Uber Ride Analysis

Question

Given certain hourly times throughout the day and days of the week, from November 2018 through February 2020, which Chicago community areas have the most expensive fares in pick-up areas for rideshare trips?

Hypothesis

We hypothesize that Chicago O'Hare Airport provides the most expensive rideshare fares for any given hourly time of day and day of week.

Justification:

By intuition, people landing from the airport are more likely to be riders going on further trips, which in turn likely makes it an expensive pick-up area since trip duration and miles are the main factors in the fare calculation. This makes sense when considering that airports are more scarce compared to other commercial establishments, but also because airports service more than just the people living in the city where the airport is located. Furthermore, rideshare drivers themselves already pool up in airports, to the point of teaming up to artificially create surge pricing[1]. Additionally, transportation services outside of rideshare already queue in waiting areas such as along the outsides of popular hotels. Because drivers already gather in these aforementioned locations, we predict that there does indeed exist geographic hotspots at certain times.

Background Information

We chose this question because we believe that this information would be useful to drivers for determining hotspots and increasing their productivity by working near these particular locations and times in order to increase their pay. This will be useful to drivers who would like to know where the highest fare rides start, because we believe that there is a likely possibility that they would get more money from those trips. Ideally
we would like to explore where drivers should start their trips for most revenue, but there is not specific data for each trip on the other mitigating factors such as surge pricing and the way fees such as these are calculated and distributed to drivers each trip.

The price of a trip is calculated based on the estimated length and duration of a trip, on top of any unspecified service fees[2,4]. In order for the upfront pricing to stay the same for passengers despite any unprecedented changes in trip route or time, rideshare companies instead adjust how much is taken out of the total cost, sometimes even taking a loss[3]. Drivers are paid by the actual time/distance of the trip, instead of the estimated fare paid by the passenger. As such, if the trip takes longer than expected, then the company would take less out of the passenger's payment in order for the driver to receive more of the fare. The same method of paying drivers applies to surge pricing.

In a study found in the *Journal of Transport and Land Use*, the difficulty and cost of parking proves to be one of the top reasons why people would choose to use rideshare over other modes of transportation[5]. With this in mind, locations with less parking spots should result in not only more rideshare users going to those destinations, but also more users leaving those areas compared to places where parking is cheaper/accessible. The same article also lists alcohol consumption to be another leading factor for passengers to use rideshare. With such people drinking responsibly and not getting behind the wheel, we might also observe potential hotspots in places where bars and pubs are concentrated, especially late at night when people are going home from typical popular club and bar hours.

This information may also be valuable in regards to city planning, public policy, and traffic management. Furthermore, if time allows, we can also explore the density of trips leaving or entering a certain area, which may help city planners combat the congestion that might occur because of rideshare trips. Unlike someone

driving their own car, rideshare drivers don't need parking but do need a safe place to stop and wait for their passenger that would not cause congestion for the rest of the vehicles on the road.

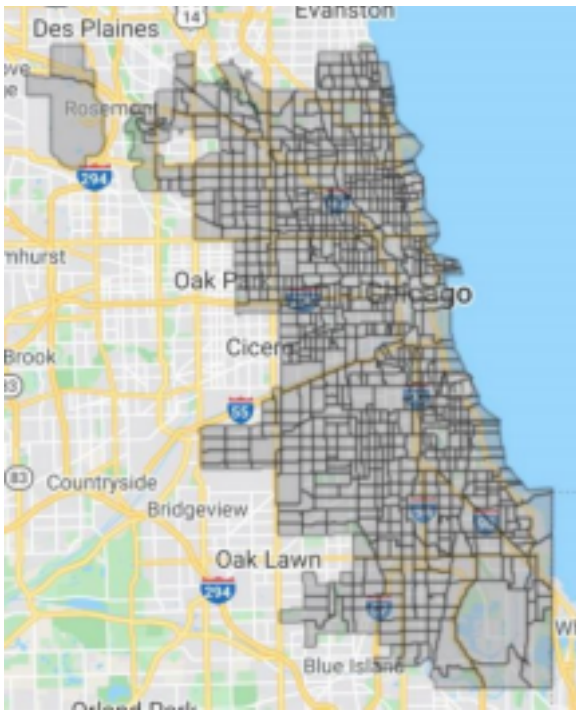
Data

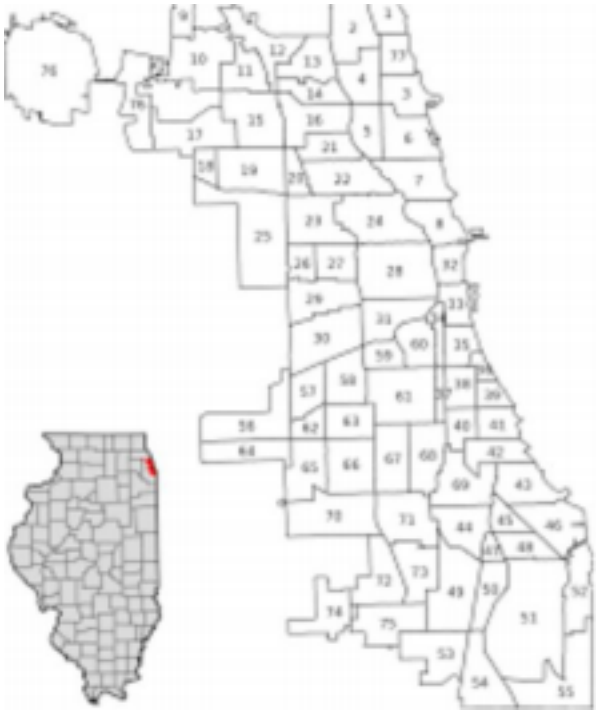
The City of Chicago has collected anonymized data from every rideshare trip within their city since November 2018 and is continuously updating the data to be up to date on a regular basis. Here is the dataset.The variables that we will use are the pickup community area, the start time of each trip, and the fare. In the extra credit section we look at the pickup area centroid, but we will not use that for our primary analysis. The pick up community area is a geographic area classified by an integer (these community areas are on average 3 square miles and are specific to Chicago). The start time is a string in ISO8601 format. The fare is a floating point number. Pickup centroid is of the form {float, float} and contains the latitude and longitude marking the center of "pickup census tract or the community area if the census tract has been hidden for privacy."

Specifically, we will be looking at trips that took place from the start of the dataset through February 2020, which is prior to the Coronavirus pandemic in order to avoid inconsistencies in the data trends due to the wide range of significant impacts from COVID-19. The entire dataset has a total of about 169 million observation entries, however after filtering the dataset to only include the observations prior to the pandemic in March 2020, the dataset will then contain about 147 million observations. The original dataset has 23 variables, but to answer this question, we will be focusing on the variables mentioned above. In particular, an approach that we are considering using to perform our analysis that helps answer our question is to group by the hours of the day and days of the week in order to find the highest frequency pick-up community areas for each hour of the day and days of the week which may be determined through data visualizations such as a line plot where we can find the peaks in the graph.

The ideal dataset would also include census tract, more accurate times, and the specific rideshare company that was offering the rideshare in order to make our question more specific, which will in turn provide us with more accurate and meaningful results and analysis. Particularly, it would have been ideal to have datasets that are separated by rideshare companies and services (such as premium vehicle options) in order to make our analysis more accurate, since there would be less chances of inaccuracies from the differences and inconsistencies in how these companies operate. For example, a driver only registered with Uber does not care where Lyft has high fares. In addition some rideshare services may provide outlier data through their services such as Uber, Black, or Lyft Lux which would greatly increase averages, skewing the data for a driver registered for regular service. In addition, the current data eliminates about a third of the census tract values in certain scenarios to protect the privacy of the data collected, but in a perfect dataset, having all the trips' census tract and exact timestamps would provide more granular location data to make our data and results more precise on which pick up community areas have the highest fare. As seen in figure 1 and 2, census tracts are smaller than community areas, thus providing more precision. The current data also rounds timestamps to the nearest 15 minutes, which also further reduces precision. Moreover, if it is possible, separating the data of rideshare companies, is also very important to take into consideration since hotspots of areas with the highest fares may vary from app to app which means users of this data may be constrained by which app they are registered with.

Census Tracts Community Areas





Source #8 Source #7

Ethical Considerations

When conducting this project to answer our data science question it is important for us to consider and keep in mind all these possible ethical concerns and their implications throughout the entirety of the process of answering this question. To begin with, a significant ethical concern regarding data collection that we will be able to address is ensuring disclosure to those whose' data is involved and informing them of the possible risks, benefits, or consequences of the study. In our project informed consent has been addressed, given that the dataset simply states that the data collected is from all rideshare providers in the city, after reviewing the privacy policy from the major providers, we conclude that every provider strictly enforces consent. Prior to collecting or distributing user data, they will request consent from the user. In addition, users have the right to opt out at any time or request the deletion of the collected data. Granted these privacy policies are from the bigger providers like Lyft, Uber, and Blacklane, it is safe to assume the remaining providers abide by a similar ethical guideline.

Additionally, this data set also addresses the concern of the privacy of individuals within the data collected, which has been addressed through the aggregation of trips by time and geographic location. This makes it so that it is not possible to determine the precise time and location of specific trips since time stamps have been aggregated into the nearest 15 minute intervals and locations are aggregated throughout about 800 census tracts throughout Chicago and community areas that give a generalized location rather than a precise location. On top of this, another level of protection added within the data collection is that any aggregated data is required to have two or fewer unique trips in the same census tract and 15 minute window which inhibits identification through the use of other more specific data sets. This results in about a third of the census tracts to be empty, however this has been compensated for through a community area variable that represents an area of the city that is on average 3 square miles.[6]

Furthermore, concerns for data storage have already been addressed due to the fact that the data set is open to the public, because of the protective steps taken in the data collection to anonymize the data. However, in regards to our analysis it is important for us to take into account ethical considerations that are reflected in factors such as ensuring clean data to avoid any errors or misinformation in our results, which can possibly occur if we do not properly clean our data by filtering out missing values that may result in inaccurate values and analysis. In addition, it is important to consider all possible perspectives to avoid any blind spots, and to eliminate biases such as cognitive bias when interpreting the data, which if not addressed, leads to unethically making analysis from personal biases, which can possibly occur if we

interpret data and form an analysis through personal beliefs or historical influences rather than the from what the data provides, which will result in ethical concerns within our model.

Having these ethical considerations addressed to ensure accurate and proper analysis is essential to the ethical considerations that we must have for our modeling which also go hand in hand with eliminating biases to make sure we are being fair across all groups by considering all possible biases, and forming our model around what the data suggests instead of own biases such as cognitive bias. Specifically, we will ensure that our project has ethical modeling since we are using variables from the data set that are understandable metrics that do not pertain to certain demographics, which eliminates the possibility of unfairness through underrepresenting minorities or vulnerable groups. In addition we are going to well document all of the steps, analysis, and variables, in order to make it clear how we are forming our results and model to others, unlike black boxes.

Lastly, on top of the other ethical concerns, the ones regarding deployment are also significant in order to ensure that our model is ready to be used for its intended purposes and has been organized in a way to prevent abuse of the model, and to keep it updated by regularly re-analyzing variable inputs and their results to ensure fairness over time. Due to the nature of our model providing helpful information to users and drivers, and by having fair, unbiased, and nondiscriminatory attributes, it is hard to see how this model can be abused or used unethically. However, a possible ethical concern may arise if sufficient checks within our analysis were not taken and there is errors or misleading information in the model we deploy which raises ethical concerns when others are making misinformed decisions based off our model that may possibly have unintended negative outcomes in possible scenarios such as in public policy, traffic management, or business decisions.

