Assignment 8: Time Series Analysis

Samantha Pace

Fall 2023

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on generalized linear models.

Directions

1. Rename this file `<FirstLast>_A08_TimeSeries.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.

Set up

1. Set up your session:

- Check your working directory
- Load the tidyverse, lubridate, zoo, and trend packages
- Set your ggplot theme

```
getwd()
```

```
## [1] "/home/guest/R/EDE_Fall2023"
```

```
setwd("/home/guest/R/EDE_Fall2023")
getwd()
```

```
## [1] "/home/guest/R/EDE_Fall2023"
```

```
#checking that the EDE_Fall2023 Folder is my wd

# loading packages
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- 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
```

```
library(lubridate)
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(trend)

# creating my theme based on the black and white theme
mytheme <-
  theme_bw(base_size = 14)+
    theme(panel.grid.major = element_line(colour = "grey80"))+
  theme(plot.title = element_text(size = rel(1)), legend.position = "top")
```

2. Import the ten datasets from the Ozone_TimeSeries folder in the Raw data folder. These contain ozone concentrations at Garinger High School in North Carolina from 2010-2019 (the EPA air database only allows downloads for one year at a time). Import these either individually or in bulk and then combine them into a single dataframe named `GaringerOzone` of 3589 observation and 20 variables.

```
#1
# the following 10 objects are the 10 individual datasets being importing.
File.2010 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2010_raw.csv",
           stringsAsFactors = TRUE)
File.2011 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2011_raw.csv",
           stringsAsFactors = TRUE)
File.2012 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2012_raw.csv",
           stringsAsFactors = TRUE)
File.2013 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2013_raw.csv",
           stringsAsFactors = TRUE)
File.2014 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2014_raw.csv",
           stringsAsFactors = TRUE)
File.2015 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2015_raw.csv",
           stringsAsFactors = TRUE)
File.2016 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2016_raw.csv",
           stringsAsFactors = TRUE)
File.2017 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2017_raw.csv",
           stringsAsFactors = TRUE)
File.2018 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2018_raw.csv",
           stringsAsFactors = TRUE)
File.2019 <-
  read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2019_raw.csv",
           stringsAsFactors = TRUE)
```

```r
# Combining all 10 dataframes into one object called GaringerOzone with rbind.
GaringerOzone <-
  rbind(File.2010, File.2011, File.2012, File.2013,
        File.2014, File.2015, File.2016, File.2017,
        File.2018, File.2019)
```

Wrangle

3. Set your date column as a date class.

4. Wrangle your dataset so that it only contains the columns Date, Daily.Max.8.hour.Ozone.Concentration, and DAILY_AQI_VALUE.

5. Notice there are a few days in each year that are missing ozone concentrations. We want to generate a daily dataset, so we will need to fill in any missing days with NA. Create a new data frame that contains a sequence of dates from 2010-01-01 to 2019-12-31 (hint: `as.data.frame(seq())`). Call this new data frame Days. Rename the column name in Days to "Date".

6. Use a `left_join` to combine the data frames. Specify the correct order of data frames within this function so that the final dimensions are 3652 rows and 3 columns. Call your combined data frame GaringerOzone.

```r
# 3
# Fixing date column to be a date object
GaringerOzone$Date <- mdy(GaringerOzone$Date)

# 4
# selecting the only three columns
GaringerOzone <-
  GaringerOzone %>%
  select(Date, Daily.Max.8.hour.Ozone.Concentration, DAILY_AQI_VALUE)

# 5
# Creating a new data frame called Days with dates
Days <- as.data.frame(seq(as.Date('2010-01-01'),as.Date('2019-12-31'),by = 1))

colnames(Days) <- "Date"

# 6
# joining the Days and GaringerOzone datasets such that all the rows from Days will
# included and all values of both dataframes
GaringerOzone <-
  left_join(Days, GaringerOzone)
```

```
## Joining with `by = join_by(Date)`
```

Visualize

7. Create a line plot depicting ozone concentrations over time. In this case, we will plot actual concentrations in ppm, not AQI values. Format your axes accordingly. Add a smoothed line showing any linear trend of your data. Does your plot suggest a trend in ozone concentration over time?

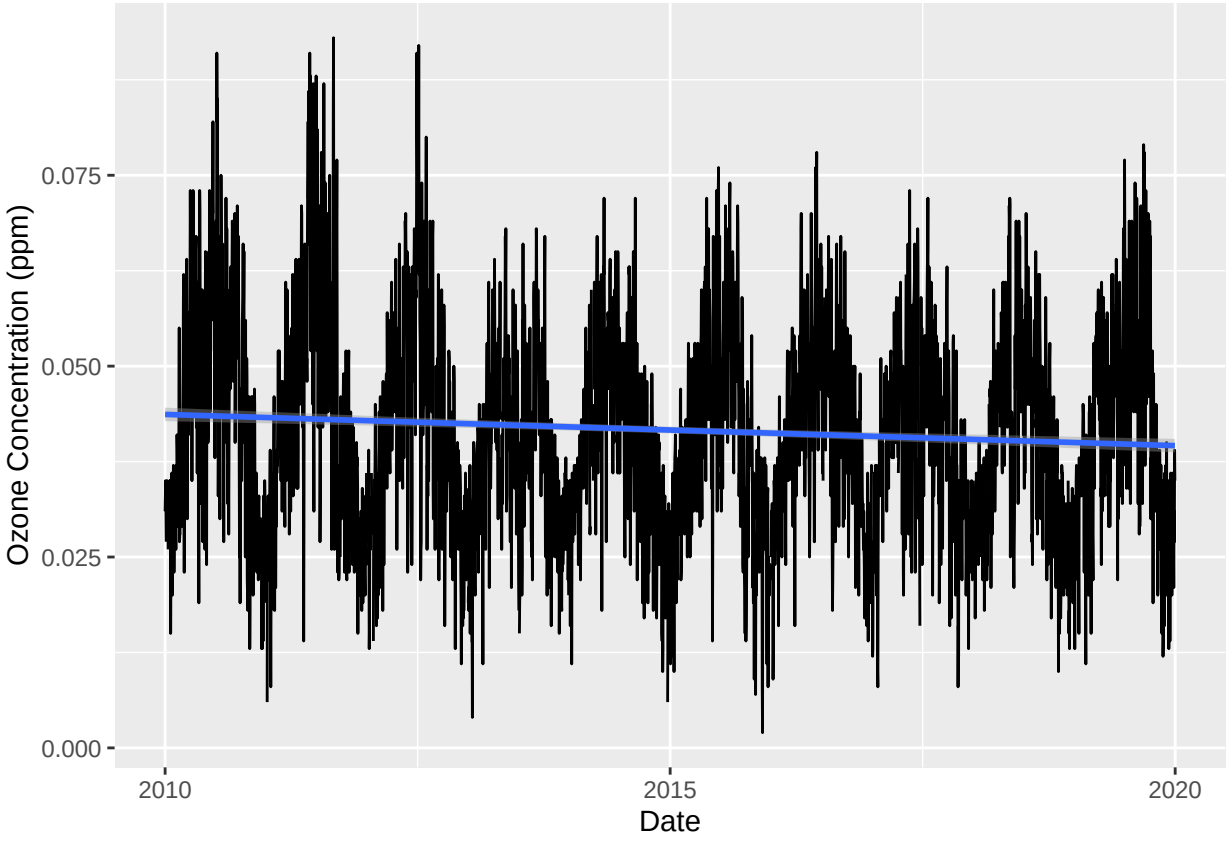
```r
#7
# creating a ggplot mapping the daily ozone over time and adding a smoothed line
GaringerOzone.plot <-
  ggplot(GaringerOzone, (aes(x = Date, y = Daily.Max.8.hour.Ozone.Concentration))) +
```

```
  geom_line() +
  geom_smooth(method=lm) +
  ylab("Ozone Concentration (ppm)")

#displaying plot
print(GaringerOzone.plot)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 63 rows containing non-finite values (`stat_smooth()`).
```



Answer: The plot shows a slight downward trend in ozone concentration over time. In 2010, the ozone concentration was closer to 0.044 ppm and by 2020 the ozone concentration was approximately 0.039 ppm.

Time Series Analysis

Study question: Have ozone concentrations changed over the 2010s at this station?

8. Use a linear interpolation to fill in missing daily data for ozone concentration. Why didn't we use a piecewise constant or spline interpolation?

```
#8
#determines that there are NA's
head(GaringerOzone$Daily.Max.8.hour.Ozone.Concentration)
```

```
## [1] 0.031 0.033 0.035 0.031 0.027    NA
```

```
# determines the number of NA's
summary(GaringerOzone$Daily.Max.8.hour.Ozone.Concentration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## 0.00200 0.03200 0.04100 0.04163 0.05100 0.09300      63
```

```
# use na.approx in order to do a linear interpolation
GaringerOzone <-
  GaringerOzone %>%
  mutate(Daily.Max.8.hour.Ozone.Concentration = na.approx(
    GaringerOzone$Daily.Max.8.hour.Ozone.Concentration))

# check that there are no NA's in the column of interpolated data
summary(GaringerOzone$Daily.Max.8.hour.Ozone.Concentration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00200 0.03200 0.04100 0.04151 0.05100 0.09300
```

Answer: With the seasonality, trend, and linearity of this data, it is best to use the linear interpolation approach and connect the lines between the missing points. The piecewise function would set the missing data equal to the nearest known data, but this approach may be less accurate and put more noise in the data so we don't want to use that. The spline function applies a quadratic equation, and this is not appropriate for this data set because it doesn't follow a quadratic or parabolic pattern. Linear interpolation is the best approach.

9. Create a new data frame called `GaringerOzone.monthly` that contains aggregated data: mean ozone concentrations for each month. In your pipe, you will need to first add columns for year and month to form the groupings. In a separate line of code, create a new Date column with each month-year combination being set as the first day of the month (this is for graphing purposes only)

```
#9
# creating a new object with the GaringerOzone data; creating a new year column,
# a new month column, and combining the columns. Then grouping them by Year,
# Month, MonthYear, and summarizing with the daily mean ozone concentration.
GaringerOzone.monthly <-
  GaringerOzone %>%
  mutate(Year = year(Date)) %>%
  mutate(Month = month(Date)) %>%
  mutate(MonthYear = my(paste0(Month, "-", Year))) %>%
  group_by(Year, Month, MonthYear) %>%
  summarise(MeanOzoneAggregated = mean(Daily.Max.8.hour.Ozone.Concentration))
```

```
## `summarise()` has grouped output by 'Year', 'Month'. You can override using the
## `.groups` argument.
```

10. Generate two time series objects. Name the first `GaringerOzone.daily.ts` and base it on the dataframe of daily observations. Name the second `GaringerOzone.monthly.ts` and base it on the monthly average ozone values. Be sure that each specifies the correct start and end dates and the frequency of the time series.

```
#10

#creating time series values for daily and monthly with the ts function
GaringerOzone.daily.ts <-
  ts(GaringerOzone$Daily.Max.8.hour.Ozone.Concentration,
     start = c(2010,1), frequency = 365)
```

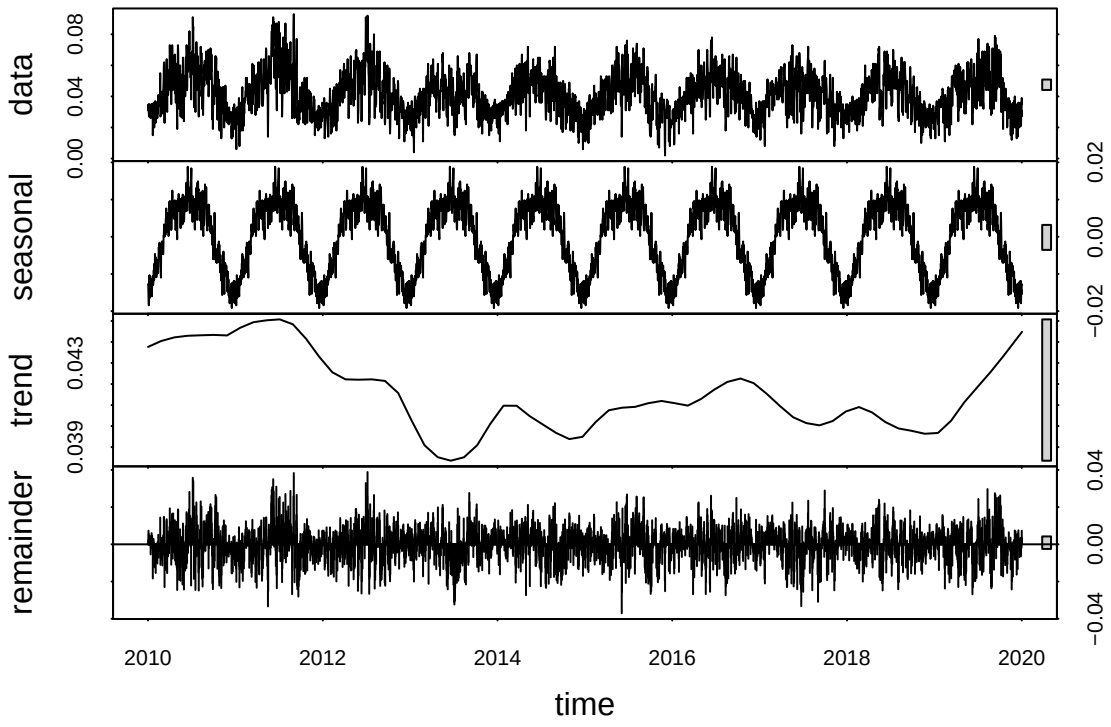
```
  GaringerOzone.monthly.ts <-
  ts(GaringerOzone.monthly$MeanOzoneAggregated,
     start = c(2010,1), frequency = 12)
```

11. Decompose the daily and the monthly time series objects and plot the components using the `plot()` function.

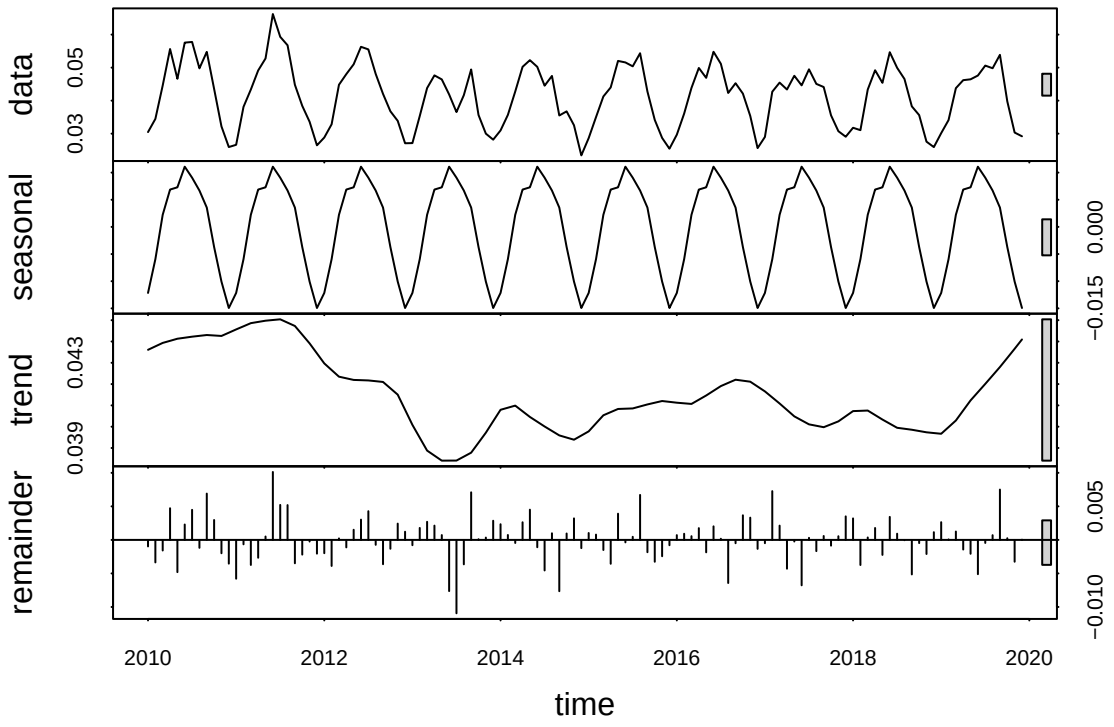
```
#11
# decomposing daily and monthly time series with stl function
GaringerOzone.daily.decomposed <-
  stl(GaringerOzone.daily.ts, s.window = "periodic")

GaringerOzone.monthly.decomposed <-
  stl(GaringerOzone.monthly.ts, s.window = "periodic")

#plotting each decomposed time series
plot(GaringerOzone.daily.decomposed)
```




```
plot(GaringerOzone.monthly.decomposed)
```



12. Run a monotonic trend analysis for the monthly Ozone series. In this case the seasonal Mann-Kendall is most appropriate; why is this?

```
#12
# Running Seasonal Mann Kendall test on monthly time series
Monthly.Ozone.trend <-
  Kendall::SeasonalMannKendall(GaringerOzone.monthly.ts)

Monthly.Ozone.trend
```

```
## tau = -0.143, 2-sided pvalue =0.046724
```

```
summary(Monthly.Ozone.trend)
```

```
## Score =  -77 , Var(Score) = 1499
## denominator =  539.4972
## tau = -0.143, 2-sided pvalue =0.046724
```

```
# summary shows the p value is 0.0467, which is less than 0.05
# so we can reject the null hypothesis that the data is stationary
```

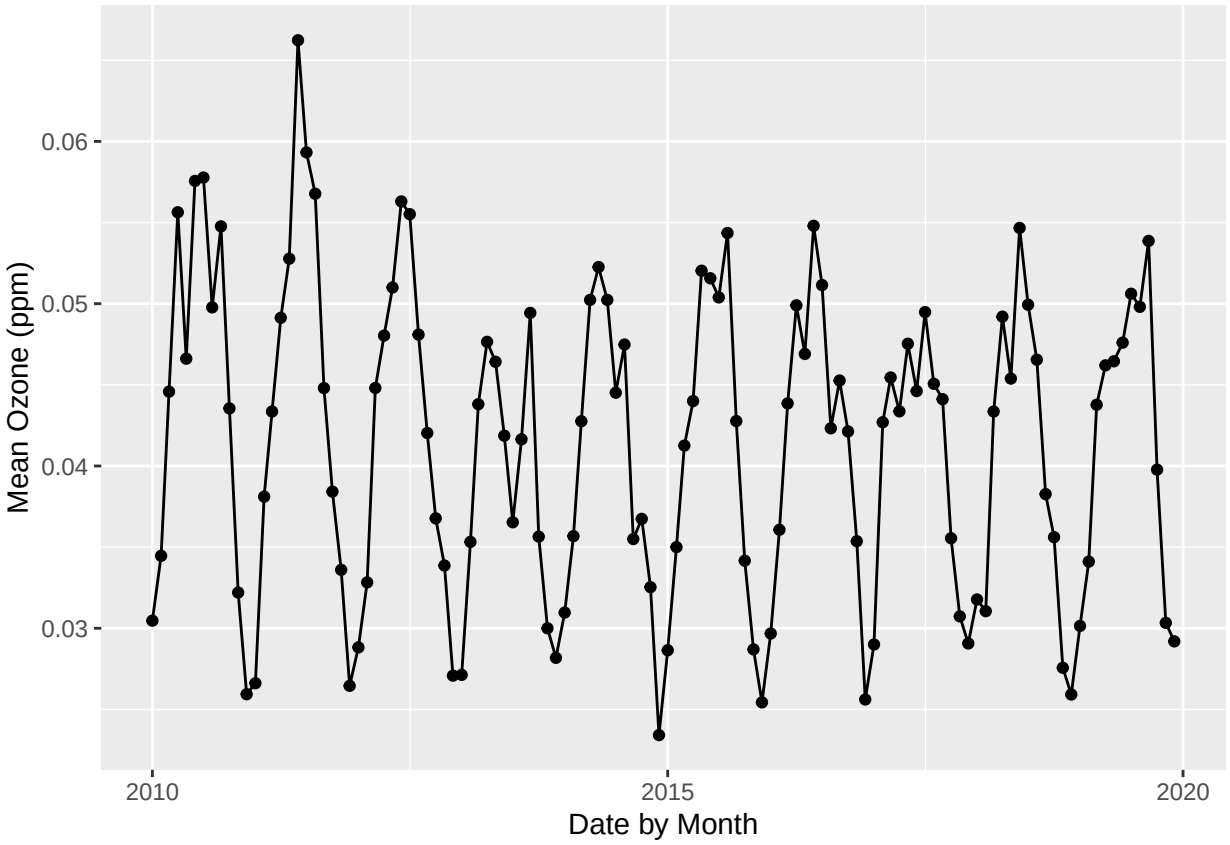
Answer: Seasonal Mann Kendall is most appropriate because there is seasonality in the data and Seasonal Mann Kendall is the only test that accommodates seasonality.

13. Create a plot depicting mean monthly ozone concentrations over time, with both a geom_point and a geom_line layer. Edit your axis labels accordingly.

```
# 13
# creating ggplot with point and line function
```

```r
monthly_ozone_plot <-
  ggplot(GaringerOzone.monthly,
         aes(x = MonthYear,
             y = MeanOzoneAggregated)) +
  geom_point() +
  geom_line()+
  ylab("Mean Ozone (ppm)") +
  xlab("Date by Month")

print(monthly_ozone_plot)
```



14. To accompany your graph, summarize your results in context of the research question. Include output from the statistical test in parentheses at the end of your sentence. Feel free to use multiple sentences in your interpretation.

    Answer: The summary of the monthly ozone trend test shows that the p value is less than 0.05, so we reject the null hypothesis and know that the data is not stationary(0.0467). Given this information, we know tha that the mean and standard deviation of the data are not constant through time, and that the ozone concentrations at this station have changed over time.

15. Subtract the seasonal component from the `GaringerOzone.monthly.ts`. Hint: Look at how we extracted the series components for the EnoDischarge on the lesson Rmd file.

16. Run the Mann Kendall test on the non-seasonal Ozone monthly series. Compare the results with the ones obtained with the Seasonal Mann Kendall on the complete series.

```r
#15
# Creating components into dataframe
```

```r
Garinger.Ozone.Components <-
  as.data.frame(GaringerOzone.monthly.decomposed$time.series[,1:3])

#Mutating dataframe to include date and nonseasonal calculation
Garinger.Ozone.Components <-
  mutate(Garinger.Ozone.Components,
           Date = GaringerOzone.monthly$MonthYear,
         Nonseasonal = Garinger.Ozone.Components$trend +
           Garinger.Ozone.Components$remainder)

#16

# Converting non-seasonal dataframe into a time series object
  GaringerOzone.nonseasonal.ts <-
  ts(Garinger.Ozone.Components$Nonseasonal,
     start = c(2010,1), frequency = 12)

# Running the Mann Kendall test on the non seasonal time series
Monthly.Ozone.Nonseasonal.trend <-
  Kendall::MannKendall(GaringerOzone.nonseasonal.ts)

# Inspecting the non seasonal time series
Monthly.Ozone.Nonseasonal.trend
```

```
## tau = -0.165, 2-sided pvalue =0.0075402
```

```r
summary(Monthly.Ozone.Nonseasonal.trend)
```

```
## Score =  -1179 , Var(Score) = 194365.7
## denominator =  7139.5
## tau = -0.165, 2-sided pvalue =0.0075402
```

Answer: The p value for the nonseasonal test is 0.007 while the p value for the data with seasonality is 0.0467, so we can still reject the null hypothesis. The p value is much lower for the nonseasonal data so we can associate that with higher confidence.