ENV 790.30 - Time Series Analysis for Energy Data | Spring 2022

Assignment 5 - Due date 02/28/22

Alex Baad

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 project open the first thing you will do is change "Student Name" on line 3 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. Rename the pdf file such that it includes your first and last name (e.g., "LuanaLima_TSA_A05_Sp22.Rmd"). Submit this pdf using Sakai.

R packages needed for this assignment are listed below. 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(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
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 packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.6     v purrr   0.3.4
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

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 January 2021
Monthly Energy Review.

```
#Importing data set - using xlsx package
energy_data <- read_xlsx(path="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.
```

```
## New names:
## * `` -> ...1
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * ...
```

```
#Now let's extract the column names from row 11 only
read_col_names <- read_xlsx(path="./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Sour
```

```
## New names:
## * `` -> ...1
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * ...
```

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

```
## # A tibble: 6 x 14
##   `c(94694400, 9737280~ `c(129.63, 117.19~ `c("Not Available~ `c(129.787, 117.3~
##   <dttm>                              <dbl> <chr>                            <dbl>
## 1 1973-02-01 00:00:00                  117. Not Available                     117.
## 2 1973-03-01 00:00:00                  130. Not Available                     130.
## 3 1973-04-01 00:00:00                  125. Not Available                     126.
## 4 1973-05-01 00:00:00                  130. Not Available                     130.
## 5 1973-06-01 00:00:00                  125. Not Available                     126.
## 6 1973-07-01 00:00:00                  130. Not Available                     130.
## # ... with 10 more variables: ...
```

```
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!

```
energy_data <- energy_data[,1:9]
energy_data <- energy_data[,-c(2:7)]
colnames(energy_data)=c("Date","Solar","Wind")
energy_data <- data.frame("Date"=energy_data$Date,"Solar"=as.numeric(energy_data$Solar),"Wind"=as.numeri
```

```
## Warning in data.frame(Date = energy_data$Date, Solar =
## as.numeric(energy_data$Solar), : NAs introduced by coercion
```

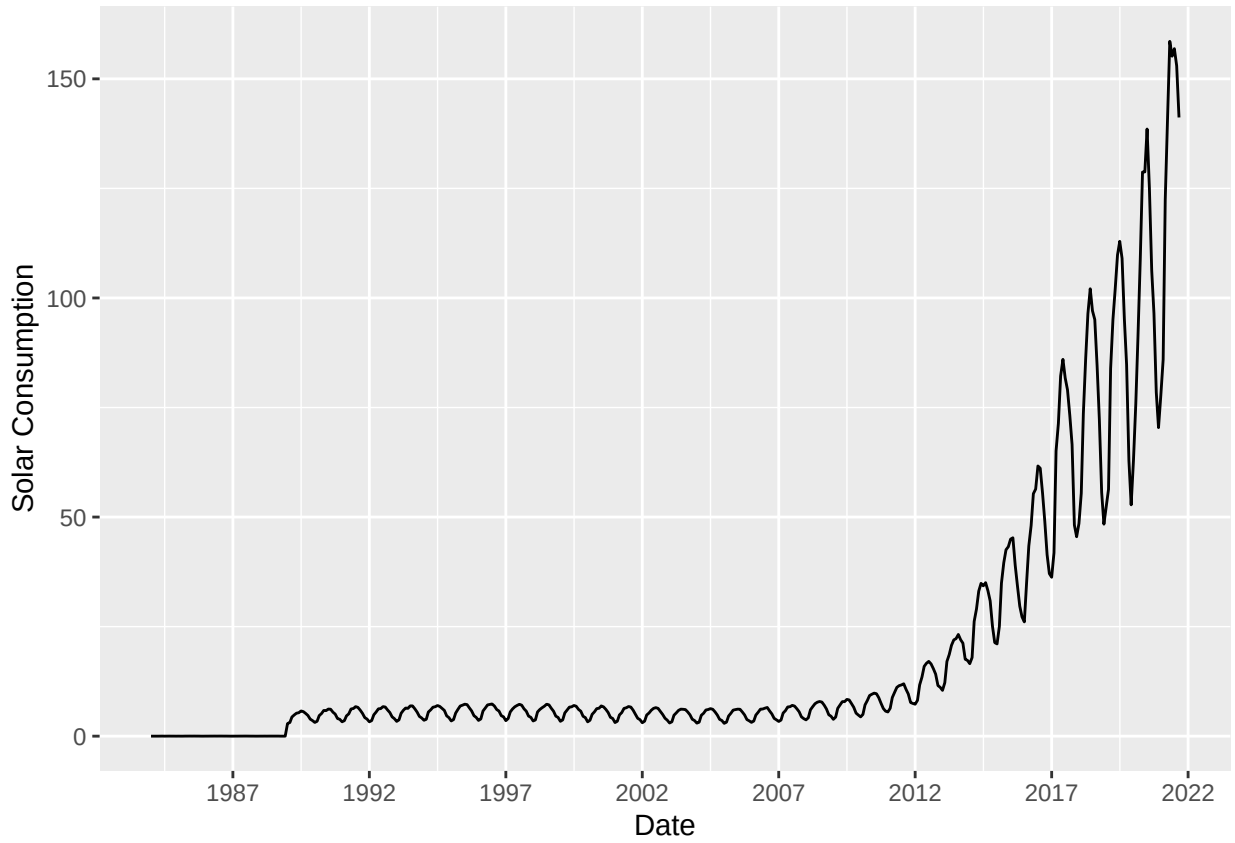
```
## Warning in data.frame(Date = energy_data$Date, Solar =
## as.numeric(energy_data$Solar), : NAs introduced by coercion
```

