Assignment 8: Time Series Analysis

Wynona Curaming

Spring 2024

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

```r
getwd()
```

```
## [1] "/home/guest/ENV 872/EDA_Spring2024"
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(lubridate)
library(zoo)
```

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

```r
library(trend)
library(here)
```

```
## here() starts at /home/guest/ENV 872/EDA_Spring2024
```

```r
library(ggplot2)
library(ggthemes)

my_theme <- theme_base() + theme(line=element_line(color='black',linewidth=1),
                                 plot.background = element_rect(color='beige', fill='beige'),
                                 plot.title = element_text(color='black', size=12),
                                 legend.background = element_rect(
      color='darkgrey',
      fill= 'darkgrey'),legend.text = element_text(color='white',size=12),
      axis.title = element_text(size=12),
      panel.grid.major = element_line(color = "lightgray", linewidth = 0.5),
    panel.grid.minor = element_line(color = "lightgray", linewidth = 0.25))
```

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.

```r
#1
ozone2010<-read.csv(here("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2010_ra

ozone2011<-read.csv(here("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2011_ra

ozone2012<-read.csv(here("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2012_ra

ozone2013<-read.csv(here
                    ("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2013_raw.cs

ozone2014<-read.csv(here
                    ("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2014_raw.cs

ozone2015<-read.csv(here
                    ("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2015_raw.cs

ozone2016<-read.csv(here
                    ("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2016_raw.cs
ozone2017<-read.csv(here
                    ("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2017_raw.c

ozone2018<-read.csv(here
                    ("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2018_raw.c

ozone2019<-read.csv(here
                    ("~/ENV 872/EDA_Spring2024/Data/Raw/Ozone_TimeSeries/EPAair_O3_GaringerNC2019_raw.c

GaringerOzone<-rbind(ozone2010, ozone2011, ozone2012,
                     ozone2013, ozone2014, ozone2015, ozone2016, ozone2017,
                     ozone2018, ozone2019)

colnames(GaringerOzone)
```

```
##  [1] "Date"
##  [2] "Source"
##  [3] "Site.ID"
##  [4] "POC"
##  [5] "Daily.Max.8.hour.Ozone.Concentration"
##  [6] "UNITS"
##  [7] "DAILY_AQI_VALUE"
##  [8] "Site.Name"
##  [9] "DAILY_OBS_COUNT"
## [10] "PERCENT_COMPLETE"
## [11] "AQS_PARAMETER_CODE"
## [12] "AQS_PARAMETER_DESC"
## [13] "CBSA_CODE"
## [14] "CBSA_NAME"
## [15] "STATE_CODE"
## [16] "STATE"
## [17] "COUNTY_CODE"
## [18] "COUNTY"
## [19] "SITE_LATITUDE"
## [20] "SITE_LONGITUDE"
```

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

# 4
Garinger_MaxOzone_AQI_df<-GaringerOzone%>%
  select(Date, Daily.Max.8.hour.Ozone.Concentration, DAILY_AQI_VALUE)

summary(Garinger_MaxOzone_AQI_df)
```

```
##       Date            Daily.Max.8.hour.Ozone.Concentration DAILY_AQI_VALUE
##  Min.   :2010-01-01   Min.   :0.00200                      Min.   :  2.00
##  1st Qu.:2012-07-03   1st Qu.:0.03200                      1st Qu.: 30.00
##  Median :2015-01-04   Median :0.04100                      Median : 38.00
##  Mean   :2015-01-01   Mean   :0.04163                      Mean   : 41.57
##  3rd Qu.:2017-07-02   3rd Qu.:0.05100                      3rd Qu.: 47.00
##  Max.   :2019-12-31   Max.   :0.09300                      Max.   :169.00
```

```r
# 5
Days <- as.data.frame(seq(from = as.Date('2010-01-01'), to = as.Date('2019-12-31'), by = '1 day'))
colnames(Days) <- "Date"

