy.Consumption,frequency = 12, start=1984)
```

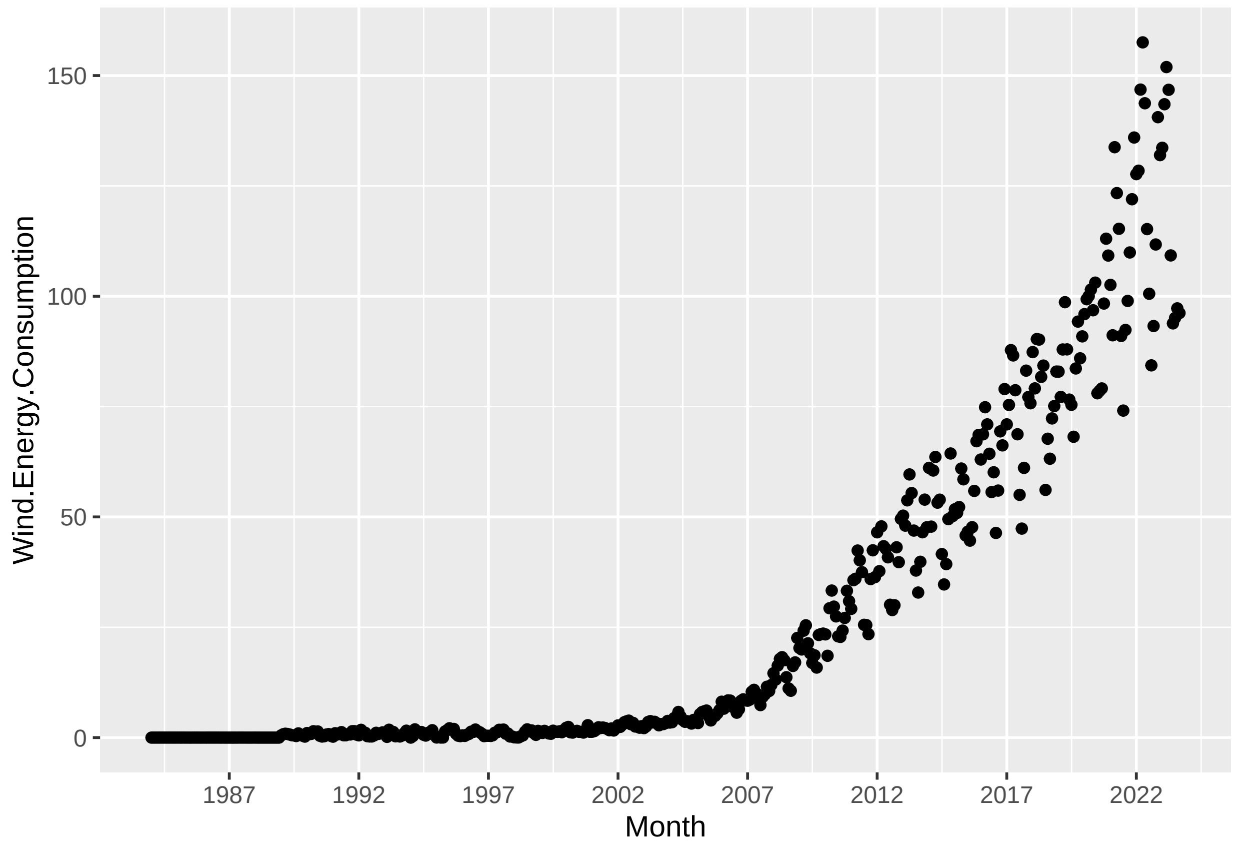
Q2

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(energy_data)+
  geom_point(aes(x=Month,y=Solar.Energy.Consumption))+
  labs(ylab="Solar Energy Consumption", xlab="Month")+
  scale_x_date(date_breaks = "5 years", date_labels= "%Y")
```



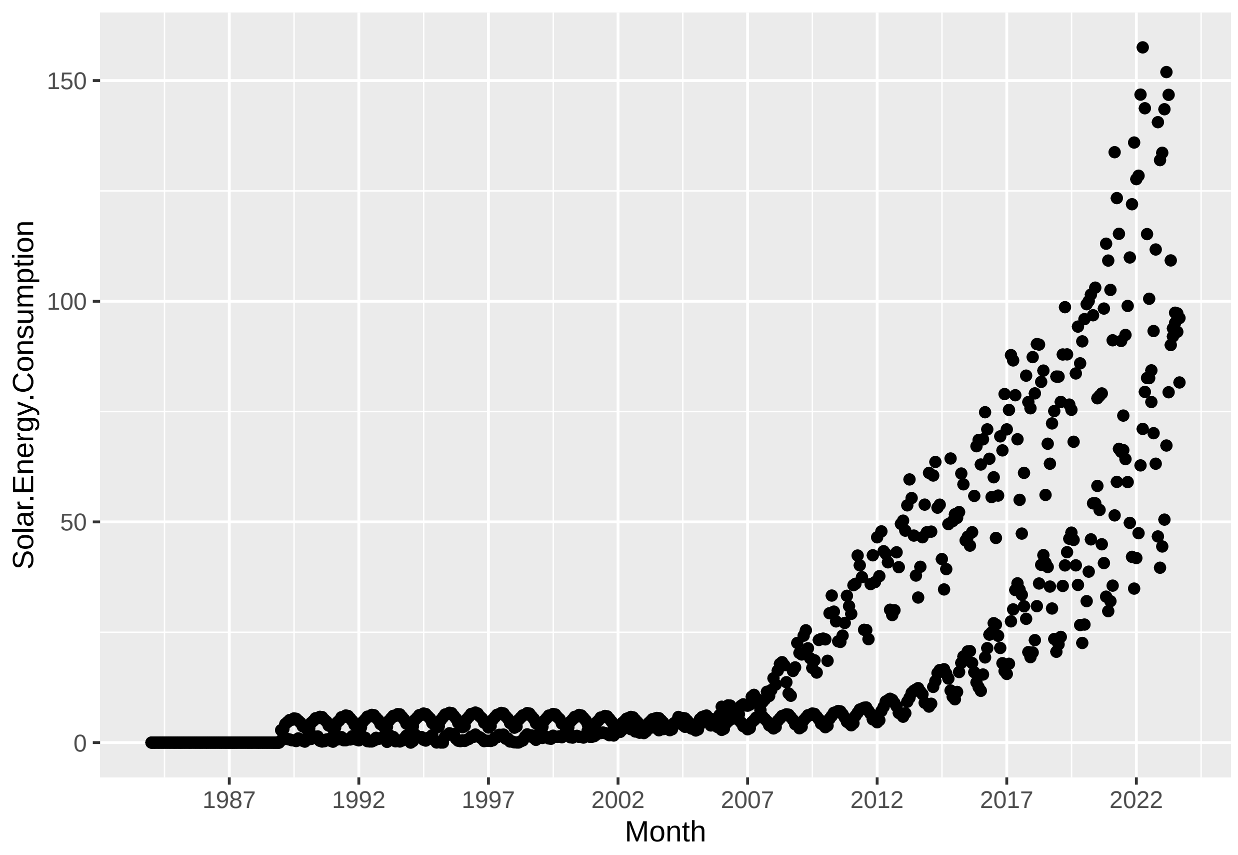
```
ggplot(energy_data)+
  geom_point(aes(x=Month,y=Wind.Energy.Consumption))+
  labs(ylab="Wind Energy Consumption", xlab="Month")+
  scale_x_date(date_breaks = "5 years", date_labels= "%Y")
```



