Maxwell Reeser

EXST 7087 Final Project Report

1. Introduction

For this project I chose to investigate some EPA pollution data that I've used before. I previously used it in a project correlating parish/county level pollution with various types of mortality. This time I decided to see if I could do some time series analysis to see if pollution levels were significantly different between two time periods. This is because I recently learned that differences of averages are normally distributed, so I thought I'd try fitting models for as many counties as I could to generate a sample of models for each time period. After the models were generated I could predict forward using each model, determine averages of the generated dates for each time period, and test whether or not the averages were significantly different. This significance test would then serve as a proxy way of determining whether or not pollution levels were significantly different in these two time periods.

2. Data & Data Cleaning

Although I started out with some PM10 particulate pollution data, I eventually changed it to Ozone ppm pollution data. Both sets were provided by the EPA's website[1]. The ozone data had the following fields:

StateCode – Code representing which state this refers to,

CountyCode – Code representing which county within the state it is (not globally unique), SiteNum – Code for site,

ParameterCode – Code for pollutant,

POC – Integer value,

Latitude,

Longitude,

Datum – String value,

ParameterName – Pollutant name,

SampleDuration – Collection period for observation,

PollutantStandard – Standard for pollutant observation,

DateLocal date,

UnitsOfMeasure – Density of pollutant in atmosphere,

EventType – String value,

ObservationCount – How many observations are made,

ObservationPercent – Float value,

ArithmeticMean – Observed pollutant measurement,

FirstMaxVal – Float value,

FirstMaxHour – Int value,

AQI – Air Quality Index,

MethodCode – Code representing data gathering method,

MethodName – Name of data gathering method,

LocalSiteName – Abbreviated name of local site,

Address – Address of local site,

StateName,

CountyName,

CityName,

CBSAName – Site unique identifier,

DateOfLastChange – Date last modified in EPA settings

In order to get the necessary data at the quality desired for these models, I downloaded the daily county level ozone data available on the EPA website[1]. I chose to download the data from 1996 to 2004. I then imported each dataset into a master database using pgAdmin4. As I imported each file, I found data which was not usable and caused errors in the imports, requiring some modification of the data so it could be imported. In most of the situations where a segment of data was unusable, for missing or null data values for example, that segment of data was simply deleted. These segments were all small relative to the size of the entire dataset.

Once all of the data was imported, I was able to generate a daily average for each county. Initially I attempted to use the raw daily values for each data collection point (weather station), in order to have more consistent data for training. However, it was my goal to have continuous data collection for the entire eight year period, and using only the individual weather stations narrowed that pool to almost no viable locations. However, using daily averages on the county level was quite beneficial, and yielded a pool of 51 counties which had virtually continuous data for the entire eight year period.

3. Method Selection

I chose to use a time series analysis of the data, using the nnetar function in R. This is a single layer neural network with special functionality for time series. This choice was made as an avenue to becoming more familiar with analyzing time series data in a more independent setting than we found in class.

For those unfamiliar, a neural network is a graph which has weights on all edges, and applies those weights to input data represented as vertices to calculate an output value, represented with another vertex. They are usually organized in layers, and nnetar is a neural network with a single hidden layer (layer between the input and output layers).

A time series is essentially any data which is associated with a timestamp, which is collected at multiple times, and may vary according to predictable patterns over time.

4. Literature Review

Ozone levels became of greater concern in the 90s and 00s as cities like Los Angeles began to have to deal with more and more smog, and other pollutants. The Environmental Protection Agency in the United States has been tracking levels since the 80s, and there were political pushes in the 80s and 90s to reduce those levels, including amendments to the Clean Air Act in 1990, which brought more requirements for reduction of pollution in the form of atmospheric ozone, and other pollutants. The EPA and others have done research on ozone levels and its detrimental effects, and have tracked ozone levels over time to verify compliance and safeguard the environment.