```
energy_data$Date <- ymd(energy_data$Date)
energy_data <- na.omit(energy_data)
ts_energy_data <- ts(energy_data[,2:3],frequency=12)
```

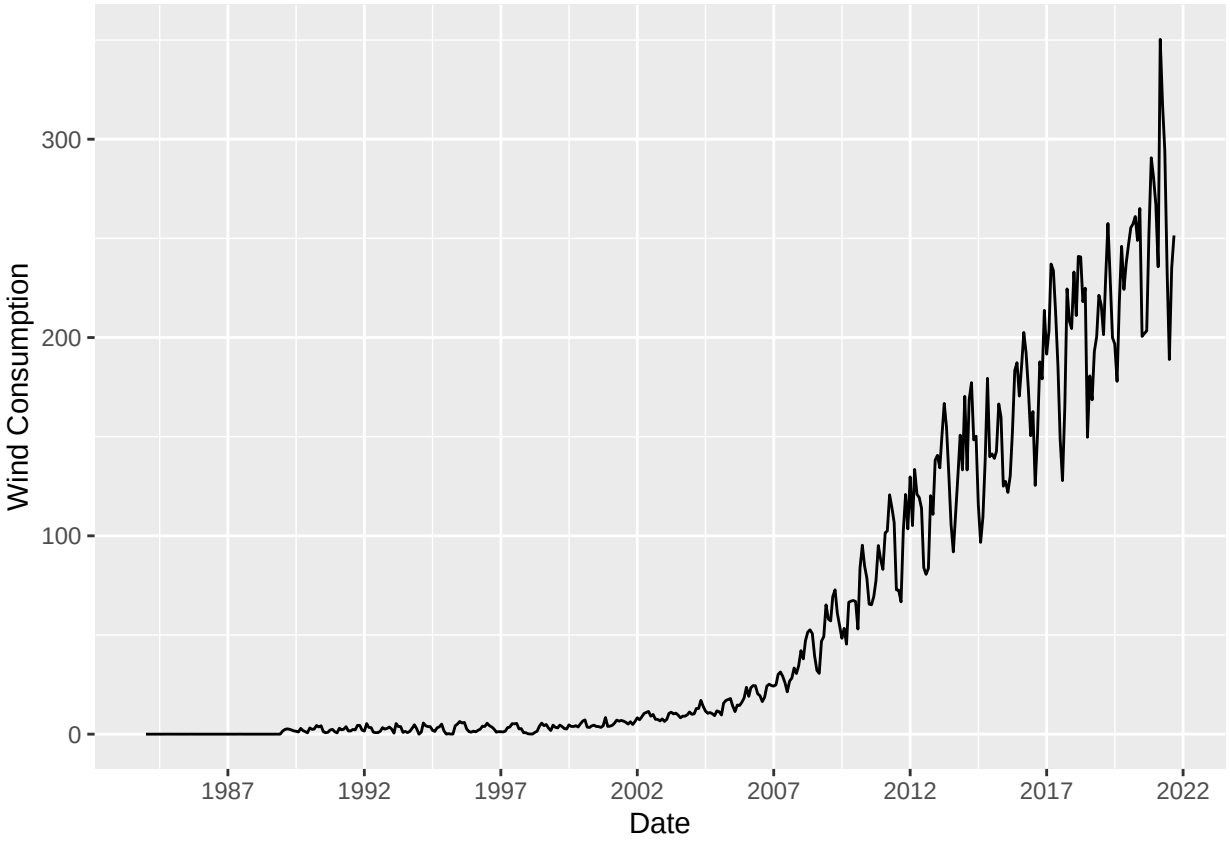
Q2

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