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(energy_data)+
  geom_point(aes(x=Month,y=Solar.Energy.Consumption))+
  labs(ylab="Energy Consumption", xlab="Month")+
  geom_point(aes(x=Month,y=Wind.Energy.Consumption))+
  scale_x_date(date_breaks = "5 years", date_labels= "%Y")+
  scale_color_manual(values=cols)
```



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.

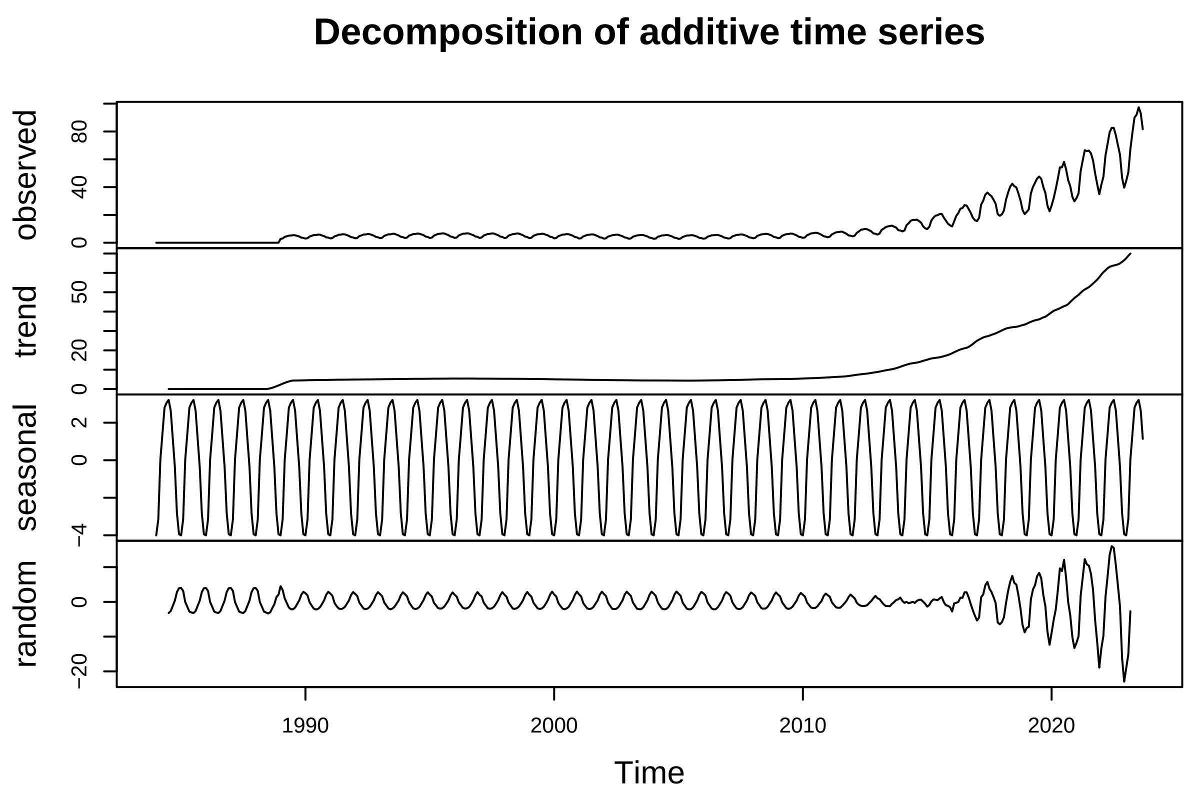
Additional info on `decompose()`.

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.

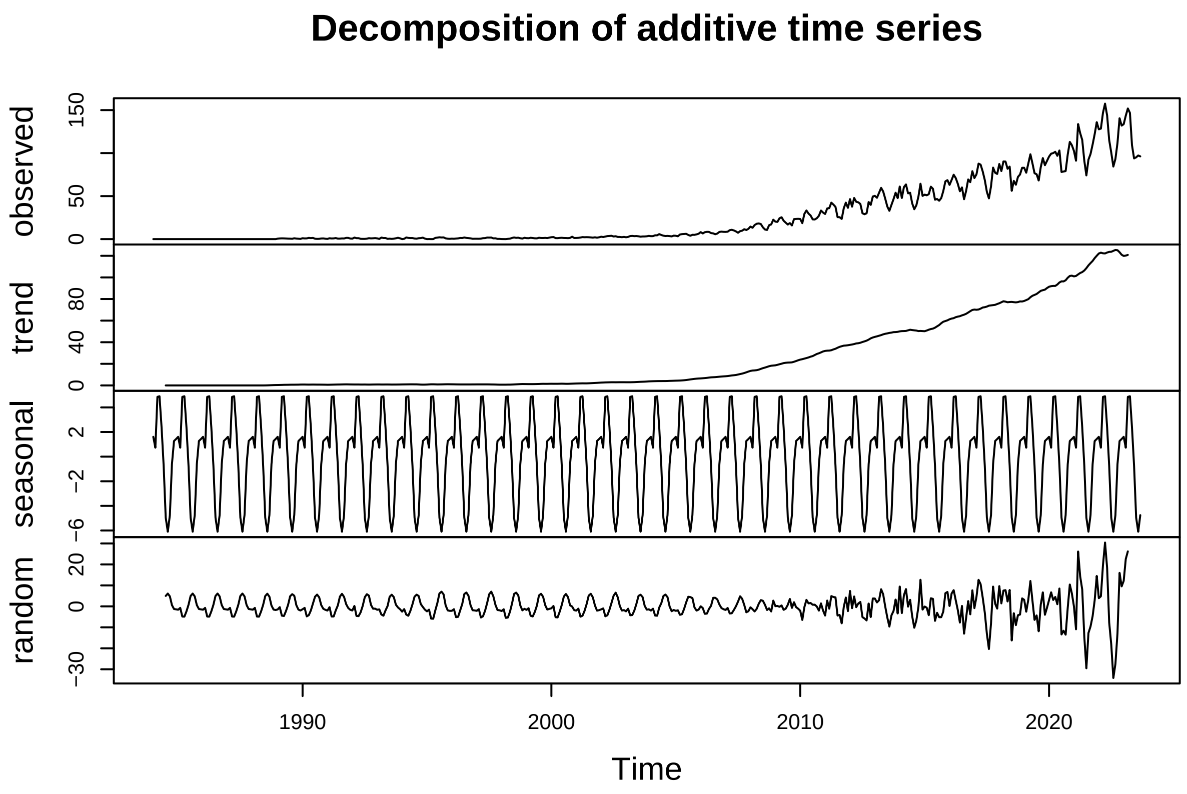
Q4

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
decomposed_solar_add <- decompose(solar_ts,type="additive")
plot(decomposed_solar_add)
```



```r
decomposed_wind_add <- decompose(wind_ts,type="additive")
plot(decomposed_wind_add)
```



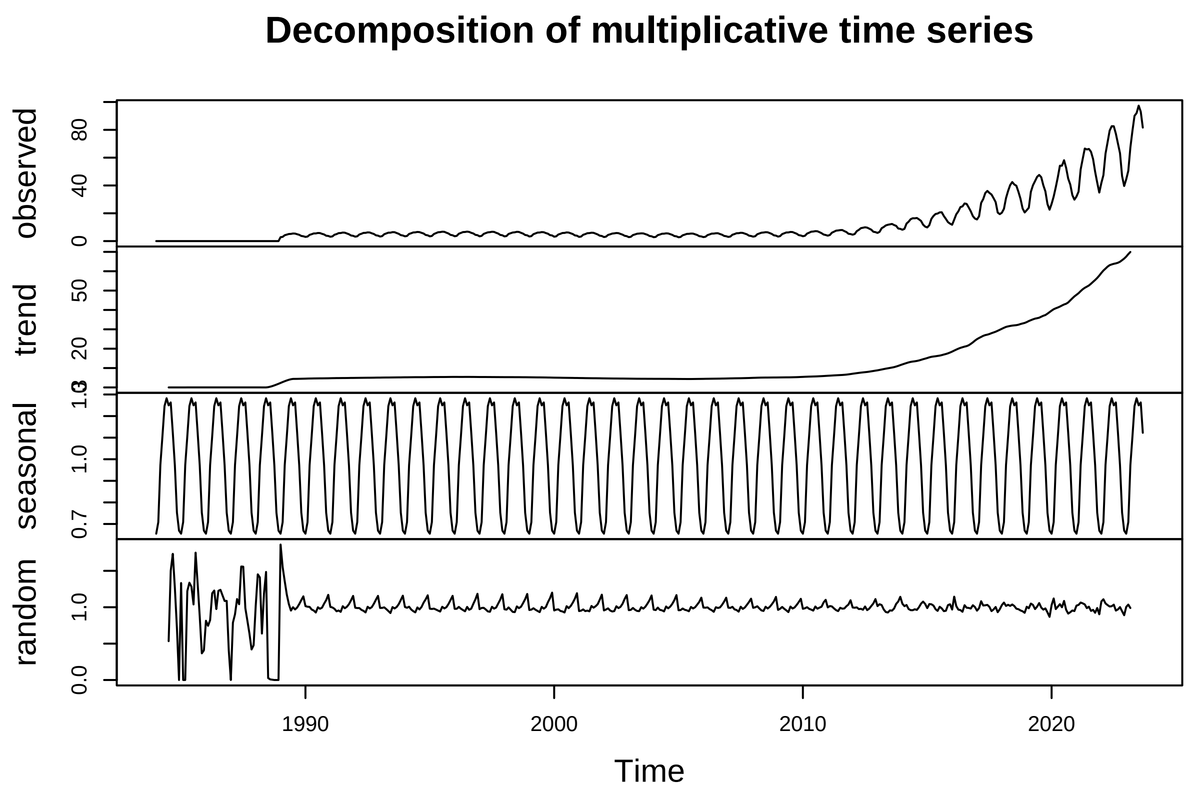
The solar trend component increases with time, whereas the seasonal component stays the same over time. The random component appears that it could still have a seasonal component because it has changes at regular intervals.

