Assignment 8: Time Series Analysis

Sena McCrory

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., "Salk_A06_GLMs_Week1.Rmd") prior to submission.

The completed exercise is due on Tuesday, March 3 at 1:00 pm.

Set up

1. Set up your session:

- Check your working directory
- Load the tidyverse, lubridate, zoo, and trend packages
- Set your ggplot theme
- 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). Call these GaringerOzone201*, with the star filled in with the appropriate year in each of ten cases.

```
getwd()
```

```
## [1] "C:/Users/senam/Box Sync/My Documents/MEM classes/Duke Spring 2020/DataAnalytics/Environmental_Da
```

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

my.theme <- theme_minimal()+
  theme(legend.position = "top")
theme_set(my.theme)

# read in and bind rows
ozone.files <- list.files(path = "./Data/Raw/Ozone_TimeSeries", pattern = "*.csv", full.names = T)
GaringerOzone <- sapply(ozone.files, read.csv, simplify=FALSE) %>%
  bind_rows()
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector
```

Wrangle

2. Combine your ten datasets into one dataset called GaringerOzone. Think about whether you should use a join or a row bind.

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 2019-12-13 (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 comine 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
# 2
# see above, found more efficient way to do this

# 3
GaringerOzone$Date <- as.Date(x=GaringerOzone$Date, format = "%m/%d/%Y")


# 4
GaringerOzone <- select(GaringerOzone, c(Date, Daily.Max.8.hour.Ozone.Concentration, DAILY_AQI_VALUE))
summary(GaringerOzone$DAILY_AQI_VALUE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   30.00   38.00   41.57   47.00  169.00
```

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

```
## [1] "seq(as.Date(\"2010-01-01\"), as.Date(\"2019-12-31\"), \"day\")"
```

```r
Days <- Days %>%
  rename(Date = 'seq(as.Date(\"2010-01-01\"), as.Date(\"2019-12-31\"), \"day\")')
class(Days$Date)
```

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

```r
# 6
GaringerOzone <- left_join(Days,GaringerOzone, by = "Date")

dim(GaringerOzone) # hooray!
```