EPA data shows that after 2002 in the United States there was a substantial drop-off in atmospheric ozone, presumably as a result of a decade of phasing out over-polluting sources like air conditioners using CFCs as refrigerants. It is not clear if anyone has performed similar experiments using time series NNs to detect a statistical difference between the levels around this time.

5. Model Generation

With the data selected using the SQL database I could move onto modeling the time series. The first model I ran was an even split of the eight year period into two four year periods, starting January first, 1997, and January first 2001, respectively. At this point, I realized that the PM10 particulate data I started out with did not really have a cyclical quality. Luckily, the EPA data is almost all formatted the same, so I swapped to Ozone observations, and was able to, with minimal manipulation, make use of the data pipeline I had created for the particulate data.

Trying time series modeling with the Ozone data was much more successful, and an obvious yearly trend popped out.

After I had done some exploratory analysis for the ozone data then, I was ready to move into generating models for every county in the set of 51 that had continuous data, on the given split. This step involved a lot of troubleshooting R code, and as a result took much longer than expected. Once I had successfully written the code for generating these models I could proceed to performing the means test. In order to do that, I had to generate a month's worth of data, using the models, and format them so I could get every generated January first in one row, and every January second in another, etc. etc.. With this, I could calculate averages for each training set, for each day. Once I had those averages, I could calculate their difference, and check that difference against the standard deviation for the distribution of differences, to see if the confidence interval included zero. That is, to see if it was plausible that the difference of these averages was zero, or, that they were the same. Once that code was written I was able to reformat my code to allow me to perform these analyses for any arbitrary split I supplied, and on every day in the generated month.

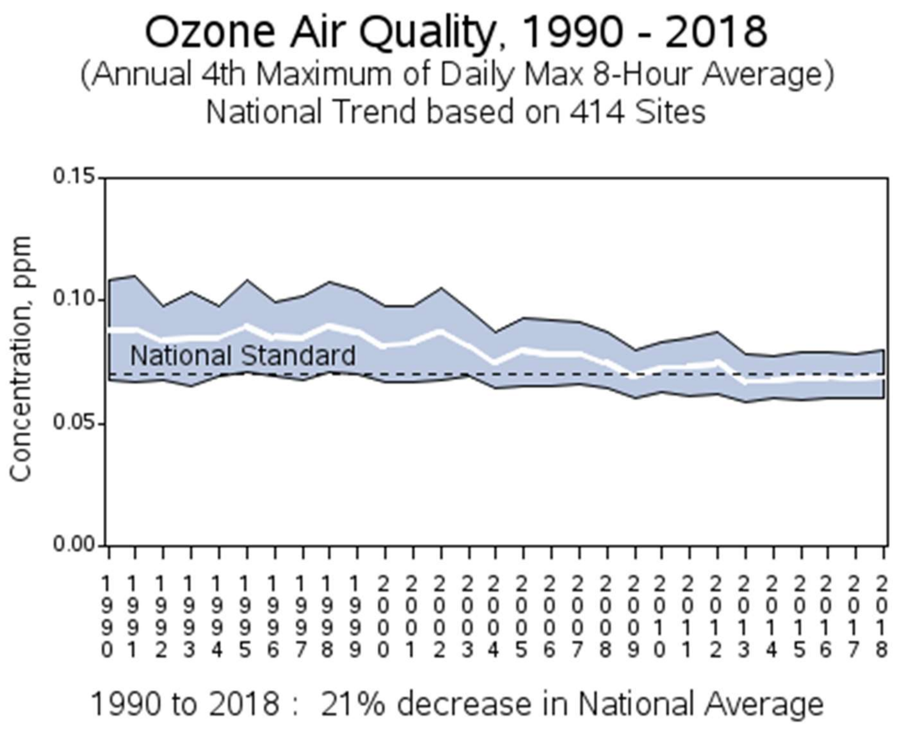
That process did not take that long, so I moved on to testing several different splits, and the results are below.

Model Split Name	**Left period end date**	**Right Period Start Date**	**Proportion Significantly Different**	**Days significantly diff.**	**Days not significantly diff.**	**Graphical Representation of split**
Even_split	2000-12-31	2000-12-31	0.355	11	20	🟩🟩🟩🟩🟧🟧🟧🟧

25_75	1998-12-31	1998-12-31	0.226	7	24	🟩🟩🟧🟧🟧🟧🟧🟧
75_25	2002-12-31	2002-12-31	0.516	16	15	🟩🟩🟩🟩🟩🟩🟧🟧
7_to_1	2003-12-31	2003-12-31	0.935	29	2	🟩🟩🟩🟩🟩🟩🟩🟧
1_to_7	1997-12-31	1997-12-31	0.387	12	19	🟩🟧🟧🟧🟧🟧🟧🟧
Even_split_4_year_gap	1998-12-31	2002-12-31	0.0645	2	29	🟩🟩🟦🟦🟦🟦🟧🟧
66_33_2_year_gap	2000-12-31	2002-12-31	0.0645	2	29	🟩🟩🟩🟩🟦🟦🟧🟧
Late_even_split	2003-12-31	2003-12-31	0.935	29	2	🟦🟦🟦🟦🟩🟩🟧🟧
Late_75_25	2003-12-31	2003-12-31	0.935	29	2	🟦🟦🟦🟦🟩🟩🟩🟧
Late_25_75	2001-12-31	2001-12-31	0.935	29	2	🟦🟦🟦🟦🟩🟧🟧🟧

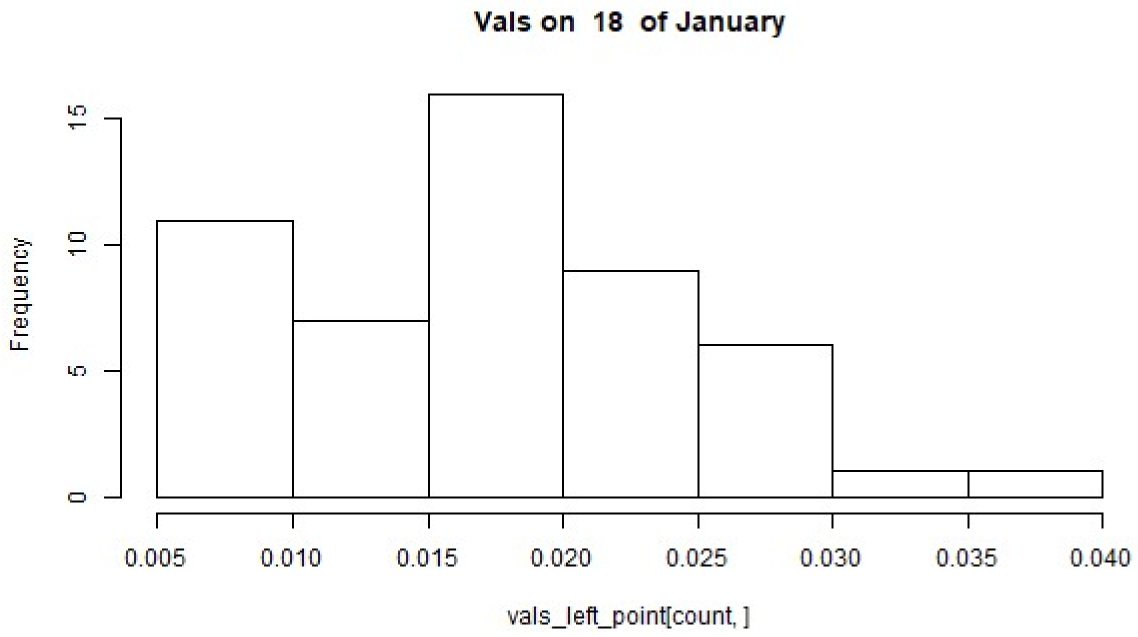
Left_625	2001-12-31	2001-12-31	.484	15	16	🟩🟩🟩🟩🟩🟧🟧🟧

In the chart, we can see that the 7_to_1 split was the first split with a majority of days with a significant difference. From there, the split where only 2001,2002,2003, and 2004 had all splits register as significant. Thinking that the earlier split tests may have missed something with the left_625 split, I then tried to fit that split. Unfortunately, this did not register the difference, either. Consider the below EPA plot.