The wind trend component increases earlier in time and has a more linear slope beyond this point, whereas the solar trend component has a more exponential slope once it changes. The seasonal component has three different inflection points per seasonal period rather than two per seasonal period as in the solar series. Wind's random component does not look as seasonal as the solar random component as the interval is less regular, especially after period 25.

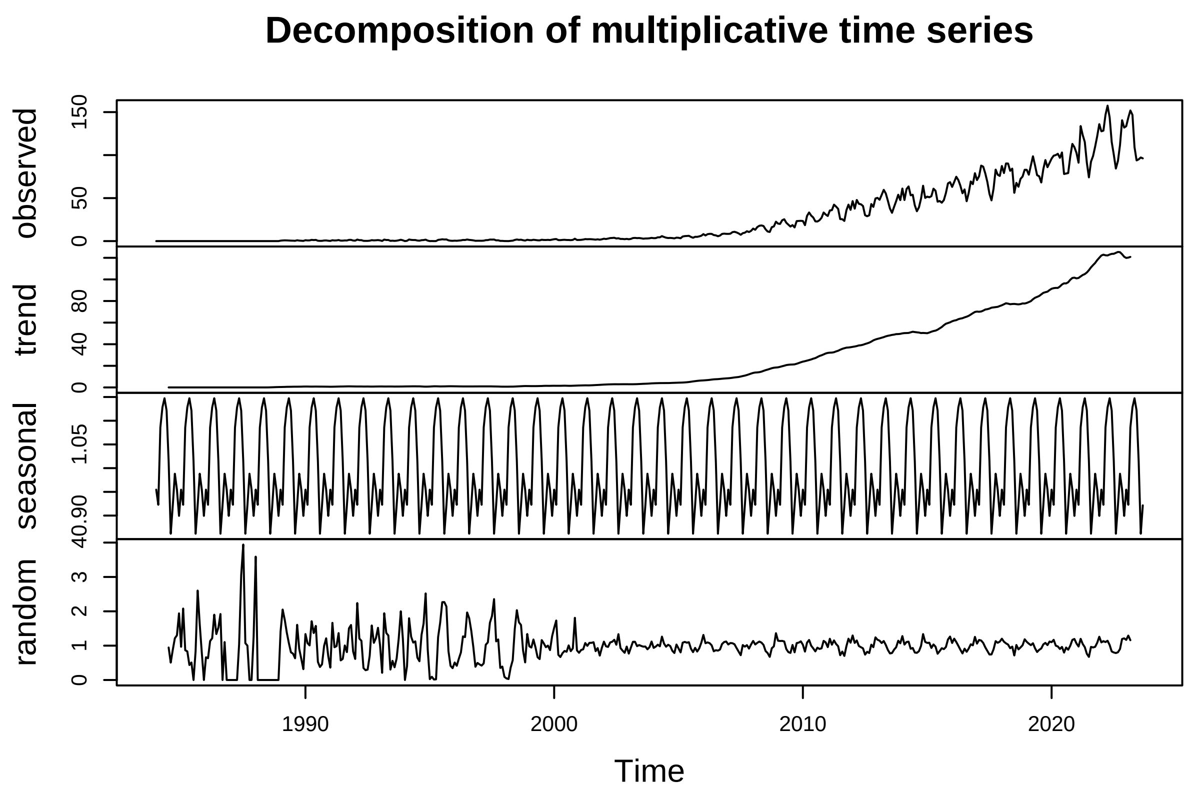
Q5

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
decomposed_solar_mult <- decompose(solar_ts,type="multiplicative")
plot(decomposed_solar_mult)
```



```r
decomposed_wind_mult <- decompose(wind_ts,type="multiplicative")
plot(decomposed_wind_mult)
```



In the solar decomposition, the random component dampens greatly after period ~8. It's randomness looks far less seasonal and it looks far smaller over time now.

Both the solar and wind decomposition have random components that now appear far more random, which should be more expected.

Q6

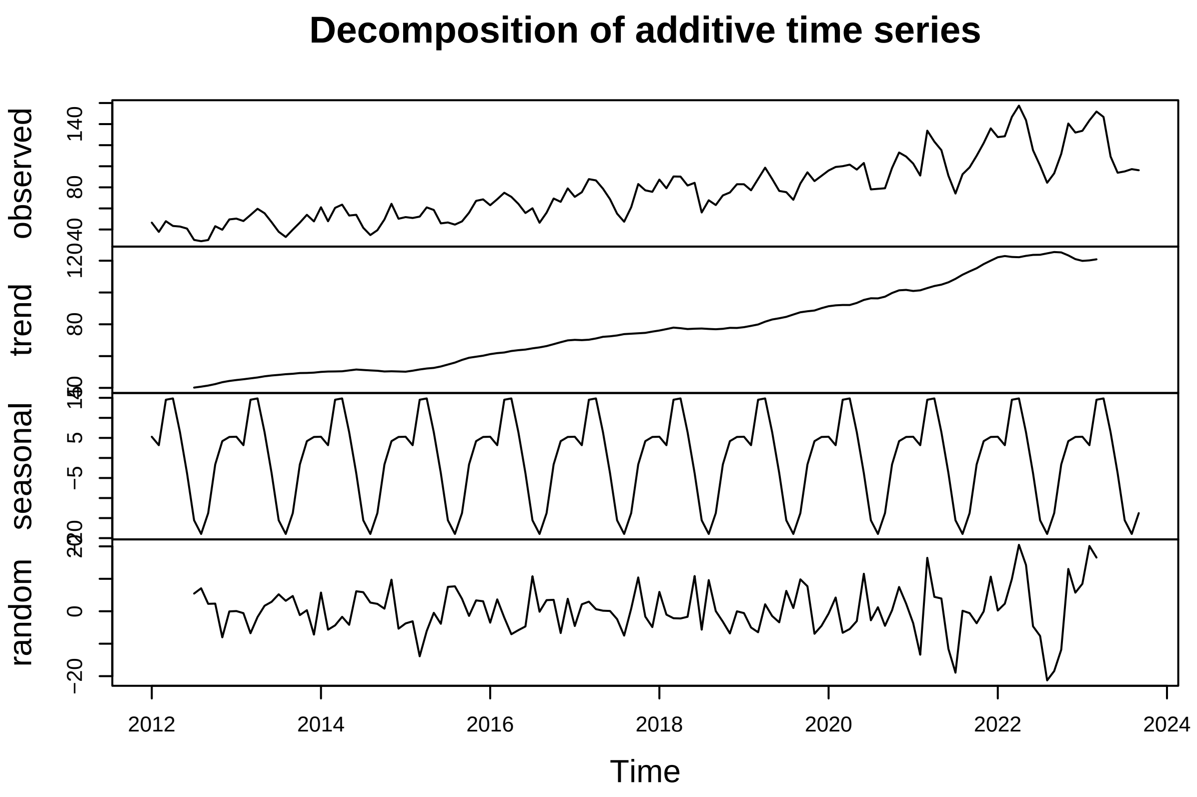
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: It appears that there is a level shift after the 90s and early 20s. Therefore, all the historical data may help interpoliation/ filling in information given seasonal patterns, however using closer historical data is best given the change in the level and the differences in the data.

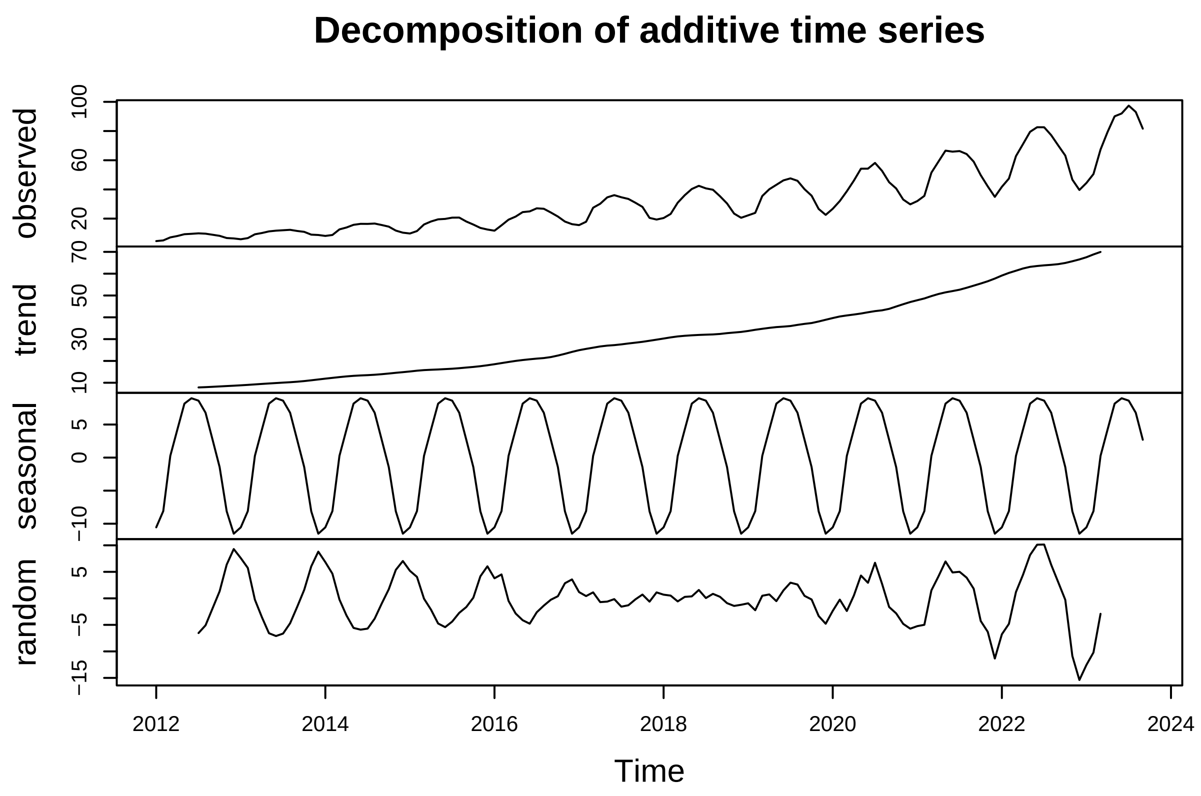
Q7

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.

```
library(stats)

energy_newer <- energy_data[337:477,]
solar_ts_newer <- ts(energy_newer$Solar.Energy.Consumption, frequency = 12, start=2012)
wind_ts_newer <- ts(energy_newer$Wind.Energy.Consumption,frequency = 12, start = 2012)

wind_newer_decomp_add <- decompose(wind_ts_newer, type= "additive")
plot(wind_newer_decomp_add)
```



```
solar_newer_decomp_add <- decompose(solar_ts_newer, type= "additive")
plot(solar_newer_decomp_add)
```



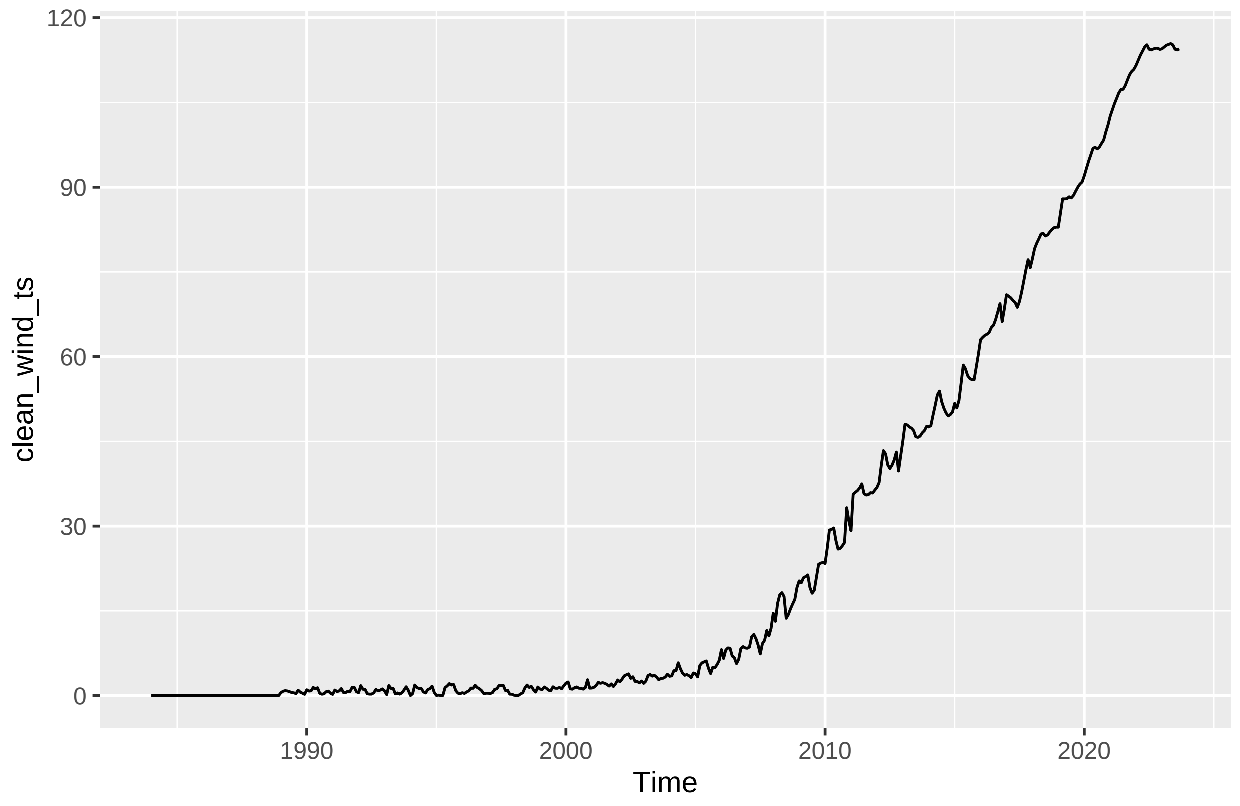
Answer: The wind time series now has a much more random looking random component, even and less dramatic seasonal component, and lacks the level shift in the trend component. The same goes for the solar decomposed series, with a linear trend increase and a more random and different looking random component.

Identify and Remove outliers

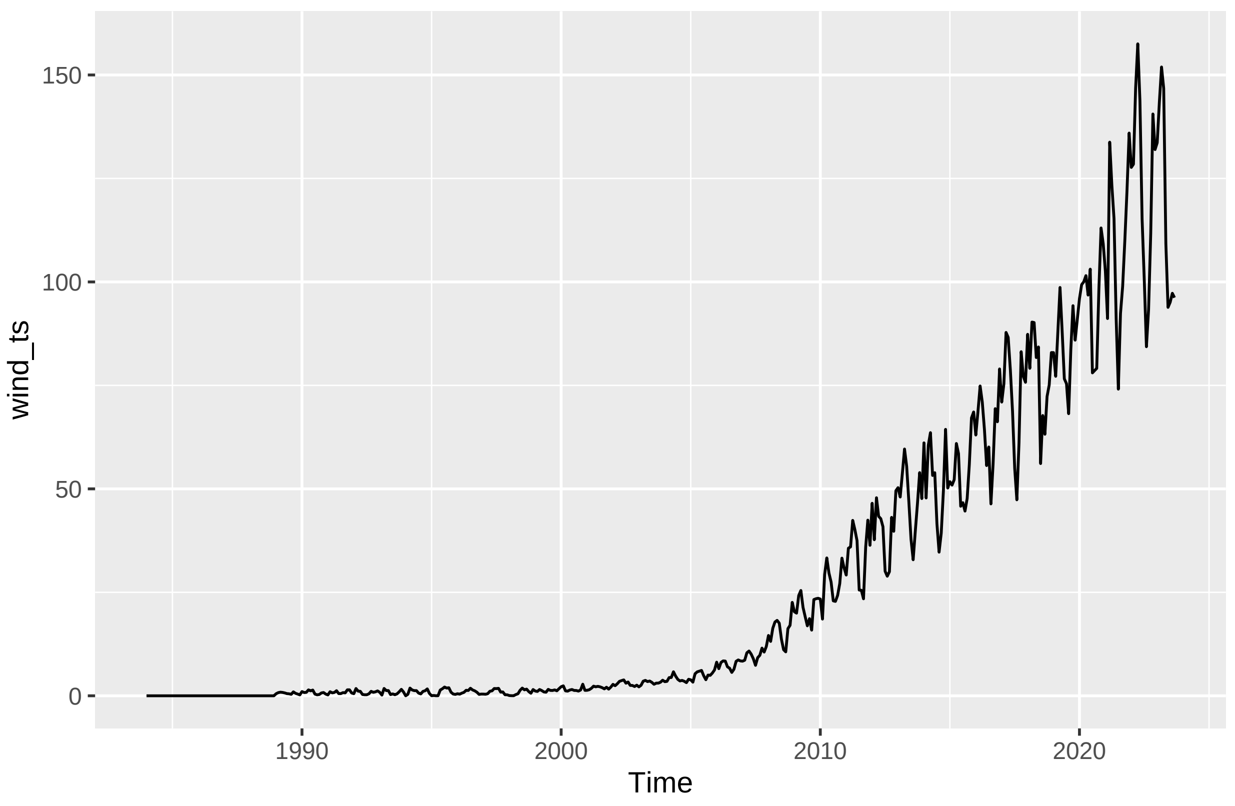
Q8

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.

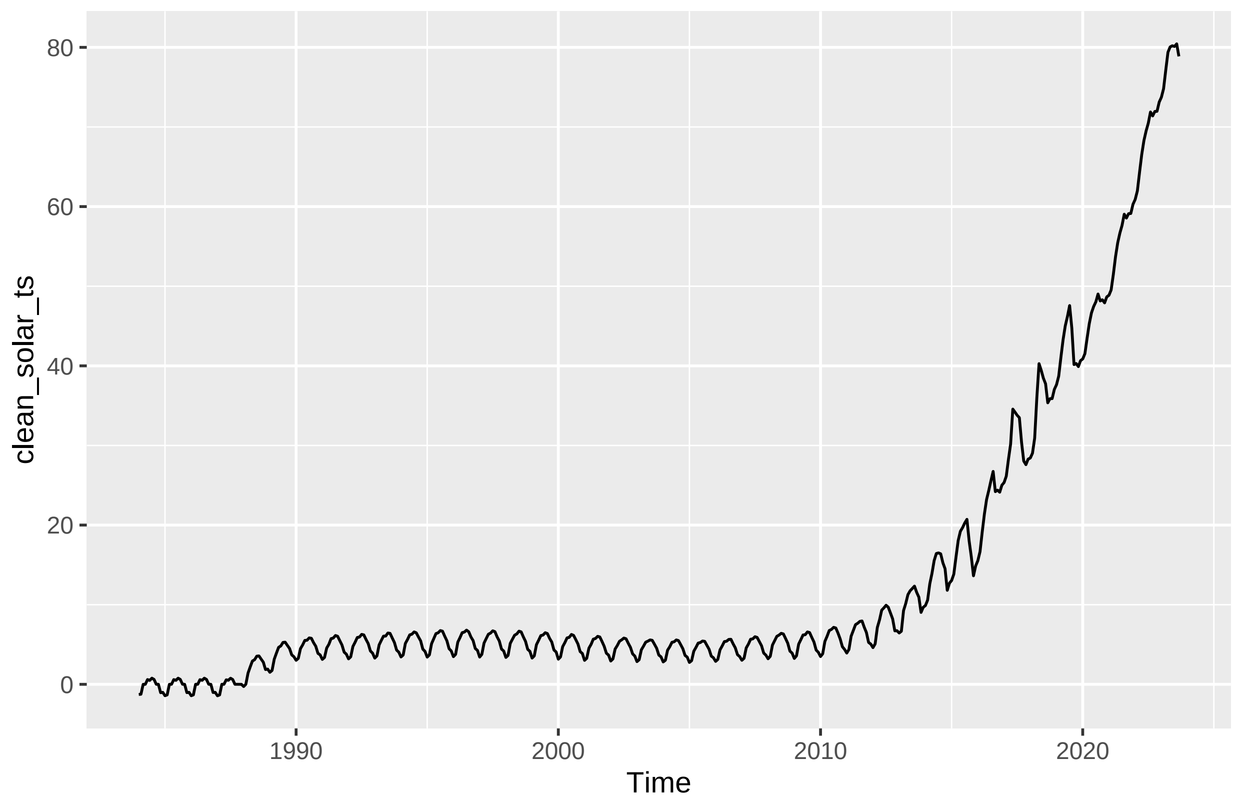
```r
# assuming you actually meant the non-newer plots...

clean_wind_ts <- tsclean(wind_ts)
autoplot(clean_wind_ts)
```



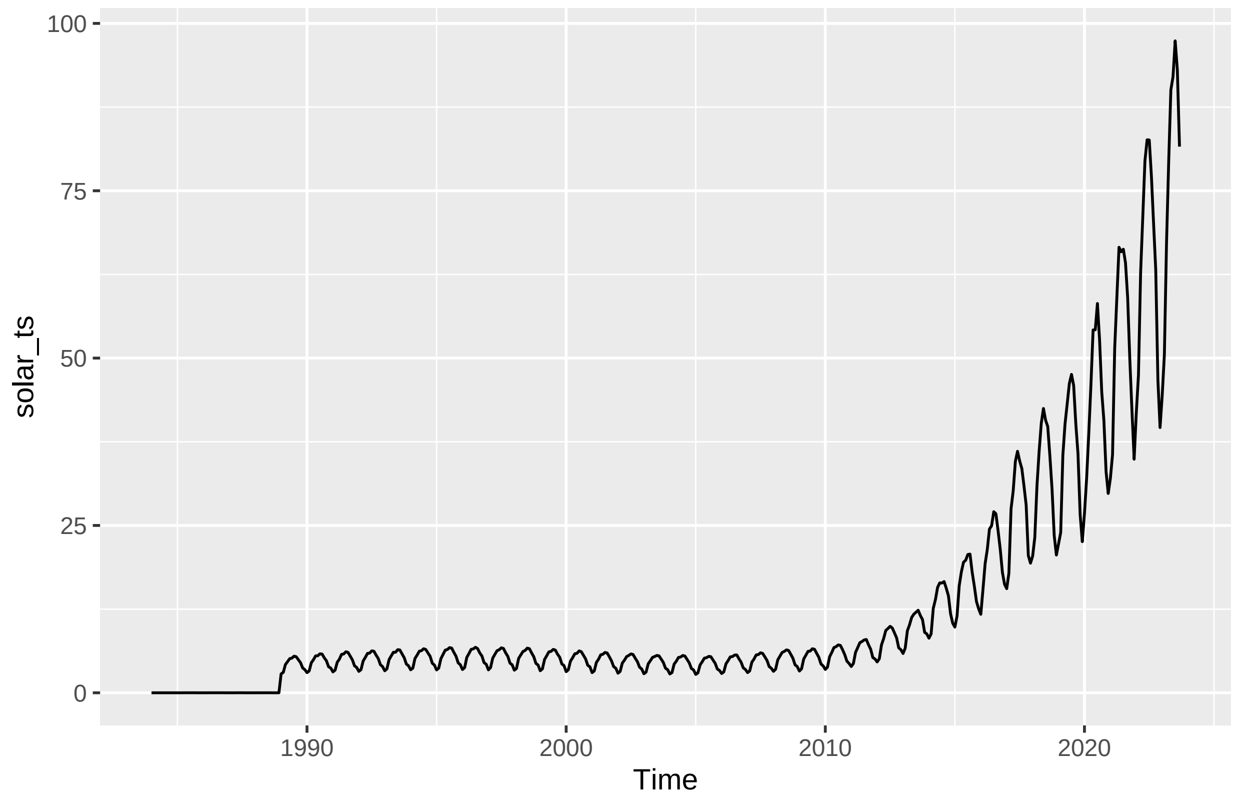
```r
autoplot(wind_ts)
```



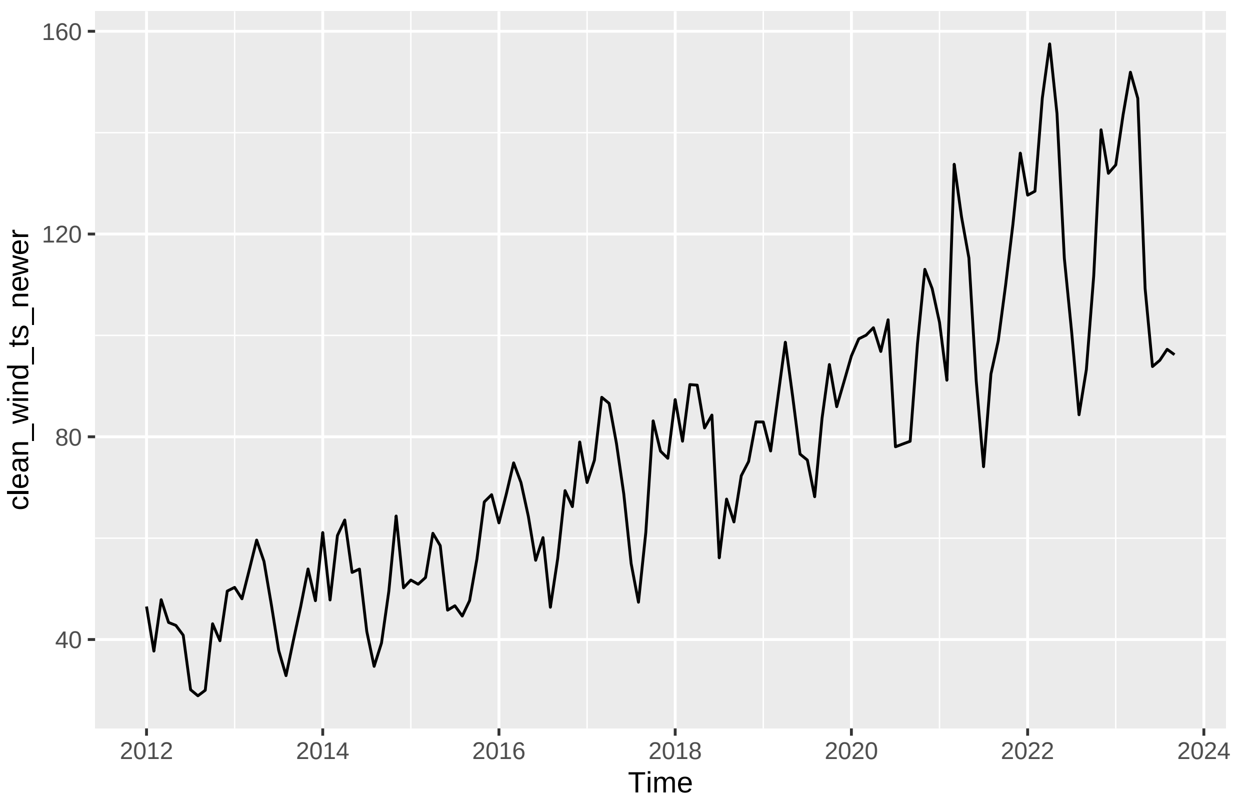
```
clean_solar_ts <- tsclean(solar_ts)
autoplot(clean_solar_ts)
```



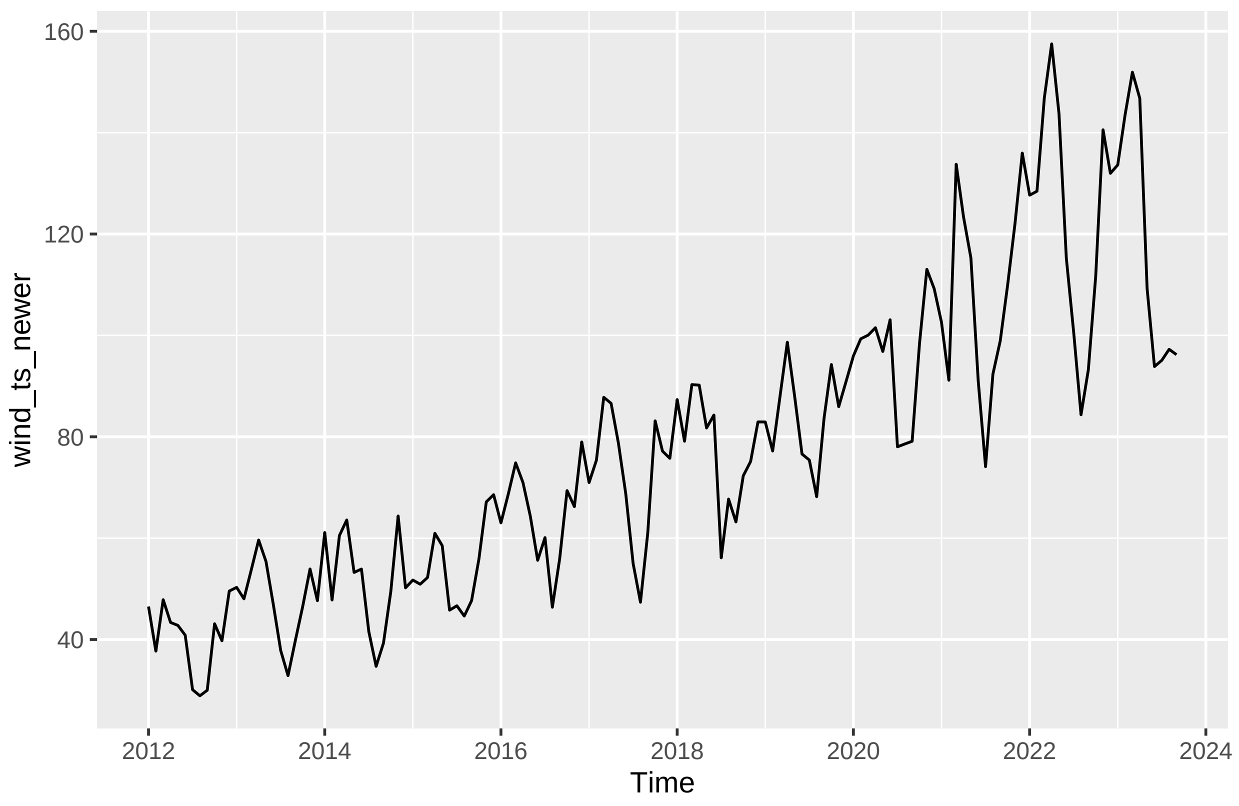
```
autoplot(solar_ts)
```



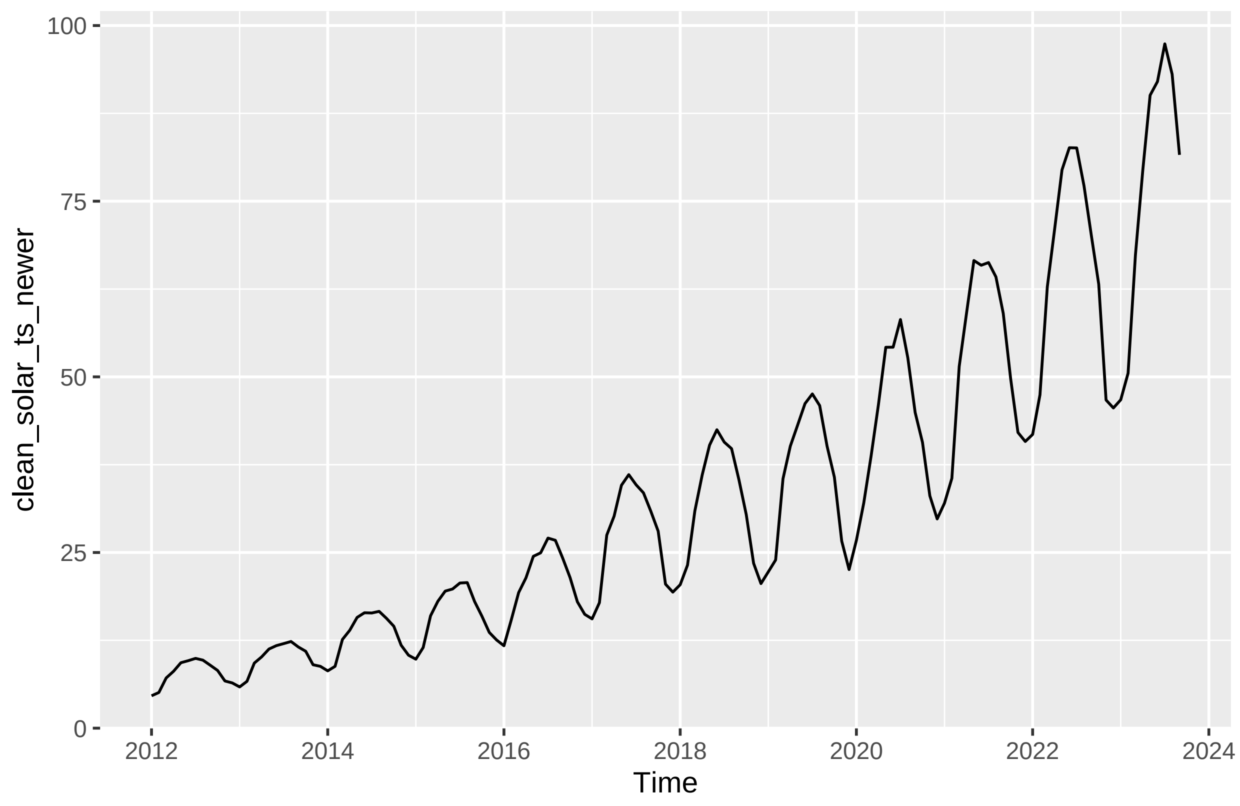
```r
clean_wind_ts_newer<- tsclean(wind_ts_newer)

autoplot(clean_wind_ts_newer)
```



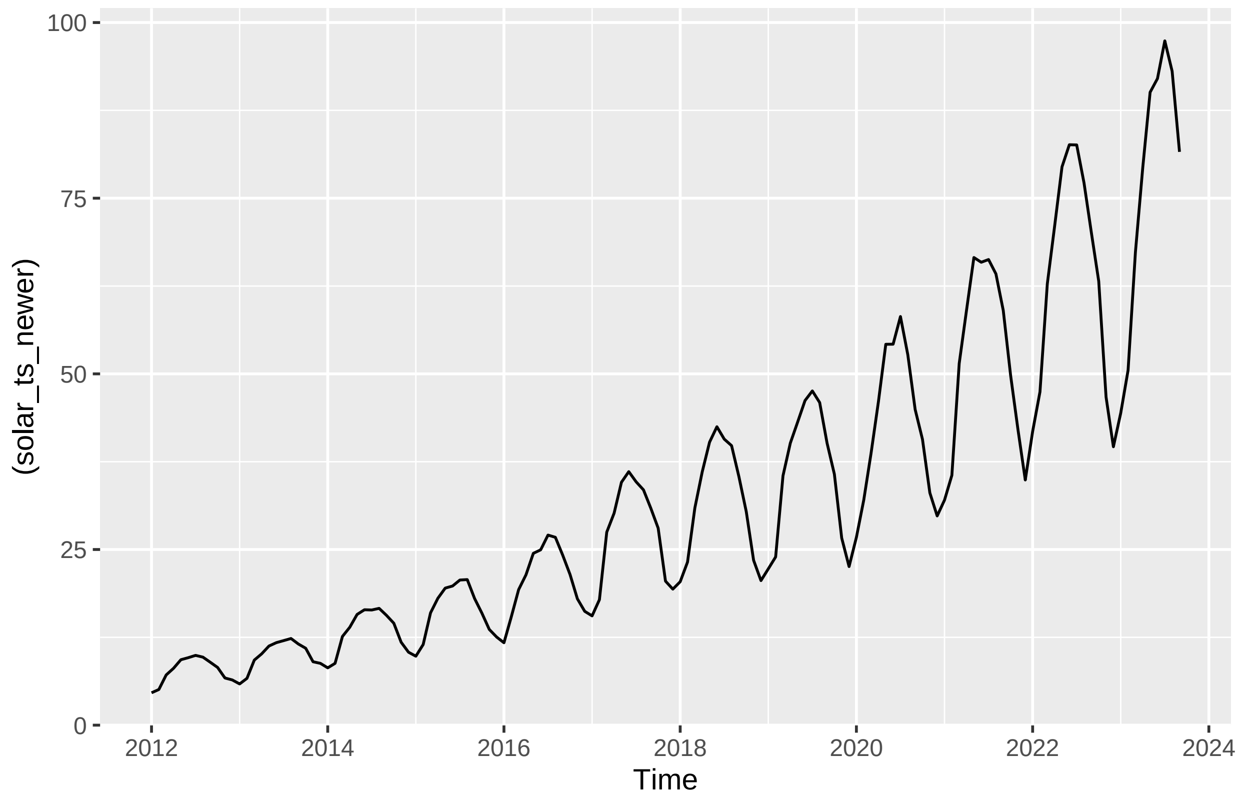
```r
autoplot(wind_ts_newer)
```



```r
clean_solar_ts_newer <- tsclean(solar_ts_newer)

autoplot(clean_solar_ts_newer)
```



```r
autoplot((solar_ts_newer))
```

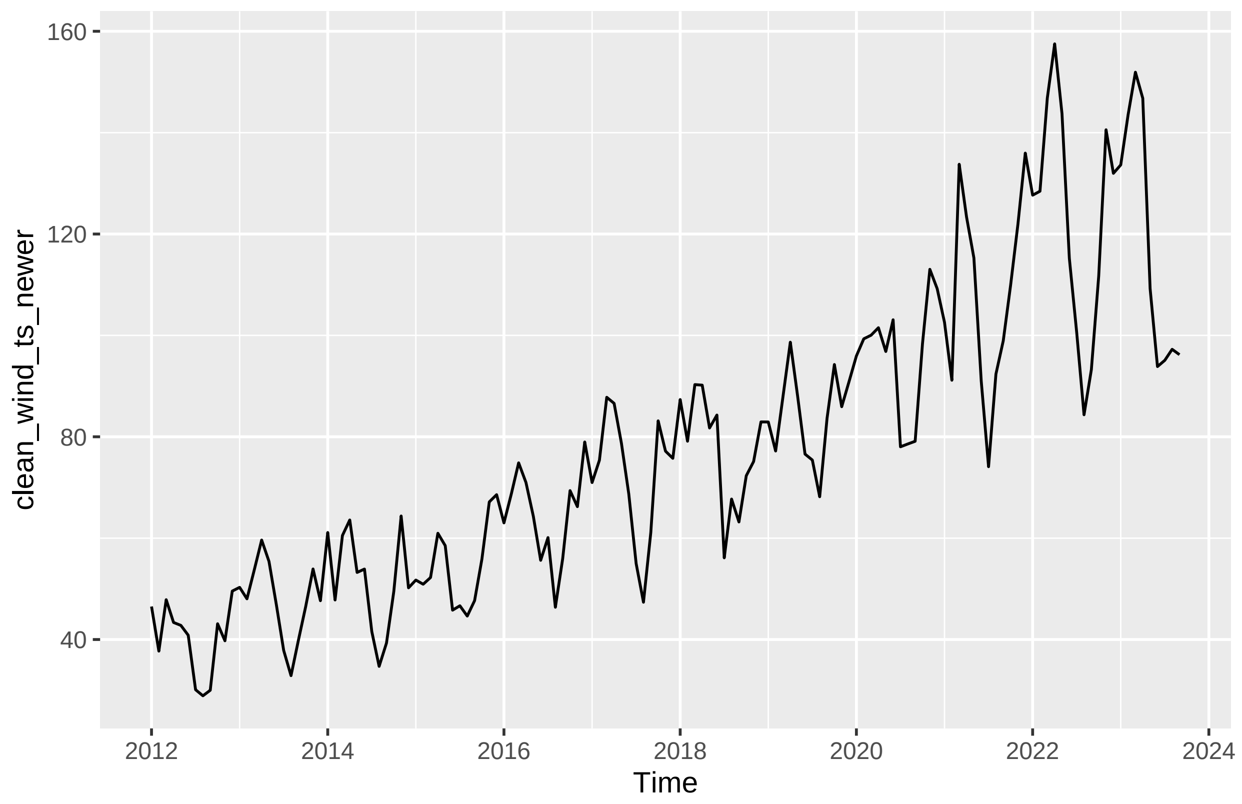


The clean function appears to have removed outliers from the solar and wind time series.

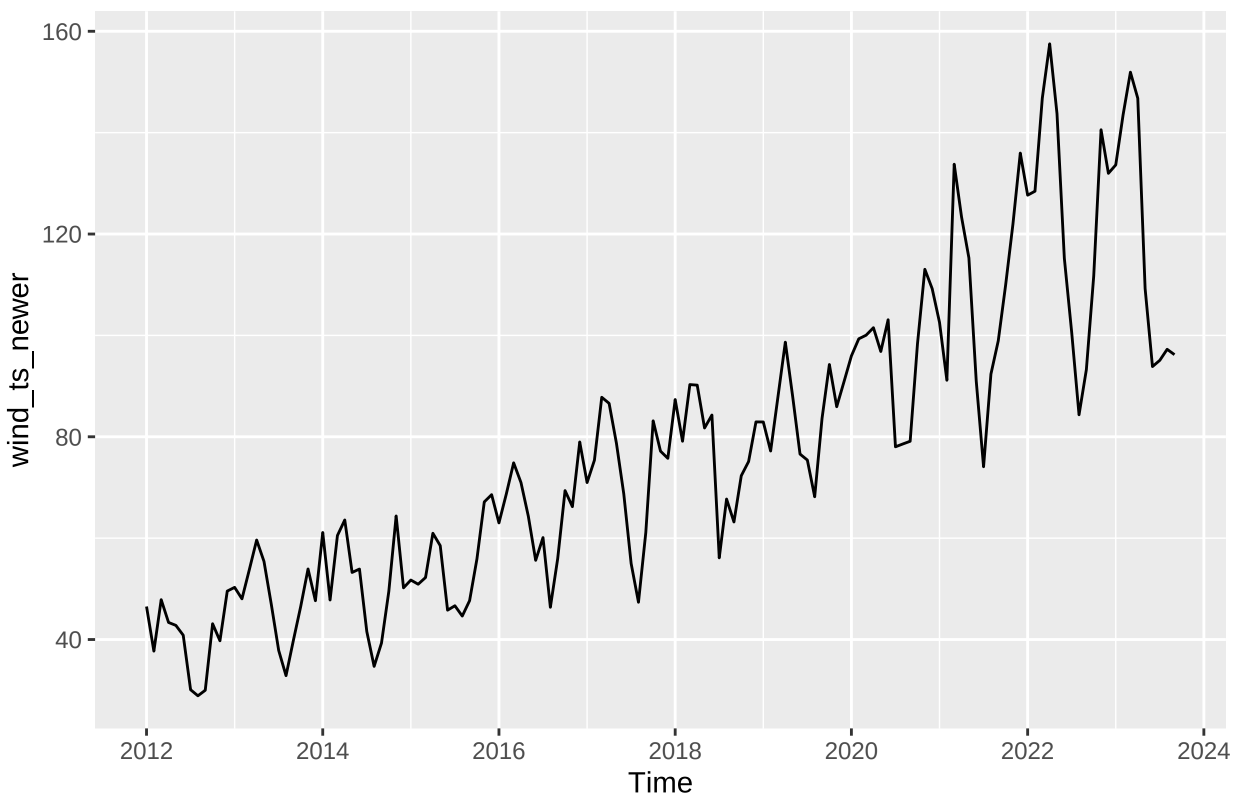
Q9

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?

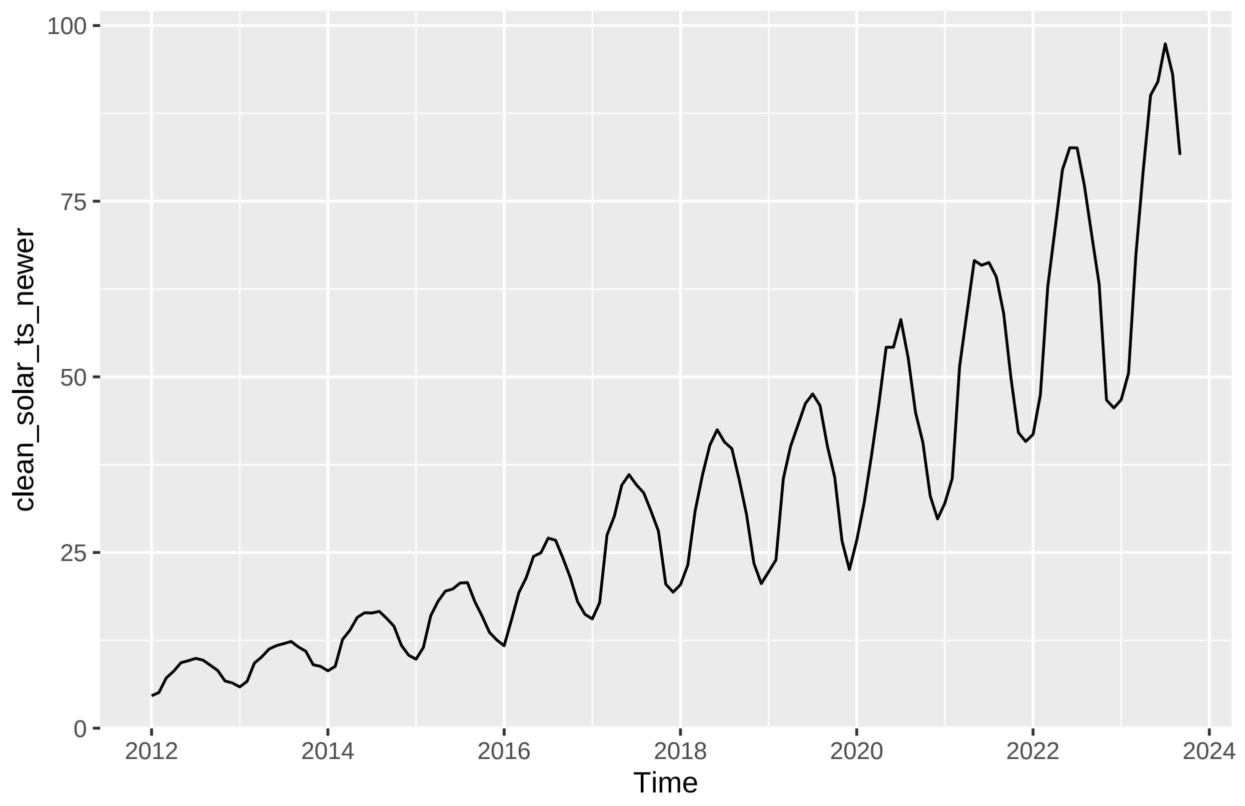
```
clean_wind_ts_newer<- tsclean(wind_ts_newer)

autoplot(clean_wind_ts_newer)
```



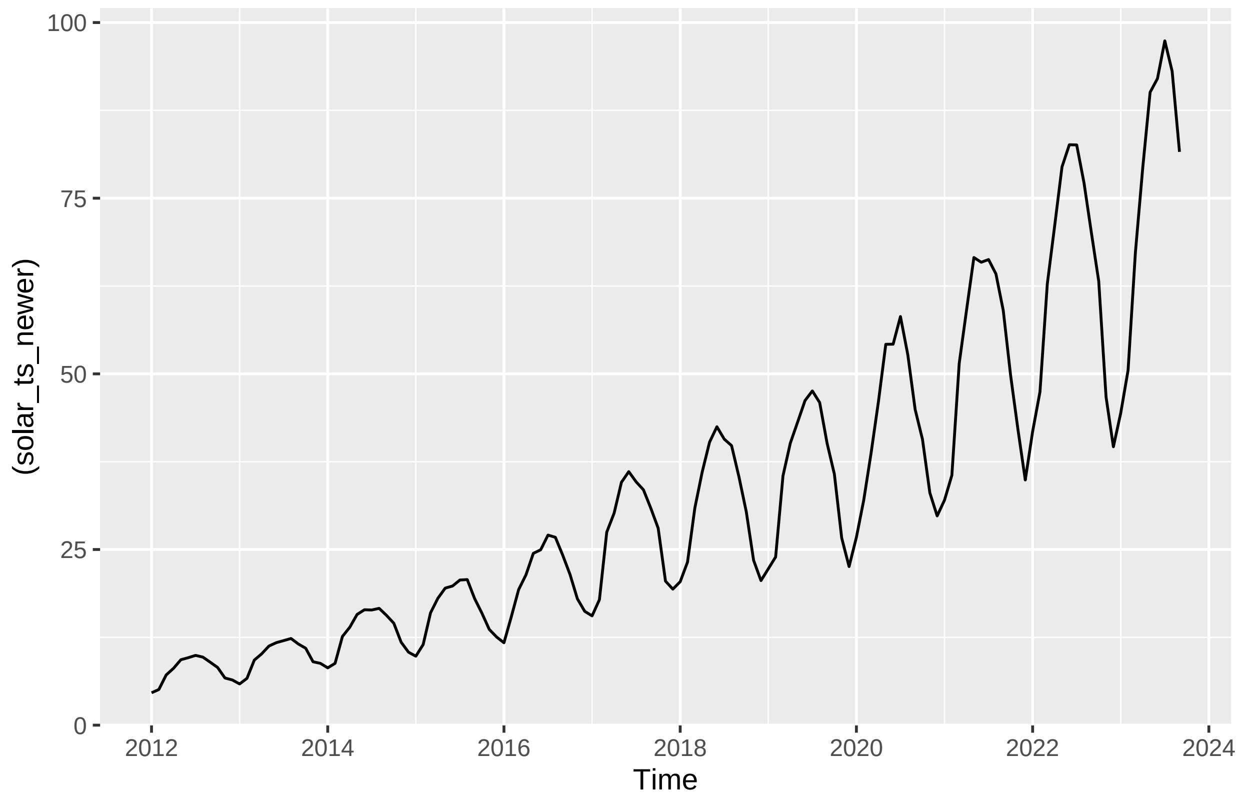
```
autoplot(wind_ts_newer)
```



```
clean_solar_ts_newer <- tsclean(solar_ts_newer)

autoplot(clean_solar_ts_newer)
```



```
autoplot((solar_ts_newer))
```



Answer:The clean function did not remove any outliers from wind that I can see on the autoplot, nor does it appear that it removed any outliers from the solar plots.