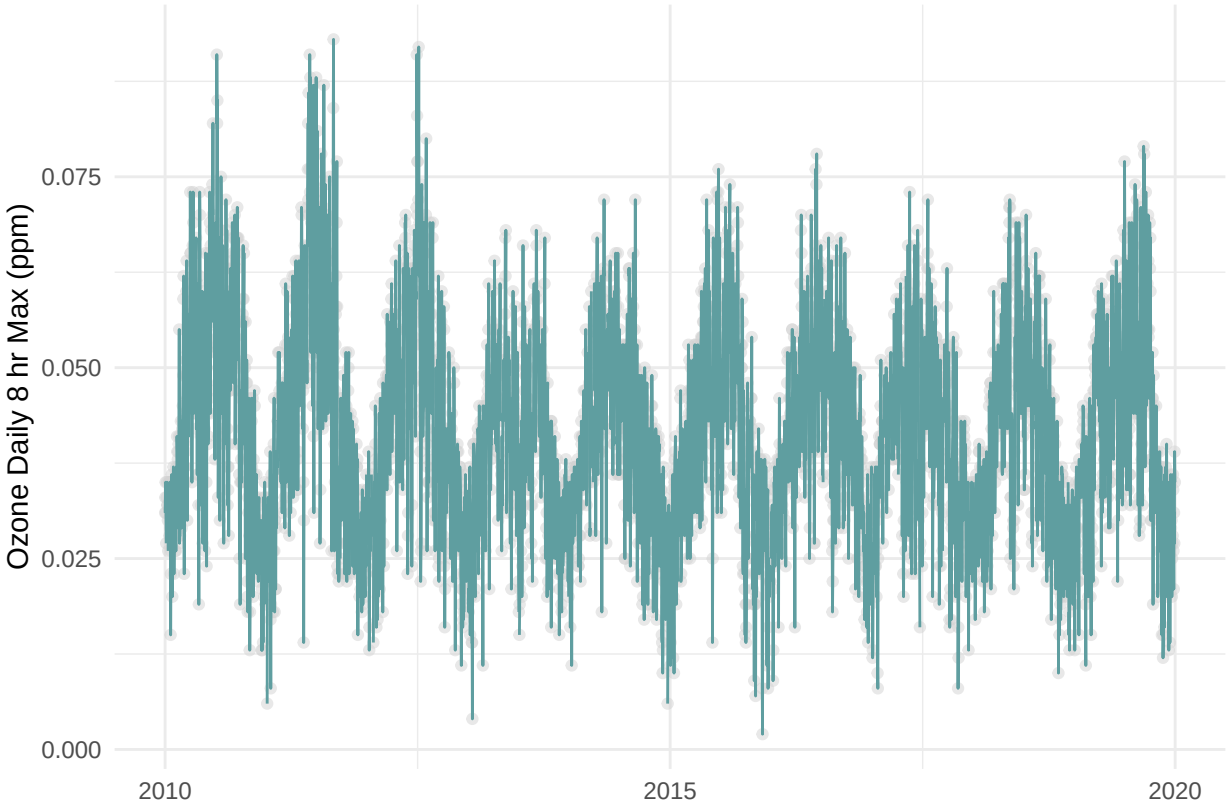
```
## [1] 3652    3
```

Visualize

7. Create a ggplot depicting ozone concentrations over time. In this case, we will plot actual concentrations in ppm, not AQI values. Format your axes accordingly.

```r
ggplot(GaringerOzone, aes(x=as.Date(Date), y = Daily.Max.8.hour.Ozone.Concentration))+
  geom_point(color = "lightgray", alpha = 0.5)+
  geom_line(color = "cadetblue")+
  labs(x= "", y = "Ozone Daily 8 hr Max (ppm)")
```

```
## Warning: Removed 63 rows containing missing values (geom_point).
```



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?

   Answer: Ozone concentrations are likely to have some temporal autocorrelation and so the missing value on one day is likely to be related to the values the day before or the day after - a linear interpolation is therefore a better option than a piecewise constant interpolation which would simply assign the same value as the closest day which would be a less accurate estimate than linear interpolation. Also, we ar mostly trying to fill in small NA gaps where only a day or two are missing in a row. A spline approach could also have been used in this case if we were concerned that that high variability (lots of high highs or low lows) could lead to unrealistic interpolated values if using a linear approach. A higher order spline interpolation could help prevent unusually high or low values from affecting interpolated values as much. In this case, a simpler linear (0 order) approach does the job.

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)

10. Generate a time series called GaringerOzone.monthly.ts, with a monthly frequency that specifies the correct start and end dates.

11. Run a time series analysis. In this case the seasonal Mann-Kendall is most appropriate; why is this?

Answer: The plot we created in part 7 shows that there seems to be a seasonal pattern in the data, and so a seasonal Mann-Kendall should be used. Also, Mann Kendall is a non-paramentric test and so our data does not have to follow a normal distribution.

12. To figure out the slope of the trend, run the function `sea.sens.slope` on the time series dataset.

13. Create a plot depicting mean monthly ozone concentrations over time, with both a geom_point and a geom_line layer. No need to add a line for the seasonal Sen's slope; this is difficult to apply to a graph with time as the x axis. Edit your axis labels accordingly.

```
# 8
GaringerOzone$Daily.Max.8.hour.Ozone.Concentration <- na.approx(GaringerOzone$Daily.Max.8.hour.Ozone.Co



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
```

```
# 9
GaringerOzone.monthly <- GaringerOzone %>%
  mutate(
    Month = month(Date),
    Year = year(Date)) %>%
  group_by(Year, Month)%>%
  dplyr :: summarise(
    monthly.mean.max8hr.ozone = mean(Daily.Max.8.hour.Ozone.Concentration)
  )
GaringerOzone.monthly$PlotDate <- as.Date(paste(
  GaringerOzone.monthly$Year,
  GaringerOzone.monthly$Month,
  1, sep="-"), format = "%Y-%m-%d")

# 10
GaringerOzone.monthly.ts <- ts(GaringerOzone.monthly$monthly.mean.max8hr.ozone,
                               frequency = 12,
                               start = c(2010,1,1),
                               end = c(2019,12,1))

# 11
GaringerOzone.monthly.ts_trend <- smk.test(GaringerOzone.monthly.ts)
GaringerOzone.monthly.ts_trend # there is a significant monotonic trend, p < 0.05
```

```
##
##  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(GaringerOzone.monthly.ts_trend) # trend not significantly different by month
```

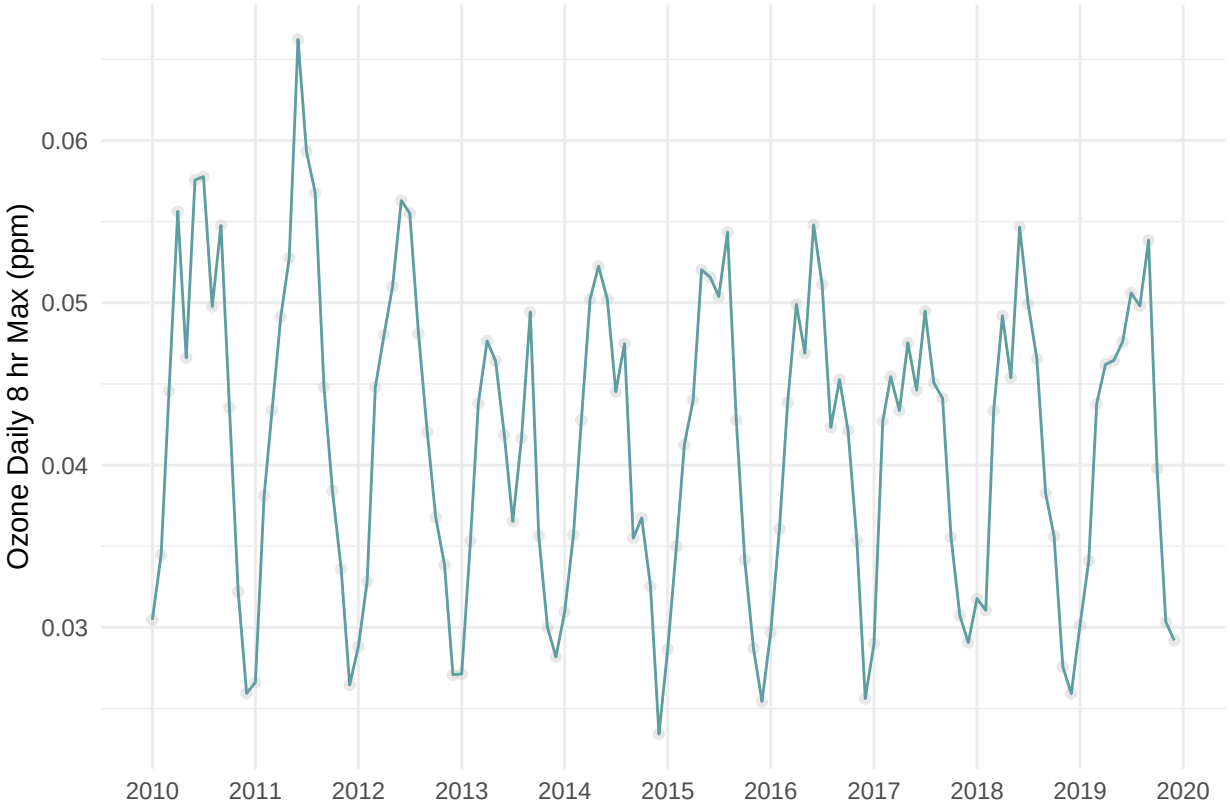
```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
```

```
##
## 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
```

```
# 12
sea.sens.slope(GaringerOzone.monthly.ts) # is this per month??
```

```
## [1] -0.0002044163
```

```
# 13
ggplot(GaringerOzone.monthly, aes(x=as.Date(PlotDate), y = monthly.mean.max8hr.ozone))+
  geom_point(color = "lightgray", alpha = 0.5)+
  geom_line(color = "cadetblue")+
  labs(x= "", y = "Ozone Daily 8 hr Max (ppm)")+
  scale_x_date(breaks = "year", labels = date_format("%Y"),
               minor_breaks = "year")
```



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: Mean 8 hour maximum ozone concentrations at Garinger have decreased significantly from 2010 to 2019 by and estimated 0.000204 ppm (Seasonal Mann-Kendall, Z = -1.963, p <0.05; Sen's Slope = -0.000204 ppm). Trend was not found to differ significantly by month.