(as.data.frame(energy_data), aes(x=Date)) +
  geom_line(aes(y=Solar)) +
  ylab(label="Solar Consumption") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```



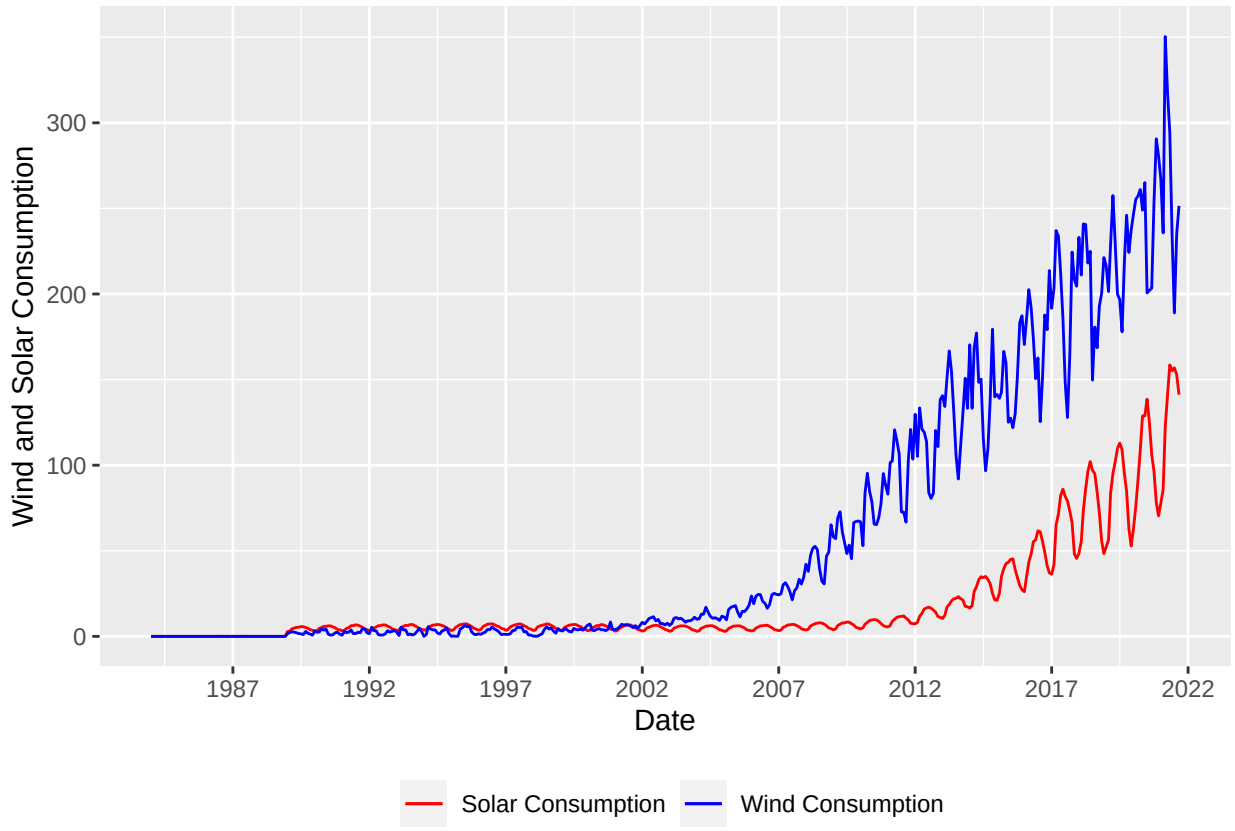

