Assignment 7: Time Series Analysis

Jiawei Liang

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

```
## Warning: package 'trend' was built under R version 4.2.2
```

```
library(zoo)
```

```
## Warning: package 'zoo' was built under R version 4.2.2
```

```
theme_default <- theme_set(theme_bw())
theme_set(theme_default)
#2
getwd()
```

```
## [1] "C:/Users/Jiawei Liang/Documents/EDA-Fall2022/Assignments"
```

```
setwd('c:/Users/Jiawei Liang/Documents/EDA-Fall2022/Data/Raw/Ozone_TimeSeries')
NC2010 <- read.csv('EPAair_O3_GaringerNC2010_raw.csv',stringsAsFactors =TRUE)
NC2011 <- read.csv('EPAair_O3_GaringerNC2011_raw.csv',stringsAsFactors =TRUE)
NC2012 <- read.csv('EPAair_O3_GaringerNC2012_raw.csv',stringsAsFactors =TRUE)
NC2013 <- read.csv('EPAair_O3_GaringerNC2013_raw.csv',stringsAsFactors =TRUE)
NC2014 <- read.csv('EPAair_O3_GaringerNC2014_raw.csv',stringsAsFactors =TRUE)
NC2015 <- read.csv('EPAair_O3_GaringerNC2015_raw.csv',stringsAsFactors =TRUE)
NC2016 <- read.csv('EPAair_O3_GaringerNC2016_raw.csv',stringsAsFactors =TRUE)
NC2017 <- read.csv('EPAair_O3_GaringerNC2017_raw.csv',stringsAsFactors =TRUE)
NC2018 <- read.csv('EPAair_O3_GaringerNC2018_raw.csv',stringsAsFactors =TRUE)
NC2019 <- read.csv('EPAair_O3_GaringerNC2019_raw.csv',stringsAsFactors =TRUE)

GaringerOzone <- rbind(NC2010, NC2011, NC2012, NC2013, NC2014, NC2015, NC2016, NC2017, NC2018, NC2019)
```

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
class(GaringerOzone$Date)
```

```
## [1] "factor"
```

```
GaringerOzone$Date <- as.Date(GaringerOzone$Date, format = "%m/%d/%Y")
# 4
GaringerOzone_1 <-   select(GaringerOzone, Date, Daily.Max.8.hour.Ozone.Concentration,         DAILY_AQ
# 5
Days <- as.data.frame(seq.Date(from = as.Date("2010-01-01"),to= as.Date("2019-12-31"),by=1))
       colnames(Days) <- c("Date")
# 6
GaringerOzone_2 <- left_join(Days,GaringerOzone_1, by = c("Date"))
```

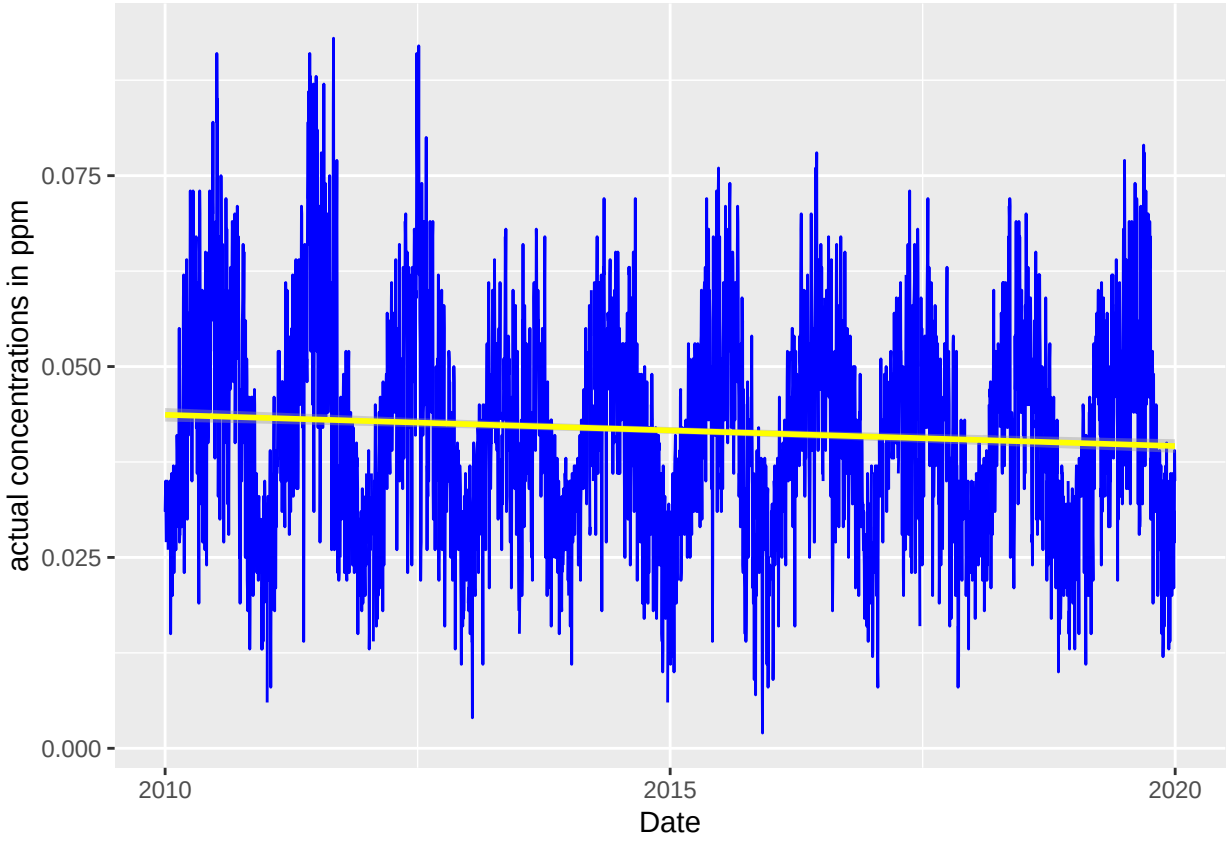
Visualize

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
Ozone_data_plot <-
ggplot(GaringerOzone_2, aes(x = Date, y = Daily.Max.8.hour.Ozone.Concentration)) +
  #geom_point(color = "red") +
  geom_line(color = "blue") +
  ylab("actual concentrations in ppm") +
  geom_smooth( method = lm, color = "yellow" )
print(Ozone_data_plot)
```

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

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



Answer:

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(GaringerOzone_2)
```

```
##         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
```

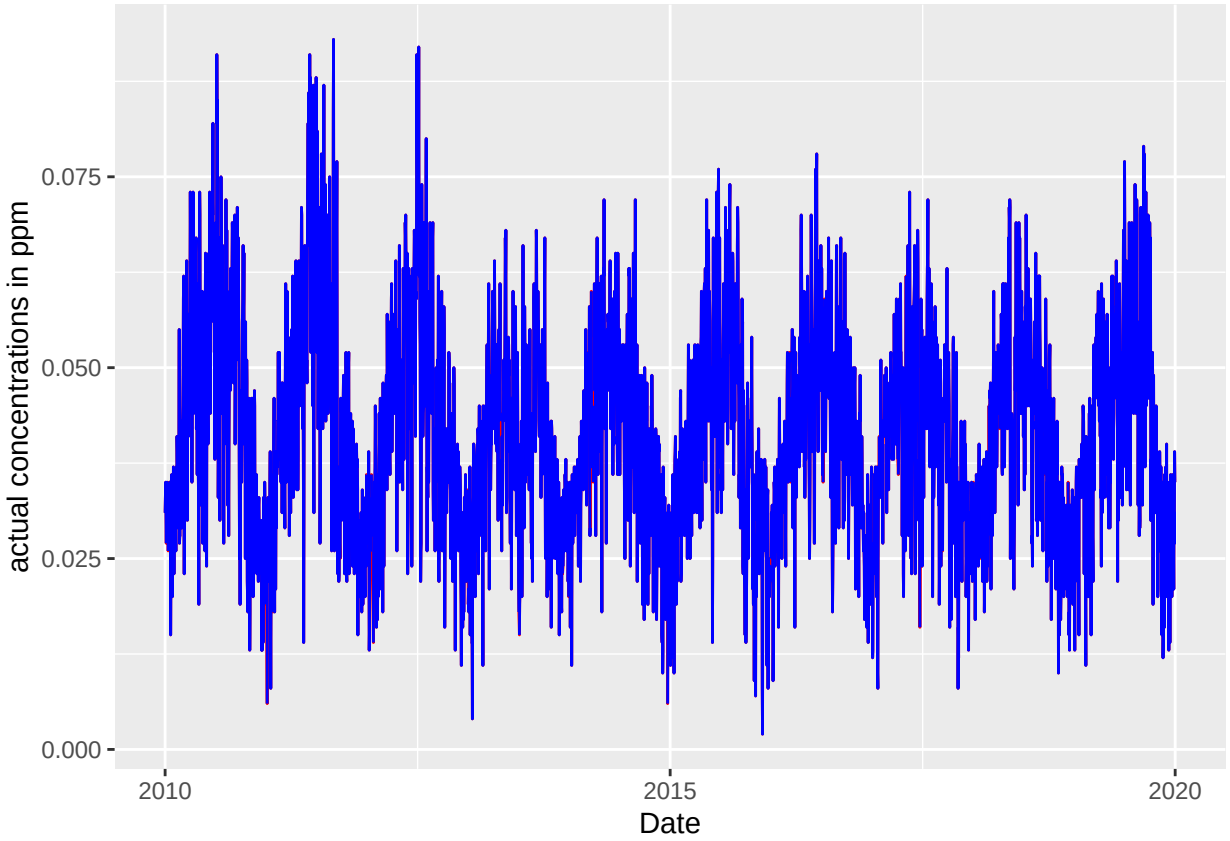
```
summary(GaringerOzone_2$Daily.Max.8.hour.Ozone.Concentration)
```

```
##    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, just for illustration purpose
# In real applications you will simply replace NAs
GaringerOzone_2_clean <-
  GaringerOzone_2 %>%
  mutate( Daily.Max.8.hour.Ozone.Concentration.clean = zoo::na.approx(Daily.Max.8.hour.Ozone.Concentrati
summary(GaringerOzone_2_clean$Daily.Max.8.hour.Ozone.Concentration.clean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00200 0.03200 0.04100 0.04151 0.05100 0.09300
```

```
#Note the NA is gone
ggplot(GaringerOzone_2_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 = "blue") +
  ylab("actual concentrations in ppm")
```



Answer: Because, cubic interpolation is better than linear interpolation in most aspects, like smoothness of the function. However linear interpolation is better in the linear interpolation. Because there will not produce the "overshoot" situation.

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_2.monthly <- GaringerOzone_2_clean %>%
  mutate(month_year = floor_date(Date, "month")) %>%
group_by(month_year) %>%
summarise(mean.concentration = mean(Daily.Max.8.hour.Ozone.Concentration.clean))
```

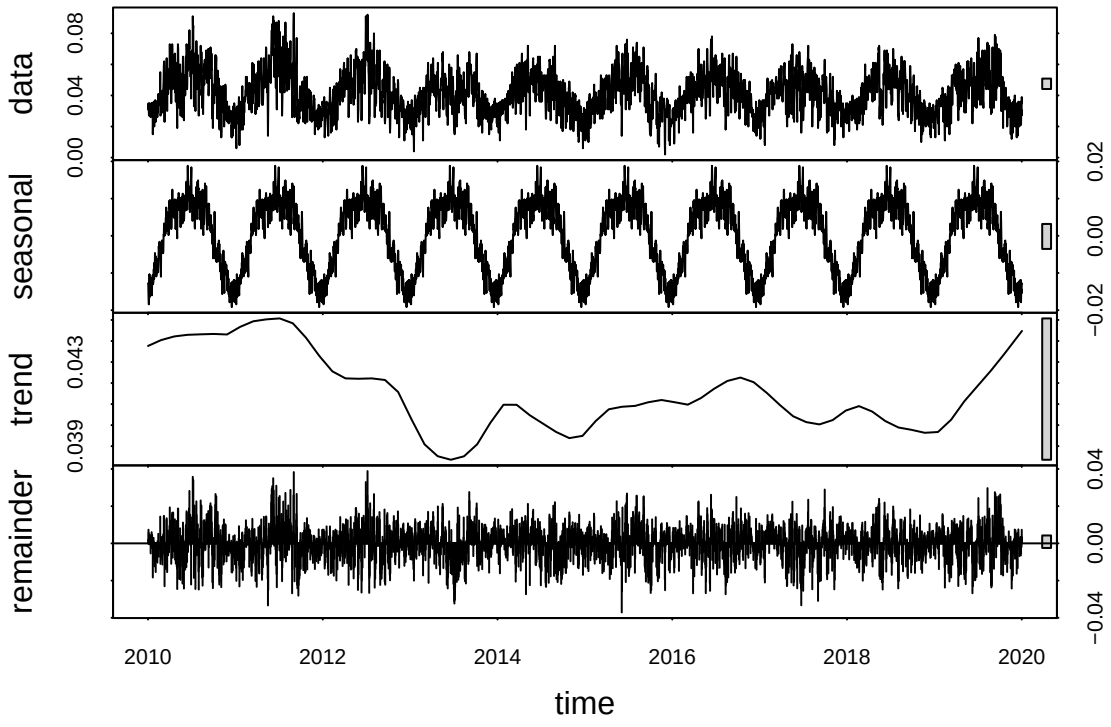
10. Generate two time series objects. Name the first `GaringerOzone.daily.ts` and base it on the dataframe of daily observations. Name the second `GaringerOzone.monthly.ts` and base it on the monthly average ozone values. Be sure that each specifies the correct start and end dates and the frequency of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

```
#10
f_day <- day(first(GaringerOzone_2_clean$Date))
f_month <- month(first(GaringerOzone_2.monthly$month_year))
```

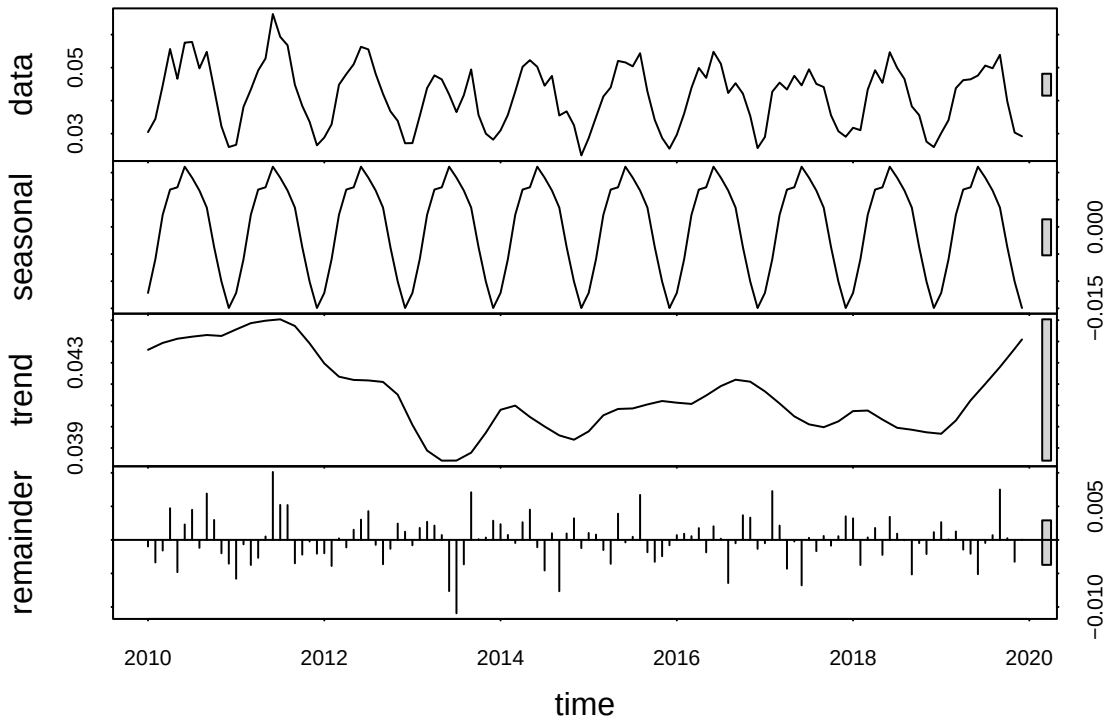
```
f_year <- year(first(GaringerOzone_2_clean$Date))

GaringerOzone.daily.ts <- ts(GaringerOzone_2_clean$Daily.Max.8.hour.Ozone.Concentration.clean,
                   start=c(f_year,f_day),
                   frequency=365)

GaringerOzone.monthly.ts <- ts(GaringerOzone_2.monthly$mean.concentration,
                   start=c(f_year,f_month),
                   frequency=12)
```