# 6
GaringerOzone<-left_join(Days,Garinger_MaxOzone_AQI_df)
```

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

```r
glimpse(GaringerOzone)
```

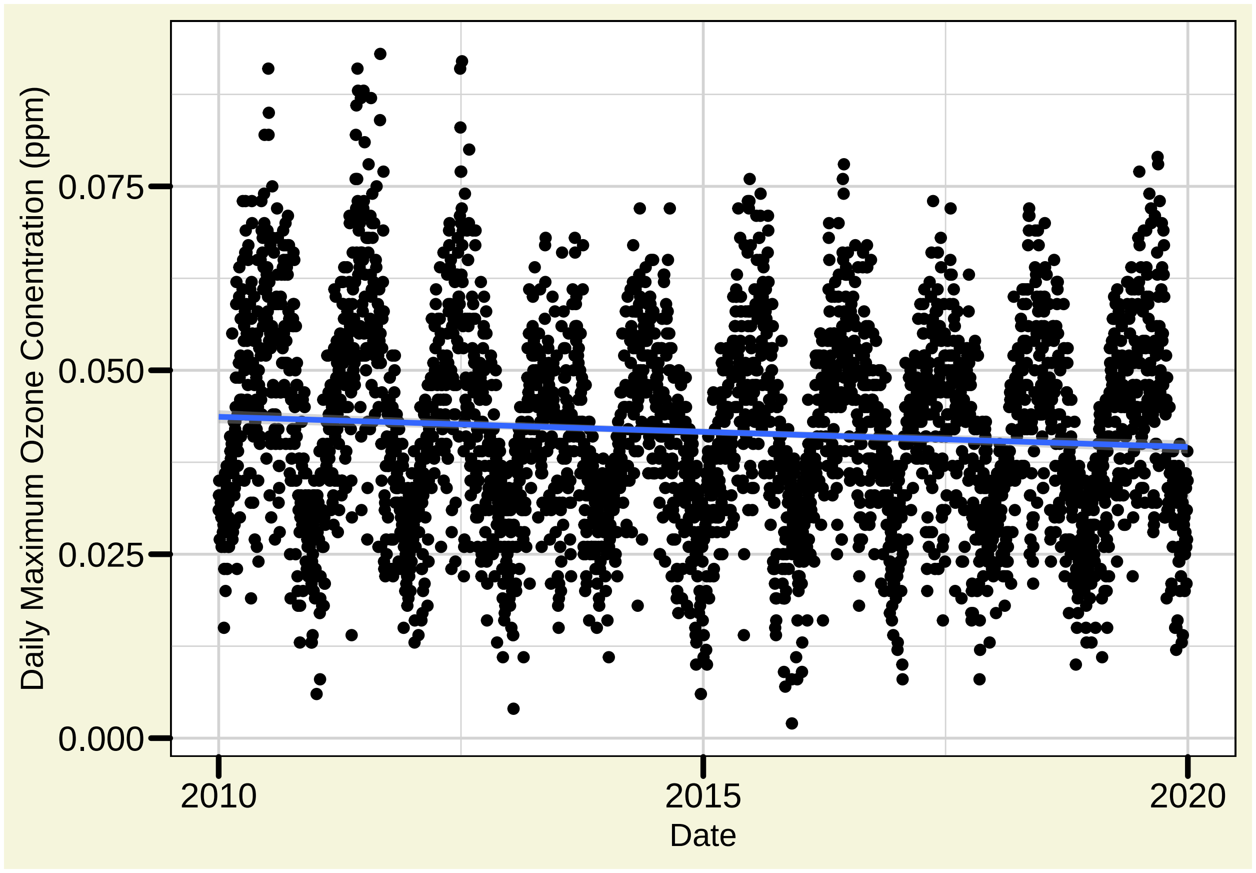
```
## Rows: 3,652
## Columns: 3
## $ Date                                 <date> 2010-01-01, 2010-01-02, 2010-01-~
## $ Daily.Max.8.hour.Ozone.Concentration <dbl> 0.031, 0.033, 0.035, 0.031, 0.027~
## $ DAILY_AQI_VALUE                      <int> 29, 31, 32, 29, 25, NA, 31, 32, 3~
```

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
