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

Assignment 5 - Due date 02/28/22

Ben Joseph

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(xlsx)
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
```

```
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 dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
## v purrr   0.3.4

## -- 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
df <- read_xlsx("./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx", col_nam
energy_data <-df[2:586,]

colnames(energy_data) <- colnames(df)

nobs=nrow(energy_data)
nvar=ncol(energy_data)

head(energy_data)
```

```
## # A tibble: 6 x 14
##   Month               `Wood Energy Production` `Biofuels Produ~ `Total Biomass ~
##   <dttm>              <chr>                    <chr>            <chr>
## 1 1973-01-01 00:00:00 129.63                   <NA>             129.787
## 2 1973-02-01 00:00:00 117.194                  <NA>             117.338
## 3 1973-03-01 00:00:00 129.763                  <NA>             129.938
## 4 1973-04-01 00:00:00 125.462                  <NA>             125.636
## 5 1973-05-01 00:00:00 129.624                  <NA>             129.834
## 6 1973-06-01 00:00:00 125.435                  <NA>             125.611
## # ... with 10 more variables: 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>,
## #   Total Renewable 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!

```
head(energy_data,26)
```

```
## # A tibble: 26 x 14
##    Month               `Wood Energy Prod~ `Biofuels Product~ `Total Biomass Ene~
##    <dttm>              <chr>              <chr>              <chr>
##  1 1973-01-01 00:00:00 129.63             <NA>               129.787
##  2 1973-02-01 00:00:00 117.194            <NA>               117.338
##  3 1973-03-01 00:00:00 129.763            <NA>               129.938
##  4 1973-04-01 00:00:00 125.462            <NA>               125.636
##  5 1973-05-01 00:00:00 129.624            <NA>               129.834
##  6 1973-06-01 00:00:00 125.435            <NA>               125.611
##  7 1973-07-01 00:00:00 129.616            <NA>               129.787
##  8 1973-08-01 00:00:00 129.734            <NA>               129.918
##  9 1973-09-01 00:00:00 125.603            <NA>               125.782
## 10 1973-10-01 00:00:00 129.769            <NA>               129.97
## # ... with 16 more rows, and 10 more variables:
## #   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>,
## #   Total Renewable Energy Consumption <chr>
```

```
energy_data_processed <- energy_data[,c("Month","Solar Energy Consumption","Wind Energy Consumption")]
energy_data_processed$`Solar Energy Consumption` <- as.numeric(energy_data_processed$`Solar Energy Consu
energy_data_processed$`Wind Energy Consumption` <- as.numeric(energy_data_processed$`Wind Energy Consump
energy_data_processed<- drop_na(energy_data_processed)
energy_data_processed$Month<-as.Date(energy_data_processed$Month)

head(energy_data_processed)
```

```
## # A tibble: 6 x 3
##   Month      `Solar Energy Consumption` `Wind Energy Consumption`
##   <date>                          <dbl>                     <dbl>
## 1 1984-01-01                     -0.001                     0
## 2 1984-02-01                      0.001                     0.002
## 3 1984-03-01                      0.002                     0.002
## 4 1984-04-01                      0.003                     0.006
## 5 1984-05-01                      0.007                     0.008
## 6 1984-06-01                      0.01                      0.006
```

