Assignment 8: Time Series Analysis

Monisha Eadala

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.

```
# To check your working directory
getwd()

## [1] "/Users/monishaeadala/Environmental_Data_Analytics_2020"

# To load the necessary packages
library(tidyverse)
library(lubridate)
#install.packages("trend")
library(trend)
#install.packages("zoo")
library(zoo)
```

```
# To set my ggplot theme
mytheme <- theme_classic(base_size = 14) +
  theme(axis.text = element_text(color = "black"),
        legend.position = "top")
theme_set(mytheme)

# To import the necessary datasets
GaringerOzone2019 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2019_raw.csv")
GaringerOzone2018 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2018_raw.csv")
GaringerOzone2017 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2017_raw.csv")
GaringerOzone2016 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2016_raw.csv")
GaringerOzone2015 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2015_raw.csv")
GaringerOzone2014 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2014_raw.csv")
GaringerOzone2013 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2013_raw.csv")
GaringerOzone2012 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2012_raw.csv")
GaringerOzone2011 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2011_raw.csv")
GaringerOzone2010 <-
read.csv("./Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2010_raw.csv")
```

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 2019-12-31 (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 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.

```
# 2
GaringerOzone <- bind_rows(GaringerOzone2010, GaringerOzone2011,
```

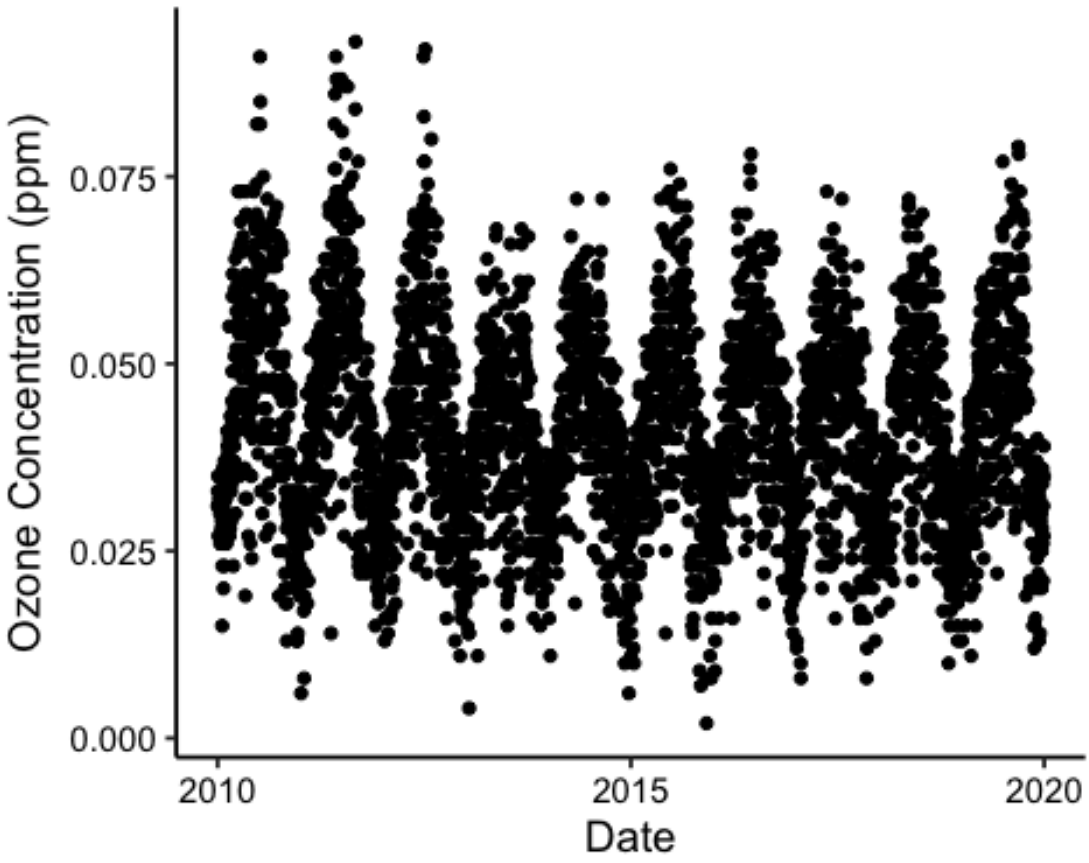
```
GaringerOzone2012, GaringerOzone2013, GaringerOzone2014, GaringerOzone2015,
GaringerOzone2016, GaringerOzone2017, GaringerOzone2018, GaringerOzone2019) #
Uses row bind instead of join to combine the 10 data sets

## 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

# 3
GaringerOzone$Date <- as.Date(GaringerOzone$Date, format = "%m/%d/%Y") # Sets
my date column as a date class
class(GaringerOzone$Date) # Checks to see if it is date now

## [1] "Date"

# 4
GaringerOzone <- GaringerOzone %>%
  select(Date, Daily.Max.8.hour.Ozone.Concentration, DAILY_AQI_VALUE) #
Wrangles the dataset so that it only contains the required columns

# 5
Days <- as.data.frame(seq(as.Date("2010-01-01"), as.Date("2019-12-31"),
"day"))
names(Days)[1] <- "Date"

# 6
GaringerOzone.combined <- left_join(Days, GaringerOzone)

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

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.