Analysis Proposal

In our data collection will be using the **API** specified here. The dataset that we chose is a Socrata open dataset where we can get the JSON of the data using the sodapy python library. We will then convert this to a pandas dataframe to process it. We will leave out the columns of the original dataset that are not relevant to our goal; we will only include the trip fare, the trip starting community area, and the date and time the trip started. We will exclude data that have missing entries for the community area, and observations that are 0 for the time traveled, or 0 for the miles traveled. Including these trips would skew and artificially reduce our average fares.

Data Cleaning

First, we will look at a **histogram** for each community area's fares. Once we have explored the data and seen its distributions, we can check for **outliers** with **box plots** or seeing if any data points are farther than ± 3 **standard deviations**. Depending on the circumstances of these observations, we can decide whether or not to take them out. For instance, trips that are unusually more expensive than trips of similar distance/time must be scrutinized properly. We could further clean our dataset by checking for erroneous data as well as those with missing values or incomplete observations. Additionally, we can use information learned from our exploratory data analysis to check if any new statistics must be calculated for further analysis and validation. For instance, we might want to check the rate of increase in the number of trips in a certain area over time. We might also want to look at relative values, such as percent increase of total fares in a specific time frame.

Once we do all of this preliminary exploratory analysis and data cleaning, we can approach our question and examine the highest fare community areas at different times of day and days of week. There are two ways that we can analyze this relationship, which vary in complexity:

Analysis Method 1

We can use **regression** to predict the fare given a community area and a time of day. Using all of the

cleaned data, we could create a regression model for each community area and day of the week. On the x-axis would be the time of day, and on the y-axis would be fare. This almost certainly will not be a linear regression, but perhaps there will be a curve that fits the data nicely that we can write a regression for. To compare between different community areas, we can compare the curve fits for each of the 77 community areas.

Analysis Method 2

We will find the **mean** fare in certain time intervals. This method is much simpler than method 1. We will split each week into weekdays and weekends, and each day into intervals such as morning, afternoon, and night. Then for each community area and time interval, we can calculate an average fare. We will also look at the median price for each community area, but we don't anticipate that this will change which community areas have the highest central tendency. The average for this time interval does not have the predicting power of the regression model, but it is much simpler to implement. We will use standard error to see overlap between fares in each community area to determine the significance of our result.

When using the mean, it is a good idea to check if the data is **normally distributed**, we want to check if the mean is a good estimate of the data in each of these intervals. We can check if the fares in each community area are normally distributed using a **Shapiro-Wilk test** or a **Chi-Square test**. The null hypothesis for both tests of normality is that the data is normally distributed. If the test proves to be statistically insignificant ($p>.05$), we fail to reject the null hypothesis and conclude that the data is normal. When we are simply comparing the means of different community areas, the chi-square test would suffice for checking normality. On the other hand, we would have to use the Shapiro-Wilk test when we begin to compare different mean fares from different hours. This is due to the fact that the chi-square test requires random sampling of data, which would not work when we involve time series in hourly fares. Checking for normality would allow us to see if any further tests that rely on normality; t-tests and others, are appropriate for our methods of analysis. Once we have confirmed that the data is normally distributed, we could take areas and/or times and check to see if their fares are statistically significantly different from other areas/hours. This can be accomplished using t-tests, especially since we would have confirmed normality in fares. Depending on whether or not two means we are comparing have equal variance, we would either use Student's t-test or Welch's t-test.

Visualizations

Visualizations are common between the two methods. At any one time in the regression model or for any time interval for method 2, we can create a **choropleth** or a **bar chart, or density dot**. This density dot and choropleth are ways we can have a color gradient indicating the average fare in each region. While this would be the most user friendly visualization, we can get more info from a bar chart. It may be hard to include all 77 community areas, but a Bar chart will allow us to view **standard error(error bars)** superimposed on the mean data. To visualize trends throughout the day, the best approach would be to use method 1 and plot the best fit curve generated by the regression model.

Analysis - Extra Credit

The extra credit portion of our project addressed our question using the techniques in the analysis proposal, but we made the following changes to our proposed analysis to work within our knowledge and our resources, given the computational limitations with processing and compute time with a data as large as ours.

- To address this limitation, we performed a convenience sample to acquire a subset of the data that we filtered to be within the time interval of the first week of the dataset. Therefore we were now able to work with a much more reasonable data set size of about 2 million trips which fits within the

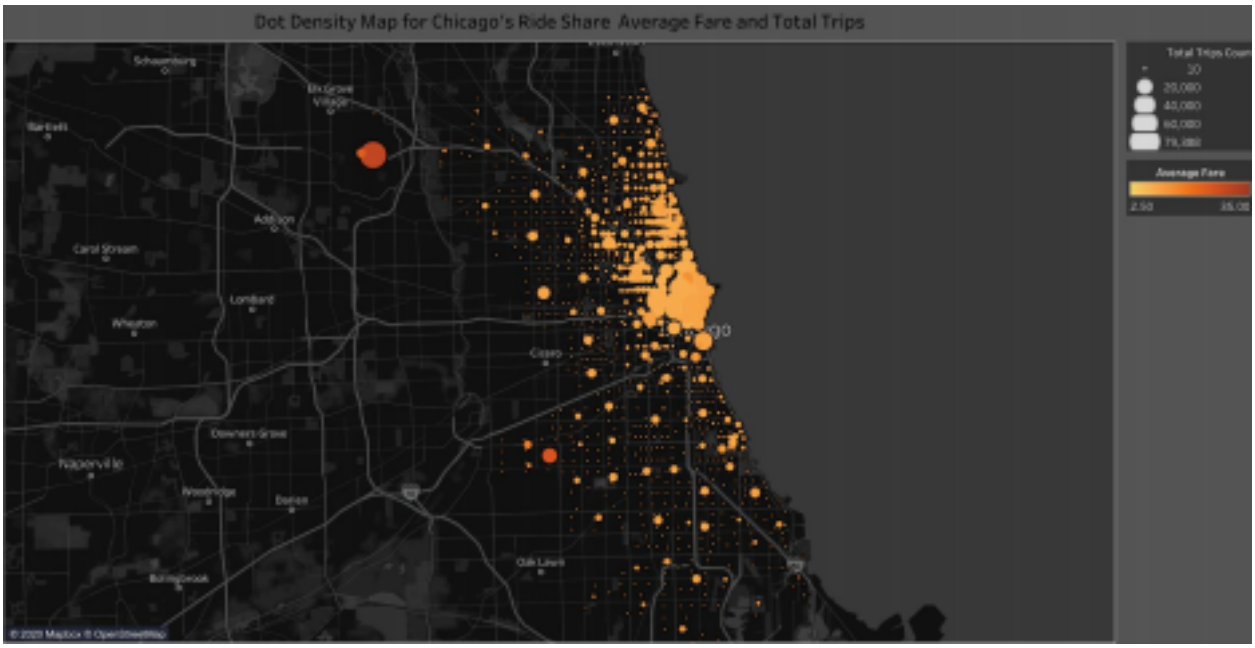
computational resources we have. Just downloading the dataset itself in CSV would have taken multiple hours, and processing since based on our calculations this would be around 100 GB in size and would require upwards of 30 GB RAM.

- We didn't filter out the fares that were ± 3 standard deviations from the mean. However we filtered out erroneous, blank, and unnecessary columns such as those for the drop off information to reduce errors and increase compute efficiency.
- Our regressions may not be as good as we want them - we did a regression with a degree 3 polynomial curve. This best fit curve didn't capture all of the features of the data - there were increases and decreases in fare that the best fit curve did not capture. (see figure 2)
- Also because of our limitations discussed above we were not able to compute all of our desired statistical tests such as checking if the data is normal with the shapiro-wilk test.

We initially approached our analysis using Jupyter notebooks however we found it a lot better in terms of our analysis to use a software such as Tableau, since we are able to provide better tailored and detailed visualizations that are also interactive. We were able to conveniently set up an app token through the city of Chicago website, and were able to efficiently use the Oauth endpoint directly with tableau, whereas requesting through the API was much harder to efficiently export and filter accordingly to our question. However, here is the git repo with our initial work. For all of the following visualizations, you can click on the links to see more clearly as it is interactive when hovering over data points. In this section, we have also included any insights that we can gain from each of these visualizations.

Figure 1 - (interactive visualization is provided in the link below)

This first graph is a dot density map of both ride frequency measured in the total number of trips out of a centroid location and average cost of fares. The size of each bubble refers to trip frequency (how many trips started within each area) and the color gradients refers to the fare. We couldn't use the ideal implementation of geographic coordinates for community areas for this visualization, since the data set did not provide variables for these ideal coordinates, therefore we instead used the provided centroid parameters given from the dataset, that are closely aligned with representing our goal of this visualization. This centroid refers to the center latitude and longitude of the pickup census tract "or the community area if the census tract has been hidden for privacy." This effort for anonymity results in missing entries and less of a generalized area which is inconsistent with the community area used in the rest of the analysis. If we were to do this professionally, we wouldn't use centroids to create the map visualization and instead we would find a solution to this by getting this information from more data or by finding map layer data that will allow us to do this.