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")")*

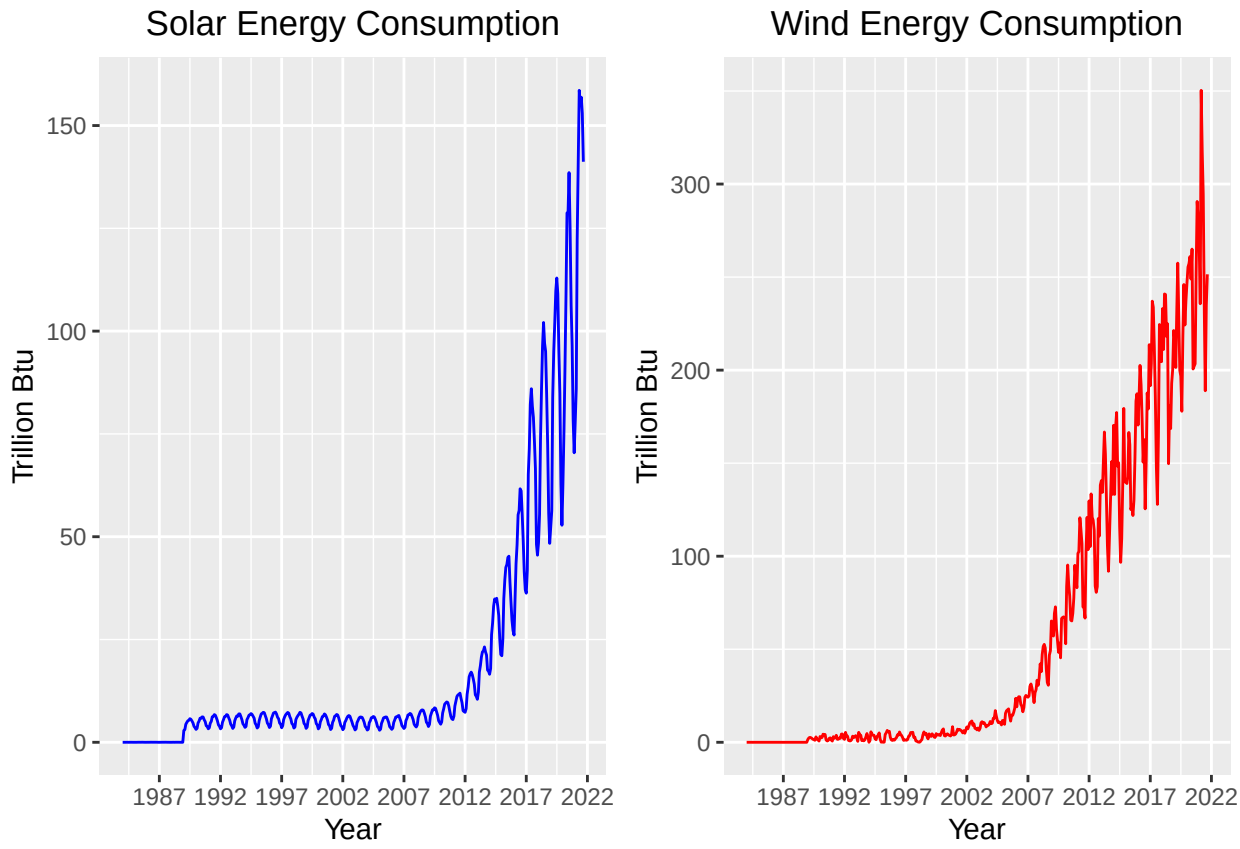
```
require(gridExtra)
```

```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

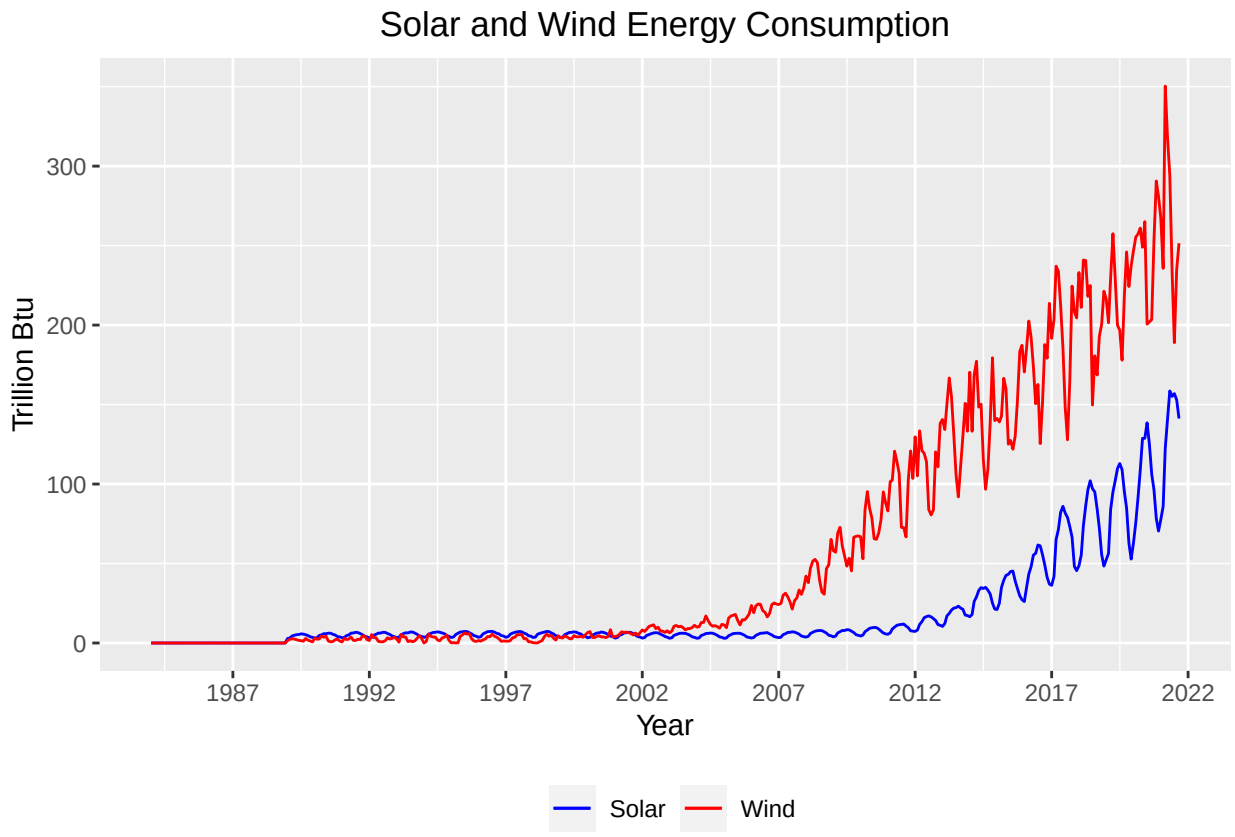
```
plot1 <- ggplot(energy_data_processed, aes(x=energy_data_processed$Month, y=energy_data_processed$`Sola
            geom_line(color="blue") +
            ggtitle("Solar Energy Consumption")+
            theme(plot.title = element_text(hjust = 0.5))+  xlab("Year") + ylab("Trillion Btu")+
            scale_x_date(date_breaks = "5 years", date_labels = "%Y")

plot2 <- ggplot(energy_data_processed, aes(x=energy_data_processed$Month, y=energy_data_processed$`Wind
            geom_line(color="red") +
            ggtitle("Wind Energy Consumption")+
            theme(plot.title = element_text(hjust = 0.5))+  xlab("Year") + ylab("Trillion Btu")+
            scale_x_date(date_breaks = "5 years", date_labels = "%Y")

grid.arrange(plot1, plot2, ncol=2)
```



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(data.frame(date = energy_data_processed$Month, solar = energy_data_processed$`Solar Energy Consu
  geom_line(aes(x= date, y = solar, color = "Solar")) +
  geom_line(aes(x= date, y = wind, color = "Wind")) +
  labs(color="") +
  scale_color_manual(values = c("Solar" = "blue", "Wind" = "red"), labels=c("Solar", "Wind")) +
  theme(legend.position = "bottom") +
  ggtitle("Solar and Wind Energy Consumption") +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("Year") +
  ylab("Trillion Btu")+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```

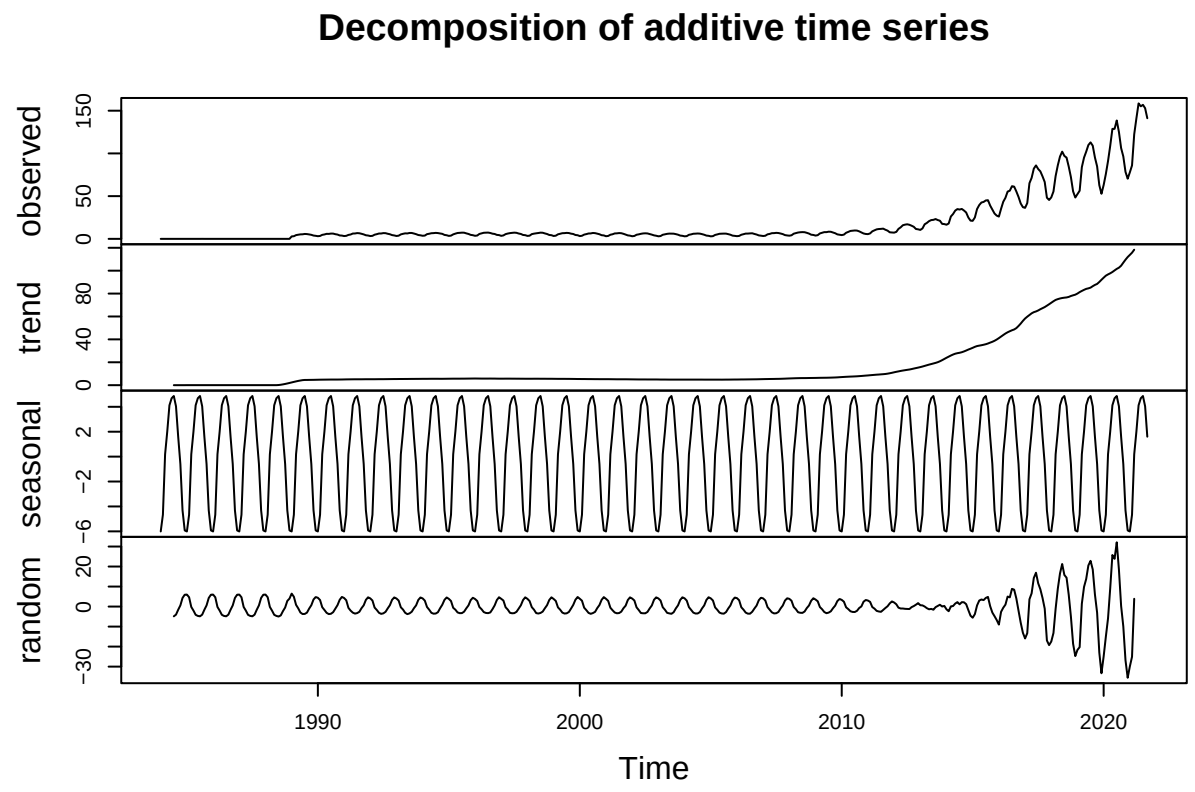


Q3

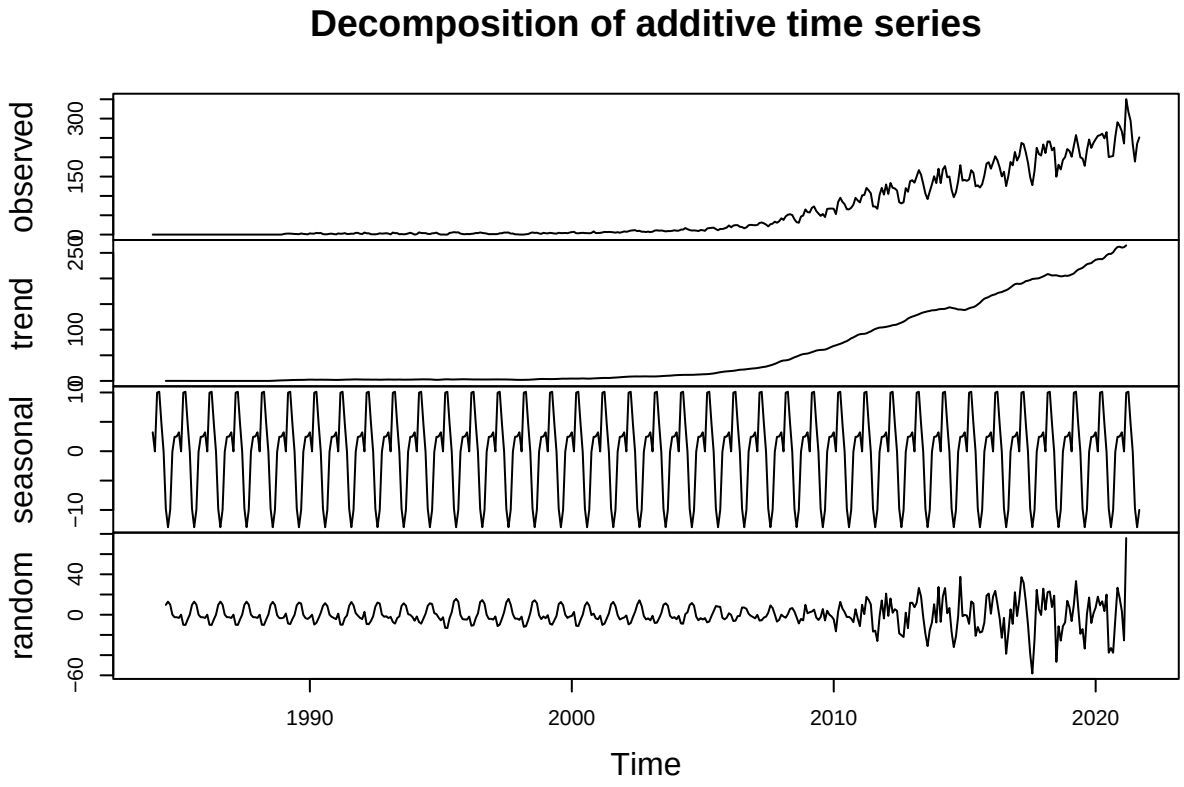
Transform wind and solar series into a time series object and apply the decompose function on them using the additive option, i.e., `decompose(ts_data, type = "additive")`. What can you say about the trend component? What about the random component? Does the random component look random? Or does it appear to still have some seasonality on it?

Answer: The trend component has a clear upward trend which is obvious at first glance of the observed series. The random component does not look random. It still has some seasonality on it. I would imagine that this is because the number of wind and solar generators grow over time such that the aggregated data shows a growing seasonal trend that can't be captured using a static seasonal trend like is used in the decompose function.