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

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




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



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
# Run SMK test
GaringerOzone_monthly_trend1 <- Kendall::SeasonalMannKendall(GaringerOzone.monthly.ts)

# Inspect results
GaringerOzone_monthly_trend1
```

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

```
summary(GaringerOzone_monthly_trend1)
```

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

```
GaringerOzone_monthly_trend2 <- trend::smk.test(GaringerOzone.monthly.ts)

# Inspect results
GaringerOzone_monthly_trend2
```

```
##
##  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_trend2)
```

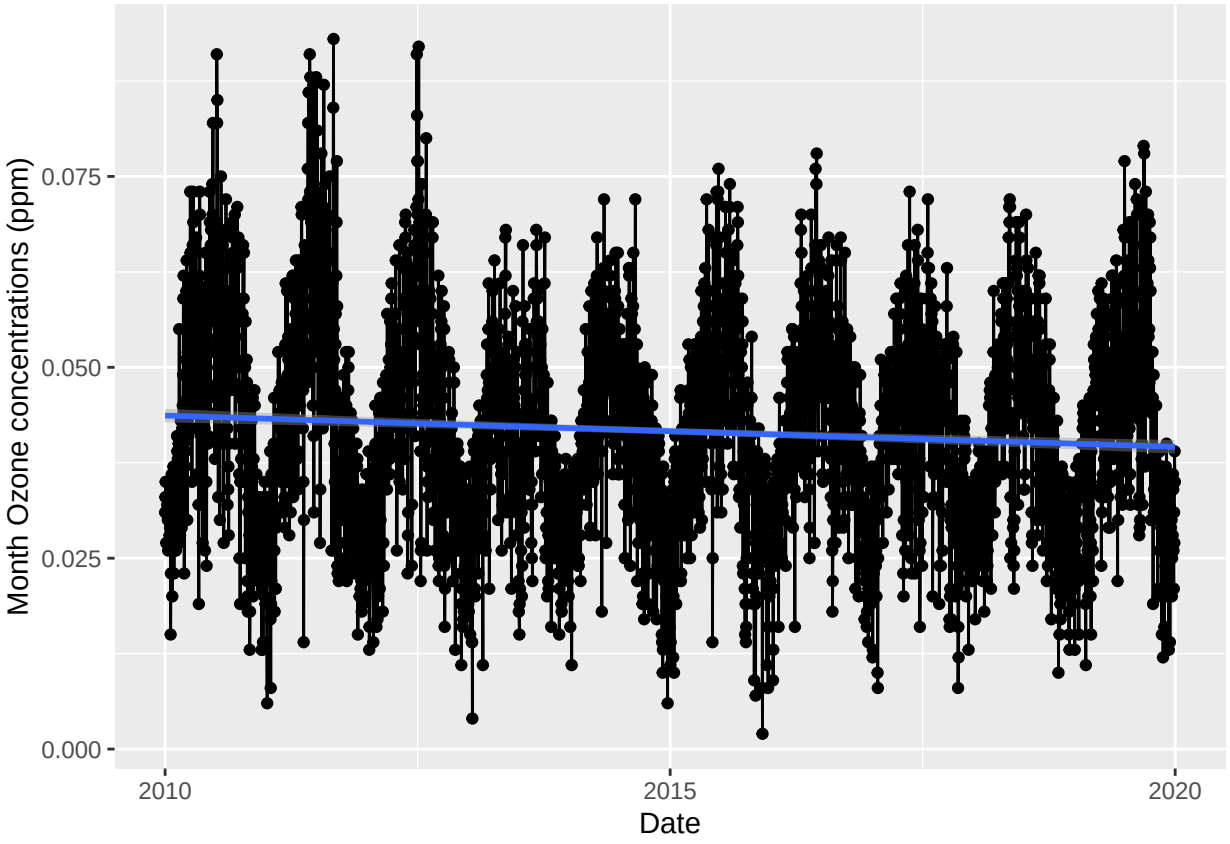
```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data: GaringerOzone.monthly.ts
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                      S varS    tau      z Pr(>|z|)
## Season 1:   S = 0   15  125  0.333  1.252  0.21050
## Season 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
```

Answer: Seasonal has selected the Mann-Kendall test for trend over other statistical trend tests because it can be used for indicators with varying units and time periods and does not require confidence intervals (which are not available for all indicators). In addition, the test can still be performed even when there are missing values in the set.

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

MonthlyOzone_data_plot <-
ggplot(GaringerOzone, aes(x = Date, y = Daily.Max.8.hour.Ozone.Concentration)) +
  geom_point() +
  geom_line() +
  ylab("Month Ozone concentrations (ppm)") +
  geom_smooth( method = lm )
print(MonthlyOzone_data_plot)
```

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



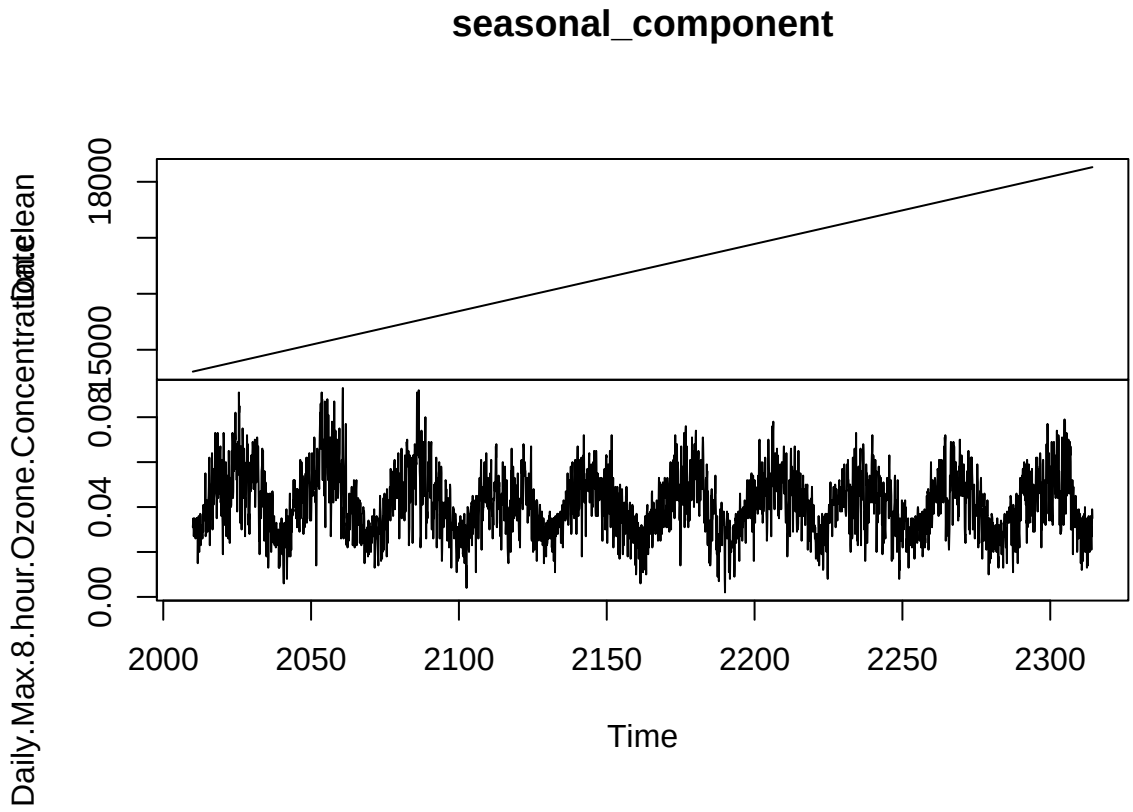
14. To accompany your graph, summarize your results in context of the research question. Include output from the statistical test in parentheses at the end of your sentence. Feel free to use multiple sentences in your interpretation.

    Answer: From 2020, the actual concentrations in ppm have a clear trend of gradually decreasing.It is clearly clearly seen that concentrations in ppm change greatly with the season.