```
ggplot(as.data.frame(energy_data), aes(x=Date)) +
  geom_line(aes(y=Wind)) +
  ylab(label="Wind Consumption") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```



Q3

Now plot both series in the same graph, also using ggplot(). Look at lines 142-149 of the file `05_Lab_OutliersMissingData_Solution` to learn how 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() again to improve x axis.

```
ggplot(as.data.frame(energy_data), aes(x=Date)) +
  geom_line(aes(y=Solar,col="Solar")) +
  geom_line(aes(y=Wind,col="Wind")) +
  labs(color="") +
  scale_color_manual(values = c("Solar" = "red", "Wind" = "blue"),
                                labels=c("Solar Consumption", "Wind Consumption")) +
  theme(legend.position = "bottom") +
  ylab(label="Wind and Solar Consumption") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```

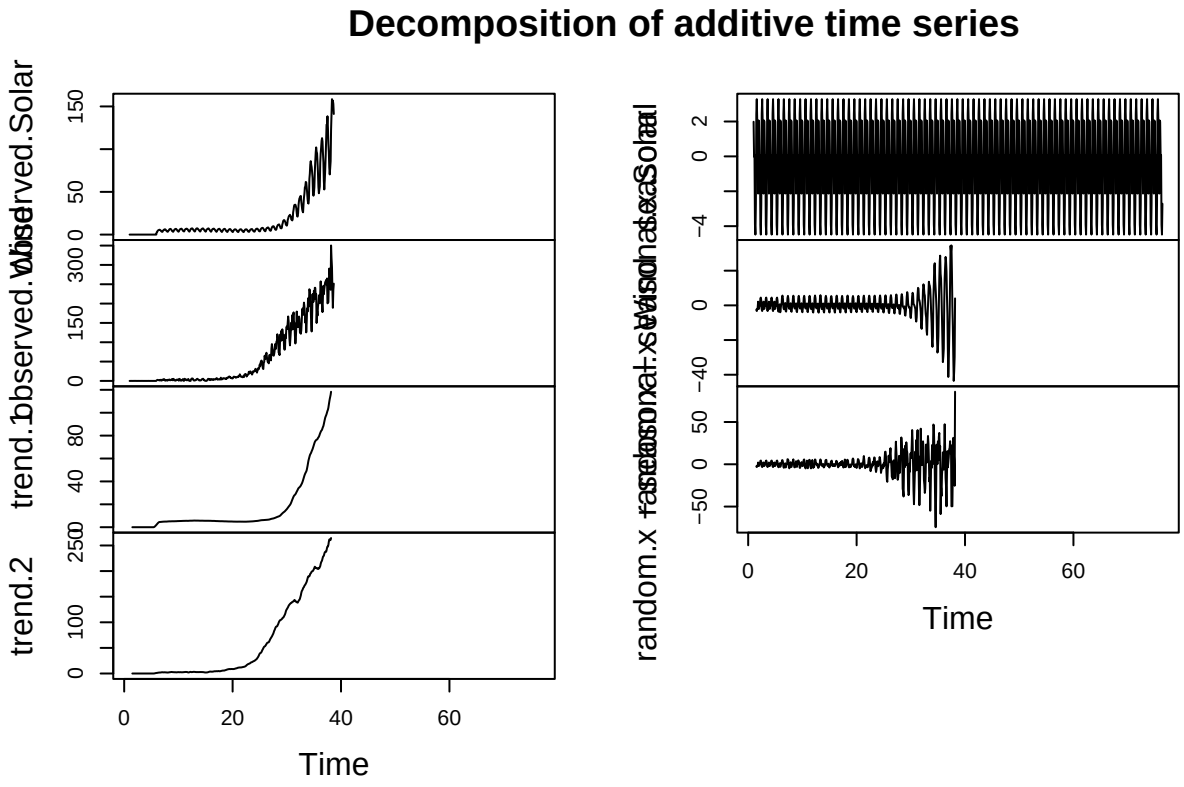


Q3

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?

The trend component seems to have an upward trend starting around year 20. For the random component it seems to have some randomness until the same point that the trend component started seeing an upward trend. The random component starts to grow at this point and appears to have some seasonality to it.

