Assignment 7: Time Series Analysis

Melissa Merritt

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on time series analysis.

Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Work through the steps, **creating code and output** that fulfill each instruction.
3. Be sure to **answer the questions** in this assignment document.
4. When you have completed the assignment, **Knit** the text and code into a single PDF file.
5. After Knitting, submit the completed exercise (PDF file) to the dropbox in Sakai. Add your last name into the file name (e.g., "Fay_A07_TimeSeries.Rmd") prior to submission.

The completed exercise is due on Tuesday, March 16 at 11:59 pm.

Set up

1. Set up your session:

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

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
getwd()
```

```
## [1] "/home/guest/Documents/EDA-Fall2022"
```

```
# getwd() allows us to check the working
# directory.

library(tidyverse)
library(lubridate)
library(zoo)
library(trend)
library(scales)

# the library() function allows us to load
# the packages we will need.

mytheme <- theme_bw(base_size = 12) + theme(axis.text = element_text(color = "black"),
    legend.position = "top")
```

```
theme_set(mytheme)

# I created my theme and used theme_set() to
# establish this theme for the entire
# document.

# 2
EPAair_2010 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2010_raw.csv",
    stringsAsFactors = TRUE)
EPAair_2011 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2011_raw.csv",
    stringsAsFactors = TRUE)
EPAair_2012 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2012_raw.csv",
    stringsAsFactors = TRUE)
EPAair_2013 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2013_raw.csv",
    stringsAsFactors = TRUE)
EPAair_2014 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2014_raw.csv",
    stringsAsFactors = TRUE)
EPAair_2015 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2015_raw.csv",
    stringsAsFactors = TRUE)
EPAair_2016 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2016_raw.csv",
    stringsAsFactors = TRUE)
EPAair_2017 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2017_raw.csv",
    stringsAsFactors = TRUE)
EPAair_2018 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2018_raw.csv",
    stringsAsFactors = TRUE)
EPAair_2019 <- read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2019_raw.csv",
    stringsAsFactors = TRUE)

# We use the read.csv() function to import
# our data. Stringasfactors = TRUE allows us
# to treat the data as factors.

GaringerOzone <- rbind(EPAair_2010, EPAair_2011,
    EPAair_2012, EPAair_2013, EPAair_2014, EPAair_2015,
    EPAair_2016, EPAair_2017, EPAair_2018, EPAair_2019)

# The rbind() function allows us to bring
# all the data into one data frame. This
# works because we have the same columns for
# each data file.
```

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.

```
# 3
GaringerOzone$Date <- as.Date(GaringerOzone$Date,
    format = "%m/%d/%Y")
# as.Date() function to change the date
# column to date class.
class(GaringerOzone$Date)
```

```
## [1] "Date"
```

```
# class() to confirm the date column is now
# a date class.

# 4
GaringerOzone_W <- GaringerOzone %>%
    select(Date, Daily.Max.8.hour.Ozone.Concentration,
        DAILY_AQI_VALUE)
# The pipe function (%>%) allows us to
# select() only the three columns we need in
# our data frame.

# 5
Days <- as.data.frame(seq(as.Date("2010/01/01"),
    as.Date("2019/12/31"), "days"))
# We can use the as.data.frame() function to
# create a new data frame. The seq()
# function allows us to create a sequence of
# dates (needed to use the as.Date()
# function to have them read as dates).
# Specifying days at the end will make sure
# we have every day from 2010/01/01 to
# 2019/12/31.
colnames(Days)[1] = "Date"
# The colnames() function allows us to
# change the name of column 1 to Date.

# 6
GaringerOzone <- left_join(Days, GaringerOzone_W,
    by = "Date")
# We can use the left_join function to
# combine the two data frames. We specify to
# combine them by date, so the two date
# columns can merge leaving us with the
# correct dimensions.
```

Visualize

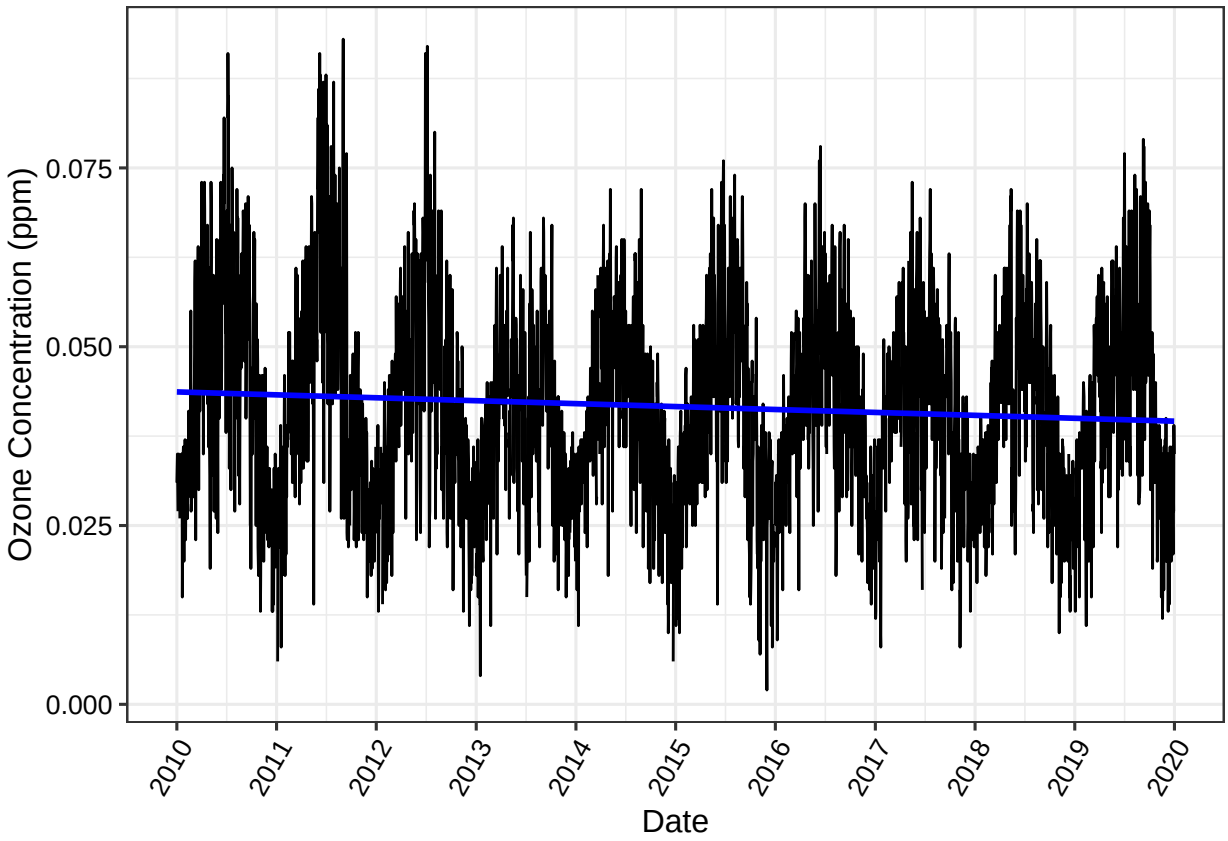
7. Create a line plot depicting ozone concentrations over time. In this case, we will plot actual concentrations in ppm, not AQI values. Format your axes accordingly. Add a smoothed line showing any linear trend of your data. Does your plot suggest a trend in ozone concentration over time?

```
# 7
OzoneConcetration_plot <- ggplot(data = GaringerOzone,
    aes(x = Date, y = Daily.Max.8.hour.Ozone.Concentration)) +
    geom_line() + scale_x_date(labels = date_format("%Y"),
```

```
    date_breaks = "1 year") + geom_smooth(method = lm,
    color = "blue", se = F) + theme(axis.text.x = element_text(angle = 60,
    hjust = 1)) + ylab("Ozone Concentration (ppm)")
print(OzoneConcetration_plot)
```

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

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




```
# To create the line plot, we can use the
# ggplot() function and specify that date is
# the x axis and onze concentration is the y
# axis. geomline() gives us the line plot.
# We use scale_x_date() to make sure each
# year is represented on the x-axis, and
# this shows the seasonality more clearly.
# The geomsmooth regression line shows the
# trend if there is one. We use the theme()
# function to change the angle of the x axis
# text to make it more readable. ylab() is
# used to change the label on the y -axis.
# We use print() to view our plot.
```

Answer: This plot suggests that there is a slight downward linear trend based on the regression line. This would suggest that ozone concentration is decreasing over time across the ten years. It is also clear that there are 10 "peaks and valleys" which could suggest yearly trends. Although the regression line is slightly downward, it is hard to make inferences about this data at this point

due to the large "peaks and valleys".

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
summary(GaringerOzone)
```

```
##       Date            Daily.Max.8.hour.Ozone.Concentration DAILY_AQI_VALUE
##  Min.   :2010-01-01   Min.   :0.00200                      Min.   :  2.00
##  1st Qu.:2012-07-01   1st Qu.:0.03200                      1st Qu.: 30.00
##  Median :2014-12-31   Median :0.04100                      Median : 38.00
##  Mean   :2014-12-31   Mean   :0.04163                      Mean   : 41.57
##  3rd Qu.:2017-07-01   3rd Qu.:0.05100                      3rd Qu.: 47.00
##  Max.   :2019-12-31   Max.   :0.09300                      Max.   :169.00
##                       NA's   :63                           NA's   :63
```

```
# Summary shows us that we have 63 NAs in
# the data frame.

GaringerOzone$Daily.Max.8.hour.Ozone.Concentration <- na.approx(GaringerOzone$Daily.Max.8.hour.Ozone.Co
# we can use the na.approx() function from
# the zoo package to interpolate the missing
# daily data for ozone concentration.
```

Answer: We used a linear interpolation because the data we are using appears relatively linear between the peaks and valleys, so this interpolation method will "connect the dots" between the data points. The process of "connecting the dots" will give us values inbetween the different ozone concentration data points where values are missing.

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
GaringerOzone.monthly <- GaringerOzone %>%
    mutate(Year = year(Date), Month = month(Date)) %>%
    group_by(Year, Month) %>%
    summarise(Mean_ozone_concentration = mean(Daily.Max.8.hour.Ozone.Concentration))
```

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

```
# Using the pipe function (%>%) I was able
# to mutate() and add the year and month
# columns using the year() and month()
# functions. Then we can use the group_by()
# function to specify that we want to use
# the rows that have the same values in
# month and year column for the means. We
# use the summarise() function to then get
# the mean.
```

```
GaringerOzone.monthly <- GaringerOzone.monthly %>%
    mutate(FirstDay = my(paste0(Month, "-", Year)))

class(GaringerOzone.monthly$FirstDay)
```

```
## [1] "Date"
```

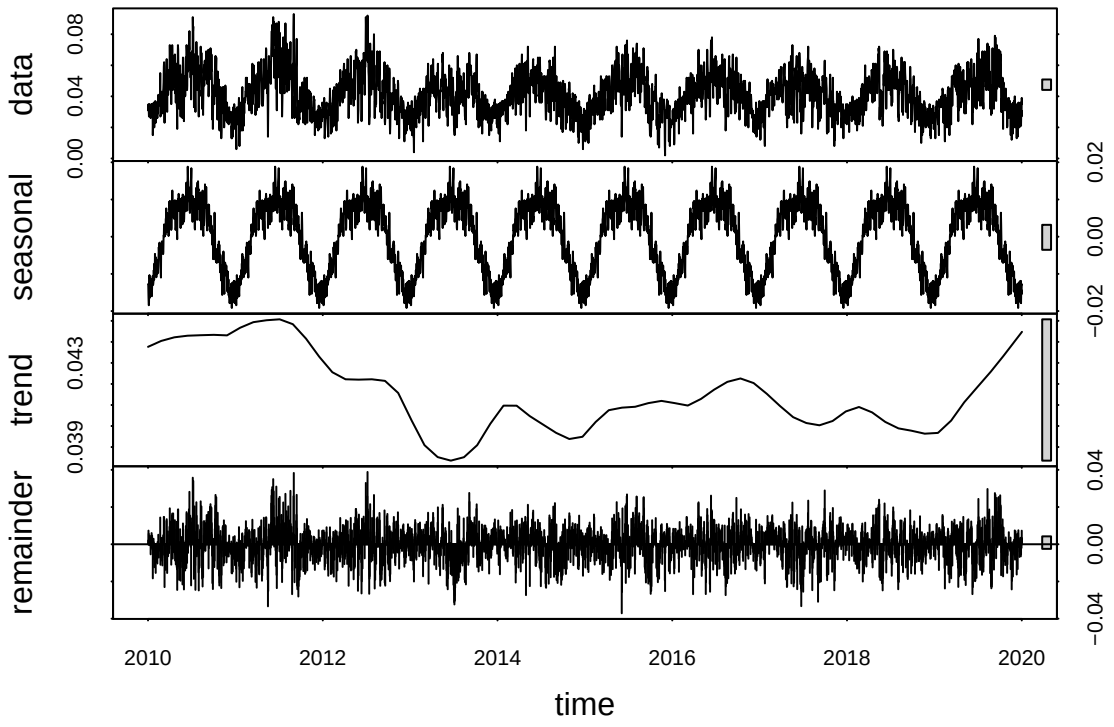
```
# In the same data frame we create a new
# column using the mutate() function, that
# has the date organized as the first day of
# each month. The my() function is from the
# lubridate package, and extracts the month
# and year to create the format we want for
# graphing.
```

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.

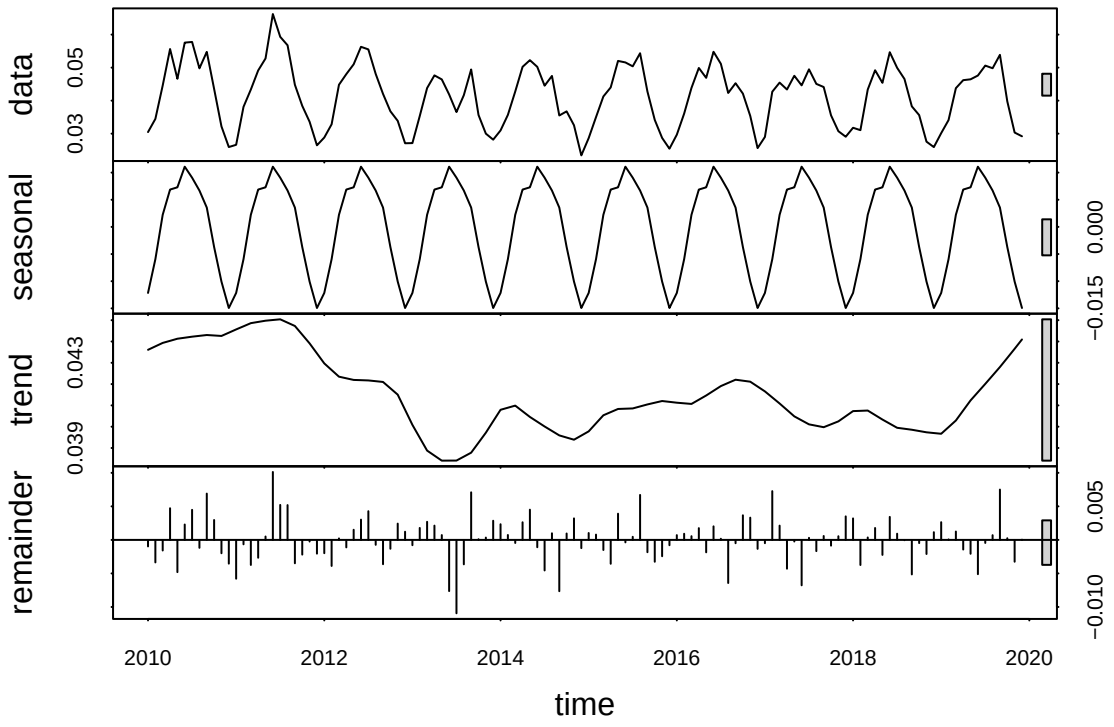
```
# 10
GaringerOzone.daily.ts <- ts(GaringerOzone$Daily.Max.8.hour.Ozone.Concentration,
    start = c(2010, 1), frequency = 365)

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

# To create the time series focusing on
# ozone concentration we can use the ts()
# function. Within the function we specify
# the start and end dates along with the
# frequency. For the frequency of the daily
# time series, we use 365 and we use 12 for
# the month frequency.
```

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

```
# 11
daily.decomp <- stl(GaringerOzone.daily.ts, s.window = "periodic")
monthly.decomp <- stl(GaringerOzone.monthly.ts,
    s.window = "periodic")

# To decompose the data we use the stl()
# functions with the argument s.window =
# 'periodic'

plot(daily.decomp)
```




```
plot(monthly.decomp)
```




```
# Using the plot function we can plot the
# decomposition which allows us to see the
# variability in the remainder, trend,
# seasonal, data.
```

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

GaringerOzone_monthly_trend <- Kendall::SeasonalMannKendall(GaringerOzone.monthly.ts)
summary(GaringerOzone_monthly_trend)
```

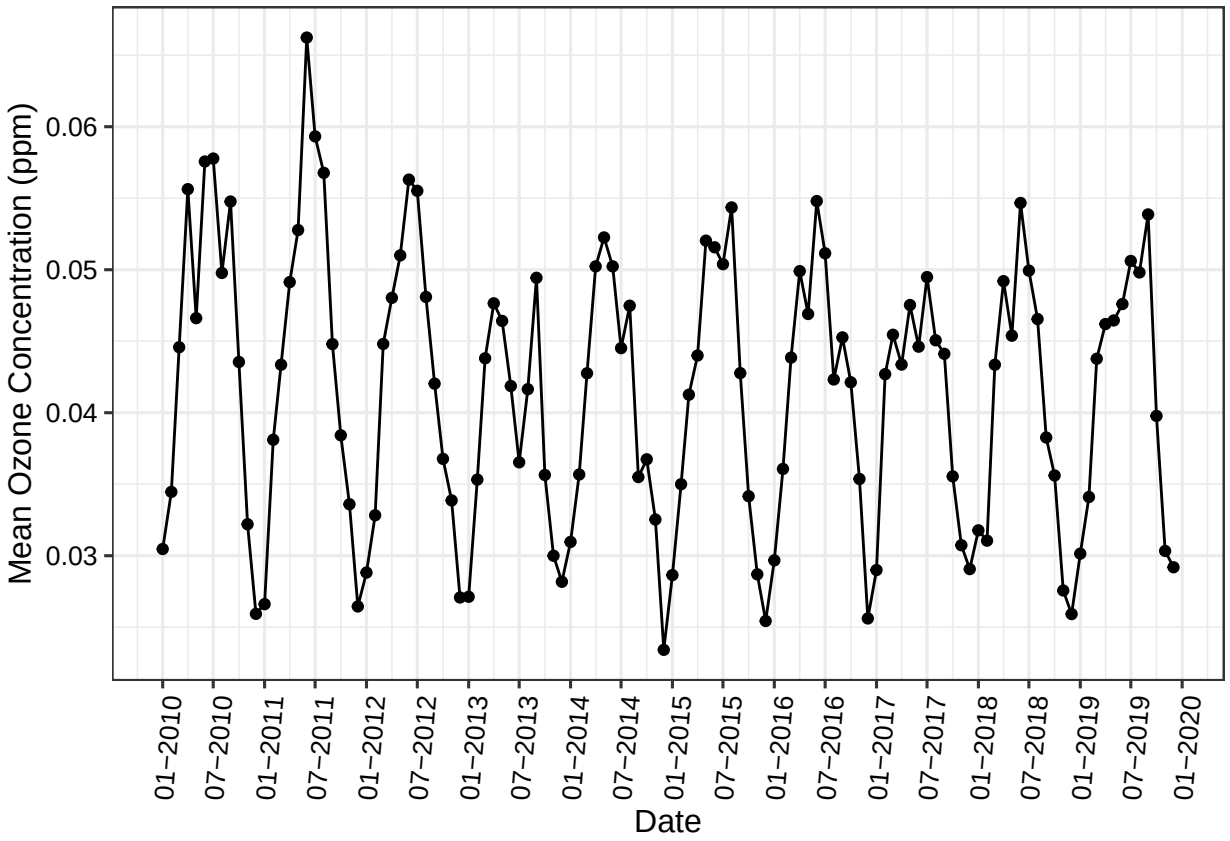
```
## Score =  -77 , Var(Score) = 1499
## denominator =  539.4972
## tau = -0.143, 2-sided pvalue =0.046724
```

```
# We can use the seasonalMannKendall()
# function with the time series for the
# monthly data. We can then get the summary
# of the trend to observe the p-value to see
# if there is a statistical trend.
```

Answer: The seasonal Mann-Kendall is the most appropriate test because there is clear seasonality in the data throughout each year. You can see in the plot of the decomposition that seasonality explains a large portion of variability of the data.

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

Mean_monthly_ozone_plot <- ggplot(GaringerOzone.monthly,
    aes(x = FirstDay, y = Mean_ozone_concentration)) +
    geom_point() + geom_line() + scale_x_date(labels = date_format("%m-%Y"),
    date_breaks = "6 months") + theme(axis.text.x = element_text(angle = 85,
    hjust = 1)) + ylab("Mean Ozone Concentration (ppm)") +
    xlab("Date")
print(Mean_monthly_ozone_plot)
```




```
# To create the plot we use the ggplot()
# function with the Garinger monthly data.
# We specify that we want the mean ozone on
# the y axis, and the Date on the x axis. We
# can use the geom_point() and geom_line()
# functions to plot both the points and the
# line. The scale_x_date() function allows
# us to create 6 month breaks in the
# labeling on the x axis, and this allows us
# to see how the data changes throughout
# each year. The theme() function allows us
# to angle the date labels. The xlab() and
# ylab() allow us to label the axes, and the
# print() function allows us to view our
# plots.
```

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 research question asked if ozone concentrations have changed over the 2010s. Looking at the graph, there does appear to be less variability as we approach the end of 2019. The highs in the summer months appear to decrease, but there is not a noticeable change in the monthly mean ozone concentrations on the graph. (The monotonic trend analysis will tell us whether there is an overall change while accounting for the seasonality, and the seasonal Mann Kendall 2-sided p-value was 0.046724, showing only slight statistical significance).

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.

```
# 15
GaringerOzone.monthly_sub <- as.data.frame(monthly.decomp$time.series[,
    2:3])

Monthly.Garinger.nonseasonal.components <- GaringerOzone.monthly_sub %>%
    mutate(GaringerOzone.monthly_sub, Data = (GaringerOzone.monthly_sub$trend +
        GaringerOzone.monthly_sub$remainder),
        Date = GaringerOzone.monthly$FirstDay) %>%
    select(Data:Date)
view(Monthly.Garinger.nonseasonal.components)

Monthly.Garinger.nonseasonal.components.ts <- ts(Monthly.Garinger.nonseasonal.components$Data,
    start = c(2010, 1), frequency = 12)

# We can use the as.data.frame() to create a
# data frame with the observed and the
# remainder columns. Using that data frame
# we can use the pipe function (%>%) to
# specify that we only want the data (trend
# plus remainder) and the date in our data
# frame to get rid of the seasonal
# component. We used the view() function to
# ensure the columns are correct. We then
# can run a time series analysis using the
# ts() function.

# 16
Nonseasonal.monthly.trend <- Kendall::MannKendall(Monthly.Garinger.nonseasonal.components.ts)
summary(Nonseasonal.monthly.trend)
```

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

```
# With our new time series that does not
# include the seasonal component we can run
# the nonseasonal mann kendall test to get a
# new 2-sided p-value.
```

Answer: The p-value from the test without the seasonal component is 0.0075402, which is smaller

than the p-value in the seasonal mann kendall. These results from the monotonic trend analysis tell us that the changes in the data are more statistically significant when we remove the seasonal component from our analysis. It also tells us that there is an overall statistically significant changing trend in the data in the 2010s.