Assignment 8: Time Series Analysis

Emma Childs

Spring 2023

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on generalized linear models.

Directions

1. Rename this file `<FirstLast>_A08_TimeSeries.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.

Set up

1. Set up your session:

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

```
knitr::opts_chunk$set(tidy.opts = list(width.cutoff=45), tidy=TRUE)

#1
getwd()
```

```
## [1] "/home/guest/EDA-Spring2023"
```

```
library(tidyverse)
library(lubridate)
install.packages("trend")
library(trend)
install.packages("zoo")
library(zoo)
library(dplyr)
library(scales)
library(here)
library(Kendall)
```

```
mytheme <- theme_classic(base_size = 12) +
  theme(axis.text = element_text(color = "black"),
        legend.position = "top")
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 Importing Data
Garinger2010 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2010_raw.csv"),
    stringsAsFactors = TRUE)
Garinger2011 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2011_raw.csv"),
    stringsAsFactors = TRUE)
Garinger2012 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2012_raw.csv"),
    stringsAsFactors = TRUE)
Garinger2013 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2013_raw.csv"),
    stringsAsFactors = TRUE)
Garinger2014 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2014_raw.csv"),
    stringsAsFactors = TRUE)
Garinger2015 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2015_raw.csv"),
    stringsAsFactors = TRUE)
Garinger2016 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2016_raw.csv"),
    stringsAsFactors = TRUE)
Garinger2017 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2017_raw.csv"),
    stringsAsFactors = TRUE)
Garinger2018 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2018_raw.csv"),
    stringsAsFactors = TRUE)
Garinger2019 <- read.csv(here("Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2019_raw.csv"),
    stringsAsFactors = TRUE)

GaringerOzone <- rbind(Garinger2010, Garinger2011,
    Garinger2012, Garinger2013, Garinger2014,
    Garinger2015, Garinger2016, Garinger2017,
    Garinger2018, Garinger2019)
```

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$Date <- mdy(GaringerOzone$Date)
# turning date column to date class

# 4
GaringerOzone_subset <- GaringerOzone %>%
    select(Date, Daily.Max.8.hour.Ozone.Concentration,
        DAILY_AQI_VALUE)
# subsetting data to just include these 4
# columns

summary(GaringerOzone_subset$Daily.Max.8.hour.Ozone.Concentration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00200 0.03200 0.04100 0.04163 0.05100 0.09300
```

```
# trying to find missing values - not sure
# if it was successful

# 5
Days <- as.data.frame(seq(as.Date("2010-01-01"),
    as.Date("2019-12-31"), by = "1 day"))
colnames(Days) <- "Date"
# creating daily dataset, filling in missing
# values with NA