ggplot(GaringerOzone, aes(x=Date,y=Daily.Max.8.hour.Ozone.Concentration))+
  geom_point() +geom_smooth(method="lm") +
  ylab("Daily Maximum Ozone Conentration (ppm)") + my_theme
```

```
## `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: The downward slope suggests that the ozone concentrations have decreased 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?

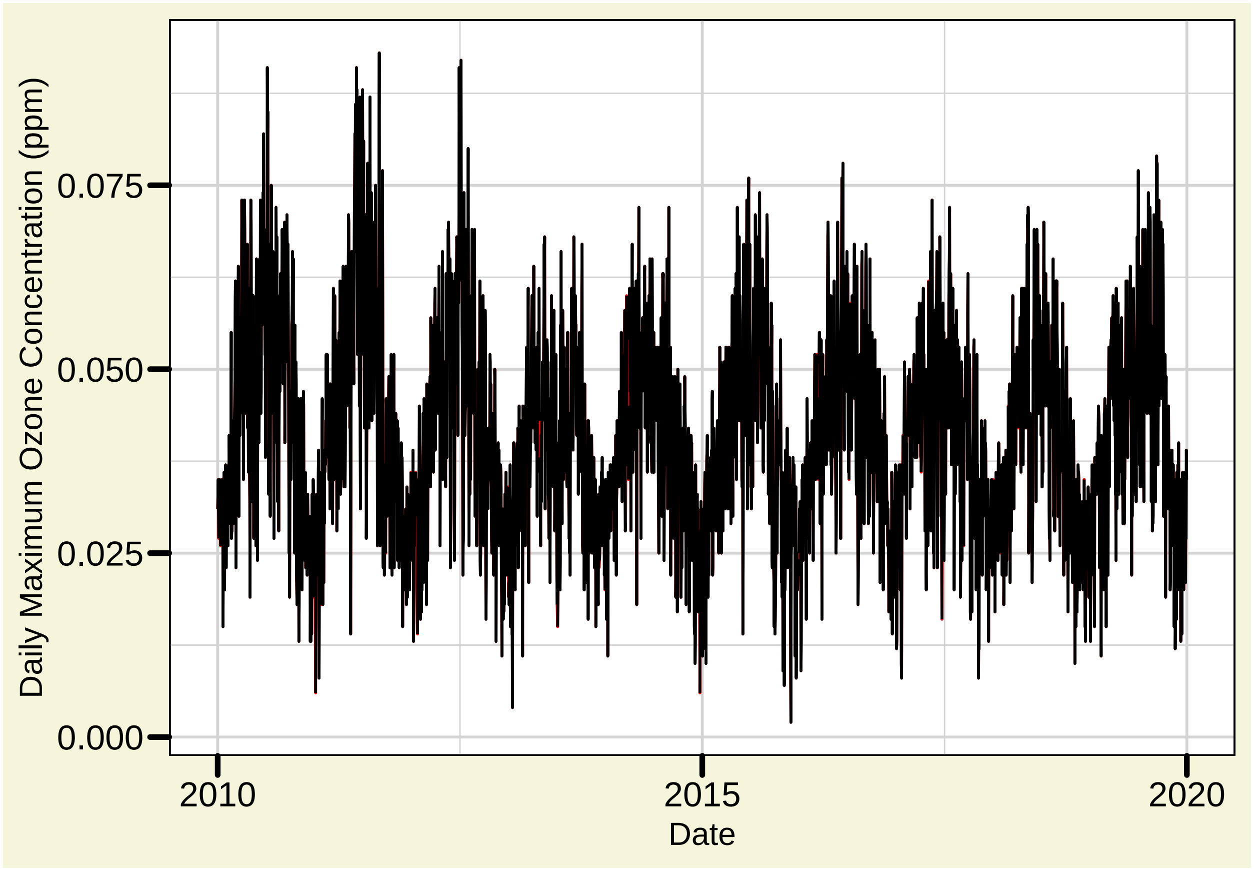
```
#8
summary(GaringerOzone$Daily.Max.8.hour.Ozone.Concentration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## 0.00200 0.03200 0.04100 0.04163 0.05100 0.09300      63
```

```
# Adding new column with no missing obs

Daily.Max.8.hour.Ozone.Concentration_clean <-
  GaringerOzone %>%
  mutate(Daily.Max.8.hour.Ozone.Concentration.clean=
           zoo::na.approx(Daily.Max.8.hour.Ozone.Concentration))

summary(Daily.Max.8.hour.Ozone.Concentration_clean)
```

```
##        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.Max.8.hour.Ozone.Concentration.clean
##  Min.   :0.00200
##  1st Qu.:0.03200
##  Median :0.04100
##  Mean   :0.04151
##  3rd Qu.:0.05100
##  Max.   :0.09300
##
```

```
# Plotting it out
ggplot(Daily.Max.8.hour.Ozone.Concentration_clean) +
  geom_line(aes(x = Date, y = Daily.Max.8.hour.Ozone.Concentration.clean), color = "red") +
  geom_line(aes(x = Date, y = Daily.Max.8.hour.Ozone.Concentration), color = "black") +
  ylab("Daily Maximum Ozone Concentration (ppm)") + my_theme
```



Answer: In linear interpolation, any missing data are assumed to fall between the previous and next measurement, which makes sense for ozone concentrations since there seems to be a clear seasonal pattern where ozone tends to rise approaching summer months then fall approaching winter. Piecewise constant or spline interpolation are inappropriate in this case because if we go by the "nearest neighbor" approach, the trend will be over or underestimated, whereas for spline, a quadratic function doesn't seem necessary to interpolate.

also known as a "nearest neighbor" approach. Any missing data are assumed to be equal to the measurement made nearest to that date (could be earlier or later). * **Linear**: could be thought of as a "connect the dots" 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. * **Spline**: similar to a linear interpolation except that a quadratic function is used to interpolate rather than drawing a straight line.

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<- Daily.Max.8.hour.Ozone.Concentration_clean%>%
  mutate(Year = year(Date))%>%
  mutate(Month = month(Date))%>%
  mutate(Date=my(paste0(Month,"-",Year)))%>%
  group_by(Year, Month, Date)%>%
  summarise(mean_ozone=mean(Daily.Max.8.hour.Ozone.Concentration.clean, na.rm= TRUE))
```

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

```
summary(GaringerOzone.monthly)
```

```
##       Year          Month             Date              mean_ozone
##  Min.   :2010   Min.   : 1.00   Min.   :2010-01-01   Min.   :0.02342
##  1st Qu.:2012   1st Qu.: 3.75   1st Qu.:2012-06-23   1st Qu.:0.03380
##  Median :2014   Median : 6.50   Median :2014-12-16   Median :0.04335
##  Mean   :2014   Mean   : 6.50   Mean   :2014-12-16   Mean   :0.04149
##  3rd Qu.:2017   3rd Qu.: 9.25   3rd Qu.:2017-06-08   3rd Qu.:0.04915
##  Max.   :2019   Max.   :12.00   Max.   :2019-12-01   Max.   :0.06623
```

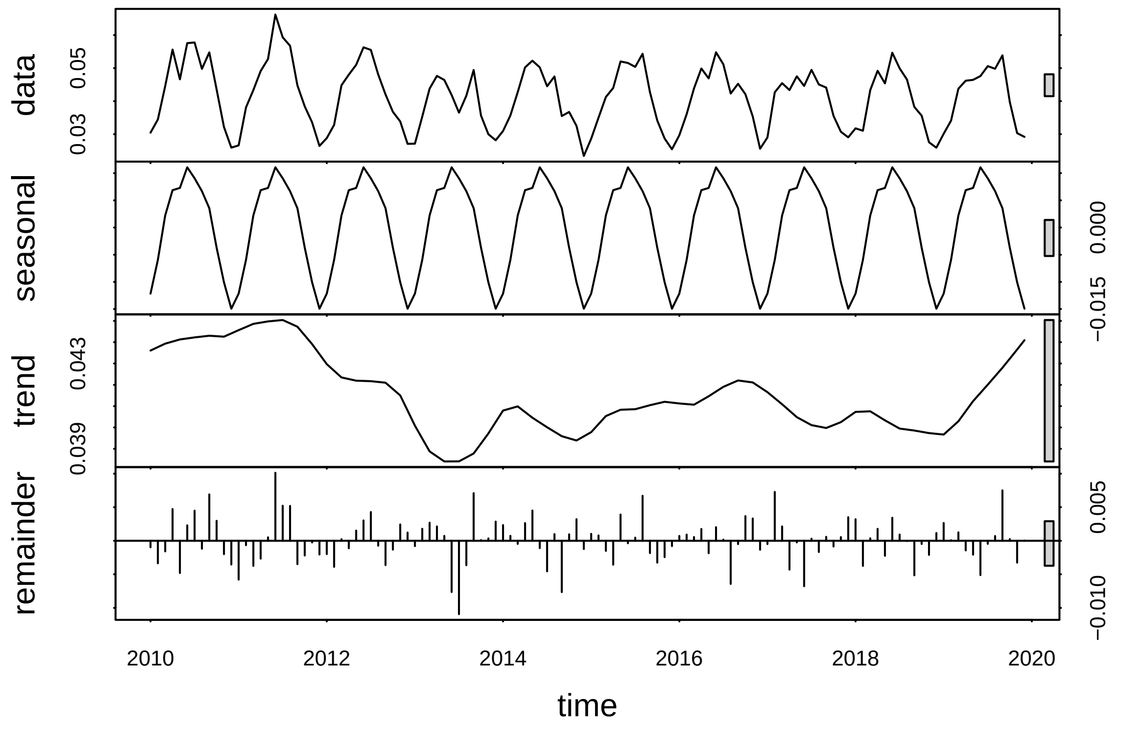
10. Generate two time series objects. Name the first `GaringerOzone.daily.ts` and base it on the dataframe of daily observations. Name the second `GaringerOzone.monthly.ts` and base it on the monthly average ozone values. Be sure that each specifies the correct start and end dates and the frequency of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

```
#10
# Generate time series (trend test needs ts, not data.frame)
f_month <- month(first(GaringerOzone$Date))
f_year <- year(first(GaringerOzone$Date))
f_day<- day(first(GaringerOzone$Date))

l_month <- month(last(GaringerOzone$Date))
l_year <- year(last(GaringerOzone$Date))
```

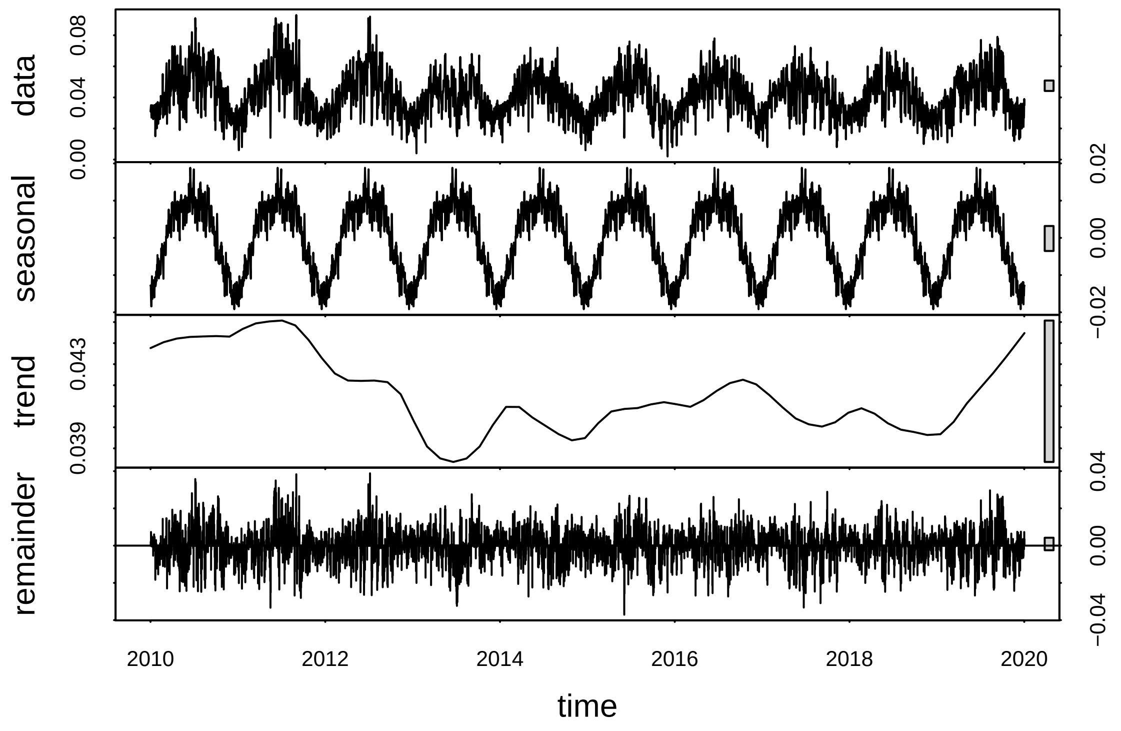
```r
l_day<- day(last(GaringerOzone$Date))

GaringerOzone.monthly.ts<- ts(GaringerOzone.monthly$mean_ozone,
                              start=c(f_year,f_month,f_day),
                              frequency=12)

GaringerOzone.daily.ts<- ts(Daily.Max.8.hour.Ozone.Concentration_clean$
                              Daily.Max.8.hour.Ozone.Concentration.clean,
                            start=c(f_year,f_month),
                            frequency=365)
```

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

```r
#11
ozone_monthly_decomposed <- stl(GaringerOzone.monthly.ts, s.window = "periodic")
plot(ozone_monthly_decomposed)
```



```r
ozone_daily_decomposed<- stl(GaringerOzone.daily.ts, s.window = "periodic")
plot(ozone_daily_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
ozone_monthly_trend <- Kendall::SeasonalMannKendall(GaringerOzone.monthly.ts)
ozone_monthly_trend
```

```
## tau = -0.143, 2-sided pvalue =0.046724
```

```
summary(ozone_monthly_trend)
```

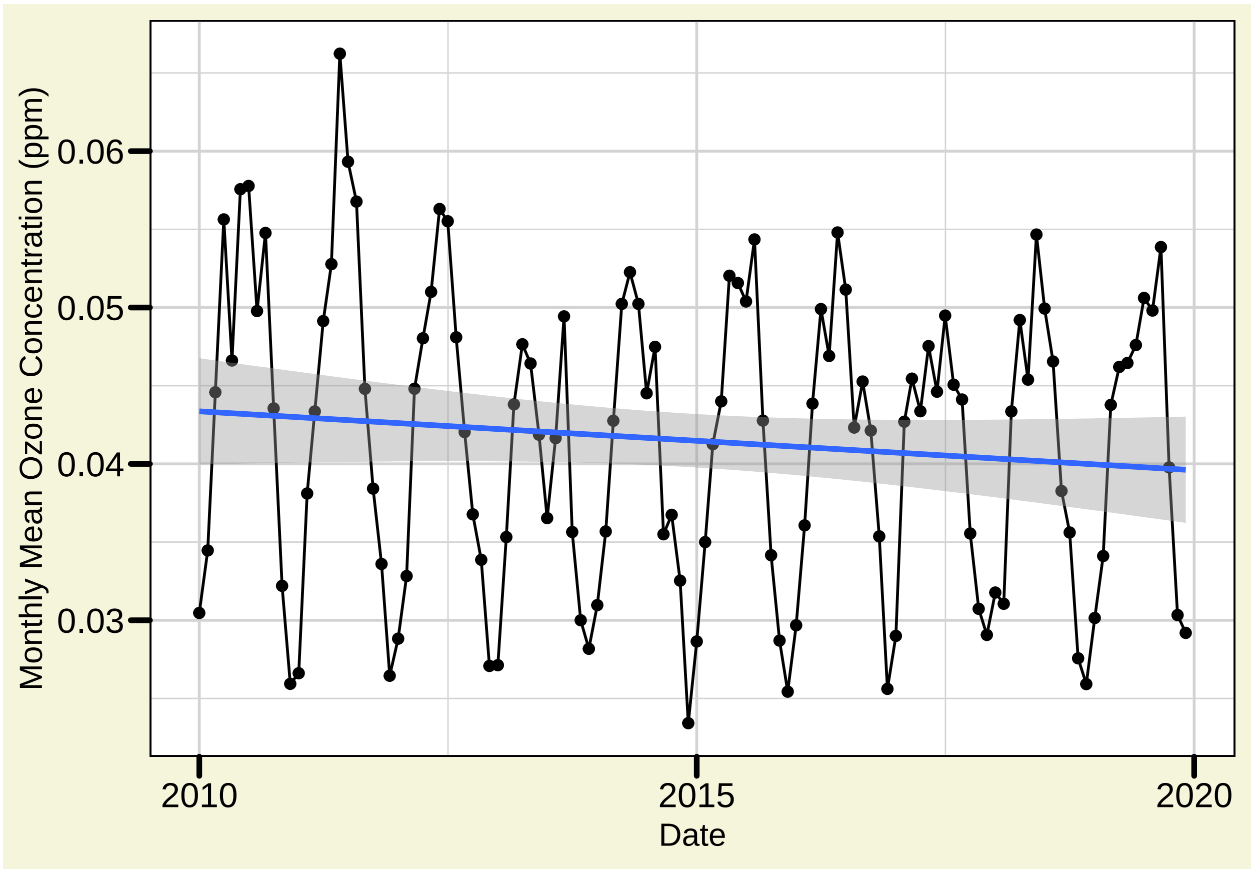
```
## Score =  -77 , Var(Score) = 1499
## denominator =  539.4972
## tau = -0.143, 2-sided pvalue =0.046724
```

Answer: The plot shows that the trend for ozone is both seasonal (it goes up and down, rising up approaching summer) and non-parametric (does not make assumptions about the parameters of the population distribution from which the sample is drawn).

