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

Assignment 5 - Due date 02/13/24

Samantha Pace

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
#install.packages("forecast")
#install.packages("tseries")
#install.packages("ggplot2")
#install.packages("Kendall")
#install.packages("lubridate")
#install.packages("tidyverse")
#install.packages("readxl")

library(readxl)
library(readxl)
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
```

```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 error
```

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

energy_data <- read_excel("Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx",

energy_data <- energy_data[2:609,]

head(energy_data)
```

```
## # A tibble: 6 x 14
##   Month               `Wood Energy Production` `Biofuels Production`
##   <dttm>              <chr>                    <chr>
## 1 1973-01-01 00:00:00 129.63                   Not Available
## 2 1973-02-01 00:00:00 117.194                  Not Available
## 3 1973-03-01 00:00:00 129.763                  Not Available
## 4 1973-04-01 00:00:00 125.462                  Not Available
## 5 1973-05-01 00:00:00 129.624                  Not Available
## 6 1973-06-01 00:00:00 125.435                  Not Available
## # i 11 more variables: `Total Biomass Energy Production` <chr>,
## #   `Total Renewable Energy Production` <chr>,
## #   `Hydroelectric Power Consumption` <chr>,
## #   `Geothermal Energy Consumption` <chr>, `Solar Energy Consumption` <chr>,
## #   `Wind Energy Consumption` <chr>, `Wood Energy Consumption` <chr>,
## #   `Waste Energy Consumption` <chr>, `Biofuels Consumption` <chr>,
## #   `Total Biomass Energy Consumption` <chr>, ...
```

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 a pipe to select, mutate, and drop NA's; added underscores to column names as well

colnames(energy_data)[8] <- "Solar.Energy.Consumption"
colnames(energy_data)[9] <- "Wind.Energy.Consumption"

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

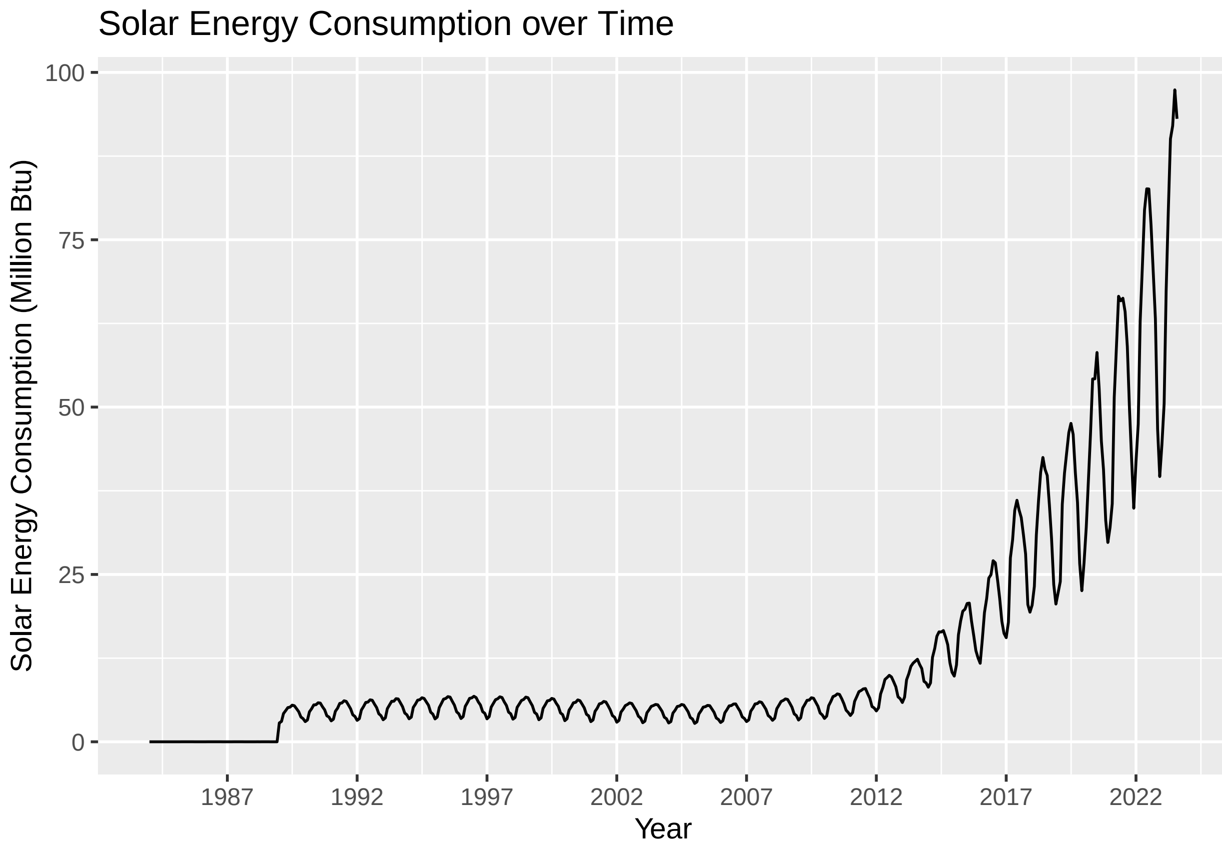
```
## Warning: There was 1 warning in ‘mutate()‘.
## i In argument: ‘Solar.Energy.Consumption =
##   as.numeric(energy_data$Solar.Energy.Consumption)‘.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## Warning: There was 1 warning in ‘mutate()‘.
## i In argument: ‘Wind.Energy.Consumption =
##   as.numeric(energy_data$Wind.Energy.Consumption)‘.
## Caused by warning:
## ! NAs introduced by coercion
```

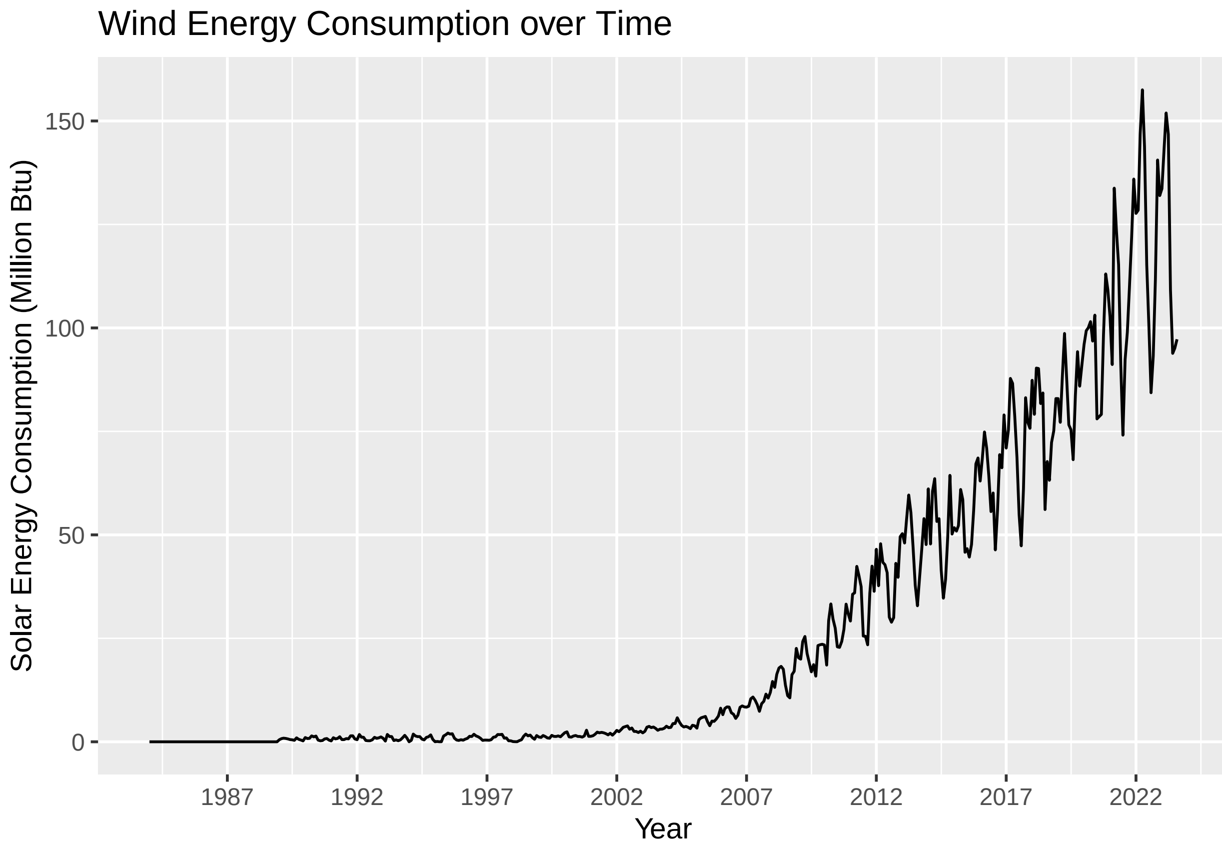
Q2

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")`

```
# graphing solar consumption
solar.graph <- ggplot(energy_data,
       aes(x = Month, y = Solar.Energy.Consumption)) +
  geom_line() +
  labs(x = "Year",
       y ="Solar Energy Consumption (Million Btu)",
       title = "Solar Energy Consumption over Time") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
print(solar.graph)
```



