Assignment 7: Time Series Analysis

Enikoe Bihari

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

```r
#1

# check working directory
getwd()
```

```
## [1] "C:/Users/eniko/Documents/Duuuuuke/2021-22/Data Analytics/Environmental_Data_Analytics_2022/Assig
```

```r
# load packages and csv
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(zoo)
```

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

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```
library(trend)
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

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

## The following object is masked from 'package:purrr':
##
##     compact
```

```
library(dplyr)
library(Kendall)

# create a theme with gray defaults
theme1 <- theme_gray(base_size = 12) +
  theme(axis.text = element_text(color = "grey50"),
        legend.position = "top",
        axis.title = element_text(color = "grey20"),
        legend.key.width = unit(2, "cm"))

# set it as the default theme
theme_set(theme1)
```

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

# read in the files and create one single data frame
ozone.csv = list.files(path="../Data/Raw/Ozone_TimeSeries/", pattern="*.csv", full.names=TRUE)
# ozone.csv
```

```r
GaringerOzone = ldply(ozone.csv, read_csv)
# GaringerOzone
# class(GaringerOzone)
```

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

# Set date to date format
GaringerOzone$Date <- as.Date(GaringerOzone$Date , format = "%m/%d/%Y")

# 4

# replace ozone column name with a shorter name
GaringerOzone$Ozone = GaringerOzone$"Daily Max 8-hour Ozone Concentration"

# select the correct columns
GaringerOzone = GaringerOzone %>%
  select(Date, Ozone, DAILY_AQI_VALUE)

# 5

# create a data frame filled in with all the missing days
first.date = first(GaringerOzone$Date)
last.date = last(GaringerOzone$Date)
full.dates = data.frame(Date = seq(from = first.date, to = last.date, by = 1))

# 6

# replace the old date column with the new filled-in date column
GaringerOzone = left_join(full.dates, GaringerOzone)
```

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

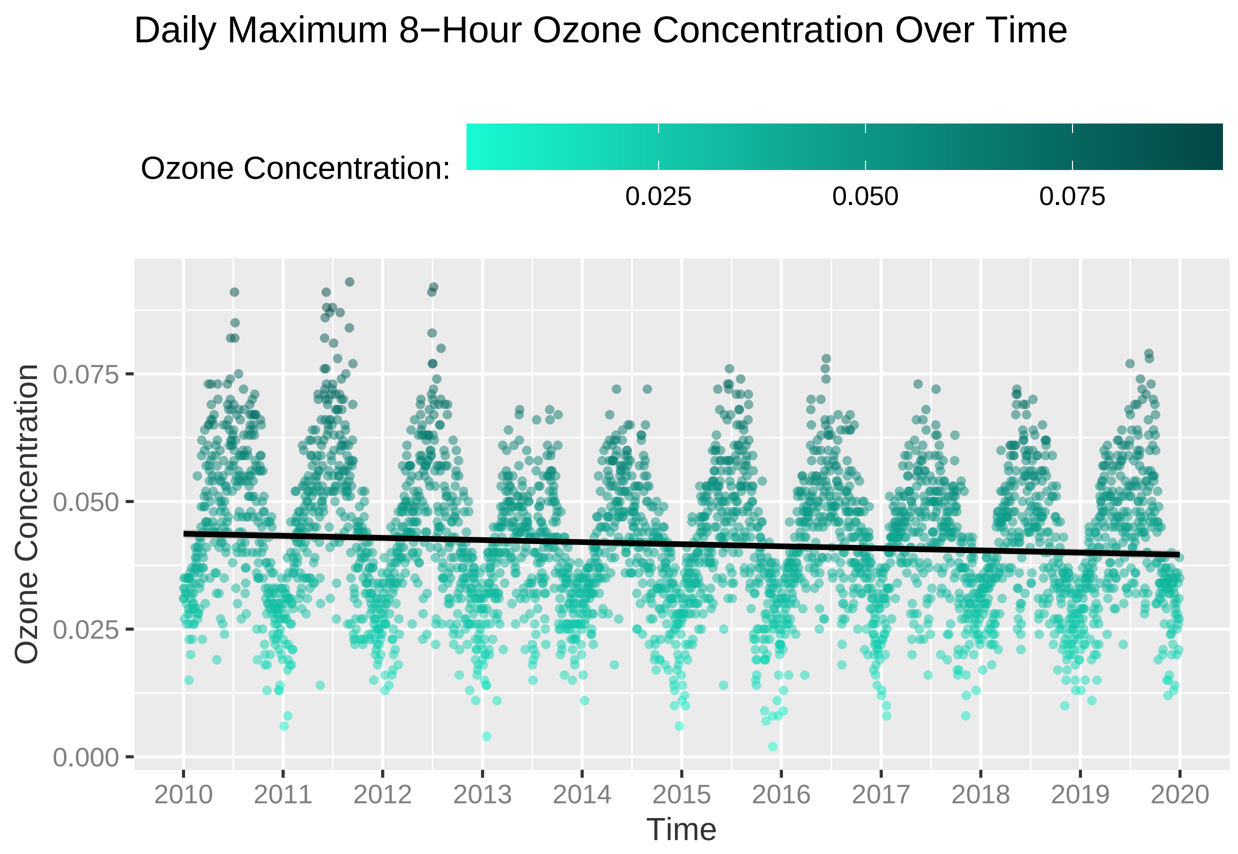
Visualize

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?

```r
#7
```

```r
# create a plot of the data points
ozone <-
ggplot(GaringerOzone) +
  geom_point(aes(x=Date, y=Ozone, color=Ozone),
             size = 1,
             alpha = 0.5) +
  geom_smooth(aes(x=Date, y=Ozone),
              method = lm,
              size = 1,
              color = 'black',
              alpha = 0.4) +
  scale_colour_gradient(low = "#18fbd2", high = "#034746") +
  ylab(expression("Ozone Concentration")) +
  xlab(expression("Time")) +
  ggtitle("Daily Maximum 8-Hour Ozone Concentration Over Time\n") +
  labs(color = "Ozone Concentration:")+
  scale_x_date(date_breaks = "years" , date_labels = "%Y")

print(ozone)
```

```
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 63 rows containing non-finite values (stat_smooth).
## Warning: Removed 63 rows containing missing values (geom_point).
```



Answer: Yes, there seems to be a trend of decreasing ozone concentration over time.

However, there also seems to be a pretty drastic seasonal component to the data, with peaks and valleys occuring in ozone concentration on a yearly basis.

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

# interpolate the missing values
GaringerOzone$Ozone = zoo::na.approx(GaringerOzone$Ozone)
```

Answer: We are only missing some individual days from out observations, so we don't need to split up the data into different sections like you do when you have large gaps. We can assume our short linear interpolations between points will be pretty accurate (the daily fluctuation is not drastic, so we can pretty safely assume that a missing data point between two adjacent days will be somewhere between the values measured on those two days). There is also no reason to believe that the data would follow a quadratic function in these short gaps, since our other data points seem to indicate that directly adjacent observations share roughly linear trends with each other.

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

# create a new data set with just mean monthly ozone levels
GaringerOzone.monthly = GaringerOzone %>%
  mutate(Month = month(Date)) %>%
  mutate(Year = year(Date)) %>%
  group_by(Year, Month) %>%
  # summarise_at(vars(Ozone, DAILY_AQI_VALUE, Date), funs(mean(Ozone, DAILY_AQI_VALUE), min(Date)))
  dplyr::summarise(Ozone.monthly = mean(Ozone), Date = first(Date))
```

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

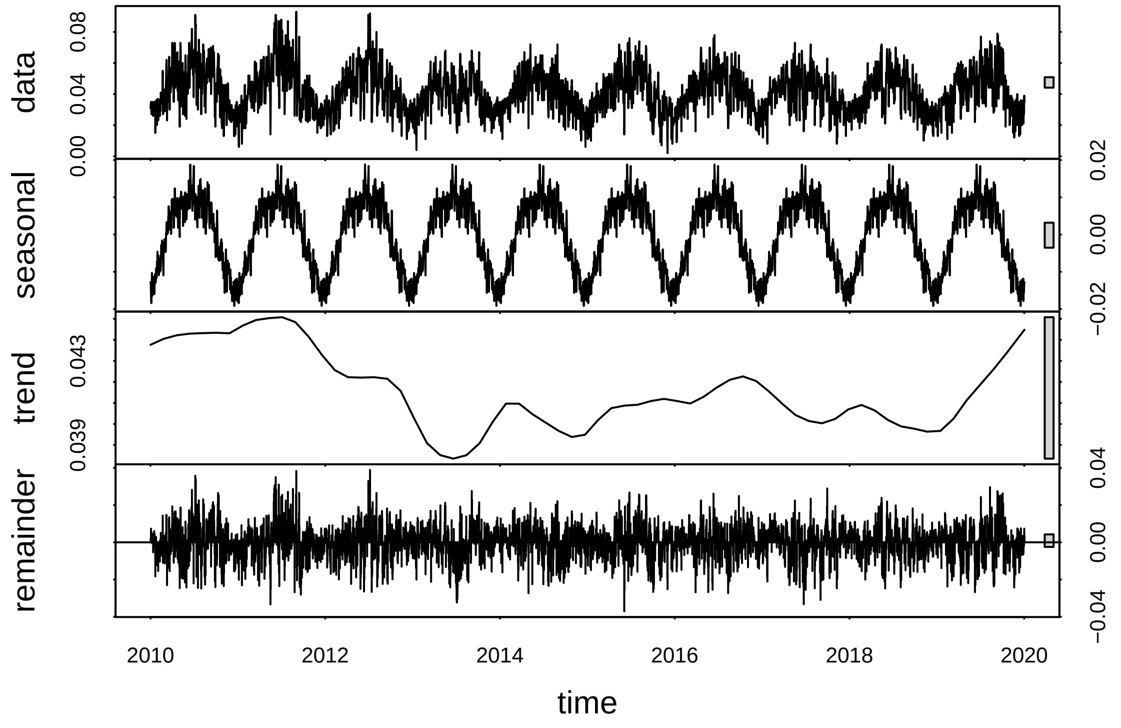
10. Generate two time series objects. Name the first `GaringerOzone.daily.ts` and base it on the dataframe of daily observations. Name the second `GaringerOzone.monthly.ts` and base it on the monthly average ozone values. Be sure that each specifies the correct start and end dates and the frequency of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

```
#10

# create daily time series
f_day <- day(first(GaringerOzone$Date))
f_month <- month(first(GaringerOzone$Date))
f_year <- year(first(GaringerOzone$Date))
GaringerOzone.daily.ts <- ts(GaringerOzone$Ozone,
                   start=c(f_year,f_month, f_day),
                   frequency=365)
```

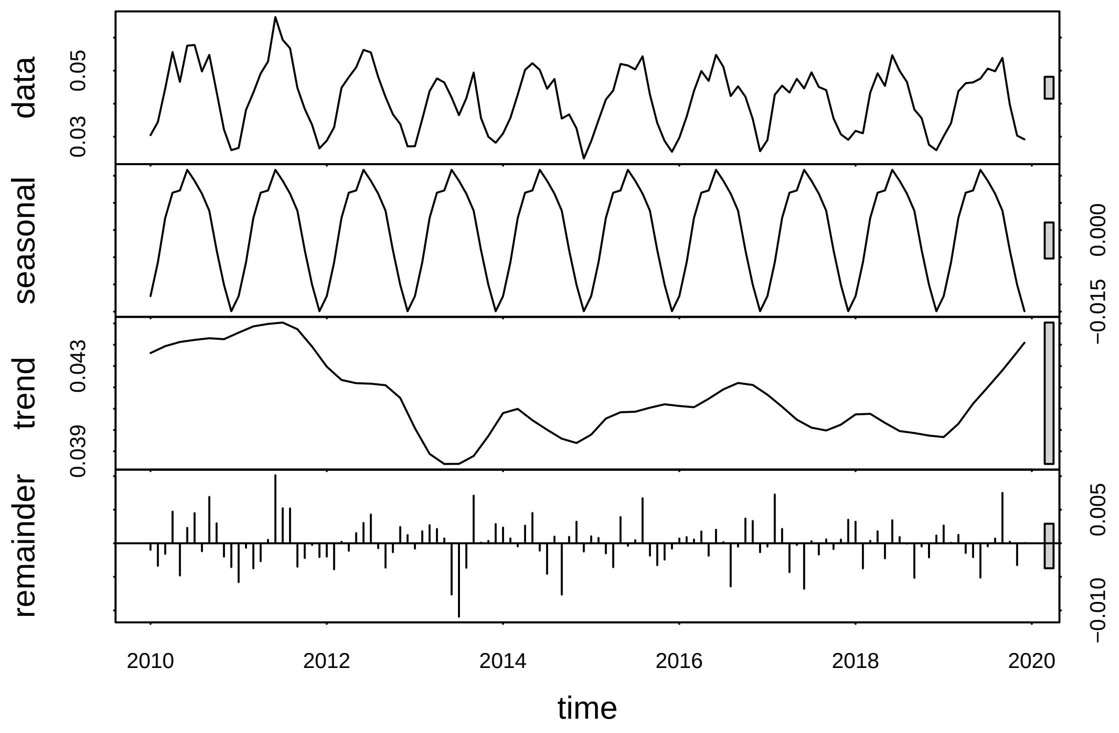
```r
# create monthly time series
f_month <- month(first(GaringerOzone.monthly$Date))
f_year <- year(first(GaringerOzone.monthly$Date))
GaringerOzone.monthly.ts <- ts(GaringerOzone.monthly$Ozone.monthly,
                   start=c(f_year,f_month),
                   frequency=12)
# GaringerOzone.monthly.ts
```

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

```r
#11

# decompose the daily ozone ts
GaringerOzone.daily.decomp <- stl(GaringerOzone.daily.ts,s.window = "periodic")
plot(GaringerOzone.daily.decomp)
```



```r
# decompose the monthly ozone ts
GaringerOzone.monthly.decomp <- stl(GaringerOzone.monthly.ts,s.window = "periodic")
plot(GaringerOzone.monthly.decomp)
```



```
# GaringerOzone.monthly.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

# Run SMK test
GaringerOzone.monthly.trend <- Kendall::SeasonalMannKendall(GaringerOzone.monthly.ts)
# GaringerOzone.monthly.trend
summary(GaringerOzone.monthly.trend)
```

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

```
GaringerOzone.monthly.trend2 <- trend::smk.test(GaringerOzone.monthly.ts)
# GaringerOzone.monthly.trend2
summary(GaringerOzone.monthly.trend2)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data: GaringerOzone.monthly.ts
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
```

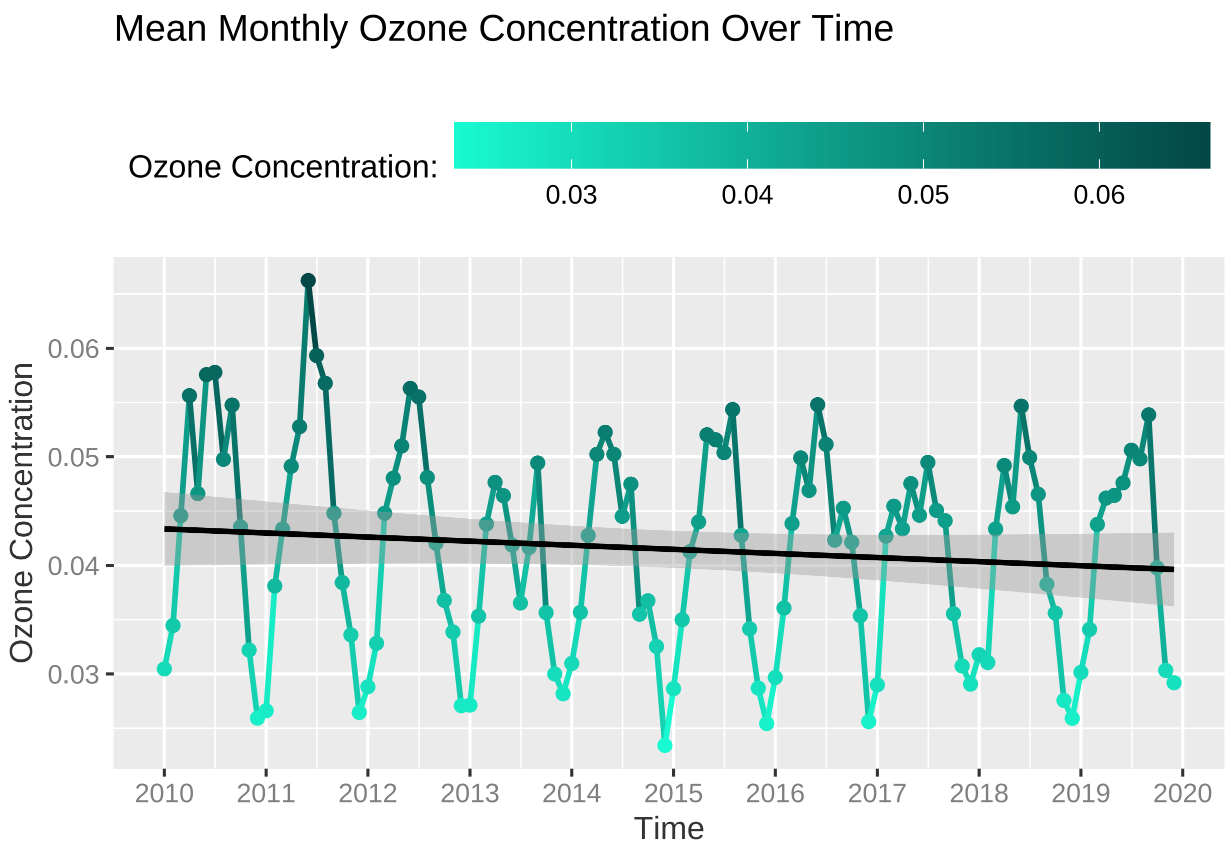
```
## H0
##                         S  varS    tau      z  Pr(>|z|)
## Season 1:    S = 0   15   125  0.333  1.252  0.21050
## Season 2:    S = 0   -1   125 -0.022  0.000  1.00000
## Season 3:    S = 0   -4   124 -0.090 -0.269  0.78762
## Season 4:    S = 0  -17   125 -0.378 -1.431  0.15241
## Season 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: There is definitely a seasonal component to the data, with clear annual peaks and valleys in ozone concentration. Thus, we need to account for this pattern when trying to find the overall trend (by reomving it). Otherwise, the model will try to consider the repeating small-scale increases/decreases as part of the trend.

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

# create a plot of the data points
ozone.m <-
ggplot(GaringerOzone.monthly) +
  geom_line(aes(x=Date, y=Ozone.monthly, color=Ozone.monthly),
            size = 1) +
  geom_point(aes(x=Date, y=Ozone.monthly, color=Ozone.monthly),
            size = 2) +
  geom_smooth(aes(x=Date, y=Ozone.monthly),
              method = lm,
              size = 1,
              color = 'black',
              alpha = 0.4) +
  scale_colour_gradient(low = "#18fbd2", high = "#034746") +
  ylab(expression("Ozone Concentration")) +
  xlab(expression("Time")) +
  ggtitle("Mean Monthly Ozone Concentration Over Time\n") +
  labs(color = "Ozone Concentration:")+
  scale_x_date(date_breaks = "years" , date_labels = "%Y")

print(ozone.m)
```

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



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: Ozone has decreased with time from 2010 to 2020. With a p-value of 0.046 and a significance level of 0.05, we can reject the null hypothesis of stationarity to conclude that ozone levels have in fact changed during this window of time. The S-values show that in most years, ozone levels dropped from one year to the next, but most of these incremental decreases are actually not statistically significant on their own (they have high p-values).

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

#  get just the seasonal component of the time series
GaringerOzone.monthly.comp = as.data.frame(GaringerOzone.monthly.decomp$time.series[,1:3])
# GaringerOzone.monthly.comp

# subract the seasonal component to get the non-seasonal component
GaringerOzone.monthly.comp = GaringerOzone.monthly.comp %>%
  mutate(GaringerOzone.monthly.comp,
         Date = GaringerOzone.monthly$Date,
         Ozone.monthly = GaringerOzone.monthly$Ozone.monthly) %>%
  mutate(GaringerOzone.monthly.comp,
         Ozone.monthly.ns = Ozone.monthly-seasonal)
# GaringerOzone.monthly.comp

#16

# make into a ts again
GaringerOzone.monthly.ns.ts <- ts(GaringerOzone.monthly.comp$Ozone.monthly.ns,
                   start=c(f_year,f_month),
                   frequency=12)
GaringerOzone.monthly.ns.ts
```

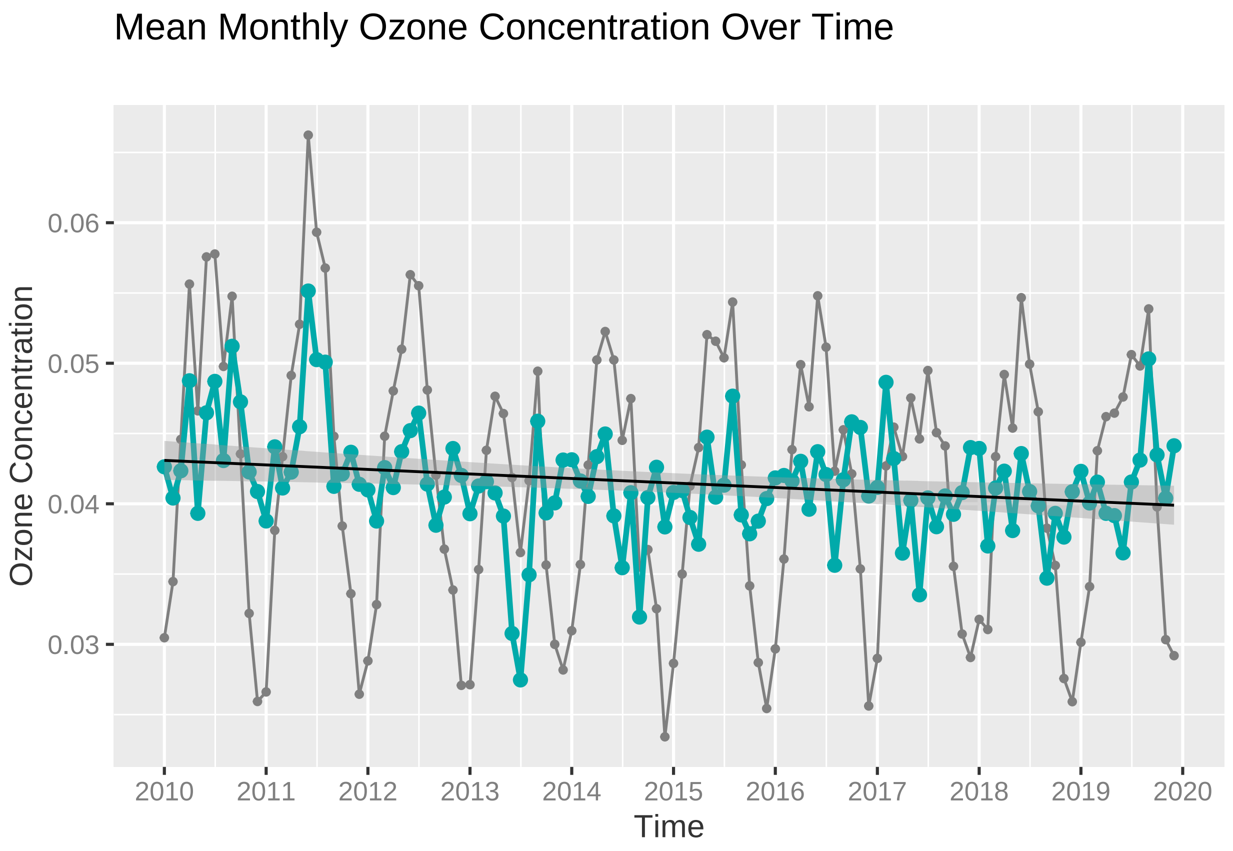
```
##             Jan        Feb        Mar        Apr        May        Jun
## 2010 0.04263190 0.04041003 0.04234881 0.04875492 0.03932081 0.04647348
## 2011 0.03877706 0.04405289 0.04112300 0.04225492 0.04548211 0.05514015
## 2012 0.04098674 0.03877333 0.04257462 0.04115492 0.04370791 0.04520681
## 2013 0.03929319 0.04126717 0.04157462 0.04077159 0.03912727 0.03077348
## 2014 0.04313190 0.04162432 0.04052623 0.04335492 0.04496598 0.03914015
## 2015 0.04080932 0.04094575 0.03902623 0.03712159 0.04474017 0.04047348
## 2016 0.04184158 0.04201471 0.04162300 0.04302159 0.03961114 0.04370681
## 2017 0.04116416 0.04864217 0.04321978 0.03648826 0.04024017 0.03352348
## 2018 0.04393835 0.03699932 0.04112300 0.04232159 0.03809501 0.04357348
## 2019 0.04230932 0.04005289 0.04154236 0.03932159 0.03915952 0.03650681
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2010 0.04871023 0.04307797 0.05120815 0.04725211 0.04226527 0.04087082
## 2011 0.05025862 0.05007797 0.04124148 0.04212308 0.04366527 0.04138695
## 2012 0.04645216 0.04140056 0.03847481 0.04047792 0.04393193 0.04201598
## 2013 0.02746829 0.03494894 0.04587481 0.03934888 0.04006527 0.04311275
## 2014 0.03545216 0.04078765 0.03194148 0.04044566 0.04259860 0.03835469
## 2015 0.04132313 0.04765862 0.03920815 0.03786501 0.03876527 0.04037082
## 2016 0.04208120 0.03562636 0.04170815 0.04583275 0.04543193 0.04054824
## 2017 0.04041991 0.03836830 0.04055815 0.03925211 0.04079860 0.04399985
## 2018 0.04087152 0.03985217 0.03470815 0.03931662 0.03763193 0.04085469
## 2019 0.04154894 0.04311023 0.05030815 0.04347792 0.04039860 0.04412888
```

```
# Run MK test
GaringerOzone.monthly.ns.ts.trend <- Kendall::MannKendall(GaringerOzone.monthly.ns.ts)
# GaringerOzone.monthly.ns.ts.trend
summary(GaringerOzone.monthly.ns.ts.trend)
```

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

```
# GaringerOzone.monthly.ns.ts.trend2 <- trend::mk.test(GaringerOzone.monthly.ns.ts)
# # GaringerOzone.monthly.ns.ts.trend
# summary(GaringerOzone.monthly.ns.ts.trend2)

# create a plot of the data points
ozone.m.ns <-
ggplot(GaringerOzone.monthly) +
  geom_point(aes(x=Date, y=Ozone.monthly),
             color ="grey50",
             size = 1) +
  geom_line(aes(x=Date, y=Ozone.monthly),
             color='grey50',
             size = 0.5) +
  geom_line(aes(x=Date, y=GaringerOzone.monthly.comp$Ozone.monthly.ns),
            color="#00aaaa",
             size = 1) +
  geom_point(aes(x=Date, y=GaringerOzone.monthly.comp$Ozone.monthly.ns),
             color="#00aaaa",
             size = 2) +
  geom_smooth(aes(x=Date, y=GaringerOzone.monthly.comp$Ozone.monthly.ns),
              method = lm,
              size = .5,
              color = 'black',
              alpha = 0.4) +
  scale_colour_gradient(low = "#18fbd2", high = "#034746") +
  ylab(expression("Ozone Concentration")) +
  xlab(expression("Time")) +
  ggtitle("Mean Monthly Ozone Concentration Over Time\n") +
  labs(color = "Ozone Concentration:") +
  scale_x_date(date_breaks = "years" , date_labels = "%Y")

print(ozone.m.ns)
```

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



*__Answer: This regular Mann-Kendall test on the non-seasonal data yields a much better p-value (p=0.0075) than the seaonal Mann-Kendall test on the seasonal data__

(p=0.0467). This allows us to more confidently reject the null hypothesis that the ozone data is stationary throughout time.