DSC 365 Project Report: Tracking CTA Ridership

Group 2: Transit Trackers
Sachit Patel, Logan Williams, Kevin Thompson, Abigail Van Nuland, Rohan Dhoyda

Table of Contents

1. Introduction

This report involves the visualization and discussion of data regarding daily ridership of the CTA rail-car train system throughout the city of Chicago. Corresponding datasets were sourced directly from the CTA, through their open-source database on their website. The daily ridership data dates back to the 1970s, counting ridership at all CTA train stations. *Ridership* in this sense is when someone passes through the turnstile at a CTA train station.Various cleaning and visualization techniques were applied to investigate trends within ridership. Additionally, Shiny apps were also created to allow for querying and spatial observations.

Lines of research included visualizing the effect of events on ridership (a significant and recurring example would be Chicago Cubs' games increasing ridership to the Addison Red Line station regularly) and weather (temperature, snowfall, precipitation) and their effects on daily ridership. Additionally, weather data was sourced from Meteoblue's Chicago Weather Data Archive.

1.1. Dataset Description; Attribute Details

Multiple datasets were used for this analysis which were not limited to but included the following: data regarding daily ridership totals, monthly ridership totals and average (by day type, more on that later), and a database list of all CTA stops for cross-reference. Key variables from these datasets include the ridership totals mentioned above, dayType (listed as either weekday, Saturday, or Sunday/holiday), the stop's station ID, and the date. In the selected sourced dataset on weather, key variables include daily records for mean temperature, rainfall, and snowfall. The time period of interest that we chose to focus on was January 2022-June 2023 in large part because of the limited nature of the weather data. It's worth noting that the temperature records are not location specific, meaning they are for the entirety of Chicago. Throughout the analysis, all stations share the same weather data for that day.

2. Data Cleaning & Exploratory Analysis

2.1. Data Cleaning:

Before being able to analyze, combine, and visualize the data, some cleaning and mutations were necessary. Such tasks included but were not limited to: cleaning the date variable (converting from character to date), organizing station_ids into their specific lines and assigning geographical shape values, mutating new variables for month, day, and year based off the date variable, and using an inner join to combine the weather and daily ridership datasets (on date) to then be worked on.

It is important to note that because a single ride constitutes a passenger passing through a turnstile, it is not possible to distinguish what line a passenger rode if they passed through a turnstile at a station with multiple lines. Because of this, we created a category for train line color: “Multiple Lines” to delineate a stop with multiple lines. It is also worth noting that the Lawrence and Berwyn stops on the Red Line were removed from our analysis as they were undergoing construction during the timeframe of study and therefore had zero ridership.

Refer to figure 2.1a in the “additional visualizations” section in the appendix for an exploratory density plot viewing the depth and distribution of ridership according to mean temperature records by day. A specific binning amount was selected to give a “continuous” look to the distribution, whilst being able to point out outliers and dense clusters with the viridis color scale.

2.2. Exploratory Analysis Regarding Precipitation:

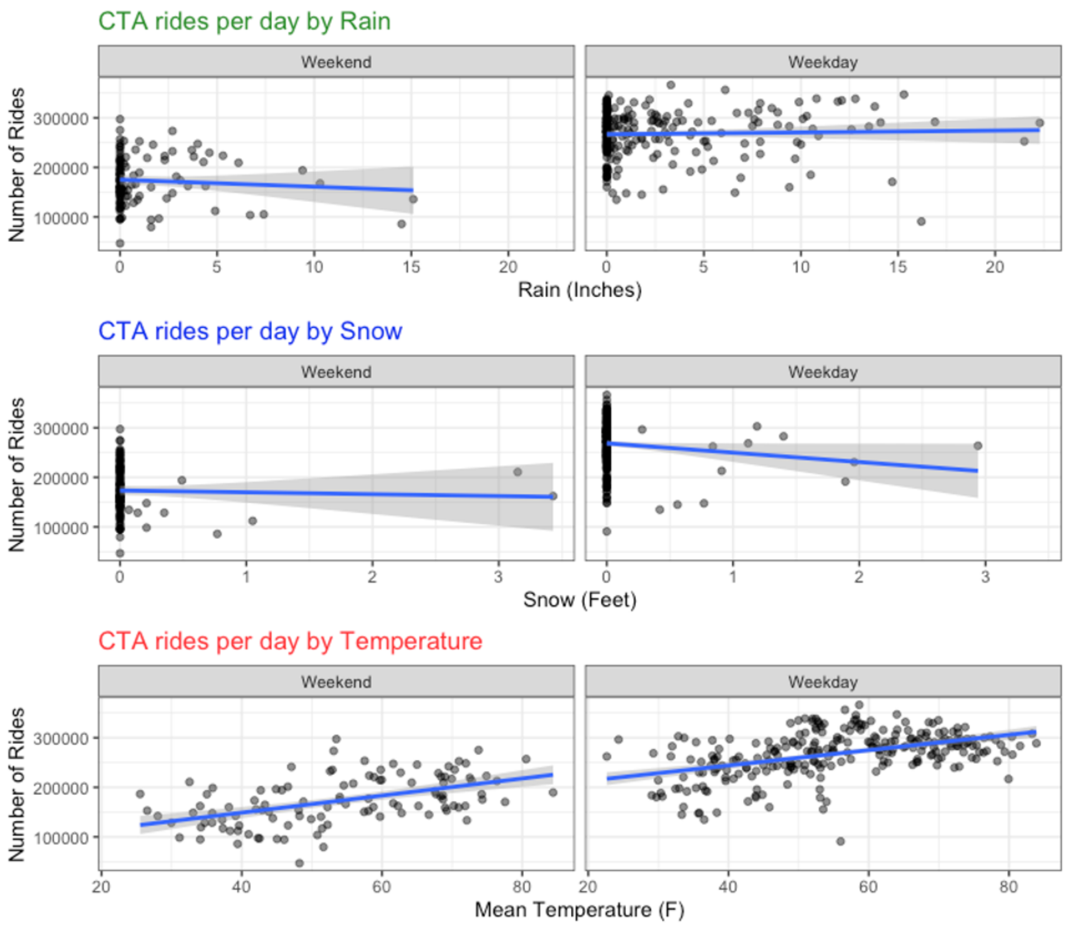
To explore the relationship between precipitation and CTA ridership, exploratory scatterplots were made. The purpose of the scatterplots was to see if there was specifically a linear relationship between precipitation and CTA ridership. If precipitation increased, did ridership go down? The scatterplots initially showed that there was no relationship between the two.

However, when breaking the data down by daytype (saturday, sunday/holiday, and weekdays), we found that there was in fact a negative relationship between CTA ridership and precipitation on Sundays/Holidays (refer to figure 2.2a in the "additional visualizations" section of the appendix). When precipitation increased, ridership went down on those days.

Now having seen this relationship, we were able to take our visualizations further in a later visualization that would only be looking at ridership and the impact of precipitation on Sundays/Holidays. Furthermore, we were also able to make a conclusion based on this exploration: precipitation does not impact the amount of daily CTA riders. We can take this conclusion and use it to make inferences for example: people need to get where they are going regardless of the weather. Specifically on weekdays when people need to get to work.

3. Explanatory Visualizations

3.1. Precipitation and Temperature Visualizations:



-

Figure 3.1a. Multi-plot output detailing CTA ridership against other factors.

This visualization depicts the relationship between CTA ridership, day type, and different types of weather. The grid of scatter plots depict the relationship between the weather (snow, rain, or temperature) and CTA ridership. These plots are then divided between type of day (Weekend or Weekday). Distinction between type of day is important in this dataset because weekday ridership generally depicts people commuting to work, and weekend ridership depicts usage for non-work reasons including local recreation and tourism.

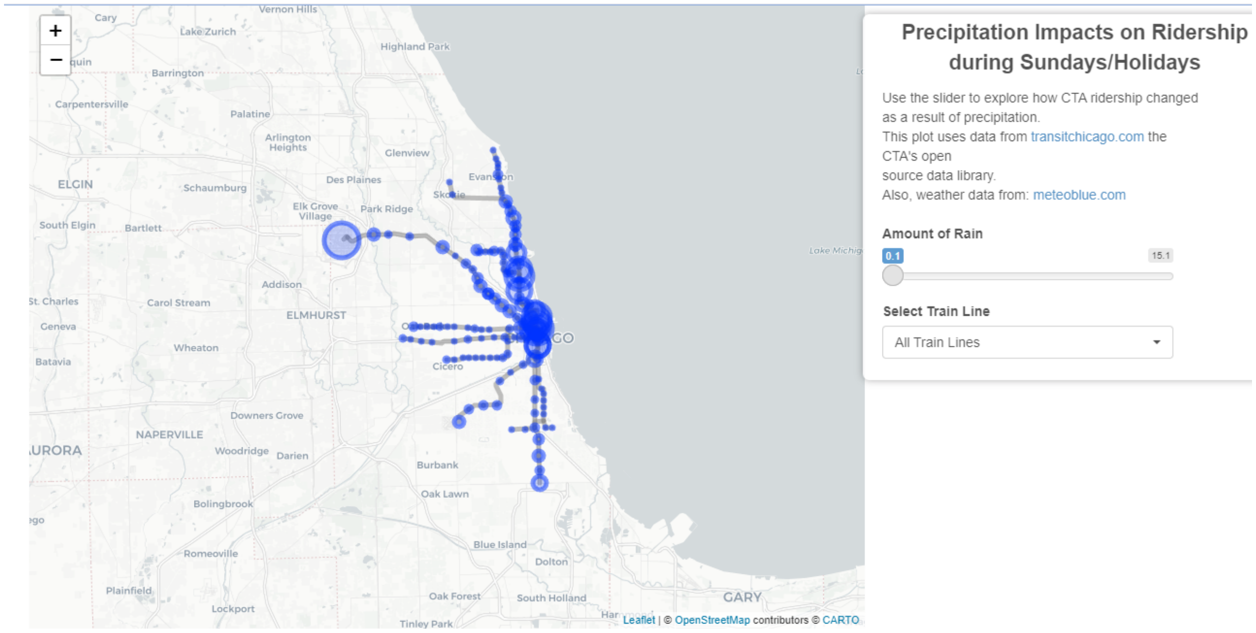
This visualization went through several iterations. At first, we just compared ridership and temperature, rain, and snow. In initial exploration, there was absolutely no relationship between any type of weather and ridership. However, the original dataset includes data from everyday all the way back to the 1970s. We then cleaned up the data by narrowing down the year to be only 2022, and by summing ridership by day, not just looking at data from individual stations. With those changes there was still no relationship between any type of weather and ridership.

However, when we divided ridership up by type of day, patterns began to emerge. Here we see that snow and rain do not have major impacts on ridership, however, perhaps increased temperature has an impact on ridership. For all three types of weather, weekend ridership is lower, but this could be attributed to weekend ridership being overall lower instead of the weather. However, weather does not impact weekday ridership because most people still must go to necessary occupations such as work or school, no matter the weather.