```
ts_solar <- ts(data = energy_data_processed$`Solar Energy Consumption`, start=1984, frequency = 12)
ts_wind <- ts(data = energy_data_processed$`Wind Energy Consumption`, start=1984, frequency = 12)

solar_decomp <- decompose(ts_solar,type = "additive")
plot(solar_decomp)
```




```
wind_decomp <- decompose(ts_wind,type = "additive")
plot(wind_decomp)
```

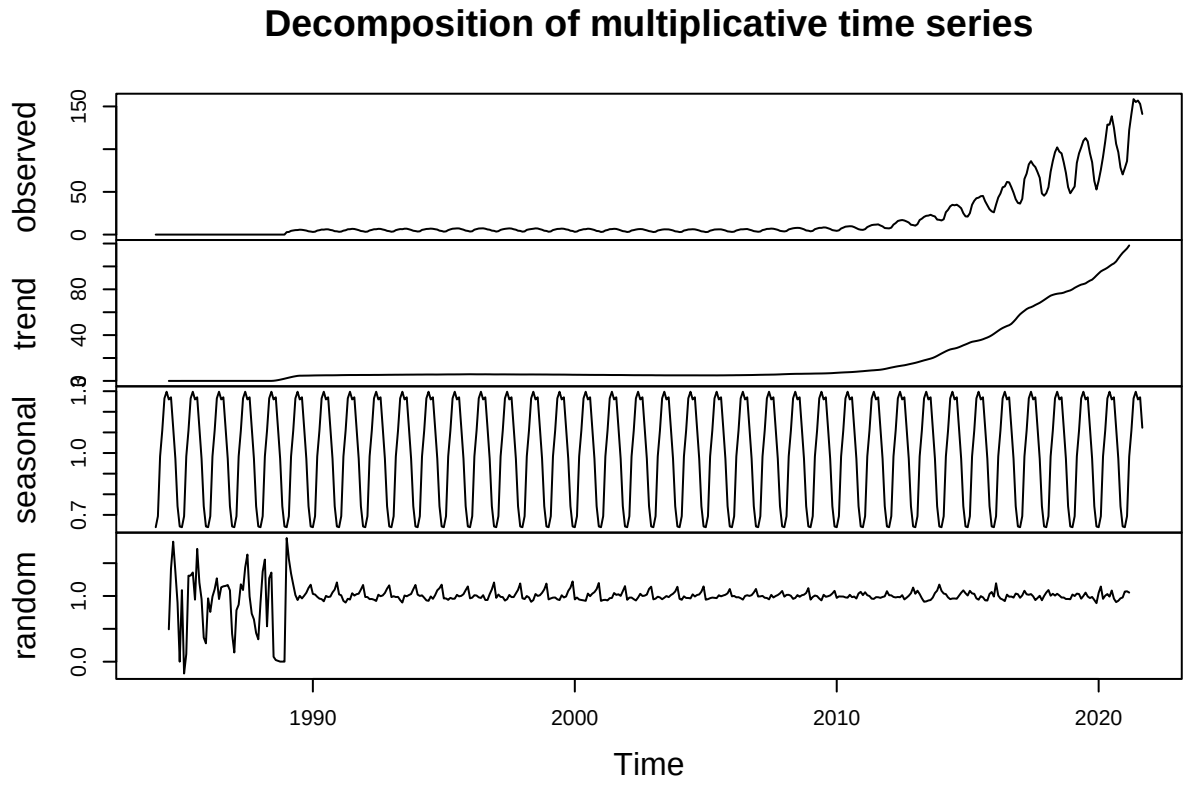


Q4

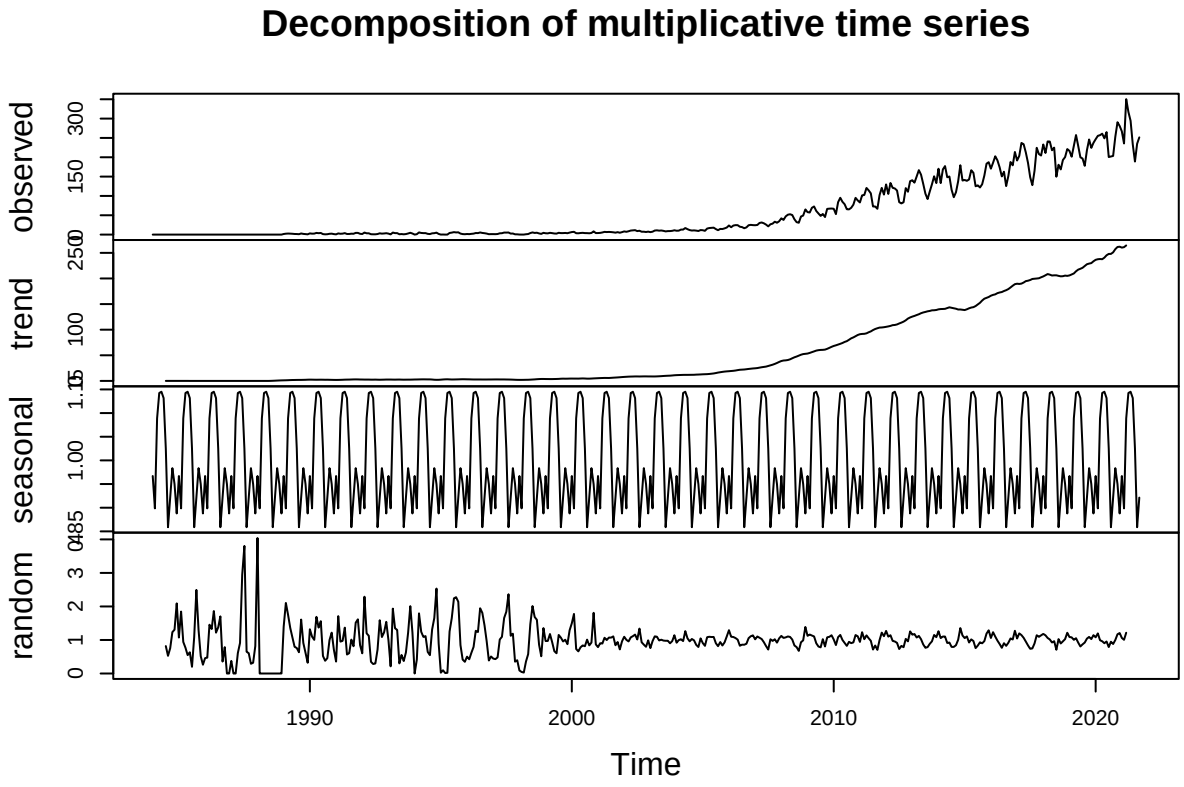
Use the decompose function again but now change the type of the seasonal component from additive to multiplicative. What happened to the random component this time?

> Answer: This time, the random component deviates much less as the values are much closer to zero, and most of the seasonality has been removed.

```
solar_decomp_mult <- decompose(ts_solar,type = "multiplicative")
plot(solar_decomp_mult)
```




```
wind_decomp_mult <- decompose(ts_wind,type = "multiplicative")
plot(wind_decomp_mult)
```



Q5

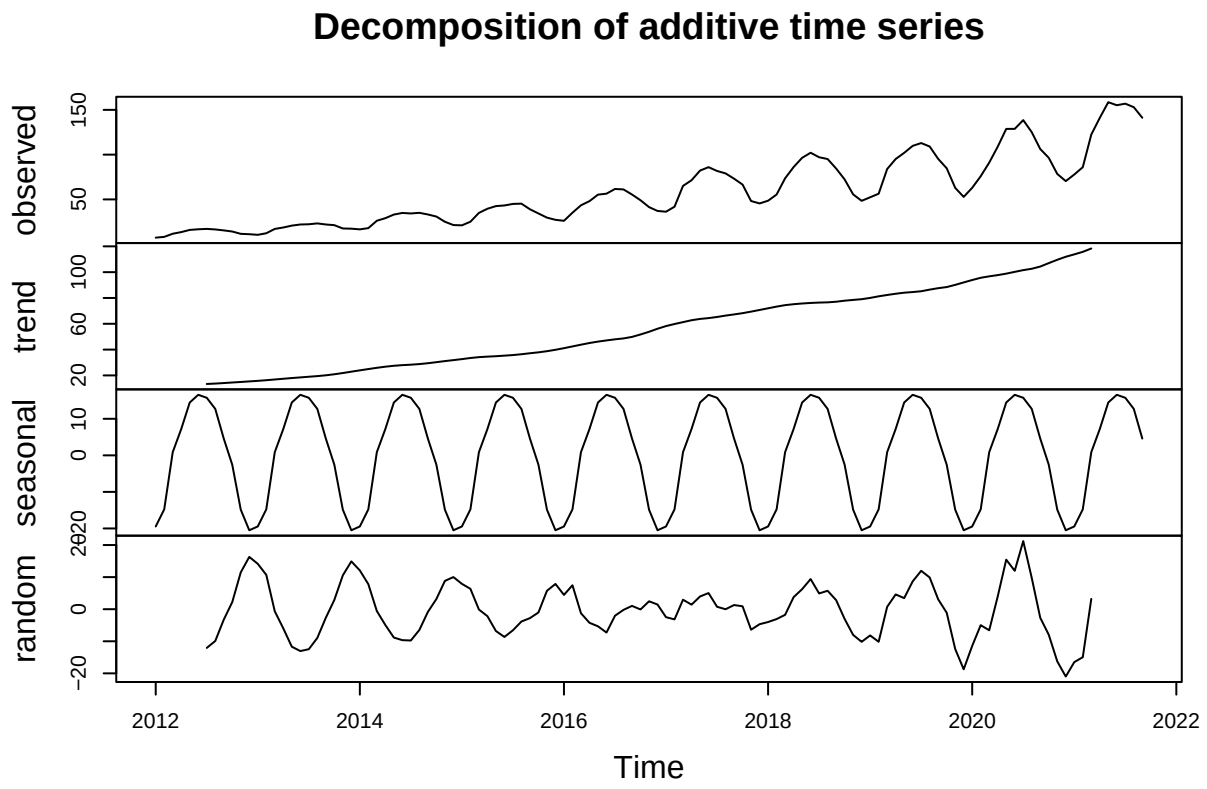
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: The amount of noise in the random series in the 90s and early 00s indicates that the seasonal trend identified by the decomosition model is a bad fit for the data. This is because the solar and wind data was so negligible that either it could count as outliers or its seasonal trend was too small to even be recognized by the decomposition function. The only issue is you should be careful of removing too much data because it may make the upward trend in the 2010s and 2020s look linear when perhaps it should look more exponential. The fact that the random series in the 2010s and 2020s looks very random and does not deviate very far from 0 indicates that the seasonal trend in the decompositioin function does a good job at modeling the seasonal component. The success of the seasonal trend is unlikely to have anything to do with the observed data from the 1990s and 2000s so I would think that we need any information from the early years to forecast the next six months of solar and wind consumption.

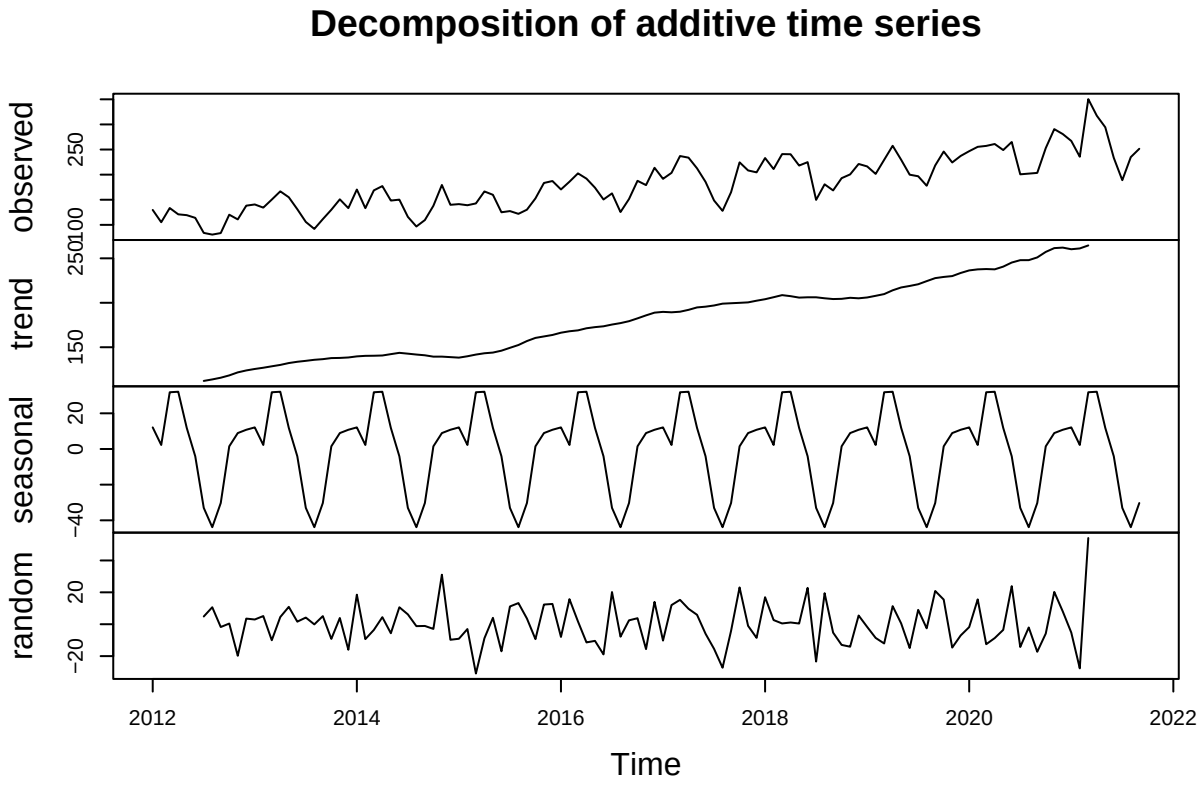
Q6

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.

```r
energy_data_filtered <- filter(energy_data_processed, year(energy_data_processed$Month) >= 2012)

ts_solar_filtered <- ts(data = energy_data_filtered$`Solar Energy Consumption`, start=2012, frequency =
ts_wind_filtered <- ts(data = energy_data_filtered$`Wind Energy Consumption`, start=2012, frequency = 1

solar_decomp_filtered <- decompose(ts_solar_filtered,type = "additive")
plot(solar_decomp_filtered)
```




```r
wind_decomp_filtered <- decompose(ts_wind_filtered,type = "additive")
plot(wind_decomp_filtered)
```



Answer: The wind data's random series appears fairly random. It does, however, seem to vary farther from zero as the series progresses. Linking this back to class, this reflects the conversation we had that a challenge to estimating seasonal trend might be a growing lack of seasonal predictability.

The solar data's random series does not. This seems to be because the seasonal variation in the observed solar data grows significantly through the time series. This is the same issue we saw in the unfiltered series. With less data, we can see that the seasonal pattern in the random data flips in the middle of the time series showing that the static seasonal trend estimation under the additive decomposition type overestimated the seasonal variation when the observed seasonal variation was less than the time series' average seasonal variation and underestimated it when the the observed variation was greater than average. Relating this back to class, this illustrates why we have to be careful when modeling a seasonal component because it might grow over time. This can be overcome using more complex regression and modeling techniques like SARIMA models!