An analysis of vehicular emissions at Kansas State University

by

Jessica Marie Struck

B.S., California State University Bakersfield, 2016

A REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Statistics
College of Arts and Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2019

Approved by:

Major Professor
Dr. Michael J. Higgins

Abstract

There has been a wave of recent interest in understanding the dynamics of vehicular emissions in university towns. Using data from a recent survey of Kansas State University students, faculty, and staff—which includes a detailed itinerary of a day's worth of travel—I assess spatial and temporal trends of emissions at Kansas State University. By combining the survey data with secondary sources of data, including vehicular emissions data from Fueleconomy.gov and trip distance information from Google, I obtain improved measures for the quantity of emissions produced during each trip. After this cleaning to obtain detailed emissions data by trip and time, I develop heat maps for emissions. There are composed of shape files identifying zip-code boundaries and a raster layer. I find that most vehicular emissions are concentrated around campus, with the highest level of emissions occurring during rush hour. Furthermore, faculty and staff appear to, on average, produce more emissions than students. I also investigate how proposed methods for reducing emissions will affect these spatio-temporal trends. Specifically, I show that walking or biking short distances instead of driving may lead to a small overall reduction in vehicular emissions.

Table of Contents

List of Figures

Acknowledgments

I would like to thank Dr. Michael J Higgins for all the work he put into helping me write this paper, and for trusting me on this research. Thank you for meeting with me countless days and hours with little to no notice. Thank to Dr. Gregory Newmark for developing the survey that made this research come to life. Thank you to Dr. Trevor Hefley for guiding this research through the spatial component of the analysis. Without any of your contributions this work could not be done.

Dedication

I dedicate this to my husband, Bun. You make anything and everything I chose to do possible.

Chapter 1

Introduction

The purpose of this report is to examine the amount of CO_2 emissions emitted by students, faculty, and staff who travel to and from, as well as near and around, Kansas State University in Manhattan, Kansas. Understanding how emissions resulting from traveling to and from higher education institutions affect one community's environment can bring awareness at not only a local level, but globally as well. Using the information provided by survey participants, CO_2 emissions are calculated based off their distance(s) traveled. Assessing these calculations allowed for distinguishing between excess emissions related to convenient transportation, versus somewhat reasonable emissions for relevant transportation. The goal for severing emissions by different methods of transportation is to propose ways to reduce overall emissions.

The contribution of CO_2 emissions to global warming is at an all time high, with many environmentalist suggesting urgent changes needing to take affect immediately in order to sustain suitable living conditions in 2050 (Allen et al., 2019). An interest in emissions from higher education institutions has been seen from studies conducted in the United Kingdom, New Jersey, and Chile. Vásquez et al. (2015) examined the greenhouse gas emissions for the Cuircoc campus of Universidad de Talca to evaluate improvement scenarios associated with areas making the most CO_2 contribution in order to reduce emissions. Although this paper expanded to all emissions, such as those emitted from the campus for fuel consumption

and electricity, it did note that their campus had the lowest normalized per student GHG emission totals compared to 5 other institutions, since approximately 76 percent of the campus population uses public transportation. Roy et al. (2008) summarizes a study which examined CO_2 emissions per student per 100 hours of degree study based on their method of study. UK Open University took survey data from 20 courses, 13 campus-bases, 7 print based and online distance learning, and found that part-time students studying on campus reduced CO_2 emissions by 61 percent per student per credit point. One focus of this paper was to present the concept of distance learning as a means of reduction in CO_2 emissions. Campus emissions in the form of electric, heating, ventilation, air conditioning and hot water use was the topic of interest in the paper (Riddell et al., 2009). The article found that 40 percent of the nearly 38,000 tons of CO_2 emitted during the 2007 fiscal year came from purchased grid electricity. The other 60 percent of emissions came from generating heat and hot water and meeting the remaining needs of the electricity demands. Therefore, campus emissions is a subject of significant interest.

1.1 Survey

Dr. Greg Newmark is an Assistant Professor of Landscape Architecture and Regional & Community Planning at Kansas State University (KSU). In 2017, he conducted a survey among students, faculty, and staff at KSU in relation to their daily transportation to and from campus, and around Manhattan. The survey was developed in Qualtrics and included 2892 participants. Participants were asked to provide several descriptive statistics pertaining to their daily commute along with their demographics. The survey was taken over the week of March 13th-17th.

Although the survey is broad and includes a wide range of information regarding transportation, the scope of this research focuses on the EPA emissions from participants that drove or were driven to campus, along with other daily trips around Manhattan. During the week of the survey, participants were asked to track all locations they traveled to on a day that included traveling to campus. They were able to drag the cursor on the screen to their

locations on a map that stored the corresponding coordinates. They input the times for each location along with how they got to and from each location. This information includes if they were a driver or passenger, if they walked, biked, skated, or took the bus. The goal of the research is to understand how people are traveling to campus and around Manhattan in order to provide changes that can help eliminate unnecessary trips for the sake of pollution.

1.2 Design/methodology/approach

This report begins with an extensive cleaning of the data in several components. A large part of the work on this paper pertains to transferring, manipulating, and organizing the data. The cleaning began with setting up locations in R programming followed by arranging data by trip number. Next, organizing time and changing the format so that time could be added or subtracted. After this is the calculation of emissions which were obtained outside the scope of the survey data. This follows with obtaining trip distances through the use of Google Maps within R. Finally, the final working data set is obtained.

An assessment of where, and how much, emissions occur in Manhattan is the focal study of this report. A summary of campus emissions began with an analysis of descriptive statistics, such as who is emitting the most out of surveyed participants. Next, the heat maps which are developed through the use of shapefiles, a raster layer and a generalized linear model. The heat maps assess overall emissions as well as emissions over different hourly time periods. Finally, hypothetical emissions create scenarios which provide insight on the need to make changes towards CO_2 emission reductions.

The report is organized as follows. Chapter 2 discusses the cleaning of the data. The analysis of the data, including investigating statistical summaries of the data and the construction of heat maps, is contained in Chapter 3. Chapter 4 concludes with inferences and proposed future work.

Chapter 2

Cleaning the Data

My first job was to read the data into R from Qualtrics and begin the lengthy process of cleaning and formatting the data for analysis. My first variable of interest was the location, which included latitude, longitude, and several punctuation marks for each entry per cell. For instance, the entry for one individual's location read in a single cell as: "lat": "39.188831012619715", "long": "-96.57811485977174". This was an issue because I needed to observe latitude in one cell and longitude in another cell to include in my developing data frame. Also, there was no need for any punctuation.

I extracted the column that asked for the starting locations of participants who traveled to campus on any one day during this week. These starting locations indicated where each participant started his or her day. First, I used the `gsub()` function in R to remove all punctuation and all uppercase and lowercase letters from the cells of this columns. The `gsub()` function allows one to make formatting adjustments to character strings in R. Once all punctuation and letters were removed, I was left with just latitude and longitude coordinates in one cell. I needed to separate these to their own columns, so I generated a for-loop and used the `strsplit()` function to divide latitude and longitude into two separate columns, and stored them to a 2 by 2892 empty matrix.

The `strsplit()` function allows one to split a character string or a vector of character strings at a specified character in the string, such as splitting at commas, periods, semicolons,

brackets and so on. I set the split to take place at the first space mark, which occurred between the coordinates for latitude and the coordinates for longitude. Upon each split, the new separation appeared as four parts or columns: a space, latitude coordinates, a space, and longitude coordinates. Within the for-loop and following each split, I stored columns 2 and 4 (the latitude and longitude coordinates) of the new split to the empty i^{th} row of the 2 by 2892 matrix.

The data had similar columns for all other locations that participants traveled to on their travel day, so I repeated this process for all other locations traveled to by each participant. Thus, I generated a total of 16 for-loops to iterate the same process since 16 destinations was the maximum amount of travels logged by an individual. This means I had 16 columns representing latitude coordinates, and 16 columns representing longitude coordinates. However, not every participant traveled to 16 destinations, therefore, they would have a blank cell under columns they did not make trips for. I used the `cbind()` function in R which allows for columns of a matrix to merge in any order that is specified. Thus, I merged all latitude columns in order from starting location to destination 15 and named the columns Stating Location, Destination 1,...,Destination 15. I did the same for all longitude columns, so I had two 16 by 2892 matrices. I also created a column vector, ID, containing numbers 1 through 2892 to identify each individual in the dataset. I used `cbind()` to join ID in column one to each of the two matrices, making each one with dimensions 2892 by 17.

Before adding any new variables to the developing data frame, I wanted to observe the data matrix by a new row per trip, per ID. Therefore, I designed a for-loop within a for-loop to check for empty cells in all columns of the latitude and longitude matrices for each participant, meaning, if a cell was empty for any i^{th} column then the corresponding ID did not have a trip for the i^{th} trip number. The outside loop iterated for the number of rows in the matrices (thus, for each ID), and the inside loop iterated for the number of trips.

This organized the data so that each trip number is its own row, for all participants. I also used the for-loop to `cbind()` new columns for trip method, and trip reason. Thus, for each trip number, I include how the trip was made (car, walk, bike and so forth), and the reason the trip was made (school, work, recreation, and so forth). At this point, the

data frame included columns for ID number, trip number, starting latitude location, ending latitude location, starting longitude location, ending longitude location, trip method and reason for trip.

The next thing to include would be the times for each trip. Trip times were divided throughout three columns as hour, minute, and time of day (a.m. or p.m.), respectively. I began by combining all hour columns to one matrix, all minute columns to one matrix, and all time of day columns to one matrix. Then, I generated a for-loop to do several things:

1. I used `cbind()`for the the i^{th} column of the hours matrix with the i^{th} column of the minutes matrix, and I named this matrix "time."

2. I collapsed "time" to one column for hour and minute, separated by a semicolon.

3. I merged the new "time" matrix with the i^{th} daytime matrix using `cbind()`.

4. I changed all hour 12 values to 0.

5. I changed the time format to `POSIX`, meaning time would now read as "year-month-day-hour-minute-second" (where hours read 1–24). The `POSIX` format allows for mathematical operations for time to be computed easily.

This created a 2224 by 31 dimension matrix which included the leaving and arriving times for each trip.

Finally, the same procedure which was used to organize the data by trip number for the existing data frame was used to organize the time matrix by trip number. This created a 5452 by 4 dimension matrix, which included columns for ID number, trip number, starting time, and ending time. However, after obtaining the time matrix, I noticed there were some problematic entries. It became apparent that there was an issue when the data matrix was not the same dimension as the time matrix. Going back to the original data, I noticed there were cases where people recorded p.m. when they meant a.m., and a.m. where they meant p.m. Another common mistake was simply forgetting to log the time for a trip.

To begin, I made corrections to the latter, those who forgot to log trip times. I used the `which()` function in R to see which ID numbers from the data matrix had trip numbers

that not match up with a time for trip numbers in the time matrix. There were 12 IDs that were causing the matrices to be imbalanced. After I found which trip numbers were causing the imbalance, I removed that trip and all following trips for each corresponding ID due to invalid entry. Once the matrices were balanced, I merged the data frame with the time matrix. Next, I tackled the trips that mistakenly logged a.m. for p.m. and vice versa using `which()` commands and checking for trips that took, for example, -11 hours to complete. Then, I removed any trips longer than four hours. Four hours was selected because it included those who commute to and from Topeka, Kansas City, or other locations, although, anything over four hours seemed to be a miscue. Finally, I created a new data frame, which included the adjusted times with the original data frame.

Once the foundation of the data frame was established, I began adding categorical variables related to transportation, as well as some demographics. I started by adding the following: 1) participants affiliation (undergraduate student, graduate student, faculty, or staff); 2) home street name 3) nearest cross street; 4) driver status (driver, passenger that gets dropped off before the driver parks, or passenger that gets out where the driver parks, 5) vehicle type (midsize, large car, minivan, SUV, truck, and so on); 6) vehicle make; 7) vehicle model; 8) vehicle year; 9) fuel type; 10) year of college (if a student); 11) associated department; 12) if respondent lives on campus; 13) year born; 14) sex; 15) income. Each of these variables was added to the data frame by matching on ID number using the `match()` function. I would obtain each variable from the original data, "df" and `cbind()`it with ID, calling it "idcovs." Then I would match on the data frame via "idcovs." I also included the parking location or location to most likely get dropped off. This was done using the same techniques as cleaning the latitude and longitude coordinates. The new data frame now included these 16 additional components.

I then added CO_2 emissions data and the MPG data. I attempeted to obtain the MPG for each car based off its make, model, and year. Then I could need to obtain the CO_2 emissions for each of these cars based off their MPG. I was able to find what I needed from `Fueleconomy.gov`. I downloaded the .csv file and imported the data into R.

However, certain vehicle descriptions from the `Fueleconomy.gov` dataset did not match

up with vehicle descriptions from the surveyed data. For example, the survey data uses the description "Small SUV," whereas the fuel economy data uses the descriptions "Small Sports Utility Vehicle." Another issue was that not all entries of the survey data included all variables to describe the vehicle. For instance, some entries only provided the make and model, others only the year and model, and so forth. Therefore, I needed to establish the categories that would be used for vehicle descriptions, and I identified them as the following, ranging from least amount of information provided to most: 1) only vehicle class, 2) just class and year, 3) class, make and year, or 4) make, model and year. Since there was not an exact one-to-one correspondence between the two data sets for vehicle descriptions, I would need to match the descriptions across data sets and provide labels.

I stared by using the `aggregate()` function in R to apply average combined miles per gallon across the four categories. The `aggregate()` function splits the data into subsets and computes summary statistics for each. I merged this data with the current data frame by matching on ID. However, not all entries of the data involved emissions since respondents included those who may have walked, biked, skateboarded and so forth. I multiplied the emissions variable by a vector of zeros and ones, zero if emissions did not pertain to a trips mode of transportation and one otherwise, and called this column "actual emissions," which was now included in the data frame.

The next thing I needed to include would be trip distances. The R package `gmapsdistance` uses Google Maps to calculate the distance between two locations based off the mode of transportation (Melo et al., 2018). Thus, I easily found all distances for each trip and merged this data with the existing data frame.

At this point, I started to ponder the best way to represent trip times in a plot. If I were to plot trip locations over time, it would be hard to interpret the plot because I wanted to break up each trip by the minute so that I could interpret precise locations in a plot. Therefore, I added a "trip minute" column to the data frame that would match on ID, such that there would be a new row for each trip minute. For example, a five-minute trip would be represented by 5 rows of the data frame, a new row for each trip minute, however, the same ID number and trip number for all five rows. Next, I did the same thing for the locations and

emissions. I broke up emissions to per minute, and the latitude and longitude coordinates to per minute. This helps with obtaining a more accurate display of emissions at locations over time.

Chapter 3

Results

3.1 Descriptive Statistics

The descriptive statistics examined emissions from different groups of survey participants. Figure 3.1 shows emissions based off KSU affiliate with faculty and staff showing the highest emissions. Figure 3.2 shows emissions from those who live on campus and those who live off campus. It is evident that emissions are much higher for those who live off campus. Figure 3.3 compares emissions between different colleges. Emissions were highest for the college of agriculture and the college of business. Figure 3.4 plots emissions versus the year of birth for survey participants. It appears that those born prior to 1980 emitted higher amounts of CO_2 than those born in the years after.