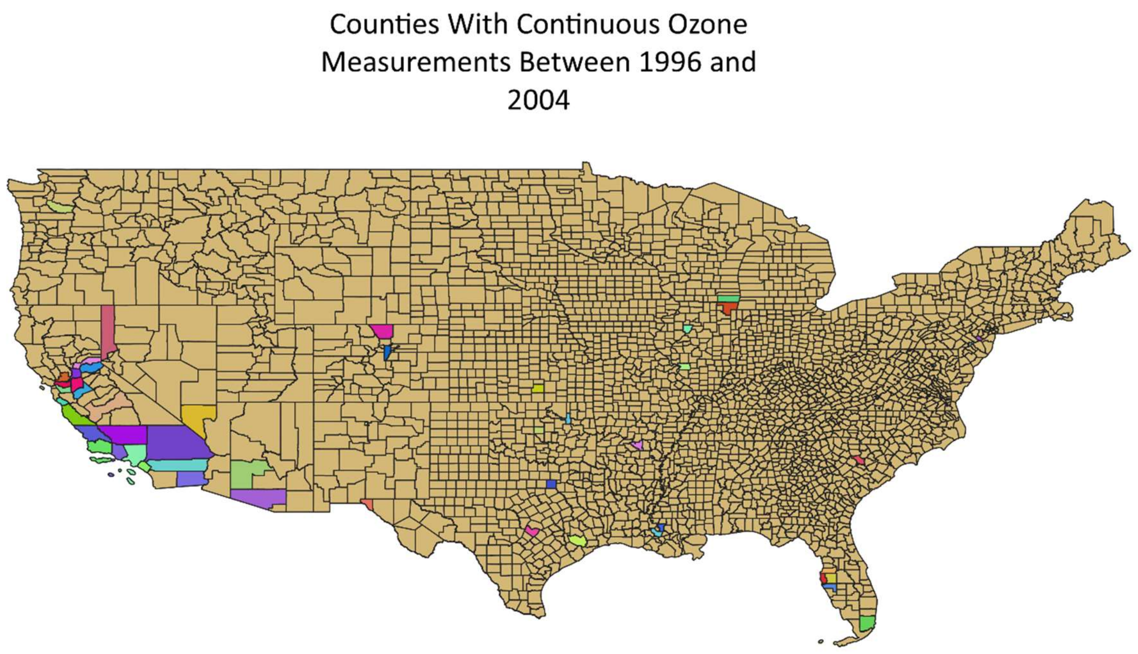


If I had included more data beyond 2004, the first even split may have registered a difference, since the consistently lower ppm counts would have helped create a consistently lower average to be tested against.

We do have to consider whether the samples used to create the averages were normally distributed, and whether they were independent or not. The plot below shows the values generated for the 18th of January using the 7_to_1 split, and is fairly typical of the histograms between all 51 models. These samples are fairly skewed, so they are not perfectly normal, but for the size of the sample, the deviation is not extreme. We also have to ask whether the elements of the samples are independent of one another as well. Since all of them have very good data continuity it is possible that they represent their own slice of the environmental landscape, perhaps wealthier areas with greater access to products with lower ozone emissions, for example, than the rest of the country.



Another concern is that this analysis could fail to be representative of the entire country. However, with the following map, we can see that although 19 of the counties represented are from California, the rest are fairly evenly distributed across the country.


Counties With Continuous Ozone Measurements Between 1996 and 2004

6. Conclusion

This project required a lot of work on the data cleaning side, and using R, but the results suggest with reasonable confidence that there was a drop in ozone levels across the US centered around 2002. We were able to show this using times series in R across 51 counties with continuous data from the period 1996 to 2004.

Bibliography:

US Environmental Protection Agency. Air Quality System Data Mart [internet database] available via https://www.epa.gov/airdata. Accessed May 8*, 2020.*

Ozone Air Quality, 1990-2018. 2020. Environmental Protection Agency. https://www.epa.gov/air-trends/ozone-trends#oznat. Accessed 8 May 2020.

Appendix:

```
 library(easypackages) # this makes calling other packages ... easy using the next line to put
them all in one process,

libraries("plyr","reshape2","ggplot2","readxl","foreign","forecast","tseries","lubridate","xts","ts
box","fpp2") # these are the packages needed for this code to run. They are all loaded here.


source("./run_models.R")

ozone = read.csv("./ozone/cleaned_sorted_ozone_1996_2004.csv",header=T)

ozone$datelocal <- as.Date(ozone$datelocal)

ozone2 <- subset(ozone, datelocal>=as.Date('1997-01-01'), c('datelocal', 'arithmean', 'concat'))

#ozone2_ <- subset(ozone2, concat=='ArizonaMaricopa')


setwd("C:/Users/Maxwell/Documents/LSU/Classes/EXST/EXST7087/PROJECT")

#ozone2 = ozone2_

# Original run

run_models(ozone2, "2000-12-31","2000-12-31", "even_split")

check_models(ozone2, "even_split")


run_models(ozone2, "1998-12-31","1998-12-31", "25_75")

check_models(ozone2, "25_75")


run_models(ozone2, "2002-12-31","2002-12-31", "75-25")

check_models(ozone2, "75-25")


run_models(ozone2, "2001-12-31","2001-12-31", "left_625")

check_models(ozone2, "left_625")



run_models(ozone2, "2003-12-31","2003-12-31", "7_to_1")
```

```
check_models(ozone2, "7_to_1")

run_models(ozone2, "1997-12-31","1997-12-31", "1_to_7")
check_models(ozone2, "1_to_7")

run_models(ozone2, "1998-12-31","2002-12-31", "even_split_4_year_gap")
check_models(ozone2, "even_split_4_year_gap")

run_models(ozone2, "2000-12-31","2002-12-31", "66_33_2_year_gap")
check_models(ozone2, "66_33_2_year_gap")



ozone3 = subset(ozone2, datelocal>=as.Date('2001-01-01'))

run_models(ozone3,"2002-12-31","2002-12-31","late_even_split")
check_models(ozone3, "late_even_split")

run_models(ozone3,"2003-12-31","2003-12-31","late_75_25")
check_models(ozone3, "late_75_25")

run_models(ozone3,"2001-12-31","2001-12-31","late_25_75")
check_models(ozone3, "late_25_75")
```

run_models.R

```
run_models <- function(df, left_date_bound, right_date_bound, name) {
  models_left <- list(c())
  models_right <- list(c())
  dir.create(paste("./Models/",name,sep=""))
  workpath = getwd()
  newpath = paste(workpath,"/Models/",name, sep = "")
  (newpath)
  setwd(newpath)
  count <- 1
  for(i in seq_along(levels(df$concat))){
    level_name = levels(df$concat)[count]
    level_data = subset(df, concat==level_name, c('datelocal','arithmean'))
    train_left = subset(level_data, datelocal<=as.Date(left_date_bound)) #, 
c('arithmean'))[['arithmean']]
    train_right = subset(level_data, datelocal>as.Date(right_date_bound))  #, 
c('arithmean'))[['arithmean']]

    mean_train_left=ts(train_left$arithmean,start=c(1997,1,1),frequency=365.25)
    mean_train_right=ts(train_right$arithmean,start=c(2001,1,1),frequency=365.25)

    fit_left=nnetar(mean_train_left,p=30,P=,Size=10,repeats=15,lambda = "auto")
    fit_right=nnetar(mean_train_right,p=30,P=,Size=10,repeats=15,lambda = "auto")

    print(paste("County: ", count,  level_name))
    left_filename = paste(level_name,"_left_",name,".rds",sep="")
    right_filename = paste(level_name,"_right_",name,".rds",sep="")
    saveRDS(fit_left,left_filename)
    saveRDS(fit_right,right_filename)
    models_left[[1]][count] <- fit_left
    models_right[[1]][count] <- fit_right
    pred=forecast(fit_left,31)
    autoplot(pred)
    predictions=pred$mean
    autoplot(predictions)

    count <- count+1
  }
  print(models_left)



  count <- 1
  models_left <- list(c())
  models_right <- list(c())
  for(i in seq_along(levels(df$concat))){
    level_name = levels(df$concat)[count]
    fit_left=readRDS(paste(level_name,"_left_",name,".rds",sep=""))
    fit_right=readRDS(paste(level_name,"_right_",name,".rds",sep=""))

    models_left[[1]][count] <- fit_left
    models_right[[1]][count] <- fit_right

    count <- count+1
  }
  setwd(workpath)
}

check_models <- function(df, name){
  count <- 1
```

```
models_left <- list(c())
models_right <- list(c())
for(i in seq_along(levels(df$concat))){
  level_name = levels(df$concat)[count]
  path=paste("./Models/",name,"/",level_name,"_left_",name,".rds",sep="")
  fit_left=readRDS(path)
  fit_right=readRDS(paste("./Models/",name,"/",level_name,"_right_",name,".rds",sep=""))

  models_left[[1]][count] <- fit_left
  models_right[[1]][count] <- fit_right

  count <- count+1
}

count <- 1
pred_left_1 <- list(c())
pred_left_2 <- list(c())
pred_left_3 <- list(c())
pred_right_1 <- list(c())
pred_right_2 <- list(c())
pred_right_3 <- list(c())
for(i in seq_along(levels(df$concat))){
  #mean_train_left=ts(train_left$arithmean,start=c(1997,1,1),frequency=365.25)
  #mean_train_right=ts(train_right$arithmean,start=c(2001,1,1),frequency=365.25)

  fit_left=models_left[[1]][count]
  fit_right=models_right[[1]][count]
  pred_97=forecast(fit_left[[1]],31)
  pred_val_left = data.frame(summary(pred_97))
  pred_01=forecast(fit_right[[1]],31)
  pred_val_right = data.frame(summary(pred_01))
  if( count == 1){
    pred_left = pred_val_left

    pred_right = pred_val_right
  }  else {
    pred_left = pred_left+pred_val_left

    pred_right = pred_right+pred_val_right
  }

  pred_val_right = subset(pred_val_right, T, c('Point.Forecast','Lo.95', 'Hi.95'))
  pred_val_left = subset(pred_val_left, T, c('Point.Forecast','Lo.95', 'Hi.95'))
  pred_left_1[[1]][count] <- list(pred_val_left$Point.Forecast)
  pred_right_1[[1]][count] <- list(pred_val_right$Point.Forecast)
  pred_left_2[[1]][count] <- list(pred_val_left$Lo.95)
  pred_right_2[[1]][count] <- list(pred_val_right$Lo.95)
  pred_left_3[[1]][count] <- list(pred_val_left$Hi.95)
  pred_right_3[[1]][count] <- list(pred_val_right$Hi.95)

  count <- count+1
}
pred_left_point = subset(pred_left, T, c('Point.Forecast'))/51
pred_left_low = subset(pred_left, T, c('Lo.95'))/51
pred_left_high = subset(pred_left, T, c('Hi.95'))/51
```

```
pred_right_point = subset(pred_right, T, c('Point.Forecast'))/51
pred_right_low = subset(pred_right, T, c('Lo.95'))/51
pred_right_high = subset(pred_right, T, c('Hi.95'))/51

dir.create(paste("./Results/",name,sep=""))

vals_left_point = data.frame(pred_left_1[[1]])
vals_left_point <- matrix(unlist(vals_left_point), ncol = 31, byrow = TRUE)
vals_left_point <- t(vals_left_point)
vals_left_lo = data.frame(pred_left_2)
vals_left_lo <- matrix(unlist(vals_left_lo), ncol = 31, byrow = TRUE)
vals_left_lo <- t(vals_left_lo)
vals_left_hi = data.frame(pred_left_3[[1]])
vals_left_hi <- matrix(unlist(vals_left_hi), ncol = 31, byrow = TRUE)
vals_left_hi <- t(vals_left_hi)
vals_right_point = data.frame(pred_right_1[[1]])
vals_right_point <- matrix(unlist(vals_right_point), ncol = 31, byrow = TRUE)
vals_right_point <- t(vals_right_point)
vals_right_lo = data.frame(pred_right_2)
vals_right_lo <- matrix(unlist(vals_right_lo), ncol = 31, byrow = TRUE)
vals_right_lo <- t(vals_right_lo)
vals_right_hi = data.frame(pred_right_3[[1]])
vals_right_hi <- matrix(unlist(vals_right_hi), ncol = 31, byrow = TRUE)
vals_right_hi <- t(vals_right_hi)
count <- 1
num_same <- 0
num_diff <- 0
for(i in seq(length(vals_left_point[,1]))){
```

```
mu_left <- sum(vals_left_point[count,])/length(vals_left_point[count,])
mu_right <- sum(vals_right_point[count,])/length(vals_right_point[count,])

sd_d = sqrt((sd(vals_left_point[count,])^2)/length(vals_left_point[count,])+
            (sd(vals_right_point[count,])^2)/length(vals_right_point[count,]))

lower_bound <- (mu_left-mu_right)-1.96*sd_d
upper_bound <- (mu_left-mu_right)+1.96*sd_d

if( lower_bound < upper_bound){
  if( lower_bound < 0 & upper_bound >0){
    print(paste('Day ',count, ' '))
    num_same <- num_same+1
  } else {
    print(paste("Day ",count," Significantly different                    ***"))
    num_diff <- num_diff+1
    png(file=paste("./Results/",name,"/",name,"_day_",count,".png",sep=""),
        width=600,height=350)
    hist(vals_left_point[count,],main=paste("Vals on ",count," of January"))
    dev.off()
  }
} else if (upper_bound<lower_bound){
  if( upper_bound < 0 & lower_bound > 0){
    print(paste("Day ",count," "))
    num_same <- num_same+1
  } else{
    print(paste("Day ",count," Significantly different                    ***"))
    num_diff <- num_diff+1
```

```
        png(file=paste("./Results/",name,"/",name,"_day_",count,".png",sep=""),
          width=600,height=350)
        hist(vals_left_point[count,],main=paste("Vals on ",count," of January"))
        dev.off()
      }
    } else{
      print(paste("Day ",count," "))
      num_same <- num_same+1
    }
    print(paste("Lower: ", lower_bound, " Upper: ", upper_bound))
    print("--------------------------------------------------")
    count <- count+1
  }
  print(paste("Number of days same: ",num_same))
  print(paste("Number of days different: ", num_diff))
}
```

SQL

```
CREATE TABLE daily_44201 (
StateCode varchar(100),
CountyCode varchar(100),
SiteNum varchar(100),
ParameterCode varchar(100),
POC int,
Latitude numeric(10,6),
Longitude numeric(10,6),
Datum varchar(100),
ParameterName varchar(100),
SampleDuration varchar(100),

PollutantStandard varchar(100),

DateLocal date,

UnitsOfMeasure varchar(100),

EventType varchar(100),

ObservationCount int,

ObservationPercent numeric(10,5),

ArithmeticMean numeric(10,5),

FirstMaxVal numeric(10,5),

FirstMaxHour int,

AQI int,

MethodCode varchar(100),

MethodName varchar(100),

LocalSiteName varchar(100),

Address varchar(100),

StateName varchar(100),

CountyName varchar(100),

CityName varchar(100),

CBSAName varchar(100),

DateOfLastChange date

);
```

```
select *, date_part('year', r.DateLocal) as Year

into sec_daily_44201

from daily_44201 as r
```

```
select Year
from sec_daily_44201
group by Year

select  StateName, StateCode, CountyName, CONCAT(StateName, CountyName), CountyCode, 
DateLocal, Year, avg(ArithmeticMean)
into county_avg_44201_2
from sec_daily_44201
group by StateName, StateCode, CountyName, CountyCode, DateLocal, Year

select StateName, CountyName, concat, CountyCode, Year, count(*)
into county_avg_44201_counts
from county_avg_44201
group by StateName, CountyName, concat, CountyCode, Year
having count(*) >= 330 and count(*) <= 366
order by StateName, CountyName, Year


select StateName, StateCode, CountyName, concat, CountyCode, Year, count(*)
into county_avg_44201_counts_2
from county_avg_44201_2
group by StateName, StateCode, CountyName, concat, CountyCode, Year
having count(*) >= 330 and count(*) <= 366
order by StateName, CountyName, Year

select StateName, CountyName, concat, CountyCode, Year, count(*)
into county_avg_44201_counts_high
from county_avg_44201
group by StateName, CountyName, concat, CountyCode, Year
```

```
having count(*) >= 365 and count(*) <= 366
order by StateName, CountyName, Year

select StateName, StateCode, CountyName, concat, CountyCode, Year, count(*)
into county_avg_44201_counts_high_2
from county_avg_44201_2
group by StateName, StateCode, CountyName, concat, CountyCode, Year
having count(*) >= 365 and count(*) <= 366
order by StateName, CountyName, Year

select StateName, CountyName, CountyCode, concat, count(*)
into county_year_counts_44201
from county_avg_44201_counts
group by StateName, CountyName, CountyCode, concat
having count(*) >= 9
order by StateName, CountyName

select StateName, CountyName, CountyCode, concat, count(*)
into county_year_counts_44201_high
from county_avg_44201_counts_high
group by StateName, CountyName, CountyCode, concat
having count(*) >= 9
order by StateName, CountyName

select StateName, StateCode, CountyName, CountyCode, concat, count(*)
into county_year_counts_44201_high_2
from county_avg_44201_counts_high_2
group by StateName, StateCode, CountyName, CountyCode, concat
```

```
having count(*) >= 9
order by StateName, CountyName



select StateName, CountyName, concat, CountyCode, Year, count(*)
from county_avg_44201
where concat in (select concat
                                        from county_year_counts_44201)
group by StateName, CountyName, concat, CountyCode, Year
having count(*) >= 325 and count(*) <= 366
order by StateName, CountyName, Year

select StateName, CountyName, concat, CountyCode, datelocal, avg as arithmean, Year
into threshold_county_avg_44201
from county_avg_44201
where concat in (select concat
                                        from county_year_counts_44201)

select StateName, CountyName, concat, CountyCode, datelocal, avg as arithmean, Year
into threshold_county_avg_44201_high
from county_avg_44201
where concat in (select concat
                                        from county_year_counts_44201_high)
order by StateName, CountyName, datelocal

select StateName, StateCode, CountyName, concat, CountyCode, datelocal,
CONCAT(StateCode, CountyName) as statecodecountyname, avg as arithmean, Year
into threshold_county_avg_44201_high_3
from county_avg_44201_2
```

```
where concat in (select concat
                                        from county_year_counts_44201_high_2)
order by StateName, CountyName, datelocal

select concat, datelocal, arithmean
from threshold_county_avg_44201_high

select datelocal, avg(arithmean) as value
into january_96_avg
from threshold_county_avg_44201_high
where datelocal <= '1996-01-31 00:00:00'
group by datelocal
order by datelocal
```