Assignment 8: Time Series Analysis

Hanbin Lyu

Fall 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

```
#1
#checking working directory
getwd()
```

```
## [1] "/home/guest/EDE_Fall2024"
```

```
#load package
library(tidyverse)
library(lubridate)
library(here)
library(zoo)
library(trend)

#set my own plot theme
my_theme = theme(
    line = element_line(color = "darkseagreen4", linewidth = 2,
                        linetype = "solid", lineend = "round"),
```

```
    rect = element_rect(fill = "honeydew",color = "darkolivegreen4",
                        linewidth = 1, linetype = "solid"),
    text = element_text(family = "serif", face = "plain",
                        color = "darkolivegreen",size = 12, hjust = 0.5,
                        vjust = 0.5, angle = 0, lineheight = 1.5),

    # Modified inheritance structure of text element
    plot.title = element_text(family = "serif", face = "bold",
                              color = "darkolivegreen", size = 16, hjust = 0.5,
                              vjust = 0.5, angle = 0, lineheight = 1.5),
    axis.title.x = element_text(family = "serif", face = "plain",
                              color = "darkolivegreen", size = 12, hjust = 0.5,
                              vjust = 0.5, angle = 0, lineheight = 1.5),
    axis.title.y = element_text(family = "serif", face = "plain",
                              color = "darkolivegreen", size = 12, hjust = 0.5,
                              vjust = 0.5, angle = 0, lineheight = 1.5),
    axis.text = element_text(family = "serif", face = "plain",
                            color = "darkolivegreen", size = 12, hjust = 0.5,
                            vjust = 0.5, angle = 0, lineheight = 1.5),

    # Modified inheritance structure of line element
    axis.ticks = element_blank(),
    panel.grid.major =  element_line(color = "gray85", linewidth = 0.5,
                                     linetype = "solid", lineend = "square"),
    panel.grid.minor =  element_blank(),

    # Modified inheritance structure of rect element
    plot.background = element_rect(fill = "ivory"),
    panel.background = element_rect(fill = "lightyellow"),
    legend.key = element_rect(fill = "honeydew", linewidth = 0.5),

    # Modifiying legend.position
    legend.position = 'right',

    complete = TRUE
    )
```

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
#upload those data
data1 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2010_raw.csv"),
  stringsAsFactors = TRUE)
data2 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2011_raw.csv"),
  stringsAsFactors = TRUE)
data3 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2012_raw.csv"),
  stringsAsFactors = TRUE)
```

```
data4 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2013_raw.csv"),
  stringsAsFactors = TRUE)
data5 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2014_raw.csv"),
  stringsAsFactors = TRUE)
data6 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2015_raw.csv"),
  stringsAsFactors = TRUE)
data7 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2016_raw.csv"),
  stringsAsFactors = TRUE)
data8 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2017_raw.csv"),
  stringsAsFactors = TRUE)
data9 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2018_raw.csv"),
  stringsAsFactors = TRUE)
data10 = read.csv(
  here("Data","Raw", "Ozone_TimeSeries", "EPAair_O3_GaringerNC2019_raw.csv"),
  stringsAsFactors = TRUE)

#combine the data
GaringerOzone = rbind(data1,data2,data3,data4,data5,data6,data7,data8,data9,data10)

#check how many observations and variables are there
dim(GaringerOzone)
```

```
## [1] 3589   20
```

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
#4
GaringerOzone1 = select(GaringerOzone, Date,
                        Daily.Max.8.hour.Ozone.Concentration,
                        DAILY_AQI_VALUE) %>%
  #select columns we are interested in
  mutate(Date = mdy(Date))
```

```r
  #set the Date column as a date class

#5
#create a sequence of dates from 2010-01-01 to 2019-12-31
Days = as.data.frame(seq(from = as.Date("2010-01-01"),
                         to = as.Date("2019-12-31"), by = "day"))

#rename the column to "Date"
colnames(Days) = "Date"

#6
#combine two dataset
finaldata = left_join(Days, GaringerOzone1, by = "Date")

#check how many columns and rows are there
dim(finaldata)
```

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

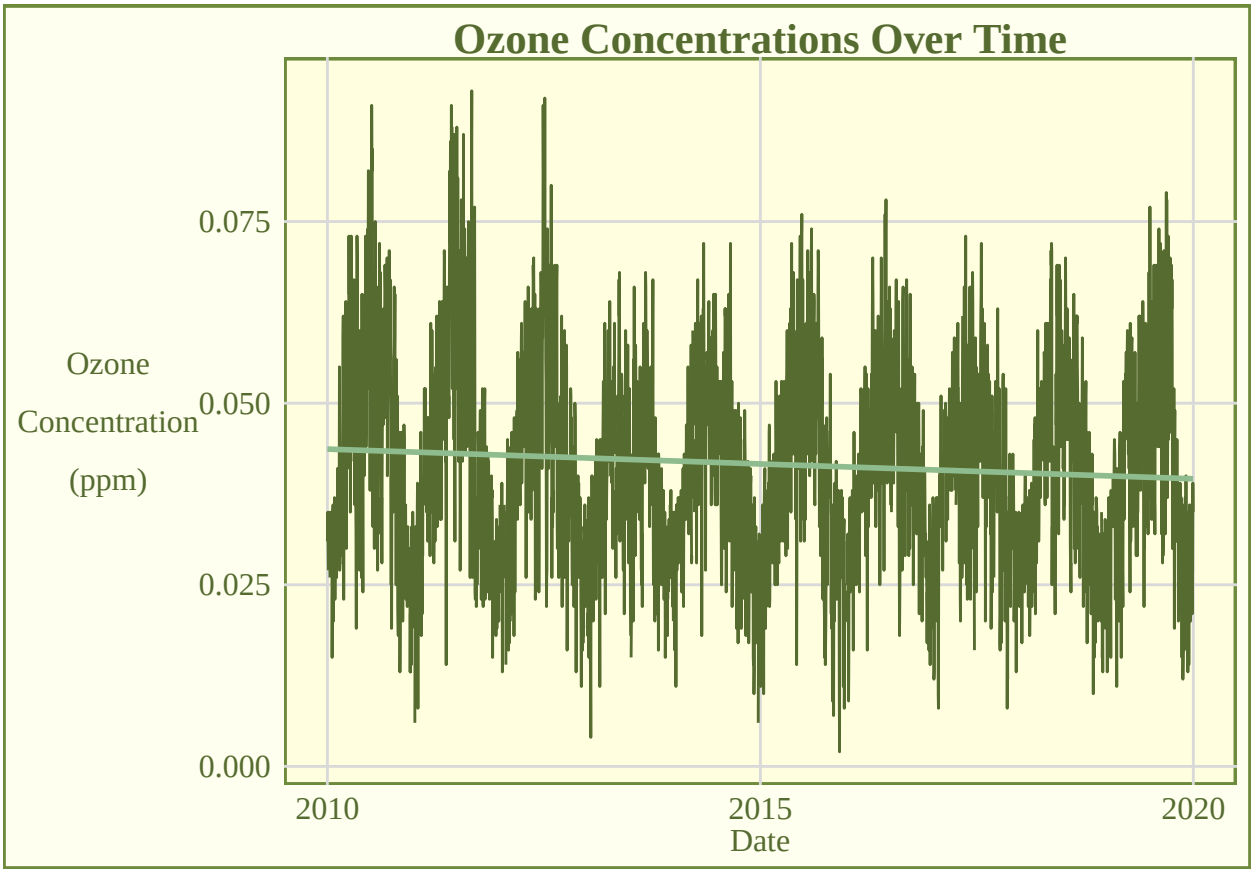
Visualize

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(finaldata, aes(x = Date, y = Daily.Max.8.hour.Ozone.Concentration)) +
  geom_line(color = "darkolivegreen") +  #line plot
  geom_smooth(method = "lm", color = "darkseagreen", se = FALSE) +
  #add linear trend line
  labs(
    title = "Ozone Concentrations Over Time",
    x = "Date",
    y = "Ozone\nConcentration\n(ppm)") +
  #add title and axis lables
  my_theme
```

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

```
## Warning: Removed 63 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```



Answer:the linear trend line appears to slope slightly downward, which suggests a decreasing trend in ozone concentration over time at Garinger High School from 2010 to 2019.

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
finaldata$Daily.Max.8.hour.Ozone.Concentration =
  na.approx(finaldata$Daily.Max.8.hour.Ozone.Concentration,
            x = finaldata$Date)
```

Answer:linear interpolation is chosen because it is simple, provides reasonable estimates, and avoids adding artificial fluctuations. It is a good fit for filling gaps in environmental data where daily changes are generally gradual.

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)

```r
#9
GaringerOzone.monthly = finaldata %>%
  mutate(Year = format(Date, "%Y"), Month = format(Date, "%m")) %>%
  #add columns for year and month
  group_by(Year, Month) %>%
  summarize(ozone_mean = mean(Daily.Max.8.hour.Ozone.Concentration))
```

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

```r
  #calculate mean ozone concentration for each month

#a column with each month-year combination set to the first day of the month
GaringerOzone.monthly = GaringerOzone.monthly %>%
  mutate(Date = as.Date(paste(Year, Month, "01", sep = "-")))

#check the resulting data frame
head(GaringerOzone.monthly)
```

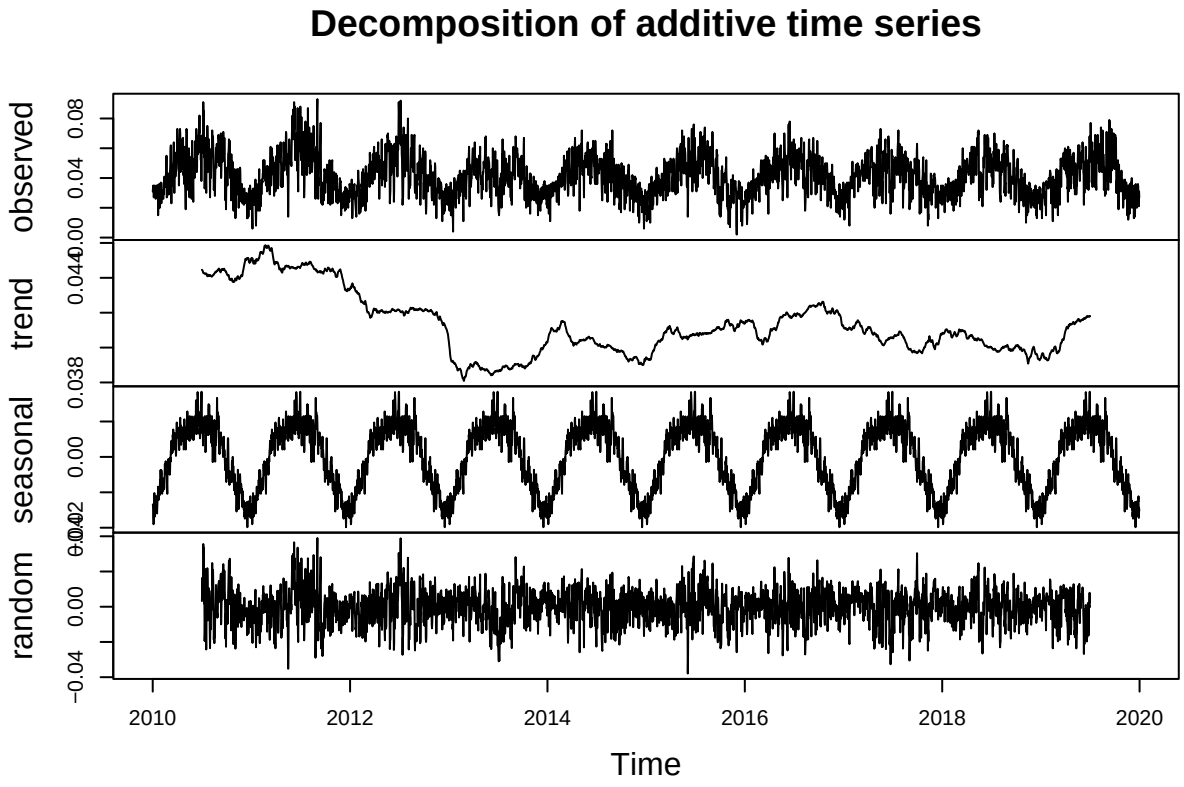
```
## # A tibble: 6 x 4
## # Groups:   Year [1]
##   Year  Month ozone_mean Date
##   <chr> <chr>      <dbl> <date>
## 1 2010  01        0.0305 2010-01-01
## 2 2010  02        0.0345 2010-02-01
## 3 2010  03        0.0446 2010-03-01
## 4 2010  04        0.0556 2010-04-01
## 5 2010  05        0.0466 2010-05-01
## 6 2010  06        0.0576 2010-06-01
```

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.

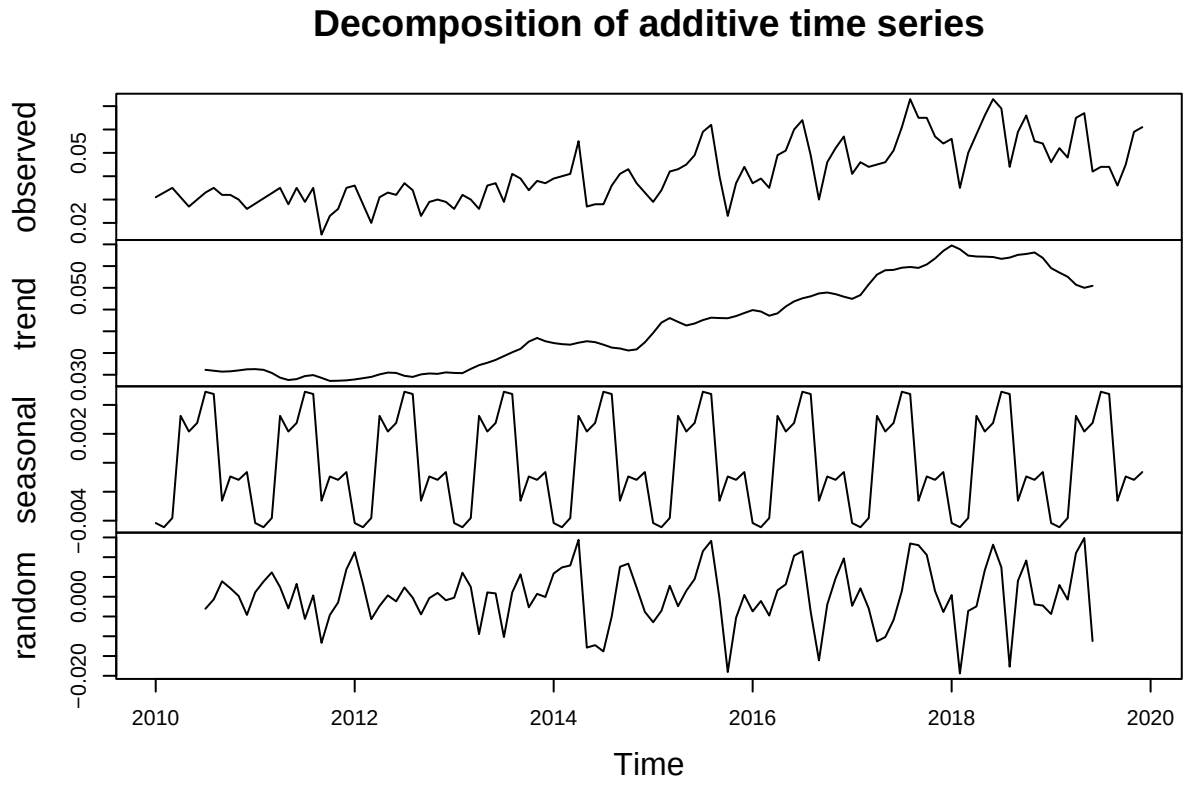
```r
#10
#generate daily time series object
GaringerOzone.daily.ts = ts(finaldata$Daily.Max.8.hour.Ozone.Concentration,
  start = c(2010, 1), end = c(2019, 365), frequency = 365)

#generate monthly time series object
GaringerOzone.monthly.ts = ts(finaldata$Daily.Max.8.hour.Ozone.Concentration,
  start = c(2010, 1), end = c(2019, 12), frequency = 12)
```

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

```r
#11
#decompose the daily time series
GaringerOzone.daily.decomp = decompose(GaringerOzone.daily.ts)
#plot the components of the daily decomposition
plot(GaringerOzone.daily.decomp)
```




```
#decompose the monthly time series
GaringerOzone.monthly.decomp = decompose(GaringerOzone.monthly.ts)
#plot the components of the monthly decomposition
plot(GaringerOzone.monthly.decomp)
```



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

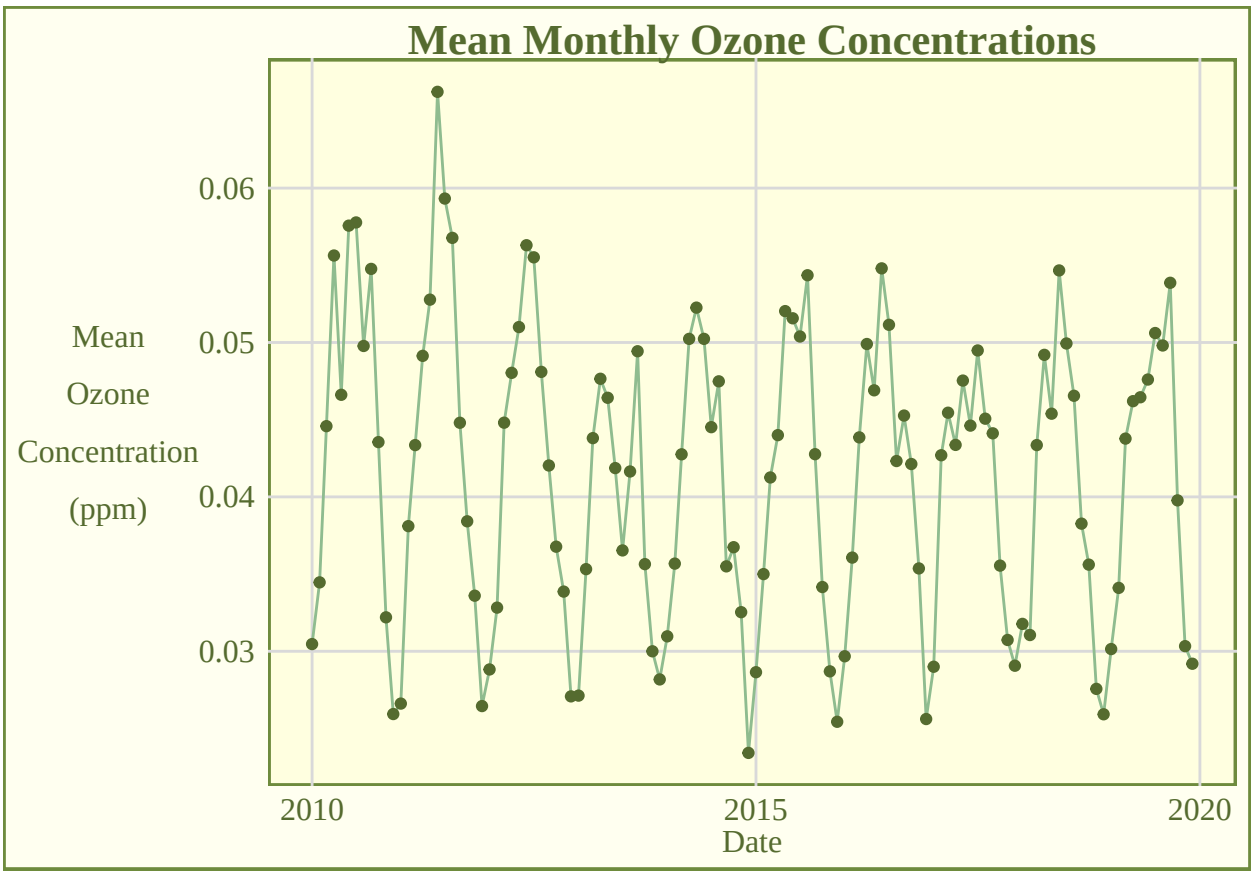
```
#12
#run the Mann-Kendall test
result_seasonal = smk.test(GaringerOzone.monthly.ts)
print(result_seasonal)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  GaringerOzone.monthly.ts
## z = 8.4714, p-value < 2.2e-16
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##    S varS
##  328 1490
```

Answer:the Mann-Kendall test is ideal because ozone levels follow a yearly pattern due to seasonal changes. This test can detect a real long-term trend without being confused by monthly fluctuations.

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
#plot mean monthly ozone concentrations
ggplot(GaringerOzone.monthly, aes(x = Date, y = ozone_mean)) +
  geom_line(color = "darkseagreen") +
  geom_point(color = "darkolivegreen") +
  labs(
    title = "Mean Monthly Ozone Concentrations",
    x = "Date",
    y = "Mean\nOzone\nConcentration\n(ppm)") +
  my_theme
```



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 mean monthly ozone concentrations over the 2010s at this station show some seasonal fluctuations, but there does not appear to be a strong, clear long-term trend in ozone levels. The values oscillate each year, likely due to seasonal factors, but the overall levels remain relatively stable across the decade.

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
#subtract the seasonal component
GaringerOzone.monthly.subtract =
  GaringerOzone.monthly.ts -
  GaringerOzone.monthly.decomp$seasonal

#16
#run the Mann-Kendall test
result_non_seasonal = smk.test(GaringerOzone.monthly.ts)
print(result_non_seasonal)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  GaringerOzone.monthly.ts
## z = 8.4714, p-value < 2.2e-16
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##    S varS
##  328 1490
```

Answer:both the seasonal and non-seasonal Mann-Kendall tests show a significant trend in ozone levels, with very low p-value and high Z-score. This suggests that the trend is real and not just due to seasonal patterns. Ozone levels have genuinely changed over time at this station, independent of seasonal fluctuations.