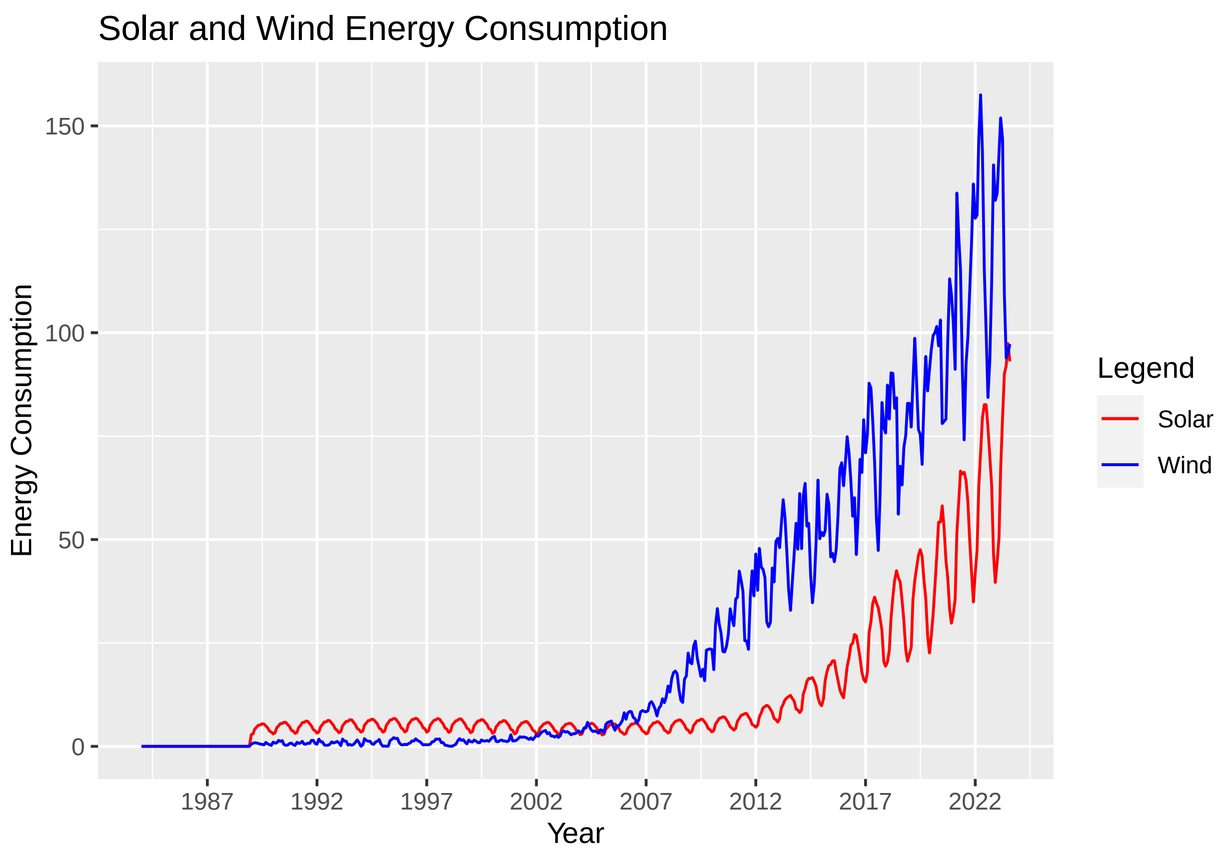

```r
# graphing wind energy consumption
wind.graph <- ggplot(energy_data,
                     aes(x = Month,
                         y = Wind.Energy.Consumption)) +
  geom_line() +
  labs(x = "Year",
       y = "Solar Energy Consumption (Million Btu)",
       title = "Wind Energy Consumption over Time") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
print(wind.graph)
```



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.

```
# graph of both wind and solar energy
solar.wind <- ggplot(energy_data,
       aes(x = Month)) +
  geom_line(aes(y = Solar.Energy.Consumption, col = 'Solar')) +
  geom_line(aes(y = Wind.Energy.Consumption, col = 'Wind')) +
  labs(x = "Year",
       y = "Energy Consumption",
       title = "Solar and Wind Energy Consumption") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  scale_color_manual(name = "Legend",
                     values = c("Solar" = 'red',
                                "Wind" = 'blue'))
print(solar.wind)
```



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.

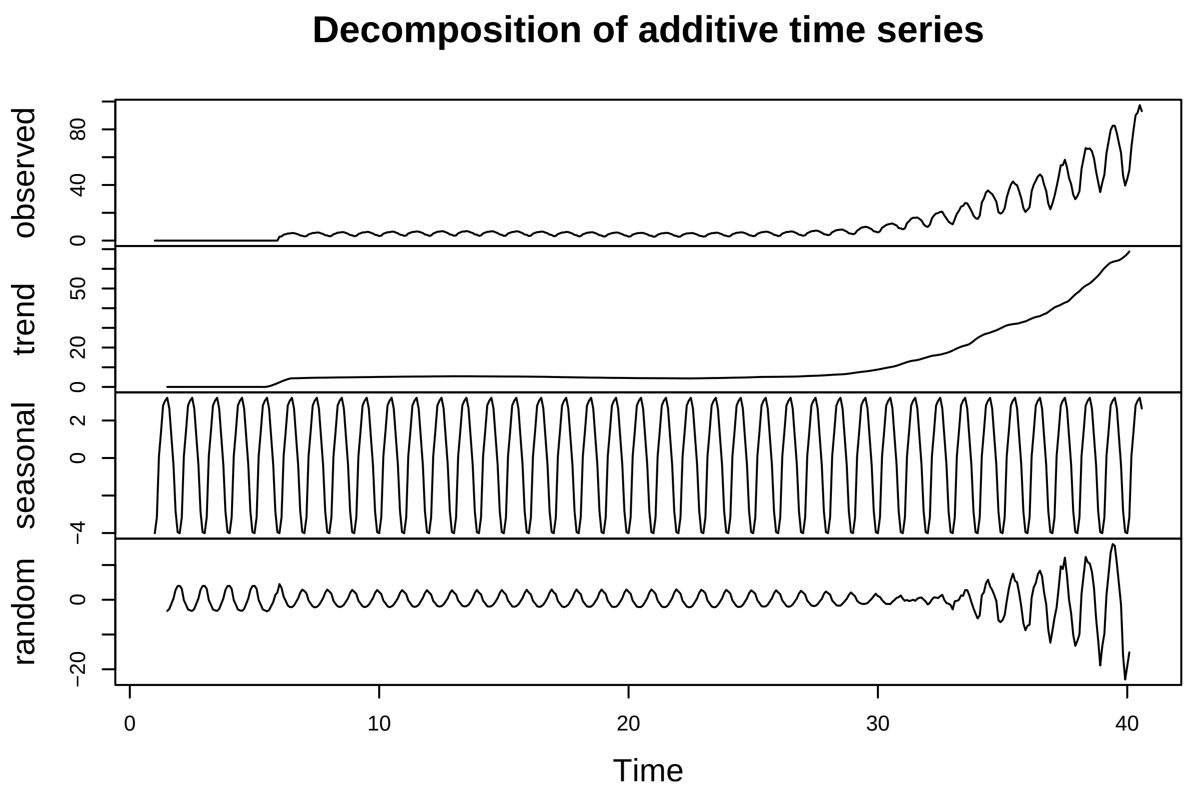
Additional info on `decompose()`.

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.

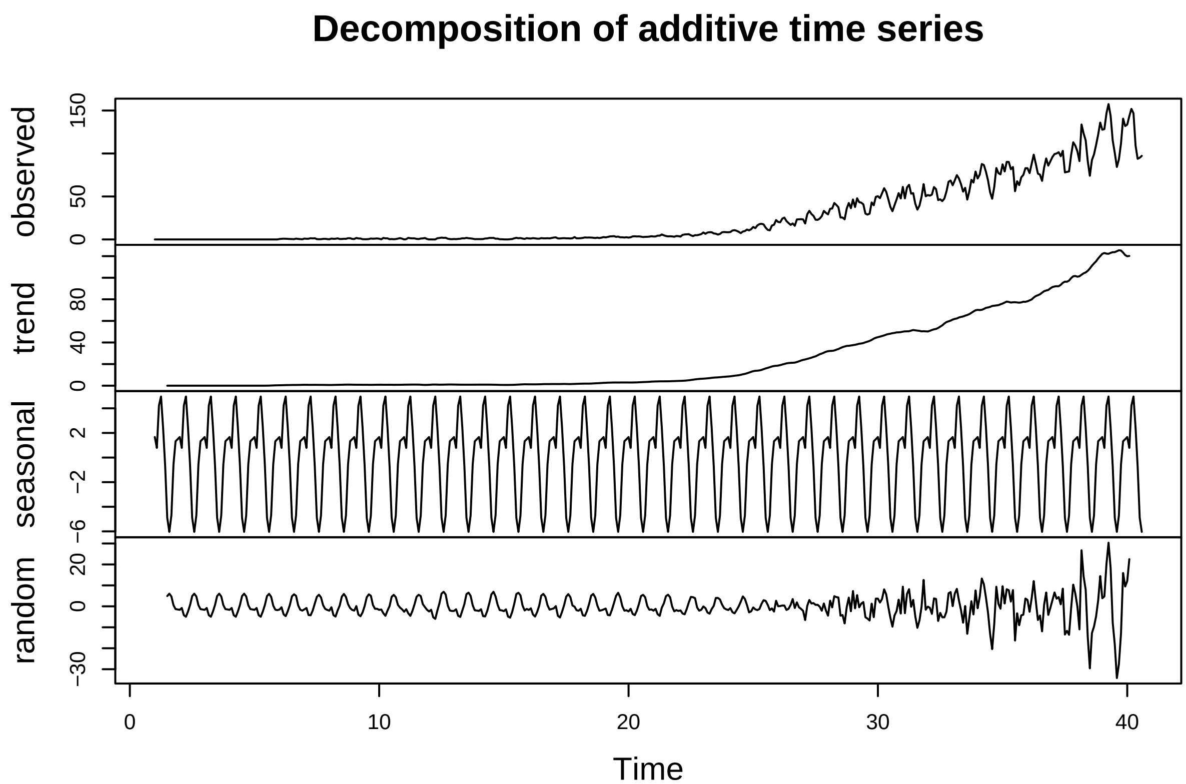
Q4

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
# Creating ts objects
solar.ts <- ts(energy_data$Solar.Energy.Consumption, frequency = 12)
wind.ts <- ts(energy_data$Wind.Energy.Consumption, frequency = 12)

# decomposing
decompose.solar <- decompose(solar.ts, "additive")
plot(decompose.solar)
```




```r
decompose.wind <- decompose(wind.ts, "additive")
plot(decompose.wind)
```



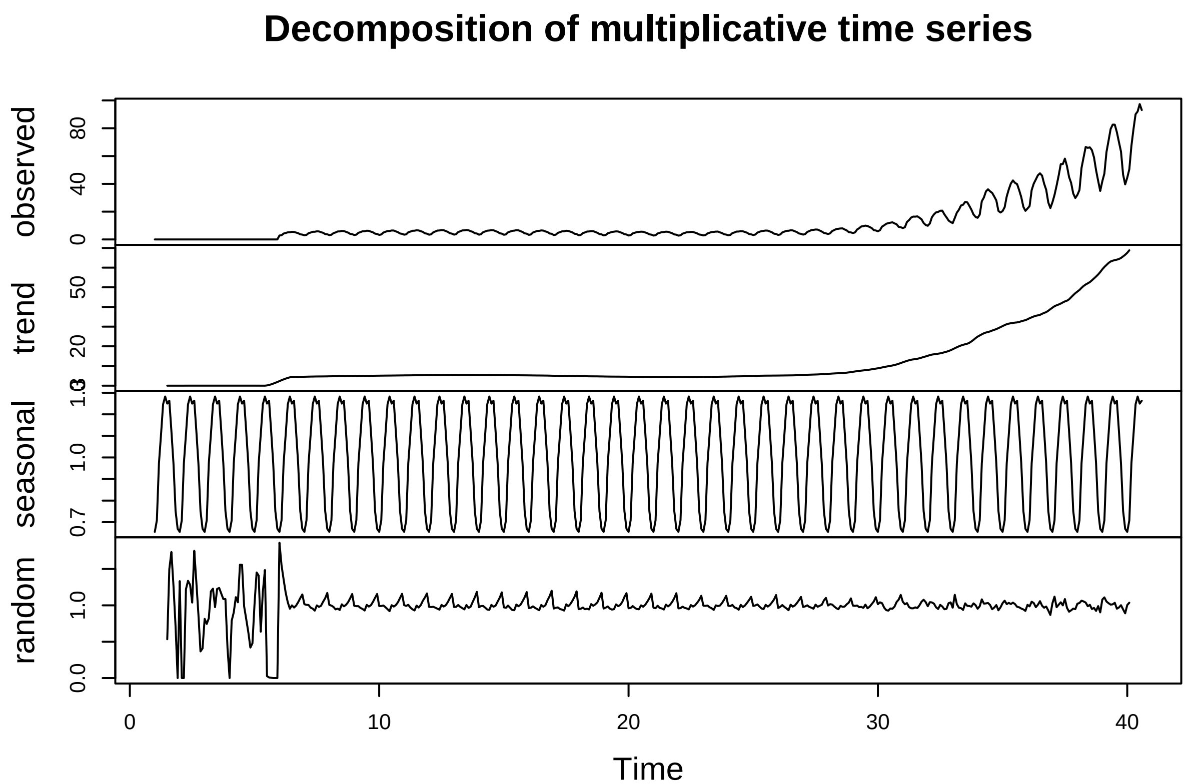
Answer: For the decomposed solar graphs, I can say that the trend is increasing and the random component has regularly spaced wave-like patterns suggesting a seasonality component remains. The magnitude of the random component waves are increasing between lags 35-40.

For the decomposed wind series, the trend is increasing for a majority of the time series, however at lag 40, it appears to take the smallest dip. The random component has a somewhat regular wave like pattern that is repetitive until lag 25 or so. After lag 25, the random component becomes more irregular and the magnitude of the peaks and valleys increases. It seems that until about lag 25 there is potentially still a seasonal component present.

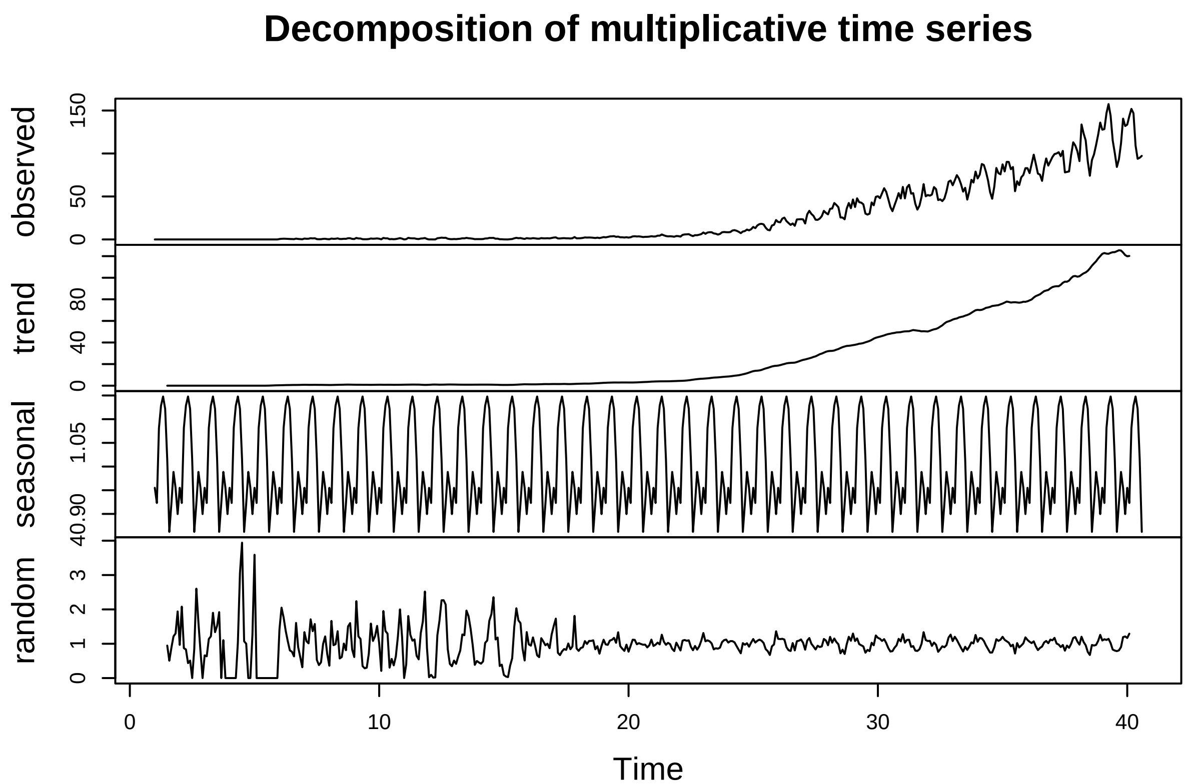
Q5

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
# decomposing as multipl.
decompose.solar.m <- decompose(solar.ts, "multiplicative")
plot(decompose.solar.m)
```




```r
decompose.wind.m <- decompose(wind.ts, "multiplicative")
plot(decompose.wind.m)
```



Answer: For the solar data, when changing to the multiplicative type, the random variable shows significant variability between lag 1 until lag 7. From lag 7, it has a small peaked, regular wave pattern until about lag 32 where the random component loses the regular peaks and has irregular short peaks.

For the wind data, the random component has wide ranging irregularities until about lag 18 and then the magnitude of the peaks and valleys decreases significantly. From lag 28 onwards, there is a slight regular wave pattern in the random variable.

Q6

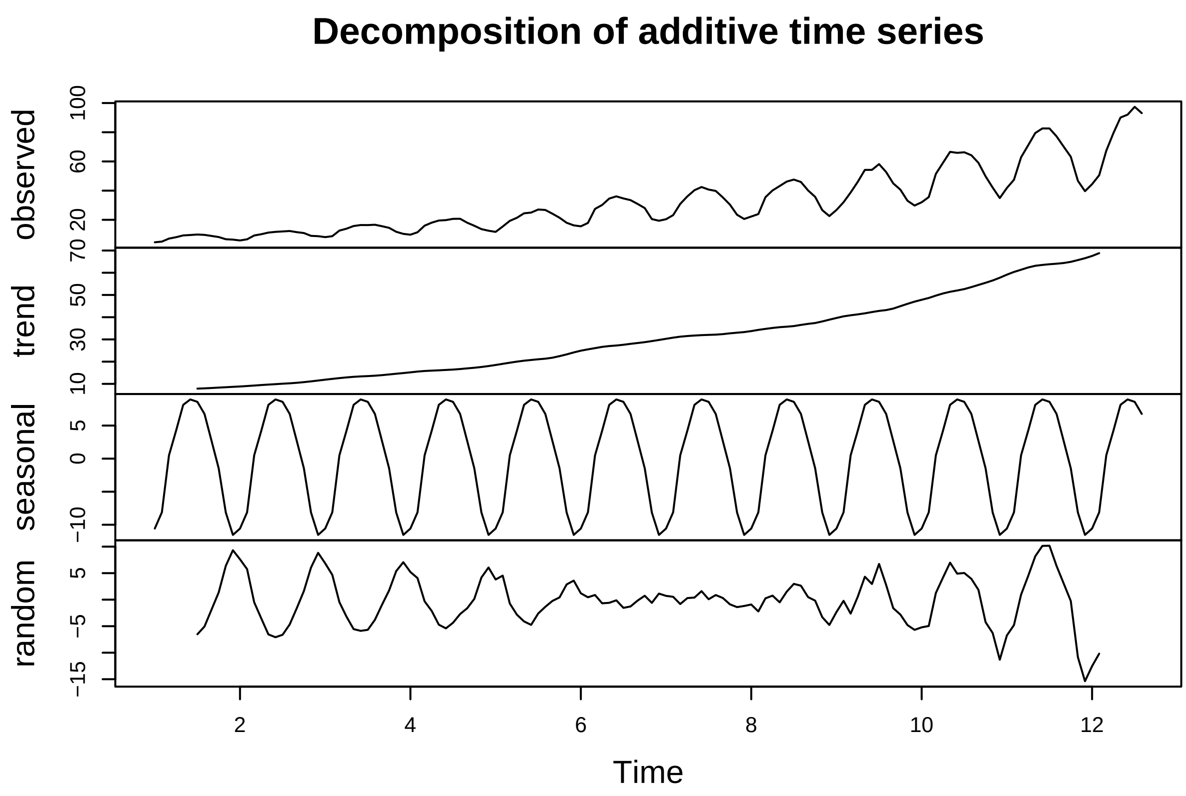
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: I don't think I need all the historical data in order to forecast the next six months of Solar or Wind consumption. The trends, seasonality, and random components of the last 10-15 years of data would capture and extrapolate enough data for a prediction, especially one of that short term. Until about 2007, the solar and wind consumption energy was fairly minimal, and likewise, the trend, seasonality, and random of this time period.

Q7

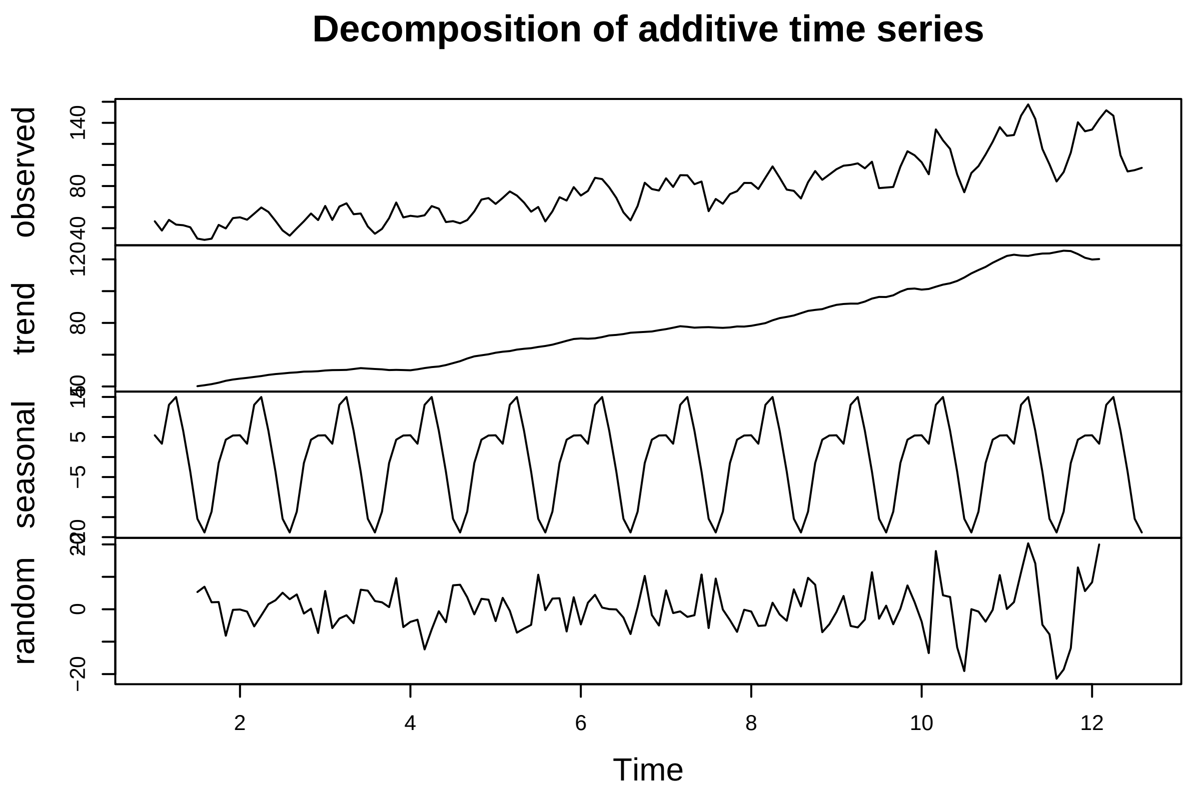
Create a new time series object where historical data starts on January 2012. Hint: use `filter()` function so that you don't need to point to row numbers, .i.e, `filter(xxxx, year(Date) >= 2012 )`. Apply the decompose function `type=additive` to this new time series. Comment the results. Does the random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

```
# filter data, after Jan 2012
energy_data2012 <-
  filter(energy_data, year(energy_data$Month) >= 2012)

# solar - ts, decompose, plot
solar.2012.ts <-
  ts(energy_data2012$Solar.Energy.Consumption,
     frequency = 12)

solar.2012.decomp <-
  decompose(solar.2012.ts,"additive")

plot(solar.2012.decomp)
```




```
# wind - ts, decompose, plot
wind.2012.ts <-
  ts(energy_data2012$Wind.Energy.Consumption,
     frequency = 12)

wind.2012.decomp <-
  decompose(wind.2012.ts, "additive")
```

```r
plot(wind.2012.decomp)
```



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.

> Answer: The random components of both the solar and wind do not appear random - they look to have some regularity in wave like patterns. It looks like that in 2007, the wind and solar consumption began to increase, and while it still exhibited the peaks and valleys of a seasonal trend, it was trending upward.It appears that this data is highlighting a level shift.

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.

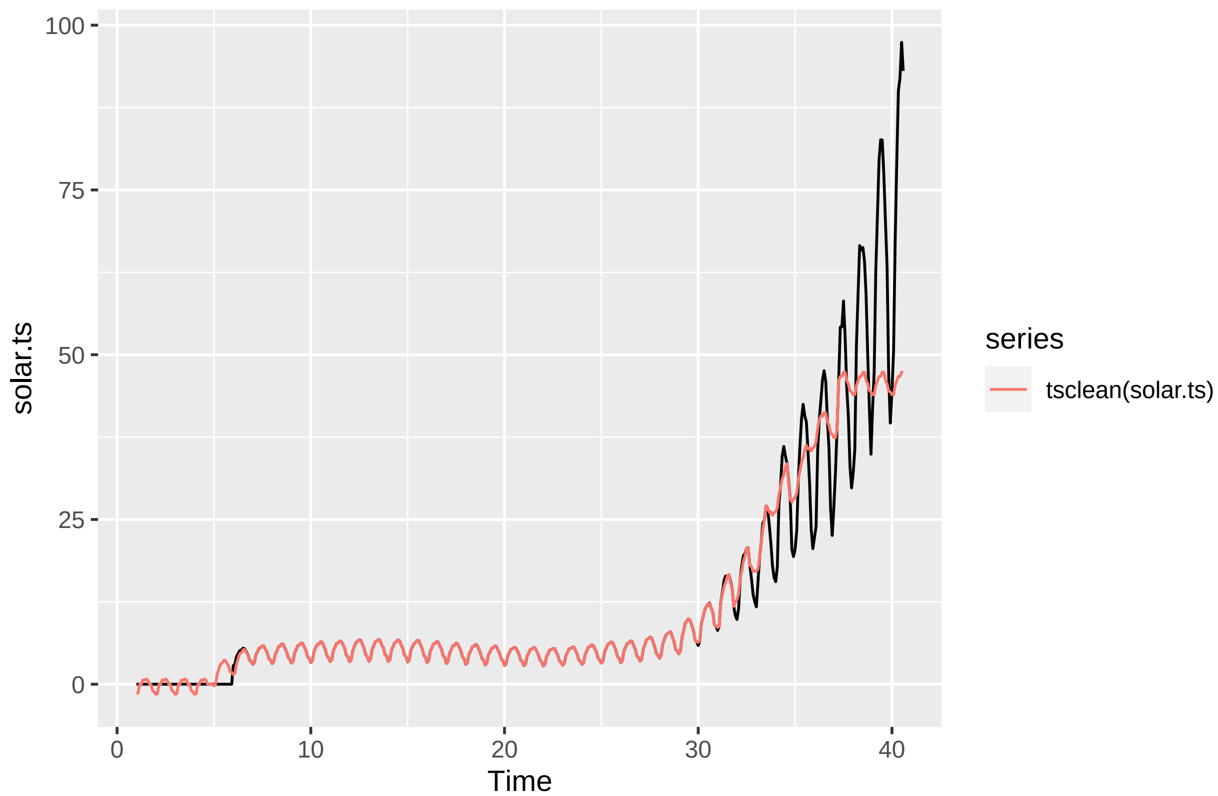
```r
# using tsclean on entire series - solar
tsclean(solar.ts)
```

```
##           Jan        Feb        Mar        Apr        May        Jun        Jul
## 1  -1.3755190 -1.3609351  0.0010000  0.0010000  0.6646788  0.6149104  0.7874419
## 2  -1.5273620 -1.4374444  0.0000000  0.0040000  0.6617139  0.6108281  0.7900242
## 3  -1.5263528 -1.4355276  0.0030000  0.0050000  0.6567133  0.6047023  0.7905551
```

```
## 4  -1.5274939 -1.4341818  0.0030000  0.0040000  0.6426623  0.5874235  0.7889671
## 5  -0.2456880  0.0030000  1.5044094  2.2300598  3.0526753  3.1912822  3.6055898
## 6   1.4584647  1.6559340  3.1506686  3.8632289  4.6486551  4.7869060  5.2331385
## 7   3.0150000  3.2560000  4.4890000  4.9460000  5.5150000  5.5340000  5.8320000
## 8   3.1250000  3.3690000  4.5610000  5.0260000  5.7390000  5.8030000  6.1330000
## 9   3.1960000  3.4510000  4.7520000  5.2840000  5.8830000  5.9310000  6.2740000
## 10  3.2880000  3.5700000  4.9540000  5.5150000  6.0530000  6.0580000  6.4450000
## 11  3.4320000  3.6800000  5.1370000  5.6400000  6.2160000  6.2920000  6.5840000
## 12  3.4150000  3.6920000  5.1050000  5.7350000  6.3700000  6.4710000  6.7500000
## 13  3.4720000  3.7600000  5.2930000  5.8650000  6.4870000  6.5620000  6.8040000
## 14  3.4220000  3.7620000  5.1760000  5.7480000  6.2920000  6.4330000  6.7210000
## 15  3.3750000  3.6290000  5.1380000  5.6640000  6.1730000  6.3270000  6.6790000
## 16  3.2940000  3.5700000  4.9840000  5.5360000  6.1210000  6.1840000  6.4820000
## 17  3.1550000  3.4400000  4.7330000  5.2390000  5.8320000  5.8950000  6.2610000
## 18  3.0080000  3.2610000  4.5420000  5.0200000  5.6720000  5.7550000  6.0290000
## 19  2.9230000  3.1890000  4.4350000  4.8770000  5.4080000  5.5850000  5.8120000
## 20  2.8480000  3.0830000  4.3340000  4.7950000  5.2980000  5.4210000  5.5790000
## 21  2.8090000  3.0190000  4.2890000  4.7220000  5.2780000  5.3430000  5.5750000
## 22  2.7420000  2.9680000  4.1580000  4.6380000  5.1780000  5.2470000  5.4380000
## 23  2.8810000  3.1240000  4.3270000  4.8210000  5.3680000  5.4100000  5.6400000
## 24  3.0160000  3.2690000  4.5790000  5.0510000  5.6580000  5.7070000  5.9810000
## 25  3.2060000  3.5180000  4.9330000  5.4780000  6.0290000  6.1810000  6.4080000
## 26  3.2470000  3.5780000  5.0530000  5.6140000  6.2000000  6.2290000  6.5820000
## 27  3.4850000  3.8330000  5.3860000  6.0330000  6.7600000  6.8950000  7.1580000
## 28  3.9240000  4.3740000  6.0430000  6.7600000  7.4960000  7.6670000  7.9030000
## 29  4.6070000  5.0770000  7.1480000  8.0960000  9.3160000  9.6050000  9.9340000
## 30  6.3615928  6.6630000  9.2600000 10.1510000 11.2640000 11.7450000 12.2123187
## 31  8.6202239  8.7990000 12.6240000 13.6519558 14.8395706 15.4590802 16.3950000
## 32 12.8272677 13.6171606 15.9890000 