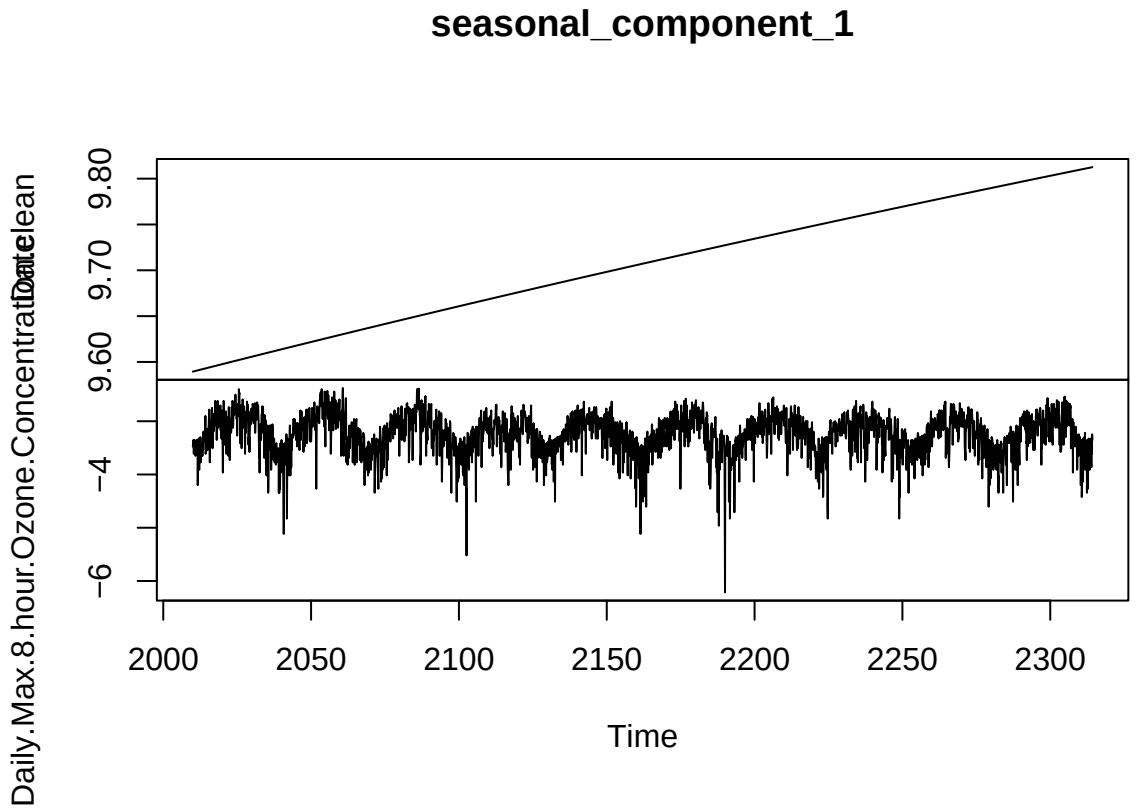
15. Subtract the seasonal component from the `GaringerOzone.monthly.ts`. Hint: Look at how we extracted the series components for the EnoDischarge on the lesson Rmd file.

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.

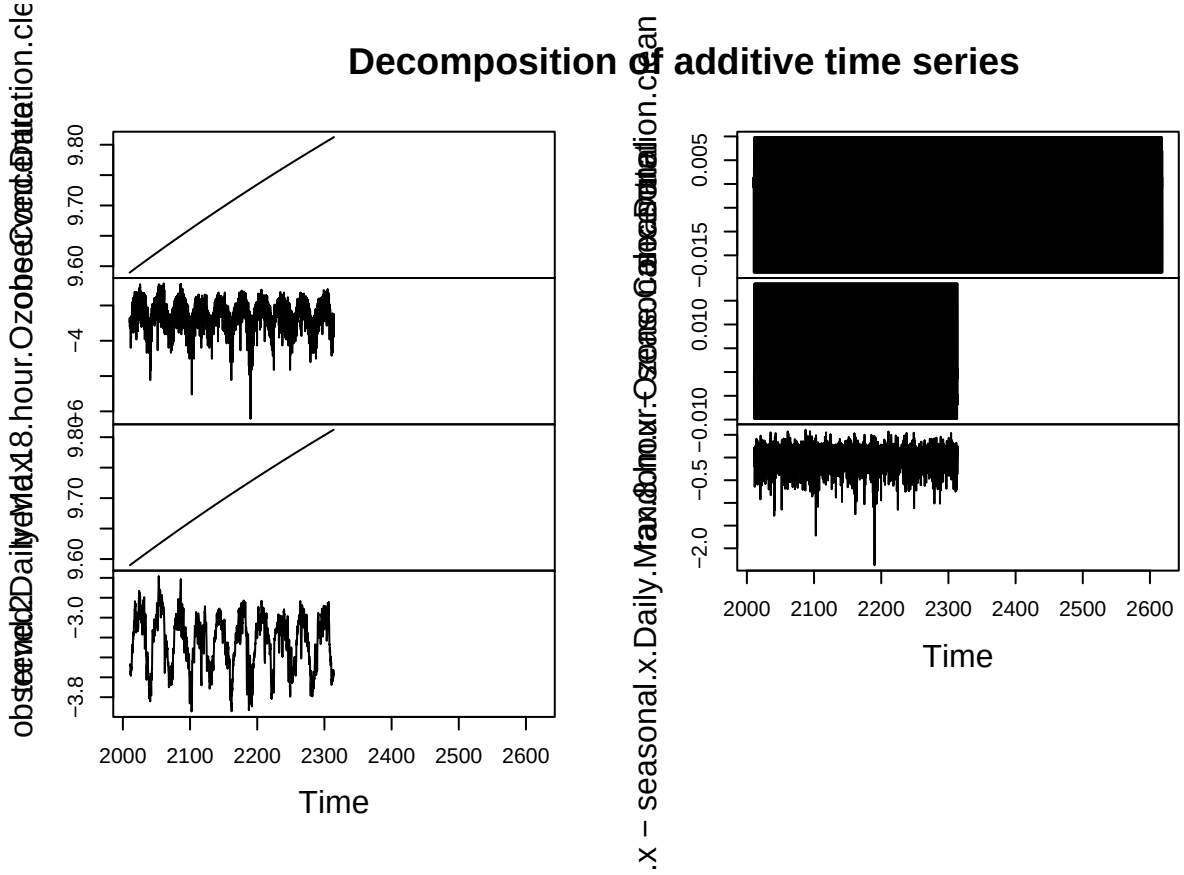
```
#15
GaringerOzone_5 <-   select(GaringerOzone_2_clean, Date, Daily.Max.8.hour.Ozone.Concentration.clean)

seasonal_component <- ts(GaringerOzone_5, frequency=12, start=c(2010,1))
plot.ts(seasonal_component)
```




```
seasonal_component_1 <- log(seasonal_component)
plot.ts(seasonal_component_1)
```




```
seasonal_component_2 <- decompose(seasonal_component_1)
plot(seasonal_component_2)
```




```
#16

GaringerOzone_monthly_trend_1 <- Kendall::SeasonalMannKendall(seasonal_component_1)


GaringerOzone_monthly_trend_1
```

```
## tau = -0.262, 2-sided pvalue =< 2.22e-16
```

```
summary(GaringerOzone_monthly_trend_1)
```

```
## Score =  -579902 , Var(Score) = 301373196
## denominator =  2213566
## tau = -0.262, 2-sided pvalue =< 2.22e-16
```

```
GaringerOzone_monthly_trend_2 <- trend::smk.test(seasonal_component_1)


GaringerOzone_monthly_trend_2
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  seasonal_component_1
```

```
## z = -33.404, p-value < 2.2e-16
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##         S      varS
##   -579890 301373193
```

```
summary(GaringerOzone_monthly_trend_2)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data: seasonal_component_1
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                          S     varS    tau       z   Pr(>|z|)
## Season 1:   S = 0  -48984 25279134 -0.264  -9.742 < 2.22e-16 ***
## Season 2:   S = 0  -49949 25279719 -0.270  -9.934 < 2.22e-16 ***
## Season 3:   S = 0  -47765 25279748 -0.258  -9.500 < 2.22e-16 ***
## Season 4:   S = 0  -48875 25279252 -0.264  -9.721 < 2.22e-16 ***
## Season 5:   S = 0  -48507 25032232 -0.263  -9.695 < 2.22e-16 ***
## Season 6:   S = 0  -49296 25032117 -0.268  -9.853 < 2.22e-16 ***
## Season 7:   S = 0  -47164 25032072 -0.256  -9.427 < 2.22e-16 ***
## Season 8:   S = 0  -46675 25031677 -0.254  -9.329 < 2.22e-16 ***
## Season 9:   S = 0  -46938 25031729 -0.255  -9.381 < 2.22e-16 ***
## Season 10:   S = 0 -46984 25031764 -0.255  -9.391 < 2.22e-16 ***
## Season 11:   S = 0 -48530 25031846 -0.264  -9.700 < 2.22e-16 ***
## Season 12:   S = 0 -50223 25031904 -0.273 -10.038 < 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Answer: