Regional Analysis

Marc Los Huertos

1/1/2023

Introduction

Background

Air qualily data varies seasonally, regionally, and with particular events. These create a distribution of values, some of which pose potential health effects.

Goals

We will collect EPA air quality data to compare with the local data we collect to assses if our data fall within the confidence intervals of the EPA database.

Rationale

By collecting and analyzing air quality data, we will be able to estimate the central tendencies of the data. With these estimates, we can determine if our sensor data fall within these estimates.

Learning Outcomes

We have learned how to manipulate data in R (using the Rstudio) to analyze data before. In this case, we will

Grading – low stakes

As an assignment with relatively low stakes, 10 points, you will be assessed using the following criteria:

1. Did you collect 5 years of PM2.5 from a nearby air quality station.
2. Report the mean, standard deviation, and 95% confidence interval.
3. Plot 5-years of data and create a threshold line based on state and federal standards for PM2.5. Please include the folowing:

- Rotated y-axis
- Appropriate axis labels
- Scatter plot with added EPA or state health limits.
- Estimate the mean, S.D. and 95% confidence intervals for each month.

4. Using the peer reviewed literature, discuss the implications of the results. I suggest several (more than 2 and less than 9) peer reviewed articles and a few paragraphs (more than 1 and less than 4) discussion the regional causes and implications of the results.

5. Submit via Sakai as a pdf.

Assessing EPA Data

We will use the EPA website (https://www.epa.gov/outdoor-air-quality-data/download-daily-data)to collect the data. Select a location and download 5 years of data. From what I can tell, we'll need to download each year separately and then combine them in R.

NOTE: As you can probably appreciate, every project requires some clean up.

Upload to Rstudio Server Folder

First, we'll upload the data to Rstudio. After getting the data from the EPA downloaded, you can upload the data to R.

NOTE: We are not going to be using the github site, so you can start a new project with a new directory. We won't be needing the collaboration work that we used in our previous project.

Read Data

First, we read data into R, using read.csv() and create an dataframe for each year. I prefer to do this in two steps.

1. Create file path that points to the csv file.

```
SCZ2022.csv <- "/home/mwl04747/EJnPi/Air_Quality_Data_Analysis/ad_viz_plotval_data2022.csv"
SCZ2021.csv <- "/home/mwl04747/EJnPi/Air_Quality_Data_Analysis/ad_viz_plotval_data2021.csv"
SCZ2020.csv <- "/home/mwl04747/EJnPi/Air_Quality_Data_Analysis/ad_viz_plotval_data2020.csv"
SCZ2019.csv <- "/home/mwl04747/EJnPi/Air_Quality_Data_Analysis/ad_viz_plotval_data2019.csv"
SCZ2018.csv <- "/home/mwl04747/EJnPi/Air_Quality_Data_Analysis/ad_viz_plotval_data2018.csv"
```

2. Read the csv files into dataframes

```
SCZ2019 = read.csv(SCZ2019.csv); SCZ2020 = read.csv(SCZ2020.csv)
SCZ2018 = read.csv(SCZ2018.csv)
```

3. Always a good idea of checkin gwhat we created:

```
str(SCZ2020)
```

4. Bind the files together. In this case, we "row bind' ', rbind(), where each dataframe bound together. This works when all the columns are the same – so if you get an error look at the structure of the files to make sure have the same voarabiel names.

```
SCZ=rbind(SCZ2018, SCZ2019, SCZ2020)
```

Fix Dates

As usually, we need to fix the data to make sure R can read them properly.

1. Create character string of dates. When imported they are defined as factors – so first by strip them of this format and then redefine.

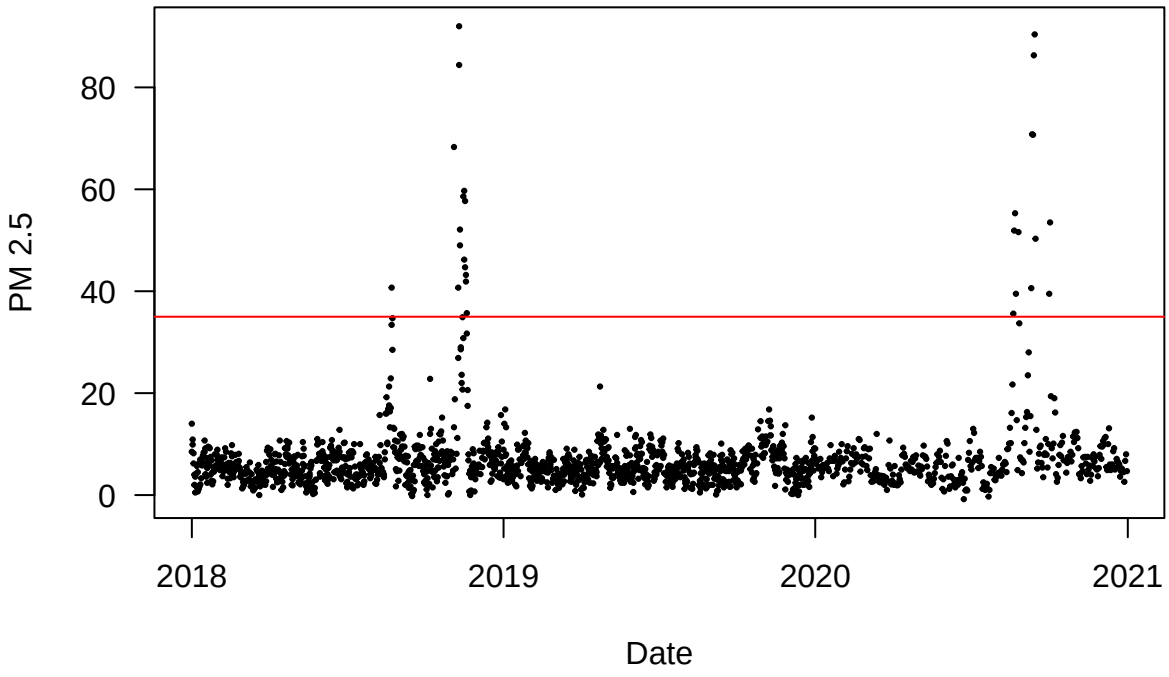
```r
Date.char = as.character(SCZ$Date)
# testing to make sure this works...
# as.Date(Date.char, format="%m/%d/%Y")
SCZ$Date = as.Date(Date.char, format="%m/%d/%Y")

str(SCZ)
names(SCZ)
```

Okay, that seems to work.

Graphic Analysis

I have two stations together, might need to decide which one to focus on!

Remember the syntax plot(y~x, data=dataframe)! And you can use abline to put the EPA standard on there, abline(h=some value), where the line is a "horizontal line" that equals some value that you specify.



Estimate the Central Tendency for each Month

Next we will determine the central tendency, e.g. average, standard deviation.

So, we will use some packages or libraries to make our job easier. To add packages that are not installed be default, use the `Packages'' tab and click on`Install'' – and then search for dpylr and lubridate to install. NOTE: If you have some response that R needs to restart let me know, there is some bug for some installations not allowing the package to be installed correctly – but I didn't have the problem, so we need to reach out to POM-ITS to sort it out (fun stuff).

the dplyr is described as "a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges.''

The lubridate makes it easier manipulate date-times.

We call these libraries in the rcode as below.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

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

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
str(SCZ)

tmp1 <- mutate(SCZ, Date = ymd(Date),Year = year(Date), Month = month(Date))
tmp2 <- group_by(tmp1, Month, Year)
Monthly <- summarise(tmp2, result = mean(Daily.Mean.PM2.5.Concentration) )
```

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

We can do the same thing using %>%. This function as has no builtin meaning but the user (or a package) can define operators of the form %whatever% in any way they like. For example, this function will return a string consisting of its left argument followed by a comma and space and then it's right argument.

```
Monthly <- SCZ %>%
  mutate(Date = ymd(Date),Year = year(Date), Month = month(Date)) %>%
  group_by(Month) %>%
  summarise(mean = mean(Daily.Mean.PM2.5.Concentration),
            sd = sd(Daily.Mean.PM2.5.Concentration),
            N = length(Daily.Mean.PM2.5.Concentration) )
```

Estimating Confidence Intervals

Although the standard deviation describe variability, the statistic is not explicilty associated with a probability. We cna convert these to probabilities by mulitplying by a theoretical probability distribution.

```
qnorm(.975)
```

```
## [1] 1.959964
```

```
Monthly$UCL95 = Monthly$mean + qnorm(.975)*Monthly$sd/sqrt(Monthly$N)
Monthly$LCL95 = Monthly$mean - qnorm(.975)*Monthly$sd/sqrt(Monthly$N)
```

Table of confidence intervals...

NOTE: At some point, I'll convert the numbers to months – but for now you can translate them for yourself.

```
library(xtable)
Monthly$Months
```

```
## Warning: Unknown or uninitialised column: `Months`.
```

NULL

```
print(xtable(Monthly), include.rownames=FALSE)
```

% latex table generated in R 4.2.1 by xtable 1.8-4 package % Fri Dec 15 20:05:54 2023

Month	mean	sd	N	UCL95	LCL95
1.00	6.17	2.78	151	6.62	5.73
2.00	5.60	1.99	136	5.93	5.27
3.00	4.07	2.08	149	4.40	3.73
4.00	5.77	2.95	149	6.24	5.29
5.00	4.93	2.58	155	5.33	4.52
6.00	5.91	2.84	150	6.36	5.45
7.00	5.20	2.48	150	5.60	4.81
8.00	8.92	9.69	154	10.45	7.39
9.00	8.77	13.29	150	10.90	6.64
10.00	7.57	6.08	140	8.57	6.56
11.00	13.51	16.08	140	16.17	10.84
12.00	6.09	3.04	144	6.59	5.59

Hypothesis Testing

Later Stages – This section is for when we have our pi data (and not done yet :-)

What if want to evalaute a single value and determine if the it the values in inside the confidence intervals – you can use the table above and determine the value, if the parameter is within the expected mean. We'll use the t-test for the test.

If we obtain a mean concentration of 14 using our sensors, we can easily test the value using t.test and all the Septmber values. So, we'll create a vector of all the September readings.

```
September = subset(SCZ, select=Daily.Mean.PM2.5.Concentration, subset = month(SCZ$Date)==9)
```

The null hypothesis is that there is no difference between the mean from the sensor and the values currated by the EPA.

```
t.test(September, mu=14)
```

```
##
##  One Sample t-test
##
## data:  September
## t = -4.8182, df = 149, p-value = 3.533e-06
## alternative hypothesis: true mean is not equal to 14
## 95 percent confidence interval:
##   6.626964 10.915703
## sample estimates:
```

```
## mean of x
##  8.771333
```

More likely that we'll have bunch of values from our sensor – not just a mean, then were comparing lots of sensor measurements to the EPA data.

Comparing Sensor Values with EPA Data

First, I'll create simulated data to show how this might be done:

```
simulateddata=rnorm(40, mean=14, sd=12)
```

```
t.test(September, simulateddata)
```

```
##
##  Welch Two Sample t-test
##
## data:  September and simulateddata
## t = -2.4533, df = 69.184, p-value = 0.01668
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -9.4453946 -0.9734709
## sample estimates:
## mean of x mean of y
##  8.771333 13.980766
```