ENV 790.30 - Time Series Analysis for Energy Data | Spring 2023

Assignment 5 - Due date 02/27/23

Tony Jiang

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: "xlsx" or "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(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 1.3.2 --
```

```
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.3.0      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## v purrr   1.0.1
```

```
## -- 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()
```

```
library(Kendall) #Use MannKendall test to test series' stationarity
```

Decomposing Time Series

Consider the same data you used for A04 from the spreadsheet "Table_10.1_Renewable_Energy_Production_and_Consumptio

```
#Importing data set - using xlsx package

renewables <- read_xlsx("./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx",
                        skip = 10)

read_col_names <- read_xlsx(path = "./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Sou
                            col_names = FALSE, skip = 10, n_max = 1)
```

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

```
colnames(renewables) <- read_col_names

renewables <- renewables[-1,] #remove notes

# change month to date format
renewables$Month = ymd(renewables$Month)

head(renewables)
```

```
## # A tibble: 6 x 14
##   Month      Wood Ener~1 Biofu~2 Total~3 Total~4 Hydro~5 Geoth~6 Solar~7 Wind ~8
##   <date>     <chr>       <chr>   <chr>   <chr>   <chr>   <chr>   <chr>   <chr>
## 1 1973-01-01 129.63      Not Av~ 129.787 403.981 272.703 1.491   Not Av~ Not Av~
## 2 1973-02-01 117.194     Not Av~ 117.338 360.9   242.199 1.363   Not Av~ Not Av~
## 3 1973-03-01 129.763     Not Av~ 129.938 400.161 268.81  1.412   Not Av~ Not Av~
## 4 1973-04-01 125.462     Not Av~ 125.636 380.47  253.185 1.649   Not Av~ Not Av~
## 5 1973-05-01 129.624     Not Av~ 129.834 392.141 260.77  1.537   Not Av~ Not Av~
```

```
## 6 1973-06-01 125.435     Not Av~ 125.611 377.232 249.859 1.763   Not Av~ Not Av~
## # ... with 5 more variables: `Wood Energy Consumption` <chr>,
## #   `Waste Energy Consumption` <chr>, `Biofuels Consumption` <chr>,
## #   `Total Biomass Energy Consumption` <chr>,
## #   `Total Renewable Energy Consumption` <chr>, and abbreviated variable names
## #   1: `Wood Energy Production`, 2: `Biofuels Production`,
## #   3: `Total Biomass Energy Production`,
## #   4: `Total Renewable Energy Production`, ...
```

```
str(renewables)
```

```
## tibble [597 x 14] (S3: tbl_df/tbl/data.frame)
##  $ Month                              : Date[1:597], format: "1973-01-01" "1973-02-01" ...
##  $ Wood Energy Production             : chr [1:597] "129.63" "117.194" "129.763" "125.462" ...
##  $ Biofuels Production                : chr [1:597] "Not Available" "Not Available" "Not Available" "
##  $ Total Biomass Energy Production    : chr [1:597] "129.787" "117.338" "129.938" "125.636" ...
##  $ Total Renewable Energy Production : chr [1:597] "403.981" "360.9" "400.161" "380.47" ...
##  $ Hydroelectric Power Consumption    : chr [1:597] "272.703" "242.199" "268.81" "253.185" ...
##  $ Geothermal Energy Consumption      : chr [1:597] "1.491" "1.363" "1.412" "1.649" ...
##  $ Solar Energy Consumption           : chr [1:597] "Not Available" "Not Available" "Not Available" "
##  $ Wind Energy Consumption            : chr [1:597] "Not Available" "Not Available" "Not Available" "
##  $ Wood Energy Consumption            : chr [1:597] "129.63" "117.194" "129.763" "125.462" ...
##  $ Waste Energy Consumption           : chr [1:597] "0.157" "0.144" "0.176" "0.174" ...
##  $ Biofuels Consumption               : chr [1:597] "Not Available" "Not Available" "Not Available" "
##  $ Total Biomass Energy Consumption   : chr [1:597] "129.787" "117.338" "129.938" "125.636" ...
##  $ Total Renewable Energy Consumption: chr [1:597] "403.981" "360.9" "400.161" "380.47" ...
```

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!

```
#convert chr to numeric, subset the dataset, and create time series

ts_renewable_sub5 <- subset(renewables, select = c("Solar Energy Consumption",
                                    "Wind Energy Consumption")) %>%
  sapply(as.numeric) %>%
  ts(start = c(1973, 1), frequency = 12) %>%
  na.omit()
```

```
## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion
```

```
str(ts_renewable_sub5)
```

```
##  Time-Series [1:465, 1:2] from 1984 to 2023: -0.001 0.001 0.002 0.003 0.007 0.01 0.003 0.009 0.01 0.
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "Solar Energy Consumption" "Wind Energy Consumption"
##  - attr(*, "na.action")= 'omit' int [1:132] 1 2 3 4 5 6 7 8 9 10 ...
```

```
# first: convert the data frame to a time series analysis, then omit all missing
# values, so obs will still have correct time indice.
```

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")"*

```
# have a non-ts data frame to use in ggplot because ggplot can't plot
# time series data. I convert this to a dataframe so it can be easily deployed
renewable_sub5 <- renewables %>%
  subset(select = c("Solar Energy Consumption",
                    "Wind Energy Consumption")) %>%
  sapply(as.numeric) %>%
  cbind(as.Date(renewables$Month)) %>%
  na.omit() %>%
  as.data.frame()
```

```
## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion
```

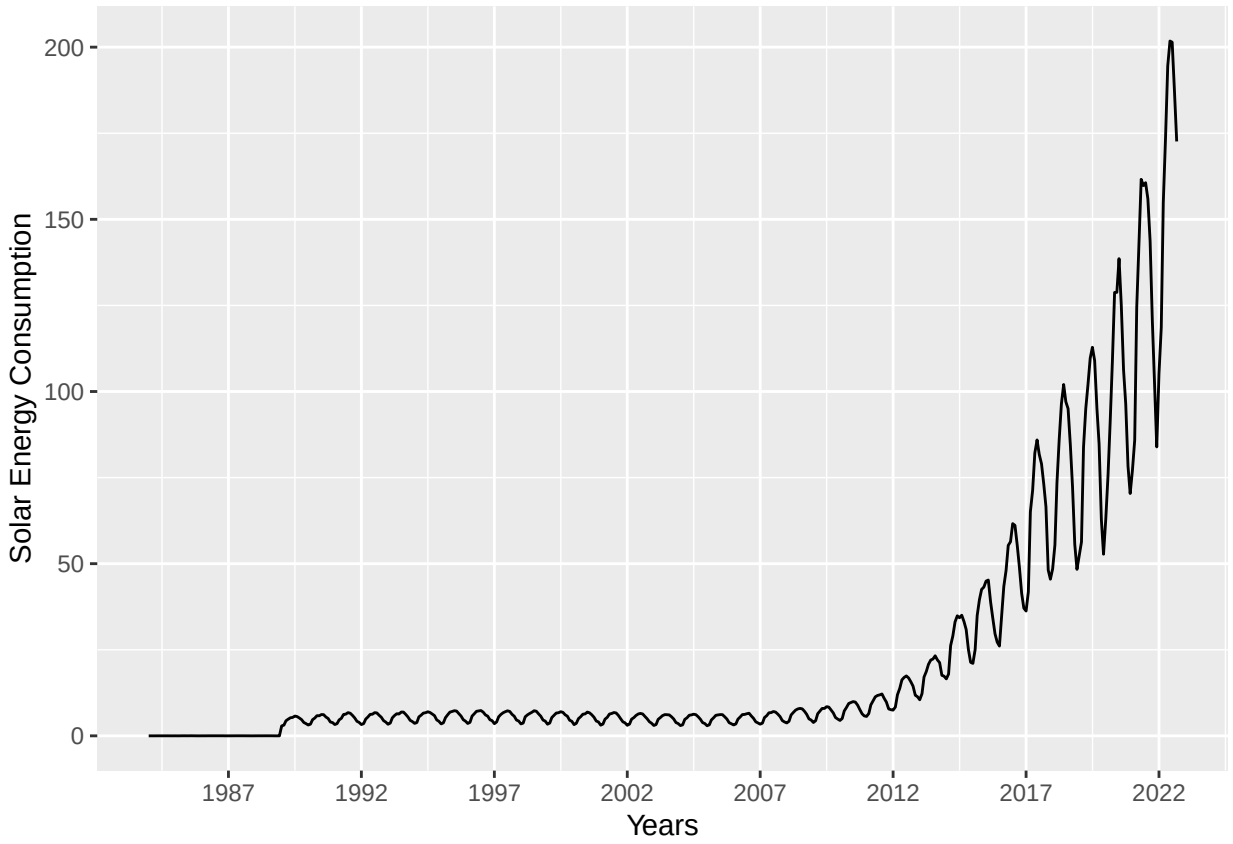
```
# rename columns
colnames(renewable_sub5) <- c("Solar Energy Consumption",
                              "Wind Energy Consumption",
                              "Month"
                              )

# convert numeric numbers back to data format. As.numeric() covert date to
# numbers. To make ggplot have the right x-axis text, I need to convert it back.
renewable_sub5[,3] <- as.Date(renewable_sub5[,3], origin = "1970-01-01")

str(renewable_sub5)
```

```
## 'data.frame':    465 obs. of  3 variables:
##  $ Solar Energy Consumption: num  -0.001 0.001 0.002 0.003 0.007 0.01 0.003 0.009 0.01 0.007 ...
##  $ Wind Energy Consumption : num  0 0.002 0.002 0.006 0.008 0.006 0.005 0.003 0.005 0.009 ...
##  $ Month                   : Date, format: "1984-01-01" "1984-02-01" ...
```

```
#ggplot
ggplot(data = renewable_sub5,
       aes(x = renewable_sub5$Month)
       ) +
  geom_line(aes(y = renewable_sub5$`Solar Energy Consumption`)) +
  scale_x_date(date_breaks = "5 years",
               date_labels = "%Y") +
  xlab("Years") +
  ylab("Solar Energy Consumption")
```

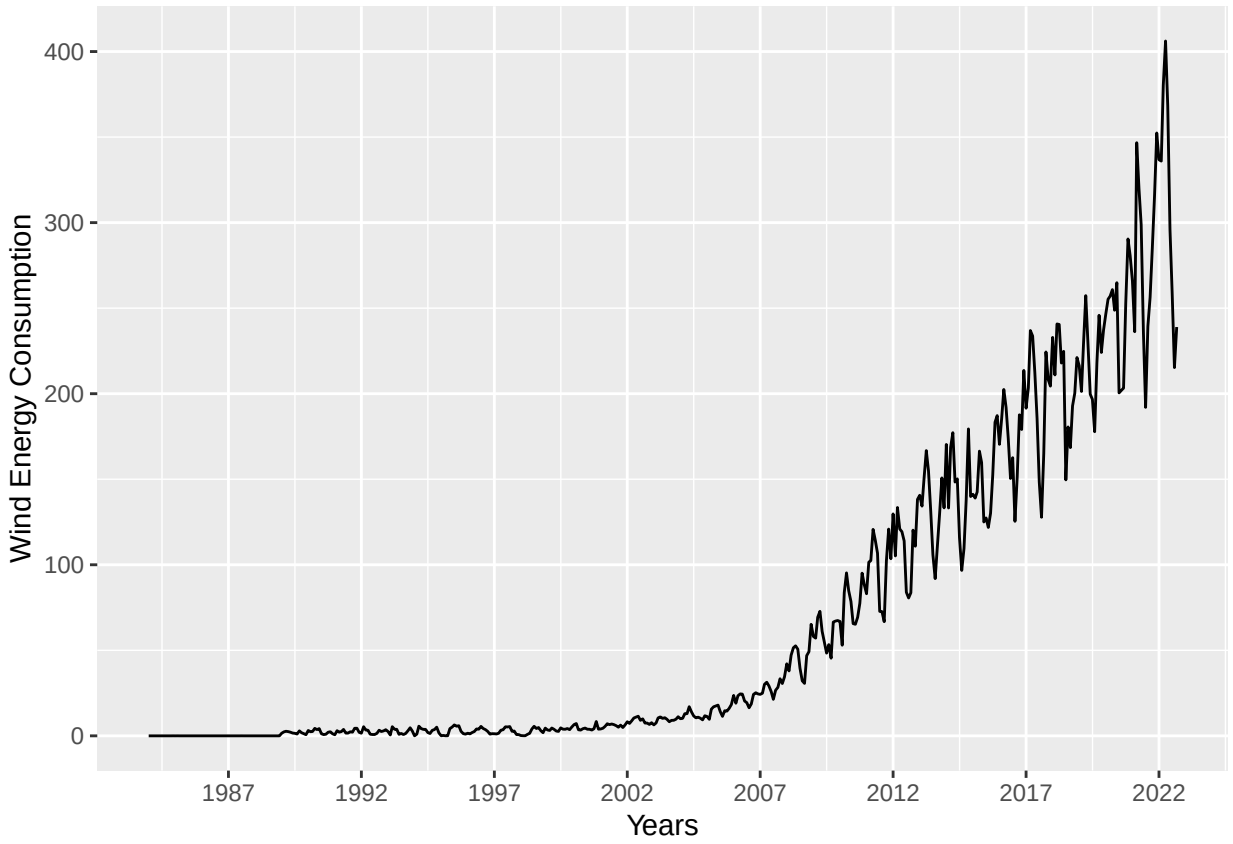
```
## Warning: Use of `` renewable_sub5$`Solar Energy Consumption` `` is discouraged.
## i Use `Solar Energy Consumption` instead.
```

```
## Warning: Use of `renewable_sub5$Month` is discouraged.
## i Use `Month` instead.
```




```
ggplot(data = renewable_sub5,
       aes(x = renewable_sub5$Month)
       ) +
  geom_line(aes(y = renewable_sub5$`Wind Energy Consumption`)) +
  scale_x_date(date_breaks = "5 years",
               date_labels = "%Y") +
  xlab("Years") +
  ylab("Wind Energy Consumption")
```

```
## Warning: Use of `` renewable_sub5$`Wind Energy Consumption` `` is discouraged.
## i Use `Wind Energy Consumption` instead.
## Use of `renewable_sub5$Month` is discouraged.
## i Use `Month` instead.
```



Q3

Now plot both series in the same graph, also using ggplot(). Look at lines 141-148 of the file `M4_OutliersMissingData_Part2_Complete.Rmd` 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. I reconfigured grid type and the position of the legend
ggplot(data = renewable_sub5,
       aes(x = renewable_sub5$Month)
       ) +
  geom_line(aes(y = renewable_sub5$`Wind Energy Consumption`,
                color = "Wind Energy Consumption")) +
  geom_line(aes(y = renewable_sub5$`Solar Energy Consumption`,
                color = "Solar Energy Consumption")) +
  scale_color_manual(values = c("Wind Energy Consumption" = "blue",
                                "Solar Energy Consumption" = "red"),
                     labels = c("Solar Energy Consumption",
                                "Wind Energy Consumption"
                                )
                     ) +
  theme_bw() +
  theme(legend.position = c(0.2, 0.88),
        legend.title = element_blank()) +
  xlab("Years") +
  ylab("Energy Consumption") +
  scale_x_date(date_breaks = "5 years",
```

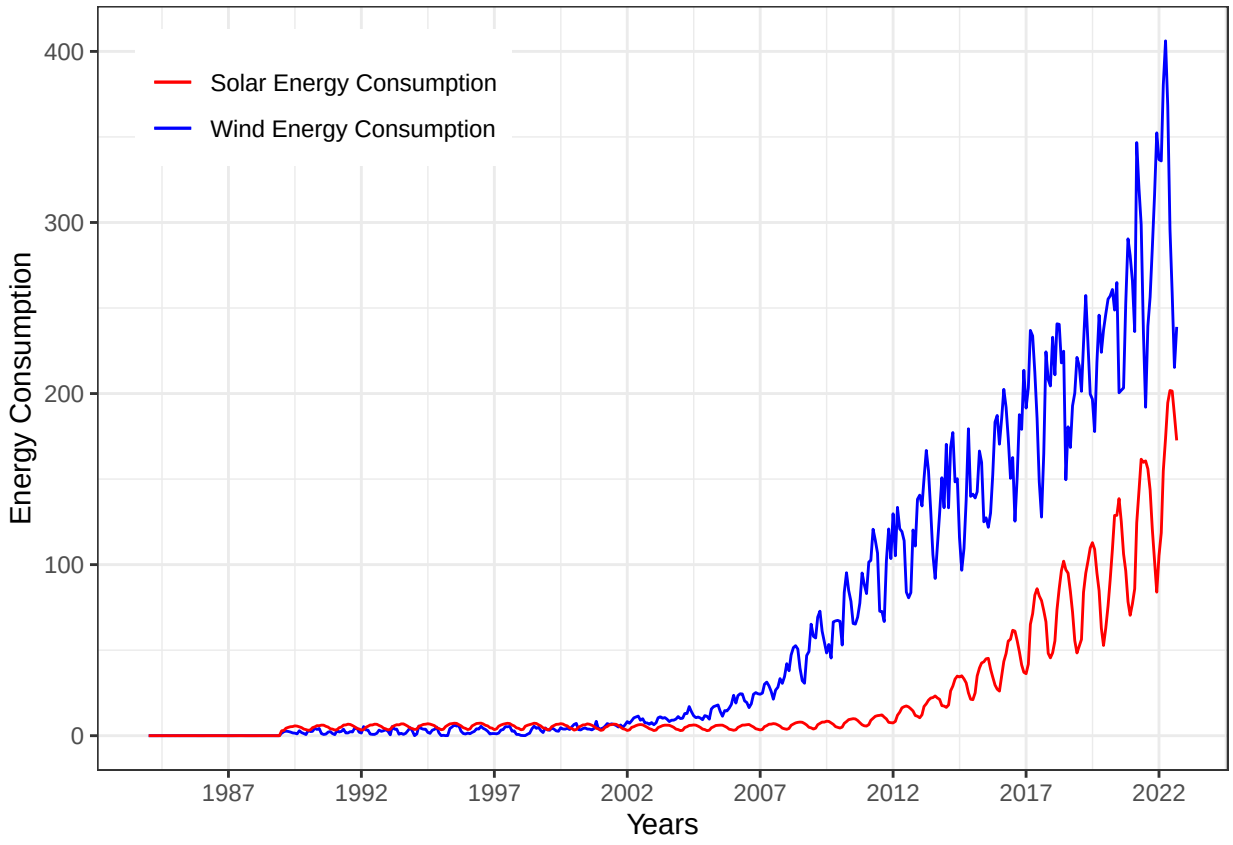
```
                date_labels = "%Y")
```

```
## Warning: Use of `` renewable_sub5$`Wind Energy Consumption` `` is discouraged.
## i Use `Wind Energy Consumption` instead.
```

```
## Warning: Use of `renewable_sub5$Month` is discouraged.
## i Use `Month` instead.
```

```
## Warning: Use of `` renewable_sub5$`Solar Energy Consumption` `` is discouraged.
## i Use `Solar Energy Consumption` instead.
```

```
## Warning: Use of `renewable_sub5$Month` is discouraged.
## i Use `Month` instead.
```



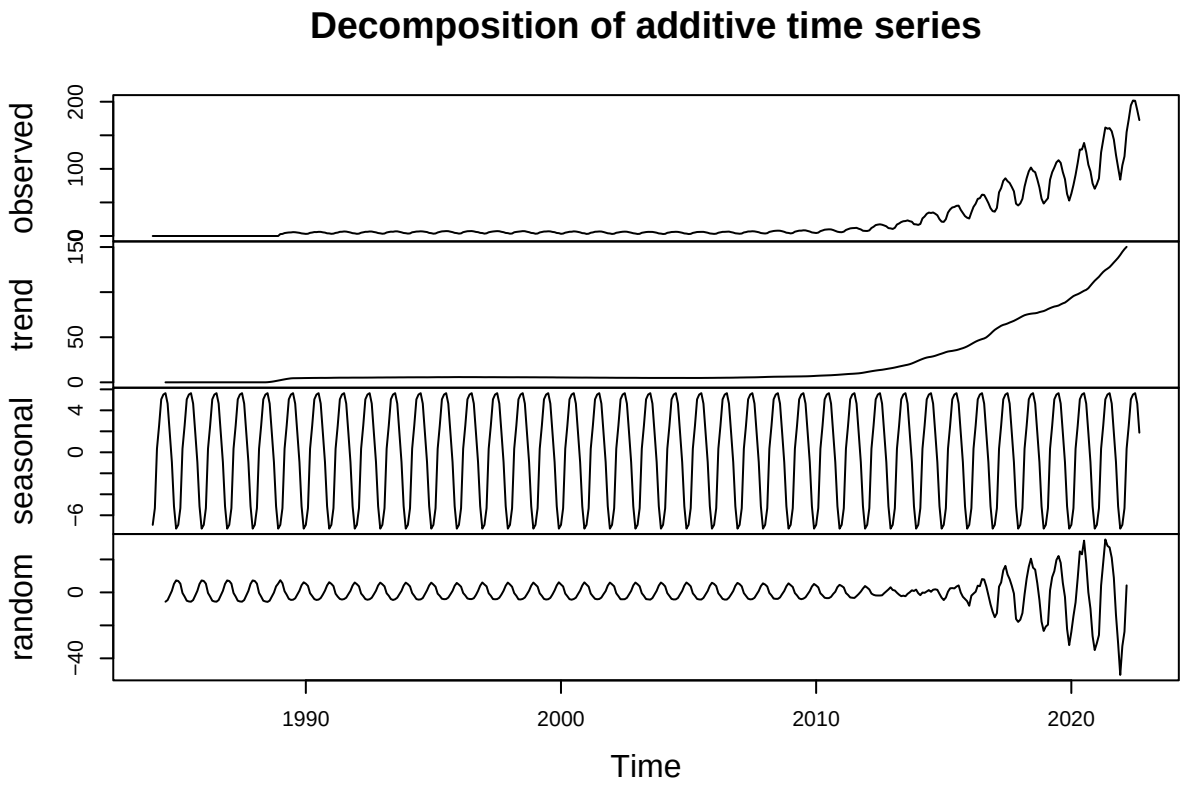
Q3

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?

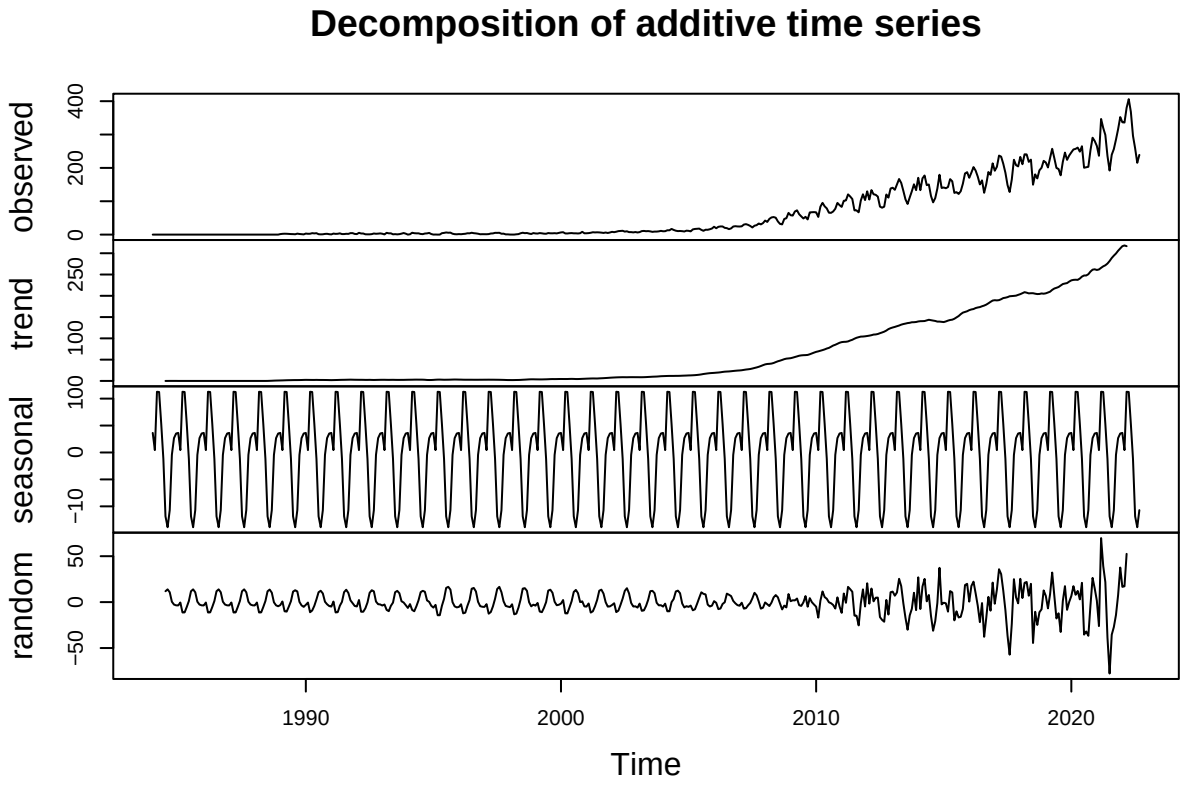
> Answer: The random component appears to contain some seasonality in it. This may be attributed to the possibility that the seasonal component is not additive but multiplicative (because the magnitude of the seasonal component in the original series seems change along with time.)

```
# Decompose two series and plot the results separately

# decompose solar energy consumption using additive
ts_renewable_sub5[,1] %>%
```

```
  decompose(type = "additive") %>%
  plot()
```




```
# decompose wind energy consumption using additive
ts_renewable_sub5[,2] %>%
  decompose(type = "additive") %>%
  plot()
```

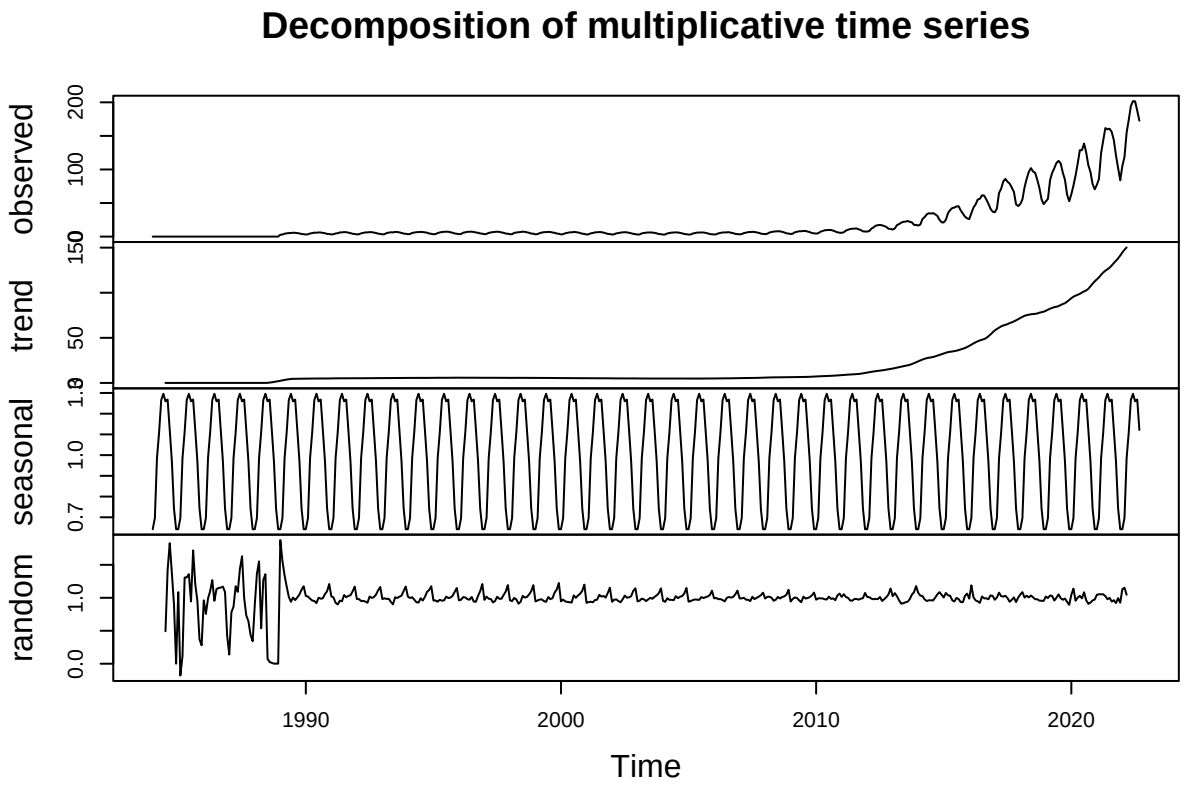


Q4

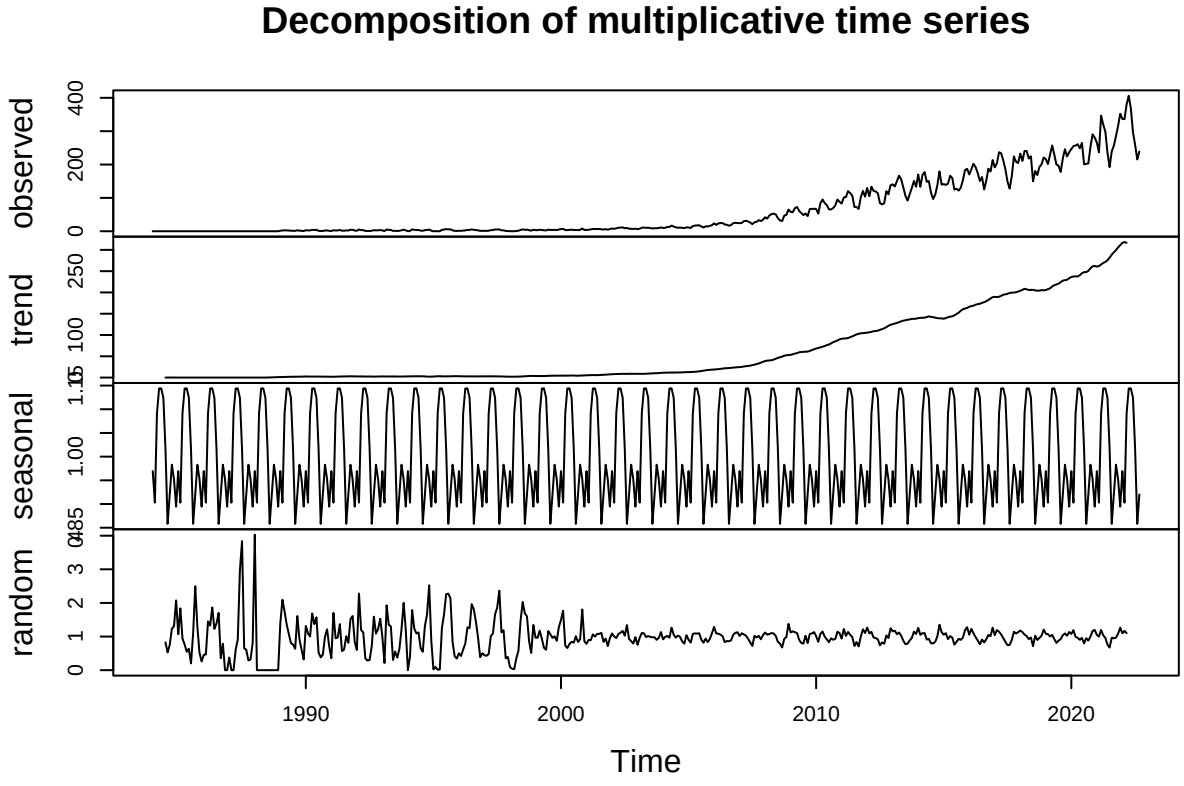
Use the decompose function again but now change the type of the seasonal component from additive to multiplicative. What happened to the random component this time?

> Answer: The random components of two series looks similar to a stationary trend. Seasonality seems not existing. So I ran MannKendall test for both random components. For solar energy consumption series, its MannKendall p-value is 0.91. So we fail to reject the null hypothesis, which means the random component of solar energy consumption data is stationary. For wind energy consumption series, its MannKendall p-value is 0.00528. So we reject the null hypothesis, which means the random component of wind energy consumption data is not stationary even after being decomposed. Maybe we need to difference the seasonal component of the wind series to solve this problem (very likely since the plot of the random component still contains some weak seasonality). But maybe this nonstationarity is not caused by seasonality. So, maybe we need to difference the de-seasoned series to detrend rather than use OLS to detrend. In practice, we can try both methods.

```
# Decompose two series and plot the results seperately

# decompose solar energy consumption using multiplicative
ts_renewable_sub5[,1] %>%
  decompose(type = "multiplicative") %>%
  plot()
```




```
# decompose wind energy consumption using multiplicative
ts_renewable_sub5[,2] %>%
  decompose(type = "multiplicative") %>%
  plot()
```



```r
# use MannKendall test to test the stationarity of the random component of
# solar energy consumption
decompose_solar <- ts_renewable_sub5[,1] %>%
  decompose(type = "multiplicative")

MannKendall(decompose_solar$random)
```

```
## tau = 0.00348, 2-sided pvalue =0.91219
```

```r
# use MannKendall test to test the stationarity of the random component of
# wind energy consumption
decompose_wind <- ts_renewable_sub5[,2] %>%
  decompose(type = "multiplicative")

MannKendall(decompose_wind$random)
```

```
## tau = 0.0877, 2-sided pvalue =0.0052873
```

Q5

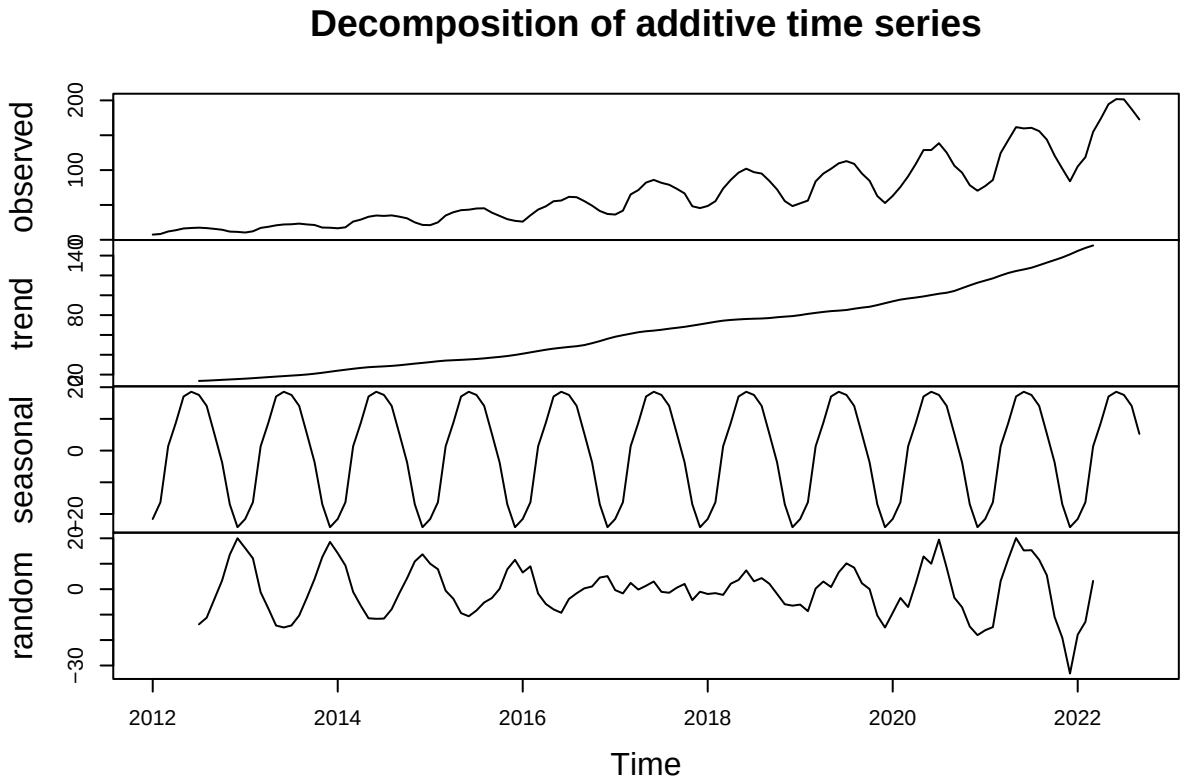
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

> Answer: We don't need all the historical data. From the trend component, we can see both consumptions remain almost constant before 2010s. There is not much useful information from those years. Having data points before this time will actually bias our analysis since these data points flatten the curve and may lead to underestimate of future consumption. So, I will suggest that we should remove these historical data points and redo our analysis.

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 seasonal components that depends on the level of the series.

```r
# Create a new data frame because date format will be converted to numbers
# by ts(), which can't be used to distill year component.
# So non-tsed series is better in this question.

ts_renewable_sub5_2012 <- renewable_sub5 %>%
  filter(year(Month) >= 2012)

# decompose solar energy consumption after 2012 using additive
ts_renewable_sub5_2012[,1] %>%
  ts(start = c(2012, 1), frequency = 12) %>%
  decompose(type = "additive") %>%
  plot()

# decompose wind energy consumption after 2012 using additive
ts_renewable_sub5_2012[,1] %>%
  ts(start = c(2012, 1), frequency = 12) %>%
  decompose(type = "additive") %>%
  plot()
```



Answer: the new random component still contains a potential pattern (seasonality). This is because seasonal components in both series are not additive, but multiplicative. The magnitudes of the seasonal components are changing along with time. So using additive can't fully exclude seasonality from the series.