17.2480314 18.6297900 19.4286809 20.6600000
## 33 17.1379329 17.4486650 19.2970000 21.3340288 23.4568531 24.9822035 27.0560000
## 34 26.1990201 26.7677009 28.8170573 29.9806354 31.1801305 31.7759664 32.9412000
## 35 28.3292956 28.9912054 31.0768633 32.3751864 33.6595479 34.3342132 35.5992276
## 36 36.0983255 36.7612065 38.8330680 40.1460000 40.7191769 40.6822550 41.2369080
## 37 37.4366643 37.3867257 38.7310000 46.0450000 46.6881985 46.7207428 47.3460876
## 38 43.9355244 43.9527461 45.3612670 46.0450250 46.6891591 46.7227882 47.3491939
## 39 43.9339990 43.9516970 45.3577803 46.0449955 46.6900548 46.7247534 47.3522044
## 40 43.9331515 43.9511030 45.3559791 46.0449562 46.6905535 46.7258301 47.3537724
##           Aug        Sep        Oct        Nov        Dec
## 1   0.6416210  0.0030000  0.0020000 -1.0047980 -1.1484563
## 2   0.6441210  0.0050000  0.0030000 -0.9966313 -1.1426860
## 3   0.6445953  0.0050000  0.0040000 -0.9895071 -1.1390078
## 4   0.6428994  0.0030000  0.0020000  0.0010000  0.0010000
## 5   3.6565455  3.2134735  2.8059254  1.9109114  1.8164833
## 6   5.2840801  4.8392952  4.4323714  3.5706763  3.4650000
## 7   5.7810000  5.2280000  4.7820000  3.8840000  3.7000000
## 8   6.0350000  5.4870000  4.9350000  4.0340000  3.7920000
## 9   6.2050000  5.6280000  5.1150000  4.1870000  3.9140000
## 10  6.4010000  5.8180000  5.2430000  4.2970000  4.0390000
## 11  6.4710000  5.9490000  5.4580000  4.4150000  4.1360000
## 12  6.6740000  6.0470000  5.4710000  4.4730000  4.1920000
## 13  6.6020000  5.9560000  5.4940000  4.4950000  4.2620000
## 14  6.6120000  5.9580000  5.4370000  4.4240000  4.1870000
## 15  6.5840000  5.9560000  5.3950000  4.3890000  4.1470000
## 16  6.3770000  5.7750000  5.3110000  4.3070000  4.0680000
```

```
## 17  6.1440000  5.6080000  5.0560000  4.1030000  3.8820000
## 18  5.9470000  5.3800000  4.7730000  3.9050000  3.6750000
## 19  5.7190000  5.1620000  4.6470000  3.8210000  3.5510000
## 20  5.5220000  5.0340000  4.5350000  3.6690000  3.4520000
## 21  5.4880000  4.9860000  4.4700000  3.6250000  3.4180000
## 22  5.3960000  4.8940000  4.3990000  3.5470000  3.3350000
## 23  5.6570000  5.0850000  4.5770000  3.7180000  3.4870000
## 24  5.8860000  5.3670000  4.8480000  3.9220000  3.6620000
## 25  6.3260000  5.7580000  5.1660000  4.1670000  3.9260000
## 26  6.5010000  5.8920000  5.3070000  4.3000000  4.0230000
## 27  7.0720000  6.4330000  5.6930000  4.7210000  4.3840000
## 28  7.9580000  7.1780000  6.5070000  5.2590000  5.0560000
## 29  9.6850000  8.9600000  8.2140000  6.7150000  6.4390000
## 30 12.2565362 11.5510000 10.9460000  9.0280000  8.8000000
## 31 16.6240000 15.3426651 14.5070000 11.8050000 12.4070276
## 32 20.7200000 18.0260000 17.8572168 17.0923213 17.2085001
## 33 26.7410000 26.0968366 26.1710361 25.6600674 26.0262120
## 34 33.4920000 30.4073731 28.0420000 27.6258341 28.0769415
## 35 36.2476996 35.6670245 35.8068298 35.3980946 35.8470847
## 36 41.1715564 39.8664156 39.2889367 38.1682551 37.9023615
## 37 47.3478652 46.0992115 45.5853580 44.5336399 44.3337814
## 38 47.3509960 46.0990431 45.5838371 44.5331682 44.3330046
## 39 47.3540309 46.0987787 45.5822394 44.5326388 44.3321976
## 40 47.3554462
```

```
autoplot(solar.ts) +
  autolayer(tsclean(solar.ts))
```