```
decompose_energy_data=decompose(ts_energy_data,"additive")
plot(decompose_energy_data)
```

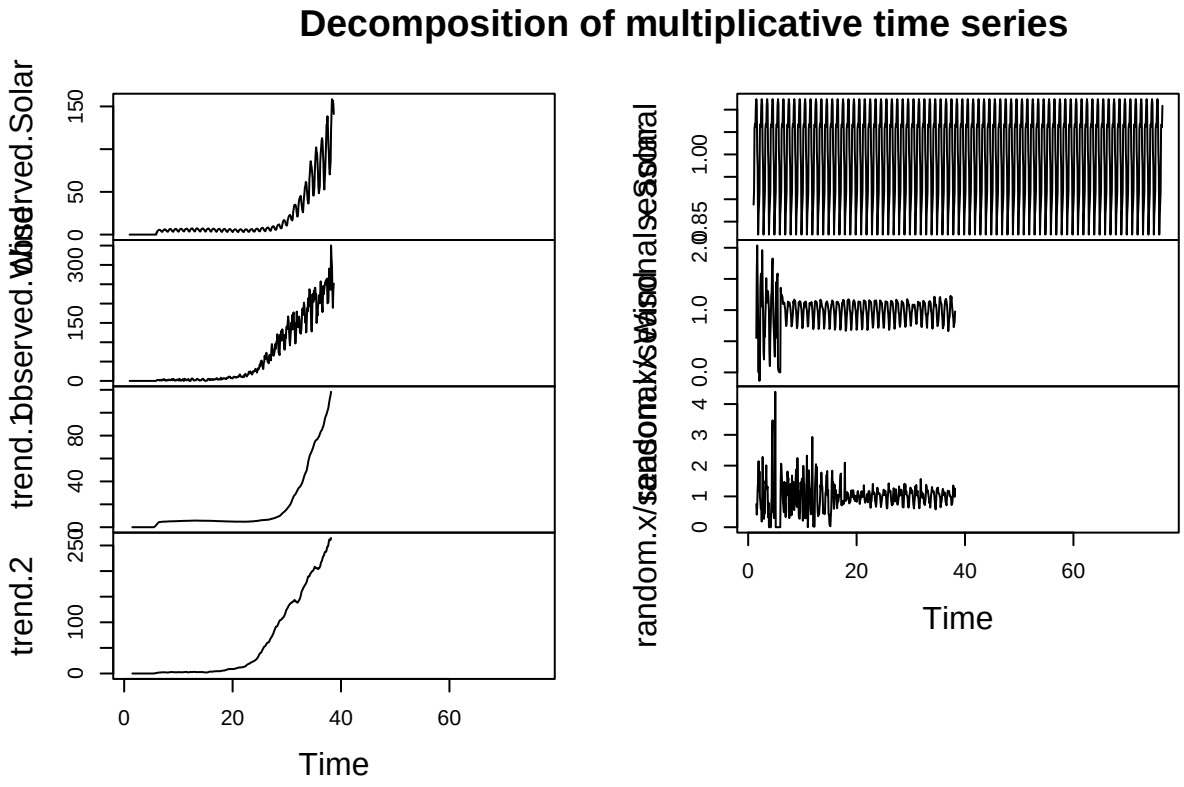



```
energy_random <- decompose_energy_data$random
mean_inflow <- mean(energy_random)
sd_inflow <- sd(energy_random)
```

Q4

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? Now that the seasonal component is multiplicative, the random component now seems to have less of a seasonal trend at the some point that the trend component has an upward trend, thought there is more of a variance in the random component when the trend component has not yet experienced an upward trend.

```
decompose_energy_data=decompose(ts_energy_data,"multiplicative")
plot(decompose_energy_data)
```




```
energy_random <- decompose_energy_data$random
mean_inflow <- mean(energy_random)
sd_inflow <- sd(energy_random)
```

Q5

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 would say no, we do not need that historical data. This is because there was not much solar and wind development during that time so it is not relevant to the trend we are seeing. If we are trying to forecast the next six months, we should start when the trend component starts to have an upward trend as this is when solar and wind development takes off.

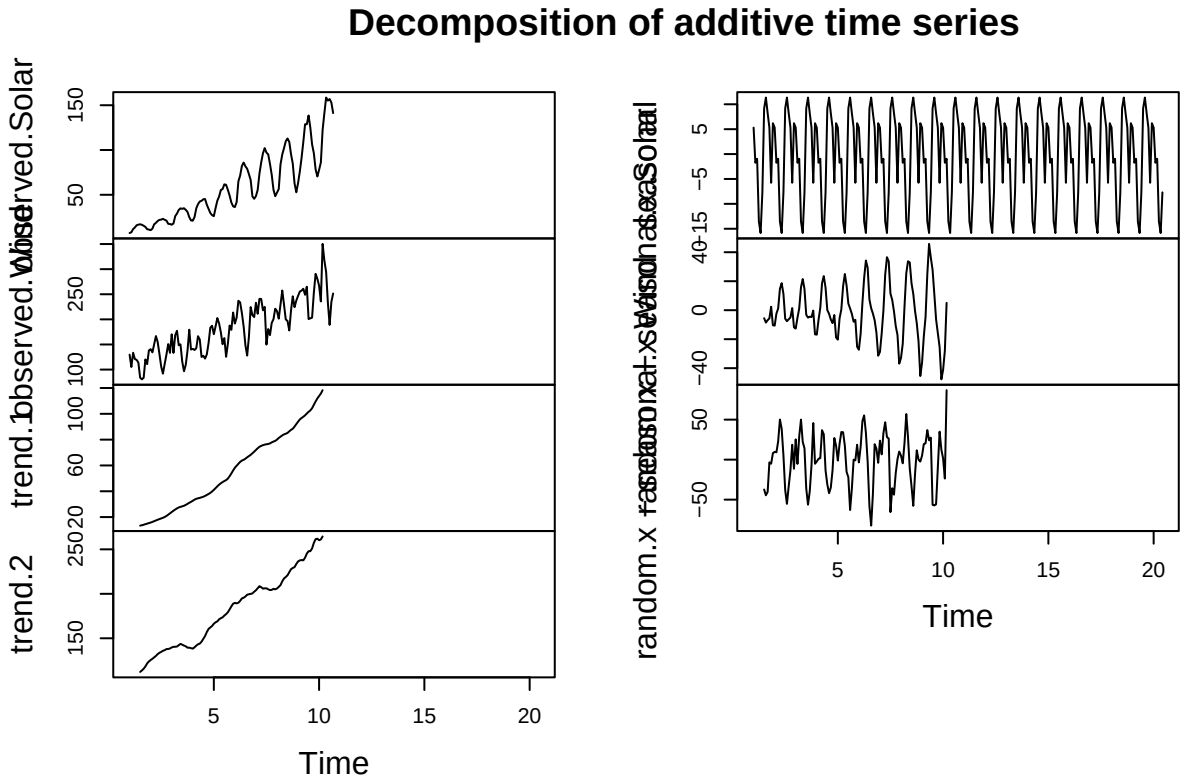
Q6

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 trying to remove the seasonal component and the challenge of trend on the seasonal component.

By removing the years prior to 2012, it is now more obvious that there is a seasonal component to both the trend and random component. This is most likely since wind patterns and amount of sunlight both vary

depending on the time of year. It would make more sense to remove the seasonal component first and then look at the random component.

```
energy_data <- filter(energy_data, year(Date) >= 2012)
ts_energy_data <- ts(energy_data[,2:3],frequency=12)
decompose_energy_data=decompose(ts_energy_data,"additive")
plot(decompose_energy_data)
```




```
energy_random <- decompose_energy_data$random
mean_inflow <- mean(energy_random)
sd_inflow <- sd(energy_random)
```

Answer: