Assignment 7: Time Series Analysis

Megan Lundequam

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 Monday, March 14 at 7:00 pm.

Set up

1. Set up your session:

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

```
#1
getwd()
```

```
## [1] "/Users/meganlundequam/Desktop/Spring 2022/Environmental Data Analytics/Git/Environmental_Data_A
```

```
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v stringr 1.4.0
## v tidyr   1.1.4     v forcats 0.5.1
## v readr   2.1.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()   masks plyr::arrange()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::mutate()    masks plyr::mutate()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(trend)
library(zoo)
```

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

```
library(Kendall)
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
mytheme <- theme_classic(base_size = 14) +
  theme(axis.text = element_text(color = "black"),
        legend.position = "bottom")
theme_set(mytheme)
```

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.

```
#2
GaringerOzoneFiles = list.files(path = "../Data/Raw/Ozone_TimeSeries",
                                pattern="*.csv", full.names=TRUE)
```

```
GaringerOzoneFiles
```

```
##  [1] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2010_raw.csv"
##  [2] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2011_raw.csv"
##  [3] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2012_raw.csv"
##  [4] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2013_raw.csv"
##  [5] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2014_raw.csv"
##  [6] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2015_raw.csv"
##  [7] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2016_raw.csv"
##  [8] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2017_raw.csv"
##  [9] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2018_raw.csv"
## [10] "../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2019_raw.csv"
```

```
GaringerOzone <- GaringerOzoneFiles %>%
  ldply(read.csv)
```

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")

# 4
GaringerOzoneWrangled <-
  GaringerOzone %>%
  select(Date, Daily.Max.8.hour.Ozone.Concentration, DAILY_AQI_VALUE)

# 5
Days <-
  as.data.frame(seq(as.Date("2010-01-01"), as.Date("2019-12-31"), by = "day"))

Days <- data.frame("Date"=seq(as.Date("2010-01-01"), as.Date("2019-12-31"), by = "day"))

# 6
GaringerOzone <- left_join(Days, GaringerOzoneWrangled)
```

```
## Joining, by = "Date"
```

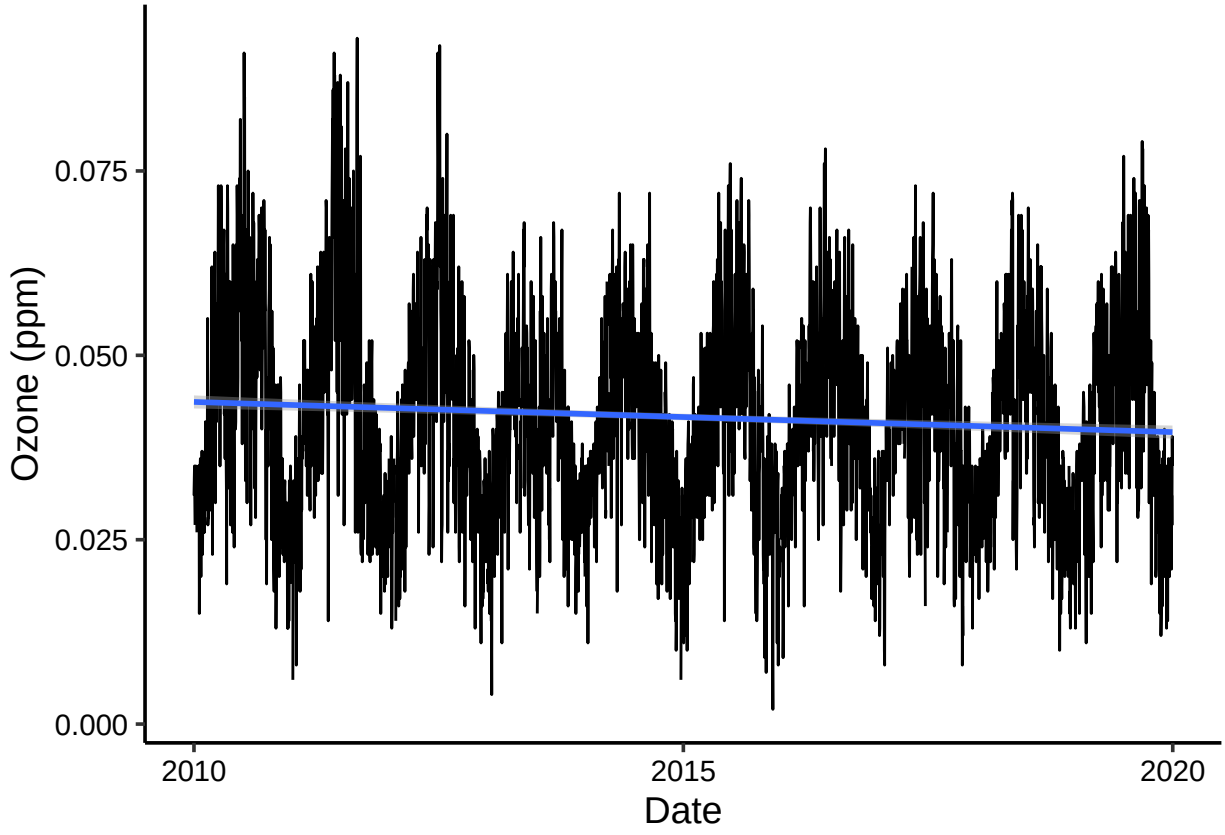
Visualize

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?

```
#7
ggplot(GaringerOzone, aes(x = Date, y = Daily.Max.8.hour.Ozone.Concentration)) +
  geom_line() +
  geom_smooth(method = "lm") +
  ylab("Ozone (ppm)")
```

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

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



Answer: Yes. The plot shows annual fluctuations but the trendline displays a slight decrease over time.

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
head(GaringerOzone)
```

```
##         Date Daily.Max.8.hour.Ozone.Concentration DAILY_AQI_VALUE
## 1 2010-01-01                                0.031              29
## 2 2010-01-02                                0.033              31
## 3 2010-01-03                                0.035              32
## 4 2010-01-04                                0.031              29
```

```
## 5 2010-01-05                                0.027             25
## 6 2010-01-06                                   NA             NA
```

```
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
```

```
# Adding new column with no missing obs
GaringerOzone_clean <-
  GaringerOzone %>%
  mutate( Ozone.clean = zoo::na.approx(Daily.Max.8.hour.Ozone.Concentration) )

summary(GaringerOzone_clean$Ozone.clean)
```

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

Answer:
We used a linear interpolation to fill missing daily data because we simply wanted to connect the dots. The concentrations follow a strong trend so it is safe to assume that the missing data falls between the previous and following measurement. We did not have reason to assume that the missing data would be equal to the measurement nearest to that date as piecewise constant interpolation would produce, nor did we have data that followed a quadratic function which is what spline uses to interpolate, so we would not use either of those methods for this interpolation.

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_clean %>%
  mutate(Year = year(Date)) %>%
  mutate(Month = month(Date))

GaringerOzone.monthly$Month.Year <-
  floor_date(GaringerOzone.monthly$Date, "month")

GaringerOzone.monthly <-
  GaringerOzone.monthly %>%
  group_by(Month.Year) %>%
  dplyr::summarise(MeanOzone = mean(Ozone.clean)) %>%
  as.data.frame()

f_month_monthly <- month(first(GaringerOzone.monthly$Month.Year))
f_year_monthly <- year(first(GaringerOzone.monthly$Month.Year))
```

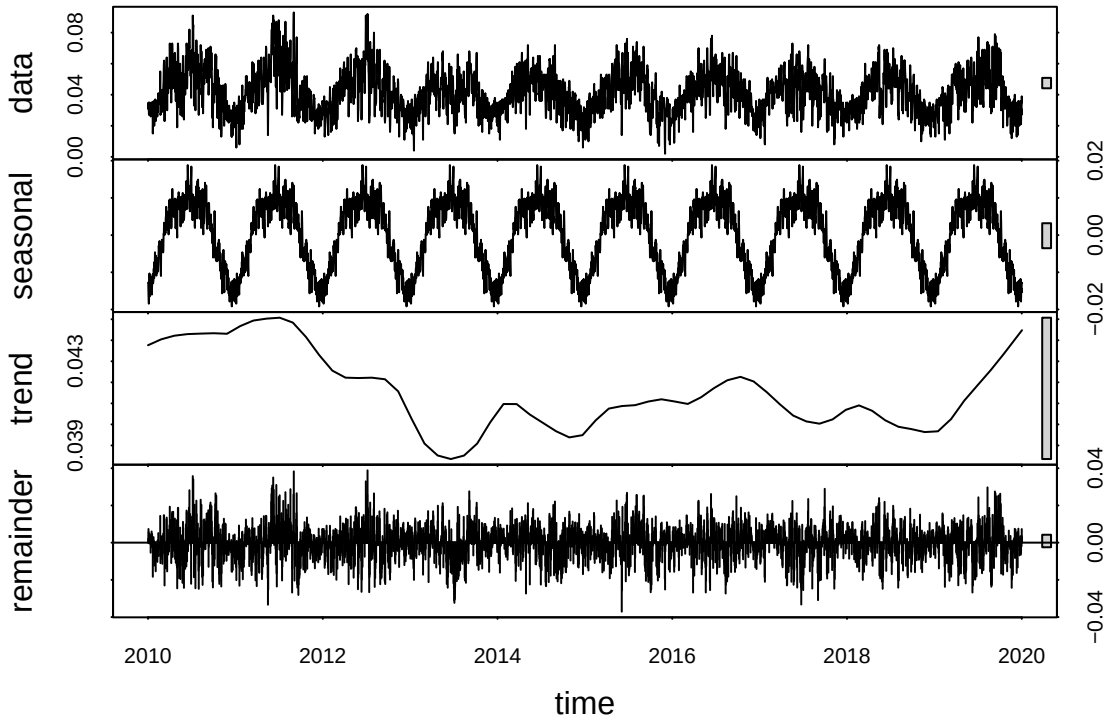
10. Generate two time series objects. Name the first `GaringerOzone.daily.ts` and base it on the dataframe of daily observations. Name the second `GaringerOzone.monthly.ts` and base it on the monthly average ozone values. Be sure that each specifies the correct start and end dates and the frequency of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

```
#10
f_month_daily <- month(first(GaringerOzone_clean$Date))
f_year_daily <- year(first(GaringerOzone_clean$Date))
```

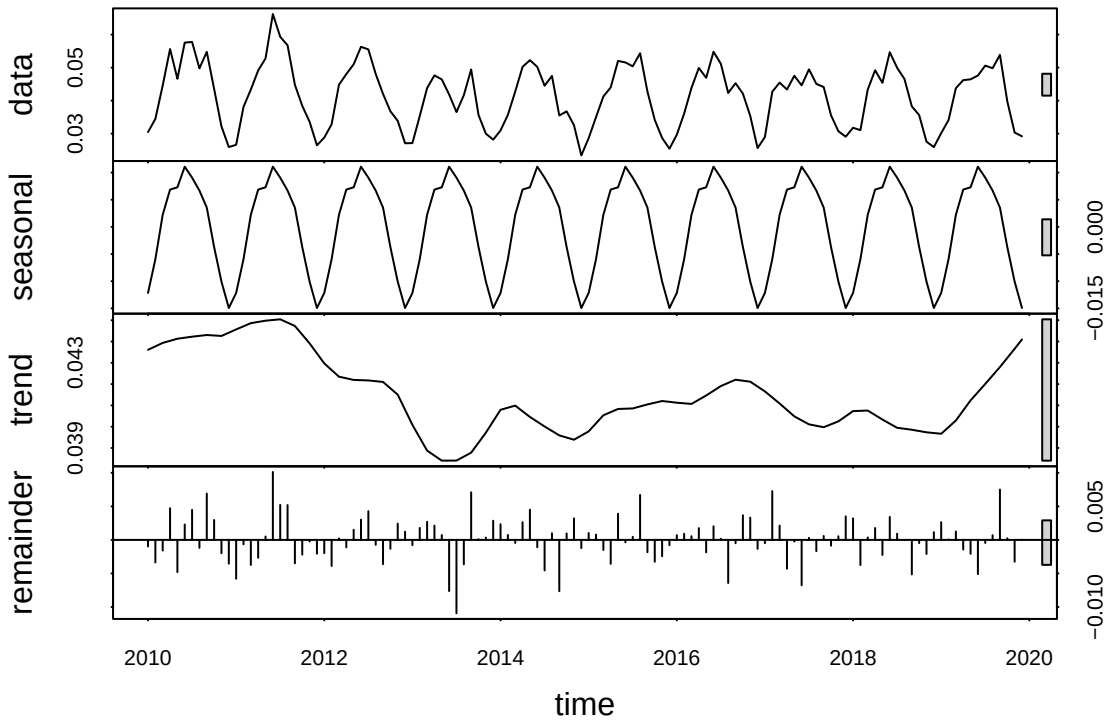
```
GaringerOzone.daily.ts <- ts(GaringerOzone_clean$Ozone.clean,
                             start = c(f_year_daily,f_month_daily),
                             frequency = 365)

GaringerOzone.monthly.ts <- ts(GaringerOzone.monthly$MeanOzone,
                   start=c(f_year_monthly,f_month_monthly),
                   frequency=12)
```

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

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




```
monthly_data_decomp <- stl(GaringerOzone.monthly.ts,s.window = "periodic")
plot(monthly_data_decomp)
```



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
monthly_data_trend1 <- Kendall::SeasonalMannKendall(GaringerOzone.monthly.ts)
# Inspect results
monthly_data_trend1
```

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

```
summary(monthly_data_trend1)
```

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

```
monthly_data_trend2 <- trend::smk.test(GaringerOzone.monthly.ts)
# Inspect results
monthly_data_trend2
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  GaringerOzone.monthly.ts
## z = -1.963, p-value = 0.04965
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##    S varS
```

```
##  -77 1499
```

```
summary(monthly_data_trend2)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data: GaringerOzone.monthly.ts
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                      S varS    tau      z Pr(>|z|)
## Season 1:   S = 0   15  125  0.333  1.252  0.21050
## Season 2:   S = 0   -1  125 -0.022  0.000  1.00000
## Season 3:   S = 0   -4  124 -0.090 -0.269  0.78762
## Season 4:   S = 0  -17  125 -0.378 -1.431  0.15241
## Season 5:   S = 0  -15  125 -0.333 -1.252  0.21050
## Season 6:   S = 0  -17  125 -0.378 -1.431  0.15241
## Season 7:   S = 0  -11  125 -0.244 -0.894  0.37109
## Season 8:   S = 0   -7  125 -0.156 -0.537  0.59151
## Season 9:   S = 0   -5  125 -0.111 -0.358  0.72051
## Season 10:   S = 0 -13  125 -0.289 -1.073  0.28313
## Season 11:   S = 0 -13  125 -0.289 -1.073  0.28313
## Season 12:   S = 0  11  125  0.244  0.894  0.37109
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Answer: The seasonal Mann-Kendall analysis is most appropriate because it looks at each season of the year and determines whether each season has a trend. Our data is clearly displaying a seasonal trend as we see regular fluctations at regular periods within each year.

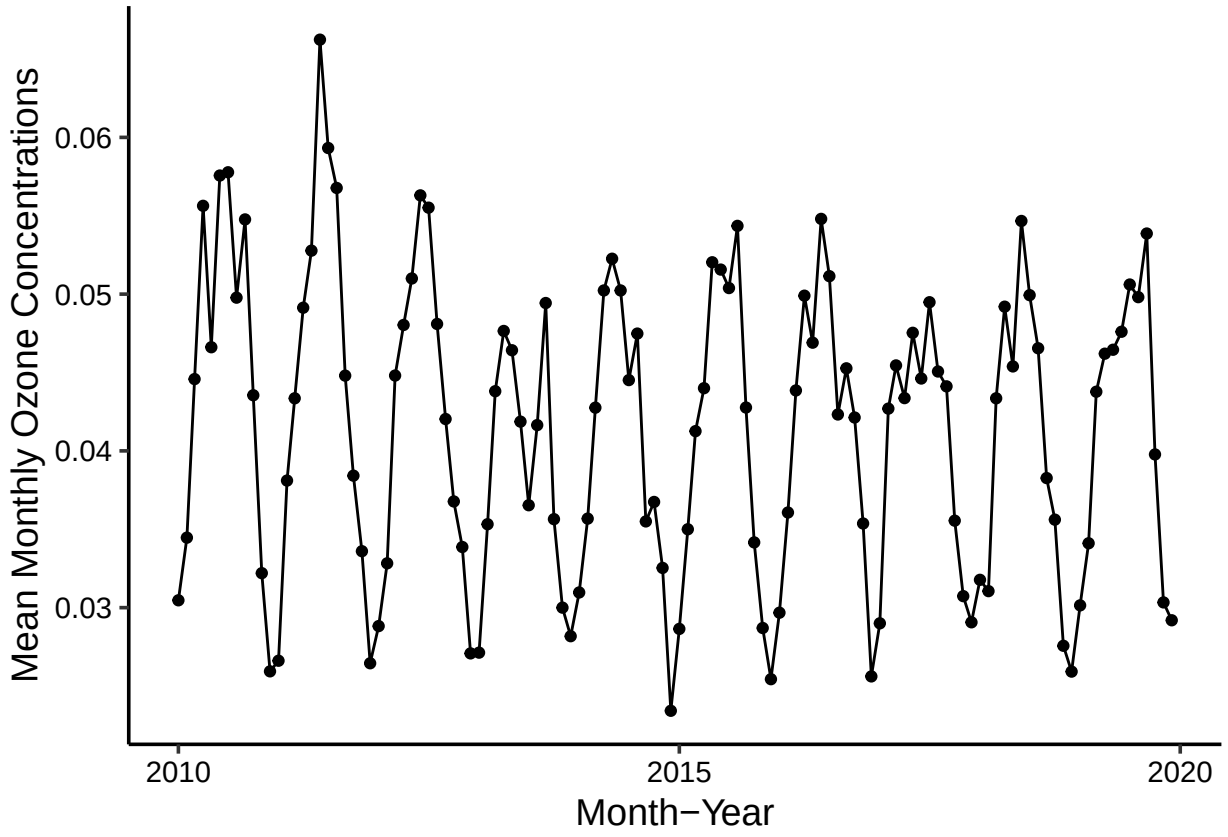
13. Create a plot depicting mean monthly ozone concentrations over time, with both a geom_point and a geom_line layer. Edit your axis labels accordingly.

```
# 13
Monthly_Ozone_data_plot <-
ggplot(GaringerOzone.monthly, aes(x = Month.Year, y = MeanOzone)) +
  geom_point() +
  geom_line() +
  ylab("Mean Monthly Ozone Concentrations") +
  xlab("Month-Year")
  geom_smooth( method = lm )
```

```
## geom_smooth: na.rm = FALSE, orientation = NA, se = TRUE
## stat_smooth: na.rm = FALSE, orientation = NA, se = TRUE, method = function (formula, data, subset, w
## {
##     ret.x <- x
##     ret.y <- y
##     cl <- match.call()
##     mf <- match.call(expand.dots = FALSE)
##     m <- match(c("formula", "data", "subset", "weights", "na.action", "offset"), names(mf), 0)
##     mf <- mf[c(1, m)]
##     mf$drop.unused.levels <- TRUE
##     mf[[1]] <- quote(stats::model.frame)
```

```
##     mf <- eval(mf, parent.frame())
##     if (method == "model.frame")
##         return(mf)
##     else if (method != "qr")
##         warning(gettextf("method = '%s' is not supported. Using 'qr'", method), domain = NA)
##     mt <- attr(mf, "terms")
##     y <- model.response(mf, "numeric")
##     w <- as.vector(model.weights(mf))
##     if (!is.null(w) && !is.numeric(w))
##         stop("'weights' must be a numeric vector")
##     offset <- model.offset(mf)
##     mlm <- is.matrix(y)
##     ny <- if (mlm)
##         nrow(y)
##     else length(y)
##     if (!is.null(offset)) {
##         if (!mlm)
##             offset <- as.vector(offset)
##         if (NROW(offset) != ny)
##             stop(gettextf("number of offsets is %d, should equal %d (number of observations)", NROW(
##     }
##     if (is.empty.model(mt)) {
##         x <- NULL
##         z <- list(coefficients = if (mlm) matrix(NA, 0, ncol(y)) else numeric(), residuals = y, fitt
##         if (!is.null(offset)) {
##             z$fitted.values <- offset
##             z$residuals <- y - offset
##         }
##     }
##     else {
##         x <- model.matrix(mt, mf, contrasts)
##         z <- if (is.null(w))
##             lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
##         else lm.wfit(x, y, w, offset = offset, singular.ok = singular.ok, ...)
##     }
##     class(z) <- c(if (mlm) "mlm", "lm")
##     z$na.action <- attr(mf, "na.action")
##     z$offset <- offset
##     z$contrasts <- attr(x, "contrasts")
##     z$xlevels <- .getXlevels(mt, mf)
##     z$call <- cl
##     z$terms <- mt
##     if (model)
##         z$model <- mf
##     if (ret.x)
##         z$x <- x
##     if (ret.y)
##         z$y <- y
##     if (!qr)
##         z$qr <- NULL
##     z
## }
## position_identity
```

```
print(Monthly_Ozone_data_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.

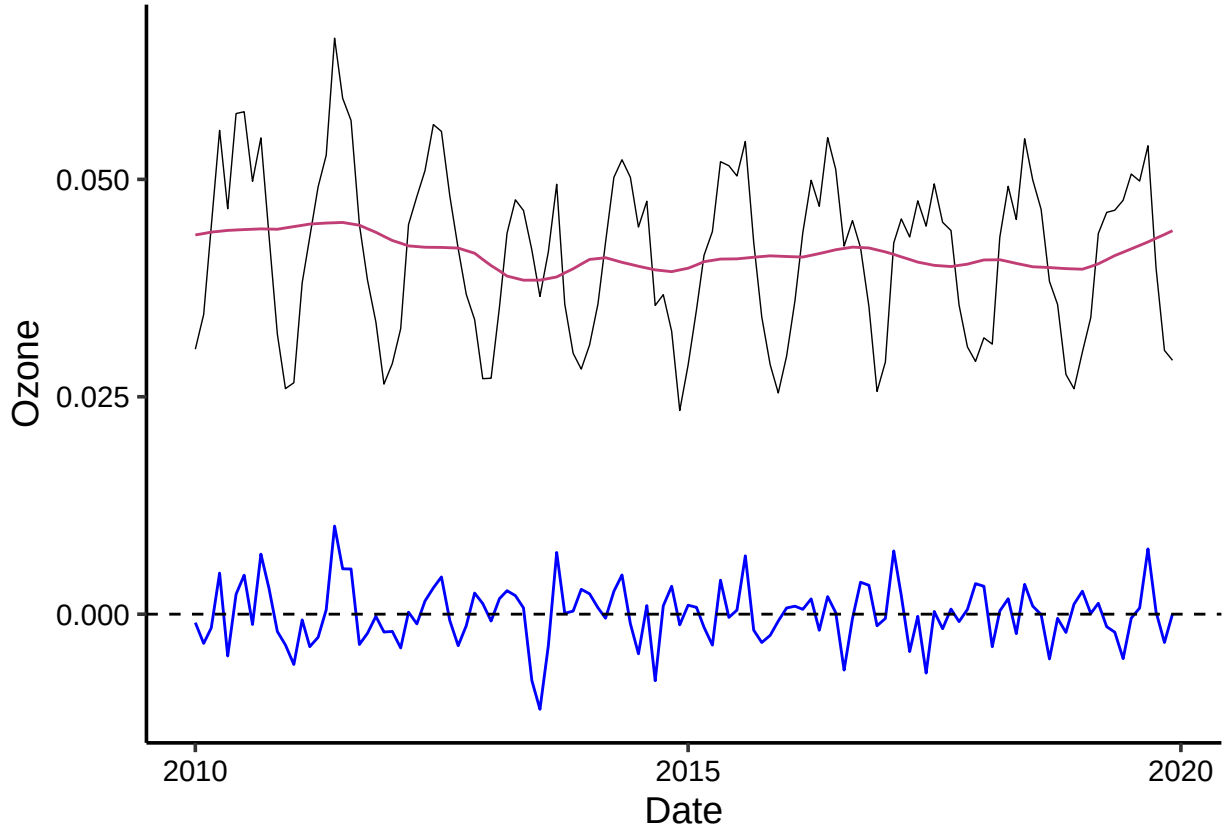
    Answer: Study question: Have ozone concentrations changed over the 2010s at this station? Based on the results from the Mann-Kendall test, we can say with some confidence that this data displays an overall monotonic trend (p-value = 0.04965) with y-values increasing and decreasing with some consistency. Based on the negative scores in each season, we can conclude that ozone concentrations have changed over the 2010s at this station.

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
# We can extract the components and turn them into data frames
monthly_data_decomp_df <- as.data.frame(monthly_data_decomp$time.series[,2:3])

# Visualization
monthly_data_decomp_df <- mutate(monthly_data_decomp_df,
        Observed = GaringerOzone.monthly$MeanOzone,
        Date = GaringerOzone.monthly$Month.Year)

ggplot(monthly_data_decomp_df) +
```

```
  geom_line(aes(y = Observed, x = Date),  size = 0.25) +
  geom_line(aes(y = trend, x = Date), color = "#c13d75ff") +
  geom_line(aes(y = remainder, x = Date), color = "blue") +
  geom_hline(yintercept = 0, lty = 2) +
  ylab(expression("Ozone"))
```




```
#16
GaringerOzone.monthly.NS.ts <- ts(monthly_data_decomp_df$Observed,
                   start=c(f_year_monthly,f_month_monthly),
                   frequency=12)
monthly_data_trend_NS <- Kendall::SeasonalMannKendall(GaringerOzone.monthly.NS.ts)
# Inspect results
monthly_data_trend_NS
```

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

```
summary(monthly_data_trend_NS)
```

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

Answer: Non seasonal results: Score = -77 , Var(Score) = 1499 denominator = 539.4972 tau = -0.143, 2-sided pvalue =0.046724 Seasonal results: Score = -77 , Var(Score) = 1499 denominator = 539.4972 tau = -0.143, 2-sided pvalue =0.046724 They are identical!