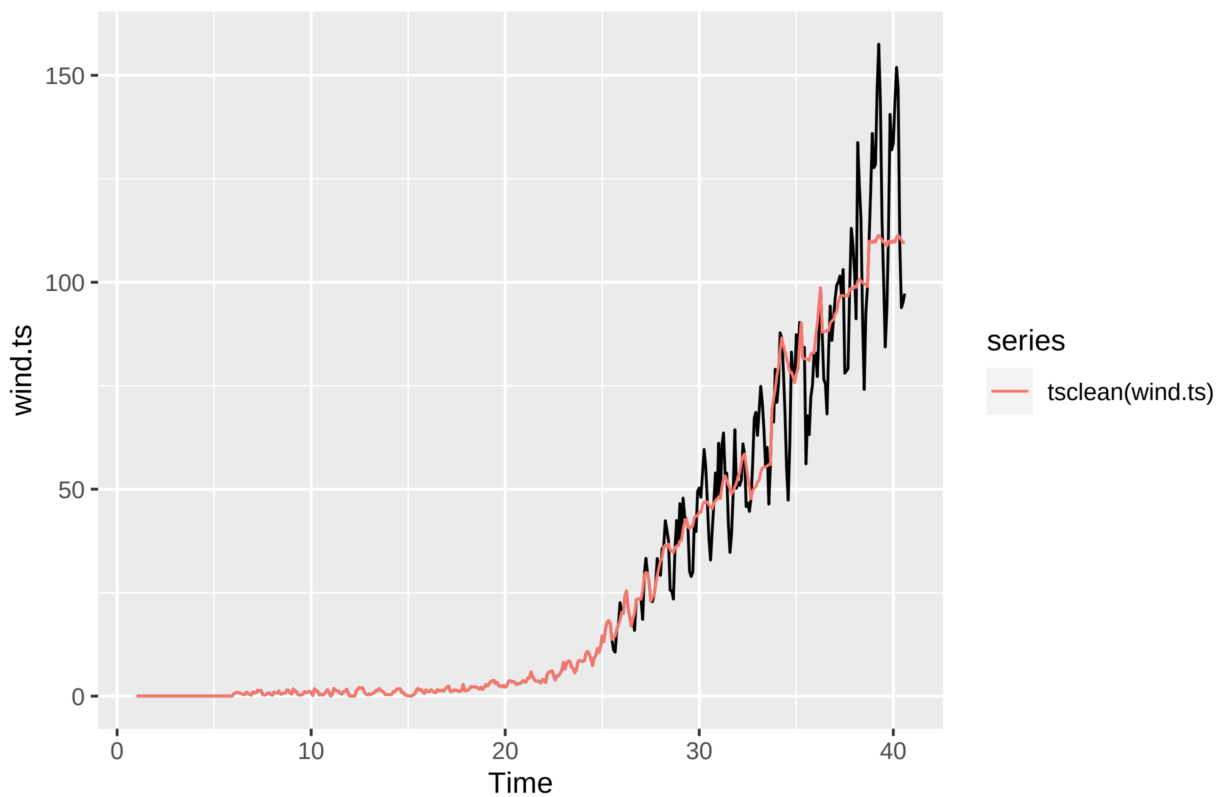

```r
# tsclean - wind
tsclean(wind.ts)
```

```
##          Jan     Feb     Mar     Apr     May     Jun     Jul
## 1    0.00000 0.00100 0.00100 0.00200 0.00300 0.00200 0.00200
## 2    0.00200 0.00400 0.00200 0.00200 0.00100 0.00100 0.00000
## 3    0.00100 0.00100 0.00200 0.00200 0.00300 0.00200 0.00200
## 4    0.00000 0.00000 0.00000 0.00000 0.00100 0.00300 0.00400
## 5    0.00200 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
## 6    0.49700 0.76300 0.87300 0.81200 0.69700 0.53400 0.48500
## 7    1.02800 0.78900 0.83700 1.43400 1.19500 1.38700 0.40600
## 8    0.21200 0.98900 0.70700 0.81300 1.23700 0.53000 0.53000
## 9    0.51200 1.75900 1.12000 1.08800 0.32000 0.25600 0.25600
## 10   0.84400 0.16900 1.77300 1.26600 1.26600 0.29500 0.46400
## 11   0.00000 0.34300 1.86500 1.40800 1.25600 1.25600 0.64700
## 12   0.01900 0.08000 0.01600 0.02300 1.39400 1.70100 2.11500
## 13   0.50300 0.38200 0.64000 0.83400 1.33600 1.28200 1.82600
## 14   0.41100 0.37200 0.50700 1.10600 1.19600 1.76400 1.73800
## 15   0.05900 0.02800 0.02100 0.29300 0.48900 1.34800 1.86800
## 16   1.15000 1.05400 1.52400 1.27400 0.94000 0.86600 1.55600
## 17   2.20200 2.40900 1.19400 1.12600 1.39200 1.52500 1.29600
## 18   1.32800 1.47100 1.81600 2.33600 2.16700 2.28500 2.16700
## 19   2.76900 2.43600 2.90700 3.49500 3.67700 3.84300 3.03800
## 20   2.15800 2.54300 3.53500 3.72900 3.43400 3.57300 3.25300
## 21   3.41000 3.48600 4.40600 4.41800 5.80600 4.76700 3.97300
## 22   3.86100 3.29800 5.32500 5.79200 5.95900 6.13100 4.84800
```

```
## 23   8.13000   6.55800   8.04800   8.43400   8.38900   7.00100   6.67100
## 24   8.36700   8.59800  10.39700  10.82200  10.07300   8.94100   7.36400
## 25  14.58000  13.14200  16.31600  17.82900  18.22100  17.53900  13.67700
## 26  20.30400  19.96800  24.22200  25.44600  21.36600  19.10500  16.90600
## 27  23.38700  25.40518  29.30600  29.91801  29.67600  27.46300  22.94200
## 28  32.63867  33.36765  35.97800  36.48108  36.07045  36.74924  35.29141
## 29  37.52345  37.70900  40.38571  42.07354  42.78800  40.85000  40.54926
## 30  44.46734  44.39011  46.12854  46.92529  46.73308  46.90900  46.12125
## 31  48.18714  47.79800  50.54095  52.38876  53.23200  52.18137  50.99613
## 32  53.16327  53.89141  56.35965  57.94251  58.52000  55.81990  53.04454
## 33  51.81288  52.06599  54.04811  55.15438  55.25449  55.41333  55.55624
## 34  77.42684  79.61718  83.53862  86.59000  84.84024  83.17243  81.48520
## 35  77.92929  79.12300  85.08459  90.18200  81.72800  81.55361  81.35631
## 36  86.95553  89.99091  94.75532  98.65900  87.95900  88.02543  88.07323
## 37  92.40116  92.90608  95.13584  96.50845  96.82400  96.75656  96.67478
## 38  98.99745  98.72631 100.17939 100.77488 100.31313  99.94029  99.55255
## 39 110.08091 109.64573 110.93416 111.36460 110.73763 110.19971 109.64633
## 40 110.08170 109.64644 110.93464 111.36421 110.73720 110.19860 109.64438
##          Aug       Sep       Oct       Nov       Dec
## 1    0.00100   0.00200   0.00300   0.00300   0.00400
## 2    0.00100   0.00300   0.00200   0.00100   0.00000
## 3    0.00200   0.00000   0.00100   0.00000   0.00000
## 4    0.00100   0.00100   0.00000   0.00000   0.00100
## 5    0.00000   0.00000   0.00000   0.00000   0.00000
## 6    0.35200   0.95600   0.59900   0.41900   0.21900
## 7    0.23900   0.31100   0.69300   0.78900   0.40600
## 8    0.77700   0.70700   1.44800   1.44800   0.67100
## 9    0.48000   1.08800   0.83200   0.96000   1.18400
## 10   0.25300   0.46400   0.97100   1.56200   0.92900
## 11   0.45700   1.06600   1.21800   1.67500   0.57100
## 12   1.86200   1.95500   0.87600   0.42700   0.32900
## 13   1.41600   1.19900   0.86300   0.33700   0.41700
## 14   1.80900   0.88800   0.93700   0.24800   0.24400
## 15   1.43800   1.62300   1.01900   0.61800   1.51900
## 16   1.30400   1.28800   1.42800   1.20900   1.72000
## 17   1.28400   1.13400   1.40400   2.80800   1.30900
## 18   1.96900   1.67300   2.07100   1.60500   2.10100
## 19   3.33200   2.51100   2.50600   2.23800   2.57700
## 20   2.78200   3.05400   3.06100   3.28000   3.77000
## 21   3.58500   3.71800   3.51100   3.18100   3.99900
## 22   3.88300   5.01000   4.93400   5.49200   6.23700
## 23   5.64700   6.41200   8.33300   8.66800   8.43300
## 24   9.21000   9.78100  11.52100  10.56000  11.90900
## 25  14.24874  14.47757  16.22900  17.03800  18.36912
## 26  18.64500  20.12595  23.24800  23.45800  23.56300
## 27  23.79277  24.24400  27.10400  29.12785  30.91000
## 28  35.15043  34.61677  35.91300  36.25643  36.35800
## 29  41.02752  41.11994  43.11300  43.47997  43.60499
## 30  45.88870  45.24609  46.52300  47.23108  47.65600
## 31  50.14134  48.85229  49.50100  50.65938  51.49326
## 32  50.57361  47.67100  49.58212  50.26551  50.59777
## 33  55.97717  55.96900  69.38400  72.00854  74.25518
## 34  80.07498  78.23538  78.29970  77.16200  75.74900
## 35  81.43515  81.08432  82.63020  82.98841  82.93300
```

```
## 36  88.40010  88.30834  90.10592  90.71252  90.90900
## 37  96.87521  96.66804  98.34300  98.51913  98.28870
## 38  99.44790  98.94100 109.91800 109.92966 109.53615
## 39 109.37699 108.71073 109.91988 109.93102 109.53842
## 40 109.37478
```

```
autoplot(wind.ts) +
  autolayer(tsclean(wind.ts))
```



Answer: The tsclean function removed many data points it considered to be outliers in both the solar and wind datasets.

Q9

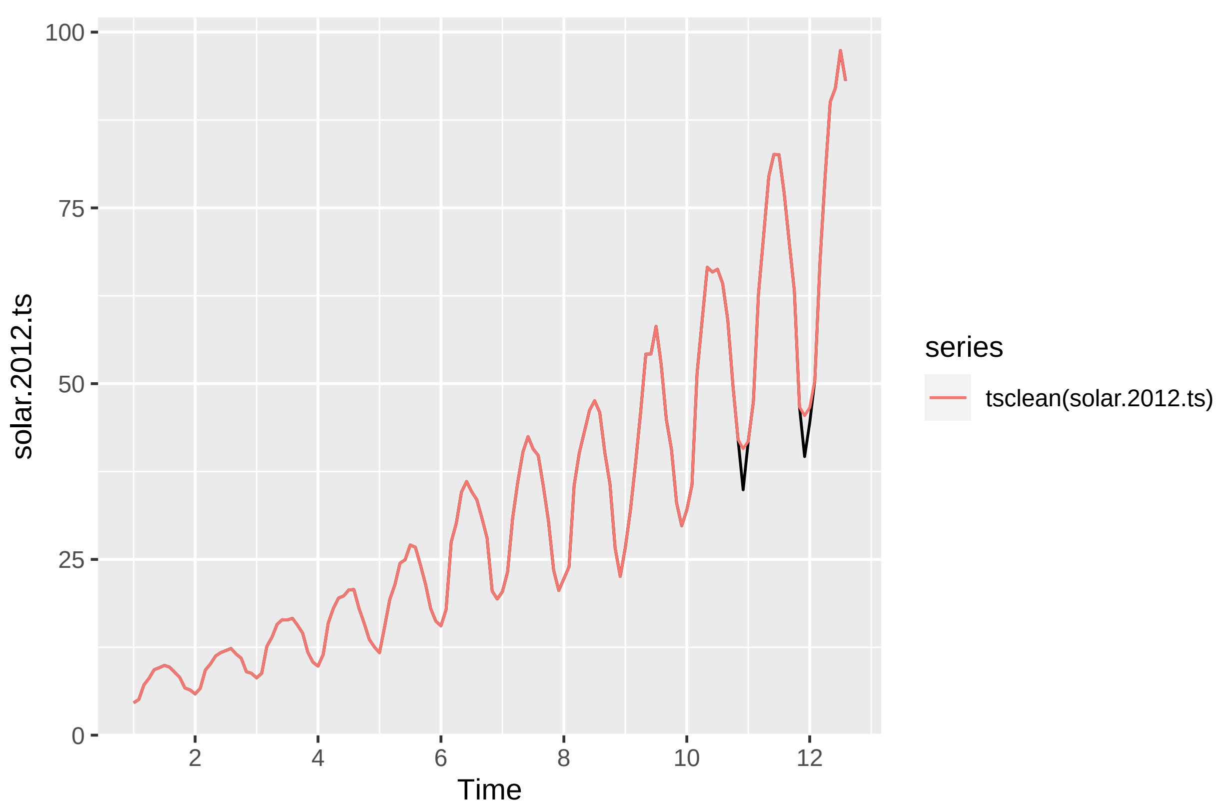
Redo number Q8 but now with the time series you created on Q7, i.e., the series starting in 2014. Using what `autoplot()` again what happened now? Did the function removed any outliers from the series?

```
# tsclean for 2012 dataset for solar
tsclean(solar.2012.ts)
```

```
##         Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1   4.60700  5.07700  7.14800  8.09600  9.31600  9.60500  9.93400  9.68500
## 2   5.86900  6.66300  9.26000 10.15100 11.26400 11.74500 12.03800 12.33600
## 3   8.15700  8.79900 12.62400 13.93400 15.75800 16.42800 16.39500 16.62400
## 4   9.81500 11.48000 15.98900 18.05800 19.51000 19.80400 20.66000 20.72000
```

```
## 5  11.72800 15.42800 19.29700 21.40100 24.45900 24.95500 27.05600 26.74100
## 6  15.55500 17.85700 27.47200 30.17500 34.56700 36.08300 34.63500 33.49200
## 7  20.41700 23.21300 30.91800 36.04900 40.27700 42.47600 40.71500 39.78500
## 8  22.24900 23.94200 35.49000 40.14600 43.14600 46.19800 47.57200 45.91400
## 9  26.74100 32.04900 38.73100 46.04500 54.20800 54.21900 58.15900 52.71200
## 10 32.03400 35.56500 51.47700 59.06800 66.55900 65.88200 66.26900 64.22900
## 11 41.80800 47.44600 62.80600 71.07200 79.45900 82.61100 82.58400 77.16900
## 12 46.59600 50.52300 67.31200 79.38000 90.07900 92.02400 97.39700 93.06800
##         Sep      Oct      Nov      Dec
## 1   8.96000  8.21400  6.71500  6.43900
## 2  11.55100 10.94600  9.02800  8.80000
## 3  15.63100 14.50700 11.80500 10.38700
## 4  18.02600 15.93800 13.62800 12.54700
## 5  24.19900 21.43800 17.98500 16.20200
## 6  30.88100 28.04200 20.50100 19.36200
## 7  35.35500 30.38600 23.46800 20.57600
## 8  40.15700 35.72400 26.63400 22.57300
## 9  44.93300 40.67400 33.06800 29.77800
## 10 59.02600 49.77800 42.08200 40.75125
## 11 70.10500 63.19000 46.70800 45.44538
## 12
```

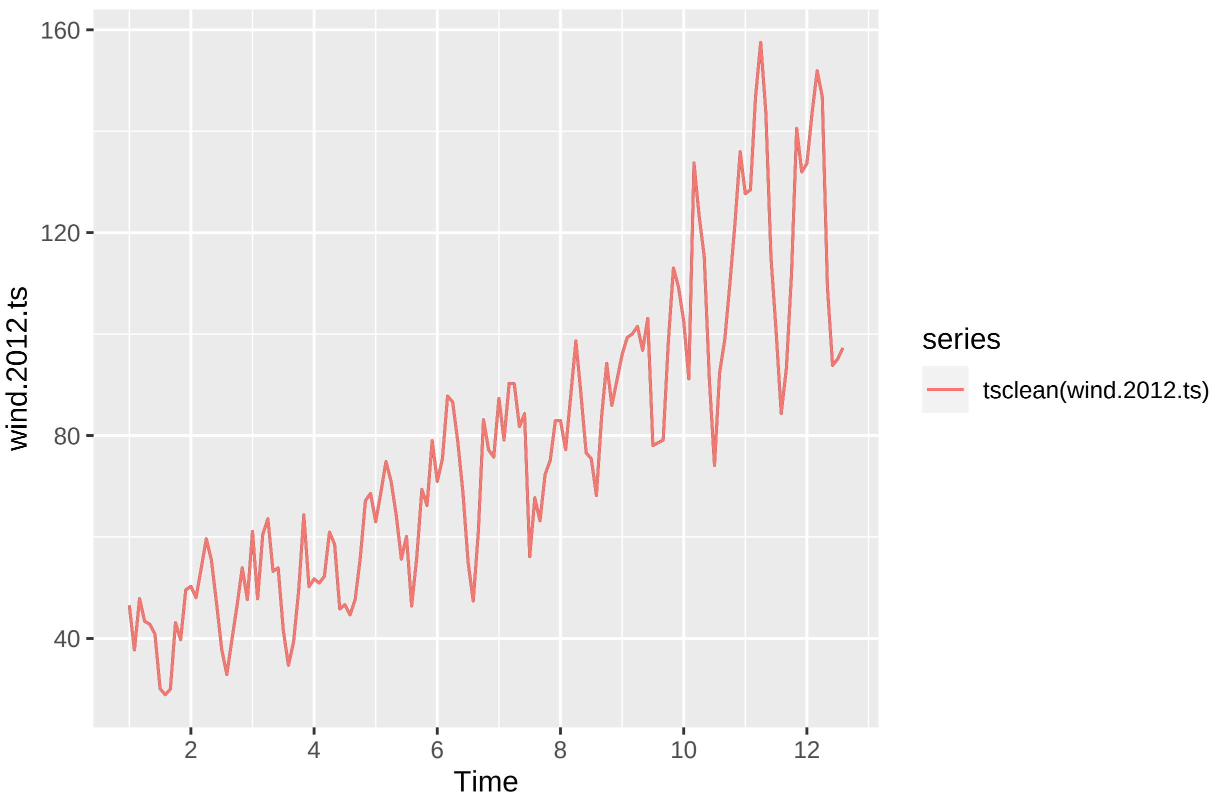
```r
autoplot(solar.2012.ts) +
  autolayer(tsclean(solar.2012.ts))
```



```r
# tsclean for 2012 dataset for wind
tsclean(wind.2012.ts)
```

```
##        Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep
## 1   46.514  37.709  47.858  43.364  42.788  40.850  30.099  28.896  29.990
## 2   50.288  48.026  53.758  59.629  55.406  46.909  37.851  32.871  39.832
## 3   61.113  47.798  60.515  63.584  53.232  53.906  41.583  34.702  39.305
## 4   51.733  50.912  52.231  60.963  58.520  45.793  46.661  44.629  47.671
## 5   63.007  68.712  74.857  70.967  64.309  55.627  60.114  46.367  55.969
## 6   70.965  75.375  87.793  86.590  78.707  68.724  55.001  47.355  61.115
## 7   87.343  79.123  90.294  90.182  81.728  84.286  56.116  67.716  63.189
## 8   82.917  77.189  87.936  98.659  87.959  76.586  75.408  68.165  83.640
## 9   95.950  99.325 100.039 101.515  96.824 103.085  78.019  78.576  79.111
## 10 102.566  91.153 133.768 123.370 115.280  91.003  74.093  92.367  98.941
## 11 127.664 128.443 146.820 157.522 143.726 115.215 100.569  84.339  93.254
## 12 133.636 143.503 151.927 146.775 109.246  93.850  95.085  97.256
##        Oct     Nov     Dec
## 1   43.113  39.745  49.557
## 2   46.523  53.921  47.656
## 3   49.501  64.374  50.195
## 4   55.889  67.154  68.576
## 5   69.384  66.212  78.973
## 6   83.146  77.162  75.749
## 7   72.314  75.118  82.933
## 8   94.255  85.929  90.909
## 9   98.343 113.038 109.220
## 10 109.918 121.983 135.965
## 11 111.725 140.570 131.975
## 12
```

```r
autoplot(wind.2012.ts) +
  autolayer(tsclean(wind.2012.ts))
```



Answer: When looking at the dataset that is 2012 and onwards, there are far less, and potentially no, outliers removed. The outliers seem to be relative to the average of the dataset, and since the low values found in the more historic data are not part of this dataset, the average is not skewing lower. There appear to be no outliers removed on the 2012 wind dataset, and just a few for the 2012 solar dataset.