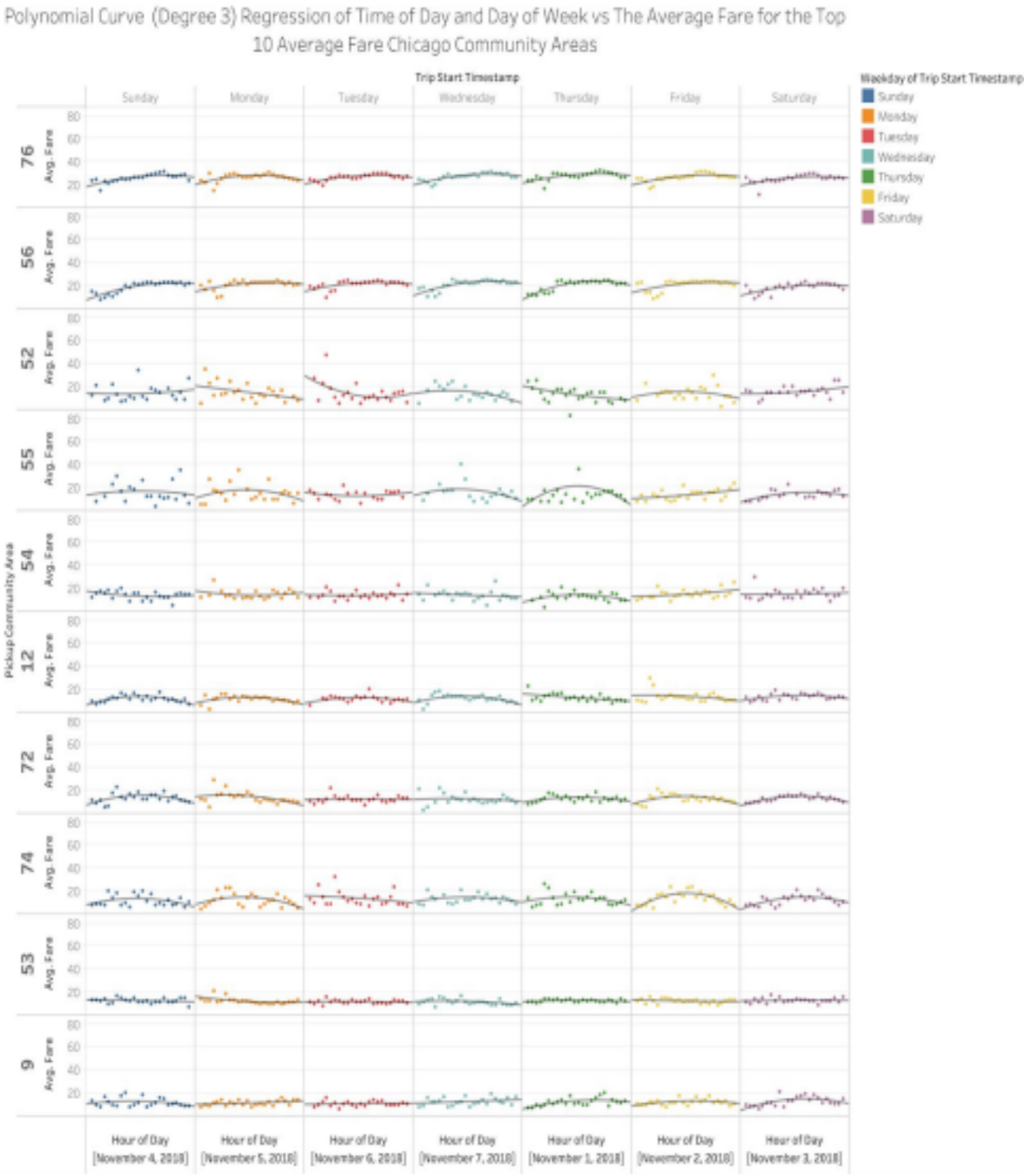


https://public.tableau.com/profile/matin1920#!/vizhome/regress/Sheet5

This visualization helps support our hypothesis, since In both this screenshot and interactive visualization above we see one of the largest circles and the darkest red bubble is located at O'Hare Airport, which shows that the highest fares are likely around O'Hare airport. In addition, we see another sizable dark red bubble around Midway airport. We see a high frequency of trips downtown, but the fares are not particularly high, since they are likely much shorter trips within the city as opposed to the airports that seem to be on the outskirts of the city.

Figure 2 - (interactive visualization is provided in the link below)

This next graph shows method 1 from the analysis proposal, for each community area and each day of the week for our one week sample from 11/01/2018. On the x-axis is the time of day for each day of the week, and on the y-axis is the average fares for each of the top 10 averages for pickup community areas .

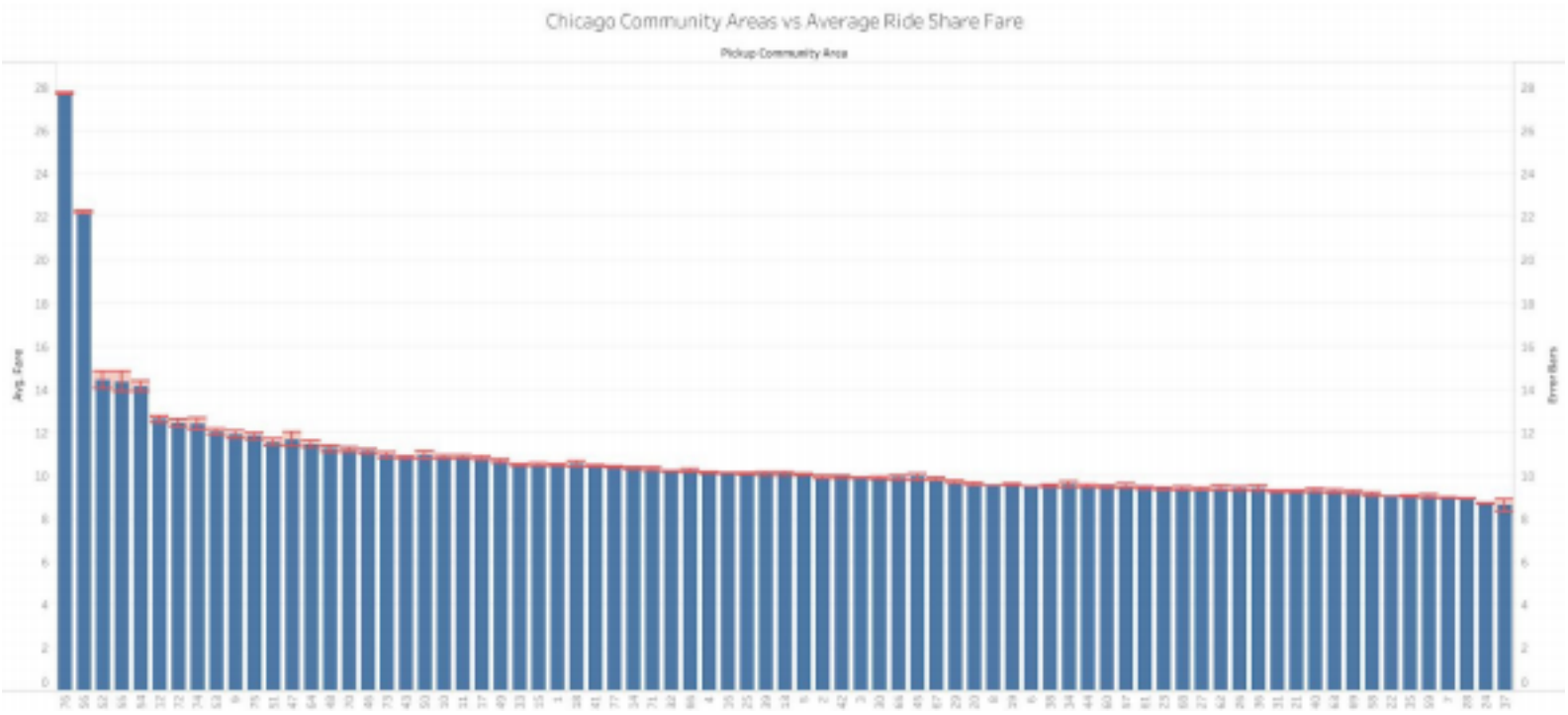


https://public.tableau.com/profile/matin1920#!/vizhome/tt_16083302787310/Sheet1?publish=yes

In this visualization above, we can see that no matter the time, community area 76, the community area for O'Hare, has higher fares than any other community area. The trends each day are different depending on the community area and day of the week. It's difficult to make claims using this regression because it was made using few points and a degree 3 polynomial fit likely isn't the best function to describe trends. In general, we see that 76 and 56, O'Hare and Midway airport respectively, have less variance in fares as the points fit nicely to the curve. This may be because rideshare companies regulate the fares in these areas. Although we can't make to many observations from this smaller dataset, O'Hare airport consistently has low fares around 3-4AM, and high fares around 7AM and 4PM.

Figure 3 - (interactive visualization is provided in the link below)

The next graph we have is a bar graph of the average fare for each community area. We didn't split into time intervals like in method 2, but we still wanted to know which areas were the most expensive. We included error bars of ± 1 standard error, to see how much fares overlapped between community areas, to gauge the significance of our results



https://public.tableau.com/profile/matin1920#!/vizhome/areaVSavgFare/Sheet8?publish=yes

In this graph, we see that on average, O'Hare and Midway have the highest fares. We know that these fares are significantly higher than the rest of the fares as there is no overlap in the error bars.

FIgure 4 - (interactive visualization is provided in the link below)

https://public.tableau.com/profile/matin1920#!/vizhome/areaVSavgFare2/Sheet2

Although this visualization is quite crowded and busy with colors and information, I believe this visualization is meaningful in the sense that it captures the entire time frame of our sample with all the variables that we are interested in. This is because, on the x-axis we are representing each hour of each day and the y-axis measures the total average fares, and with the use of an area line plot we can group the averages of each community area and find the largest average for a community area by finding the largest area and matching it with the color representing these particular community areas. Furthermore, this visualization allows to roughly see the total amount of average fares for each community area given any day and hour on the x-axis.

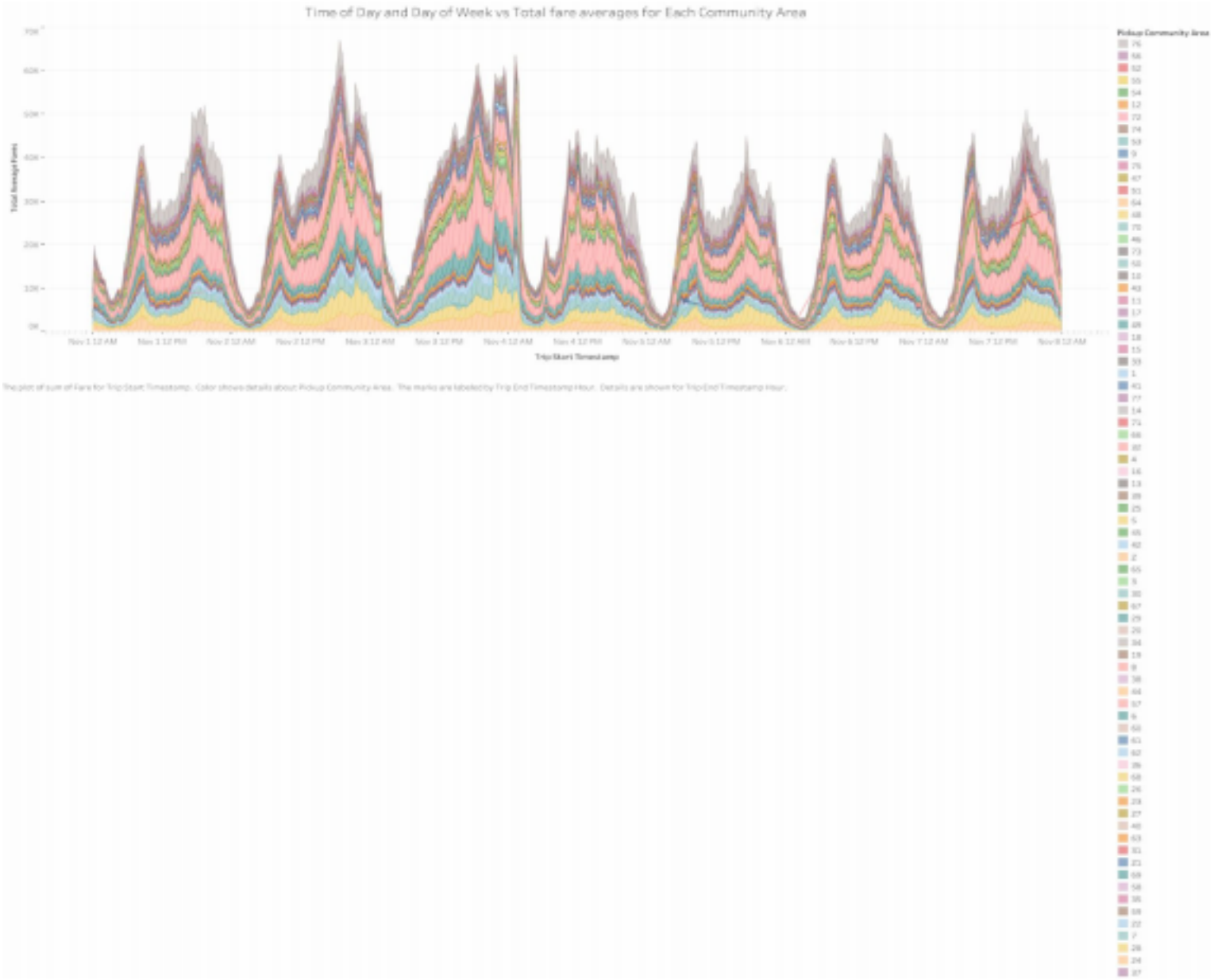
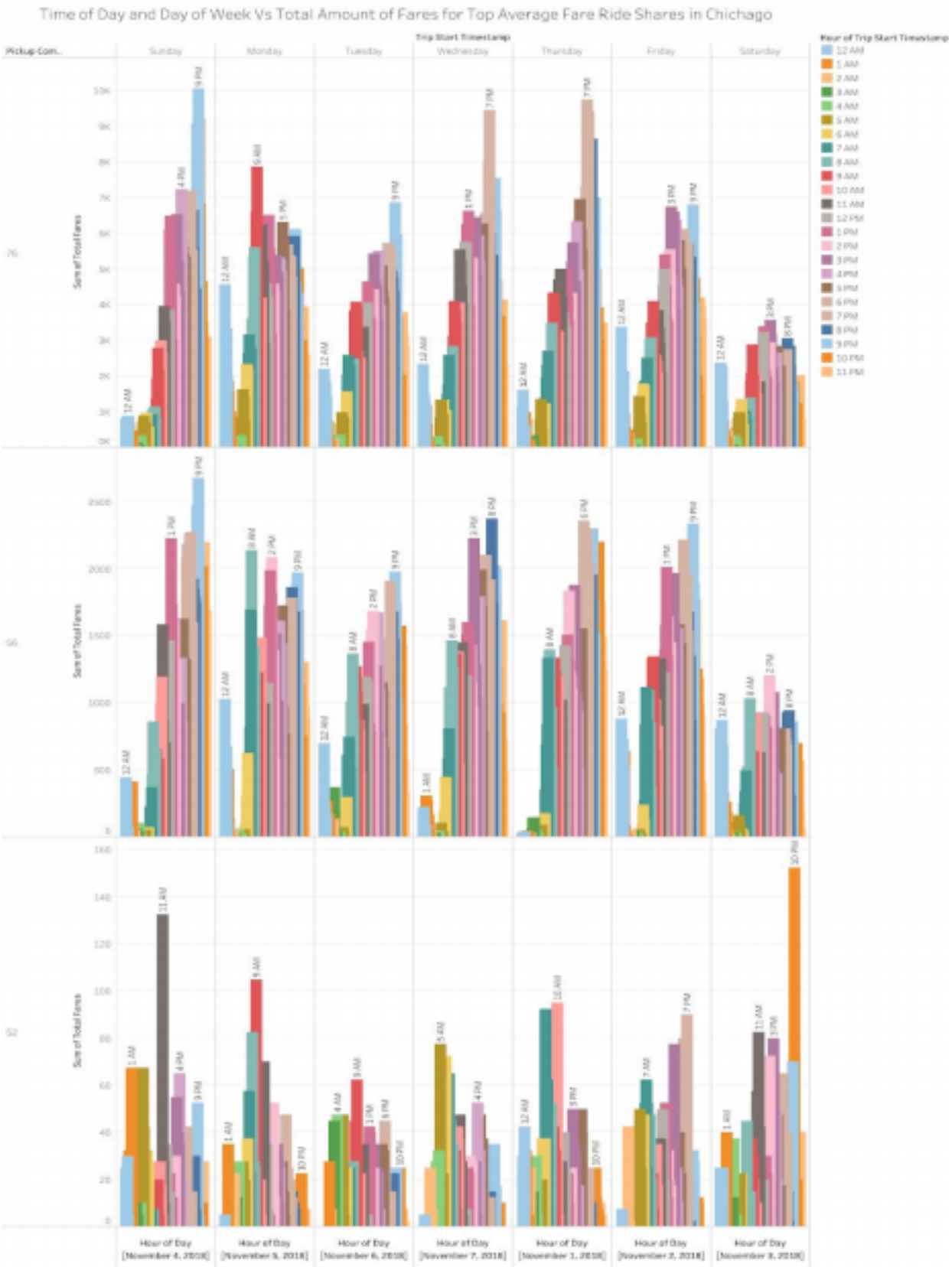


FIgure 5 - (interactive visualization is provided in the link below)



https://public.tableau.com/profile/matin1920#!/vizhome/areaVSavgFare24/Sheet33?publish=yes

This visualization above is also quite crowded and busy with colors and information however despite this, this is another visualization that I believe is useful in trying to help answer our question. This is because this visualization shows us the highest average total fares for the top 3 fare averages for community areas. As a result we can see the highest total average community area is O'Hare airport and the second highest is Midway airport which can be used to help answer our hypothesis as well as our question, because the overlaid bars and time

labels helps us determine all of the total averages for each time period which may help us answer what time and day of week is most popular for the highest averaging community area fares. And when answering this question based off of this plot we can see that within the top community areas of these 2 airports, that the plot suggests Sunday 9 PM is the most popular, followed by Thursday and Wednesday around 6-8 PM.

Discussion

Interpreting Results of our proposed analysis :

Once we generate the aforementioned visualizations, we can understand which pick up locations are the most expensive. Perhaps the easiest way that we can understand the effects of both time and community area on price is by looking at the curve that best fits generated by the regression. We can put multiple curve fits on the same graph, seeing which areas are the most expensive at any given time. However, this does depend on how "good" of a prediction the curve fit is; if the R^2 value of the curve fit is very low, we know that the curve is not a good estimate of the data, since this indicates there is to much variance amongst the data to have a proper fitting curve. In that case, it would be better to look at the averages in each time interval (method 2). We can gain a lot from bar graphs of mean fare in each community area for each time interval. Error bars will help us understand how drastic these differences in mean are. For a more comprehensive analysis of how different the means are, we can use t-tests (see analysis proposal). Please refer to the extra credit section to see how we interpret the results from our smaller analysis with our convenience sample of almost 2 million observations which is capturing the first entire week of November 2018 .

Limitations, pitfalls, and potential confounds of our methods, or biases in our data sources: Throughout this project, we had many questions. We decided to analyze this dataset with the goal of finding the most profitable places to drive, so that drivers can optimize where they wait for passengers. Does the research question we posed actually meet the goal of informing drivers where they can make the most money? Although our method could locate the community area that generates the highest average fare for drivers to go to, it does not necessarily follow that this area would be the most profitable place for drivers to camp at. For example, an area could have a high average fare, but it could also generate the longest trips that drivers end up going on. This could potentially result in a lower overall profit because the driver might end up having to drive all the way to somewhere with hardly any trips, making them take the time to have to drive somewhere more populated. A possible solution is to analyze the data for fare/tip per unit time, since the dataset also provides the duration of the trip. Even this, however, can't completely achieve the goal. We have no evidence to say that fare is directly correlated to the revenue that the driver makes. The perfect dataset for answering this new question would allow us to calculate how much revenue drivers make for each trip. Additionally, we weren't able to find information on how rideshare companies calculate the surge price rates as well as the exact expanded formula used by rideshare companies to determine the proportion of the fare that drivers receive as revenues for each trip since the current data provides only a very generalized formula. However, if the data included all the needed specific variables that are factors in the fare calculation such as surge pricing, this would have allowed us to develop a meaningful analysis based on specific community areas that gave the highest revenue per hour of the day and days of the week, which would help drivers to maximize their profits. While this dataset is highly informative, we are unable to determine how much a driver earns from their drives, in addition, there are additional factors to see if these high fare rides are even worth it, such as the amount of time spent idling or the geographical tolls on fuel.

One possible limitation in our dataset is that there are almost 169M observations. With such a large amount of data, there would either need to be a large amount of processing power or time. We could simplify this by

taking a smaller convenience sample to get a more workable subset of the data for our resources. In our extra credit work, we learned that there is no easy way to download only a random sample. While the API does let you use SoQL queries to filter only columns and observations that meet certain criteria, there isn't a great way to download a random sample from the dataset. Downloading the entire dataset can take hours and takes up significant RAM. In addition, we noticed limitations of traditional practices of reading in a local CSV file since the data exported has untidy data which may be seen through the unconventional formatting of the dates in addition to spaces within variable names, and missing and erroneous data points. We addressed this by in Initially in our analysis with Juptyter and Pandas by using the datetime module in order to pull meaning out of the untidy data that is given, in addition to filtering out values for the distance, time traveled, and community that were equal or missing to zero or blank.

Due to the nature of the data we used, there are likely many confounding variables that must be considered while using our results. While none of these confounds impact our analysis or the answer to our research question, these can impact people that use our analysis. The geolocation of each of the community areas can act as a confounding variable, as it can complicate the cause and effect relationship between community area and fare. For example, community area 76 is far from the rest of the areas, and as a result trips would be longer, and therefore fares are higher. For example someone may explore the relationship between property values and Uber fares. Not taking into account the confounding variable of location can cause biases in this analysis - looking at an expensive neighborhood on the outskirts of the city can cause a perceived correlation between property values and Uber fares, when in reality the two variables aren't related.

Addressing Ethical Concerns and Societal Implications:

Due to the nature of our question being used to provide helpful information and insights, it is hard to see how our analysis model may be abused and have possible negative societal implications. However, a possible ethical concern may arise if sufficient checks within our analysis are not taken and there are errors or misleading information. Our work can raise ethical concerns when others are making misinformed decisions based on our model. This may possibly have unseen and unintended negative outcomes in possible scenarios such as when one is using our analysis as insight in important decisions that could be in public policy, traffic management, or business decisions. Misinformation from our analysis may be a consequence if we were to take a sample that was possibly unfairly representative or too small in regards to the entire population which could lead to having bias towards certain groups, which in our case is likely within our community areas, or if measurement variables are bias with uneven distribution of data points which may likely overrepresent some areas while under presenting others, which results in significant effects from outliers or accidental missing data point. Therefore, for this reason we can mitigate and solve these issues with statistical tests, such as a Shapiro-Wilk to check if the data is normally distributed and along with other statistical tests to evaluate the significance of our results to further increase our accuracy and limit our uncertainty.

Group Participation

Matin: In this project I wrote about the ethical considerations section and the societal implications/ethical consideration segment of the discussion section, as well as the discussion section regarding our limitations, possible pitfalls, and biases within our data source. Furthermore, I also collaborated with Miguel and Daniel on forming and refining our question, hypothesis, data, and background information, in addition to all of us getting together to conduct research, form a question, and refine it. Also I contributed along with my teammates to formulate our analysis proposal since we together researched, discussed, and wrote the

methods, visualizations, and statistics that we together reasoned and found to be the best. After this I conducted the analysis section of this project where I authenticated myself with the City of Chicago website to get an API key to request the data, where I then produced an analysis with Jupyter and pandas as well as the visualizations including those above with Tableau.

Daniel: I wrote the original data section in assignment 1 that was edited to what you see now. I added more info about the particular data types that we were using for the final project. For the analysis proposal, I wrote the first paragraph, Method 1, the first paragraph from method 2, and the visualization section. For the extra credit, I made src/import_data.ipynb and src/compute_averages.ipynb (see the git repo). This was made to efficiently import from the API, store the dataset locally, and compute the averages per community area. This part was really fun for me as this was my first time using Pandas and I had to learn a little bit of SQL. I didn't focus on creating visualizations with my code in order to focus on the written part of the assignment and Matin was already using Tableau to create more detailed and interactive visualizations. For this reason, my notebooks are not included in this report, but are still on the git repository. For the written part of the EC, I wrote the captions and info about the figures with Matin. I also wrote chunks of discussion (most of the first paragraph), and helped combine many people's work for the discussion.

Miguel: I worked on taking the original question and narrowing it down to the professors/TAs standards. After Matin brought it up to the staff, I simplified it according to their suggestions. Furthermore, I wrote a portion of our hypothesis' justifications and made sure it aligned with our ideas in the rest of the proposal. I also came up with the main ideas for data cleaning and statistical methods of analysis involving tests for normality and usage of t-tests. This includes explaining why these tests are appropriate and their purposes.

Shasta: I primarily focussed on making final edits on the assignments, particularly changing and editing contents in the data, background information, and hypothesis sections. In the final project, I continued to make edits and helped filter out information and content across the assignments. I also edited and formatted images to finalize the document.

Ben: I researched how the big rideshare companies like Lyft, Uber, and Blacklane collected their data and cross-referenced each source on their privacy policy to see if it aligns with the ethical considerations. For the final project, I filtered out any unnecessary topics from A1 that did not pertain to our original question and reformatted it. Also worked on how the intervals should be set up for the whole week then for each day (i.e weekend, morning, night, etc).

Bibliography

1. https://www.inc.com/minda-zetlin/uber-lyft-drivers-artificial-surge-pricing-reagan-national-washington-arlington-drive-united.html

2. https://marketplace.uber.com/pricing

3. https://marketplace.uber.com/pricing/service-fee

4. https://help.lyft.com/hc/en-us/articles/115013080008-How-and-when-driver-pay-is-calculated

5. https://www.jstor.org/stable/26911261

6. dev.cityofchicago.org/open%20data/data%20portal/2019/04/12/tnp-taxi-privacy.html

7. https://www.lyft.com/privacy

8. https://www.uber.com/legal/pt-br/document/?country=united-states&lang=en&name=privacy-notic e

9. https://www.blacklane.com/en/privacy-policy/

df

December 18, 2020

[1]:
```
import babypandas as bpd
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import numpy as np
import datetime

#https://data.cityofchicago.org/resource/8krh-nhfb.json
#questions = bpd.read_csv('nov18week1clean.csv')\
```

1 Cogs 9 final project extra credit.

1.1 Prior to using Tableau and the City of Chicago's useful filtering options, we were using an API Key and authentication to request the url of the JSON data Set here in this notebook.

[2]:
```
import pandas as pd
from sodapy import Socrata

# Unauthenticated client only works with public data sets. Note 'None' # in place of
application token, and no username or password:
#client = Socrata("data.cityofchicago.org", None)

#Example authenticated client (needed for non-public datasets):
client = Socrata("data.cityofchicago.org",
                 "CUjOWfE57JOYcOIzZBS1LV9wZ",
                 username="nims104@gmail.com",
                 password="Cogs9!!9")

# First 2000 results, returned as JSON from API / converted to Python list of # dictionaries
by sodapy.
results = client.get("8krh-nhfb", limit=1936828)

# Convert to pandas DataFrame
results_df = pd.DataFrame.from_records(results)
```

2 Here is the data set after getting only the first week of November 2018 as our convience sample to improve efficiency in computa tion proccesses

[3]: results_df

[3]: trip_id trip_start_timestamp \ 0 971beb1419f688efeff76a37f7bbce0db1027c11 2018-11-07T23:45:00.000 1 96fdd5737ff418f493ba58aceb678481e41cc34c 2018-11-07T23:45:00.000 2 96c87ea59efebb6a4541619a6ba2f1abd1006d8b 2018-11-07T23:45:00.000 3 969de69a352f0f2ea1766fbee1f2b675dfd5ff84 2018-11-07T23:45:00.000 4 9658c724217d0298e1f6c286de46922b6c044192 2018-11-07T23:45:00.000 … … … 1936823 d82a3591bd391eb5679cbcd69c6f59677488cb2f 2018-11-01T00:00:00.000 1936824 d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 2018-11-01T00:00:00.000 1936825 d7867b9d481d3f3d8225badf3be655a1f56de496 2018-11-01T00:00:00.000 1936826 d70e65b5142ee7cbddc009b4939087afda184aac 2018-11-01T00:00:00.000 1936827 d54ca075925cb437317a5f89df300f56efe8d1fb 2018-11-01T00:00:00.000

trip_end_timestamp trip_seconds trip_miles \
0 2018-11-07T23:45:00.000 771 5.23118898528
1 2018-11-08T00:00:00.000 726 2.81755890464
2 2018-11-08T00:00:00.000 519 2.20618176654288
3 2018-11-07T23:45:00.000 256 1.08501496416
4 2018-11-08T00:00:00.000 1066 11.79099308704
… … … …
1936823 2018-11-01T00:30:00.000 1100 8.57323393216
1936824 2018-11-01T00:45:00.000 2158 9.88755618256
1936825 2018-11-01T00:15:00.000 459 2.03847066970998
1936826 2018-11-01T00:15:00.000 800 2.49130870064
1936827 2018-11-01T00:15:00.000 479 4.8906459152

pickup_community_area fare tip additional_charges trip_total \
0 32 10 2 2.5 14.5 1 28 5 0 0 5 2 3 5 0 2.5 7.5 3 6 5 0 0 5 4 8 17.5 0 2.5 20 … …
… .. … …
1936823 31 12.5 1 2.5 16 1936824 24 17.5 0 0 17.5 1936825 28 5 0 2.5 7.5
1936826 8 7.5 0 0 7.5 1936827 5 7.5 0 0 7.5

pickup_centroid_latitude pickup_centroid_longitude \

2
0 41.8774061234 -87.6219716519
1 41.8786674201 -87.6716536214
2 41.96581197 -87.6558787862
3 41.9442266014 -87.6559981815
4 41.899602111 -87.6333080367
… … …
1936823 41.8502663663 -87.667569312
1936824 41.9012069941 -87.6763559892
1936825 41.874005383 -87.6635175498
1936826 41.8991556134 -87.6262105324

1936827 41.9477915865 -87.6838349425

pickup_centroid_location

0 {'type': 'Point', 'coordinates': [-87.62197165…
1 {'type': 'Point', 'coordinates': [-87.67165362…
2 {'type': 'Point', 'coordinates': [-87.65587878…
3 {'type': 'Point', 'coordinates': [-87.65599818…
4 {'type': 'Point', 'coordinates': [-87.63330803…
… …
1936823 {'type': 'Point', 'coordinates': [-87.66756931…
1936824 {'type': 'Point', 'coordinates': [-87.67635598…
1936825 {'type': 'Point', 'coordinates': [-87.66351754…
1936826 {'type': 'Point', 'coordinates': [-87.62621053…
1936827 {'type': 'Point', 'coordinates': [-87.68383494…

[1936828 rows x 13 columns]

[4]: datetime.datetime.strptime('2020-11-04T23:59:00.0000', '%Y-%m-%dT%H:%M:%S.0000') [4]:

datetime.datetime(2020, 11, 4, 23, 59)

3 We are using the Trip_ID string unique identifier as our indicies instead of the default values to best abide by Tidy Data and to make Quering process more robust

[5]: trips_with_index = results_df.set_index('trip_id')
trips_with_index

[5]: trip_start_timestamp \ trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 2018-11-07T23:45:00.000
96fdd5737ff418f493ba58aceb678481e41cc34c 2018-11-07T23:45:00.000
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 2018-11-07T23:45:00.000
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 2018-11-07T23:45:00.000

3
9658c724217d0298e1f6c286de46922b6c044192 2018-11-07T23:45:00.000
… … d82a3591bd391eb5679cbcd69c6f59677488cb2f
2018-11-01T00:00:00.000 d7c725a5528f5435c7cd4c4b2348f0e72544e3c2
2018-11-01T00:00:00.000 d7867b9d481d3f3d8225badf3be655a1f56de496
2018-11-01T00:00:00.000 d70e65b5142ee7cbddc009b4939087afda184aac
2018-11-01T00:00:00.000 d54ca075925cb437317a5f89df300f56efe8d1fb
2018-11-01T00:00:00.000

trip_end_timestamp \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 2018-11-07T23:45:00.000
96fdd5737ff418f493ba58aceb678481e41cc34c 2018-11-08T00:00:00.000
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 2018-11-08T00:00:00.000

969de69a352f0f2ea1766fbee1f2b675dfd5ff84 2018-11-07T23:45:00.000
9658c724217d0298e1f6c286de46922b6c044192 2018-11-08T00:00:00.000
… …
d82a3591bd391eb5679cbcd69c6f59677488cb2f 2018-11-01T00:30:00.000
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 2018-11-01T00:45:00.000
d7867b9d481d3f3d8225badf3be655a1f56de496 2018-11-01T00:15:00.000
d70e65b5142ee7cbddc009b4939087afda184aac 2018-11-01T00:15:00.000
d54ca075925cb437317a5f89df300f56efe8d1fb 2018-11-01T00:15:00.000

trip_seconds trip_miles \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 771 5.23118898528
96fdd5737ff418f493ba58aceb678481e41cc34c 726 2.81755890464
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 519 2.20618176654288
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 256 1.08501496416
9658c724217d0298e1f6c286de46922b6c044192 1066 11.79099308704 … … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f 1100 8.57323393216
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 2158 9.88755618256
d7867b9d481d3f3d8225badf3be655a1f56de496 459 2.03847066970998
d70e65b5142ee7cbddc009b4939087afda184aac 800 2.49130870064
d54ca075925cb437317a5f89df300f56efe8d1fb 479 4.8906459152

pickup_community_area fare tip \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 32 10 2
96fdd5737ff418f493ba58aceb678481e41cc34c 28 5 0
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 3 5 0
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 6 5 0
9658c724217d0298e1f6c286de46922b6c044192 8 17.5 0 … … … ..
d82a3591bd391eb5679cbcd69c6f59677488cb2f 31 12.5 1
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 24 17.5 0
d7867b9d481d3f3d8225badf3be655a1f56de496 28 5 0

4
d70e65b5142ee7cbddc009b4939087afda184aac 8 7.5 0
d54ca075925cb437317a5f89df300f56efe8d1fb 5 7.5 0

additional_charges trip_total \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 2.5 14.5
96fdd5737ff418f493ba58aceb678481e41cc34c 0 5
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 2.5 7.5
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 0 5
9658c724217d0298e1f6c286de46922b6c044192 2.5 20 … … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f 2.5 16
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 0 17.5
d7867b9d481d3f3d8225badf3be655a1f56de496 2.5 7.5
d70e65b5142ee7cbddc009b4939087afda184aac 0 7.5
d54ca075925cb437317a5f89df300f56efe8d1fb 0 7.5

pickup_centroid_latitude \
trip_id

971beb1419f688efeff76a37f7bbce0db1027c11 41.8774061234
96fdd5737ff418f493ba58aceb678481e41cc34c 41.8786674201
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 41.96581197
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 41.9442266014
9658c724217d0298e1f6c286de46922b6c044192 41.899602111 … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f 41.8502663663
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 41.9012069941
d7867b9d481d3f3d8225badf3be655a1f56de496 41.874005383
d70e65b5142ee7cbddc009b4939087afda184aac 41.8991556134
d54ca075925cb437317a5f89df300f56efe8d1fb 41.9477915865

pickup_centroid_longitude \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 -87.6219716519
96fdd5737ff418f493ba58aceb678481e41cc34c -87.6716536214
96c87ea59efebb6a4541619a6ba2f1abd1006d8b -87.6558787862
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 -87.6559981815
9658c724217d0298e1f6c286de46922b6c044192 -87.6333080367 … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f -87.667569312
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 -87.6763559892
d7867b9d481d3f3d8225badf3be655a1f56de496 -87.6635175498
d70e65b5142ee7cbddc009b4939087afda184aac -87.6262105324
d54ca075925cb437317a5f89df300f56efe8d1fb -87.6838349425

pickup_centroid_location
trip_id

971beb1419f688efeff76a37f7bbce0db1027c11 {'type': 'Point', 'coordinates':
[-87.62197165…
96fdd5737ff418f493ba58aceb678481e41cc34c {'type': 'Point', 'coordinates':
[-87.67165362…
96c87ea59efebb6a4541619a6ba2f1abd1006d8b {'type': 'Point', 'coordinates':
[-87.65587878…
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 {'type': 'Point', 'coordinates':
[-87.65599818…
9658c724217d0298e1f6c286de46922b6c044192 {'type': 'Point', 'coordinates':
[-87.63330803…
…
…
d82a3591bd391eb5679cbcd69c6f59677488cb2f {'type': 'Point', 'coordinates':
[-87.66756931…
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 {'type': 'Point', 'coordinates':
[-87.67635598…
d7867b9d481d3f3d8225badf3be655a1f56de496 {'type': 'Point', 'coordinates':
[-87.66351754…
d70e65b5142ee7cbddc009b4939087afda184aac {'type': 'Point', 'coordinates':
[-87.62621053…
d54ca075925cb437317a5f89df300f56efe8d1fb {'type': 'Point', 'coordinates':
[-87.68383494…

[1936828 rows x 12 columns]

4 As we can see above the formatting of the timestamps is against tidy data standards therefore, I use the datetime module to extract the number of the month, year, day, day of week, and hour and put them into their own columns so we can use them for our analysis

```
[6]: f = '%Y-%m-%dT%H:%M:%S.000'
     trips_with_start_times = trips_with_index.assign(start_year=trips_with_index.
      ,→get('trip_start_timestamp').apply(lambda x: datetime.datetime.strptime(x, f). ,→year),
                                                          start_month=trips_with_index.
      ,→get('trip_start_timestamp').apply(lambda x: datetime.datetime.strptime(x, f). ,→month),
                                                           start_day=trips_with_index.
      ,→get('trip_start_timestamp').apply(lambda x: datetime.datetime.strptime(x, f). ,→day),
```



```
                                                          start_weekday=trips_with_index.
      ,→get('trip_start_timestamp').apply(lambda x: datetime.datetime.strptime(x, f). ,→weekday()),
                                                           start_hour=trips_with_index.
      ,→get('trip_start_timestamp').apply(lambda x: datetime.datetime.strptime(x, f). ,→hour)
                                                                )
     trips_with_start_times

[6]: trip_start_timestamp \ trip_id
     971beb1419f688efeff76a37f7bbce0db1027c11      2018-11-07T23:45:00.000
     96fdd5737ff418f493ba58aceb678481e41cc34c      2018-11-07T23:45:00.000
     96c87ea59efebb6a4541619a6ba2f1abd1006d8b      2018-11-07T23:45:00.000
     969de69a352f0f2ea1766fbee1f2b675dfd5ff84      2018-11-07T23:45:00.000
     9658c724217d0298e1f6c286de46922b6c044192      2018-11-07T23:45:00.000
     … …
     d82a3591bd391eb5679cbcd69c6f59677488cb2f      2018-11-01T00:00:00.000
     d7c725a5528f5435c7cd4c4b2348f0e72544e3c2      2018-11-01T00:00:00.000
     d7867b9d481d3f3d8225badf3be655a1f56de496      2018-11-01T00:00:00.000
     d70e65b5142ee7cbddc009b4939087afda184aac      2018-11-01T00:00:00.000
     d54ca075925cb437317a5f89df300f56efe8d1fb 2018-11-01T00:00:00.000

                                                                trip_end_timestamp \
     trip_id
     971beb1419f688efeff76a37f7bbce0db1027c11      2018-11-07T23:45:00.000
     96fdd5737ff418f493ba58aceb678481e41cc34c      2018-11-08T00:00:00.000
     96c87ea59efebb6a4541619a6ba2f1abd1006d8b      2018-11-08T00:00:00.000
     969de69a352f0f2ea1766fbee1f2b675dfd5ff84      2018-11-07T23:45:00.000
     9658c724217d0298e1f6c286de46922b6c044192      2018-11-08T00:00:00.000
     … …
     d82a3591bd391eb5679cbcd69c6f59677488cb2f      2018-11-01T00:30:00.000
     d7c725a5528f5435c7cd4c4b2348f0e72544e3c2      2018-11-01T00:45:00.000
```

d7867b9d481d3f3d8225badf3be655a1f56de496 2018-11-01T00:15:00.000
d70e65b5142ee7cbddc009b4939087afda184aac 2018-11-01T00:15:00.000
d54ca075925cb437317a5f89df300f56efe8d1fb 2018-11-01T00:15:00.000

trip_seconds trip_miles \

trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 771 5.23118898528
96fdd5737ff418f493ba58aceb678481e41cc34c 726 2.81755890464
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 519 2.20618176654288
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 256 1.08501496416
9658c724217d0298e1f6c286de46922b6c044192 1066 11.79099308704 … … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f 1100 8.57323393216
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 2158 9.88755618256

7
d7867b9d481d3f3d8225badf3be655a1f56de496 459 2.03847066970998
d70e65b5142ee7cbddc009b4939087afda184aac 800 2.49130870064
d54ca075925cb437317a5f89df300f56efe8d1fb 479 4.8906459152

pickup_community_area fare tip \

trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 32 10 2
96fdd5737ff418f493ba58aceb678481e41cc34c 28 5 0
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 3 5 0
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 6 5 0
9658c724217d0298e1f6c286de46922b6c044192 8 17.5 0 … … … ..
d82a3591bd391eb5679cbcd69c6f59677488cb2f 31 12.5 1
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 24 17.5 0
d7867b9d481d3f3d8225badf3be655a1f56de496 28 5 0
d70e65b5142ee7cbddc009b4939087afda184aac 8 7.5 0
d54ca075925cb437317a5f89df300f56efe8d1fb 5 7.5 0

additional_charges trip_total \

trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 2.5 14.5
96fdd5737ff418f493ba58aceb678481e41cc34c 0 5
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 2.5 7.5
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 0 5
9658c724217d0298e1f6c286de46922b6c044192 2.5 20 … … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f 2.5 16
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 0 17.5
d7867b9d481d3f3d8225badf3be655a1f56de496 2.5 7.5
d70e65b5142ee7cbddc009b4939087afda184aac 0 7.5
d54ca075925cb437317a5f89df300f56efe8d1fb 0 7.5

pickup_centroid_latitude \

trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 41.8774061234
96fdd5737ff418f493ba58aceb678481e41cc34c 41.8786674201
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 41.96581197
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 41.9442266014
9658c724217d0298e1f6c286de46922b6c044192 41.899602111 … …

d82a3591bd391eb5679cbcd69c6f59677488cb2f 41.8502663663
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 41.9012069941
d7867b9d481d3f3d8225badf3be655a1f56de496 41.874005383
d70e65b5142ee7cbddc009b4939087afda184aac 41.8991556134
d54ca075925cb437317a5f89df300f56efe8d1fb 41.9477915865

pickup_centroid_longitude \

8

trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 -87.6219716519
96fdd5737ff418f493ba58aceb678481e41cc34c -87.6716536214
96c87ea59efebb6a4541619a6ba2f1abd1006d8b -87.6558787862
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 -87.6559981815
9658c724217d0298e1f6c286de46922b6c044192 -87.6333080367 … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f -87.667569312
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 -87.6763559892
d7867b9d481d3f3d8225badf3be655a1f56de496 -87.6635175498
d70e65b5142ee7cbddc009b4939087afda184aac -87.6262105324
d54ca075925cb437317a5f89df300f56efe8d1fb -87.6838349425

pickup_centroid_location \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 {'type': 'Point', 'coordinates':
[-87.62197165…
96fdd5737ff418f493ba58aceb678481e41cc34c {'type': 'Point', 'coordinates':
[-87.67165362…
96c87ea59efebb6a4541619a6ba2f1abd1006d8b {'type': 'Point', 'coordinates':
[-87.65587878…
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 {'type': 'Point', 'coordinates':
[-87.65599818…
9658c724217d0298e1f6c286de46922b6c044192 {'type': 'Point', 'coordinates':
[-87.63330803…
…
…
d82a3591bd391eb5679cbcd69c6f59677488cb2f {'type': 'Point', 'coordinates':
[-87.66756931…
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 {'type': 'Point', 'coordinates':
[-87.67635598…
d7867b9d481d3f3d8225badf3be655a1f56de496 {'type': 'Point', 'coordinates':
[-87.66351754…
d70e65b5142ee7cbddc009b4939087afda184aac {'type': 'Point', 'coordinates':
[-87.62621053…
d54ca075925cb437317a5f89df300f56efe8d1fb {'type': 'Point', 'coordinates':
[-87.68383494…

start_year start_month start_day \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 2018 11 7
96fdd5737ff418f493ba58aceb678481e41cc34c 2018 11 7
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 2018 11 7
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 2018 11 7

9658c724217d0298e1f6c286de46922b6c044192 2018 11 7 … … … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f 2018 11 1 9

d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 2018 11 1
d7867b9d481d3f3d8225badf3be655a1f56de496 2018 11 1
d70e65b5142ee7cbddc009b4939087afda184aac 2018 11 1
d54ca075925cb437317a5f89df300f56efe8d1fb 2018 11 1

trip_id	start_weekday	start_hour
971beb1419f688efeff76a37f7bbce0db1027c11	2	23
96fdd5737ff418f493ba58aceb678481e41cc34c	2	23
96c87ea59efebb6a4541619a6ba2f1abd1006d8b	2	23
969de69a352f0f2ea1766fbee1f2b675dfd5ff84	2	23
9658c724217d0298e1f6c286de46922b6c044192	2	23
…	…	…
d82a3591bd391eb5679cbcd69c6f59677488cb2f	3	0
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2	3	0
d7867b9d481d3f3d8225badf3be655a1f56de496	3	0
d70e65b5142ee7cbddc009b4939087afda184aac	3	0
d54ca075925cb437317a5f89df300f56efe8d1fb	3	0

[1936828 rows x 17 columns]

5 In addition we are using the total_seconds method to find the duration, just in case if this column may be useful in analysis latter

[7]:
```
g = (trips_with_index.get('trip_end_timestamp').apply(lambda x: datetime.
 ,→datetime.strptime(x, f)) - trips_with_index.get('trip_start_timestamp'). ,→apply(lambda x:
 datetime.datetime.strptime(x, f))).apply(datetime.timedelta. ,→total_seconds)

trips = trips_with_start_times.assign(duration=g)
trips
```