```
ggplot(GaringerOzone.combined, aes(y = Daily.Max.8.hour.Ozone.Concentration,
x = Date)) +
  geom_point() +
  ylab(expression("Ozone Concentration (ppm)")) # Creates a ggplot depicting
ozone concentrations over time

## 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: In the "linear interpolation" approach any missing data are assumed to fall between the previous and next measurement, with a straight line drawn between the known points determining the values of the interpolated data on any given date. It is one of the simplest methods to use other than the "piecewise constant" method. However, it seems to be more plausible and effective compared to the piecewise constant method; since it doesn't simply assume any missing data to be equal to the measurement made nearest to that date (could be earlier or later) like the piecewise constant method. On the otherhand, the "spline interpolation" method is similar to a linear interpolation except that a quadratic function is used to interpolate rather than drawing a straight line. Therefore, linear method is more simpler to use in this case.

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?

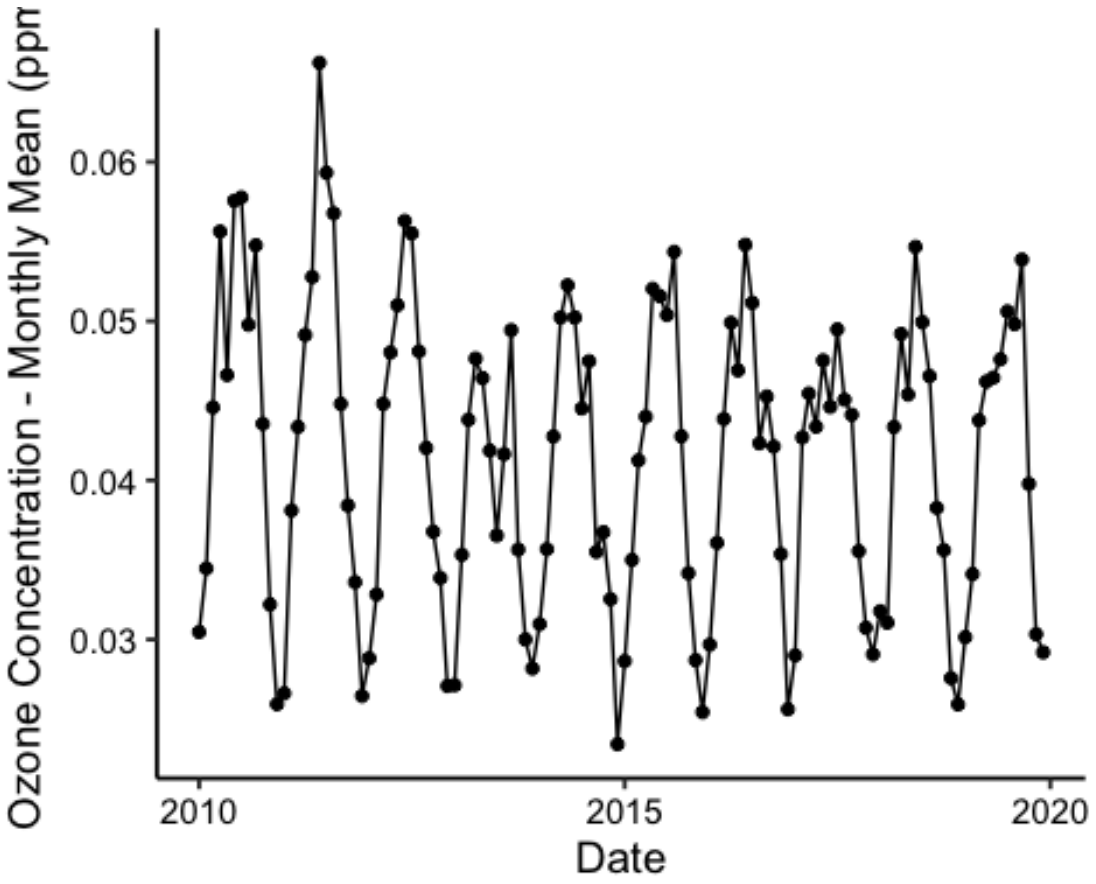
Answer: We are interested in knowing how (if) ozone concentration has changed over the course of measurement while incorporating the seasonal component. Seasonal Mann-Kendall is the most appropriate when seasonality, non-parametric, no temporal autocorelation and identical distribution is the factor. Therefore, we use the Seasonal Mann-Kendall test in order to figure out whether a monotonic trend exists.

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.combined$Daily.Max.8.hour.Ozone.Concentration <-
na.approx(GaringerOzone.combined$Daily.Max.8.hour.Ozone.Concentration)

# 9
GaringerOzone.monthly <- GaringerOzone.combined %>%
  mutate(Year = year(Date), Month = month(Date)) %>%
  group_by(Year, Month) %>%
  summarise(Daily.Max.8.hour.Ozone.Concentration.Mean =
mean(Daily.Max.8.hour.Ozone.Concentration))

GaringerOzone.monthly$Date <- as.Date(paste(GaringerOzone.monthly$Year,
                                            GaringerOzone.monthly$Month,
                                            1, sep="-"),
                                      format = "%Y-%m-%d")

# 10
GaringerOzone.monthly.ts <-
ts(GaringerOzone.monthly$Daily.Max.8.hour.Ozone.Concentration.Mean, frequency
= 12, start = c(2010, 1, 1), end = c(2019, 12, 1))

# 11
GaringerOzone.trend <- smk.test(GaringerOzone.monthly.ts)

GaringerOzone.trend # To inspect the results

##
##  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.trend) # To inspect the results

##
##  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)

## [1] -0.0002044163

# 13
GaringerOzone.monthly.plot <-
ggplot(GaringerOzone.monthly, aes(x = Date, y =
Daily.Max.8.hour.Ozone.Concentration.Mean)) +
  geom_point() +
  geom_line() +
  ylab(expression("Ozone Concentration - Monthly Mean (ppm)"))
print(GaringerOzone.monthly.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: The null hypothese states that there is no significant interaction between time (independent variable), and ozone concentration (dependent variable). We reject the null hypothesis that the season or time does not have a significant impact on the ozone concentration since the P value is less than 0.05 (p-value = 0.04965, z= -1.963). The slope of the trend is -0.0002044163.