The scatter plot is encoded using its horizontal and vertical positions on the graph. We can compare the two different types of day by comparing unaligned axes. The trend line is encoded using the tilt of the line.

One important lesson learned from this visualization is that data cleaning is an important part of this pipeline. Although the type of visualization did not change a lot, it only expanded and became a grid, the cleaning going into the data became more complex and intricate. This forced us as a group to decide what is important and focus on what questions we were asking.

3.2. Shiny Visualizations:

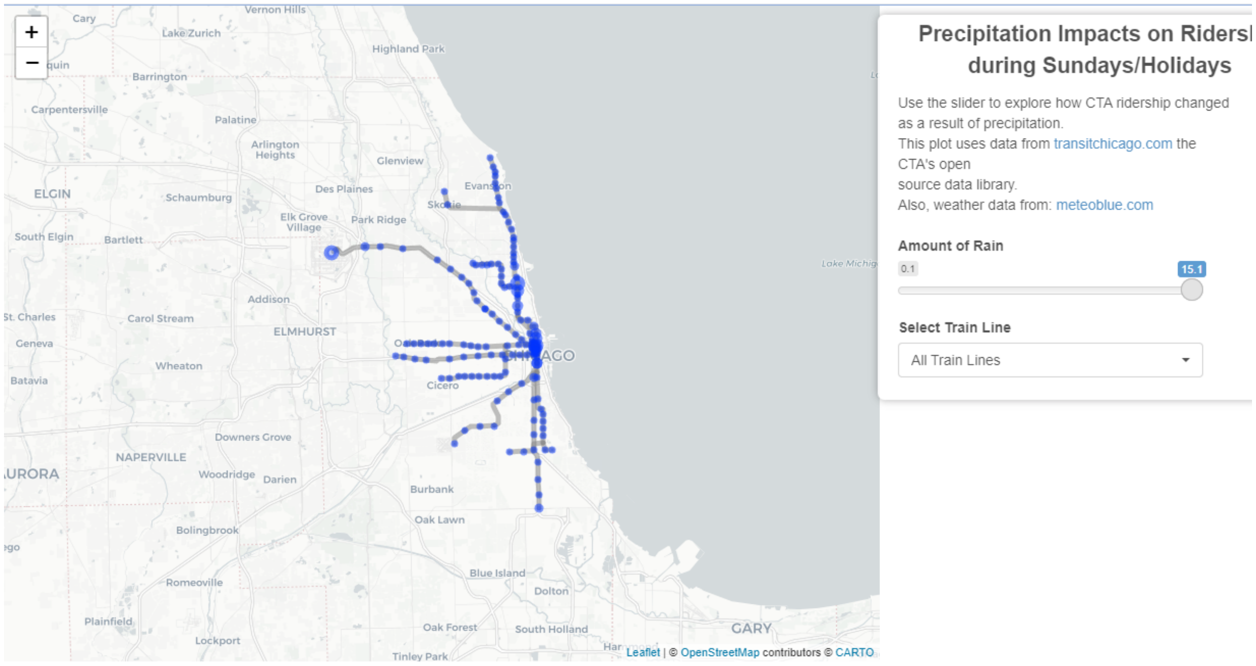


Precipitation Impacts on Ridership during Sundays/Holidays

Use the slider to explore how CTA ridership changed as a result of precipitation.
This plot uses data from transitchicago.com the CTA's open
source data library.
Also, weather data from: meteoblue.com

Amount of Rain: 0.1 (range 0.1 – 15.1)

Select Train Line: All Train Lines



Precipitation Impacts on Riders during Sundays/Holidays

Use the slider to explore how CTA ridership changed as a result of precipitation.
This plot uses data from transitchicago.com the CTA's open
source data library.
Also, weather data from: meteoblue.com

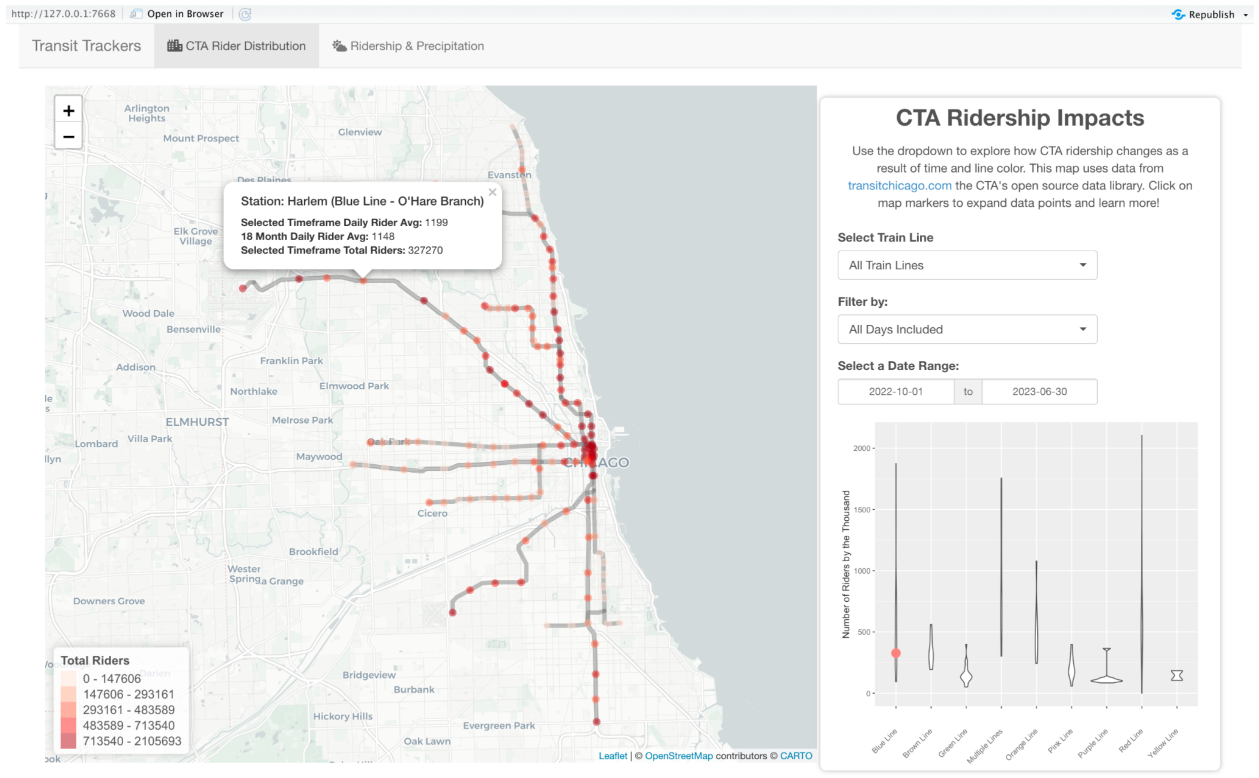
Amount of Rain: 15.1 (range 0.1 – 15.1)

Select Train Line: All Train Lines

Figure 3.2a & 3.2b: Screenshots of a Shiny app built to display precipitation impacts on ridership.

We wanted to visualize the relationship between precipitation and CTA ridership specifically on Sundays/holidays – the days for which we saw a negative linear relationship. We made an interactive cartogram using shiny. A slider was built to select the amount of precipitation. Circles were drawn around each station, and they grew or shrank depending on the amount of daily riders for the day that had the amount of rain selected. When comparing the first, smallest, amount of rain to the last, most, amount of rain, the visualization clearly shows a huge dropoff of ridership. Also, just scrolling through the slider one can see the ridership decrease as the rain increases. This did a great job visualizing the relationship between precipitation and ridership in a way that is more memorable and engaging than the exploratory scatterplot.

The story we wanted to tell was how the amount of precipitation impacted ridership and this visualization does a great job of showing that story dynamically and interactively. It is very obvious to see the relationship between the variables and in a much more interesting way than just looking at a scatterplot.



Station: Harlem (Blue Line - O'Hare Branch)
Selected Timeframe Daily Rider Avg: 1199
18 Month Daily Rider Avg: 1148
Selected Timeframe Total Riders: 327270

Total Riders
- 0 - 147606
- 147606 - 293161
- 293161 - 483589
- 483589 - 713540
- 713540 - 2105693

CTA Ridership Impacts

Use the dropdown to explore how CTA ridership changes as a result of time and line color. This map uses data from transitchicago.com the CTA's open source data library. Click on map markers to expand data points and learn more!

Select Train Line: All Train Lines

Filter by: All Days Included

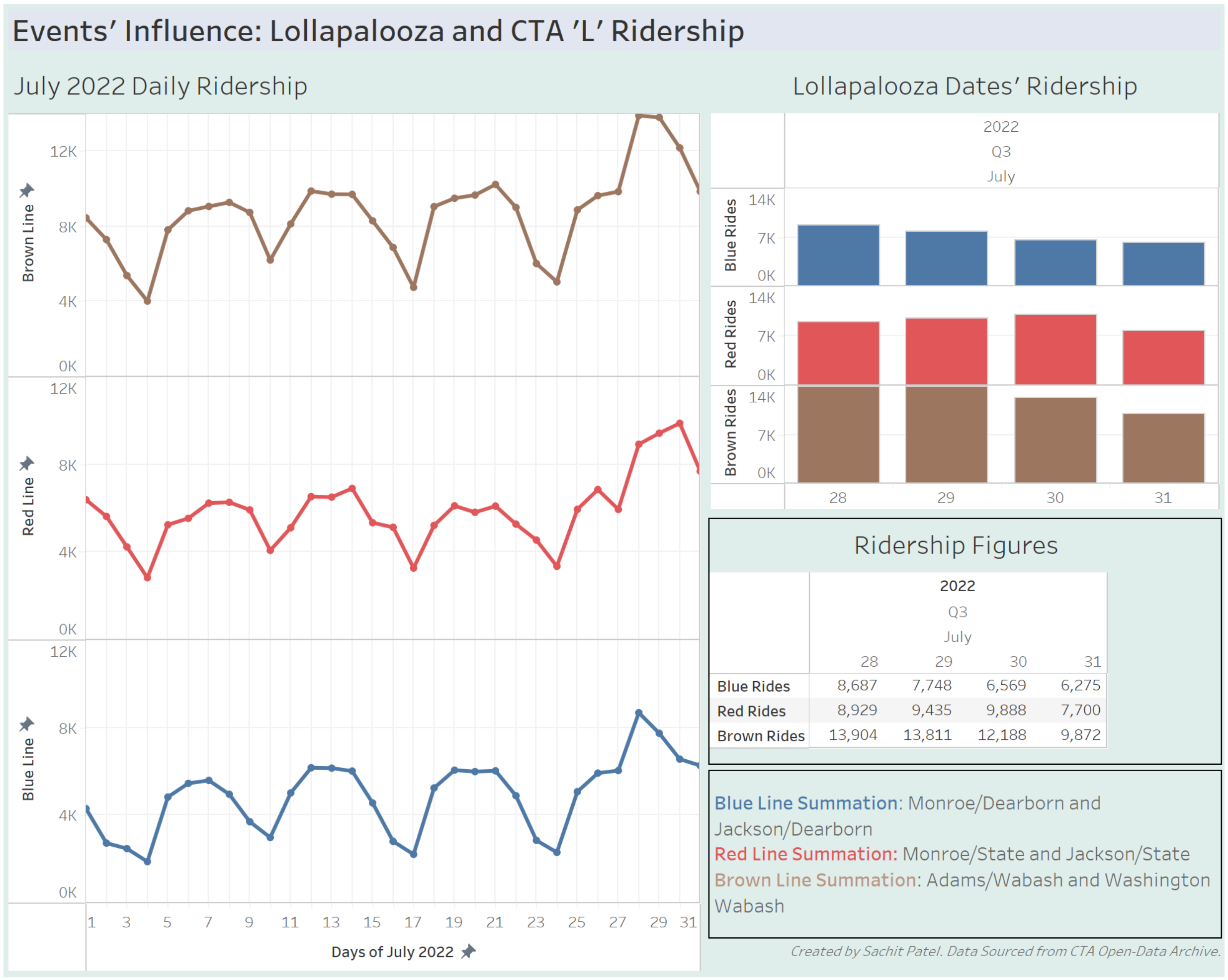
Select a Date Range: 2022-10-01 to 2023-06-30

Figure 3.2c: Screenshot of a Shiny app built to display ridership distributions by train line and day type

Our Shiny app also included an interactive map that allowed users to select a timeframe, day type and line color to see what the overall ridership distribution looked like for their selections. On the bottom right you'll see that the violin plot displays a point based on the user's selection on the map. Overall ridership is mapped to each station and changes based on selections. The violin plot consolidates the data on the map and makes it easier to disseminate between the ridership distributions of different lines. This visualization began as just a map with very little interactivity. As we continued to explore our data and the numerous relationships amongst them, it became clear that we wouldn't be able to explore everything. As such, the Shiny app was born with the purpose of allowing users to explore the countless relationships in our very dense dataset. The app itself went through several iterations as well. Slowly but surely, more functionality was added (dropdowns and date range selections) as well as the violin plot. The map popups themselves used to contain weather related data, but because of the lack of spatial context that came with the weather we removed it and added in averages for each station to provide context to the ridership totals that the user selected.

The story the Shiny app tells is entirely up to the user. As previously stated, the goal was to allow users to explore this dataset as easily as possible. The overall takeaway for this visualization, however, is that CTA ridership depends on a lot of factors. It depends on what line you're on, what kind of day of the week it is, and especially what is happening around the city at the specific station during the date ranges selected.

3.3. Lollapalooza Event Visualization:



Ridership Figures

2022 Q3 July	28	29	30	31
Blue Rides	8,687	7,748	6,569	6,275
Red Rides	8,929	9,435	9,888	7,700
Brown Rides	13,904	13,811	12,188	9,872

Blue Line Summation: Monroe/Dearborn and Jackson/Dearborn
Red Line Summation: Monroe/State and Jackson/State
Brown Line Summation: Adams/Wabash and Washington Wabash

Created by Sachit Patel. Data Sourced from CTA Open-Data Archive.

Figure 3.3a: Tableau Dashboard overview of LollaPalooza 2022 & CTA Ridership.

Figure 3.3a is a dashboard regarding Lollapalooza 2022 in relation to CTA ridership. Ridership data is shown for July 2022, and ridership event dates are individually snapshotted. A table is included to show the exact figures of the bar plots. The month of July 2022 was sampled to help give the viewer some context of what to expect in terms of ridership, and how the ridership during the days of Lollapalooza differs. These 3 lines make up the 6 closest stations to Lollapalooza. People wishing to go to Lollapalooza would most likely exit from one of these stations, as between the red line, blue line, and various stations leading to the upper loop, these stations cover traffic from all directions.

This plot is just a sample of what the shiny app can show. It’s one example of how events regularly affect (and increase) ridership on the various lines. Taking into account other types of events (like Chicago Cubs’ games regularly happening) we can explain many increases in ridership (relative to average ridership). Again, this is just a sample of the data you can query out using the shiny app when selecting a date range.

This dashboard took a lot of refinement to achieve. Individually refining each of the plots (deciding to do a 3 way split for line-graphs, showing ridership by day and encoding by color of train line). Not only that, but spacing out each graph properly on the dashboard, and choosing a color that wasn’t represented between all lines to help blend the visualizations together. Most of the dashboard is allocated to the line graphs to better show the track of ridership. The “stacked” bar plot is really a table of bar charts, but it works well. The specific order was chosen inverse of the line graphs, since brown had the most ridership. This way, the bars on top are lesser than what’s below. The specific numbers of the bars are given in the text table below it for reference, and the encoding legend below that. All-in-all, the layout is to my liking, but it could benefit from extra bolding on text/refined design choices that make the text easier to read.

4. <u>Analysis and Discussion</u>

From the analysis and visualizations of our data we have a few key takeaways. The first of which was the impact of weather on ridership on CTA. While at first we thought we would see that as precipitation and snow increased, ridership would decrease, that was not the case. Generally speaking, CTA ridership is unaffected by weather. The scatterplots used showed there was no strong linear relationship between ridership and precipitation. We also saw in the scatterplot matrix, comparing snow, rain and high temperatures, ridership was minimally impacted by weather.

The takeaway from this is that people need to go where they are going. The only time we really saw a change was on Sundays and holidays. What we can conclude from this is people have to get where they have to go – work, for example. People don't miss work because it's raining. And maybe in some cases, people actually ride the subway more when there's rain because of accidents and traffic. But on days such as holidays, where attendance is less required, people may cancel due to bad weather. Unfortunately, due to the limited nature of our dataset we can’t make any definitive claims on CTA ridership other than mentioning the trends we saw in our data. It’s worth noting that CTA ridership saw a sharp decline in ridership after the COVID-19 pandemic. Perhaps we would have noticed stronger relationships between variables had we examined a different timeframe with more data.

Furthermore, from other plots in figure 3.3a (the Tableau Dashboard regarding LollaPalooza), we saw that ridership increased on days of big events. Festivals, sporting events and so forth, people tend to use the train more. This makes sense as people may be coming from out of town from different directions. Perhaps they don't want to drive through the city or pay for parking. Perhaps they plan to drink and don't want to drive and so on. These events also give a central location that will have high traffic so we will see an increase in ridership in those areas.

To conclude, ridership is very constant. The majority of people ride the CTA for the same reasons every day -- work and their life. If you ride the subway on Monday, chances are you will ride the subway next Monday. Although there may be blips in ridership here or there, CTA riders are overwhelmingly consistent. While events around the city certainly play a role in increased ridership for short durations of time, weather seemed to play a less impactful role in overall ridership. More factors could be drawn in for modeling ridership, perhaps even qualitative data from frequent riders. This analysis didn’t feature many predictors in the dataset, nor modeling.

5. Appendix

5.1. Personal Reflections:

5.1.1. Logan Williams:

As a team member of the Transit Trackers, I played a large role in organizing our team and delegating work to be done. With several members online it was difficult to coordinate our schedules and find time where we all could collaborate, so I ultimately played a role in facilitating online discussions and doling out tasks. In terms of coding contributions, I was a part of the team that drafted and refined our data cleaning code. This was an iterative process. As we continued to get feedback from one another and from the professor, we saw our data needs changing in real time which in term demanded that we find new ways to refine our data cleaning code. This meant we had to be flexible and ready to edit our code based on our new needs. I was also responsible for developing the Shiny App code as well as some preliminary exploratory plots (that were not included in this paper). One of my teammates, Kevin, also was interested in learning Shiny so he and I spent time working together to build out his portion of the app which delved into the effects of rain on ridership.

Like I previously stated about my experience data cleaning, data visualization is similar in that it is an iterative process. The direction we started this project with was not the direction we ended with. As we began to visualize our data we noticed that things we were hoping to explore weren't always there. In that same spirit, I learned that good visualizations always need to think about data, audience, and message. Although the professor harped on it all semester long, this point really became clear when working on several different visualizations to communicate what we saw in the data. Especially with the scale of this project, we needed our visualizations to be coordinated in telling a story which proved to be a difficult task. It wasn't as easy as I had thought it might be. We got some great feedback during our presentation that really opened my eyes to how easy it is to make poor plots. Like we talked about in the early weeks of this course, if a visualization's purpose isn’t immediately clear then it’s a bad visualization. For me, one of the things that helped me approach and craft visualizations was physically writing down the message for each visualization. This served as a reminder to me that this visualization's purpose was to achieve this message. This helped hold me accountable to make sure that the things I was creating were actually achieving what they were designed to. As opposed to making a visualization and saying, “That looks about right. That’ll be super good enough” and moving on.

5.1.2. Abigail Van Nuland:

As a member of this team, I was responsible for creating a visualization to depict the relationship between weather, day type, and number of CTA train rides. I created the grid visualization of transit ridership and the type of day, and how these variables relate to level of rainfall, snow fall, and temperature variance. This grid demonstrates that rain and snow have very little impact on transit ridership, but high temperature may increase CTA ridership.

As a team, we all contributed to each other's visualizations by providing feedback, sharing data cleaning code to ensure we are all using the same data, and by integrating our visualizations into a cohesive story. Collaboration was an important part of this process and we all contributed to each other's visualizations to make a cohesive story.

This visualization went through several exploratory phases before the group landed on the final explanatory plot. The final product was a culmination of feedback from the group. This plot shows that there is really no relationship between transit ridership and weather including snow, rain, and temperature. At first this was surprising, but when we parse out the data by type of day, we see no differences on weekdays, but perhaps a slight downward trend on Saturdays, and Sundays. This makes sense because people still need to ride the train to get to work on weekdays, but on weekends individuals might choose to stay home if the weather is especially bad.

In the process of this project, I learned about how a single set of data can be visualized in many ways. For example, in one of the first exploratory visualizations the data was incredibly cluttered. However, after summing transit data by day instead of station and including an alpha value to fade clustered data points, the pattern became clearer. With the data set, we were able to use many different types of graphs to depict the same story. It also illuminates how data can be manipulated to tell an unclear story by using inappropriate visualizations.

In the past, I’ve only been familiar with visualizations depicting statistical outcomes because I come from a strong statistics background. In this course I’ve been forced to expand my options and try to visualize data in a different way. In my training, I feel it is necessary to always depict relationships and uncertainty, but this class forced me to look at data in a narrative way. In psychology, it is incredibly important to present models of behavior with uncertainty due to all the mediating variables of human behavior. In the process of this project, I learned more about making maps and graphical visualizations, but also time series visualizations.

I also learned a lot about how to deal with cluttered and messy data. The raw data set provided by the CTA is massive and needs to pare down to reduce clutter and to reveal patterns. This data set goes back to the 1970s, so we as a group narrowed down our data to specifically analyze patterns in 2022. When the entire data set was used it was extremely cluttered and unclear. Data cleaning is an important part of visualization that we did not necessarily cover in class but is important for the real world. Using real data taught me this important lesson and may be one of the more important lessons we learned in this class.

One of the last things I learned from this process is understanding the importance of collaboration and feedback. Every person has something to contribute, and it is important to get input from a variety of people because we all have different perspectives. My perspective is largely from a social science background, so the way I view a data set is very different then someone who has a data science perspective. Both are important perspectives that make important contributions to the final product but are vastly different.

5.1.3. Sachit Patel:

I made varying contributions to the group throughout the weeks. Such contributions included both organizational and direct work. Such organizational work included organizing initial communications, organizing and submitting milestone documents, initially setting up the PowerPoint for presentation. Direct work on the project involved sorting out the initial weather dataset we wanted to use (using google drive for easy sharing of the various datasets we grabbed/cleaned), adding more detailed exploratory plots, creating the initial "combined" dataset of the CTA and weather dataset, and creating multi-plot overview of an "event example" regarding CTA ridership.

All-in-all though, these contributions are quite here-and-there. I think my biggest mistake was idling by so much time trying to "get the shiny app working for myself." Not only did this not work out, Logan and Kevin already made their own apps. That's two visualizations done, but I need to be doing something different to contribute to the project. Initially, I went into this project with the mindset of data analysis projects, since I've taken both DSC 323 and DSC 324 before this course. I was used to the final projects of those classes, and the analysis techniques/steps of a project for those classes.

This class' final project was different, since the aim wasn't to do an analysis (through analysis techniques). The goal was to visualize and utilize visualization techniques. We had multiple lines of research, but by the end, they changed. Other group members did look into the weather data. I wanted to also cover events in further detail initially (like Chicago Cubs games) and how that would impact ridership on the CTA fairly regularly. I wasn't able to finish up these visualizations by the time of the presentation. And ultimately, it kind of took a backseat. The shiny apps are quite all-encompassing of the work we've done. The user can request any of the potential event dates they want to see high numbers of. Instead, it is now the multi-plot event visualizations that are used to say “hey, you can find this in the app.” Ultimately, I think I middled out with contributing to the project. I really appreciate the group for their efforts, and not too much infighting like my DSC 324 final project.

What I learned from this project entailed a wide arrange of small RStudio and Tableau insights regarding coding and creating visualizations (looking up documentation, guides, asking questions where I got stuck). This project also gave me the chance to build and tweak a loaded visualization (the tableau dashboard). Walking out of this quarter, there’s a lot I want to try out in Data Visualization for my own, particularly what I didn’t try in the course towards the last few weeks, alongside trying out what my group members did. Because there are a lot of simple quirks when it comes to visualization, the bar for good visualizations is extremely high and ever-increasing. It’s a lot like playtesting.

5.1.4. Kevin Thompson:

Our dataset looked at CTA ridership from the years following covid. We also looked at weather data for Chicago during the same years. We approached the data pretty open-endedly. At first, we wanted to look at covid, but the story was not very interesting. Simply put, covid decreased ridership. This is how we ended up in the years we did. Following that, we thought it'd be best for us to look at what else has an impact on ridership. For this, we looked up weather data for every day and then combined the datasets.

My portion of the project was specifically to look at precipitation and how it affected CTA ridership during our date range. I started looking at the data from an exploratory point of view -- simple scatter plots to check for a relationship between precipitation and amount of daily riders. At first I thought I would see an obvious relationship but this wasn't the case. And after seeing the data visualized, it was clear that there is not really any relationship between precipitation and ridership on the subway. However, there was one relationship which was seen which was on Holidays/Sundays. Ridership decreased as rain increased on those days.

I wanted to explore this relationship further. So I created another visualization which took the map of Chicago and showed circles around the station with sizes relative to the amount of riders. Then I made a slider that allows the user to select an amount of rain. This allows for the user to see the negative linear relationship visualized between precipitation and CTA ridership. Both of these visualizations were made interactively as I felt like it gave a better basis for comparison.

These two visualizations were my specific contributions of visualizations. Beyond that, everyone in the group did a really great job keeping in contact, helping each other out with errors in code, looking at each other's visuals and generally discussing the direction and message of our project. We had a few calls in which we looked for data together, continued to explore the data we did have, and created a cohesive message together.

One thing I learned while doing this project and working with visualizations is that one can go very deep into visualizations. It's really easy to make a simple, and effective, visualization. For example, scatterplots are easy to read and effective. However, the cartogram displayed pretty much the same message (however just for 1 type of day rather than 3). But the visual display of the map I think makes the data much more interesting to look at, much easier to remember, and something that one would automatically understand without having to look at legends, titles, and anything written about it. This reminds me of the lecture of know your audience. And thats probably why you rarely scatter plots in magazines and newspaper – there's no flare.

But to build off of that thought, another thing that was clear from the project was that you get as much out of visualization as you put into it. It's simple to make a couple quick plots or charts or graphs, that are completely adequate. But it seems like visualizations grow from each other, in a sense. Once you learn one thing from one visualization, it sort of makes you want to look into another thing. In this case for example, once we saw that a lot of variables didn't effect ridership, we wondered what did. Things like events in this case affected the ridership. And although that story isn't particularly interesting, there's so much depth to visualization and that was interesting.

The project was interesting and I am proud of what we learned in the class and the application to the project. Everyone in my group did a great job communicating and working together to put together our project.

5.1.5. Rohan Dhoyda:

I actively collaborated with the team. Participation in group meetings was consistent, where I provided ideas, and insights, and stayed informed about the project's progress. This collaborative approach nurtured effective communication among team members, fostering a positive and dynamic team environment.

A key area where I made substantial contributions was helping in the selection of datasets for our project. Another significant contribution was in the creation of visual plots. For example, the Bertin Matrix was designed to visually represent the intricate relationships between mean temperature, days of the week, and average daily ridership. This plot extends beyond mere representation; it allows for a nuanced understanding of data patterns. The thoughtful arrangement of elements on the x and y axes, combined with color-coded intensity, enhances the interpretability of the information conveyed by the matrix.

Similarly, the Bubble Chart was crafted to present hierarchical data in a visually compelling manner. The varying sizes of bubbles within the chart provide an immediate visual cue about differences in values among different categories and subcategories. This type of visualization not only informs but also facilitates the efficient communication of complex hierarchical structures within the data.

Reflecting on the broader impact of the project, it is evident that the skills and knowledge gained during the course significantly influenced the quality of my contributions. The course deepened my understanding of visualization techniques and introduced me to new tools like Tableau and R libraries, expanding my toolkit for creating diverse and impactful visualizations.

Proficiency in Tableau and R libraries contributed substantially to the versatility of my visualizations. These tools provided a dynamic platform for creating interactive and engaging visual representations of the data. The interactive features of Tableau, for instance, allowed for a more immersive exploration of the visualizations, enabling users to interact with the data and derive insights in a personalized manner.

Homework assignments and tutorials provided a structured learning environment, enabling me to apply theoretical knowledge to practical scenarios. The hands-on experience gained through these assignments not only improved my proficiency in R but also equipped me with the skills to navigate and manipulate data effectively. This proficiency was crucial in generating various types of visualizations tailored to different situations, as demanded by the project requirements.

In summary, my contributions to the team encompassed active participation in group meetings, assistance in dataset selection, and the creation of visual plots using tools like Tableau and R libraries. The course helped me understand visualizations better and gave me more tools to use. The skills I learned not only helped finish the project but also made me better at working with data.

5.2. Additional Visualizations:

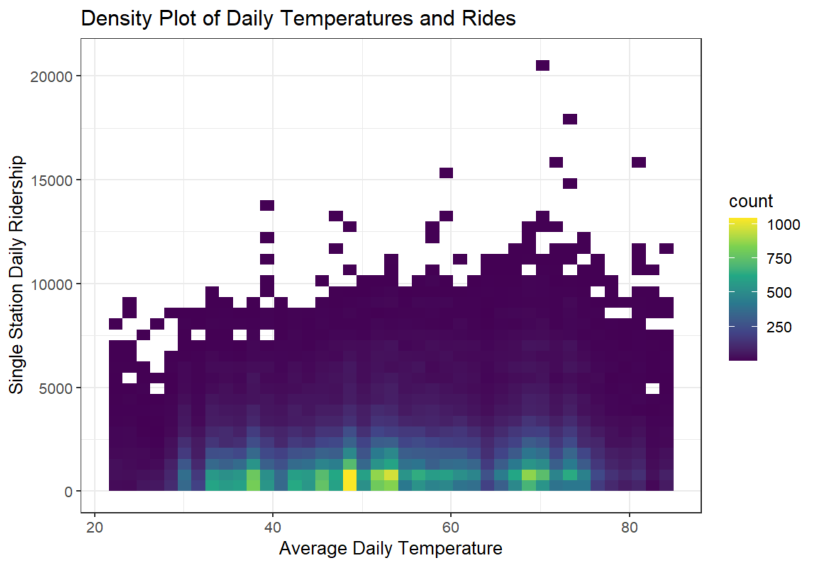
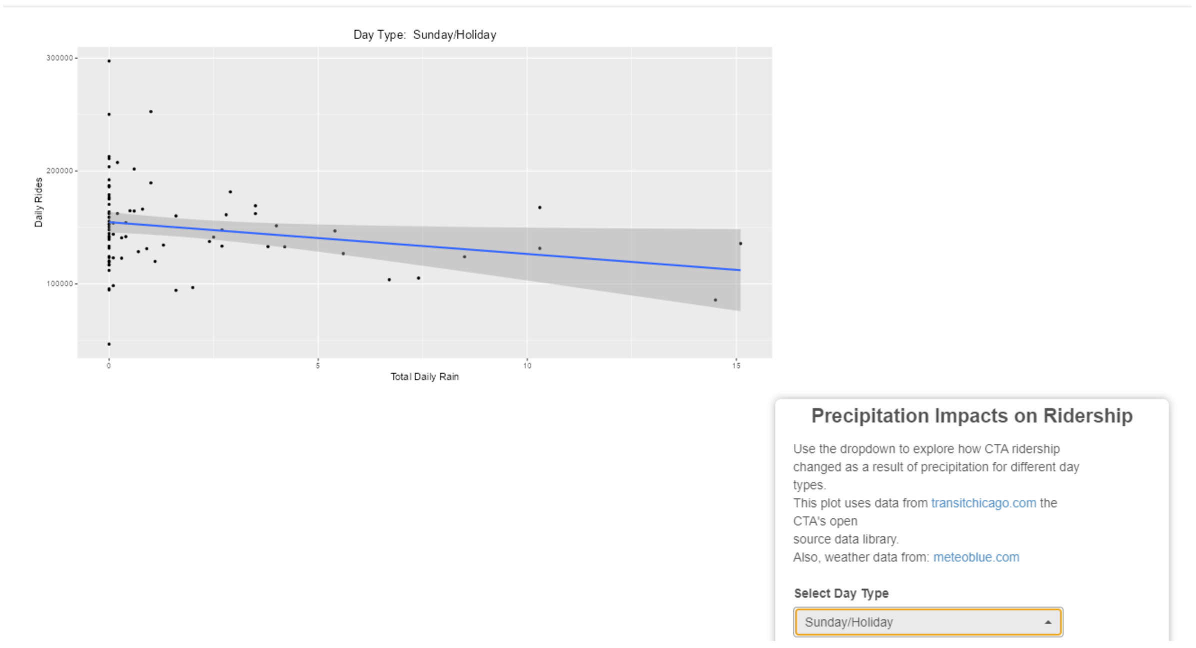


Figure 2.1a. Exploratory Density Plot detailing the count and distribution of ridership in regards to Average Daily Temperature (mean temperature for each day). A specific binning amount was selected to give a "continuous" look to the distribution, whilst being able to point out outliers and dense clusters with the viridis color scale.



Precipitation Impacts on Ridership

Use the dropdown to explore how CTA ridership changed as a result of precipitation for different day types.
This plot uses data from transitchicago.com the CTA's open source data library.
Also, weather data from: meteoblue.com

Select Day Type: Sunday/Holiday

Figure 2.2a: Ridership vs Rainfall plot filtered to Sundays and Holidays.

5.3. Code:

5.3.1. Kevin Shiny Scatterplots:

~~~
---
title: "dataCleaning"
author: "Logan Williams"
date: "11/3/2023"
output: html_document
---

#Load Dependencies
```{r}
library(dplyr)          #Tools for cleaning data
library(here)           #File path management
library(ggplot2)        #Data visualization
library(sf)             #transform shapefile to lat long
library(stringr)        #string manipulation
library(lubridate)      #time date manipulation
library(raster)         #saving shapefiles
#install.packages("rgdal")
#library(rgdal)         #read shapefiles
library(ggpubr)
library(conflicted)
library(sp)
library(shiny)
library(leaflet)
library(shinyWidgets)
conflicts_prefer(dplyr::filter)
conflicts_prefer(dplyr::lag) #try stats if not
~~~

#Load Data
```{r}
daily <- read.csv("C:/Users/19148/Downloads/cleanedDailyTotals.csv")
stops <-
read.csv("C:/Users/19148/Downloads/CTA_-_System_Information_-_List_of__L__Stops_20231
109.csv")
weather <- read.csv("C:/Users/19148/Downloads/dayAverageChicagoWeather.csv")
#load CTA train line shape file & transform to lat/long
#load CTA train line shape file & transform to lat/long
#lineShp <- st_read(here("C:/Users/19148/Downloads/CTA_RailLines.shp"))%>%
 # st_transform('+proj=longlat +datum=WGS84')
```
```
```

~~~
```{r}
#weather row names
colnames(weather) <- c('date','maxTemp','minTemp','meanTemp','sumPrecip','sumSnow')
#format dates
daily$date = ymd(daily$date)
#Decode dayType column
daily$daytype[daily$daytype=="W"] <- "Weekday"
daily$daytype[daily$daytype=="A"] <- "Saturday"
daily$daytype[daily$daytype=="U"] <- "Sunday/Holiday"
#create year, month, and day variables
daily%>%
  mutate(year = format(as.Date(daily$date, format="%d/%m/%Y"),"%Y"))%>%
  mutate(month = format(as.Date(daily$date, format="%d/%m/%Y"),"%m"))%>%
  mutate(day = format(as.Date(daily$date, format="%d/%m/%Y"),"%d"))

#filter for 2022/2023
daily <- daily[order(as.Date(daily$date, format="%m/%d/%Y")),]
daily <- daily %>%
  filter(date >= '2022-01-01')

#cleaning weather dataset
str(weather)
weather$date <- substr(weather$date, 1, nchar(weather$date)-5)
weather$date <- ymd(weather$date)

#filter dataset
weather <- weather %>%
  filter(date <= '2023-06-30')

#left join attempt
joinedData <- left_join(daily, weather, by='date')

#Create station column
stops$lineType<-qdapRegex::ex_between(stops$STATION_DESCRIPTIVE_NAME, "(",")")

#Filter for only distinct stations
stops%>%
  distinct(MAP_ID, Location, lineType,STATION_DESCRIPTIVE_NAME)%>%
  rename(station_id = MAP_ID)-> uniqueStops
uniqueStops$station_id <- as.character(uniqueStops$station_id)
splitLocation <- stringr::str_split_fixed(uniqueStops$Location, ", ",2)
splitLocation <- as.data.frame(splitLocation)
splitLocation$V1 <- str_sub(splitLocation$V1,2)
~~~

```
splitLocation$V2 <- str_sub(splitLocation$V2,1,-2)

splitLocation%>%
  rename(Lat = V1)%>%
  rename(Long = V2) -> geoStop
#rejoin with train station data
geoStopJoin <- bind_cols(uniqueStops, geoStop)
conflicts_prefer(dplyr::select)
geoStopJoin <- select(geoStopJoin, -Location)

#convert lat/long to numeric
geoStopJoin$Lat <- as.numeric(geoStopJoin$Lat)
geoStopJoin$Long <- as.numeric(geoStopJoin$Long)
joinedData$station_id = as.character(joinedData$station_id)
totalDailyJoin <- left_join(joinedData, geoStopJoin)

totalDailyJoin$lineType <- as.character(totalDailyJoin$lineType)

#Create a variable that labels stations with multiple lines
totalDailyJoin%>%
  mutate(genLineType = ifelse(grepl(",", lineType), "Multiple Lines",
                    ifelse(grepl("&", lineType), "Multiple Lines",
                    ifelse(grepl("-", lineType), "Blue Line",
                           lineType)))) -> totalDailyJoin
#Do the same for our shapefile
#lineShp%>%
#  mutate(genLineType = ifelse(grepl("ML", LEGEND), "Multiple Lines",
#                    ifelse(grepl("BL", LEGEND), "Blue Line",
#                    ifelse(grepl("BR", LEGEND), "Brown Line",
#                    ifelse(grepl("YL", LEGEND), "Yellow Line",
#                    ifelse(grepl("GR", LEGEND), "Green Line",
#                    ifelse(grepl("OR", LEGEND), "Orange Line",
#                    ifelse(grepl("RD", LEGEND), "Red Line",
#                    ifelse(grepl("PR", LEGEND), "Purple Line",
#                    ifelse(grepl("PK", LEGEND), "Pink Line",
#                           "NA")))))))))) -> lineShp


```
```{r}
#remove scientific notation
options(scipen=999)
#exploratory scatterplot
```

~~~
joinedData_grouped = joinedData %>% group_by(date, sumPrecip, daytype) %>%
summarize(rides = sum(rides))
joinedData_grouped$sumPrecip =as.numeric((joinedData_grouped$sumPrecip))

p1 = ggplot(joinedData_grouped, aes(x = sumPrecip,y=rides))+geom_point(size = 1) +
geom_smooth(method=lm)+ggtitle("Ridership on Rainy Days")+xlab("Total Daily Rain") +
ylab("Daily Rides") + theme(plot.title = element_text(hjust = .5))

p1

```



```{r}
#remove 0 precip
joinedData_no0 = joinedData_grouped[joinedData_grouped$sumPrecip!= 0., ]
joinedData_no0$sumPrecip = as.numeric(as.character(joinedData_no0$sumPrecip))
#plot

p2 = ggplot(joinedData_no0, aes(x = sumPrecip,y=rides))+geom_point(size = 1) +
geom_smooth(method=lm)+ggtitle("Ridership on Rainy Days")+xlab("Total Daily Rain") +
ylab("Daily Rides") + theme(plot.title = element_text(hjust = .5))

p2
```

```{r}
#check for relationship on weekends
joinedData_weekends = joinedData_grouped[joinedData_grouped$daytype!= "Weekday",]
saturdayOnly = joinedData_weekends[joinedData_weekends$daytype!= "Sunday/Holiday",]
saturdayOnly = saturdayOnly[saturdayOnly$sumPrecip != 0.0,]

p3 = ggplot(joinedData_weekends, aes(x = sumPrecip,y=rides))+geom_point(size = 1) +
geom_smooth(method=lm)+ggtitle("Ridership on Weekends")+xlab("Total Daily Rain") +
ylab("Daily Rides") + theme(plot.title = element_text(hjust = .5))

saturdayPlot = ggplot(saturdayOnly, aes(x = sumPrecip,y=rides))+geom_point(size = 1) +
geom_smooth(method=lm)+ggtitle("Ridership on Saturday")+xlab("Total Daily Rain") +
ylab("Daily Rides") + theme(plot.title = element_text(hjust = .5))

p3 = ggplot(joinedData_weekends, aes(x = sumPrecip,y=rides))+geom_point(size = 1) +
geom_smooth(method=lm)+ggtitle("Ridership on Weekends")+xlab("Total Daily Rain") +
ylab("Daily Rides") + theme(plot.title = element_text(hjust = .5))
~~~

```
p3
```
```{r}
joinedData_weekends_0 = joinedData_weekends[joinedData_weekends$sumPrecip!= "0", ]
p4 = ggplot(joinedData_weekends_0, aes(x = sumPrecip,y=rides))+geom_point(size = 1) +
geom_smooth(method=lm)+ggtitle("Ridership on Weekends")+xlab("Total Daily Rain") +
ylab("Daily Rides") + theme(plot.title = element_text(hjust = .5))


p4

joinedData_holiday_0 = joinedData_weekends_0[joinedData_weekends_0$daytype!=
"Saturday", ]
p5 = ggplot(joinedData_holiday_0, aes(x = sumPrecip,y=rides))+geom_point(size = 1) +
geom_smooth(method=lm)+ggtitle("Ridership on Rainy Holidays")+xlab("Total Daily Rain") +
ylab("Daily Rides") + theme(plot.title = element_text(hjust = .5))

p5
```

```{r}
#ui
ui = fluidPage(navbarPage("Rainy Days",
                              mainPanel(
                              plotOutput(outputId = 'distPlot')
                              )),

                              #Specify panel attributes
                              absolutePanel(top = 485, right = 40, width = "32vw", style =
"background-color: white;
                              opacity: 0.85;
                              padding: 20px 20px 20px 20px;
                              margin: auto;
                              border-radius: 5pt;
                              box-shadow: 0pt 0pt 6pt 0px rgba(61,59,61,0.48);
                              padding-bottom: 2mm;
                              padding-top: 1mm;",

                              #add title and intro text
                              fluidRow(
                                     column(12,
                                     align = "center",
                                     tags$p(strong("Precipitation Impacts on Ridership"), style =
"font-size: 3vh;")),
                                        br(),
                                        column(width = 10,
                                        align = "left",
                                        tags$div(
                                                "Use the dropdown to explore how CTA ridership changed
as a result of precipitation for different day types.",
                                                br(),

                                                "This plot uses data from",
                                                tags$a(href="https://www.transitchicago.com/data/",
"transitchicago.com"),
                                                "the CTA's open",
                                                br(),
                                                "source data library.",
                                                br(),
                                                "Also, weather data from: ",
                                                tags$a(href =
"https://www.meteoblue.com/en/weather/historyclimate/weatherarchive/chicago_united-states_4
887398", "meteoblue.com"),
                                                br(), br()))),

                                #Create dropdown selection box
                                pickerInput("category", label = "Select Day Type",
                                        choices = unique(joinedData_grouped$daytype),
                                        selected = unique(joinedData_grouped$daytype),
                                        multiple = FALSE)))
  # Define server logic required to draw visuals
server = function(input, output, server) {

  choiceData = reactive({
          dfSub = joinedData_grouped|>
          filter(daytype == input$category)
  })

  #choiceData = joinedData_grouped[joinedData_grouped$daytype == input$category, ]

  output$distPlot = renderPlot({
          ggplot(choiceData(), aes(x = sumPrecip,y=rides))+geom_point(size = 1) +
geom_smooth(method=lm)+ggtitle(paste("Day Type: ", choiceData()$daytype[1]))+xlab("Total
Daily Rain") + ylab("Daily Rides") + theme(plot.title = element_text(hjust = .5))
  })
}

shinyApp(ui = ui, server = server)```
```

5.3.2. Kevin Shiny Precipitation Map:

```
#
# This is a Shiny web application. You can run the application by clicking
# the 'Run App' button above.
#
# Find out more about building applications with Shiny here:
#
#         http://shiny.rstudio.com/
#

library(shiny)
library(dplyr)          #Tools for cleaning data
library(here)           #File path management
library(ggplot2)        #Data visualization
library(sf)             #transform shapefile to lat long
library(stringr)        #string manipulation
library(lubridate)      #time date manipulation
library(raster)         #saving shapefiles
#install.packages("rgdal")
#library(rgdal)         #read shapefiles
library(ggpubr)
library(conflicted)
library(sp)
library(shiny)
library(leaflet)
library(shinyWidgets)
conflicts_prefer(dplyr::filter)
conflicts_prefer(dplyr::lag) #try stats if not
library(sf)

df <- read.csv("totalDailyJoin.csv")

lineShp <- raster::shapefile("CTA_RailLines.shp")



#build interactive map visualization
ui2 = fluidPage(
  mainPanel(
        leafletOutput("mymap", height= "89vh")
  ),

  #Specify panel attributes
  absolutePanel(top = 10, right = 40, width = "32vw",style = "background-color: white;
                         opacity: 0.85;
                         padding: 20px 20px 20px 20px;
                         margin: auto;
                         border-radius: 5pt;
                         box-shadow: 0pt 0pt 6pt 0px rgba(61,59,61,0.48);
                         padding-bottom: 2mm;
                         padding-top: 1mm;",


                #add title and intro text
                fluidRow(
                column(12,
                align = "center",
                tags$p(strong("Precipitation Impacts on Ridership during Sundays/Holidays"),
style = "font-size: 3vh;")),
                br(),
                column(width = 10,
                align = "left",
                tags$div(
                "Use the slider to explore how CTA ridership changed as a result of
precipitation.",
                br(),

                "This plot uses data from",
                tags$a(href="https://www.transitchicago.com/data/", "transitchicago.com"),
                "the CTA's open",
                br(),
                "source data library.",
                br(),
                "Also, weather data from: ",
                tags$a(href =
"https://www.meteoblue.com/en/weather/historyclimate/weatherarchive/chicago_united-states_4
887398", "meteoblue.com"),
                br(), br()))),
                #create slider input:
                sliderTextInput("category", "Amount of Rain",
                         choices = c(as.numeric(unique(df$sumPrecip))),
                         selected = range(vals = c(as.numeric(unique(df$sumPrecip))))),
                pickerInput("cat", label = "Select Train Line",
                         choices = unique(df$genLineType),
                         selected = unique(df$genLineType),
                         multiple = TRUE,
                         options = pickerOptions(
                                actionBox = TRUE,
                                #If all train lines are selected, show 'All Train Lines'
                                selectedTextFormat = paste0("count > ",
length(sort(unique(df$genLineType))) -1),
                                countSelectedText = "All Train Lines"))
  ))

server2 = function(input, output, server){
  sliderChoice = reactive({
        dfSub = df|>
        filter(sumPrecip == input$category)
  })
  shapeData = reactive({
        lineShp[lineShp@data$gnLnTyp %in% input$cat,]
  })
  #Create map
  output$mymap <- renderLeaflet({

        #Create a palette & corresponding legend
        pal <- colorQuantile("Reds", sliderChoice()$rides, n = 5)
        palColors <- unique(pal(sort(sliderChoice()$rides)))
        palBreaks <- round(quantile(sliderChoice()$rides, seq(0, 1, .2)),0)
        palLabels <- paste(lag(palBreaks), palBreaks, sep = " - ")[-1] # first lag is NA


        myMap <- leaflet()%>% #filters our data based on dropdown selection
        addProviderTiles(providers$CartoDB.Positron)%>% #adds a basemap
        addPolylines(data = shapeData(), color = "Gray")%>%
        addCircleMarkers(data = sliderChoice(),
              ~Long, ~Lat,
              radius = sliderChoice()$rides/1100,
              label = ~STATION_DESCRIPTIVE_NAME)
  })
}


shinyApp(ui = ui2, server = server2)
```

5.3.3. Sachit WeatherDataCleaning:

```
#weather spreadsheet
#goals:
#extract everything properly
#combine with the original dataset, 2022 to 2023
#plots

library(ggplot2)
library(here)
library(tidyverse)
library(lubridate)
library(ggpubr)

#datasets
dailyTotals <- read.csv("ctaLDailyTotals.csv")
#very large, takes some time to instantiate.
typeTotals <- read.csv("ctaLMonthlyTotals.csv")
stops <- read.csv("ctaStops.csv")
weather <- read.csv("dayAverageChicagoWeather.csv", skip = 9)

#weather row names
colnames(weather) <- c('date','maxTemp','minTemp','meanTemp','sumPrecip','sumSnow')

#format dates
dailyTotals$date = mdy(dailyTotals$date)
#Decode dayType column
dailyTotals$daytype[dailyTotals$daytype=="W"] <- "Weekday"
dailyTotals$daytype[dailyTotals$daytype=="A"] <- "Saturday"
dailyTotals$daytype[dailyTotals$daytype=="U"] <- "Sunday/Holiday"
#create year, month, and day variables
dailyTotals%>%
  mutate(year = format(as.Date(dailyTotals$date, format="%d/%m/%Y"),"%Y"))%>%
  mutate(month = format(as.Date(dailyTotals$date, format="%d/%m/%Y"),"%m"))%>%
  mutate(day = format(as.Date(dailyTotals$date, format="%d/%m/%Y"),"%d"))

#filter for 2022/2023
dailyTotals <- dailyTotals[order(as.Date(dailyTotals$date, format="%m/%d/%Y")),]
dailyTotals <- dailyTotals %>%
  filter(date >= '2022-01-01')

#cleaning weather dataset
str(weather)
weather$date <- substr(weather$date, 1, nchar(weather$date)-5)
weather$date <- ymd(weather$date)
```

```
#filter dataset
weather <- weather %>%
  filter(date <= '2023-06-30')

#left join attempt
joinedData <- left_join(dailyTotals, weather, by='date')
#Create list of red line stops
stops%>%
  filter(RED == "true")%>%
  rename(station_id = MAP_ID)%>%
  select(station_id) -> redLineStops
redLineStops

#temperature range visualization
joinedData <- joinedData%>%
  mutate(year = format(as.Date(joinedData$date, format="%d/%m/%Y"),"%Y"))%>%
  mutate(month = format(as.Date(joinedData$date, format="%d/%m/%Y"),"%m"))%>%
  mutate(day = format(as.Date(joinedData$date, format="%d/%m/%Y"),"%d"))

joinedData%>%
  filter(year == 2022) %>%
  ggplot(aes(x = month, y = meanTemp)) + geom_line(aes(color = meanTemp)) +
  scale_color_gradient(low = "darkviolet", high = "red") + theme_grey() +
  labs(x = 'Months of 2022', y = 'Range of Daily Temperatures (F)',
       title = '2022 Recorded Temperature Ranges')

#ranges in ridership for a specific station
joinedData %>%
  filter(year == 2022 & station_id == 41220) %>%
  ggplot(aes(x = month, y = rides)) + geom_line(aes(color = rides)) +
  scale_color_gradient(low = 'darkolivegreen1', high = "green") + theme_dark() +
  labs(x = 'Months of 2022', y = 'Range of Daily Ridership',
       title = '2022 Fullerton Ridership Ranges per Month')

joinedData %>%
  filter(year == 2022 & month == 11 & station_id == 41220) %>%
  ggplot(aes(x = day, y = rides)) + geom_point() +
  theme_grey() +
  labs(x = 'Days of November 2022', y = 'Ridership',
       title = 'November 2022 Fullerton Ridership Daily Totals')


#write data
```

```r
write.csv(joinedData, here("joinedData.csv"))

#common legend output
#ranges in ridership for a specific station
Q1Plot <- joinedData %>%
  filter(year == 2022 & station_id == 40900) %>%
  ggplot(aes(x = month, y = rides)) + geom_line(aes(color = rides)) +
  scale_color_gradient(low = 'darkolivegreen1', high = "green") + theme_dark() +
  labs(x = 'Months of 2022', y = 'Ridership',
       title = 'Howard')

Q2Plot <- joinedData %>%
  filter(year == 2022 & station_id == 40560) %>%
  ggplot(aes(x = month, y = rides)) + geom_line(aes(color = rides)) +
  scale_color_gradient(low = 'darkolivegreen1', high = "green") + theme_dark() +
  labs(x = 'Months of 2022', y = 'Ridership',
       title = 'Jackson')

Q3Plot <- joinedData %>%
  filter(year == 2022 & station_id == 41420) %>%
  ggplot(aes(x = month, y = rides)) + geom_line(aes(color = rides)) +
  scale_color_gradient(low = 'darkolivegreen1', high = "green") + theme_dark() +
  labs(x = 'Months of 2022', y = 'Ridership',
       title = 'Addison-North Main')

Q4Plot <- joinedData %>%
  filter(year == 2022 & station_id == 41000) %>%
  ggplot(aes(x = month, y = rides)) + geom_line(aes(color = rides)) +
  scale_color_gradient(low = 'darkolivegreen1', high = "green") + theme_dark() +
  labs(x = 'Months of 2022', y = 'Ridership',
       title = 'Cermak-Chinatown')

combinedPlot <- ggarrange(Q1Plot, Q2Plot, Q3Plot, Q4Plot,
         common.legend = FALSE, legend = "right")

annotate_figure(combinedPlot, top = text_grob("Ridership Ranges for Red Line Stations 2022",
                              color = "skyblue", face = "bold", size = 14))

#HOMEWORK 4
#2d binning for numerical variable relationships, problem 1
joinedData %>%
  ggplot(aes(x = meanTemp, y = rides)) + geom_bin2d(bins = 40) + theme_bw() +
  scale_fill_continuous(type = "viridis") +
  labs(x = 'Average Daily Temperature', y = 'Single Station Daily Ridership',
        title = 'Density Plot of Daily Temperatures and Rides')

#special timeseries plots/tile plots, problem 2, during lollapalooza.
joinedData %>%
  ggplot(aes(x = (sumSnow + sumPrecip), y = rides)) +
    stat_bin_2d(bins = c(10,30)) + scale_fill_continuous(type = 'viridis') +
  labs(x = 'Daily Precipitation/Snowfall (inches)',
        y = 'Single Station Daily Ridership',
        title = 'All Precipitation to Daily Ridership')
```

5.3.4. Sachit EventHandling:

```
#sorting through events
#packages
library(ggplot2)
library(here)
library(tidyverse)
library(lubridate)
library(ggpubr)

#july 2022!
#stations of interest:
#washington/wabash, adams/wabash, jackson/state red, monroe/state red,
#jackson/dearborn blue, monroe/dearborn blue

#dataset
lollapalooza <- read.csv('lolladata.csv')
lollaStations <- c(40680, 41700, 40560, 41090, 40070, 40790)

#brown line
lollaBrown <- lollapalooza %>%
  filter(station_id == 40680 | station_id == 41700)
#remove x, station_id, station name
lollaBrown <- lollaBrown[, -c(1, 2, 3)]
#now to combine on ridership
lollaBrownRides <- lollaBrown %>%
  group_by(date) %>%
  summarise(across(rides, sum))

write.csv(lollaBrownRides, 'lollaBrownRides.csv')

#blue line
lollaBlue <- lollapalooza %>%
  filter(station_id == 40070 | station_id == 40790)
#remove x, station_id, station name
lollaBlue <- lollaBlue[, -c(1, 2, 3)]
#now to combine on ridership
lollaBlueRides <- lollaBlue %>%
  group_by(date) %>%
  summarise(across(rides, sum))

write.csv(lollaBlueRides, 'lollaBlueRides.csv')

#red line
lollaRed <- lollapalooza %>%
  filter(station_id == 40560 | station_id == 41090)
#remove x, station_id, station name
lollaRed <- lollaRed[, -c(1, 2, 3)]
#now to combine on ridership
lollaRedRides <- lollaRed %>%
  group_by(date) %>%
  summarise(across(rides, sum))

write.csv(lollaRedRides, 'lollaRedRides.csv')

#make a combined summed ridership dataset?
colnames(lollaBrownRides) <- c('date', 'Brown Line Rides')
colnames(lollaBlueRides) <- c('date', 'Blue Line Rides')
colnames(lollaRedRides) <- c('date', 'Red Line Rides')
lollaCombined <- cbind(lollaBrownRides, lollaBlueRides$`Blue Line Rides`,
                       lollaRedRides$`Red Line Rides`)
colnames(lollaCombined) <- c('date', 'brownRides', 'blueRides', 'redRides')
write.csv(lollaCombined, 'lollaCombined.csv')
```

5.3.5. Logan Ridership & Spatial Data Cleaning:

~~~
#Load Data
```{r}
daily <- read.csv(here("Project/dailyRides.csv"))
stops <- read.csv(here("Project/ctaStops.csv"))
weather <- read.csv(here("Project/weather.csv"))

#load CTA train line shape file & transform to lat/long
lineShp <- st_read(here("CTA_RailLines.shp"))%>%
  st_transform('+proj=longlat +datum=WGS84')
```

#Data Cleaning
```{r}
#convert to date var
daily$date <- as.Date(daily$date, format = "%m/%d/%Y")

#create month, day, year vars
daily%>%
  mutate(year = format(date, "%Y"))%>%
  mutate(month = format(date, "%m"))%>%
  mutate(day = format(date, "%d"))-> daily

#Filter for date range of interest
dailyFilter <- filter(daily, date >= "2022-01-01" & date <= "2023-6-30" & !stationname %in%
c("Berwyn", "Lawrence"))

#Decode dayType column
dailyFilter$daytype[dailyFilter$daytype=="W"] <- "Weekday"
dailyFilter$daytype[dailyFilter$daytype=="A"] <- "Saturday"
dailyFilter$daytype[dailyFilter$daytype=="U"] <- "Sunday/Holiday"

#cleaning weather dataset
weather <- rename(weather, date = variable)
weather$date <- substr(weather$date, 1, nchar(weather$date)-5)
weather$date <- ymd(weather$date)

#left join attempt
dailyWeatherJoin <- left_join(dailyFilter, weather, by='date')

#add in daily averages
dailyWeatherJoin%>%
  group_by(stationname)%>%
  summarise(avgDailyRides = round(mean(rides),0)) -> subDailyWeatherJoin
~~~

```r
dailyWeatherJoin <- left_join(dailyWeatherJoin,subDailyWeatherJoin)

#Create station column
stops$lineType<-qdapRegex::ex_between(stops$STATION_DESCRIPTIVE_NAME, "(",")")

#Filter for only distinct stations
stops%>%
  distinct(MAP_ID, Location, lineType,STATION_DESCRIPTIVE_NAME)%>%
  rename(station_id = MAP_ID)-> uniqueStops

#prep for join
dailyWeatherJoin$station_id <- trimws(dailyWeatherJoin$station_id, which = "both")
uniqueStops$station_id <- as.character(uniqueStops$station_id)

#Split location data into lat long coordinates
splitLocation <- stringr::str_split_fixed(uniqueStops$Location, ", ",2)
splitLocation <- as.data.frame(splitLocation)
splitLocation$V1 <- str_sub(splitLocation$V1,2)
splitLocation$V2 <- str_sub(splitLocation$V2,1,-2)

splitLocation%>%
  rename(Lat = V1)%>%
  rename(Long = V2) -> geoStop

#rejoin with train station data
geoStopJoin <- bind_cols(uniqueStops, geoStop)
geoStopJoin <- dplyr::select(geoStopJoin, -Location)

#convert lat/long to numeric
geoStopJoin$Lat <- as.numeric(geoStopJoin$Lat)
geoStopJoin$Long <- as.numeric(geoStopJoin$Long)

#join with monthly data
totalDailyJoin <- left_join(dailyWeatherJoin, geoStopJoin)

#convert list to character column
totalDailyJoin$lineType <- as.character(totalDailyJoin$lineType)

#Create a variable that labels stations with multiple lines
totalDailyJoin%>%
  mutate(genLineType = ifelse(grepl(",", lineType), "Multiple Lines",
                    ifelse(grepl("&", lineType), "Multiple Lines",
                    ifelse(grepl("-", lineType), "Blue Line",
                          lineType)))) -> totalDailyJoin

#Do the same for our shapefile
lineShp%>%
  mutate(genLineType = ifelse(grepl("ML", LEGEND), "Multiple Lines",
                    ifelse(grepl("BL", LEGEND), "Blue Line",
                    ifelse(grepl("BR", LEGEND), "Brown Line",
                    ifelse(grepl("YL", LEGEND), "Yellow Line",
                    ifelse(grepl("GR", LEGEND), "Green Line",
                    ifelse(grepl("OR", LEGEND), "Orange Line",
                    ifelse(grepl("RD", LEGEND), "Red Line",
                    ifelse(grepl("PR", LEGEND), "Purple Line",
                    ifelse(grepl("PK", LEGEND), "Pink Line",
                          "NA")))))))))) -> lineShp

#Write output
write.csv(totalDailyJoin, here("CTA_Train_Traffic/dailyDataClean.csv"))
st_write(lineShp, here::here("CTA_Train_Traffic/ctalines.shp"),
         delete_dsn = TRUE)

#save shapefile with lat lon coordinates
ctaLines <- readOGR(dsn= "/Users/loganbogenut/Documents/DePaul/Data
Visualization/CTA_Train_Traffic",
                 layer = "ctalines",
                 verbose = FALSE)
ctaLinesTransform <- spTransform(ctaLines, CRS("+proj=longlat +datum=WGS84 +no_defs"))
#Save output
raster::shapefile(ctaLinesTransform, filename = here::here("CTA_Train_Traffic", "ctalines.shp"),
overwrite=TRUE)
```
```
```

5.3.6. Logan Shiny App:

```
#Load dependencies
library(shiny)             #shiny app development
library(dplyr)             #tools for cleaning data
library(leaflet)           #map making
library(shinyWidgets)      #improved widget functionality
library(rgdal)             #load in shapefile
library(ggplot2)           #data visualization
library(rsconnect)         #publish shiny app

#load data
df <- read.csv("dailyDataClean.csv")
weatherDf <- read.csv("totalDailyJoin.csv")

lineShp <- readOGR(dsn= ".",
                layer = "ctalines",
                verbose = FALSE)

#convert to date var
df$date <- as.Date(df$date)

# Define UI for application
ui <- fluidPage(
   navbarPage("Transit Trackers",
           tabPanel("CTA Rider Distribution", icon = icon("city"),
                 mainPanel(
                    leafletOutput("mymap", height="89vh")),

                    #Specify panel attributes
                    absolutePanel(top = 85, right = 40,width = "32vw", style = "background-color:
white;
                    opacity: 0.85;
                    padding: 20px 20px 20px 20px;
                    margin: auto;
                    border-radius: 5pt;
                    box-shadow: 0pt 0pt 6pt 0px rgba(61,59,61,0.48);
                    padding-bottom: 2mm;
                    padding-top: 1mm;",

                           #add title and intro text
                           fluidRow(
                              column(12,
                                   align = "center",
                                        tags$p(strong("CTA Ridership Impacts"), style = "font-size:
3vh;")),
                                  br(),
                                  column(width = 12,
                                        align = "center",
                                        tags$div(
                                           "Use the dropdown to explore how CTA ridership changes as a
result of time and line color. This map uses
                                           data from ",
                                           tags$a(href="https://www.transitchicago.com/data/",
"transitchicago.com"),
                                           "the CTA's open source data library. Click on map markers to
expand
                                           data points and learn more!",
                                           br(), br()))),

                               #Create dropdown selection box
                               pickerInput("category", label = "Select Train Line",
                                    choices = unique(df$genLineType),
                                    selected = unique(df$genLineType),
                                    multiple = TRUE,
                                    options = pickerOptions(
                                          actionBox = TRUE,
                                          #If all train lines are selected, show 'All Train Lines'
                                          selectedTextFormat = paste0("count > ",
length(sort(unique(df$genLineType))) -1),
                                          countSelectedText = "All Train Lines")),

                               #Create dropdown filter by daytype
                               pickerInput("filterDay", label = "Filter by:",
                                         choices = unique(df$daytype),
                                         selected = unique(df$daytype),
                                         multiple = TRUE,
                                         options = pickerOptions(
                                            actionBox = TRUE,
                                            #If all train lines are selected, show 'All Train Lines'
                                            selectedTextFormat = paste0("count > ",
length(sort(unique(df$daytype))) -1),
                                            countSelectedText = "All Days Included")),

                                #add date range input
                               dateRangeInput(inputId = "date",
                                          label = "Select a Date Range:",
                                          start = "2022-01-01",
                                        end = "2023-06-30",
                                        min = "2022-01-01",
                                        max = "2023-06-30",
                                        format = "yyyy-mm-dd"),
                                plotOutput("violinPlot"))),

            #Create a Weather Data panel
            tabPanel("Ridership & Precipitation", icon = icon("cloud-sun"),
                 mainPanel(
                   leafletOutput("mymap2", height= "89vh")
                 ),

                 #Specify panel attributes
                 absolutePanel(top = 85, right = 40,width = "32vw", style = "background-color:
white;
                   opacity: 0.85;
                   padding: 20px 20px 20px 20px;
                   margin: auto;
                   border-radius: 5pt;
                   box-shadow: 0pt 0pt 6pt 0px rgba(61,59,61,0.48);
                   padding-bottom: 2mm;
                   padding-top: 1mm;",


                                #add title and intro text
                                fluidRow(
                                  column(12,
                                      align = "center",
                                      tags$p(strong("Precipitation Impacts on Ridership during
Sundays/Holidays"), style = "font-size: 3vh;")),
                                  br(),
                                  column(width = 10,
                                      align = "center",
                                      tags$div(
                                        "Use the slider to explore how CTA ridership changed as a
result of precipitation.
                                        This plot uses data from ",
                                        tags$a(href="https://www.transitchicago.com/data/",
"transitchicago.com"),
                                        "the CTA's open source data library. Also, weather data from ",
                                        tags$a(href =
"https://www.meteoblue.com/en/weather/historyclimate/weatherarchive/chicago_united-states_4
887398", "meteoblue.com"),
                                        br(), br()))),
                              #create slider input:
                              sliderTextInput("rainSlider", "Amount of Rain",
                                        choices = c(as.numeric(unique(weatherDf$sumPrecip))),
                                        selected = range(vals =
c(as.numeric(unique(weatherDf$sumPrecip))))),
                              pickerInput("cat", label = "Select Train Line",
                                      choices = unique(weatherDf$genLineType),
                                      selected = unique(weatherDf$genLineType),
                                      multiple = TRUE,
                                      options = pickerOptions(
                                        actionBox = TRUE,
                                        #If all train lines are selected, show 'All Train Lines'
                                        selectedTextFormat = paste0("count > ",
length(sort(unique(weatherDf$genLineType))) -1),
                                        countSelectedText = "All Train Lines"))
                    ))

          #Create an About section
          #tabPanel("About")
          ))


# Define server logic required to draw visuals
server <- function(input, output,server) {

  #filter weather tab by slider input
  sliderChoice = reactive({
    dfSub = weatherDf|>
      filter(sumPrecip == input$rainSlider
           & genLineType %in% input$cat)
  })

  # Reactive expression for the data subsetted to what the user selected
  filteredData <- reactive({
    dfSub <- df%>%
      filter(genLineType %in% input$category
           & between(date,input$date[1], input$date[2])
           & daytype %in% input$filterDay)%>%
      group_by(station_id, STATION_DESCRIPTIVE_NAME, Lat, Long, genLineType)%>%
      # summarise(totalRides = sum(rides), avgTemp = mean(Temperature..Mean.),
      #        maxTemp = max(Temperature..Max.), minTemp = min(Temperature..Min.),
      #        avgRain = mean(Precipitation.Total..mm.), totalSnowfall =
sum(Snowfall.Amount..cm.))
```

```
            summarise(totalRides = sum(rides), newAvgDailyRides = round(mean(rides),0),
avgDailyRides = round(mean(avgDailyRides),0))


    })

    #filter for lines shapefile based on user input
    shapeData <- reactive({
        lineShp[lineShp@data$gnLnTyp %in% input$category,]
    })

    weatherShapeData <- reactive({
        lineShp[lineShp@data$gnLnTyp %in% input$cat,]
    })

    #Create map
    output$mymap <- renderLeaflet({

        #Create a palette & corresponding legend
        pal <- colorQuantile("Reds", filteredData()$totalRides, n = 5)
        palColors <- unique(pal(sort(filteredData()$totalRides)))
        palBreaks <- round(quantile(filteredData()$totalRides, seq(0, 1, .2)),0)
        palLabels <- paste(lag(palBreaks), palBreaks, sep = " - ")[-1] # first lag is NA


        myMap <- leaflet()%>% #filters our data based on dropdown selection
        addProviderTiles(providers$CartoDB.Positron)%>% #adds a basemap
        addPolylines(data = shapeData(), color = "Gray")%>%
        addCircleMarkers(data = filteredData(),
                        ~Long, ~Lat,
                        popup = paste(
                            "<h5>Station: ", filteredData()$STATION_DESCRIPTIVE_NAME, "</h5>",
                            "<b>Selected Timeframe Daily Rider Avg: </b>",
round(filteredData()$newAvgDailyRides,2), "<br>",
                            "<b>18 Month Daily Rider Avg: </b>", round(filteredData()$avgDailyRides,2),
"<br>",
                            "<b>Selected Timeframe Total Riders: </b>", filteredData()$totalRides, "<br>"
                        ),
                        radius = 2,
                        label = ~STATION_DESCRIPTIVE_NAME,
                        color = ~pal(filteredData()$totalRides))%>%
        addLegend("bottomleft", colors = palColors, labels = palLabels, title = "Total Riders")
    })
    #define reactive value to store in modified plot
    updatedPlot <- reactiveVal(NULL)

    #define plot output
    output$violinPlot <- renderPlot({
      updatedPlot()
    })

    #update reactive value in observer
    observe({

      event <- input$mymap_marker_click
      #Check if marker was clicked
      if (!is.null(event$lng)) {
        updatedPlot(
          ggplot(data = filteredData(), aes(x = genLineType, y = totalRides/1000)) +
            geom_violin() +
            geom_point(data = filteredData()[filteredData()$Long == event$lng, ], aes(color =
  "Red", size = 5)) +
            ylab("Number of Riders by the Thousand") +
            xlab("")+
            theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1))+
            theme(legend.position="none")
        )
      } else {
        updatedPlot(
          ggplot(data = filteredData(), aes(x = genLineType, y = totalRides/1000)) +
            geom_violin() +
            ylab("Number of Riders by the Thousand") +
            xlab("")+
            theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1))+
            theme(legend.position="none")
        )
      }
    })

    output$mymap2<- renderLeaflet({
      myMap <- leaflet()%>% #filters our data based on dropdown selection
        addProviderTiles(providers$CartoDB.Positron)%>% #adds a basemap
        addPolylines(data = weatherShapeData(), color = "Gray")%>%
        addCircleMarkers(data = sliderChoice(),
                    ~Long, ~Lat,
                    radius = sliderChoice()$rides/1100,
                    label = ~STATION_DESCRIPTIVE_NAME)
    })

  }

  # Run the application
  shinyApp(ui = ui, server = server)
```

5.3.7. Abigail Scatterplot Matrix Code:

```
master <- read.csv("~/Downloads/joinedDataSample.csv")

library(ggplot2)
library(lubridate)
library(tidyquant)
library(tidyverse)

master$daytype = master$daytype %>% fct_collapse(Weekend =
c("Saturday","Sunday/Holiday"))

master = master %>%
  mutate(date = ymd(date)) %>%
  mutate_at(vars(date), funs(year, month, day))

master$month = factor(master$month,
                      levels = unique(master$month),
                      labels = c("January", "February", "March", "April", "May", "June", "July", "August",
"September", "October", "November", "December"))

master = subset(master, year == 2022)
master = subset(master, rides != 0)
joined = master %>% group_by(date, sumPrecip, daytype) %>% summarize(rides = sum(rides))
joined2 = master %>% group_by(date, sumSnow, daytype) %>% summarize(rides = sum(rides))
joined3= master %>% group_by(date, meanTemp, daytype) %>% summarize(rides =
sum(rides))


options(scipen=999)
#########

p1 = ggplot(joined, aes(sumPrecip, rides))+
  geom_point(alpha = 0.50)+
  facet_wrap(~daytype)+
  geom_smooth(method = "lm")+
  theme_bw()+
  xlab("Rain (Inches)")+
  ylab("Number of Rides")+
  ggtitle("CTA rides per day by Rain")+
  theme(plot.title = element_text(colour = "forestgreen"))

p1
```

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

p2 = ggplot(joined2, aes(sumSnow, rides))+
  geom_point(alpha = 0.50)+
  geom_smooth(method = "lm")+
  facet_wrap(~daytype)+
  theme_bw()+
  xlab("Snow (Feet)")+
  ylab("Number of Rides")+
  ggtitle("CTA rides per day by Snow")+
  theme(plot.title = element_text(colour = "blue2"))

p2



#####

p3 = ggplot(joined3, aes(meanTemp, rides))+
  geom_point(alpha = 0.50)+
  geom_smooth(method = "lm")+
  facet_wrap(~daytype)+
  theme_bw()+
  xlab("Mean Temperature (F)")+
  ylab("Number of Rides")+
  ggtitle("CTA rides per day by Temperature")+
  theme(plot.title = element_text(colour = "firebrick1"))

p3

grid.arrange(p1,p2, p3, nrow = 3)
```

5.3.8. Rohan Bertin Matrix Code:

```
df <- read.csv("C:/Users/user/Downloads/joinedDataSample.csv")
summary(df)
head(df)


library(dplyr)

# Convert 'date' to a Date object s
df$date <- as.Date(df$date)

# Create a new variable for weekdays
df$weekday <- weekdays(df$date)

# Calculate the average rides for each weekday
avg_rides <- df %>%
  filter(daytype %in% c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")) %>%
  group_by(weekday) %>%
  summarise(avg_rides = mean(rides))

# Merge the average rides back to the original dataframe
df <- left_join(df, avg_rides, by = "weekday")

# Create a contingency table for 'weekday' and 'meanTemp'
tab <- table(df$weekday, cut(df$meanTemp, breaks = 5))  # Adjust the breaks for binning

# Create a heatmap (Bertin matrix) using ggplot
ggplot(data = as.data.frame(tab), aes(x = Var1, y = Var2, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "blue") +
  labs(title = "Bertin Matrix for Day vs Mean Temperature", x = "Day", y = "Mean Temperature
Range")
```

5.4. Data Sources:

- CTA Open Database: https://www.transitchicago.com/data/
- Meteoblue Weather Archive Chicago: https://www.meteoblue.com/en/weather/historyclimate/weatherarchive/chicago_united-states_4887398