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.

```r
# 13
ozone_monthly_plot<- ggplot(GaringerOzone.monthly, aes(x= Date, y=mean_ozone))+
  geom_point()+ geom_line() + geom_smooth(method=lm) +
  ylab("Monthly Mean Ozone Concentration (ppm)") + my_theme

ozone_monthly_plot
```

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



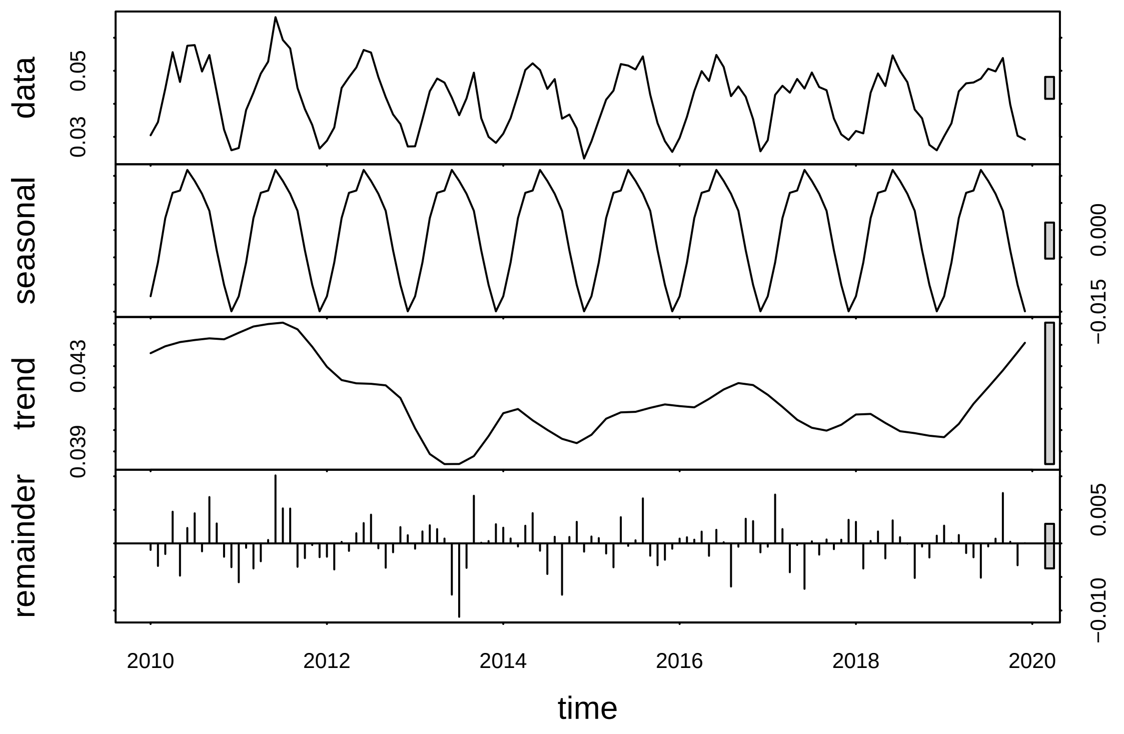
14. To accompany your graph, summarize your results in context of the research question. Include output from the statistical test in parentheses at the end of your sentence. Feel free to use multiple sentences in your interpretation.

    Answer: Overall, ozone concentrations have changed over the 2010s at this station. It has seen a statistically significant decline ($p<0.05$ since $p=0.046724$) from 2010 to 2020. Time and ozone concentration are negatively correlated.

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.

```r
#15
plot(ozone_monthly_decomposed)
```



```r
ozone_monthly_components <- as.data.frame(ozone_monthly_decomposed$time.series[,2])

ozone_no_seasonal_ts <- ts(ozone_monthly_components,
start=c(f_year,f_month),
frequency=12)

#16
Kendall::SeasonalMannKendall(ozone_no_seasonal_ts)
```

```
## tau = -0.304, 2-sided pvalue =2.291e-05
```

Answer: The p-value is even lower (0.00002291 compared to 0.046724) for the non-seasonal Ozone monthly series comparied to the seasonal complete series. This suggests that the decline in ozone is truly significant and is not just because of the seasons. The negative tau for both (-0.304 for non-seasonal and -0.143 for seasonal) supports that time and ozone are negatively correlated.