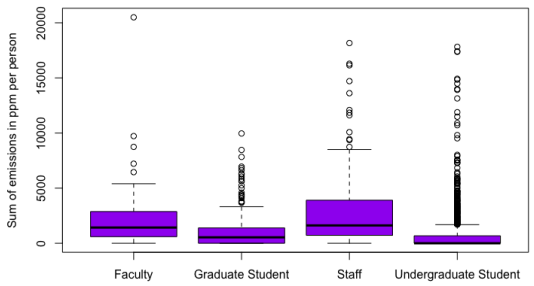

Figure 3.1: *Plot of emissions by KSU affiliate*

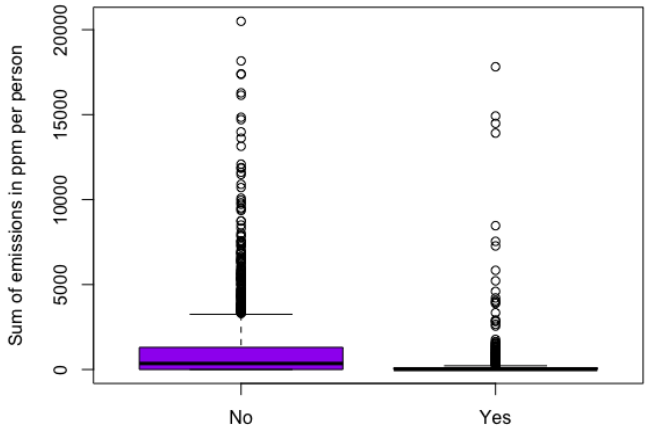

Figure 3.2: *Plot of emissions by indication of on-campus living versus off-campus living.*

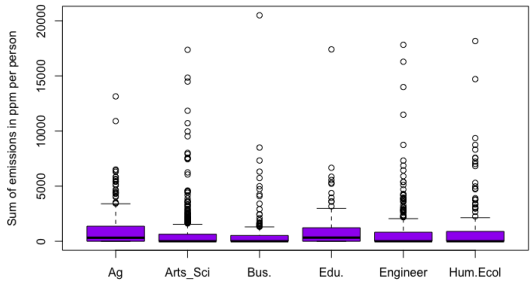

Figure 3.3: *Plot of emissions by college/department*

3.2 Predicting emissions over time

At this stage of my research, it was time to prepare for graphical displays and interpretations. Figure 3.5 plots all emissions over a 24 hour time period. The most significant peak time for emissions appears around 7:00 am, with smaller peaks around 9:00 am, 12:00 pm, and between 5:30 and 6:00 pm.

Next, I needed a map corresponding to the region of the data to plot points and/or overlay heat maps. I was able to retrieve shape files for zip codes in the state of Kansas from the United States Census Bureau government website (Bureau, 2019). A shapefile is type of

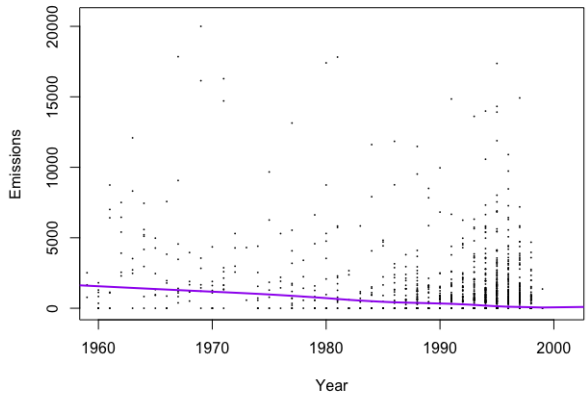

Figure 3.4: *Plot of emissions by year of birth*

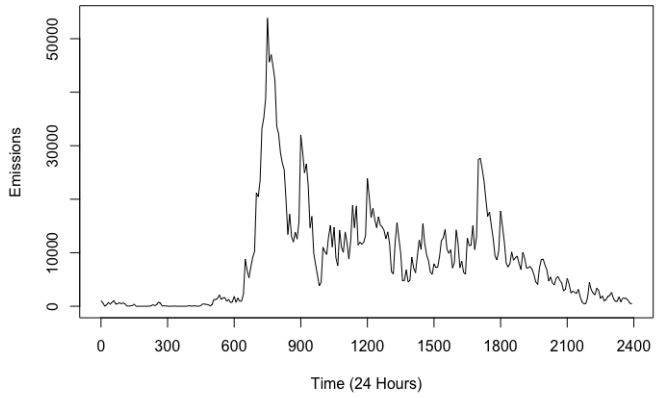

Figure 3.5: *Plot of all emissions over 24 hours*

format used for storing geometric locations. We can plot geographic features in a shapefile and represent them by points, lines, or polygons. Shapefiles are generally used to represent continuous spatial object and boundaries. To open the shapefile in R, I begin by reading the URL into R. Then I used the `tmpdir()` function to create a temporary file on my device to store the zip file. Next, I used the `baseline()` function to take the URL file name and remove all of the path, followed by the `download.file()` function to download the file into that temporary file. I used the `unzip()` function to unzip the zip file to the current working directory. Finally, I used the `readOGR()` function from the `rgdal` package (Bivand et al., 2015) to read an OGR (OpenGIS Simple Features Reference Implementation) data source

and layer into a suitable spatial vector object. Manhattan is made up of three zip codes: 66502, 66503, and 66506. I extracted only these three zip codes from the shapefile, then I used the `rbind()` function to merge all three zip codes into one shapefile seen in Figure 3.6.

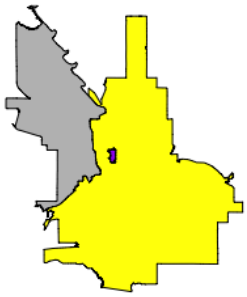

Figure 3.6: *Shapefile of Manhattan, Kansas*

The grey region represents 66503, the yellow region represents 66502, and the purple region contained inside the yellow region is 66506, Kansas State University's Campus. I started by plotting all IDs first starting location (home location or initial starting point) onto the shapefile so I could see where the majority of trips to campus originate. I needed to transform the latitude and longitude coordinates from the data to spatial points so that I could plot locations. I used the `SpatialPointsDataFrame()` function in R to covert the latitude and longitude coordinates to spatial points. Figure 3.7 shows all starting locations on the shapefile of Manhattan. We can see that the starting locations primarily occur near or on campus.

I extended the use of this shapefile into developing a heat map to display the emissions in Manhattan. Recall that the emissions were calculated based off vehicle descriptions provide by survey respondents through the correspondence of data available from `Fueleconomy.gov`. The shapefile is not sufficient to produce a heat map; I needed to include a raster layer within the shapefile. *Rasters* are geographically referenced images which consists of pixels organized into a grid where each pixel contains a value that represents information. In reference to

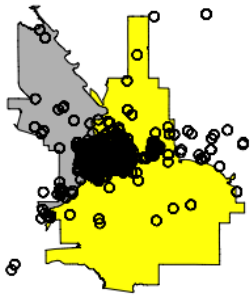

Figure 3.7: *Shapefile of Manhattan, Kansas with all staring locations*

the transportation data, each pixel may contain a value which represents something such as total emissions. The plot seen in Figure 3.8 sums emissions over a cell or pixel of the raster layer that fits the border designed from the Manhattan shapefile.

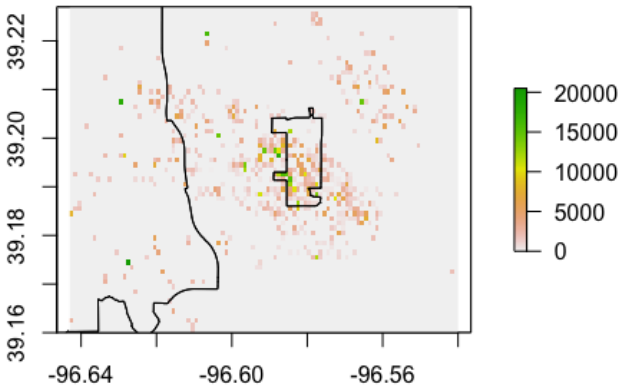

Figure 3.8: *Raster layer summing emissions over pixels*

To produce the raster layer in R, I defined a border which ranged around the shapefile so that I could examine only trips whose initial trip location originated within the limits of the shapefile. I obtained the border by using the `bbox()` function inside the `min()` and`max()`

function in R. `bbox()` function provides the minimum and maximum values of a spatial object. I obtained the minimum and maximum latitude values between the shape files for 66502 and 66503, and then same for the minimum and maximum longitude values. I used the `raster()` function to read in the maximum and minimum values of latitude and longitude so that the border dimensions of the raster would correspond with the border obtained from the shapefile of Manhattan. The `raster()` function can create raster layers from different objects, such as files, matrices, images, or even from scratch. The `rasterize()` function is then used to transfer all values associated with the pre-specified object spatial data to raster cells. Hence, I used the `rasterize()` function to transfer the spatial points for locations to appropriate raster cells. The field specification inside the `rasterize()` function takes the values to be transferred, in this case it would take emissions. The `fun` specification determines how to assign values to cells, such as sum, mean, min or max.

While the raster layer poses as a great visualization of the data for displaying emissions over Manhattan, I wanted to utilize a model that would predict emissions over locations, so I could advance the scope of my analysis with a heat map. To determine which model would be best fit the data, I began by looking at a histogram of the actual emissions data seen in Figure 3.9.

The plot of emissions shows a heavy right tail distribution which originally led me to think a gamma distribution would be a nice fit. However, the support of the gamma distribution does not include zero, while approximately 58 percent of the trips produced no emissions. Hence, a distribution which accounts for an excess amount of zero counts would be an appropriate choice. To account for these zero counts, an obvious choice may be a zero inflated Poisson distribution; however, this distribution assumes a discrete random variable, whereas the emissions data is continuous. Ultimately, we are able to incorporate both the zero-counts and the heavy right tail properties by modeling emissions using a Tweedie distribution, which includes a family of continuous normal and gamma distributions, Poisson, and the Poisson-gamma mixed compound distribution (Tweedie, 1984).

Recall that Generalized Linear Models (GLM) pertain to distributions where the response is other than normal with some degree of non-linearity in the model structure ((Christensen,

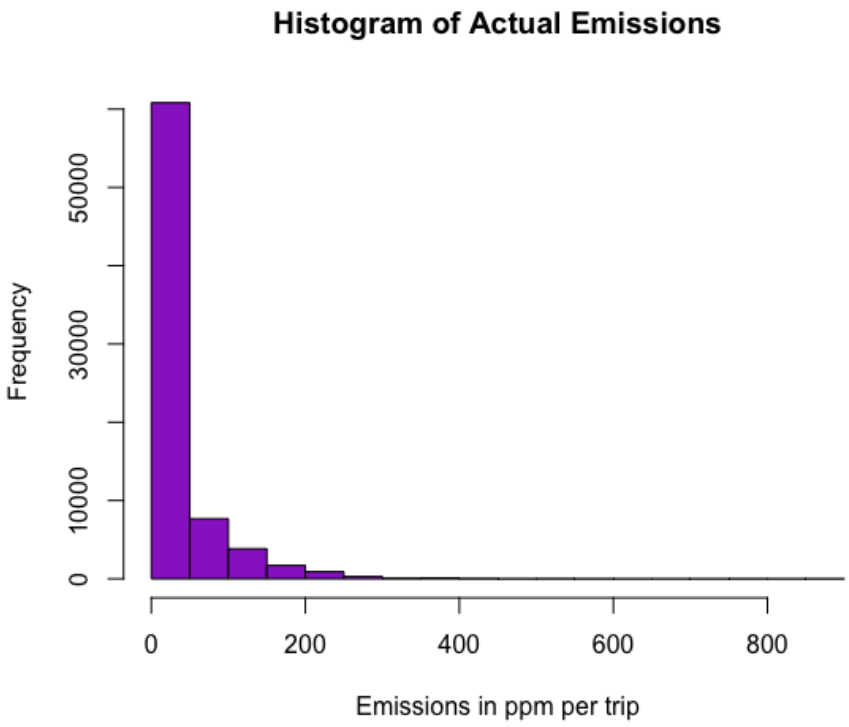

Figure 3.9: *Histogram of emissions*

2011, page. 12)). The assumptions of GLMs include (1) independence of the Y_i with their (2) distribution belonging to some exponential family. The structure of GLM is given by $g(\mu_i) = X_i\beta$, where g is a smooth monotonic link function. Generalized Additive Models (GAM) are the same as GLMs except they include "a linear predictor involving a sum of smooth functions of covariates (Wood, 2017, pg. 161)."

For my predictive model, I employed a GAM with a Tweedie distribution and a log link function, to predict counts of emissions in the raster layer regions. That is, I fit the model:

$$\log(E(Y_i)) = \alpha + f_1(lat_i) + f_2(long_i) \tag{3.1}$$

The response variable Y_i is the emissions for trip i, lat_i and $long_i$ are the latitude and longitude for trip i, and the f_1 and f_2 are smooth functions. To fit the model, I used the `gam` function from the `mgvc` package in R and specified `bs=''gp''` for Gaussian process models (Wood, 2015). Figure 3.10 shows a heat map of predicted emissions obtained from

the GAM with an overlay of data locations plotted onto the heat map.

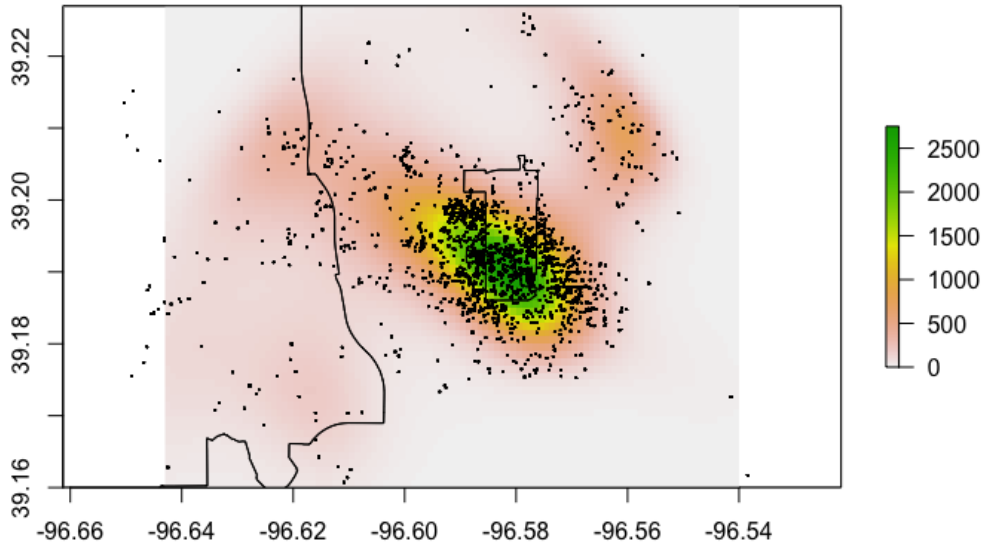

Figure 3.10: *Heat map of predicted emissions in ppm per day with overlay of locations*

We can see from Figure 3.10 that the largest amount of emissions take place near and around campus with maximum emissions exceeding 2500 parts per million (ppm). Instead of making an inference to the total emissions, I wanted to look at emissions over different time periods.

To obtain emissions over different time intervals I begin by changing the time data from hours to minutes. Then, I broke up the emissions into periods which seemed relevant to daily commute patterns. For instance, intervals such as 7:00 to 9:00 a.m. and 4:00 to 6:30 p.m. were designated as periods when we would expect to see higher emissions traveling to and from campus. Figure 3.11, Figure 3.12, Figure 3.13, Figure 3.14, and Figure 3.15 show emissions for 5 different time periods, 7:00-9:00 am, 9:00-11:00 am, 11:00 am-1:00 pm, 1:00-4:30 pm, and 4:00-6:30 pm, respectively. The five figures all use the same scale for measuring the emissions, 0-600 ppm.

I noticed emissions were most spread out during the morning hours of 7:00 to 9:00 a.m. This attributes to people traveling from their homes to work, schools, and to K State's campus. Emissions are higher around campus during lunch hours, 11:00 a.m. to 1:00 p.m., and highest around campus during 4:00 to 6:30 p.m.. This indicated a large number of people are leaving K-State around the same time in the evening contributing to higher emissions in the campus area.

I now investigate emissions if those who traveled a mile or less did not drive. Hence, I developed hypothetical data by changing emissions to zero for any trips whose distance was less than or equal to one mile. Figure 3.16 shows the differences in original emissions and hypothetical emissions from 11:00 a.m. to 1:00 p.m. I chose this period because it is lunch time, and although K-State is within a mile to several restaurants and coffee shops, many people still drive to lunch to save time or because they do not feel like walking.

The difference in emissions between the two images seen in Figure 3.16 was 7,442ppm. Similarly, Figure 3.17 compares a difference in original emissions and hypothetical emissions

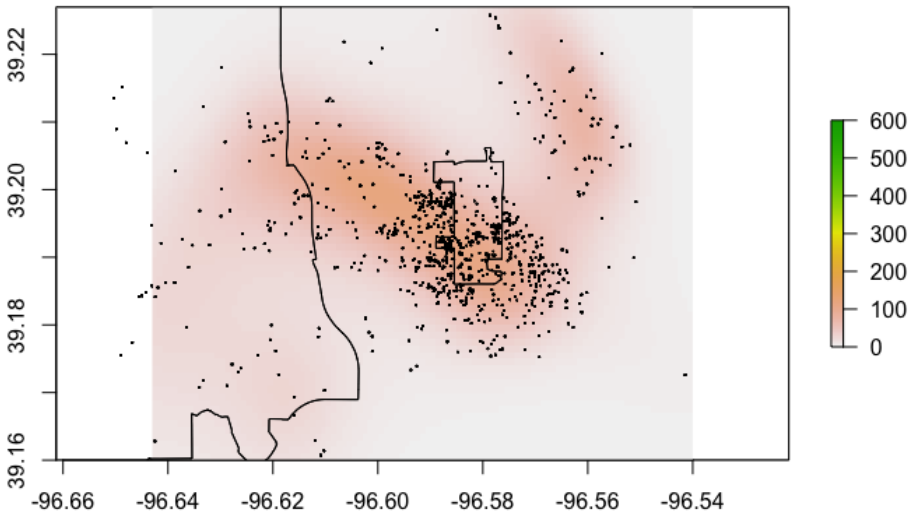

Figure 3.11: *Heat map of predicted emissions in ppm per hour with overlay of locations from 7:00-9:00 a.m.*

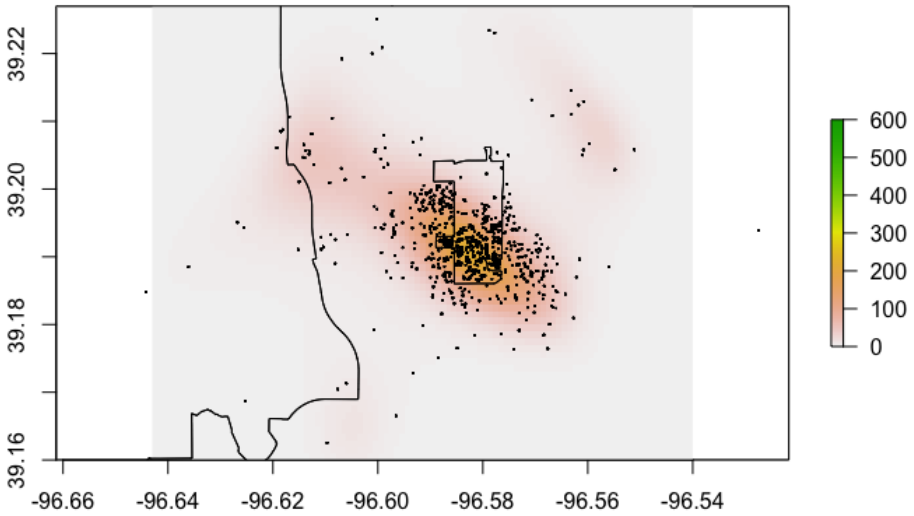

Figure 3.12: *Heat map of predicted emissions in ppm per hour with overlay of locations from 9:00-11:00 a.m.*

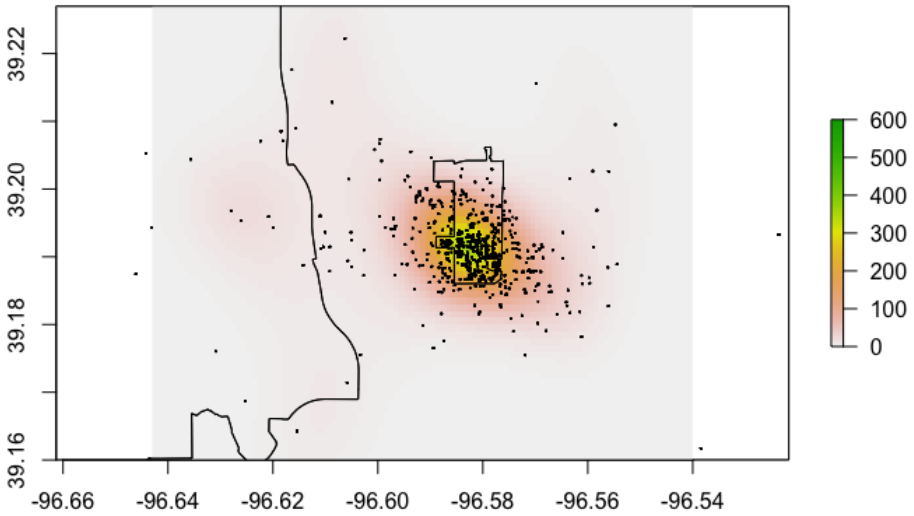

Figure 3.13: *Heat map of predicted emissions in ppm per hour with overlay of locations from 11:00 a.m.- 1:00 p.m.*

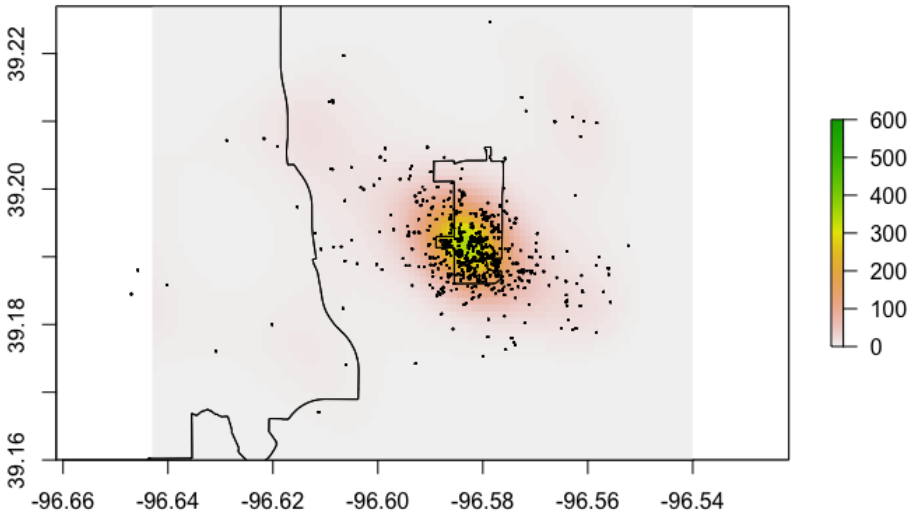

Figure 3.14: *Heat map of predicted emissions in ppm per hour with overlay of locations from 1:00-4:00 p.m*

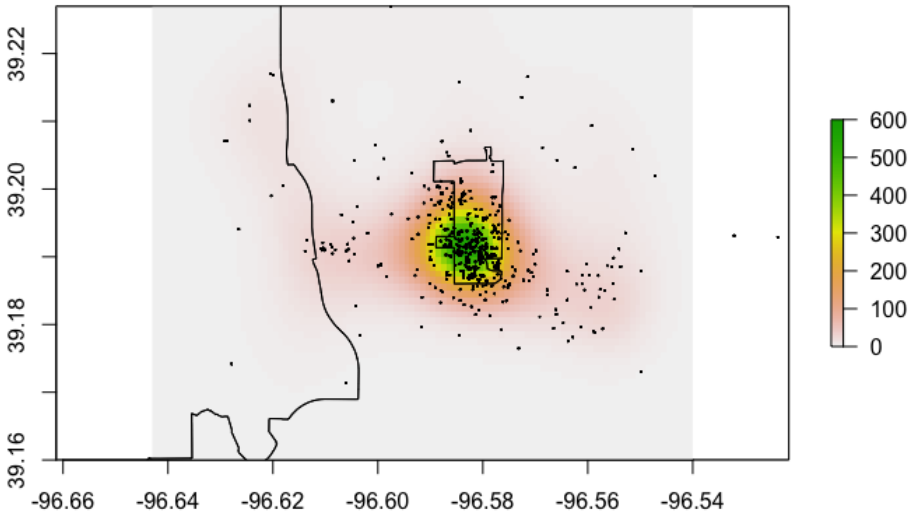

Figure 3.15: *Heat map of predicted emissions in ppm per hour with overlay of locations from 4:00-6:30 p.m*

from 7:00 to 9:00 a.m., and Figure 3.18 for 4:00 p.m. to 6:30, respectively. These figures examined the morning rush and the evening rush. Lastly, Figure 3.19 compares a difference in original emissions and hypothetical emissions from 9:00 p.m. to 2:00 a.m. to see what emissions are like around Aggieville's bar and hangout scene.

Figure 3.20 displays a comparison of all emissions over time versus hypothetical emissions over time. T here is an notable decrease in the hypothetical emissions around 7:00 am, as well as 9:00 am and 5:30 to 6:00 pm. In general, a reduction in overall emissions is a result of using an alternative means of transportation other than driving for trips within one mile.

The difference between original emissions and hypothetical emissions from 7:00 to 9:00 a.m. was 10,852 ppm, from 4:00 to 6:30 p.m. was 6,047 ppm, and from 9:00 p.m. to 2:00 a.m. was 5,550. The lagest difference in emissions took place during the morning hours of 7:00 to 9:00 a.m., when most people are headed to work or school. This time period also showed the most scattered pattern of emissions. It seemed reasonable to next consider who was doing most the driving in Manhattan.

We now examine hypothetical differences the descriptive statistics, such as the age, year in college, and K-State affiliation, of the drivers. Recall that the different affiliates of Kansas State for survey participants was faculty, staff, graduate students and undergraduate students. Figure 3.21 shows actual emissions and hypothetical emissions by affiliation to K State. Figure 3.22 shows the actual emissions and the hypothetical emissions for indication of living on-campus or off-status. Figure 3.23 shows the actual emissions and the hypothetical emissions between the different colleges/departments.

The hypothetical emissions for the descriptive statistics was primarily seen for the living on or off campus data, and for the college/department data. Both Figure 3.22 and Figure 3.23 indicated a reduction in CO_2 emissions in their hypothetical plots.

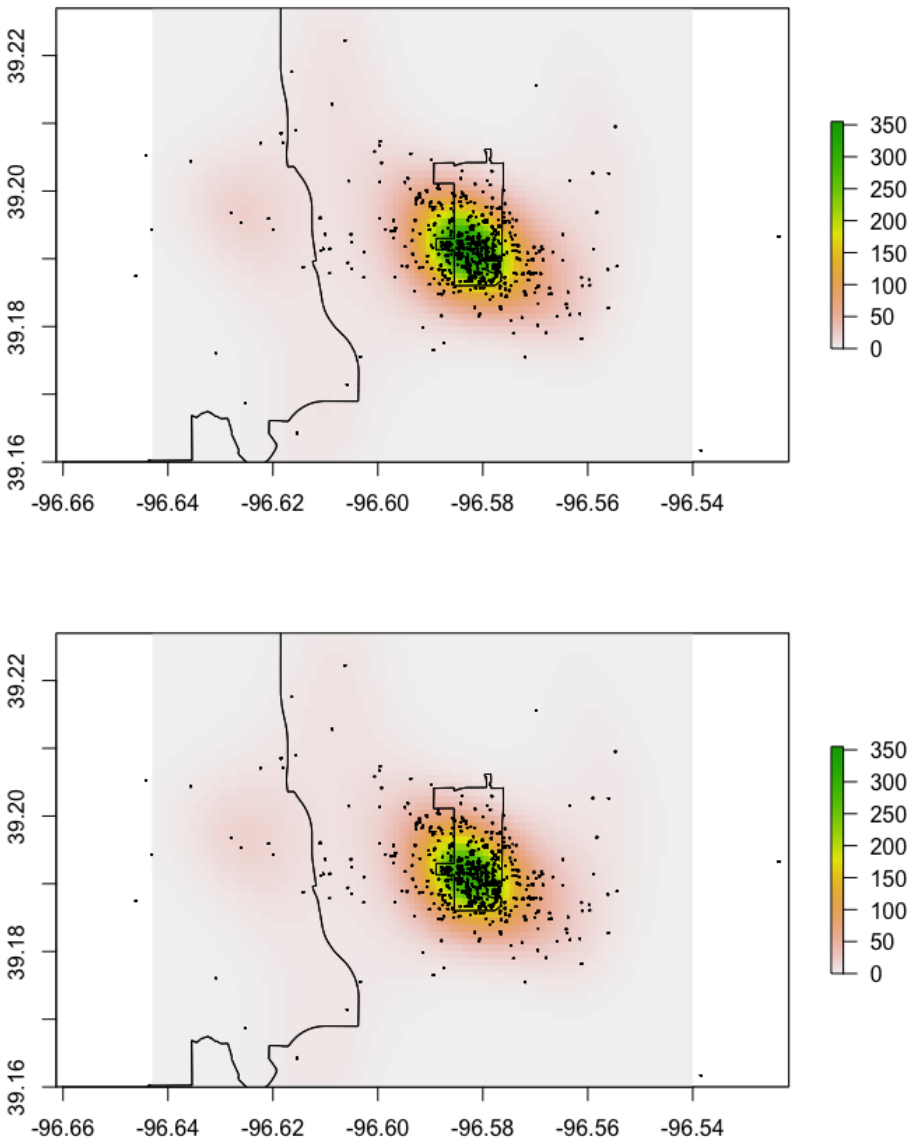

Figure 3.16: *Predicted emissions in ppm per hour with overlay of locations from 11:00 a.m. - 1:00 p.m. (Top) Original emissions; (bottom) hypothetical emissions for trips within one mile.*

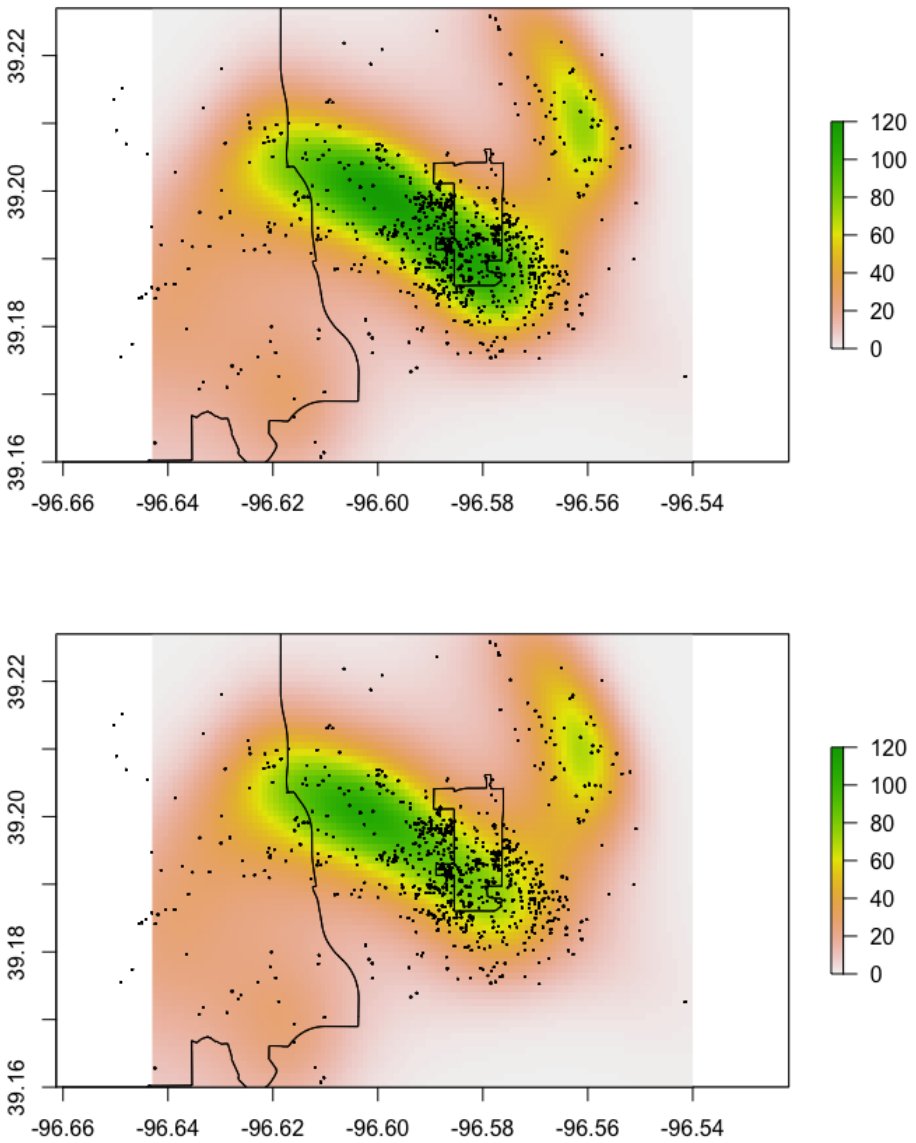

Figure 3.17: *Predicted emissions in ppm per hour with overlay of locations from 7:00 - 9:00 a.m. (Top) Original emissions; (bottom) hypothetical emissions for trips within one mile.*

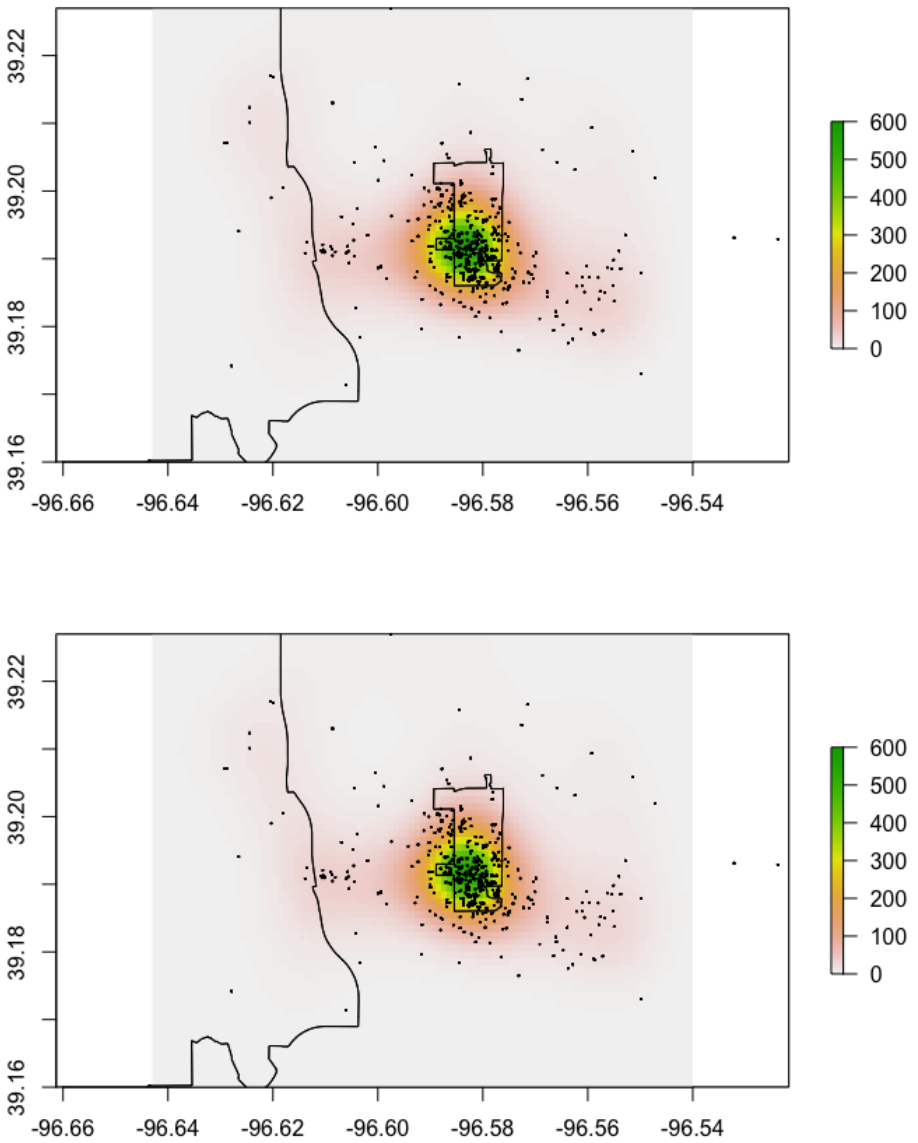

Figure 3.18: *Predicted emissions in ppm per hour with overlay of locations from 4:00 - 6:30 p.m. (Top) Original emissions; (bottom) hypothetical emissions for trips within one mile.*

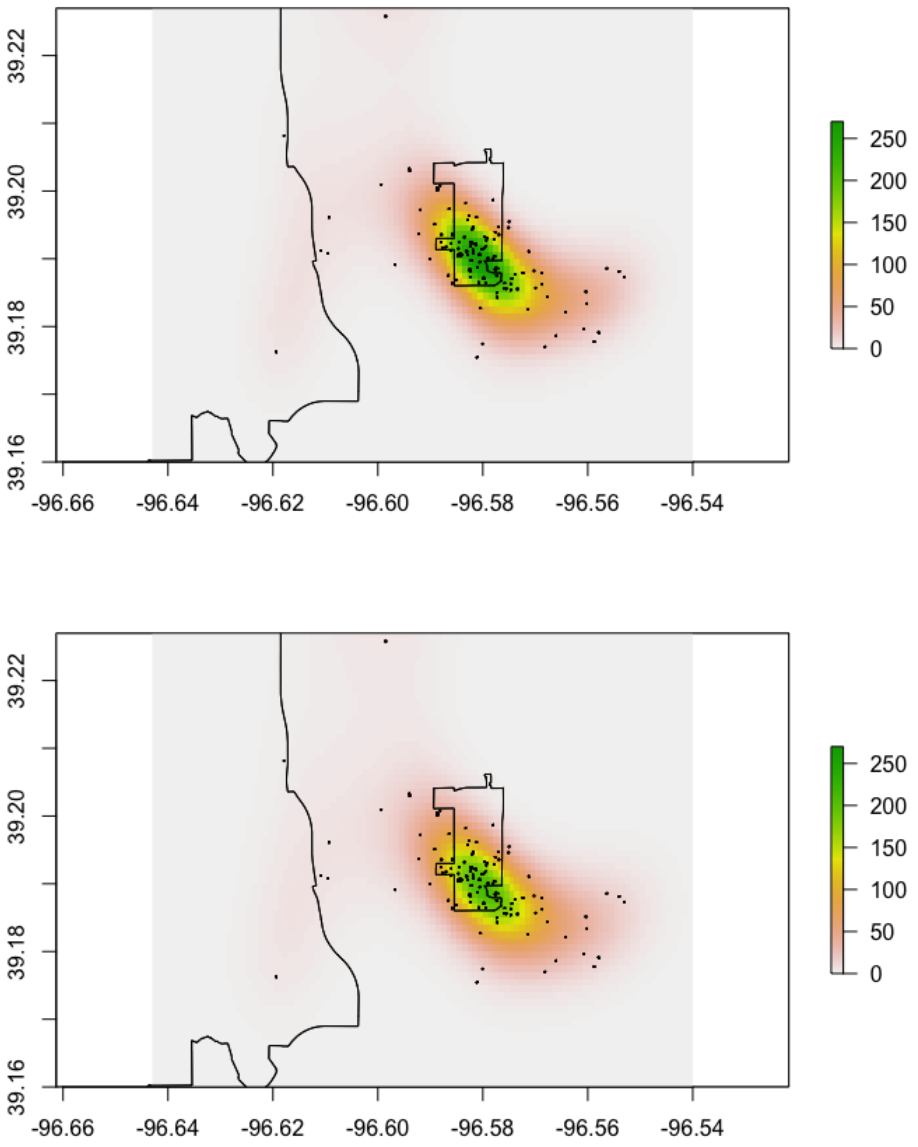

Figure 3.19: *Predicted emissions in ppm per hour with overlay of locations from 9:00 p.m. - 2:00 a.m. (Top) Original emissions; (bottom) hypothetical emissions for trips within one mile.*

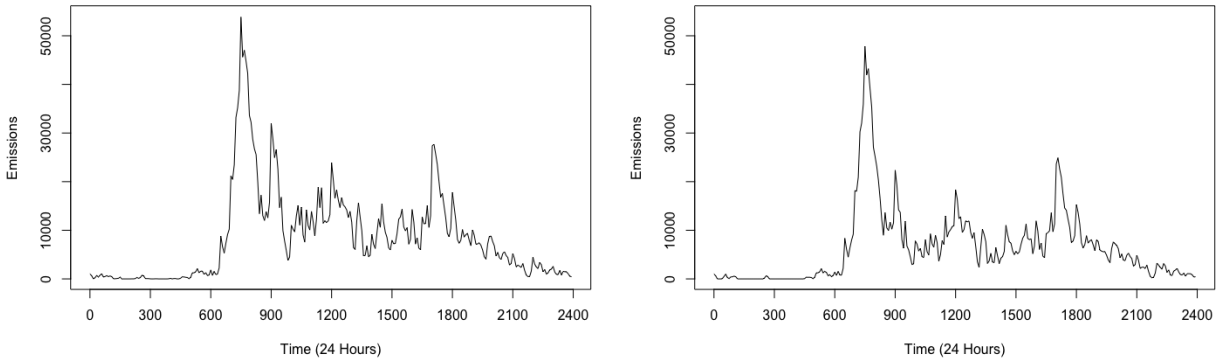

Figure 3.20: *All original emissions in ppm per hour (left) vs hypothetical emissions for trips within one mile (right).*

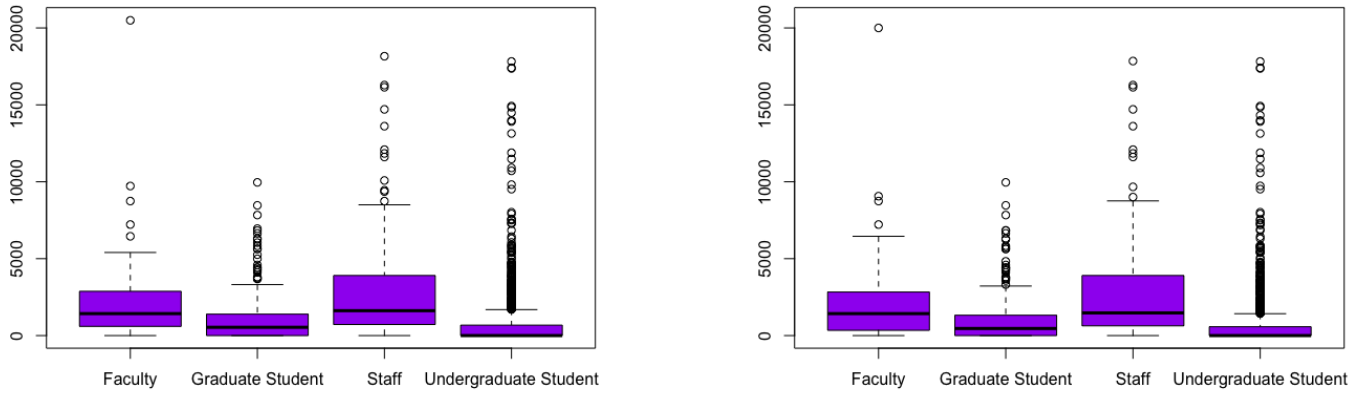

Figure 3.21: *Original emissions in ppm per hour (left) vs hypothetical emissions for trips within one mile (right) by K State affiliation.*

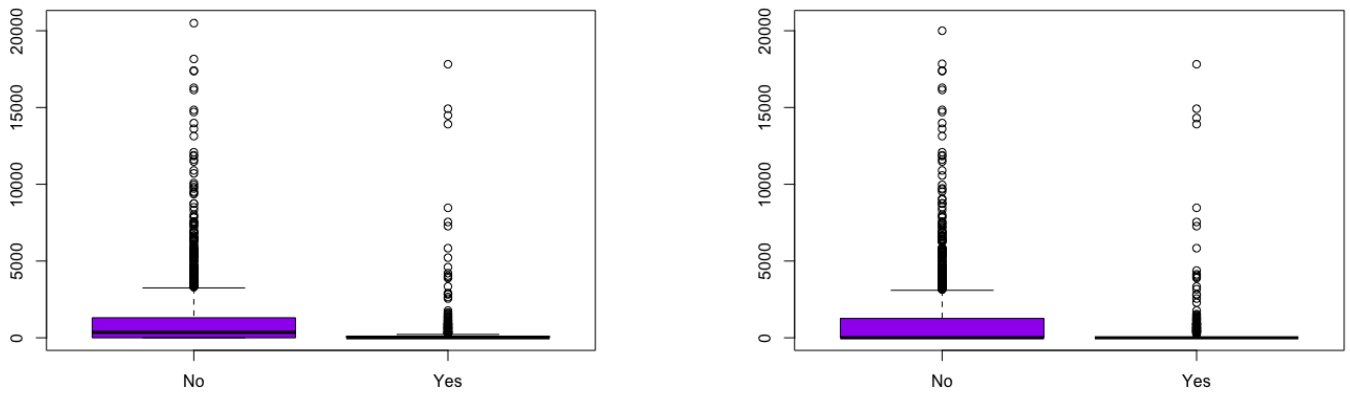

Figure 3.22: *Original emissions in ppm per hour (left) vs hypothetical emissions for trips within one mile (right) by living on campus status.*

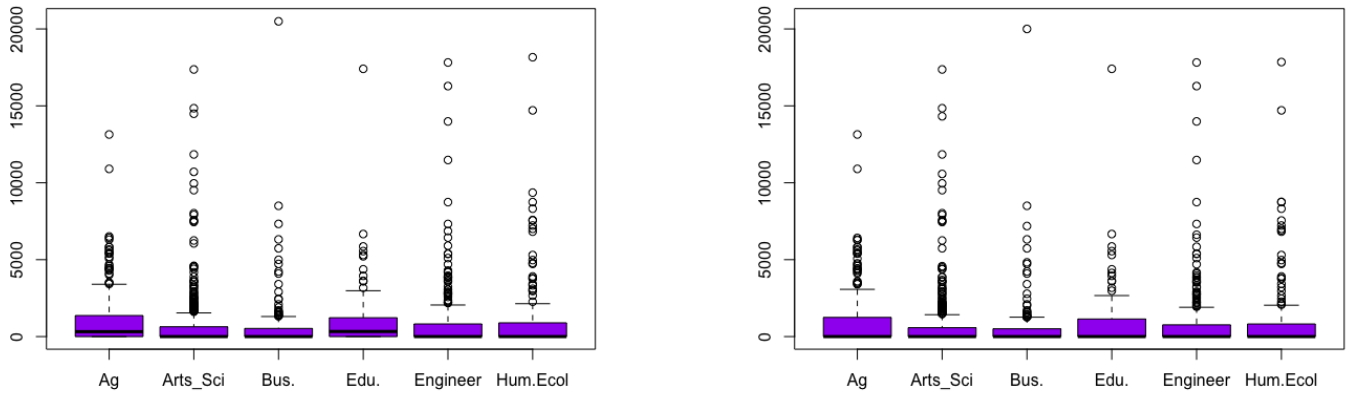

Figure 3.23: *Original emissions (left) vs hypothetical emissions for trips within one mile (right) by subject area.*

Chapter 4

Conclusion

4.1 Conclusion

The collected survey data of daily trips allowed for an assessment of emissions centered around Kansas State University in Manhattan, Kansas. A precise examination of locations became the framework of the analysis and initiated the border for the shape file. This developed into the addition of a raster layer which included the three zip codes in Manhattan, including campus. A histogram of emissions indicated a heavy right tail distribution with many zeros, and based off these finding, the Tweedie distribution was selected as the most appropriate fit to the data. The application of the Tweedie distribution was used to obtain a predictive model to estimate emissions over different time periods.

For original emissions predicted by the model, the time frame with the highest amount of emissions predicted was between 4:00 and 6:00 pm. Between the hours of 7:00 to 9:00 am is when the emissions were most scattered among the heat map. Higher emissions were also seen around lunch time, during 11:00 am to 1:00 pm.

The hypothetical emissions, which substituted zero emissions for all trips within one mile, indicated the greatest reduction during the morning hours of 7:00 to 9:00 am. This difference in emissions was 10,852 ppm compared to a 7,442 ppm reduction from 11:00 am to 1:00 pm, a 6,047 ppm reduction from 4:00 to 6:30 pm, and a 5,550 ppm reduction from 9:00 pm to

2:00 am. It is evident that using an alternative means of transportation which does not emit CO_2 for shorter trips makes a difference in the environment.

The descriptive statistics indicate that faculty and staff are the primary source of emissions. Less emissions were seen from students living on campus opposed to those who lived off campus. The college of agriculture and the college of education had the highest amount of emissions. It appeared that survey participants born prior to 1960 through 1980 had higher emissions than those born in years after.

4.2 Future Work

Future work would include adding post-stratification weights to the data to get an accurate summary of emissions over the entire campus body. Additionally, it would be constructive how emissions look for different strata. For example, it would be constructive to stratify the data by college, or class standing (freshman sophomore, etc.), or affiliation.

Analyzing different hypothetical outcomes may also be investigated in the future. For example, there is a strong interest in how public transit can make a difference in reducing emissions. More work on investigating whether, for instance, if a certain proportion of trips within proximity to a bus stop are traveled via bus opposed to driving, can be performed using this data. A more extensive analysis of this work is in progress.

Bibliography

M. Allen, P. Antwi-Agyei, F. Aragon-Durand, M. Babiker, P. Bertoldi, M. Bind, S. Brown, M. Buckeridge, I. Camilloni, A. Cartwright, W. Cramer, P. Dasgupta, A. Diedhiou, R. Djalante, W. Dong, K.L. Ebi, F. Engelbrecht, S. Fifita, J. Ford, S. Fuß, B. Hayward, J.-C. Hourcade, V. Ginzburg, J. Guiot, C. Handa, Y. Hijioka, S. Humphreys, M. Kainuma, J. Kala, M. Kanninen, H. Kheshgi, S. Kobayashi, E. Kriegler, D. Ley, D. Liverman, N. Mahowald, R. Mechler, S. Mehrotra, Y. Mulugetta, L. Mundaca, P. Newman, C. Okereke, A. Payne, R. Perez, P.F. Pinho, A. Revokatova, K. Riahi, S. Schultz, R. Seferian, S. Seneviratne, L. Steg, A.G. Rogriguez, T. Sugiyama, A. Thonas, M.V. Vilarino, M. Wairiu, R. Warren, G. Zhou, and K. Zickfeld. Technical summary: Global warming of 1.5°C. An IPCC special report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty, 2019. URL http://pure.iiasa.ac.at/id/eprint/15716/.

Roger Bivand, Tim Keitt, Barry Rowlingson, Edzer Pebesma, Michael Sumner, Robert Hijmans, and Even Rouault. Package rgdal. *Bindings for the Geospatial Data Abstraction Library. Available online: https://cran.r-project.org/web/packages/rgdal/index.html (accessed on March 30, 2019)*, 2015.

United States Census Bureau, 2019. URL https://www.census.gov/geo/maps-data/data/tiger-line.html.

Ronald Christensen. *Plane answers to complex questions: the theory of linear models*. Springer Science & Business Media, 2011.

Rodrigo Azuero Melo, Demetrio T Rodriguez, and David Zarruk. Package gmapsdistance:

Distance and travel time between two points from google maps. *R package version*, pages 1–7, 2018.

William Riddell, Krishan Kumar Bhatia, Matthew Parisi, Jessica Foote, and John Imperatore III. Assessing carbon dioxide emissions from energy use at a university. *International Journal of Sustainability in Higher Education*, 10(3):266–278, 2009.

Robin Roy, Stephen Potter, and Karen Yarrow. Designing low carbon higher education systems: Environmental impacts of campus and distance learning systems. *International Journal of Sustainability in Higher Education*, 9(2):116–130, 2008.

M.C.K. Tweedie. An index which distinguishes between some important exponential families. In *Statistics: Applications and new directions: Proc. Indian statistical institute golden Jubilee International conference*, volume 579, pages 579–604, 1984.

Leonardo Vásquez, Alfredo Iriarte, María Almeida, and Pablo Villalobos. Evaluation of greenhouse gas emissions and proposals for their reduction at a university campus in chile. *Journal of Cleaner Production*, 108:924–930, 2015.

Simon Wood. Package mgcv. *R package version*, pages 1–7, 2015.

Simon N Wood. *Generalized additive models: An introduction with R*. Chapman and Hall/CRC, 2017.

Appendix A

R Code

```
#Cleaning Lat and Long data for Starting Location
df <- read.csv("/Users/Jess/CampusSurvey.csv", header=TRUE, stringsAsFactors = FALSE)
all <- df$Q8.3[1:2892]
all <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","",all) #remove all punctuation
all <- gsub("[a-zA-Z]","", all) #get rid of all upper and lower case letters

store.data <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data) <- c("Latitude.Start", "Longitude.Start")

for (i in 1:2892){
  if (nchar(all[i]) > 0){
    split <-strsplit(all[i], "[[:space:]]")
    store.data[i,1] <- as.numeric(lapply(split, '[[', 2))
    store.data[i,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#Removing empty data from starting location
#index = which(df$Q8.2 == "" | df$Q8.2 == "No")
#new.df_1 <- store.data[-index, byrow=TRUE]
############################################################
#First Destination (to campus)
first <- df$Q8.5
first <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", first) #remove all punctuation
first <- gsub("[a-zA-Z]","", first) #get rid of all upper and lower case letters
```

```
store.data_2 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_2) <- c("Latitude.Campus", "Longitude.Campus")

for (j in 1:2892){
  if (nchar(first[j]) > 0){
    split <-strsplit(first[j], "[[:space:]]")
    store.data_2[j,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_2[j,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#Removing empty data from first destination location
#campus.loc <- store.data_2[-index, byrow=TRUE]
############################################################
#Second Destination
second <- df$Q8.12
second <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", second) #remove all punctuation
second <- gsub("[a-zA-Z]","", second) #get rid of all upper and lower case letters

store.data_3 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_3) <- c("Latitude.2", "Longitude.2")

for (k in 1:2892){
  if (nchar(second[k]) > 0){
    split <-strsplit(second[k], "[[:space:]]")
    store.data_3[k,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_3[k,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#Removing empty data from second destination location
#index_2 = which(df$Q8.10 == "" | df$Q8.10 == "No")
#second.loc <- store.data_3[-index_2, byrow=TRUE]
############################################################
#Third Destination
third <- df$Q8.19
third <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", third) #remove all punctuation
third <- gsub("[a-zA-Z]","", third) #get rid of all upper and lower case letters

store.data_4 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_4) <- c("Latitude.3", "Longitude.3")
```

```
for (n in 1:2892){
  if (nchar(third[n]) > 0){
    split <-strsplit(third[n], "[[:space:]]")
    store.data_4[n,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_4[n,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#Removing empty data from third destination location
#index_3 = which(df$Q8.17 == "" | df$Q8.17 == "No")
#third.loc <- store.data_4[-index_3, byrow=TRUE]
############################################################
#Fourth Destination
fourth <- df$Q8.26
fourth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", fourth) #remove all punctuation
fourth <- gsub("[a-zA-Z]","", fourth) #get rid of all upper and lower case letters

store.data_5 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_5) <- c("Latitude.4", "Longitude.4")

for (m in 1:2892){
  if (nchar(fourth[m]) > 0){
    split <-strsplit(fourth[m], "[[:space:]]")
    store.data_5[m,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_5[m,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#Removing empty data from third destination location
#index_4 = which(df$Q8.24 == "" | df$Q8.24 == "No")
#fourth.loc <- store.data_5[-index_4, byrow=TRUE]
############################################################
#Fifth Destination
fifth <- df$Q8.33
fifth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", fifth) #remove all punctuation
fifth <- gsub("[a-zA-Z]","", fifth) #get rid of all upper and lower case letters

store.data_6 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_6) <- c("Latitude.5", "Longitude.5")
```

```
for (p in 1:2892){
  if (nchar(fifth[p]) > 0){
    split <-strsplit(fifth[p], "[[:space:]]")
    store.data_6[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_6[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#Removing empty data from third destination location
#index_5 = which(df$Q8.31 == "" | df$Q8.31 == "No")
#fifth.loc <- store.data_6[-index_5, byrow=TRUE]
############################################################
#Sixth Destination
sixth <- df$Q8.40
sixth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", sixth) #remove all punctuation
sixth <- gsub("[a-zA-Z]","", sixth) #get rid of all upper and lower case letters

store.data_7 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_7) <- c("Latitude.6", "Longitude.6")

for (p in 1:2892){
  if (nchar(sixth[p]) > 0){
    split <-strsplit(sixth[p], "[[:space:]]")
    store.data_7[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_7[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

############################################################
#Seventh Destination
seventh <- df$Q8.47
seventh <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", seventh) #remove all punctuation
seventh <- gsub("[a-zA-Z]","", seventh) #get rid of all upper and lower case letters

store.data_8 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_8) <- c("Latitude.7", "Longitude.7")

for (p in 1:2892){
  if (nchar(seventh[p]) > 0){
    split <-strsplit(seventh[p], "[[:space:]]")
    store.data_8[p,1] <- as.numeric(lapply(split, '[[', 2))
```

```
    store.data_8[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#########################################################
#Eighth Destination
eighth <- df$Q8.54
eighth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", eighth) #remove all punctuation
eighth <- gsub("[a-zA-Z]","", eighth) #get rid of all upper and lower case letters

store.data_9 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_9) <- c("Latitude.8", "Longitude.8")

for (p in 1:2892){
  if (nchar(eighth[p]) > 0){
    split <-strsplit(eighth[p], "[[:space:]]")
    store.data_9[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_9[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#########################################################
#ninth Destination
ninth <- df$Q8.61
ninth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", ninth) #remove all punctuation
ninth <- gsub("[a-zA-Z]","", ninth) #get rid of all upper and lower case letters

store.data_10 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_10) <- c("Latitude.9", "Longitude.9")

for (p in 1:2892){
  if (nchar(ninth[p]) > 0){
    split <-strsplit(ninth[p], "[[:space:]]")
    store.data_10[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_10[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#10th Dest
tenth <- df$Q8.68
tenth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", tenth) #remove all punctuation
```

```
tenth <- gsub("[a-zA-Z]","", tenth) #get rid of all upper and lower case letters

store.data_11 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_11) <- c("Latitude.10", "Longitude.10")

for (p in 1:2892){
  if (nchar(tenth[p]) > 0){
    split <-strsplit(tenth[p], "[[:space:]]")
    store.data_11[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_11[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#Eleventh Dest
eleventh <- df$Q8.75
eleventh <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", eleventh) #remove all punctuation
eleventh <- gsub("[a-zA-Z]","", eleventh) #get rid of all upper and lower case letters

store.data_12 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_12) <- c("Latitude.11", "Longitude.11")

for (p in 1:2892){
  if (nchar(eleventh[p]) > 0){
    split <-strsplit(eleventh[p], "[[:space:]]")
    store.data_12[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_12[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}



#twelvth Dest
twelfth <- df$Q8.82
twelfth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", twelfth) #remove all punctuation
twelfth <- gsub("[a-zA-Z]","", twelfth) #get rid of all upper and lower case letters

store.data_13 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_13) <- c("Latitude.12", "Longitude.12")

for (p in 1:2892){
  if (nchar(twelfth[p]) > 0){
    split <-strsplit(twelfth[p], "[[:space:]]")
```

```
    store.data_13[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_13[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#thirteenth Dest
thirteenth <- df$Q8.89
thirteenth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", thirteenth) #remove all punctuation
thirteenth <- gsub("[a-zA-Z]","", thirteenth) #get rid of all upper and lower case letters

store.data_14 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_14) <- c("Latitude.13", "Longitude.13")

for (p in 1:2892){
  if (nchar(thirteenth[p]) > 0){
    split <-strsplit(thirteenth[p], "[[:space:]]")
    store.data_14[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_14[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#fourteenth Dest
fourteenth <- df$Q8.96
fourteenth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", fourteenth) #remove all punctuation
fourteenth <- gsub("[a-zA-Z]","", fourteenth) #get rid of all upper and lower case letters

store.data_15 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_15) <- c("Latitude.14", "Longitude.14")

for (p in 1:2892){
  if (nchar(fourteenth[p]) > 0){
    split <-strsplit(fourteenth[p], "[[:space:]]")
    store.data_15[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_15[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

#fifteenth Dest
fifteenth <- df$Q8.103
fifteenth <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","", fifteenth) #remove all punctuation
fifteenth <- gsub("[a-zA-Z]","", fifteenth) #get rid of all upper and lower case letters
```

```
store.data_16 <- matrix( 0, nrow=2892, ncol= 2)
colnames(store.data_16) <- c("Latitude.15", "Longitude.15")

for (p in 1:2892){
  if (nchar(fifteenth[p]) > 0){
    split <-strsplit(fifteenth[p], "[[:space:]]")
    store.data_16[p,1] <- as.numeric(lapply(split, '[[', 2))
    store.data_16[p,2] <- as.numeric(lapply(split, '[[', 4))
  }
}


#Merging all Lat and Long Data with trip travel method and why
ID <- c(1:2892)
mat.ID <- as.matrix(ID)

tripmeth = cbind(df$Q8.6, df$Q8.13, df$Q8.20, df$Q8.27, df$Q8.34, df$Q8.41, df$Q8.48, df$Q8.55,
                 df$Q8.62, df$Q8.69, df$Q8.76, df$Q8.83, df$Q8.90, df$Q8.97, df$Q8.104)

tripreas = cbind(df$Q8.9, df$Q8.16, df$Q8.23, df$Q8.30, df$Q8.37, df$Q8.44, df$Q8.51, df$Q8.58,
                 df$Q8.65, df$Q8.72, df$Q8.79, df$Q8.86, df$Q8.93, df$Q8.100, df$Q8.107)


new.store.data <- cbind(mat.ID, store.data)

all_lat <- cbind(new.store.data[,-(3)],store.data_2[,1],store.data_3[,1],store.data_4[,1],
store.data_5[,1],store.data_6[,1],
                 store.data_7[,1],store.data_8[,1],store.data_9[,1],
                 store.data_10[,1],store.data_11[,1],store.data_12[,1],
                 store.data_13[,1],store.data_14[,1],
                 store.data_15[,1],store.data_16[,1])
all_long <- cbind(new.store.data[,-(2)],store.data_2[,2],store.data_3[,2],store.data_4[,2],
store.data_5[,2],store.data_6[,2],
                  store.data_7[,2],store.data_8[,2],store.data_9[,2],
                  store.data_10[,2],store.data_11[,2],store.data_12[,2],
                  store.data_13[,2],store.data_14[,2],store.data_15[,2],store.data_16[,2])

rmv <- which(all_lat[,2]==0)

all_lat <- all_lat[-rmv, byrow=TRUE]
```

```
all_long <- all_long[-rmv, byrow=TRUE]
tripmeth <- tripmeth[-rmv, byrow = TRUE]
tripreas <- tripreas[-rmv, byrow = TRUE]

colnames(all_lat) <- c("ID", "Lat.Start", "Lat.Dest1", "Lat.Dest2", "Lat.Dest3", "Lat.Dest4", "
Lat.Dest5",
                       "Lat.Dest6", "Lat.Dest7", "Lat.Dest8", "Lat.Dest9",
                       "Lat.Dest10", "Lat.Dest11", "Lat.Dest12",
                       "Lat.Dest13", "Lat.Dest14",
                       "Lat.Dest15")
colnames(all_long) <- c("ID", "Long.Start", "Long.Dest1", "Long.Dest2", "Long.Dest3", "Long.Dest4",
"Long.Dest5",
                        "Long.Dest6", "Long.Dest7", "Long.Dest8", "Long.Dest9",
                        "Long.Dest10", "Long.Dest11",
                        "Long.Dest12", "Long.Dest13", "Long.Dest14", "Long.Dest15")

new.lat <- data.frame(all_lat)
new.long <- data.frame(all_long)
##############################################################################################

#Creating a data frame

#library(tidyr)
# df.lat <- data.frame(all_lat)
# df.long <- data.frame(all_long)
#
# lat_longways <- gather(df.lat, Trip, Location, Lat.Start:Lat.Dest5, factor_key=TRUE)
# lat_longways <- lat_longways[order(lat_longways$ID, lat_longways$Trip),]
# long_longways <- gather(df.long, Trip, Location, Long.Start:Long.Dest5, factor_key=TRUE)
# long_longways <- long_longways[order(long_longways$ID, long_longways$Trip),]

new.mat <- NULL
new.mat2 <- NULL
#i is number of rows in new.lat
#j is the number of trips.
for (i in 1:nrow(new.lat)){
  for (j in 1:15){
    if (new.lat[i, j+2]==0 && new.long[i, j+2]==0){
      break
    }
    new.row <- c(new.lat$ID[i],j, new.lat[i,j+1],new.lat[i,j+2], new.long[i,j+1],new.long[i,j+2])
```

```
    new.row2 <- c(tripmeth[i,j], tripreas[i,j])
    new.mat <- rbind(new.mat, new.row)
    new.mat2 <- rbind(new.mat2,new.row2)
  }
}
rownames(new.mat) = NULL
rownames(new.mat2) = NULL
new.mat = data.frame(new.mat,new.mat2)
colnames(new.mat) = c("ID", "Trip Number", "Start Lat",
"End Lat", "Start Long", "End Long", "Method", "Reason")


##############################################################################################
#Adding Start time, Stop time, and duration:

library(tidyr)

h <- c(df$Q8.4_1, df$Q8.8_1, df$Q8.11_1, df$Q8.15_1, df$Q8.18_1,
       df$Q8.22_1, df$Q8.25_1, df$Q8.29_1, df$Q8.32_1, df$Q8.36_1,
       df$Q8.39_1, df$Q8.43_1, df$Q8.46_1, df$Q8.50_1, df$Q8.53_1,
       df$Q8.57_1, df$Q8.60_1, df$Q8.64_1, df$Q8.67_1, df$Q8.71_1,
       df$Q8.74_1, df$Q8.78_1, df$Q8.81_1, df$Q8.85_1, df$Q8.88_1,
       df$Q8.92_1, df$Q8.95_1, df$Q8.99_1, df$Q8.102_1, df$Q8.106_1)
hours <- matrix(h, nrow=2892, ncol=30, byrow=FALSE)

m <- c(df$Q8.4_2, df$Q8.8_2, df$Q8.11_2, df$Q8.15_2, df$Q8.18_2,
       df$Q8.22_2, df$Q8.25_2, df$Q8.29_2, df$Q8.32_2, df$Q8.36_2,
       df$Q8.39_2, df$Q8.43_2, df$Q8.46_2, df$Q8.50_2, df$Q8.53_2,
       df$Q8.57_2, df$Q8.60_2, df$Q8.64_2, df$Q8.67_2, df$Q8.71_2,
       df$Q8.74_2, df$Q8.78_2, df$Q8.81_2, df$Q8.85_2, df$Q8.88_2,
       df$Q8.92_2, df$Q8.95_2, df$Q8.99_2, df$Q8.102_2, df$Q8.106_2)
minutes <- matrix(m, nrow=2892, ncol=30, byrow=FALSE)

d <- c(df$Q8.4_3, df$Q8.8_3, df$Q8.11_3, df$Q8.15_3, df$Q8.18_3,
       df$Q8.22_3, df$Q8.25_3, df$Q8.29_3, df$Q8.32_3, df$Q8.36_3,
       df$Q8.39_3, df$Q8.43_3, df$Q8.46_3, df$Q8.50_3, df$Q8.53_3,
       df$Q8.57_3, df$Q8.60_3, df$Q8.64_3, df$Q8.67_3, df$Q8.71_3,
       df$Q8.74_3, df$Q8.78_3, df$Q8.81_3, df$Q8.85_3, df$Q8.88_3,
       df$Q8.92_3, df$Q8.95_3, df$Q8.99_3, df$Q8.102_3, df$Q8.106_3)
daytime <- matrix(d, nrow=2892, ncol=30, byrow=FALSE)
```

```
survdate = df[,2]
survdate = as.POSIXlt(survdate, format = "%m/%d/%y %H:%M")

newtimealpha <- rep(0,2892)
for (i in 1:30){
  time <- as.matrix(cbind(hours[,i], minutes[,i]))
  time[which(time[,1] == 12)] = 0

  time <- apply(time, 1, paste, collapse=":")
  time <- cbind(time, daytime[,i])

  newtime <- as.POSIXlt(time[,1], format = "%H:%M")
  newtime$hour = newtime$hour + ifelse(time[,2] == "p.m.",12,0)

  newtime$mday = survdate$mday - 1
  newtime$mon = survdate$mon
  newtime$year = survdate$year

  newtimealpha <- data.frame(newtimealpha, newtime)
}

newtimealpha <- as.matrix(newtimealpha)

ID <- c(1:2892)
mat.ID <- as.matrix(ID)

newtimealpha <-cbind(mat.ID, newtimealpha)

newtimealpha <- newtimealpha[,-2]

del.time <- which(is.na(newtimealpha[,2]))
newtimealpha <- newtimealpha[-del.time, byrow=TRUE]
#######################################################################

df.time <- data.frame(newtimealpha)
df.time[,1] = as.numeric(as.character(df.time[,1]))

colnames(df.time) <- c("ID", "Start.Time", "Time.2", "Time.3", "Time.4", "Time.5", "Time.6",
                           "Time.7", "Time.8", "Time.9", "Time.10", "Time.11", "Time.12",
                           "Time.13", "Time.14", "Time.15", "Time.16", "Time.17", "Time.18",
                           "Time.19", "Time.20", "Time.21", "Time.22", "Time.23", "Time.24",
```

```
                          "Time.25", "Time.26", "Time.27", "Time.28", "Time.29", "Time.30")
#time_longways <- gather(df.time, Trip, Time, colnames(df.time)[2:31], factor_key = TRUE)
#time_longways[,2] = as.numeric(time_longways[,2])
                        #Start.Time:Time.30, factor_key=TRUE)
#time_longways <- time_longways[order(time_longways$ID, time_longways$Trip),]

myIDt  = NULL
tript = NULL
startt = NULL
endt = NULL
new.mat_time <- NULL
for (i in 1:nrow(df.time)){
  for (j in 1:15){
    if (is.na(df.time[i, 2*j+1])){
      break
    }
    myIDt = c(myIDt,as.numeric(as.character(df.time$ID[i])))
    tript = c(tript, j)
    startt = c(startt,as.character(df.time[i,2*j]))
    endt = c(endt,as.character(df.time[i,2*j+1]))


    #, df.time[i,j+2],df.time[i,j+3])
  }
}
startt = as.POSIXlt(startt)
endt =  as.POSIXlt(endt)
new.mat_time = data.frame(myIDt,tript,startt,endt)
new.mat_time
##############################

######################## Problematic Data ##################
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids

#ID 109(in Excel)
which(new.mat$ID=="108")
which(new.mat_time$myIDt=="108")
new.mat[236:244,]
```

```
new.mat_time[236:243,]
rmv <- which(new.mat[,1] == 108 & new.mat[,2]==8)
new.mat <- new.mat[-rmv, ]
row.names(new.mat) <- 1:dim(new.mat)[1]

#ID 504 (in Excel) uses pm where should have am so remove entire participant
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="503")
which(new.mat_time$myIDt=="503")
new.mat[1160:1164,]
new.mat_time[1160:1163,]
rmv.1 <- which(new.mat[,1] == 503 & new.mat[,2]==1)
rmv.2 <- which(new.mat[,1] == 503 & new.mat[,2]==2)
rmv.3 <- which(new.mat[,1] == 503 & new.mat[,2]==3)
rmv.4 <- which(new.mat[,1] == 503 & new.mat[,2]==4)
new.mat <- new.mat[-c(rmv.1,rmv.2,rmv.3,rmv.4), ]
row.names(new.mat) <- 1:dim(new.mat)[1]

rm.1 <- which(new.mat_time[,1] == 503 & new.mat_time[,2]==1)
rm.2 <- which(new.mat_time[,1] == 503 & new.mat_time[,2]==2)
rm.3 <- which(new.mat_time[,1] == 503 & new.mat_time[,2]==3)
new.mat_time <- new.mat_time[-c(rm.1,rm.2,rm.3), ]
row.names(new.mat_time) <- 1:dim(new.mat_time)[1]

#ID 856 (in Excel)
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="855")
which(new.mat_time$myIDt=="855")
new.mat[1872:1876,]
new.mat_time[1872:1880,]
rmv.1 <- which(new.mat_time[,1] == 855 & new.mat_time[,2]==5)
rmv.2 <- which(new.mat_time[,1] == 855 & new.mat_time[,2]==6)
rmv.3 <- which(new.mat_time[,1] == 855 & new.mat_time[,2]==7)
rmv.4 <- which(new.mat_time[,1] == 855 & new.mat_time[,2]==8)
new.mat_time <- new.mat_time[-c(rmv.1,rmv.2, rmv.3, rmv.4), ]
```

```
row.names(new.mat_time) <- 1:dim(new.mat_time)[1]

#ID 968 (in Excel)
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="967")
which(new.mat_time$myIDt=="967")
new.mat[2139:2150,]
new.mat_time[2139:2148,]
rmv.1 <- which(new.mat[,1] == 967 & new.mat[,2]==7)
rmv.2 <- which(new.mat[,1] == 967 & new.mat[,2]==8)
new.mat <- new.mat[-c(rmv.1,rmv.2), ]
row.names(new.mat) <- 1:dim(new.mat)[1]

#ID 1061 (in Excel)
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="1060")
which(new.mat_time$myIDt=="1060")
new.mat[2343:2349,]
new.mat_time[2343:2350,]
rmv.1 <- which(new.mat_time[,1] == 1060 & new.mat_time[,2]==7)
new.mat_time <- new.mat_time[-rmv.1, ]
row.names(new.mat_time) <- 1:dim(new.mat_time)[1]

#ID 1081 (in Excel)
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="1080")
which(new.mat_time$myIDt=="1080")
new.mat[2385:2390,]
new.mat_time[2385:2388,]
rmv.1 <- which(new.mat[,1] == 1080 & new.mat[,2]==4)
rmv.2 <- which(new.mat[,1] == 1080 & new.mat[,2]==5)
new.mat <- new.mat[-c(rmv.1,rmv.2), ]
```

```
row.names(new.mat) <- 1:dim(new.mat)[1]

#ID 1085 (in Excel)
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="1084")
which(new.mat_time$myIDt=="1084")
new.mat[2392:2397,]
new.mat_time[2392:2401,]
rm.1 <- which(new.mat_time[,1] == 1084 & new.mat_time[,2]==6)
rm.2 <- which(new.mat_time[,1] == 1084 & new.mat_time[,2]==7)
rm.3 <- which(new.mat_time[,1] == 1084 & new.mat_time[,2]==8)
rm.4 <- which(new.mat_time[,1] == 1084 & new.mat_time[,2]==9)
new.mat_time <- new.mat_time[-c(rm.1,rm.2,rm.3, rm.4), ]
row.names(new.mat_time) <- 1:dim(new.mat_time)[1]


#ID 1161 (in Excel)
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="1160")
which(new.mat_time$myIDt=="1160")
new.mat[2549:2556,]
new.mat_time[2549:2553,]
rmv.1 <- which(new.mat[,1] == 1160 & new.mat[,2]==5)
rmv.2 <- which(new.mat[,1] == 1160 & new.mat[,2]==6)
rmv.3 <- which(new.mat[,1] == 1160 & new.mat[,2]==7)
new.mat <- new.mat[-c(rmv.1,rmv.2,rmv.3), ]
row.names(new.mat) <- 1:dim(new.mat)[1]

#ID 1256 (in Excel)
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="1255")
which(new.mat_time$myIDt=="1255")
```

```
new.mat[2754:2760,]
new.mat_time[2754:2759,]
rmv.1 <- which(new.mat[,1] == 1255 & new.mat[,2]==6)
new.mat <- new.mat[-rmv.1, ]
row.names(new.mat) <- 1:dim(new.mat)[1]

#ID 1810 (in Excel)
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="1809")
which(new.mat_time$myIDt=="1809")
new.mat[3875:3880,]
new.mat_time[3875:3884,]
rm.1 <- which(new.mat_time[,1] == 1809 & new.mat_time[,2]==6)
rm.2 <- which(new.mat_time[,1] == 1809 & new.mat_time[,2]==7)
rm.3 <- which(new.mat_time[,1] == 1809 & new.mat_time[,2]==8)
rm.4 <- which(new.mat_time[,1] == 1809 & new.mat_time[,2]==9)
new.mat_time <- new.mat_time[-c(rm.1,rm.2,rm.3, rm.4), ]
row.names(new.mat_time) <- 1:dim(new.mat_time)[1]

#ID 1970 (in Excel)
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="1969")
which(new.mat_time$myIDt=="1969")
new.mat[4261:4270,]
new.mat_time[4261:4264,]
rmv.1 <- which(new.mat[,1] == 1969 & new.mat[,2]==4)
rmv.2 <- which(new.mat[,1] == 1969 & new.mat[,2]==5)
rmv.3 <- which(new.mat[,1] == 1969 & new.mat[,2]==6)
rmv.4 <- which(new.mat[,1] == 1969 & new.mat[,2]==7)
rmv.5 <- which(new.mat[,1] == 1969 & new.mat[,2]==8)
rmv.6 <- which(new.mat[,1] == 1969 & new.mat[,2]==9)
new.mat <- new.mat[-c(rmv.1,rmv.2,rmv.3,rmv.4,rmv.5,rmv.6), ]
row.names(new.mat) <- 1:dim(new.mat)[1]

#ID 2862 (in Excel)
```

```
table1 = table(new.mat$ID)
table2 = table(new.mat_time$myIDt)
ids = as.numeric(names(which(table1-table2 != 0)))
ids
which(new.mat$ID=="2861")
which(new.mat_time$myIDt=="2861")
new.mat[5379:5384,]
new.mat_time[5379:5382,]
rmv.1 <- which(new.mat[,1] == 2861 & new.mat[,2]==4)
rmv.2 <- which(new.mat[,1] == 2861 & new.mat[,2]==5)
new.mat <- new.mat[-c(rmv.1,rmv.2), ]
row.names(new.mat) <- 1:dim(new.mat)[1]

################################################
#The start of the new Dataset:
################################################
df.trans <- cbind(new.mat, new.mat_time[,3:4])

# m <- rep(0, nrow(df.trans))
# for (i in 1:nrow(df.trans)){
#   m[i] <- df.trans[i,8] - df.trans[i,7]
# }
m = df.trans$endt - df.trans$startt
wrongpmindlast = which(m <= -8*3600 & m >= -12*3600)

df.trans[wrongpmindlast,10] = df.trans[wrongpmindlast,10] + 12*3600

wrongpmindfirst = which(m <= 16*3600 & m >= 12*3600)
df.trans[wrongpmindfirst,9] = df.trans[wrongpmindfirst,9] + 12*3600

m = df.trans$endt  - df.trans$startt
badtrips = which(m < 0 | m > 3600*4)
cleantrans = df.trans[-badtrips,c(1,2,3,4,5,6,9,10,7,8)]
#
# m2 = df.trans[,8]- df.trans[,7]
#
# df.trans[23,]

#Add on extra info
#JUST THIS
idcovs = data.frame(ID,df$Q3.2)
```

```
cleantranswcovs = data.frame(cleantrans,idcovs[match(cleantrans$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:10)] = c("Student Title")

#adding home street
idcovs = data.frame(ID,df$Q3.10_1)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:11)] = c("Home Street")

#adding nearest cross street
idcovs = data.frame(ID,df$Q3.10_3)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:12)] = c("Nearest.Cross.Street")

#adding driver status
idcovs = data.frame(ID,df$Q6.4)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:13)] = c("Driver.Status")

#adding vehicle type
idcovs = data.frame(ID,df$Q6.5)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:14)] = c("Vehicle Type")

#adding car make
idcovs = data.frame(ID,df$Q6.7)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:15)] = c("Vehicle.Make")

#adding car model
idcovs = data.frame(ID,df$Q6.8_1)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:16)] = c("Vehicle.Model")

#adding car year
```

```
idcovs = data.frame(ID,df$Q6.8_2)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:17)] = c("Vehicle.Year")

#adding fuel
idcovs = data.frame(ID,df$Q6.6)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:18)] = c("Fuel.Type")

#adding college year
idcovs = data.frame(ID,df$Q3.3)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:19)] = c("Year")

#adding college association
idcovs = data.frame(ID,df$Q3.4)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:20)] = c("College")

#adding Live on campus
idcovs = data.frame(ID,df$Q3.6)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])
#AND THIS!
names(cleantranswcovs)[-(1:21)] = c("Live on Campus")

#location which you must likely park or get dropped off
drop.loc <- df$Q6.10[1:2892]
drop.loc <- gsub("[]$*+?[^{|(\\#%&~_/<=>'!,:;???\")}@]","",drop.loc) #remove all punctuation
drop.loc <- gsub("[a-zA-Z]","", drop.loc) #get rid of all upper and lower case letters

drop.mat <- matrix( 0, nrow=2892, ncol= 2)
colnames(drop.mat) <- c("Latitude.Start", "Longitude.Start")

for (i in 1:2892){
  if (nchar(drop.loc[i]) > 0){
    split <-strsplit(drop.loc[i], "[[:space:]]")
    drop.mat[i,1] <- as.numeric(lapply(split, '[[', 2))
```

```
    drop.mat[i,2] <- as.numeric(lapply(split, '[[', 4))
  }
}

drop.mat <- cbind(mat.ID, drop.mat)
colnames(drop.mat) <- c("ID", "Lat.Drop_off", "Long.Drop_off")
rmv <- which(drop.mat[,2]==0)
drop.mat <- drop.mat[-rmv, byrow=TRUE]

#adding drop off location
idcovs = data.frame(drop.mat)
cleantranswcovs = data.frame(cleantranswcovs,idcovs[match(cleantranswcovs$ID,idcovs$ID),-(1)])

#####################################################
######## VEHICLE MPG ##############
#####################################################

df.v <- read.csv("/Users/Jess/vehicles.csv", header=TRUE)
table(df.v$VClass)

#Quality of matches:
#1) Make, Model Year
#2) Make, Class, Year
#3) Class, year
#4) Class, use earliest year

listofmpgvclass = with(df.v,aggregate(comb08, by = list(VClass), mean   ))
listofmpg = with(df.v,aggregate(comb08, by = list(make, model, year), mean   ))
listofmpg2 = with(df.v,aggregate(comb08, by = list(make, VClass, year), mean   ))
listofmpg3 = with(df.v,aggregate(comb08, by = list(VClass, year), mean   ))

#list of mpg gives the average combined miles per gallon across the categories.


#Get quality of matches 1-3
truelabels = c("", as.character(listofmpgvclass[1,1]), "Varies Too Much",
as.character(listofmpgvclass[c(2,3,4,6,7,8,10,11,14,12,13,17,24,25,27,28,29,30,34),1]) )
names(listofmpg) = c("Vehicle.Make", "Vehicle.Model", "Vehicle.Year")
names(listofmpg2) = c("Vehicle.Make", "Vehicle.Type", "Vehicle.Year")
names(listofmpg3) = c("Vehicle.Type", "Vehicle.Year")
```

```
#Get quality of matches 4
listofmpg4 = with(listofmpg3,aggregate(Vehicle.Year, by = list(Vehicle.Type), min))
names(listofmpg4) = c("Vehicle.Type", "Vehicle.Year")
listofmpg5 = merge(listofmpg4,listofmpg3, c("Vehicle.Type", "Vehicle.Year"), all.x = TRUE)
listofmpg5 = listofmpg5[,c(1,3)]

#QOM 1
testit = merge(cleantranswcovs, listofmpg, c("Vehicle.Make", "Vehicle.Model", "Vehicle.Year"), all.x = TRUE)
testit = testit[order(testit$ID,testit$Trip.Number),]
names(testit)[22] = "MPG1"
tomerge = with(testit, aggregate(MPG1, by = list(Vehicle.Type), length))
testit$Vehicle.Type = truelabels[match(testit$Vehicle.Type,tomerge[,1])]
rownames(testit) = c()

#QOM 2
testit2 = merge(testit, listofmpg2, c("Vehicle.Make", "Vehicle.Type", "Vehicle.Year"), all.x = TRUE)
testit2 = testit2[order(testit2$ID,testit2$Trip.Number),]
names(testit2)[23] = "MPG2"
rownames(testit2) = c()

#QOM 3
testit3 = merge(testit2, listofmpg3, c("Vehicle.Type", "Vehicle.Year"), all.x = TRUE)
testit3 = testit3[order(testit3$ID,testit3$Trip.Number),]
names(testit3)[24] = "MPG3"
rownames(testit3) = c()

#QOM 4
testit4 = merge(testit3, listofmpg5, c("Vehicle.Type"), all.x = TRUE)
testit4 = testit4[order(testit4$ID,testit4$Trip.Number),]
names(testit4)[25] = "MPG4"
rownames(testit4) = c()

#GET THE REAL MPG
MPG = testit4$MPG1
MPG[which(is.na(MPG))] = testit4$MPG2[which(is.na(MPG))]
MPG[which(is.na(MPG))] = testit4$MPG3[which(is.na(MPG))]
MPG[which(is.na(MPG))] = testit4$MPG4[which(is.na(MPG))]

co2TailpipeGpm = 8887/MPG
co2TailpipeGpm[which(cleantranswcovs$Fuel.Type == "Diesel")] = 10180/MPG
[which(cleantranswcovs$Fuel.Type == "Diesel")]
```

```
rm(testit,testit2,testit3,testit4)
cleantranswcovs2 = data.frame(cleantranswcovs, MPG, co2TailpipeGpm)

#####################################################################
################## Getting distances ################################
#####################################################################
library(gmapsdistance)

trip = cleantranswcovs2[1,]



origin = paste(trip$Start.Lat,trip$Start.Long, sep = "+")
destination = paste(trip$End.Lat,trip$End.Long, sep = "+")
res = gmapsdistance(origin,destination, mode = "driving")

gmaplabels = c("driving","driving","walking","driving",
"driving","bicycling","transit","driving","bicycling","walking","driving")
allgmaplabs = gmaplabels[match(cleantranswcovs2$Method,unique
(cleantranswcovs2$Method))]

gmaplabels2 = c("driving","driving","walking","driving","driving","bicycling",
"driving","driving","bicycling","walking","driving")
allgmaplabs2 = gmaplabels2
[match(cleantranswcovs2$Method,unique(cleantranswcovs2$Method))]

length(gmaplabels)

distances = rep(0,nrow(cleantranswcovs2))
#Do for 100
#NOW THE FOR LOOP
for(i in 5235:nrow(cleantranswcovs2)){
  trip = cleantranswcovs2[i,]
  origin = paste(trip$Start.Lat,trip$Start.Long, sep = "+")
  destination = paste(trip$End.Lat,trip$End.Long, sep = "+")
  res = gmapsdistance(origin,destination, mode = allgmaplabs2[i])
  distances[i] = res$Distance
}

compdist = distances

#####################################################
```

```
############ Developing Final.Data #################
####################################################

#START HERE
final.data <- data.frame(cleantranswcovs2,distances)

#REMOVE DATA THAT HAS NO START TIME
final.data = final.data[which(!is.na(final.data$startt)),]

#CHANGE DISTANCES INTO MILES
final.data$distances <- final.data$distances*0.000621371

#Change column name
names(final.data)[20] = "Semester"

#Add the write.csv cammond here.
#use row.names = FALSE
#write.csv(final.data, "final_emissions.csv",row.names = F)
#############################################################################################

#############################################################
####### The beginning of "point process data" ###############
#############################################################

#MAKE A BORDER, USE the zip 66502,66503, 66506 data

#Use the data$bbox to get min max
final.data2 = read.csv("final_emissions.csv")
#This is going to give us the box we do the trip analysis at.
minx = min(c(sf.manhattan.02@bbox[1,],sf.manhattan.03@bbox[1,]))
miny = min(c(sf.manhattan.02@bbox[2,],sf.manhattan.03@bbox[2,]))
maxx = max(c(sf.manhattan.02@bbox[1,],sf.manhattan.03@bbox[1,]))
maxy = max(c(sf.manhattan.02@bbox[2,],sf.manhattan.03@bbox[2,]))

#To find subset of trips within
isInBox = function(lat,long,minx,miny,maxx,maxy){
  if(lat >= miny & lat <= maxy & long >= minx & long <= maxx){
    TRUE
  }
  else
    FALSE
```

```
}


whichtripsin = rep(NA, nrow(final.data2))
for(i in 1:nrow(final.data2)){
  whichtripsin[i] = isInBox(final.data2[i,]$Start.Lat, final.data2[i,]
  $Start.Long, minx,miny,maxx,maxy) &
    isInBox(final.data2[i,]$End.Lat, final.data2[i,]$End.Long, minx,miny,maxx,maxy)
}

withincitytrips = final.data[whichtripsin,]

timeinmins = with(withincitytrips, endt-startt)/60
timeinmins = as.numeric(timeinmins)

#Change 0 minute trips to 2 minute trips
timeinmins[which(timeinmins == 0)] = 2

#We know each point i in final trips will have timeinmins[i]
#  datapoints when we do the point process
sum(timeinmins)
#First is ID, second is trip number, point number, lat, long, emissionsifdrive
point.mat = matrix(0,nrow = sum(timeinmins), ncol = 6)

krow = 1
for(i in 1:nrow(withincitytrips)){
  #get paths
  travel.lat = seq(withincitytrips[i,]$Start.Lat, withincitytrips[i,]$End.Lat, length.out = timeinmins[i])
  travel.long = seq(withincitytrips[i,]$Start.Long, withincitytrips[i,]$End.Long, length.out = timeinmins[i])
  tripemit = withincitytrips[i,]$co2TailpipeGpm*withincitytrips[i,]$distances/timeinmins[i]
  for(j in 1:timeinmins[i]){
    #First is ID, second is trip number, third is point number,
    #extrapolatedlat, extrapolatedlong, emissionsifdrive
    datarow = c(withincitytrips[i,]$ID, withincitytrips[i,]$Trip.Number, j,
                travel.lat[j],travel.long[j], tripemit)
    point.mat[krow,]= datarow
    krow = krow+1
  }
}

pathemitdata = data.frame(point.mat)
```

```
names(pathemitdata) = c("ID", "Trip.Number", "Trip.Minute","Minute.Lat","Minute.Long",
"Minute.Emissions")

pathemitdata = merge(pathemitdata, withincitytrips, c("ID", "Trip.Number"))
pathemitdata = pathemitdata[order(pathemitdata$ID,pathemitdata$Trip.Number),]

########################################################################
######################### Adding Actual Emissions ######################
########################################################################

meth.col <- pathemitdata$Method

unique(meth.col)
walk <- which(meth.col == "Walk")
bike <- which(meth.col == "Bicycle")
skate <- which(meth.col == "Skateboard/kick scooter (human powered)")
scooter <- which(meth.col == "Hands free segway/self balance scooter (battery powered)")
electric <- which(meth.col == "Electric bike/moped/scooter")

Drive <- rep(1, nrow(pathemitdata))
Drive[walk] <- 0
Drive[bike] <- 0
Drive[skate] <- 0
Drive[scooter] <- 0
Drive[electric] <- 0

Actual.Emissions <- pathemitdata$Minute.Emissions*Drive
w.NA <- which(is.na(Actual.Emissions))
Actual.Emissions[w.NA] <- 0
pathemitdata <- data.frame(cbind(pathemitdata, Actual.Emissions))

#################################################################
################### Writing Data to CSV #########################
#################################################################
#write.csv(pathemitdata, "path_emissions_data.csv", row.names = FALSE)

###############################################
#START HERE WHEN RUNNING CODE
###############################################

#pathemitdata = read.csv("path_emissions_data.csv")
```

```
#############################################################
#############################################################
#############################################################



#####################################################################
#####################################################################
#The following were added on after the data frame was written.

#Year born, sex, income
idcovs = data.frame(ID, df$Q10.2,df$Q10.3,df$Q10.6)
pathemitdata = data.frame(pathemitdata,idcovs[match(pathemitdata$ID,idcovs$ID),-(1)])
names(pathemitdata)[33:35] = c("YearBorn", "Sex", "Income")
rownames(pathemitdata) = NULL

##### Considering home locations/initial trip locations #######
InitLatLong = with(pathemitdata2, cbind(
  ID[which(Trip.Number == 1 & Trip.Minute == 1)],
  Start.Lat[which(Trip.Number == 1 & Trip.Minute == 1)],
  Start.Long[which(Trip.Number == 1 & Trip.Minute == 1)]))

pathemitdata2 = data.frame(pathemitdata2,InitLatLong[match(pathemitdata2$ID,InitLatLong[,1]),-(1)])
names(pathemitdata2)[(ncol(pathemitdata2) - 1):ncol(pathemitdata2)] = c("FirstLat", "FirstLong")



#write.csv(pathemitdata, "path_emissions_data_plus.csv", row.names = FALSE)



pathemitdata2 = read.csv("path_emissions_data_plus.csv")
Working.df <- read.csv("Working_Data.csv")

#cut into 30 minute intervals
#aggregate, take the sum over 30 minute intervals



#############################################################################
###############Plotting initial Starting/Home locations#####################
#############################################################################
library(rgdal)
```

```
library(raster)

Working.df <- read.csv("Working_Data.csv")

url.2 <- "http://www2.census.gov/geo/tiger/TIGER2015/ZCTA5/tl_2015_us_zcta510.zip"
tmpdir <- tempdir()
file <- basename(url.2)
download.file(url.2, file)
unzip(file, exdir = tmpdir)
shapefile.2 <- paste(tmpdir, "/tl_2015_us_zcta510.shp", sep = "")
ogrListLayers(shapefile.2)

sf.zip <- readOGR(shapefile.2, layer = "tl_2015_us_zcta510")

sf.manhattan.02 <- sf.zip[which(sf.zip@data$GEOID10 == "66502"), ]
sf.manhattan.03 <- sf.zip[which(sf.zip@data$GEOID10 == "66503"), ]
sf.manhattan.06 <- sf.zip[which(sf.zip@data$GEOID10 == "66506"), ]

plot(sf.manhattan.02, col="yellow")
plot(sf.manhattan.03, add = TRUE, col="grey")
plot(sf.manhattan.06, add=TRUE, col="purple")

location <- cbind(Working.df$FirstLong, Working.df$FirstLat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df[notna,])

points(location)



b#############################################################################
######################Plotting Raster Layer#################################
#############################################################################

manhattan <- rbind(sf.manhattan.02, sf.manhattan.03,sf.manhattan.06)

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",fun=sum)
rl.manhattan[is.na(rl.manhattan[])] <- 0
#rl.manhattan <- mask(rl.manhattan,manhattan)

plot(rl.manhattan)
```

```
plot(manhattan ,add=TRUE)
points(pts,pch=20,cex=0.1) #points on raster as well


##########################################################################
######################Model Prediction and Heat Maps#####################################
##########################################################################

#Emissions for first trip
hist(Working.df$Actual.Emissions, main="Histogram of Actual Emissions", col="darkorchid",
     xlab="Emissions in ppm per trip")
length(which(Working.df$Actual.Emissions=="0"))/length(Working.df$Actual.Emissions) #zero emissions %

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)
'm1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)';,


rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred,zlim=c(0,1000))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)


########cleaning time############

#emissions over time
m <-strptime(Working.df$startt, "%Y-%m-%d %H:%M:%S")
head(m)
library(lubridate)
t.lub <- ymd_hms(m)
```

```
h.str <- as.numeric(format(m, "% H")) +
  as.numeric(format(m, "%M"))/60

#cutting and aggregating
head(Working.df)

#MIKE'S WORK:
#giving in between start trips minutes
starttsubtrip = as.POSIXct(Working.df$startt) + 60*(Working.df$Trip.Minute-1)
endtsubtrip = as.POSIXct(Working.df$startt) + 60*(Working.df$Trip.Minute)
#just wanting hour and minute (no day), in the form of total minutes
minstart = as.numeric(format(starttsubtrip, "%H"))*60 + as.numeric(format(starttsubtrip, "%M"))
#cutting minutes into 5 minute intervals
mincuts = cut(minstart, breaks = seq(0, 1440, by = 5), include.lowest = TRUE)
#data frame with sum of emissions by intervals of five minutes
ebttemp = aggregate(Working.df$Minute.Emissions, by = list(mincuts), sum, na.rm = TRUE, drop = FALSE)
#???????
emissionsbytime = data.frame(names(table(mincuts)),0)
emissionsbytime[match(ebttemp[,1],emissionsbytime[,1]),2] = ebttemp[,2]
plot(seq(0,1435,by = 5)/60*100, emissionsbytime[,2], type = "l", xaxp = c(0,2400,8),
     xlab = "Time (24 Hours)", ylab="Emissions", ylim=c(0,54000))

############################boxplots of descriptive stats#########################
attach(Working.df)
library("dplyr")

emissions.byID <- aggregate(Actual.Emissions, by=list(ID), sum)
colnames(emissions.byID) <- c("ID", "Emissions")

temp <- data.frame(Working.df$ID, Working.df$Student.Title, Working.df$YearBorn,
                   Working.df$Live.on.Campus, Working.df$College)
colnames(temp) <- c("ID", "Student.Title", "Year.Born", "Live.on.Campus", "College")

test <- merge(temp, emissions.byID, by ="ID")

#######This is what you need!!!!!!!!##########
no.dups <- test[!duplicated(test),]
head(no.dups)

student.title <- boxplot(no.dups$Emissions~no.dups$Student.Title, col="purple", ylim=c(0,20500),
ylab="Sum of emissions in ppm per person")
```

```
#sccatterplot of emissions by year
plot(no.dups$Year.Born, no.dups$Emissions, xlim=c(1960,2001), pch = 16, cex = .25,
     xlab="Year", ylab="Emissions", col="black", ylim=c(0,20500))
lines(lowess(no.dups$Year.Born, no.dups$Emissions), col="purple", lwd=2, cex=1.5)

Live.on.Campus <- boxplot(no.dups$Emissions~no.dups$Live.on.Campus,
col="purple", ylim=c(0,20500), ylab="Sum of emissions in ppm per person")

table(no.dups$College)
j <- which(no.dups$College %in% unique(no.dups$College)[c(1,2,3,5,6,7)], )
no.dups.col = no.dups[j,]
college <- boxplot(no.dups.col$Emissions~no.dups.col$College,
                   names = c("Ag", "Arts_Sci", "Bus.", "Edu.", "Engineer","Hum.Ecol" ), drop = TRUE,
                   col="purple", ylim=c(0,20500),ylab="Sum of emissions in ppm per person")



############################boxplots of descriptive stats (hypothetical)#######################
attach(Working.df)
library("dplyr")

emissions.byID <- aggregate(hypemissions2, by=list(ID), sum)
colnames(emissions.byID) <- c("ID", "Emissions")

temp <- data.frame(Working.df$ID, Working.df$Student.Title, Working.df$YearBorn,
                   Working.df$Live.on.Campus, Working.df$College)
colnames(temp) <- c("ID", "Student.Title", "Year.Born", "Live.on.Campus", "College")

test <- merge(temp, emissions.byID, by ="ID")

#######This is what you need!!!!!!!!##########
no.dups <- test[!duplicated(test),]
head(no.dups)

student.title <- boxplot(no.dups$Emissions~no.dups$Student.Title, col="purple", ylim=c(0,20500))

#sccatterplot of emissions by year
plot(no.dups$Year.Born, no.dups$Emissions, xlim=c(1960,2001), pch = 16, cex = .25,
     xlab="Year", ylab="Emissions", col="black", ylim=c(0,20500))
lines(lowess(no.dups$Year.Born, no.dups$Emissions), col="purple", lwd=2, cex=1.5)
```

```
Live.on.Campus <- boxplot(no.dups$Emissions~no.dups$Live.on.Campus,
col="purple", ylim=c(0,20500))

table(no.dups$College)
j <- which(no.dups$College %in% unique(no.dups$College)[c(1,2,3,5,6,7)])
no.dups.col = no.dups[j,]
college <- boxplot(no.dups.col$Emissions~no.dups.col$College,
                   names = c("Ag", "Arts_Sci", "Bus.", "Edu.", "Engineer","Hum.Ecol" ), drop = TRUE,
                   col="purple", ylim=c(0,20500))

######## Trying for Emissions over time heat map ######

#SEE EMISSIONS SCRIPT

######## Emissions over time heat map ######

manhattan <- rbind(sf.manhattan.02, sf.manhattan.03,sf.manhattan.06)

#All trips between 2:00am and 5:00am
Working.df2 = Working.df[minstart >= 120 & minstart < 300,]
numhours = 3
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
#rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=sum)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
```

```
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, main="Emissions from 2 a.m. - 5 a.m.", zlim=c(0,500))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 5:00am and 7:00am
Working.df2 = Working.df[minstart >= 300 & minstart < 420,]
numhours = 2
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
#rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=sum)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count
```

```
plot(rl.pred, main="Emissions from 5 a.m. - 7 a.m.", zlim=c(0,500))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 7:00am and 9:00am
Working.df2 = Working.df[minstart >= 420 & minstart < 540,]
numhours = 2
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
#rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=sum)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)
m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,120))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)



#All trips between 9:00am and 11:00am
Working.df2 = Working.df[minstart >= 540 & minstart < 660,]
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
```

```
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,600))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 11:00am and 1:00pm
Working.df2 = Working.df[minstart >= 660 & minstart < 780,]
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
```

```
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,355))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 1:00pm and 4:00pm
Working.df2 = Working.df[minstart >= 780 & minstart < 960,]
numhours <- 3
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
```

```
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,600))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 4:00pm and 6:30pm
Working.df2 = Working.df[minstart >= 960 & minstart < 1110,]
numhours <- 2.5
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

#m1 <- gam(emissions~s(lat,long,bs="sos"),family=tw(link = log),data=df)
m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,600))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)
```

```
#All trips between 6:30pm and 9:00pm
Working.df2 = Working.df[minstart >= 1110 & minstart < 1260,]
numhours <- 2.5
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)
m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, main="Emissions from 6:30 p.m. - 9 p.m.", zlim=c(0,500))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)
################################################################################
#All trips between 9:00pm and 2:00am
Working.df2 = Working.df[minstart >= 1260 | minstart < 120 ,]
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=sum)
```

```
rl.manhattan[is.na(rl.manhattan[])] <- 0

points(location)
df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,270))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#How do emissions change if people walk for trips < or = 0.5 miles?

hypemissions <- Working.df$Actual.Emissions
hypemissions[which(Working.df$distances <= 0.5)] <- 0
Working.df$hypemissions <- hypemissions
################################################################################
manhattan <- rbind(sf.manhattan.02, sf.manhattan.03,sf.manhattan.06)

#All trips between 7:00am and 9:00am
Working.df2 = Working.df[minstart >= 420 & minstart < 540,]
numhours = 2
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"hypemissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
```

```
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred)
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)



#All trips between 9:00am and 11:00am
Working.df2 = Working.df[minstart >= 540 & minstart < 660,]
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan, "hypemissions",fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)
```

```
rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

#plot(rl.pred, main="Hypothetical Emissions from 9 a.m. - 11 a.m.", zlim=c(0,500))
plot(rl.pred, zlim=c(0,600))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 11:00am and 1:00pm
Working.df2 = Working.df[minstart >= 660 & minstart < 780,]
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"hypemissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,600))
```

```
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 1:00pm and 4:00pm
Working.df2 = Working.df[minstart >= 780 & minstart < 960,]
numhours <- 3
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"hypemissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,600))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 4:00pm and 6:30pm
Working.df2 = Working.df[minstart >= 960 & minstart < 1110,]
numhours <- 2.5
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])
```

```
rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
#rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan <- rasterize(pts,rl.manhattan,"hypemissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,600))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)
##############################################################################################################
##############################################################################################################
##############################################################################################################
#How do emissions change if people walk for trips < or = 1 mile?

hypemissions2 <- Working.df$Actual.Emissions
hypemissions2[which(Working.df$distances <= 1)] <- 0
Working.df$hypemissions2 <- hypemissions2

#All trips between 7:00am and 9:00am
Working.df2 = Working.df[minstart >= 420 & minstart < 540,]
numhours = 2
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
```

```
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"hypemissions2",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,120))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)



#All trips between 9:00am and 11:00am
Working.df2 = Working.df[minstart >= 540 & minstart < 660,]
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan, "hypemissions2" ,
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
```

```
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,600))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 11:00am and 1:00pm
Working.df2 = Working.df[minstart >= 660 & minstart < 780,]
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"hypemissions2",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)
```

```
df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,355))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 1:00pm and 4:00pm
Working.df2 = Working.df[minstart >= 780 & minstart < 960,]
numhours <- 3
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
rl.manhattan <- rasterize(pts,rl.manhattan,"hypemissions2",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,600))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#All trips between 4:00pm and 6:30pm
```

```
Working.df2 = Working.df[minstart >= 960 & minstart < 1110,]
numhours <- 2.5
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
#rl.manhattan <- rasterize(pts,rl.manhattan,"Actual.Emissions",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan <- rasterize(pts,rl.manhattan,"hypemissions2",
fun=function(x,na.rm = TRUE){sum(x,na.rm = na.rm)/numhours})
rl.manhattan[is.na(rl.manhattan[])] <- 0

df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,600) )
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)



#All trips between 9:00pm and 2:00am
Working.df2 = Working.df[minstart >= 1260 | minstart < 120 ,]
location <- cbind(Working.df2$Start.Long, Working.df2$Start.Lat)
notna = which(!is.na(location[,1])) #which are not na
pts <- SpatialPointsDataFrame(location[notna,], Working.df2[notna,])

rl.manhattan <- raster(,xmn = -96.643, xmx = -96.540, ymn = 39.160, ymx = 39.227, res = .001)
```

```
rl.manhattan <- rasterize(pts,rl.manhattan,"hypemissions2",
fun=sum)
rl.manhattan[is.na(rl.manhattan[])] <- 0

points(location)
df <- rasterToPoints(rl.manhattan)
colnames(df) <- c("lat","long","emissions")
df <- as.data.frame(df)

library(mgcv)

m1 <- gam(emissions~s(lat,long,bs="gp"),family=tw(link = log),data=df)

rl.pred <- rl.manhattan
df.pred <- rasterToPoints(rl.pred)
colnames(df.pred) <- c("lat","long","predicted_count")
df.pred <- as.data.frame(df.pred)

df.pred$predicted_count <- predict(m1,newdata=df.pred,type="response")
rl.pred[!is.na(rl.pred[])] <- df.pred$predicted_count

plot(rl.pred, zlim=c(0,270))
plot(manhattan, add=TRUE)
points(pts,pch=20,cex=0.1)

#MIKE'S WORK
#giving in between start trips minutes
starttsubtrip = as.POSIXct(Working.df$startt) + 60*(Working.df$Trip.Minute-1)
endtsubtrip = as.POSIXct(Working.df$startt) + 60*(Working.df$Trip.Minute)
#just wanting hour and minute (no day), in the form of total minutes
minstart = as.numeric(format(starttsubtrip, "%H"))*60 + as.numeric(format(starttsubtrip, "%M"))
#cutting minutes into 5 minute intervals
mincuts = cut(minstart, breaks = seq(0, 1440, by = 5), include.lowest = TRUE)
#data frame with sum of emissions by intervals of five minutes
ebttemp = aggregate(Working.df$hypemissions2, by = list(mincuts), sum, na.rm = TRUE, drop = FALSE)
#???????
emissionsbytime = data.frame(names(table(mincuts)),0)
emissionsbytime[match(ebttemp[,1],emissionsbytime[,1]),2] = ebttemp[,2]
plot(seq(0,1435,by = 5)/60*100, emissionsbytime[,2], type = "l", xaxp = c(0,2400,8),
     xlab = "Time (24 Hours)", ylab="Emissions", ylim=c(0,54000))
```