# 6
GaringerOzone_new <- left_join(Days, GaringerOzone_subset)
```

```
## Joining with `by = join_by(Date)`
```

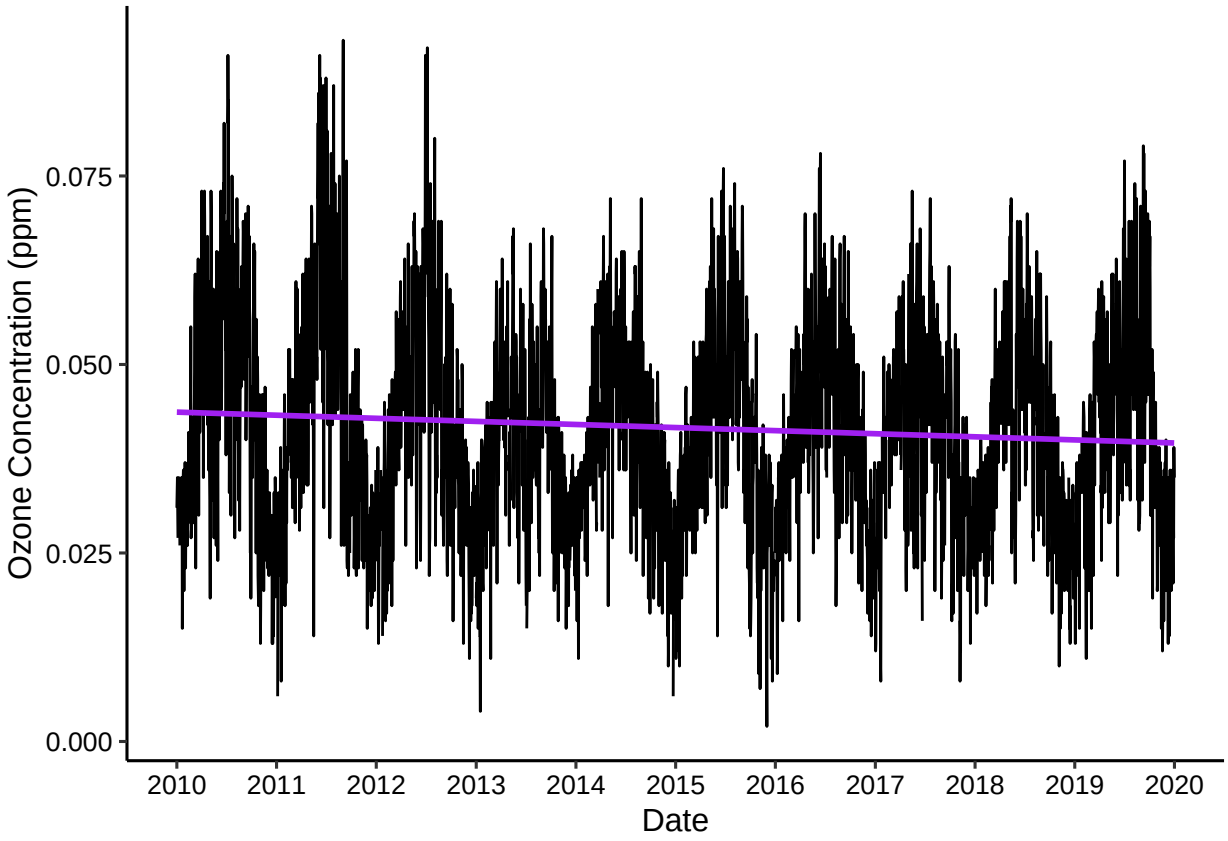
Visualize

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
OzoneOverTime_Line <- ggplot(GaringerOzone_new,
    aes(x = Date, y = Daily.Max.8.hour.Ozone.Concentration)) +
    scale_x_date(labels = date_format("%Y"), date_breaks = "1 year") +
    geom_line() + geom_smooth(method = lm, color = "purple",
    se = F) + labs(y = "Ozone Concentration (ppm)")
print(OzoneOverTime_Line)
```

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

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



Answer: The data looks like there might be slight linear decrease in ozone concentration over time, but it seems very slight, and without statistical analysis, I don't think we can verify if it is significant. But using the eyeball test of sorts, it looks like it is decreasing slightly.

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
summary(GaringerOzone_new$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
```

```
GaringerOzone_new$Daily.Max.8.hour.Ozone.Concentration <- na.approx(GaringerOzone_new$Daily.Max.8.hour.
```

Answer: We use a linear interpolation to fill in this missing data for ozone concentration because the graphed data depict a linear trend. This helps account for the na's that we estimated/approximated.

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_new %>%
    mutate(month = month(Date)) %>%
    mutate(year = year(Date)) %>%
    group_by(year, month) %>%
    summarise(meanOzoneConcentration = mean(Daily.Max.8.hour.Ozone.Concentration))
```

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

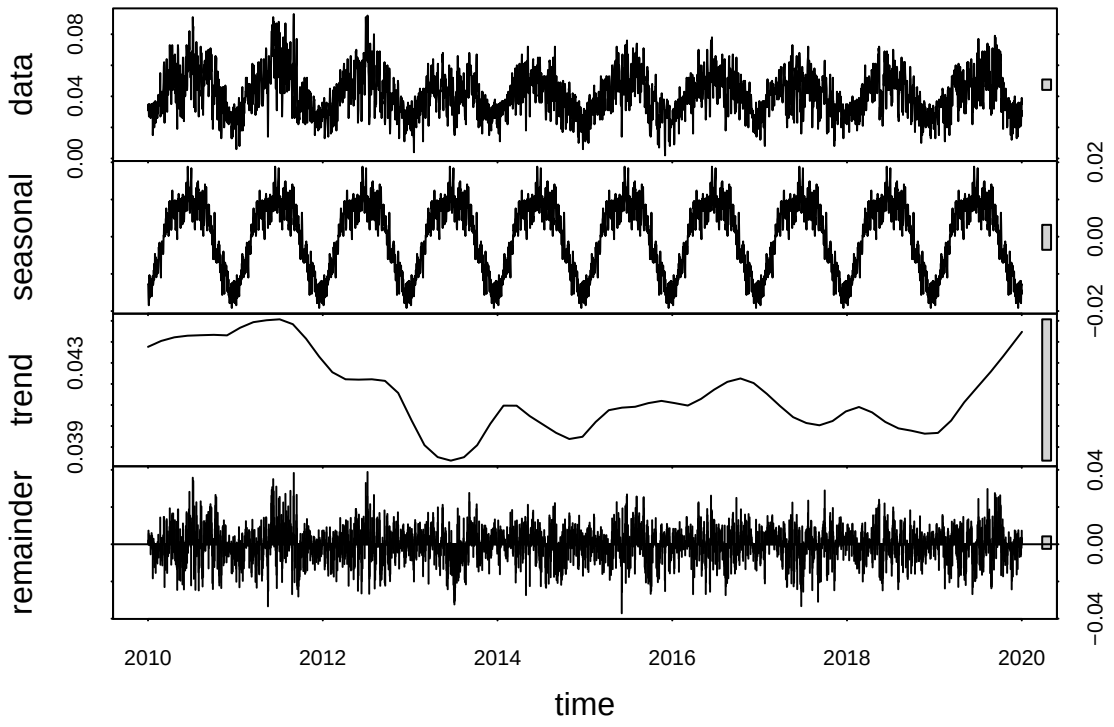
```
GaringerOzone.monthly <- GaringerOzone.monthly %>%
    mutate(firstmonth = my(paste0(month, "-",
        year)))
```

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.

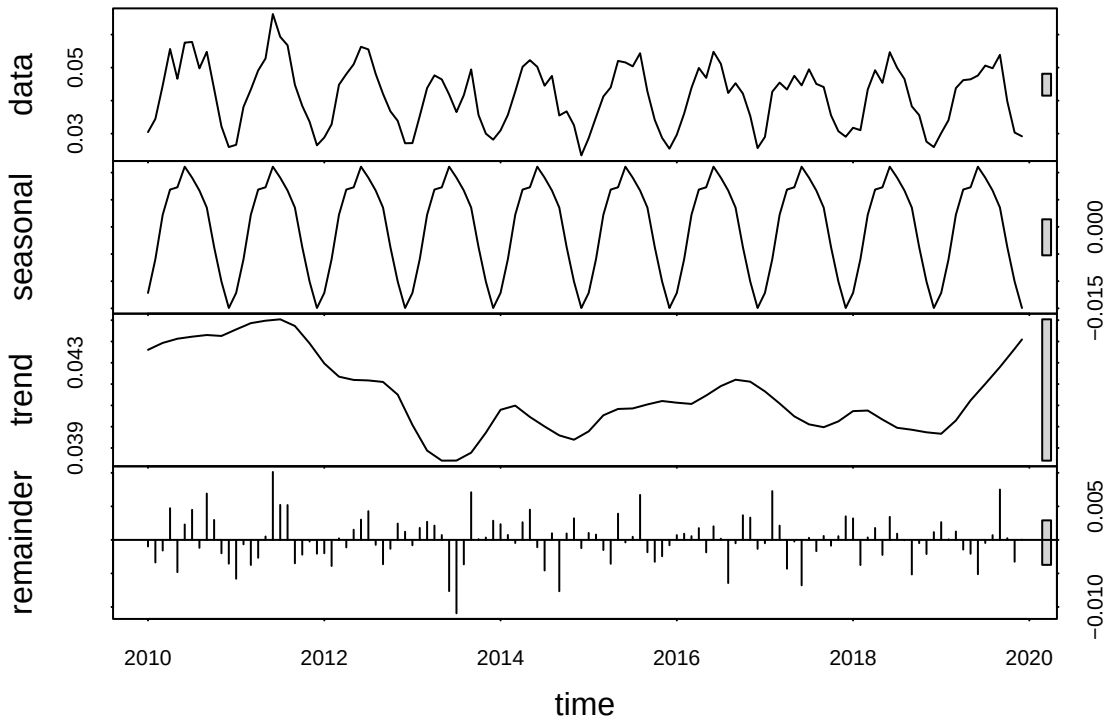
```
# 10
GaringerOzone.daily.ts <- ts(GaringerOzone_new$Daily.Max.8.hour.Ozone.Concentration,
    start = c(2010, 1), frequency = 365)
GaringerOzone.monthly.ts <- ts(GaringerOzone.monthly$meanOzoneConcentration,
    start = c(2010, 1), frequency = 12)
```

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

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




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



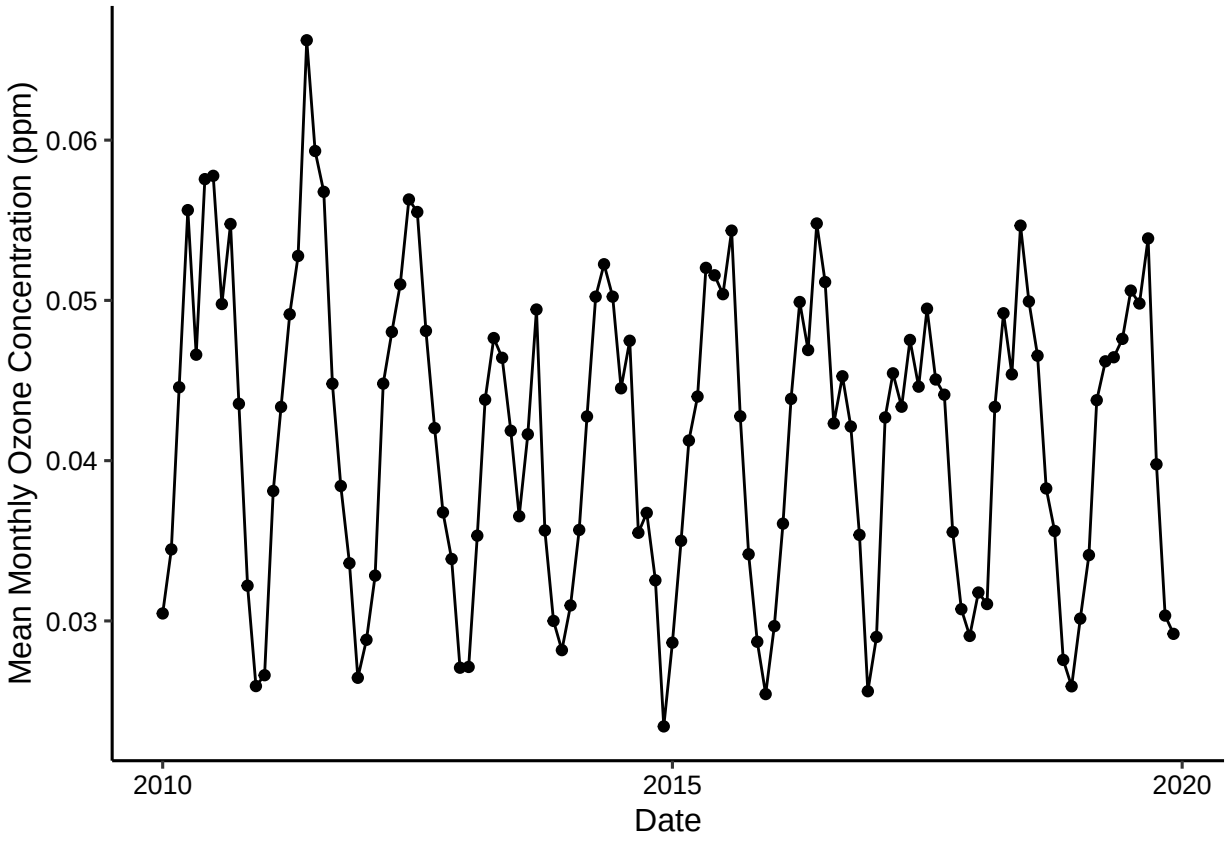
12. Run a monotonic trend analysis for the monthly Ozone series. In this case the seasonal Mann-Kendall is most appropriate; why is this?

```
# 12
Garinger_monthly_seasonal <- trend::smk.test(GaringerOzone.monthly.ts)
```

Answer: The seasonal Mann-Kendall is used for time series frames that have seasonality trends, which this one does, so it is the best tool to use.

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
GaringerOzone.monthly.plot <- ggplot(GaringerOzone.monthly,
    aes(y = meanOzoneConcentration, x = firstmonth)) +
    geom_point() + xlab("Date") + ylab("Mean Monthly Ozone Concentration (ppm)") +
    geom_line()
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:

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
GaringerOzone.Components <- as.data.frame(GOzone_monthly_decomposed$time.series[,
    2:3])

GaringerOzone.Components <- GaringerOzone.Components %>%
    mutate(data = trend + remainder) %>%
    mutate(date = GaringerOzone.monthly$firstmonth) %>%
    select(data, date)

GaringerOzone.Components.ts <- ts(GaringerOzone.Components$data,
    start = c(2010, 1), frequency = 12)

# 16
```

```
GaringerOzone.MK <- MannKendall(GaringerOzone.Components.ts)
summary(GaringerOzone.MK)
```

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

Answer: