ENV 797 - Time Series Analysis for Energy and Environment Applications | Spring 2025

Assignment 5 - Due date 02/18/25

Jessalyn Chuang

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_Sp25.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
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 errors
```

```r
library(readxl)
```

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 set - using xlsx package
energy_data <- read_excel(
  path = "./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx",
  sheet = 1,
  col_names = FALSE,
  skip = 12  # 'skip' is equivalent to 'startRow' - 1
)
```

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

```r
read_col_names <- read_excel(
  path = "./Data/Table_10.1_Renewable_Energy_Production_and_Consumption_by_Source.xlsx",
  sheet = 1,
  col_names = FALSE,
  skip = 10,    # Skip first 10 rows, start reading at row 11
  n_max = 1     # Read only 1 row (row 11)
)
```

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

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

```
## # 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)
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!

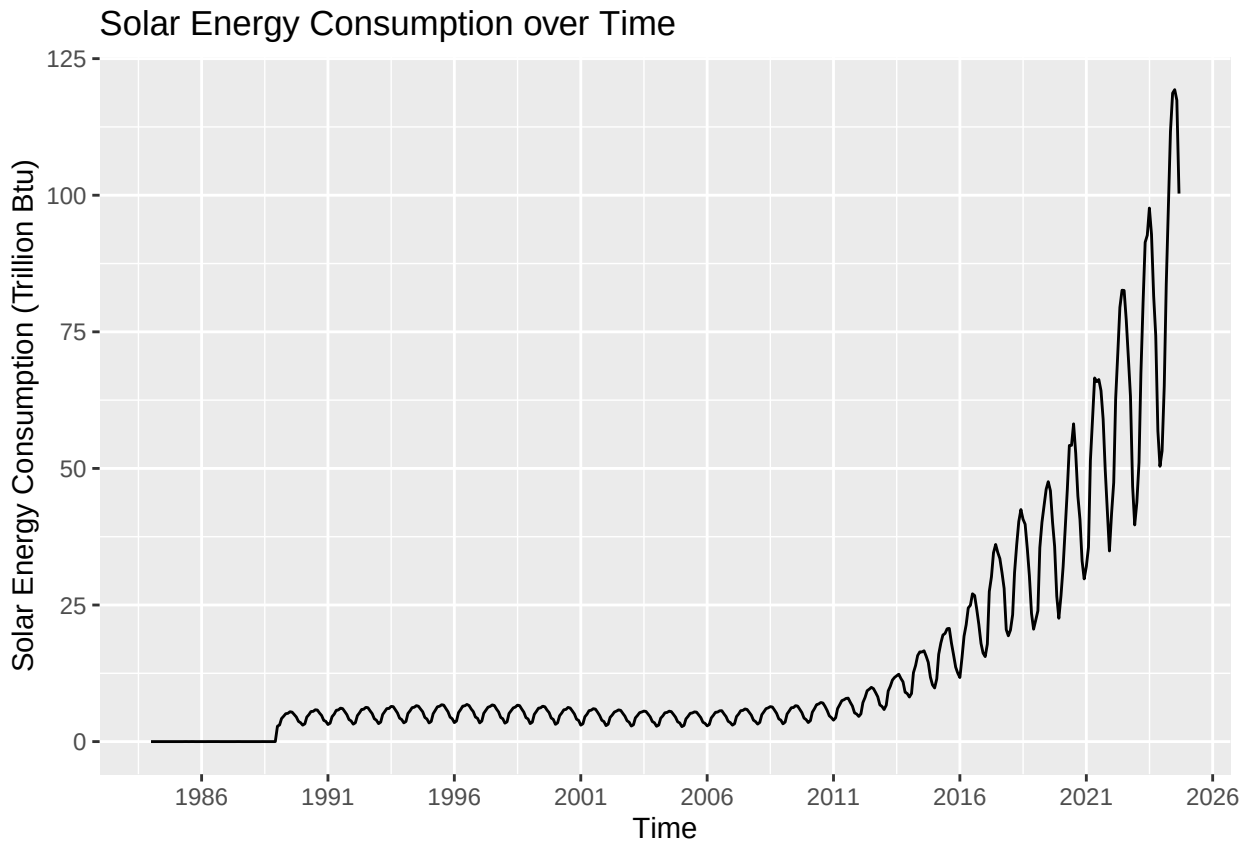
```
#pulling solar energy consumption and wind energy consumption data frame
renewable <- energy_data[, c(1, 8:9)]
renewable_cleaned <- renewable %>%
  mutate(across(2:3, as.numeric)) %>%
  drop_na() %>%
  mutate(Month = ymd(Month))
```

```
## Warning: There were 2 warnings in `mutate()`.
## The first warning was:
## i In argument: `across(2:3, as.numeric)`.
## Caused by warning:
## ! NAs introduced by coercion
## i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

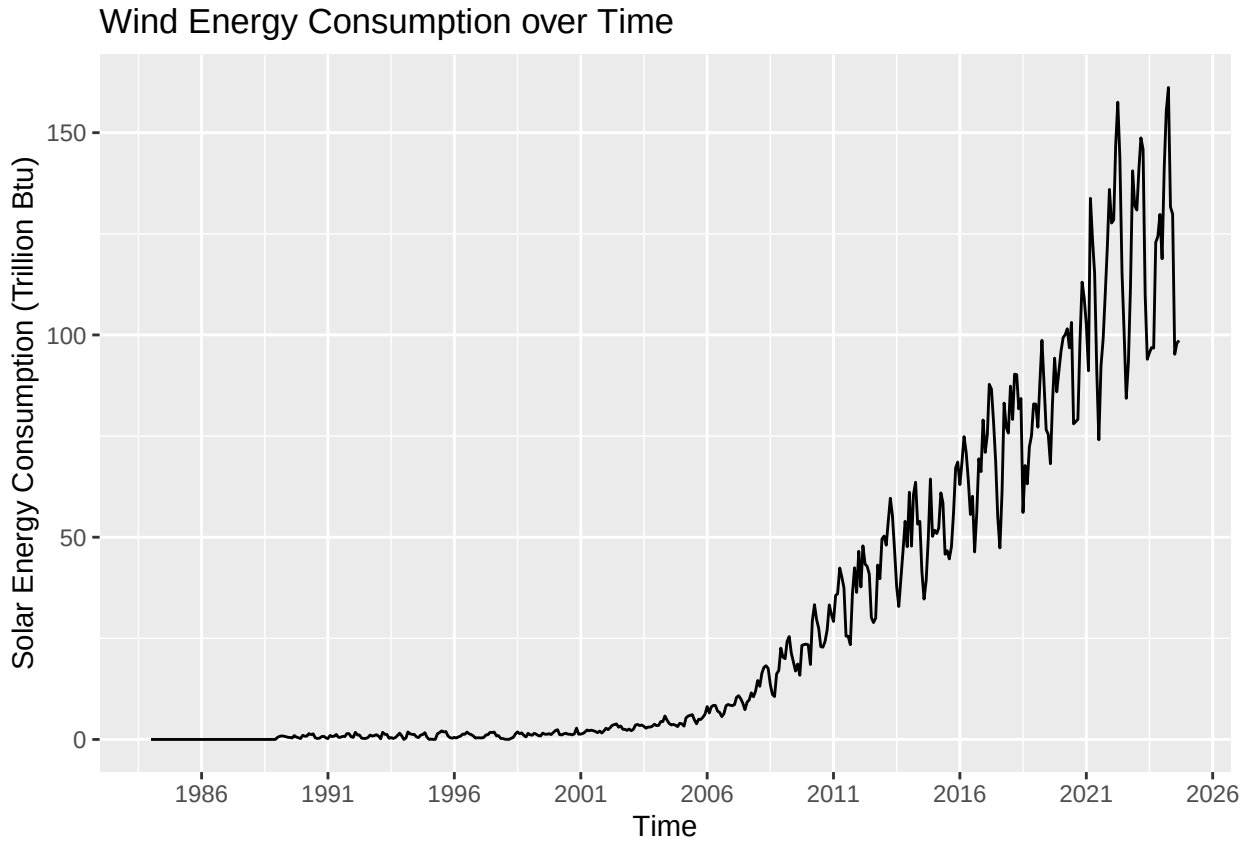
Q2

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")")`

```
#Solar Energy Consumption over time
solar_plot = ggplot(renewable_cleaned, aes(x = Month,
                                y =`Solar Energy Consumption`)) +
  geom_line() +
  labs(title = "Solar Energy Consumption over Time",
       x = "Time",
       y = "Solar Energy Consumption (Trillion Btu)") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")

solar_plot
```



```r
#Wind Energy Consumption over time
wind_plot = ggplot(renewable_cleaned, aes(x = Month,
                                y =`Wind Energy Consumption`)) +
  geom_line() +
  labs(title = "Wind Energy Consumption over Time",
       x = "Time",
       y = "Solar Energy Consumption (Trillion Btu)") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")

wind_plot
```

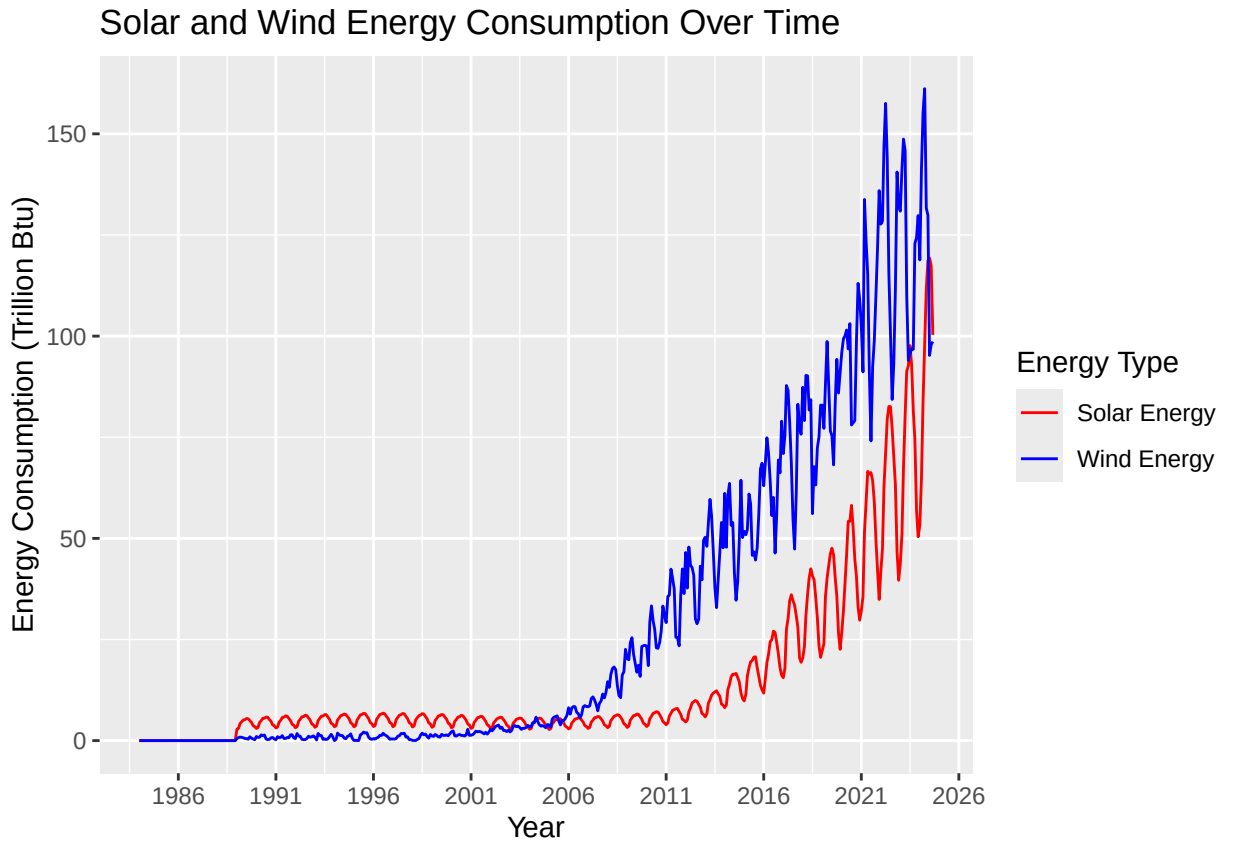


Q3

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() +
  geom_line(data = renewable_cleaned, aes(x = Month, y = `Solar Energy Consumption`, color = "Solar Ene
  geom_line(data = renewable_cleaned, aes(x = Month, y = `Wind Energy Consumption`, color = "Wind Energy
  scale_color_manual(values = c("Solar Energy" = "red", "Wind Energy" = "blue")) +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  labs(title = "Solar and Wind Energy Consumption Over Time",
```

```
       x = "Year",
       y = "Energy Consumption (Trillion Btu)",
       color = "Energy Type")
```



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()`.

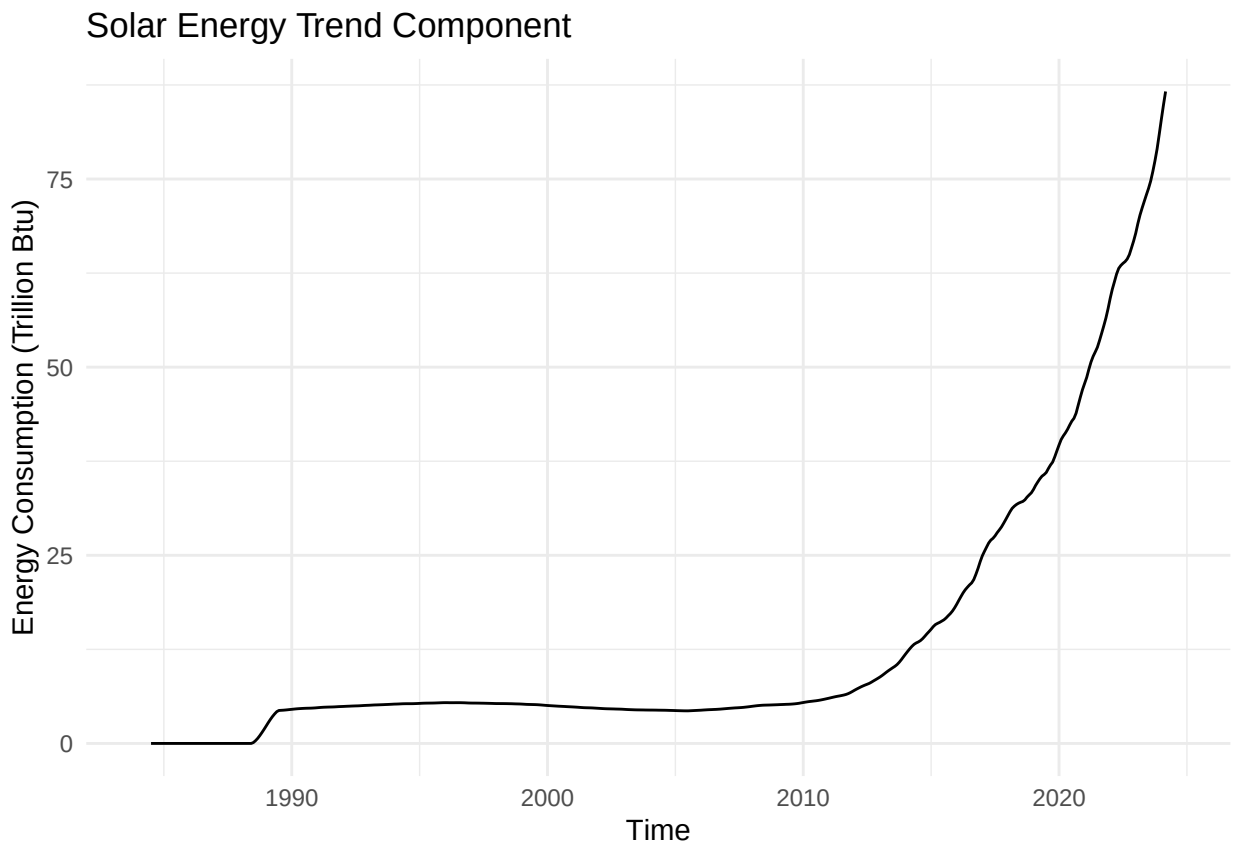
1) You have two options: alternative and multiplicative. Multiplicative models exhibit a change in frequency over time.
2) The trend is not a straight line because it uses a moving average method to detect trend.
3) The seasonal component of the time series is found by subtracting the trend component 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

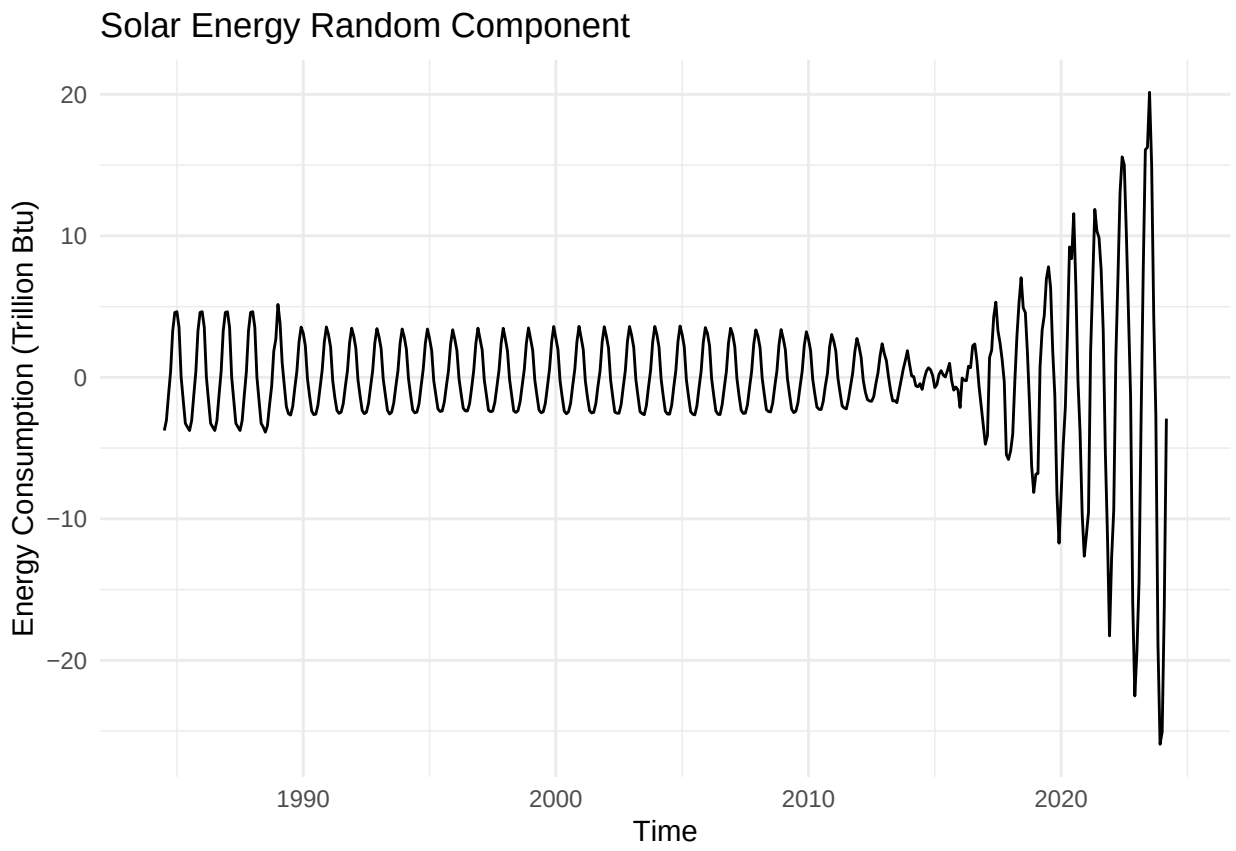
Transform wind and solar series into a time series object and apply the decompose function on them using the additive option, i.e., `decompose(ts_data, type = "additive")`. What can you say about the trend

component? What about the random component? Does the random component look random? Or does it appear to still have some seasonality on it?

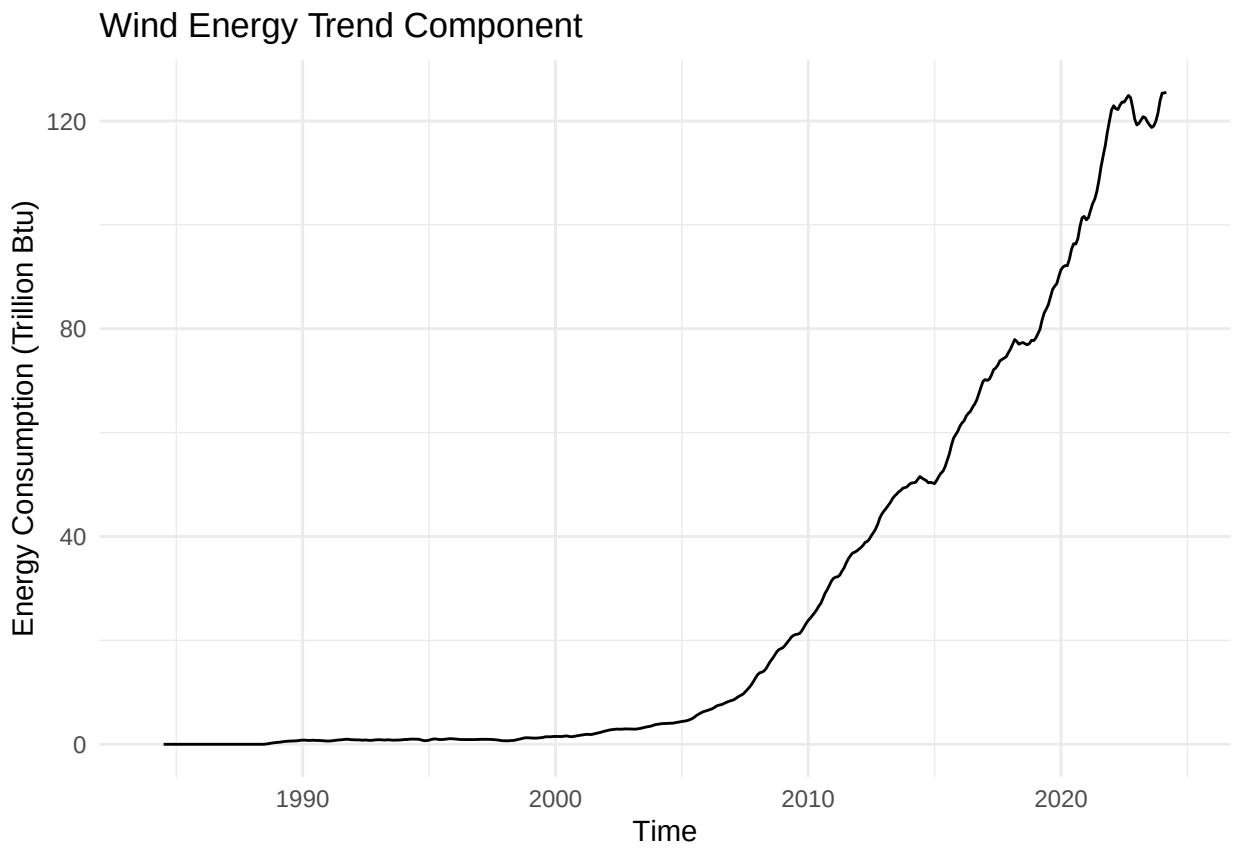
```
# Create time series for Solar and Wind separately
solar_ts <- ts(renewable_cleaned[,2], start = c(1984, 1), frequency = 12)
wind_ts <- ts(renewable_cleaned[,3], start = c(1984, 1), frequency = 12)

# Decompose Solar and Wind Time Series
solar_decomp <- decompose(solar_ts, type = "additive")
wind_decomp <- decompose(wind_ts, type = "additive")

# Plot Solar Trend and Random Components
autoplot(solar_decomp$trend) +
  ggtitle("Solar Energy Trend Component") +
  ylab("Energy Consumption (Trillion Btu)") +
  theme_minimal()
```



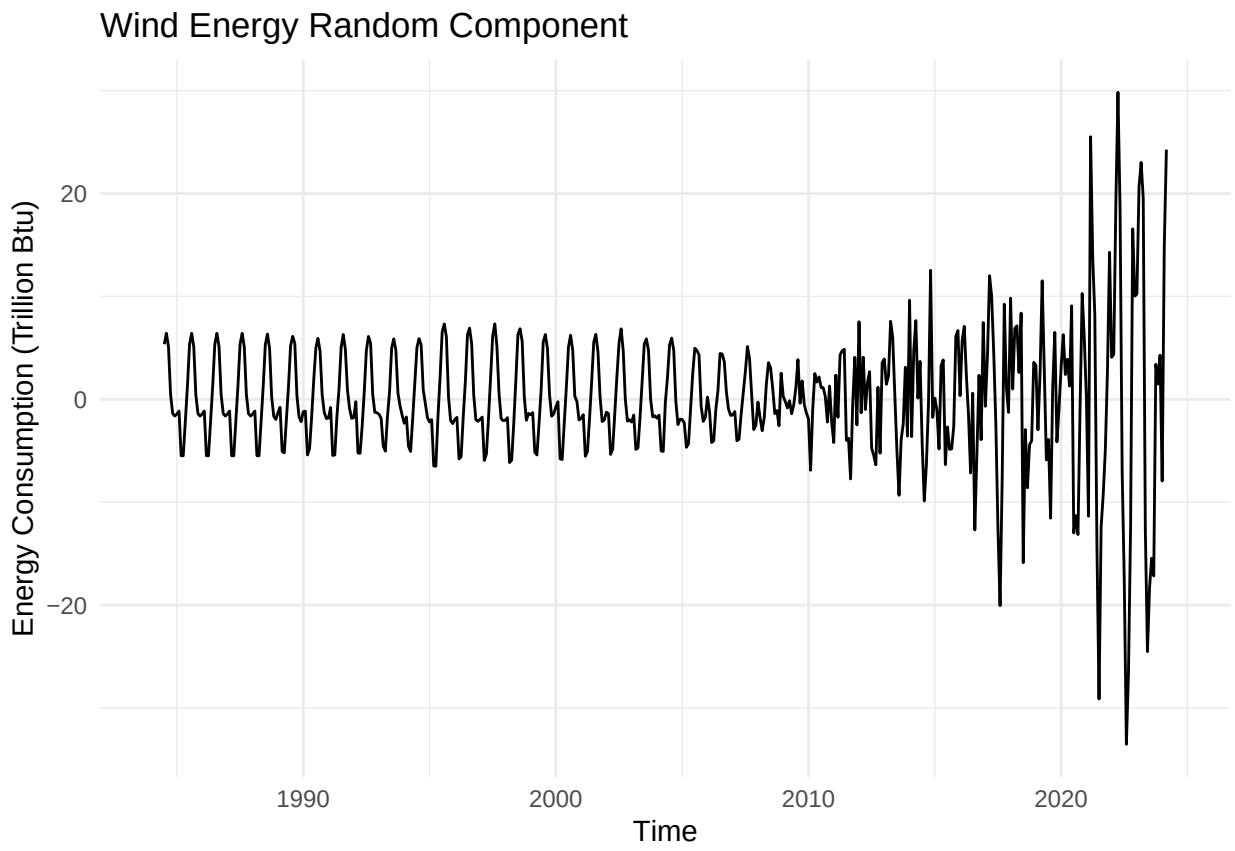

```
autoplot(solar_decomp$random) +
  ggtitle("Solar Energy Random Component") +
  ylab("Energy Consumption (Trillion Btu)") +
  theme_minimal()
```




```r
# Plot Wind Trend and Random Components
autoplot(wind_decomp$trend) +
  ggtitle("Wind Energy Trend Component") +
  ylab("Energy Consumption (Trillion Btu)") +
  theme_minimal()
```




```r
autoplot(wind_decomp$random) +
  ggtitle("Wind Energy Random Component") +
  ylab("Energy Consumption (Trillion Btu)")+
  theme_minimal()
```

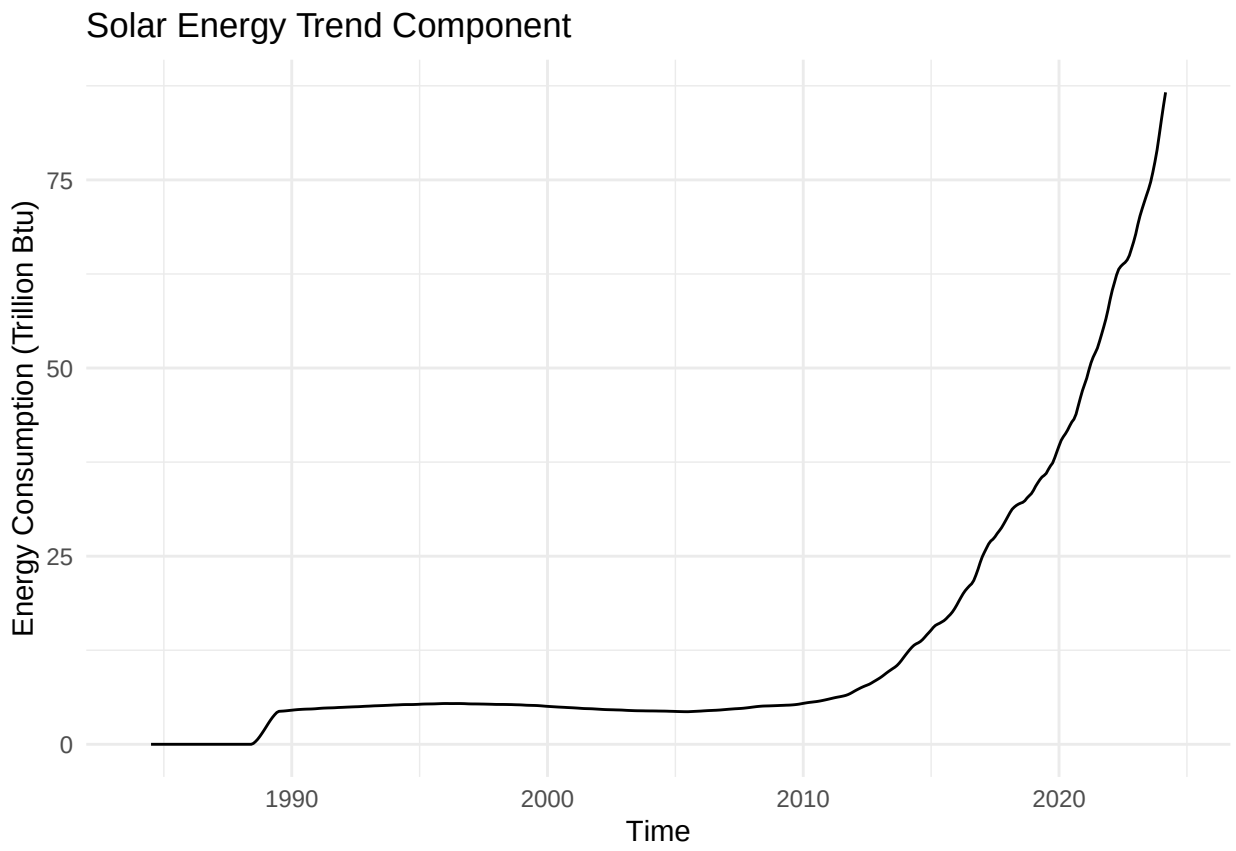


Answer: The trend components for both wind and solar energy show strong growth, especially from around 2010 onward. Wind energy has a sharper growth trajectory compared to solar energy. For the random component, it doesn't appear purely random. It shows a clear repeating pattern, especially earlier in the time series. This suggests that some seasonal pattern remains in the residuals, indicating that the decomposition may not have fully captured the seasonal structure.

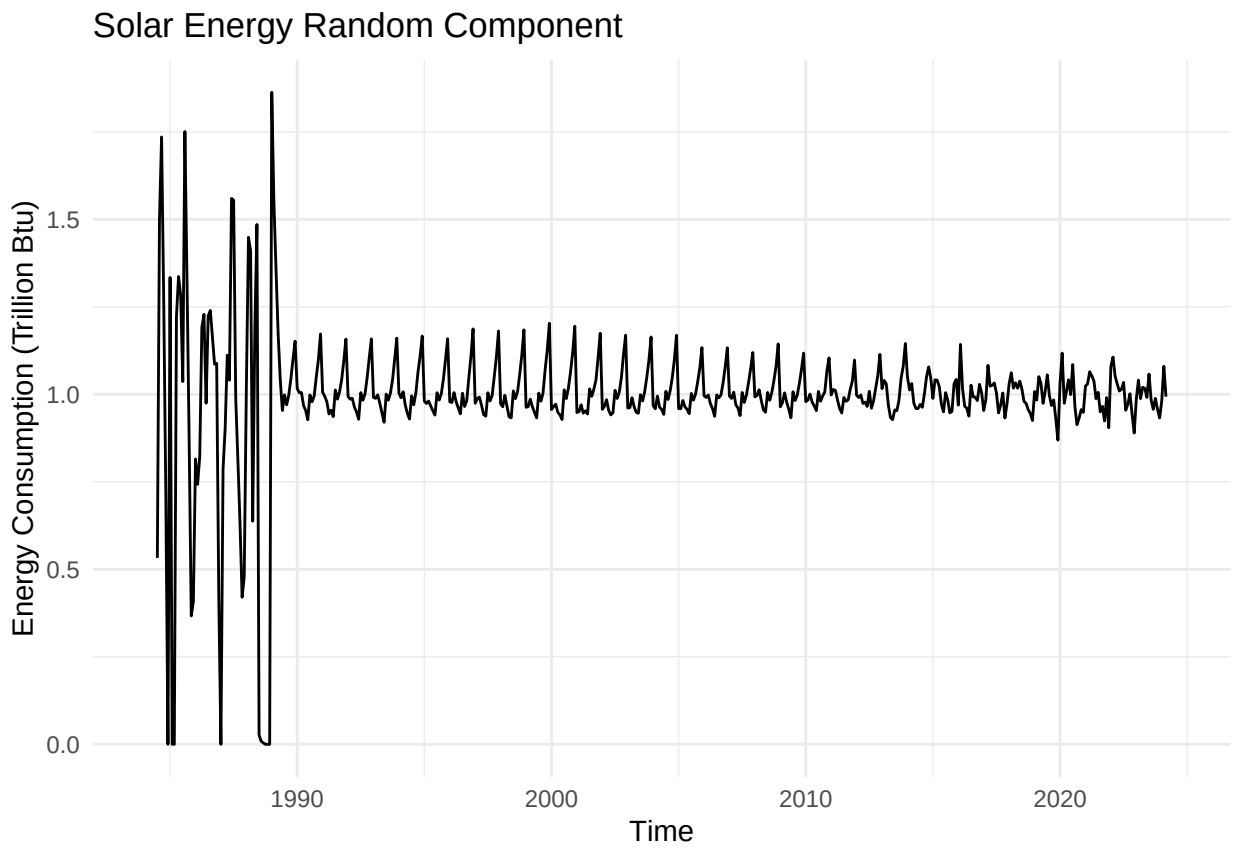
Q5

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?

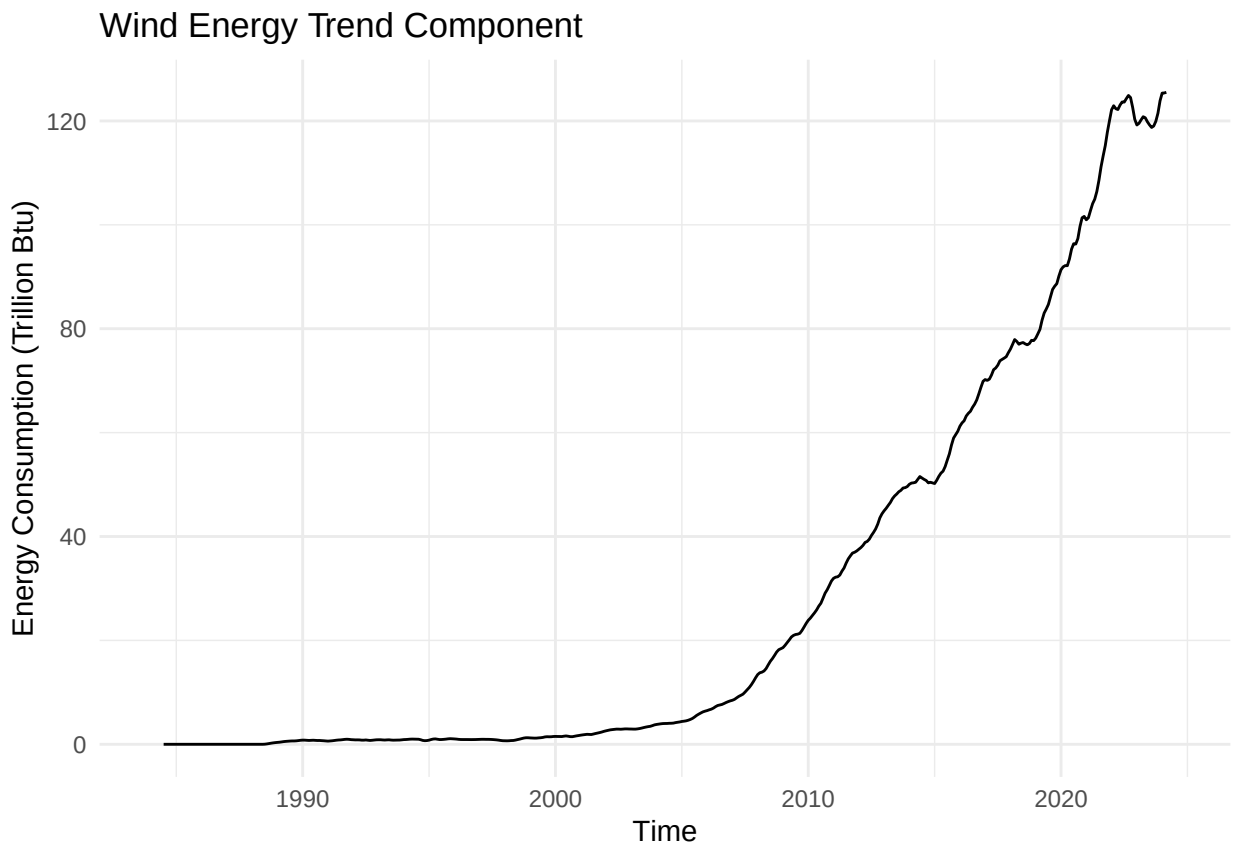
```
# Decompose Solar and Wind Time Series
solar_decomp_2 <- decompose(solar_ts, type = "multiplicative")
wind_decomp_2 <- decompose(wind_ts, type = "multiplicative")

# Plot Solar Trend and Random Components
autoplot(solar_decomp_2$trend) +
  ggtitle("Solar Energy Trend Component") +
  ylab("Energy Consumption (Trillion Btu)")+
  theme_minimal()
```



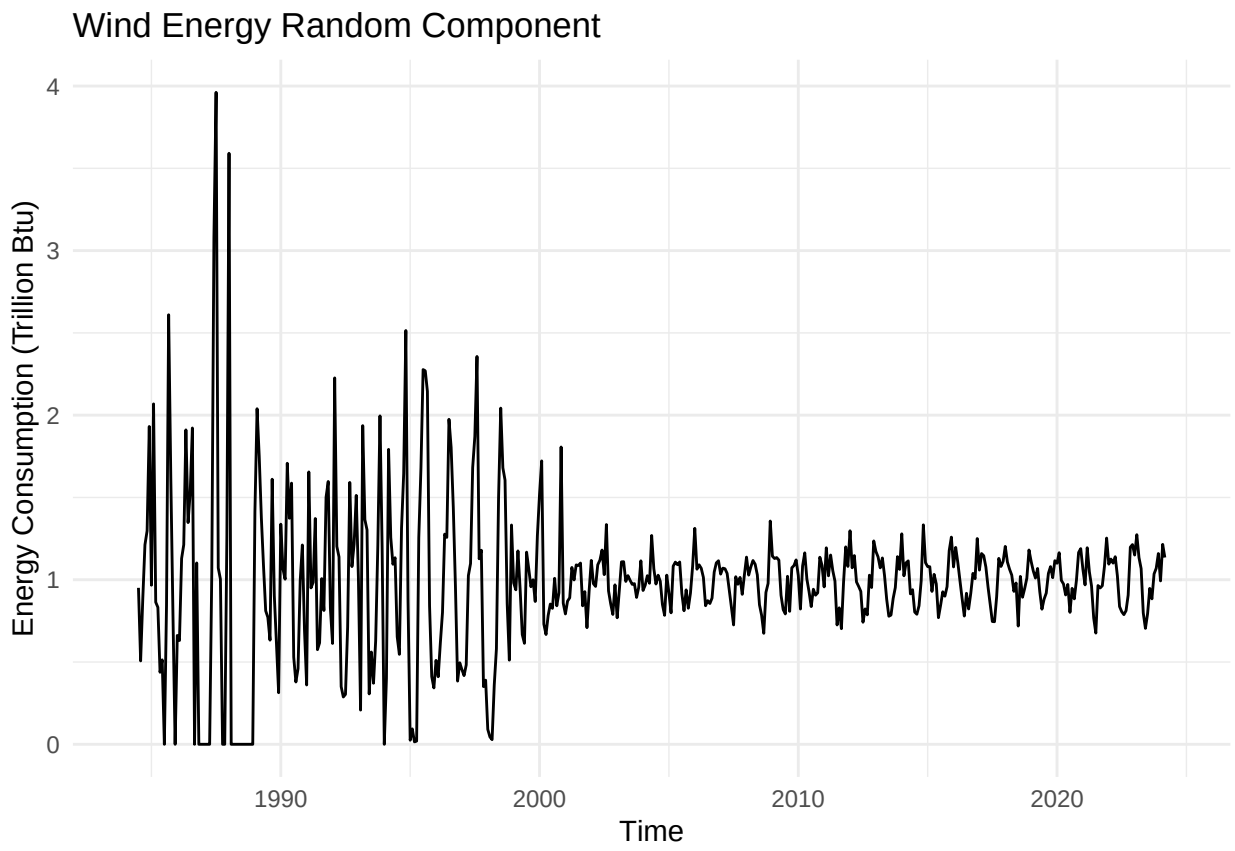

```
autoplot(solar_decomp_2$random) +
  ggtitle("Solar Energy Random Component") +
  ylab("Energy Consumption (Trillion Btu)")+
  theme_minimal()
```




```
# Plot Wind Trend and Random Components
autoplot(wind_decomp_2$trend) +
  ggtitle("Wind Energy Trend Component") +
  ylab("Energy Consumption (Trillion Btu)")+
  theme_minimal()
```




```
autoplot(wind_decomp_2$random) +
  ggtitle("Wind Energy Random Component") +
  ylab("Energy Consumption (Trillion Btu)")+
  theme_minimal()
```



Answer: In the additive decomposition, the random component sees increasing fluctuations in later years, suggesting that the model is not properly capturing the changing variability in the data. Since an additive model assumes constant variance, it forces fluctuations to remain fixed in magnitude regardless of the trend. If the actual data sees fluctuations grow as the trend increases, the additive model would misrepresent the pattern. Meanwhile, in the multiplicative decomposition, the larger oscillations are in the early years, indicating that when energy consumption was low, fluctuations were more pronounced, which aligns with the dataset having a non-constant variance. This suggests that the data's variability changes with the trend, making a multiplicative approach a better fit.

Q6

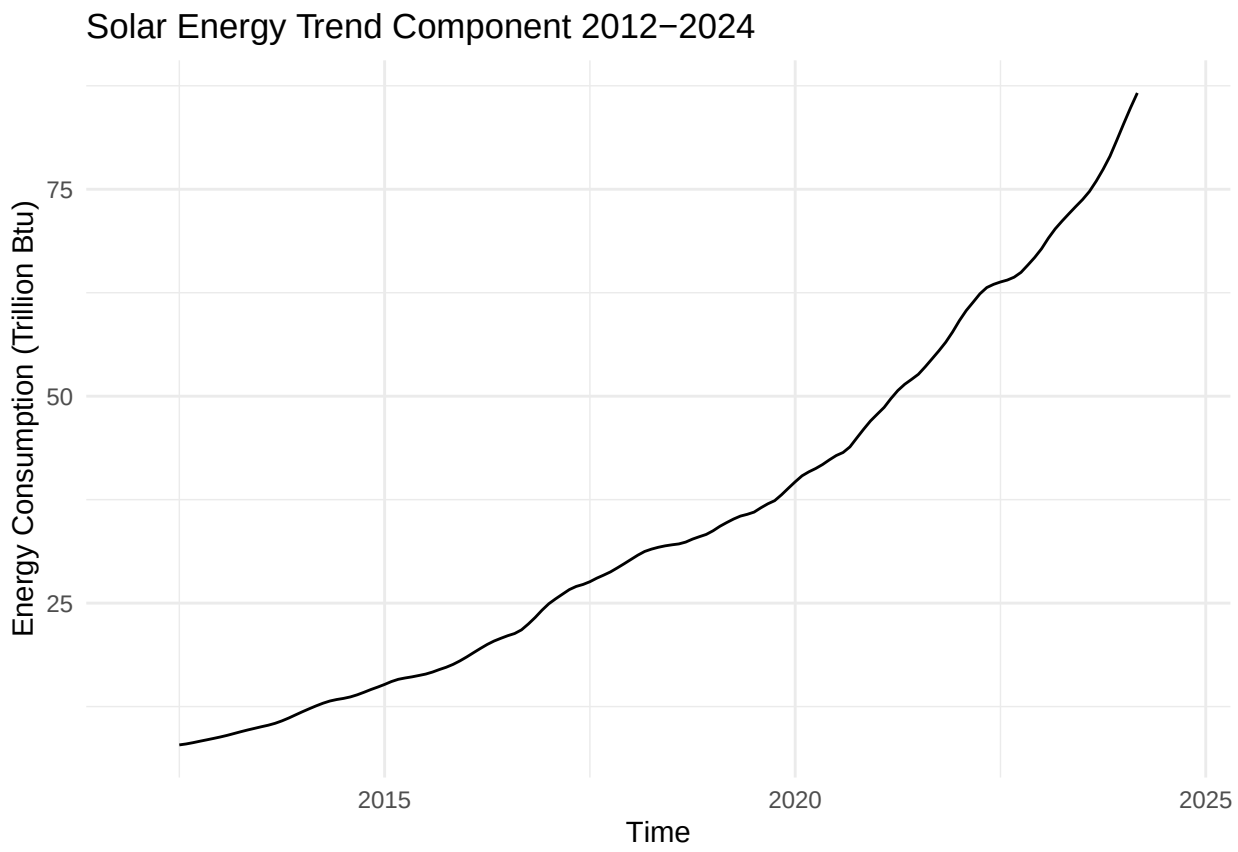
When fitting a model to this data, do you think you need all the historical data? Think about the data from 90s and early 20s. Are there any information from those years we might need to forecast the next six months of Solar and/or Wind consumption. Explain your response.

Answer: The early data from the 90s and early 2000s are not super useful for forecasting the next six months of solar and wind energy consumption because consumption remained relatively flat during that period and does not reflect the rapid growth seen in recent years. Since energy consumption patterns have shifted significantly, older data might not be helpful in explaining the recent trends. The model should be trained on more recent data to better capture the current dynamics.

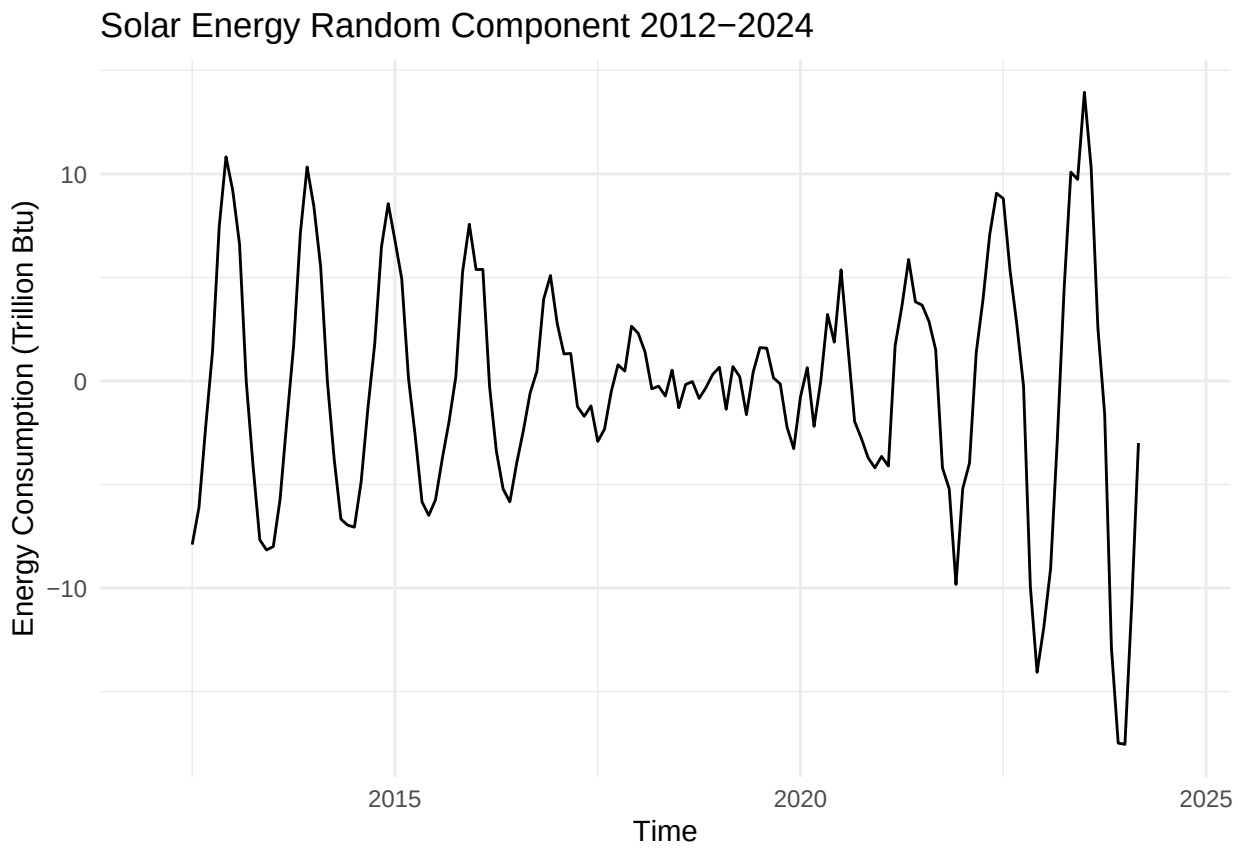
Q7

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.

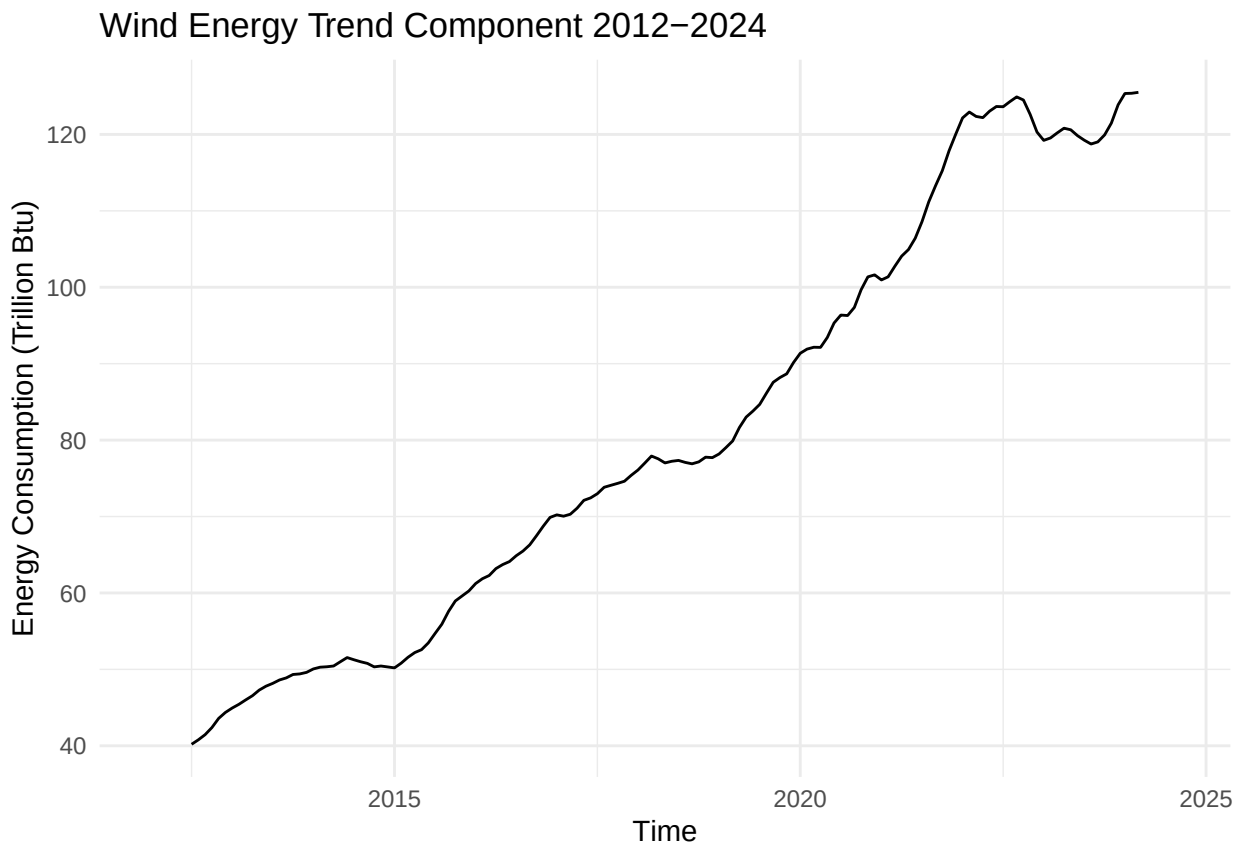
```
renewable_cleaned_filtered <- filter(renewable_cleaned, year(Month) >= 2012)
solar_ts_filtered <- ts(renewable_cleaned_filtered[,2], start = c(2012, 1), frequency = 12)
wind_ts_filtered <- ts(renewable_cleaned_filtered[,3], start = c(2012, 1), frequency = 12)

# Decompose Solar and Wind Time Series
solar_filtered_decomp <- decompose(solar_ts_filtered, type = "additive")
wind_filtered_decomp <- decompose(wind_ts_filtered, type = "additive")

# Plot Solar Trend and Random Components
autoplot(solar_filtered_decomp$trend) +
  ggtitle("Solar Energy Trend Component 2012-2024") +
  ylab("Energy Consumption (Trillion Btu)") +
  theme_minimal()
```



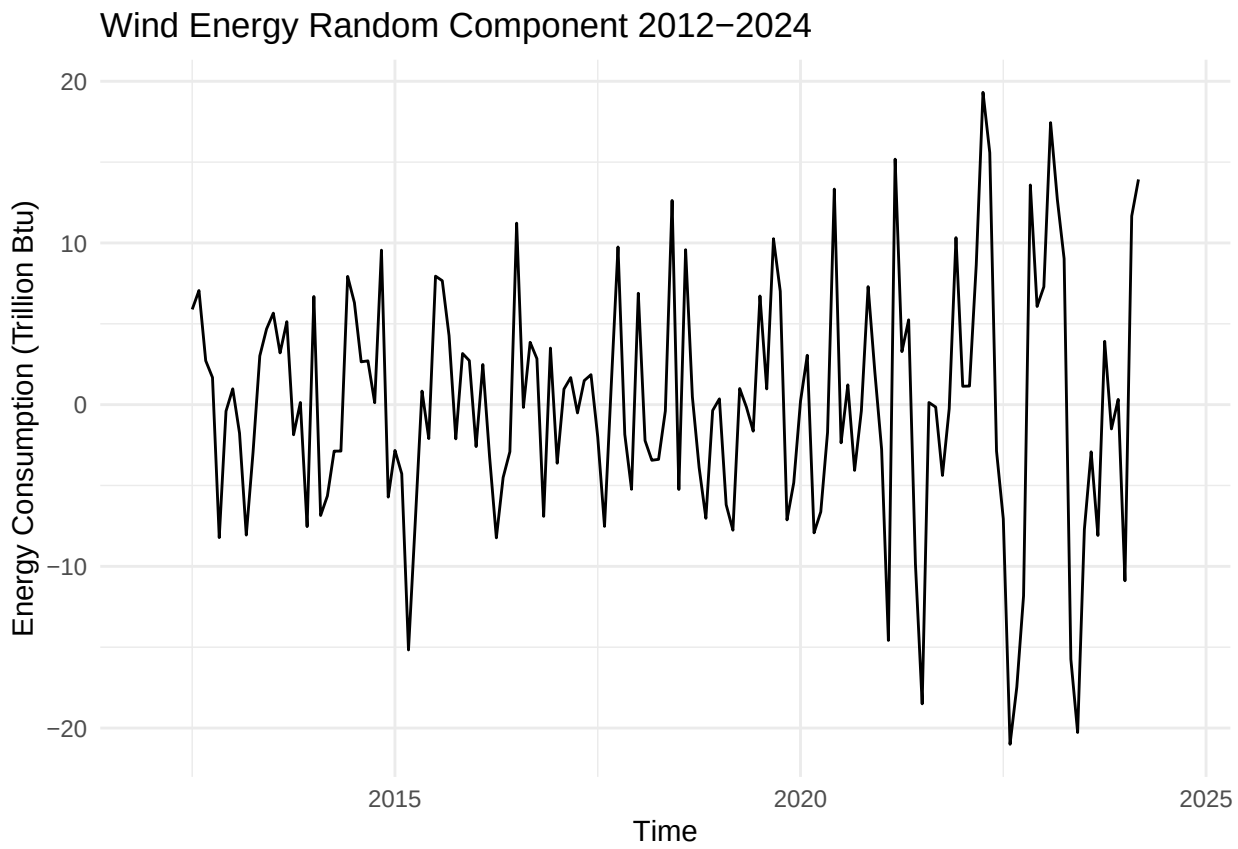

```
autoplot(solar_filtered_decomp$random) +
  ggtitle("Solar Energy Random Component 2012-2024") +
  ylab("Energy Consumption (Trillion Btu)") +
  theme_minimal()
```




```r
# Plot Wind Trend and Random Components
autoplot(wind_filtered_decomp$trend) +
  ggtitle("Wind Energy Trend Component 2012-2024") +
  ylab("Energy Consumption (Trillion Btu)") +
  theme_minimal()
```




```
autoplot(wind_filtered_decomp$random) +
  ggtitle("Wind Energy Random Component 2012-2024") +
  ylab("Energy Consumption (Trillion Btu)")+
  theme_minimal()
```



Answer: The random component in the additive decomposition does not look random because it has some structure and seems to have increasing variance over time, which suggests that the additive model is not the best choice. In our class discussion about seasonal components that depend on the level of series, when seasonal variations grow as the overall level of the series increases, an additive decomposition would force the seasonality to remain constant, which wouldn't be correct if seasonality depends on the level of the series (aka fluctuations grow as the trend increases, which it seems to do here).

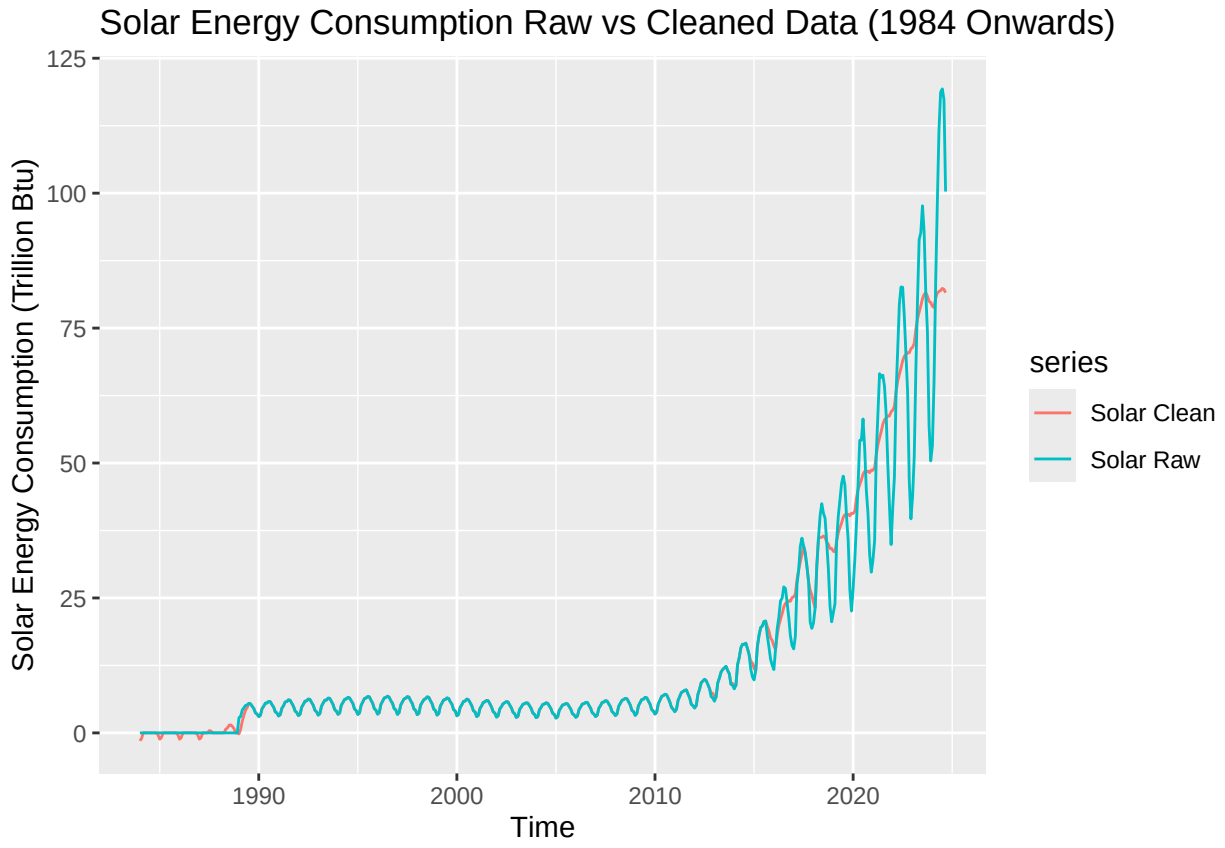
Identify and Remove outliers

Q8

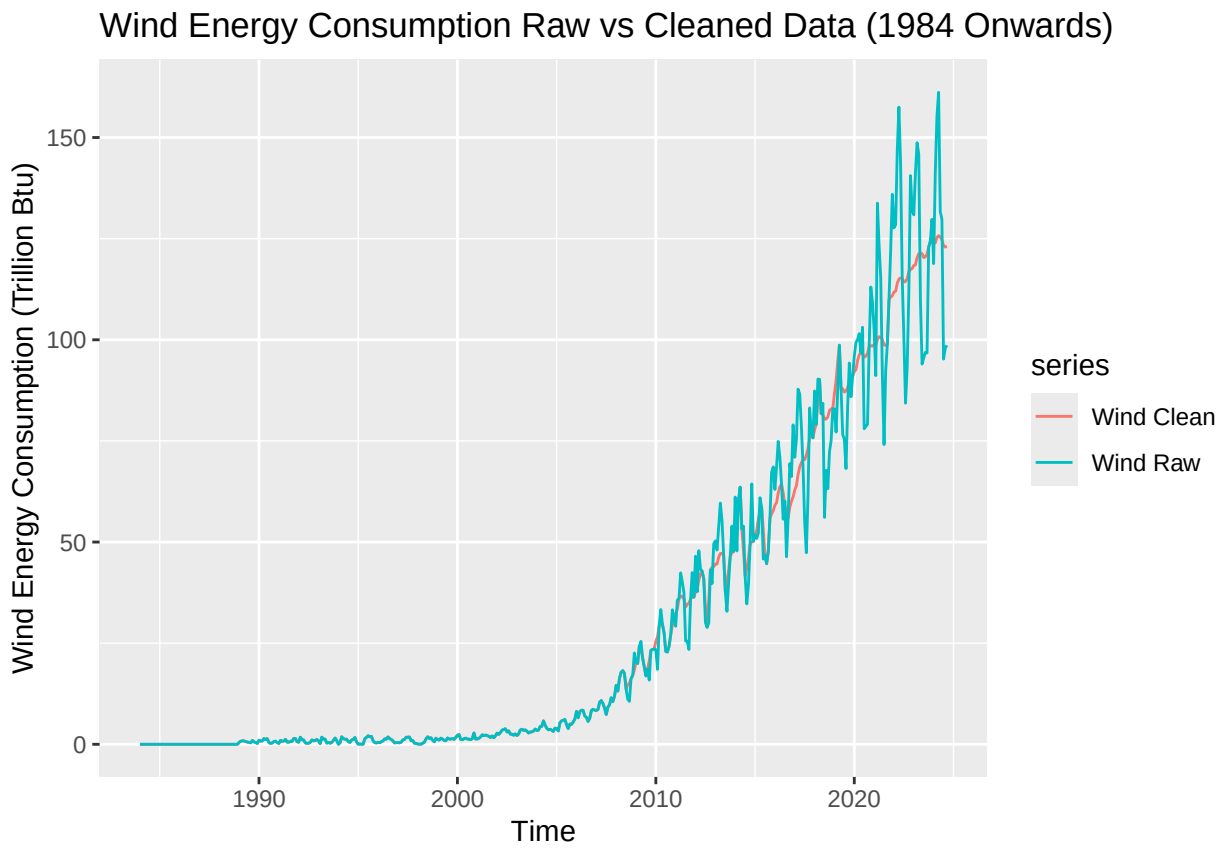
Apply the `tsclean()` to both series from Q4. Did the function removed any outliers from the series? Hint: Use `autoplot()` to check if there is difference between cleaned series and original series.

```
solar_ts_clean_1984 <- tsclean(solar_ts)
#Time index was not transferring over, so re-assigned time indices for wind_ts_clean
solar_ts_clean_1984 <- ts(solar_ts_clean_1984, start = start(solar_ts), frequency = frequency(solar_ts))

wind_ts_clean_1984 <- tsclean(wind_ts)
#Time index was not transferring over, so re-assigned time indices for wind_ts_clean
wind_ts_clean_1984 <- ts(wind_ts_clean_1984, start = start(wind_ts), frequency = frequency(wind_ts))

autoplot(solar_ts_clean_1984, series="Solar Clean") +
```

```
  autolayer(solar_ts, series="Solar Raw") +
  ylab("Solar Energy Consumption (Trillion Btu)") +
  ggtitle("Solar Energy Consumption Raw vs Cleaned Data (1984 Onwards)")
```




```
autoplot(wind_ts_clean_1984, series="Wind Clean") +
  autolayer(wind_ts, series="Wind Raw") +
  ylab("Wind Energy Consumption (Trillion Btu)") +
  ggtitle("Wind Energy Consumption Raw vs Cleaned Data (1984 Onwards)")
```

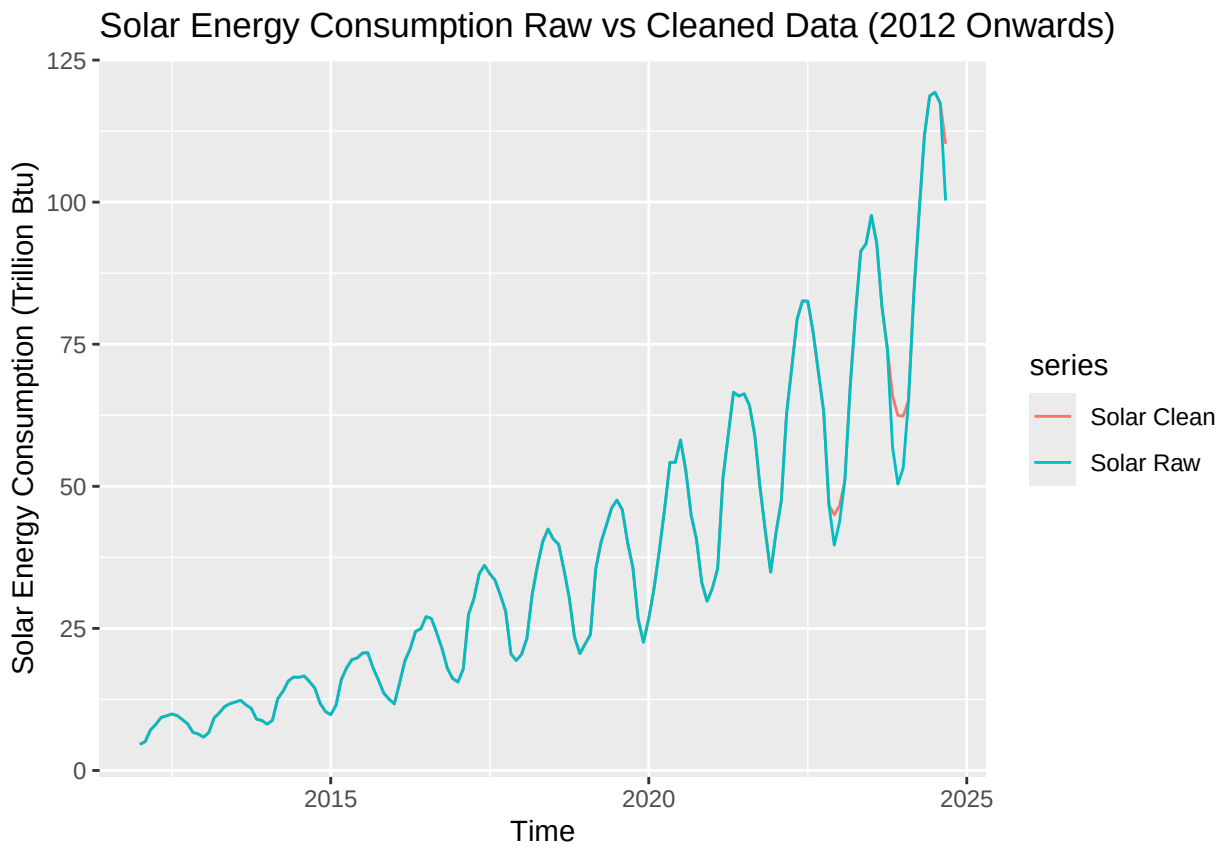


Answer: A lot of outliers were removed from the beginning of the dataset, but much more noticeably in the later years when the trend started to change and grow from the 2010s onwards for both wind and solar. This is because the function is learning from the entire dataset, which includes a long period of low and stable values before a steady trend upwards in recent years; there was much lower variance in early years compared to later years. This resulted in considerable outlier removal in efforts to make the series smoother in the later years based on what it understands from the years before there was a lot of upward trend.

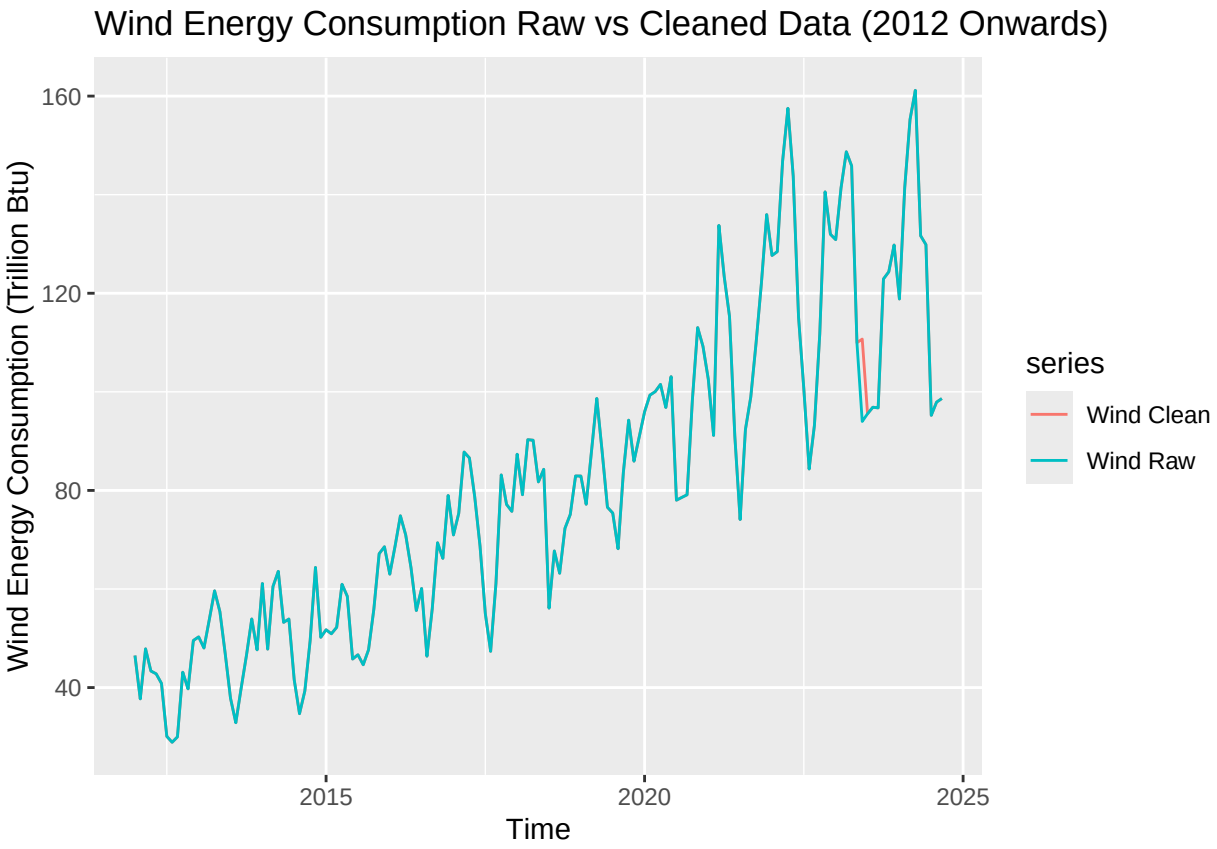
Q9

Redo number Q8 but now with the time series you created on Q7, i.e., the series starting in 2012. Using what `autoplot()` again what happened now? Did the function removed any outliers from the series?

```
solar_ts_clean <- tsclean(solar_ts_filtered)
#Time index was not transferring over, so re-assigned time indices for wind_ts_clean
solar_ts_clean <- ts(solar_ts_clean, start = start(solar_ts_filtered), frequency = frequency(solar_ts_f

wind_ts_clean <- tsclean(wind_ts_filtered)
#Time index was not transferring over, so re-assigned time indices for wind_ts_clean
wind_ts_clean <- ts(wind_ts_clean, start = start(wind_ts_filtered), frequency = frequency(wind_ts_filte

autoplot(solar_ts_clean, series="Solar Clean") +
  autolayer(solar_ts_filtered, series="Solar Raw") +
  ylab("Solar Energy Consumption (Trillion Btu)") +
  ggtitle("Solar Energy Consumption Raw vs Cleaned Data (2012 Onwards)")
```




```
autoplot(wind_ts_clean, series="Wind Clean") +
  autolayer(wind_ts_filtered, series="Wind Raw") +
  ylab("Wind Energy Consumption (Trillion Btu)") +
  ggtitle("Wind Energy Consumption Raw vs Cleaned Data (2012 Onwards)")
```



Answer: There was much less outlier removal. When using data from 2012 onwards, the function only learns from the years that have seen high growth, which means it does not consider as much of the high growth as outliers compared to when the older historical data was included. This means that tsclean() is less likely to treat large fluctuations as outliers since they are more common in the recent period.