Assignment 7: Time Series Analysis

Courtney Horn

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] "/Users/courtneyannehorn/Desktop/EDA/EDAfin/Assignments"
```

```
library(tidyverse)
library(lubridate)
library(zoo)
library(trend)

mytheme <- theme_light(base_size = 12) +
  theme(panel.grid.major = element_line(colour = "black")) +
  theme(axis.text = element_text(color = "black"),
  legend.position = c("right, center"))

theme_set(mytheme)


#2: Importing the data sets
```

```
df_2010 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2010_raw.csv", stringsAsFactors =
df_2011 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2011_raw.csv", stringsAsFactors =
df_2012 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2012_raw.csv", stringsAsFactors =
df_2013 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2013_raw.csv", stringsAsFactors =
df_2014 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2014_raw.csv", stringsAsFactors =
df_2015 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2015_raw.csv", stringsAsFactors =
df_2016 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2016_raw.csv", stringsAsFactors =
df_2017 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2017_raw.csv", stringsAsFactors =
df_2018 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2018_raw.csv", stringsAsFactors =
df_2019 <- read.csv("../Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2019_raw.csv", stringsAsFactors =

#dim(df_2010)
#dim(df_2011)
#dim(df_2012)
#dim(df_2013)
#dim(df_2014)
#dim(df_2015)
#dim(df_2016)
#dim(df_2017)
#dim(df_2018)
#dim(df_2019)
#all have same # of columns, but I want to check whether the data sets have the same columns

#colnames(df_2010)
#colnames(df_2011)
#colnames(df_2010) == colnames(df_2011)
#colnames(df_2011) == colnames(df_2012)
#colnames(df_2012) == colnames(df_2013)
#colnames(df_2013) == colnames(df_2014)
#colnames(df_2014) == colnames(df_2015)
#colnames(df_2015) == colnames(df_2016)
#colnames(df_2016) == colnames(df_2017)
#colnames(df_2017) == colnames(df_2018)
#colnames(df_2018) == colnames(df_2019)

#now to join the dataframes
GaringerOzone_pt1 <- full_join(df_2010,df_2011)
GaringerOzone_pt2 <- full_join(df_2012,df_2013)
GaringerOzone_pt3 <- full_join(df_2014,df_2015)
GaringerOzone_pt4 <- full_join(df_2016,df_2017)
GaringerOzone_pt5 <- full_join(df_2018,df_2019)
GaringerOzone_pt1.2 <- full_join(GaringerOzone_pt1,GaringerOzone_pt2)
GaringerOzone_pt3.4 <- full_join(GaringerOzone_pt3,GaringerOzone_pt4)
GaringerOzone_pt1.2.3.4 <- full_join(GaringerOzone_pt1.2,GaringerOzone_pt3.4)
GaringerOzone <- full_join(GaringerOzone_pt1.2.3.4, GaringerOzone_pt5)

#checking if it worked
#colnames(GaringerOzone)
#unique(GaringerOzone$Date)
```

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: setting the date column as a date class
#GaringerOzone$Date
#class(GaringerOzone$Date)
GaringerOzone$Date <- as.Date(GaringerOzone$Date, format = "%m/%d/%Y")
class(GaringerOzone$Date)
```

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

```
head(GaringerOzone$Date)
```

```
## [1] "2010-01-01" "2010-01-02" "2010-01-03" "2010-01-04" "2010-01-05"
## [6] "2010-01-07"
```

```
# 4: Subsetting the columns
GaringerOzoneSub <- select(GaringerOzone, Date, Daily.Max.8.hour.Ozone.Concentration, DAILY_AQI_VALUE)
colnames(GaringerOzoneSub)
```

```
## [1] "Date"
## [2] "Daily.Max.8.hour.Ozone.Concentration"
## [3] "DAILY_AQI_VALUE"
```

```
dim(GaringerOzoneSub)
```

```
## [1] 3589    3
```

```
# 5: Creating a daily data set
    #I used two different methods to generate a daily data sets (framedays and framedays_start1df)
framedays_start <- seq.Date(ymd("2010-01-01"), ymd("2019-12-31"), "days")
framedays_start1 <- seq.Date(as.Date("2010-01-01"), as.Date("2019-12-31"), "days")
#View(framedays_start)
#View(framedays_start1)
framedays_start1df <- as.data.frame(framedays_start1)
#View(framedays_start1df)
framedays <-as.data.frame(framedays_start)
dim(framedays)
```

```
## [1] 3652    1
```

```
#View(framedays)
colnames(framedays) <-c("Date")
colnames(framedays)
```

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

```
colnames(framedays_start1df) <-c("Date")
colnames(framedays_start1df)
```

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

```
#View(framedays_start1df)
#View(framedays)
#framedays and framedays_start1df appear to be identical!


# 6 Combining the data frames
GaringerOzoneDays <- left_join(framedays, GaringerOzoneSub)
```

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

```
dim(GaringerOzoneDays)
```

```
## [1] 3652    3
```

```
dim(framedays)
```

```
## [1] 3652    1
```

```
#I combined the data frames two different ways
GaringerOzoneDays1 <- left_join(framedays_start1df, GaringerOzoneSub, by = c("Date"))
#View(GaringerOzoneDays1)
#View(GaringerOzoneDays)
#it appears that GaringerOzoneDays1 and GaringerOzoneDays are identical
```

Visualize

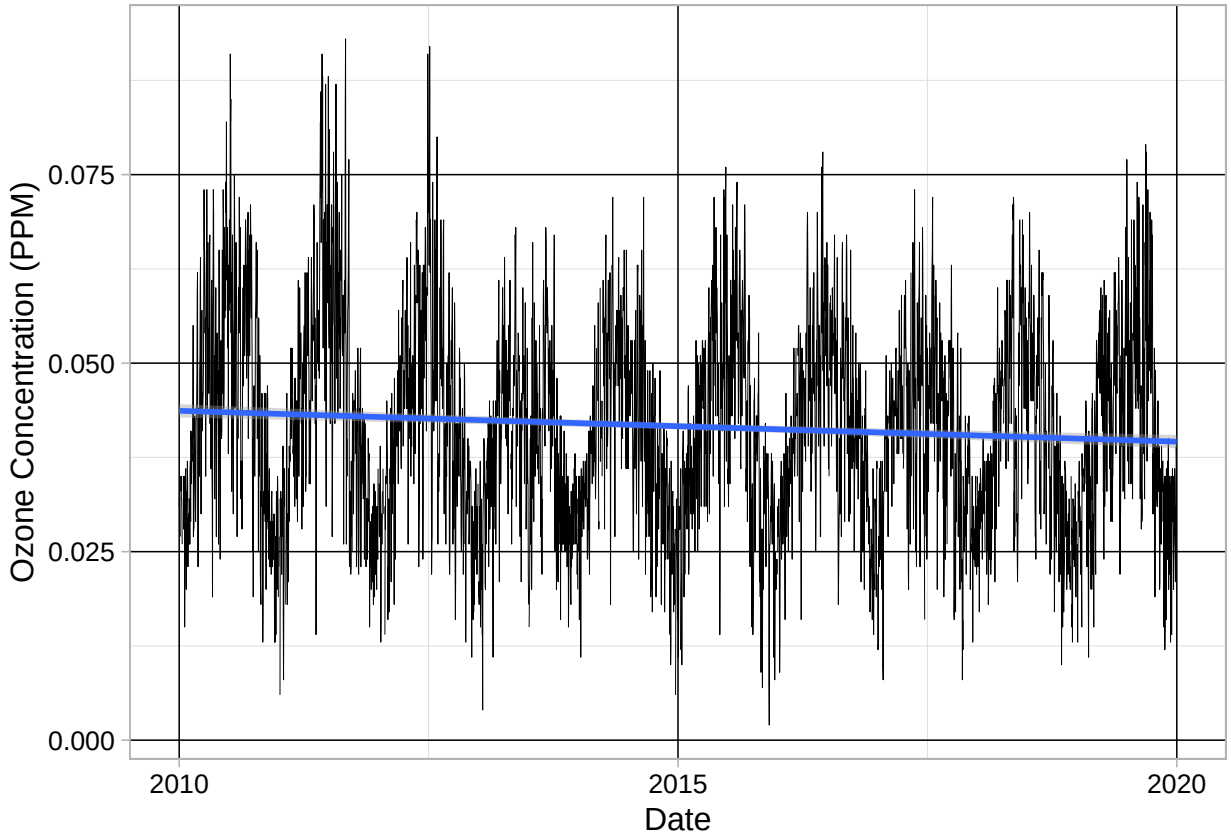
7. Create a line plot depicting ozone concentrations over time. In this case, we will plot actual concentrations in ppm, not AQI values. Format your axes accordingly. Add a smoothed line showing any linear trend of your data. Does your plot suggest a trend in ozone concentration over time?

```
# 7

ozone_time_lp1 <-
  ggplot(GaringerOzoneDays1) +
         aes(x = Date, Daily.Max.8.hour.Ozone.Concentration) +
  geom_line(size = 0.15) +
   geom_smooth( method = lm ) +
  ylab("Ozone Concentration (PPM)") + xlab("Date")
  print(ozone_time_lp1)
```

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

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




```
#ozone_time_lp and ozone_time_lp1 appear to be identical!
```

Answer: The plot indicates a decrease in ozone 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(GaringerOzoneDays)
```

```
##         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(GaringerOzoneDays)
```

```
##       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
```

```
#I found 63 NAs in both ozone columns (Daily.Max.8.hour.Ozone.Concentration) and (DAILY_AQI_VALUE)
#na.approx allows us to interpolate missing obs

GaringerOzoneDays_clean <-
  GaringerOzoneDays1 %>%
  mutate( DAILY_AQI_VALUE.clean = zoo::na.approx(DAILY_AQI_VALUE) )
#I used this pipe to remove NAs from DAILY_AQI_VALUE

GaringerOzoneDays_clean2 <-
  GaringerOzoneDays_clean %>%
  mutate( Daily.Max.8.hour.Ozone.Concentration.clean = zoo::na.approx(Daily.Max.8.hour.Ozone.Concentrati
#I used this pipe to remove NAs from Daily.Max.8.hour.Ozone.Concentration

#checking whether the linear interpolation worked
summary(GaringerOzoneDays_clean2)
```

```
##       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
##  DAILY_AQI_VALUE.clean Daily.Max.8.hour.Ozone.Concentration.clean
##  Min.   :  2.00        Min.   :0.00200
##  1st Qu.: 30.00        1st Qu.:0.03200
##  Median : 38.00        Median :0.04100
##  Mean   : 41.41        Mean   :0.04151
##  3rd Qu.: 47.00        3rd Qu.:0.05100
##  Max.   :169.00        Max.   :0.09300
##
```

```
#View(GaringerOzoneDays_clean2)
```

Answer: Piecewise constants assume that all missing data are equal to the nearest measurements. I assume that using a piecewise constant wasn't appropriate here because we can tell from our ggplots that there is a negative monotonic trend. Setting all data within a gap as equal to each other does not seem that it would represent the trend well. The spline interpolation method uses quadratic functions to interpolate. Since the trend appeared to be linear in the ggplots, it seems that a quadratic function wouldn't be a good choice for interpolating.

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
colnames(GaringerOzoneDays_clean2)
```

```
## [1] "Date"
## [2] "Daily.Max.8.hour.Ozone.Concentration"
## [3] "DAILY_AQI_VALUE"
## [4] "DAILY_AQI_VALUE.clean"
```

```
## [5] "Daily.Max.8.hour.Ozone.Concentration.clean"
```

```
GaringerOzone.monthly <-
  GaringerOzoneDays_clean2 %>%
  mutate(month = month(Date)) %>%
  mutate(year = year(Date)) %>%
  mutate(date = my(paste0(month, "-" ,year)))
#View(GaringerOzone.monthly)

#making monthly means
GaringerOzonemonthmeans <-
  GaringerOzone.monthly %>%
  group_by(date) %>%
 dplyr::summarise(meanAQI = mean(DAILY_AQI_VALUE.clean),
            meanozone = mean(Daily.Max.8.hour.Ozone.Concentration.clean))
```

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
#daily ts
GaringerOzone.daily.ts<- ts(GaringerOzoneDays_clean2$Daily.Max.8.hour.Ozone.Concentration.clean, start =
head(GaringerOzone.daily.ts, 10)
```

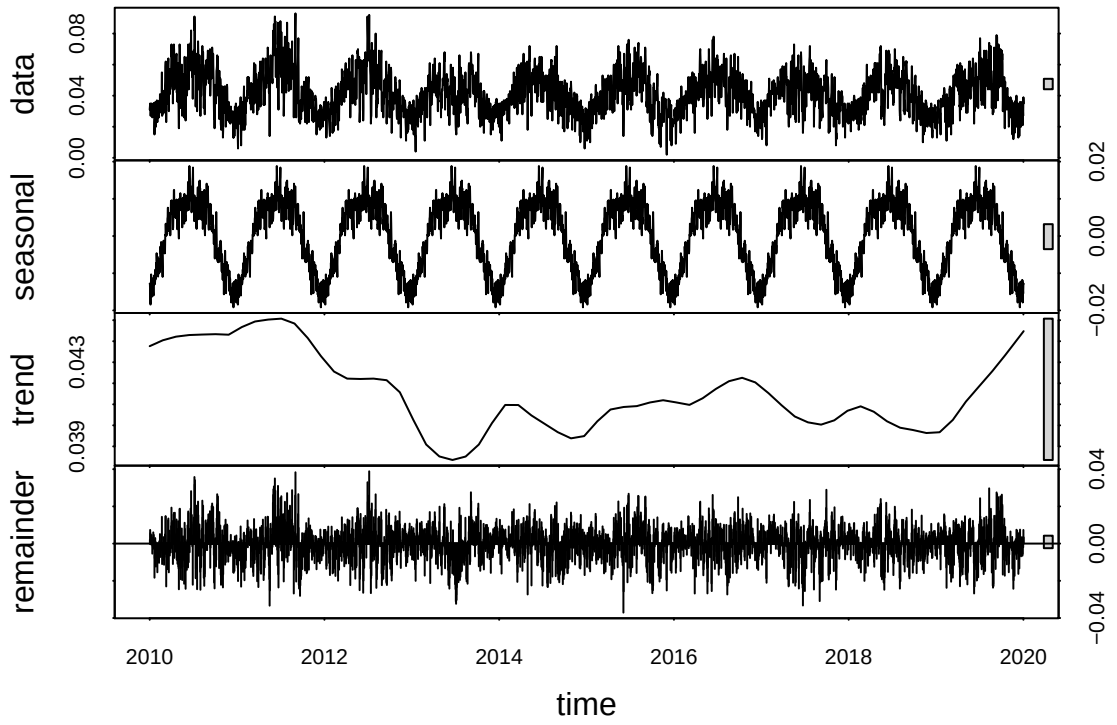
```
##  [1] 0.031 0.033 0.035 0.031 0.027 0.030 0.033 0.035 0.032 0.032
```

```
#monthly ts
GaringerOzone.monthly.ts<- ts(GaringerOzonemonthmeans$meanozone, start = c(2010, 1), frequency = 12)
#change the starting date because now monthly
head(GaringerOzone.monthly.ts, 10)
```

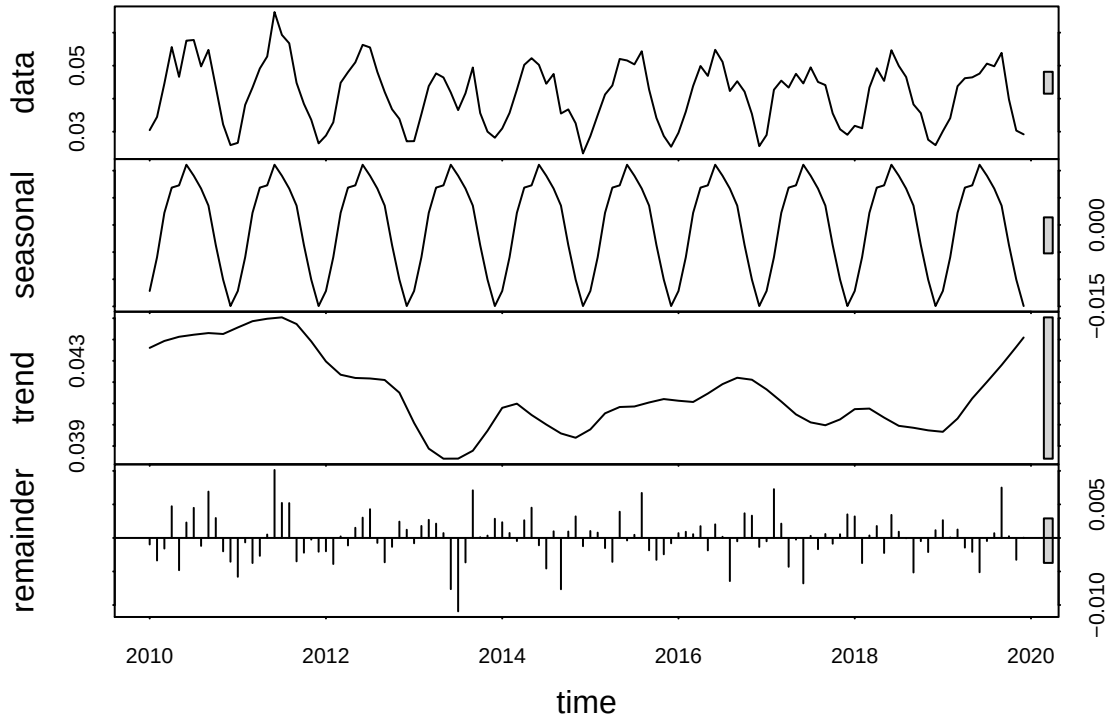
```
##  [1] 0.03046774 0.03446429 0.04458065 0.05563333 0.04661290 0.05756667
##  [7] 0.05777419 0.04977419 0.05476667 0.04354839
```

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

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




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



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
monthlyozone_trend1 <- Kendall::SeasonalMannKendall(GaringerOzone.monthly.ts)
monthlyozone_trend1
```

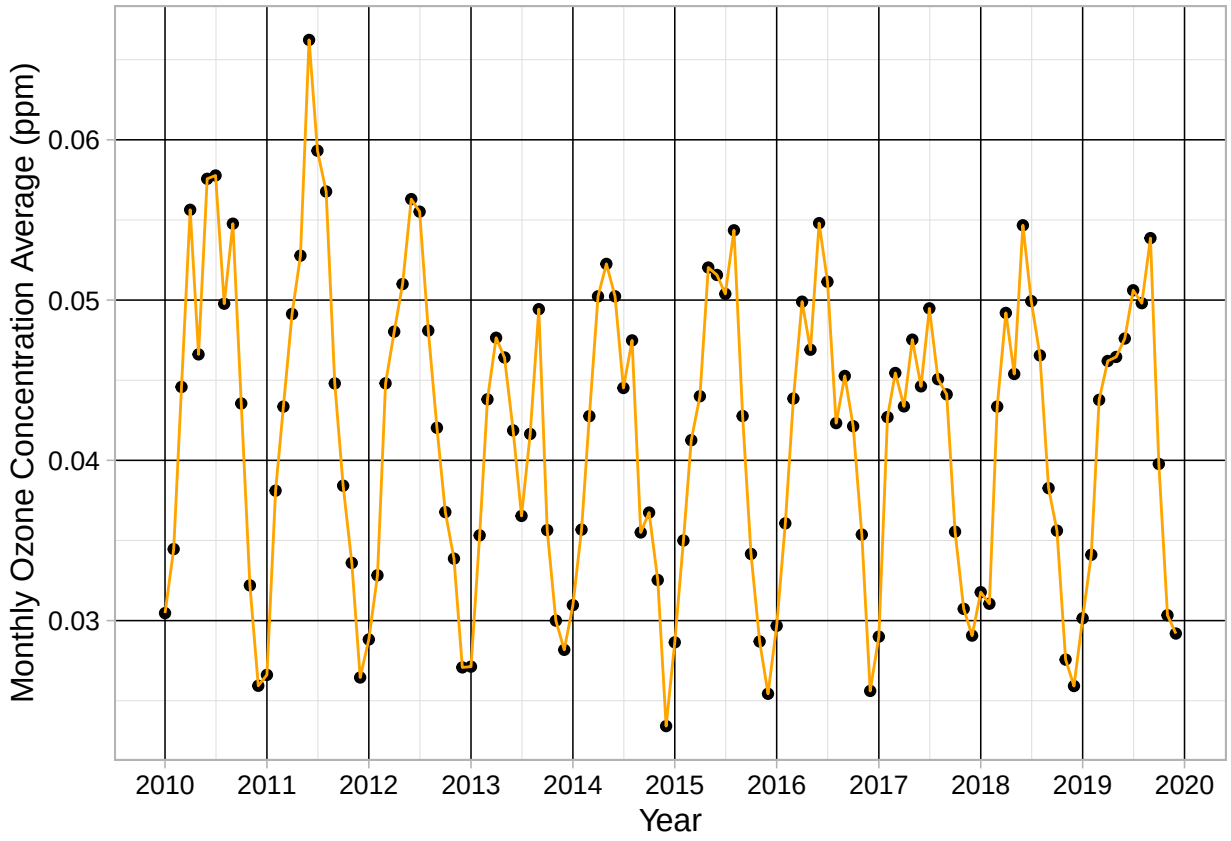
```
## tau = -0.143, 2-sided pvalue =0.046724
#the p value is significant
```

Answer: The seasonal Mann-Kendall test is the most appropriate because the data is seasonal. This is the only test we have learned that works for seasonal 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
dayplot <-
  ggplot(GaringerOzonemonthmeans, aes(x = date, y = meanozone)) +
  geom_point() +
  geom_line(col = "orange") +
  ylab("Monthly Ozone Concentration Average (ppm)") + xlab("Year") +
  scale_x_date(date_labels = "%Y", breaks = "year")

print(dayplot)
```



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 ozone values have changed over time since 2010. The seasonal Mann-Kendall trend analysis indicated that there is a monotonic trend present the monthly ozone averages (the p value of the analysis = 0.046724). Based off of my previous observations and the general shape of the graph, I beleive it is a negative trend. However, my p value was two sided and therefore did not confirm which the direction of the monotonic trend.

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 # Substract the seasonal component from the monthly ts
ozonemonthly_nonseasonal_components <- as.data.frame(monthlydecomp$time.series[,2:3])
GaringerOzone.monthly.ts
```

```
##             Jan        Feb        Mar        Apr        May        Jun
## 2010 0.03046774 0.03446429 0.04458065 0.05563333 0.04661290 0.05756667
## 2011 0.02661290 0.03810714 0.04335484 0.04913333 0.05277419 0.06623333
## 2012 0.02882258 0.03282759 0.04480645 0.04803333 0.05100000 0.05630000
## 2013 0.02712903 0.03532143 0.04380645 0.04765000 0.04641935 0.04186667
## 2014 0.03096774 0.03567857 0.04275806 0.05023333 0.05225806 0.05023333
## 2015 0.02864516 0.03500000 0.04125806 0.04400000 0.05203226 0.05156667
## 2016 0.02967742 0.03606897 0.04385484 0.04990000 0.04690323 0.05480000
## 2017 0.02900000 0.04269643 0.04545161 0.04336667 0.04753226 0.04461667
## 2018 0.03177419 0.03105357 0.04335484 0.04920000 0.04538710 0.05466667
## 2019 0.03014516 0.03410714 0.04377419 0.04620000 0.04645161 0.04760000
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2010 0.05777419 0.04977419 0.05476667 0.04354839 0.03220000 0.02593548
## 2011 0.05932258 0.05677419 0.04480000 0.03841935 0.03360000 0.02645161
## 2012 0.05551613 0.04809677 0.04203333 0.03677419 0.03386667 0.02708065
## 2013 0.03653226 0.04164516 0.04943333 0.03564516 0.03000000 0.02817742
## 2014 0.04451613 0.04748387 0.03550000 0.03674194 0.03253333 0.02341935
## 2015 0.05038710 0.05435484 0.04276667 0.03416129 0.02870000 0.02543548
## 2016 0.05114516 0.04232258 0.04526667 0.04212903 0.03536667 0.02561290
## 2017 0.04948387 0.04506452 0.04411667 0.03554839 0.03073333 0.02906452
## 2018 0.04993548 0.04654839 0.03826667 0.03561290 0.02756667 0.02591935
## 2019 0.05061290 0.04980645 0.05386667 0.03977419 0.03033333 0.02919355
```

```
nonseasonal_df <-
  GaringerOzonemonthmeans %>%
  mutate(nsobs = (meanozone - ozonemonthly_nonseasonal_components$trend))
nonseasonal_df
```

```
## # A tibble: 120 x 4
##    date       meanAQI meanozone     nsobs
##  * <date>       <dbl>     <dbl>     <dbl>
##  1 2010-01-01    28.2    0.0305 -0.0131
##  2 2010-02-01    31.9    0.0345 -0.00931
##  3 2010-03-01    43.3    0.0446  0.000647
##  4 2010-04-01    62.4    0.0556  0.0116
##  5 2010-05-01    47.7    0.0466  0.00248
##  6 2010-06-01    67.7    0.0576  0.0134
##  7 2010-07-01    69.0    0.0578  0.0135
##  8 2010-08-01    51.8    0.0498  0.00551
##  9 2010-09-01    62.9    0.0548  0.0105
## 10 2010-10-01    42.9    0.0435 -0.000734
## # ... with 110 more rows
```

```
#should I do anything about the negative numbers in here?


#16
```

```
GO_ns.monthly.ts<- ts(nonseasonal_df$nsobs, start = c(2010, 1), frequency = 12)
GO_ns.monthly.ts
```

```
##                Jan           Feb           Mar           Apr           May
## 2010 -1.314118e-02 -9.306956e-03  6.470826e-04  1.160196e-02  2.483710e-03
## 2011 -1.795373e-02 -6.606480e-03 -1.505777e-03  4.215104e-03  7.798350e-03
## 2012 -1.414853e-02 -9.832220e-03  2.457947e-03  5.761629e-03  8.805096e-03
## 2013 -1.295660e-02 -4.160085e-03  4.929059e-03  9.008464e-03  8.013676e-03
## 2014 -9.825654e-03 -5.212585e-03  1.769149e-03  9.510268e-03  1.180085e-02
## 2015 -1.113460e-02 -5.157458e-03  7.229128e-04  3.316570e-03  1.120055e-02
## 2016 -1.144886e-02 -5.028321e-03  2.786544e-03  8.633782e-03  5.439084e-03
## 2017 -1.265641e-02  1.324249e-03  4.363663e-03  2.580360e-03  7.047594e-03
## 2018 -8.956797e-03 -9.690718e-03  2.597251e-03  8.653782e-03  5.052248e-03
## 2019 -9.521649e-03 -5.871190e-03  3.484339e-03  5.438111e-03  5.217690e-03
##                Jun           Jul           Aug           Sep           Oct
## 2010  1.338922e-02  1.354850e-02  5.509315e-03  1.046261e-02 -7.335082e-04
## 2011  2.122457e-02  1.428090e-02  1.188926e-02  7.182361e-05 -5.900581e-03
## 2012  1.411661e-02  1.334425e-02  5.960500e-03 -6.734005e-05 -5.027895e-03
## 2013  3.459081e-03 -1.877234e-03  3.050873e-03  1.065425e-02 -3.602882e-03
## 2014  1.000021e-02  4.507093e-03  7.683756e-03 -4.091194e-03 -2.747393e-03
## 2015  1.072223e-02  9.529936e-03  1.340395e-02  1.722043e-03 -6.963740e-03
## 2016  1.311493e-02  9.239166e-03  2.667342e-04  3.060969e-03 -3.005282e-05
## 2017  4.319553e-03  9.374308e-03  5.021988e-03  4.141173e-03 -4.563977e-03
## 2018  1.452525e-02  9.987495e-03  6.645999e-03 -1.590120e-03 -4.183518e-03
## 2019  5.979004e-03  8.604835e-03  7.402010e-03  1.106585e-02 -3.454441e-03
##                Nov           Dec
## 2010 -1.205973e-02 -1.847770e-02
## 2011 -1.031169e-02 -1.698979e-02
## 2012 -7.636837e-03 -1.371392e-02
## 2013 -9.717003e-03 -1.207778e-02
## 2014 -6.854130e-03 -1.616426e-02
## 2015 -1.250544e-02 -1.573037e-02
## 2016 -6.745806e-03 -1.627154e-02
## 2017 -9.515900e-03 -1.142560e-02
## 2018 -1.216939e-02 -1.378208e-02
## 2019 -1.332312e-02 -1.490188e-02
```

```
monthlyozone_trend_ns <- Kendall::MannKendall(GO_ns.monthly.ts)
summary(monthlyozone_trend_ns)
```

```
## Score =  -76 , Var(Score) = 194366.7
## denominator =  7140
## tau = -0.0106, 2-sided pvalue =0.86492
```

```
#the p value isn't significant
summary(monthlyozone_trend1)
```

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

Answer: The p value is not significant when the Mann Kendall test is run on the non-seasonal time series (p value = 0.86). Therefore, we cannot reject the null hypothesis that the data is stationary. The p value was significant when the seasonal Mann Kendall test was run on the seasonal time series (p value = 0.046). However, the column used to create the non-seasonal time series object had negative values in it, which are incorrect. This may have altered the results of

the Mann Kendall analysis.