[7]:
```
trip_start_timestamp \ trip_id
971beb1419f688efeff76a37f7bbce0db1027c11      2018-11-07T23:45:00.000
96fdd5737ff418f493ba58aceb678481e41cc34c      2018-11-07T23:45:00.000
96c87ea59efebb6a4541619a6ba2f1abd1006d8b      2018-11-07T23:45:00.000
969de69a352f0f2ea1766fbee1f2b675dfd5ff84      2018-11-07T23:45:00.000
9658c724217d0298e1f6c286de46922b6c044192      2018-11-07T23:45:00.000
… …
d82a3591bd391eb5679cbcd69c6f59677488cb2f      2018-11-01T00:00:00.000
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2      2018-11-01T00:00:00.000
d7867b9d481d3f3d8225badf3be655a1f56de496 2018-11-01T00:00:00.000
```



```
d70e65b5142ee7cbddc009b4939087afda184aac 2018-11-01T00:00:00.000
d54ca075925cb437317a5f89df300f56efe8d1fb 2018-11-01T00:00:00.000
```

trip_id	trip_end_timestamp
971beb1419f688efeff76a37f7bbce0db1027c11	2018-11-07T23:45:00.000
96fdd5737ff418f493ba58aceb678481e41cc34c	2018-11-08T00:00:00.000
96c87ea59efebb6a4541619a6ba2f1abd1006d8b	2018-11-08T00:00:00.000
969de69a352f0f2ea1766fbee1f2b675dfd5ff84	2018-11-07T23:45:00.000
9658c724217d0298e1f6c286de46922b6c044192	2018-11-08T00:00:00.000
… …	
d82a3591bd391eb5679cbcd69c6f59677488cb2f	2018-11-01T00:30:00.000
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2	2018-11-01T00:45:00.000
d7867b9d481d3f3d8225badf3be655a1f56de496	2018-11-01T00:15:00.000
d70e65b5142ee7cbddc009b4939087afda184aac	2018-11-01T00:15:00.000
d54ca075925cb437317a5f89df300f56efe8d1fb	2018-11-01T00:15:00.000

trip_id	trip_seconds	trip_miles
971beb1419f688efeff76a37f7bbce0db1027c11	771	5.23118898528
96fdd5737ff418f493ba58aceb678481e41cc34c	726	2.81755890464
96c87ea59efebb6a4541619a6ba2f1abd1006d8b	519	2.20618176654288
969de69a352f0f2ea1766fbee1f2b675dfd5ff84	256	1.08501496416
9658c724217d0298e1f6c286de46922b6c044192	1066	11.79099308704 … … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f	1100	8.57323393216
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2	2158	9.88755618256
d7867b9d481d3f3d8225badf3be655a1f56de496	459	2.03847066970998
d70e65b5142ee7cbddc009b4939087afda184aac	800	2.49130870064
d54ca075925cb437317a5f89df300f56efe8d1fb	479	4.8906459152

trip_id	pickup_community_area	fare	tip
971beb1419f688efeff76a37f7bbce0db1027c11	32	10	2
96fdd5737ff418f493ba58aceb678481e41cc34c	28	5	0
96c87ea59efebb6a4541619a6ba2f1abd1006d8b	3	5	0
969de69a352f0f2ea1766fbee1f2b675dfd5ff84	6	5	0
9658c724217d0298e1f6c286de46922b6c044192	8	17.5	0 … … … ..
d82a3591bd391eb5679cbcd69c6f59677488cb2f	31	12.5	1
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2	24	17.5	0
d7867b9d481d3f3d8225badf3be655a1f56de496	28	5	0
d70e65b5142ee7cbddc009b4939087afda184aac	8	7.5	0
d54ca075925cb437317a5f89df300f56efe8d1fb	5	7.5	0

trip_id	additional_charges	trip_total
	11	
971beb1419f688efeff76a37f7bbce0db1027c11	2.5	14.5
96fdd5737ff418f493ba58aceb678481e41cc34c	0	5
96c87ea59efebb6a4541619a6ba2f1abd1006d8b	2.5	7.5
969de69a352f0f2ea1766fbee1f2b675dfd5ff84	0	5
9658c724217d0298e1f6c286de46922b6c044192	2.5	20 … … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f	2.5	16
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2	0	17.5

d7867b9d481d3f3d8225badf3be655a1f56de496 2.5 7.5
d70e65b5142ee7cbddc009b4939087afda184aac 0 7.5
d54ca075925cb437317a5f89df300f56efe8d1fb 0 7.5

pickup_centroid_latitude \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 41.8774061234
96fdd5737ff418f493ba58aceb678481e41cc34c 41.8786674201
96c87ea59efebb6a4541619a6ba2f1abd1006d8b 41.96581197
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 41.9442266014
9658c724217d0298e1f6c286de46922b6c044192 41.899602111 … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f 41.8502663663
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 41.9012069941
d7867b9d481d3f3d8225badf3be655a1f56de496 41.874005383
d70e65b5142ee7cbddc009b4939087afda184aac 41.8991556134
d54ca075925cb437317a5f89df300f56efe8d1fb 41.9477915865

pickup_centroid_longitude \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 -87.6219716519
96fdd5737ff418f493ba58aceb678481e41cc34c -87.6716536214
96c87ea59efebb6a4541619a6ba2f1abd1006d8b -87.6558787862
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 -87.6559981815
9658c724217d0298e1f6c286de46922b6c044192 -87.6333080367 … …
d82a3591bd391eb5679cbcd69c6f59677488cb2f -87.667569312
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 -87.6763559892
d7867b9d481d3f3d8225badf3be655a1f56de496 -87.6635175498
d70e65b5142ee7cbddc009b4939087afda184aac -87.6262105324
d54ca075925cb437317a5f89df300f56efe8d1fb -87.6838349425

pickup_centroid_location \
trip_id
971beb1419f688efeff76a37f7bbce0db1027c11 {'type': 'Point', 'coordinates':
[-87.62197165…
96fdd5737ff418f493ba58aceb678481e41cc34c {'type': 'Point', 'coordinates':
[-87.67165362…
96c87ea59efebb6a4541619a6ba2f1abd1006d8b {'type': 'Point', 'coordinates': 12

[-87.65587878…
969de69a352f0f2ea1766fbee1f2b675dfd5ff84 {'type': 'Point', 'coordinates':
[-87.65599818…
9658c724217d0298e1f6c286de46922b6c044192 {'type': 'Point', 'coordinates':
[-87.63330803…
…
…
d82a3591bd391eb5679cbcd69c6f59677488cb2f {'type': 'Point', 'coordinates':
[-87.66756931…
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2 {'type': 'Point', 'coordinates':
[-87.67635598…
d7867b9d481d3f3d8225badf3be655a1f56de496 {'type': 'Point', 'coordinates':
[-87.66351754…
d70e65b5142ee7cbddc009b4939087afda184aac {'type': 'Point', 'coordinates':

[-87.62621053…
d54ca075925cb437317a5f89df300f56efe8d1fb {'type': 'Point', 'coordinates':
[-87.68383494…

trip_id	start_year	start_month	start_day
971beb1419f688efeff76a37f7bbce0db1027c11	2018	11	7
96fdd5737ff418f493ba58aceb678481e41cc34c	2018	11	7
96c87ea59efebb6a4541619a6ba2f1abd1006d8b	2018	11	7
969de69a352f0f2ea1766fbee1f2b675dfd5ff84	2018	11	7
9658c724217d0298e1f6c286de46922b6c044192	2018	11	7
…	…	…	…
d82a3591bd391eb5679cbcd69c6f59677488cb2f	2018	11	1
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2	2018	11	1
d7867b9d481d3f3d8225badf3be655a1f56de496	2018	11	1
d70e65b5142ee7cbddc009b4939087afda184aac	2018	11	1
d54ca075925cb437317a5f89df300f56efe8d1fb	2018	11	1

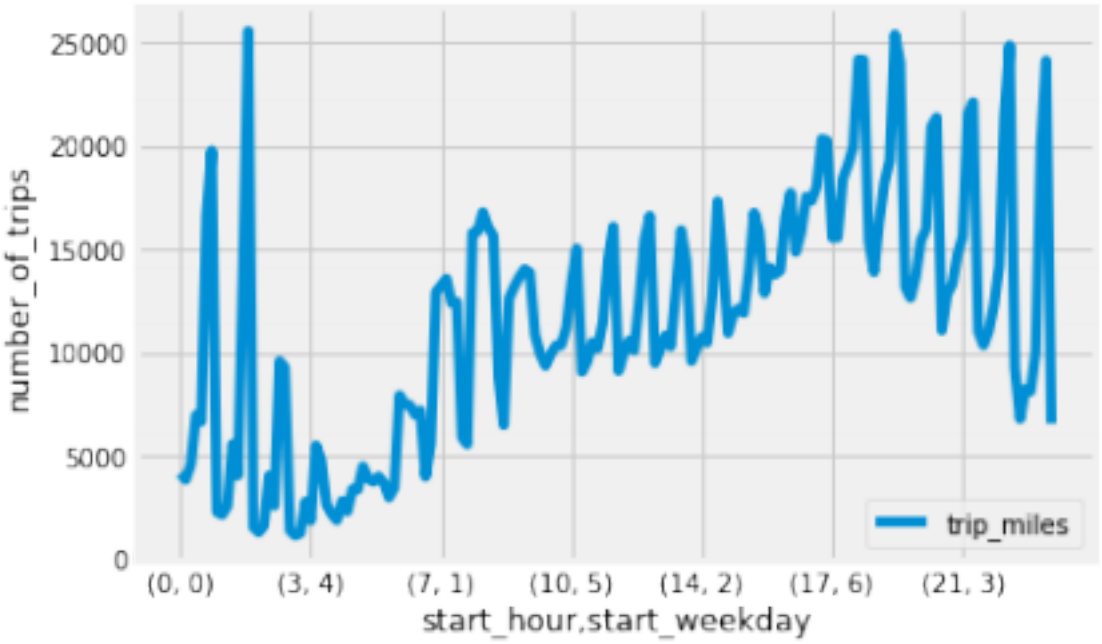
trip_id	start_weekday	start_hour	duration
971beb1419f688efeff76a37f7bbce0db1027c11	2	23	0.0
96fdd5737ff418f493ba58aceb678481e41cc34c	2	23	900.0
96c87ea59efebb6a4541619a6ba2f1abd1006d8b	2	23	900.0
969de69a352f0f2ea1766fbee1f2b675dfd5ff84	2	23	0.0
9658c724217d0298e1f6c286de46922b6c044192	2	23	900.0
…	…	…	…
d82a3591bd391eb5679cbcd69c6f59677488cb2f	3	0	1800.0
d7c725a5528f5435c7cd4c4b2348f0e72544e3c2	3	0	2700.0
d7867b9d481d3f3d8225badf3be655a1f56de496	3	0	900.0
d70e65b5142ee7cbddc009b4939087afda184aac	3	0	900.0
d54ca075925cb437317a5f89df300f56efe8d1fb	3	0	900.0

[1936828 rows x 18 columns]

6 Here we are grouping by the hour of the day and the day of the week to see how the number of trips on the y-axis fluctuate in relation to time

[18]:
```
monthly_trips =trips.groupby(["start_hour","start_weekday"]).count()
monthly_trips.plot(kind='line', y='trip_miles').set_ylabel("number_of_trips")
```

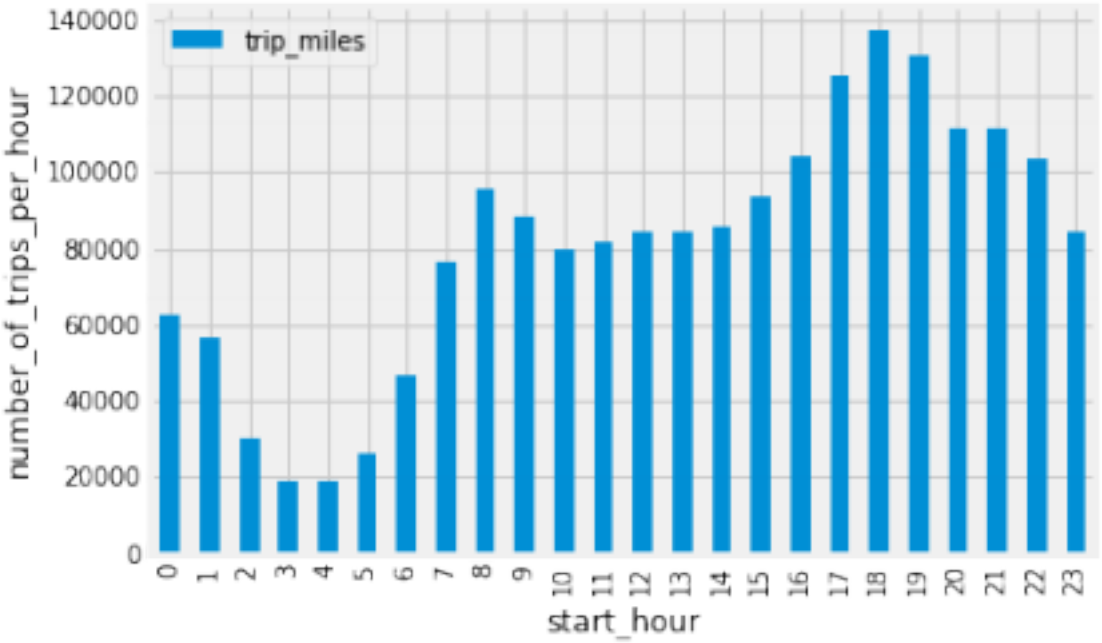
[18]: Text(0, 0.5, 'number_of_trips')



7 Here we are grouping by just the hour to see out of all the ob servations, which hour (0-23) of the day has the greatest overall number of trips which the plot suggest to be the 18th hour along with its neighbors and the least is at the 4th hour also along with its neighbors

```
[11]: hours_24 = trips.groupby("start_hour").count()
      hours_24.plot(kind="bar", y="trip_miles").set_ylabel("number_of_trips_per_hour") [11]: Text(0,

0.5, 'number_of_trips_per_hour')
```



8 Here we are printing the result of grouping by hour and us ing .count() which results in the observation values across the columns to hold the count of each one of these 24 hours

[12]: hours_24

[12]: trip_start_timestamp trip_end_timestamp trip_seconds \ start_hour
0 62833 62833 62833 1 56257 56257 54568 2 30161 30161 28631 3 18957
18957 18957 4 18810 18810 18810 … … … …
19 130529 130529 130529 20 111300 111300 111300 21 111353 111353
111353 22 103718 103718 103718 23 84054 84054 84054

trip_miles pickup_community_area fare tip \
start_hour
0 62833 62833 62833 62833

15
1 56257 56257 56257 56257 2 30161 30161 30161 30161 3 18957
18957 18957 18957 4 18810 18810 18810 18810 … … … … …
19 130529 130529 130529 130529 20 111300 111300 111299 111300
21 111353 111353 111353 111353 22 103718 103718 103718 103718
23 84054 84054 84054 84054

additional_charges trip_total pickup_centroid_latitude \ start_hour
0 62833 62833 62833 1 56257 56257 56257 2 30161 30161 30161 3 18957
18957 18957 4 18810 18810 18810 … … … …
19 130529 130529 130529 20 111299 111299 111300 21 111353 111353 111353
22 103718 103718 103718 23 84054 84054 84054

pickup_centroid_longitude pickup_centroid_location start_year \ start_hour
0 62833 62833 62833 1 56257 56257 56257 2 30161 30161 30161 3 18957 18957

18957 4 18810 18810 18810 … … … …
19 130529 130529 130529 20 111300 111300 111300 21 111353 111353 111353 22
103718 103718 103718 23 84054 84054 84054

start_month start_day start_weekday duration
start_hour
0 62833 62833 62833 62833 1 56257 56257 56257 56257 2 30161
30161 30161 30161 3 18957 18957 18957 18957 4 18810 18810
18810 18810 … … … … …

16

19 130529 130529 130529 130529
20 111300 111300 111300 111300
21 111353 111353 111353 111353
22 103718 103718 103718 103718
23 84054 84054 84054 84054

[24 rows x 17 columns]

9 Here we are now grouping by the community area number (1-77) in order to see which community areas have the highest mean for their fare in Chicago rideshares, and as we can see from the graph the top values (76 & 56) correspond with Chicago's two airports that have much higher means which could possibly be a result of its greater number of data points or as a result of an policy put into place by rideshare companies

[14]:
```
trips = trips.assign(fare=trips.get("fare").apply(float))
hours_24 = trips.groupby("pickup_community_area").mean()
hours_24 =hours_24.sort_values(by = "fare", ascending=False).iloc[:20]
hours_24.plot(kind="bar", y="fare").set_ylabel("mean_fare")
hours_24
```

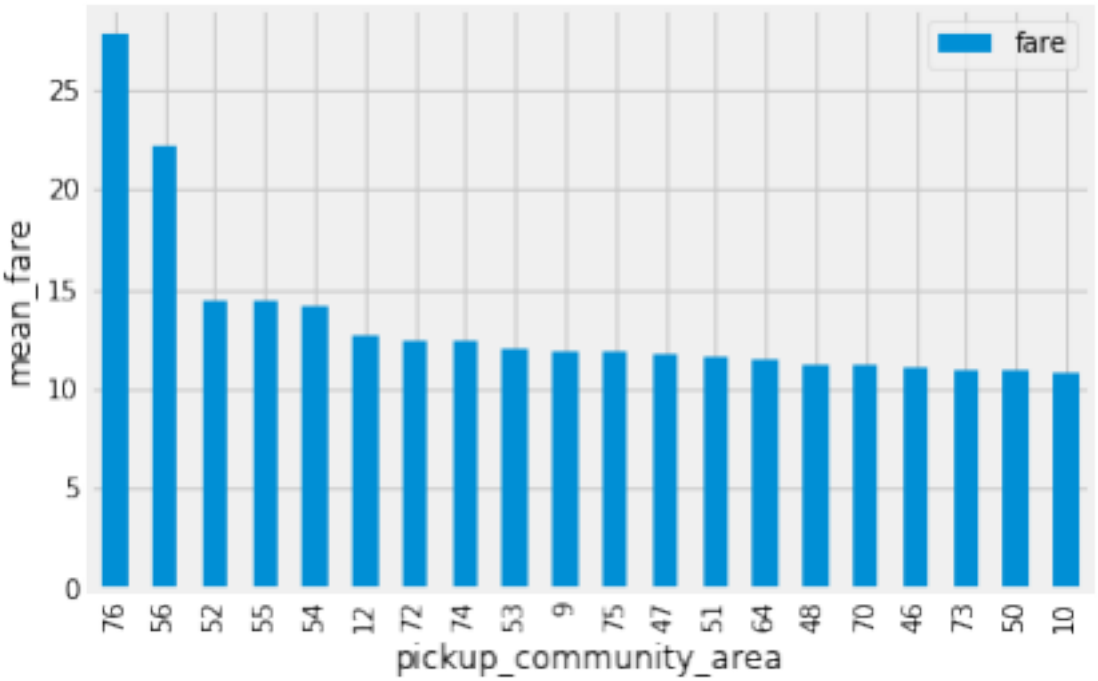
[14]: fare start_year start_month start_day \ pickup_community_area
76 27.753006 2018.0 11.0 4.091479 56 22.219570 2018.0 11.0 4.016532 52
14.446355 2018.0 11.0 3.825309 55 14.367871 2018.0 11.0 3.868821 54
14.134814 2018.0 11.0 3.982438 … … … … …
70 11.186211 2018.0 11.0 3.881483 46 11.115905 2018.0 11.0 3.885728 73
10.941341 2018.0 11.0 3.864358 50 10.931922 2018.0 11.0 4.018868 10
10.855215 2018.0 11.0 3.825744

start_weekday start_hour duration
pickup_community_area
76 2.803821 14.810955 2165.468894
56 2.970724 15.086763 1805.362004
52 3.090784 12.240715 1301.100413
55 2.941065 13.332700 1240.494297

54 3.162190 11.638430 1274.690083

... ...
70 3.022287 12.502604 1242.678609
46 3.090951 12.675840 1159.841418
73 3.084693 12.640000 1075.575419
50 2.809791 13.905660 1078.072412
10 3.301322 12.729893 992.875505

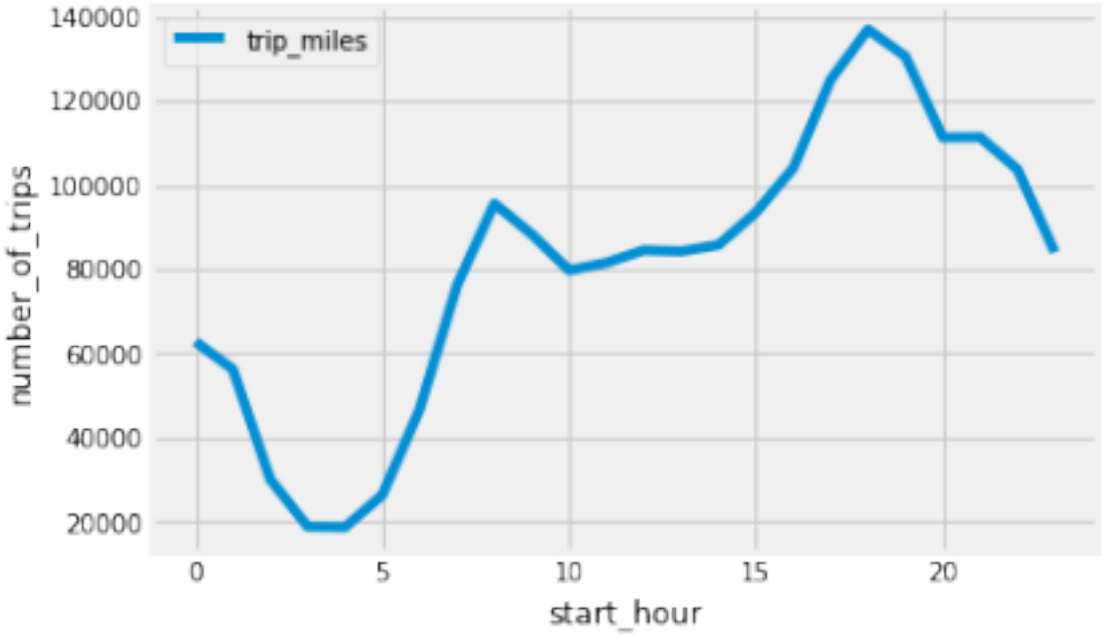
[20 rows x 7 columns]



10 This line plot is also using groupby hour and .count() as well which can show the realationship as time changes amongst the hours in a day, and as we can see like in the bar plot that the highest greatest number of trip counts is located at hour 18 and its neighbors while the lowest is at hour number 4 and its neighbors

[16]: monthly_trips =trips.groupby(["start_hour"]).count()
monthly_trips.plot(kind='line', y='trip_miles').set_ylabel("number_of_trips") [16]: Text(0, 0.5,

'number_of_trips')


trip_miles
number_of_trips
start_hour
140000
120000
100000
80000
60000
40000
20000
0
5
10
15
20