Leah Scherschel
I535 Big Data Project B Report

Introduction

In September, 2010, the District Department of Transportation of Washington, DC began a bikesharing program called Capital Bikeshare. With this program, users could check out a bike from one solar powered station, use it for their travel needs, and return it to whichever station was most convenient. This new (at the time) transportation system provided a way for people to travel more quickly than walking while not burning fossil fuels required by cars or busses. Thoughtful analysis of data taken during each bike trip could shed light on the success of this program, offer direction for future improvements of Capital Bikeshare, and provide an opportunity to use the skills and technologies I have learned in the I535: Management, Access, and Use of Big and Complex Data class at Indiana University.

Background

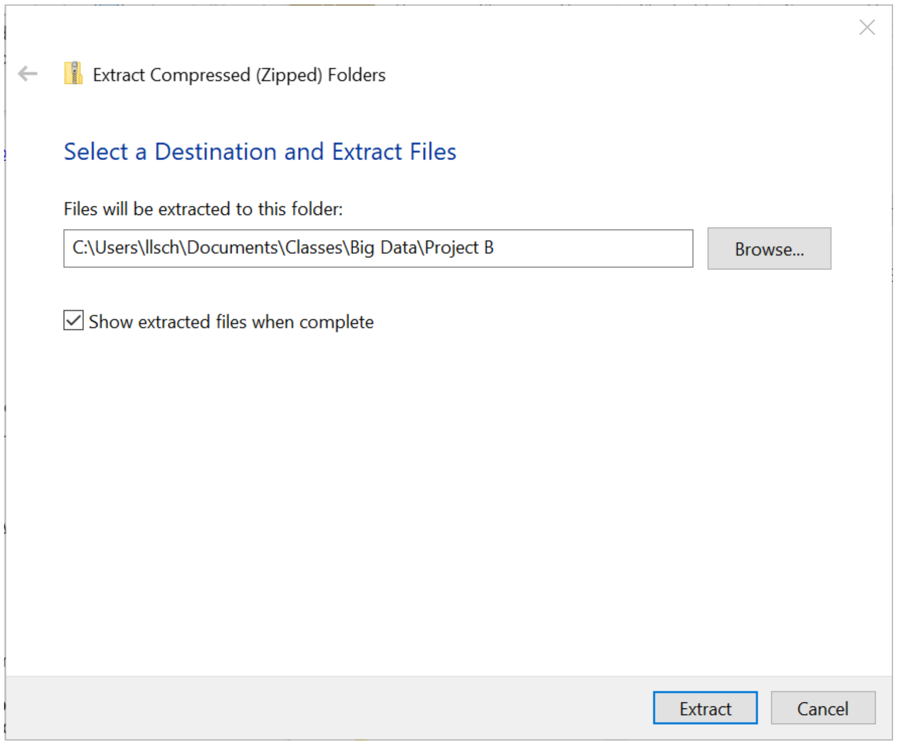
I decided to implement a simple data pipeline while using a data cleaning technology for this project. Following the data lifecycle steps of Plan, Acquire, Process, Analyze, and Share, I decided to make a plan for the pipeline and dataset I would use, download the data from a public source, clean it by using OpenRefine, perform analysis and create visualizations in RStudio, and write this report to share my findings. To implement my pipeline and cleaning technology, I selected the Capital Bikeshare data from their startup quarter, September-December, 2010. This data is published each quarter at https://s3.amazonaws.com/capitalbikeshare-data/index.html. On the prior handling of the dataset, Capital Bikeshare's website states, "This data has been processed to remove trips that are taken by staff as they service and inspect the system, trips that are taken to/from any of our "test" stations at our warehouses and any trips lasting less than 60 seconds (potentially false starts or users trying to re-dock a bike to ensure it's secure)." Each dataset includes:

- Duration – Duration of trip in seconds
- Start Date – Includes start date and time
- End Date – Includes end date and time
- Start Station – Includes starting station name and number
- End Station – Includes ending station name and number
- Bike Number – Includes ID number of bike used for the trip
- Member Type – Indicates whether user was a "registered" member (Annual Member, 30-Day Member or Day Key Member) or a "casual" rider (Single Trip, 24-Hour Pass, 3-Day Pass or 5-Day Pass)

I chose this dataset for several reasons. I wanted to investigate how a non-traditional public transportation system was used in its first quarter as well as lay groundwork for future investigation with my data pipeline. I wanted to explore how members used the bikes. I thought it would be useful if the data could be utilized to tell which bikes need service. In addition, I was curious about how much the bikes were actually being used and how that might change over time. Success of alternative forms of public transportation is important because they may improve ease of travel in metropolitan areas and decrease the creation of air pollutants.

Methodology

To begin, I acquired data by downloading the file containing the 2010 dataset, "2010-capitalbikeshare-tripdata.zip", and extracting its contents.


Extract Compressed (Zipped) Folders

Select a Destination and Extract Files

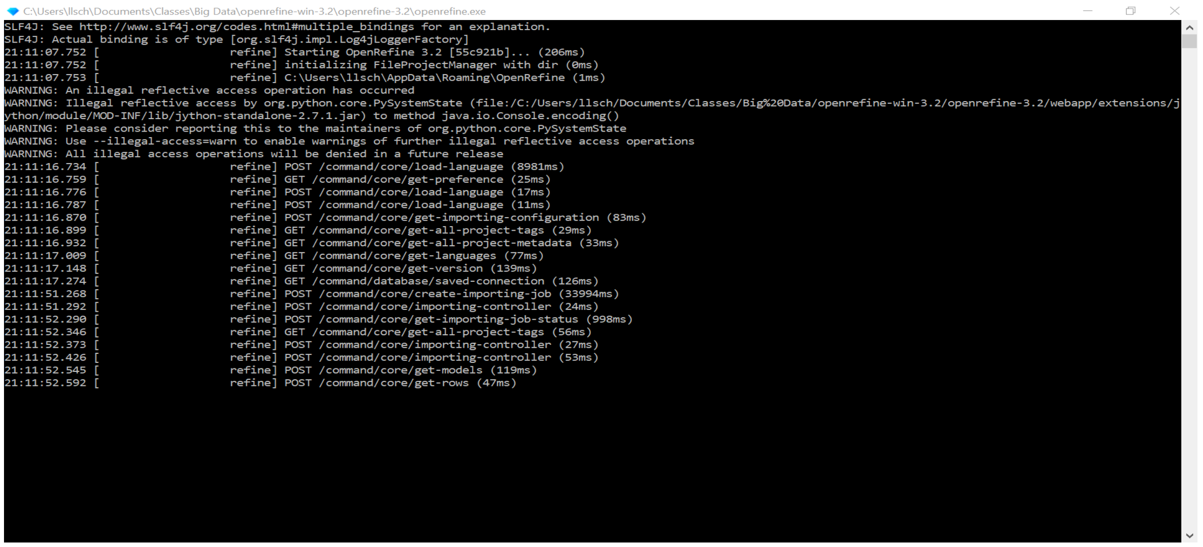
Files will be extracted to this folder:

C:\Users\llsch\Documents\Classes\Big Data\Project B

☑ Show extracted files when complete

The file only contained one comma separated value file, "2010-capitalbikeshare-tripdata.csv".

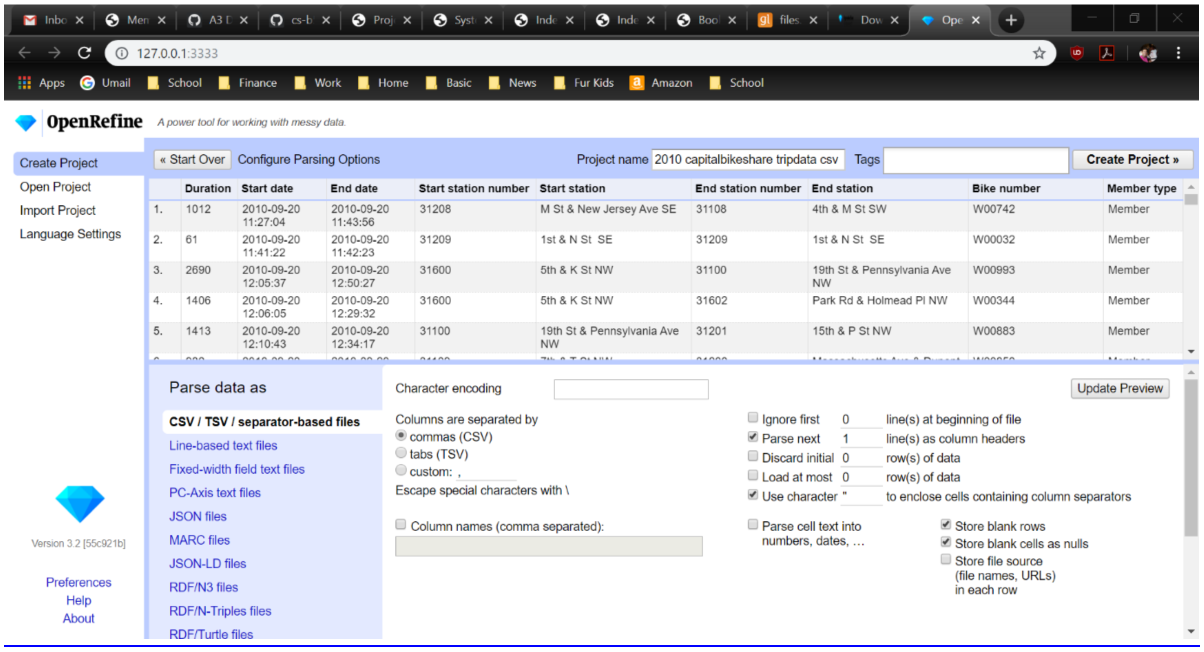
Next, I began to process and clean the data. I used the application OpenRefine for my data cleaning needs. This is available for download at: https://openrefine.org/download.html. OpenRefine is run locally, but interactions take place through your web browser. When OpenRefine is run, it opens a terminal as below:



C:\Users\llsch\Documents\Classes\Big Data\openrefine-win-3.2\openrefine-3.2\openrefine.exe

```
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
21:11:07.752 [                  refine] Starting OpenRefine 3.2 [55c921b]... (206ms)
21:11:07.752 [                  refine] initializing FileProjectManager with dir (0ms)
21:11:07.753 [                  refine] C:\Users\llsch\AppData\Roaming\OpenRefine (1ms)
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.python.core.PySystemState (file:/C:/Users/llsch/Documents/Classes/Big%20Data/openrefine-win-3.2/openrefine-3.2/webapp/extensions/j
ython/module/MOD-INF/lib/jython-standalone-2.7.1.jar) to method java.io.Console.encoding()
WARNING: Please consider reporting this to the maintainers of org.python.core.PySystemState
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
21:11:16.734 [                  refine] POST /command/core/load-language (8981ms)
21:11:16.759 [                  refine] GET /command/core/get-preference (25ms)
21:11:16.776 [                  refine] POST /command/core/load-language (17ms)
21:11:16.787 [                  refine] POST /command/core/load-language (11ms)
21:11:16.870 [                  refine] POST /command/core/get-importing-configuration (83ms)
21:11:16.899 [                  refine] GET /command/core/get-all-project-tags (29ms)
21:11:16.932 [                  refine] GET /command/core/get-all-project-metadata (33ms)
21:11:17.009 [                  refine] GET /command/core/get-languages (77ms)
21:11:17.148 [                  refine] GET /command/core/get-version (139ms)
21:11:17.274 [                  refine] GET /command/database/saved-connection (126ms)
21:11:51.268 [                  refine] POST /command/core/create-importing-job (33994ms)
21:11:51.292 [                  refine] POST /command/core/importing-controller (24ms)
21:11:52.290 [                  refine] POST /command/core/get-importing-job-status (998ms)
21:11:52.346 [                  refine] GET /command/core/get-all-project-tags (56ms)
21:11:52.373 [                  refine] POST /command/core/importing-controller (27ms)
21:11:52.426 [                  refine] POST /command/core/importing-controller (53ms)
21:11:52.545 [                  refine] POST /command/core/get-models (119ms)
21:11:52.592 [                  refine] POST /command/core/get-rows (47ms)
```

Then, it opens a new tab in your browser where you can load your local data set. I loaded "2010-capitalbikeshare-tripdata.csv" from my local files and used it to create a new project.



Configure Parsing Options

Project name: 2010 capitalbikeshare tripdata csv   Tags:

	Duration	Start date	End date	Start station number	Start station	End station number	End station	Bike number	Member type
1.	1012	2010-09-20 11:27:04	2010-09-20 11:43:56	31208	M St & New Jersey Ave SE	31108	4th & M St SW	W00742	Member
2.	61	2010-09-20 11:41:22	2010-09-20 11:42:23	31209	1st & N St SE	31209	1st & N St SE	W00032	Member
3.	2690	2010-09-20 12:05:37	2010-09-20 12:50:27	31600	5th & K St NW	31100	19th St & Pennsylvania Ave NW	W00993	Member
4.	1406	2010-09-20 12:06:05	2010-09-20 12:29:32	31600	5th & K St NW	31602	Park Rd & Holmead Pl NW	W00344	Member
5.	1413	2010-09-20 12:10:43	2010-09-20 12:34:17	31100	19th St & Pennsylvania Ave NW	31201	15th & P St NW	W00883	Member

Parse data as

CSV / TSV / separator-based files
Line-based text files
Fixed-width field text files
PC-Axis text files
JSON files
MARC files
JSON-LD files
RDF/N3 files
RDF/N-Triples files
RDF/Turtle files

Character encoding

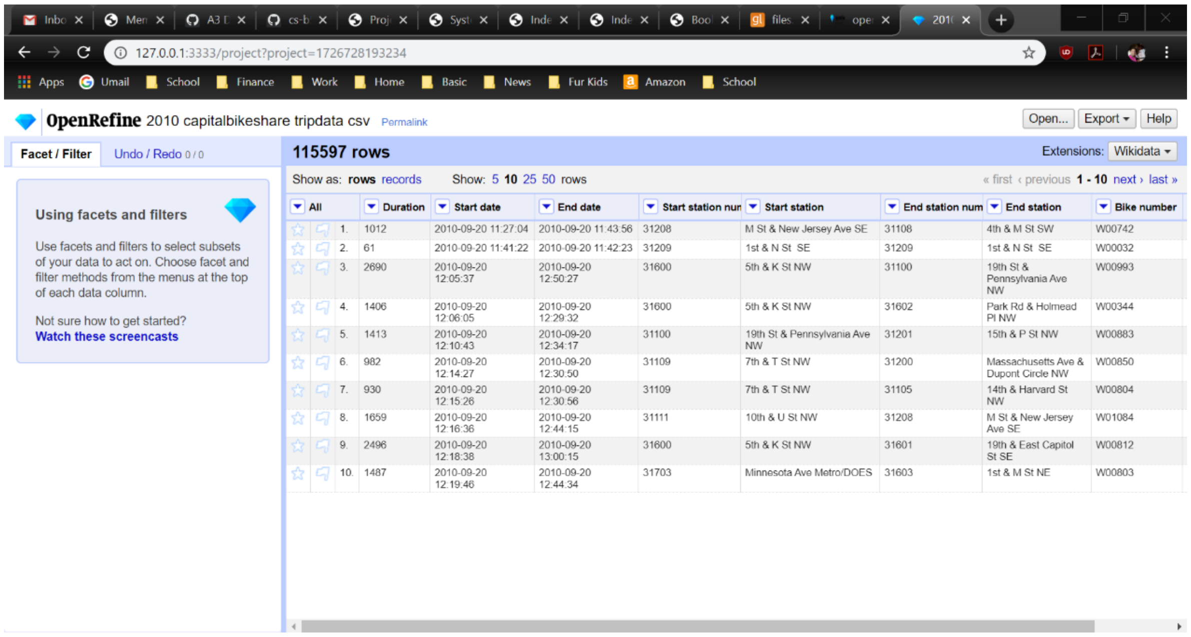
Columns are separated by
● commas (CSV)
○ tabs (TSV)
○ custom: ,
Escape special characters with \

☐ Column names (comma separated):

☐ Ignore first 0 line(s) at beginning of file
☑ Parse next 1 line(s) as column headers
☐ Discard initial 0 row(s) of data
☐ Load at most 0 row(s) of data
☑ Use character " to enclose cells containing column separators

☐ Parse cell text into numbers, dates, …

☑ Store blank rows
☑ Store blank cells as nulls
☐ Store file source (file names, URLs) in each row



OpenRefine 2010 capitalbikeshare tripdata csv Permalink

Facet / Filter | Undo / Redo 0 / 0

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

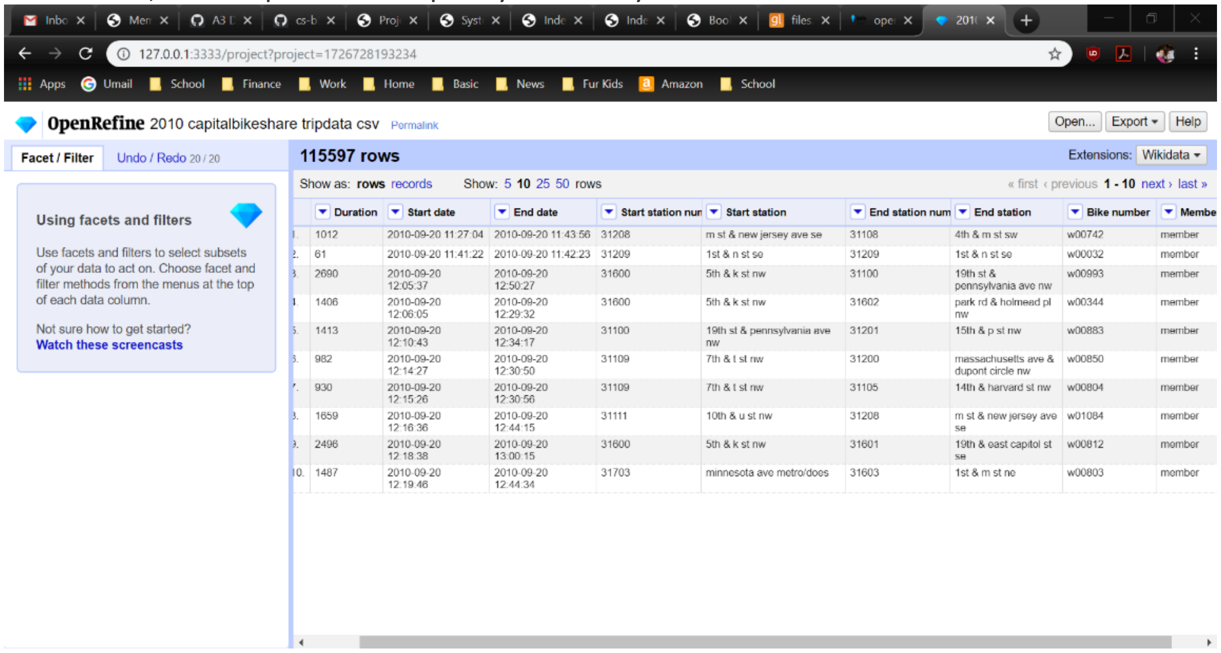
Not sure how to get started?
Watch these screencasts

115597 rows

Show as: rows records Show: 5 10 25 50 rows « first ‹ previous 1 - 10 next › last »

All	Duration	Start date	End date	Start station num	Start station	End station num	End station	Bike number
1.	1012	2010-09-20 11:27:04	2010-09-20 11:43:56	31208	M St & New Jersey Ave SE	31108	4th & M St SW	W00742
2.	61	2010-09-20 11:41:22	2010-09-20 11:42:23	31209	1st & N St SE	31209	1st & N St SE	W00032
3.	2690	2010-09-20 12:05:37	2010-09-20 12:50:27	31600	5th & K St NW	31100	19th St & Pennsylvania Ave NW	W00993
4.	1406	2010-09-20 12:06:05	2010-09-20 12:29:32	31600	5th & K St NW	31602	Park Rd & Holmead Pl NW	W00344
5.	1413	2010-09-20 12:10:43	2010-09-20 12:34:17	31100	19th St & Pennsylvania Ave NW	31201	15th & P St NW	W00883
6.	982	2010-09-20 12:14:27	2010-09-20 12:30:50	31109	7th & T St NW	31200	Massachusetts Ave & Dupont Circle NW	W00850
7.	930	2010-09-20 12:15:26	2010-09-20 12:30:56	31109	7th & T St NW	31105	14th & Harvard St NW	W00804
8.	1659	2010-09-20 12:16:36	2010-09-20 12:44:15	31111	10th & U St NW	31208	M St & New Jersey Ave SE	W01084
9.	2496	2010-09-20 12:18:38	2010-09-20 13:00:15	31600	5th & K St NW	31601	19th & East Capitol St SE	W00812
10.	1487	2010-09-20 12:19:46	2010-09-20 12:44:34	31703	Minnesota Ave Metro/DOES	31603	1st & M St NE	W00803

My first step in cleaning the data was to make sure there were no leading or trailing whitespaces, then convert all strings to lowercase. By selecting the drop down menu for each column, then "Edit cells -> Common transforms -> Trim leading and trailing whitespace/To lowercase", I accomplished this quickly and easily.



OpenRefine 2010 capitalbikeshare tripdata csv Permalink

Facet / Filter | Undo / Redo 20 / 20

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

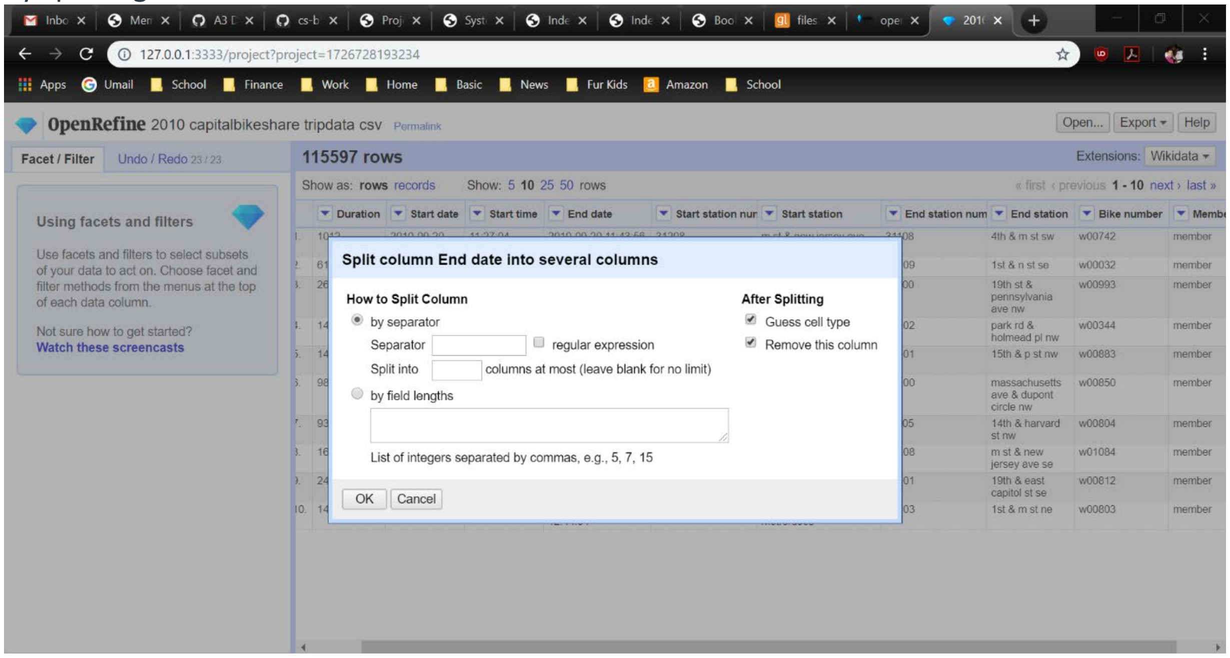
Not sure how to get started?
Watch these screencasts

115597 rows

Show as: rows records Show: 5 10 25 50 rows « first ‹ previous 1 - 10 next › last »

	Duration	Start date	End date	Start station num	Start station	End station num	End station	Bike number	Membe
1.	1012	2010-09-20 11:27:04	2010-09-20 11:43:56	31208	m st & new jersey ave se	31108	4th & m st sw	w00742	member
2.	61	2010-09-20 11:41:22	2010-09-20 11:42:23	31209	1st & n st se	31209	1st & n st se	w00032	member
3.	2690	2010-09-20 12:05:37	2010-09-20 12:50:27	31600	5th & k st nw	31100	19th st & pennsylvania ave nw	w00993	member
4.	1406	2010-09-20 12:06:05	2010-09-20 12:29:32	31600	5th & k st nw	31602	park rd & holmead pl nw	w00344	member
5.	1413	2010-09-20 12:10:43	2010-09-20 12:34:17	31100	19th st & pennsylvania ave nw	31201	15th & p st nw	w00883	member
6.	982	2010-09-20 12:14:27	2010-09-20 12:30:50	31109	7th & t st nw	31200	massachusetts ave & dupont circle nw	w00850	member
7.	930	2010-09-20 12:15:26	2010-09-20 12:30:56	31109	7th & t st nw	31105	14th & harvard st nw	w00804	member
8.	1659	2010-09-20 12:16:36	2010-09-20 12:44:15	31111	10th & u st nw	31208	m st & new jersey ave se	w01084	member
9.	2496	2010-09-20 12:18:38	2010-09-20 13:00:15	31600	5th & k st nw	31601	19th & east capitol st se	w00812	member
10.	1487	2010-09-20 12:19:46	2010-09-20 12:44:34	31703	minnesota ave metro/does	31603	1st & m st ne	w00803	member

Next, I split the values in the Start date and End date columns by selecting "Edit column -> Split into several columns..." from the drop down menu. Each column was split into date and time columns by splitting on " ", then the date columns were split into year, month, and day columns by splitting on "-".



OpenRefine 2010 capitalbikeshare tripdata csv

115597 rows

Split column End date into several columns

How to Split Column

[✓ by separator]

Separator ☐ regular expression

Split into columns at most (leave blank for no limit)

by field lengths

List of integers separated by commas, e.g., 5, 7, 15

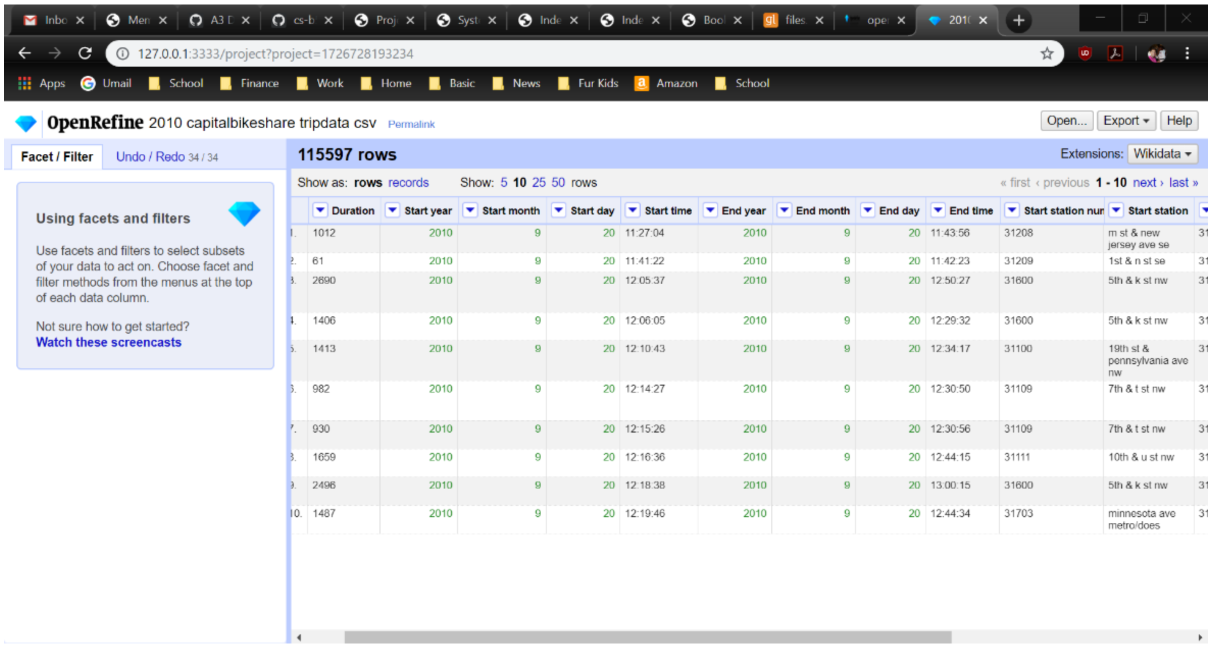
After Splitting

☑ Guess cell type

☑ Remove this column

OK Cancel

Through this, I went from 2 columns with a lot of information (Start date, End date) to 8 much more tidy variables (Start year, Start month, Start day, Start time, End year, End month, End day, and End time).

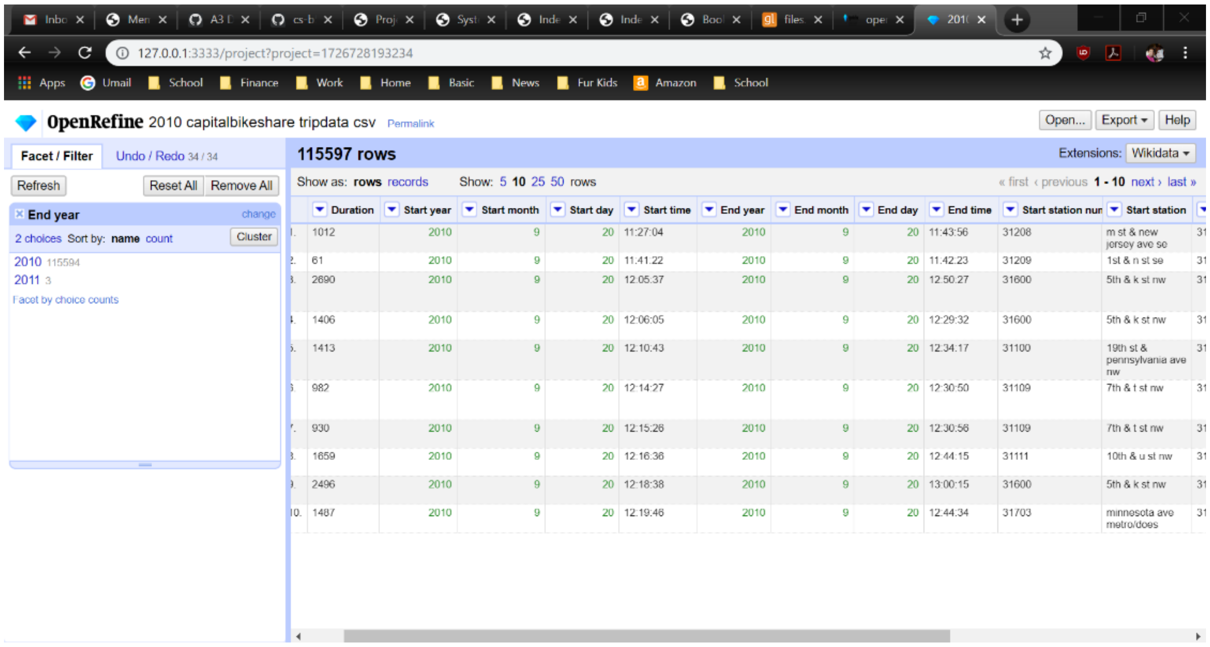


OpenRefine 2010 capitalbikeshare tripdata csv

115597 rows

	Duration	Start year	Start month	Start day	Start time	End year	End month	End day	End time	Start station nur	Start station
1.	1012	2010	9	20	11:27:04	2010	9	20	11:43:56	31208	m st & new jersey ave se
2.	61	2010	9	20	11:41:22	2010	9	20	11:42:23	31209	1st & n st se
3.	2690	2010	9	20	12:05:37	2010	9	20	12:50:27	31600	5th & k st nw
4.	1406	2010	9	20	12:06:05	2010	9	20	12:29:32	31600	5th & k st nw
5.	1413	2010	9	20	12:10:43	2010	9	20	12:34:17	31100	19th st & pennsylvania ave nw
6.	982	2010	9	20	12:14:27	2010	9	20	12:30:50	31109	7th & t st nw
7.	930	2010	9	20	12:15:26	2010	9	20	12:30:56	31109	7th & t st nw
8.	1659	2010	9	20	12:16:36	2010	9	20	12:44:15	31111	10th & u st nw
9.	2496	2010	9	20	12:18:38	2010	9	20	13:00:15	31600	5th & k st nw
10.	1487	2010	9	20	12:19:46	2010	9	20	12:44:34	31703	minnesota ave metro/does

Since I was focusing solely on trips in 2010, I removed the 3 records that had started in 2010 and ended in the year 2011.



OpenRefine 2010 capitalbikeshare tripdata csv Permalink

Facet / Filter | Undo / Redo 34 / 34

End year

2 choices Sort by: name count

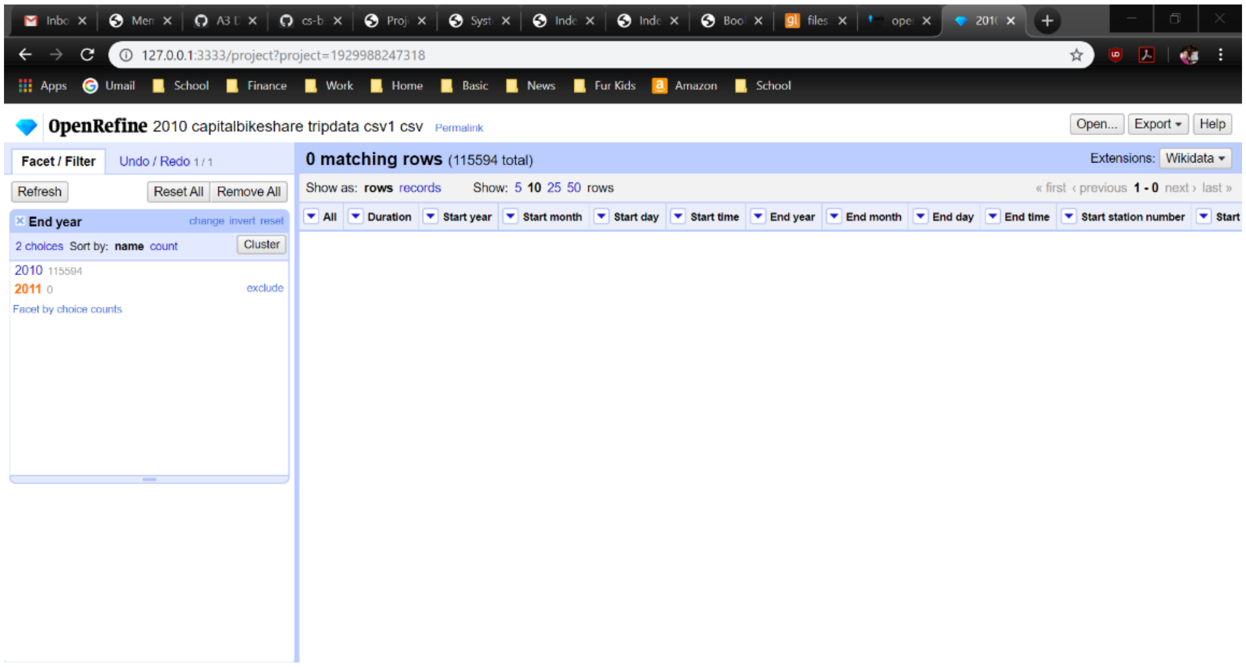
2010 115594
2011 3

Facet by choice counts

115597 rows

Show as: rows records Show: 5 10 25 50 rows « first ‹ previous 1 - 10 next › last »

	Duration	Start year	Start month	Start day	Start time	End year	End month	End day	End time	Start station nur	Start station	
1.	1012	2010	9	20	11:27:04	2010	9	20	11:43:56	31208	m st & new jersey ave se	31
2.	61	2010	9	20	11:41:22	2010	9	20	11:42:23	31209	1st & n st se	31
3.	2690	2010	9	20	12:05:37	2010	9	20	12:50:27	31600	5th & k st nw	31
4.	1406	2010	9	20	12:06:05	2010	9	20	12:29:32	31600	5th & k st nw	31
5.	1413	2010	9	20	12:10:43	2010	9	20	12:34:17	31100	19th st & pennsylvania ave nw	31
6.	982	2010	9	20	12:14:27	2010	9	20	12:30:50	31109	7th & t st nw	31
7.	930	2010	9	20	12:15:26	2010	9	20	12:30:56	31109	7th & t st nw	31
8.	1659	2010	9	20	12:16:36	2010	9	20	12:44:15	31111	10th & u st nw	31
9.	2496	2010	9	20	12:18:38	2010	9	20	13:00:15	31600	5th & k st nw	31
10.	1487	2010	9	20	12:19:46	2010	9	20	12:44:34	31703	minnesota ave metro/does	31



OpenRefine 2010 capitalbikeshare tripdata csv1 csv Permalink

Facet / Filter | Undo / Redo 1 / 1

End year

2 choices Sort by: name count

2010 115594
2011 0

Facet by choice counts

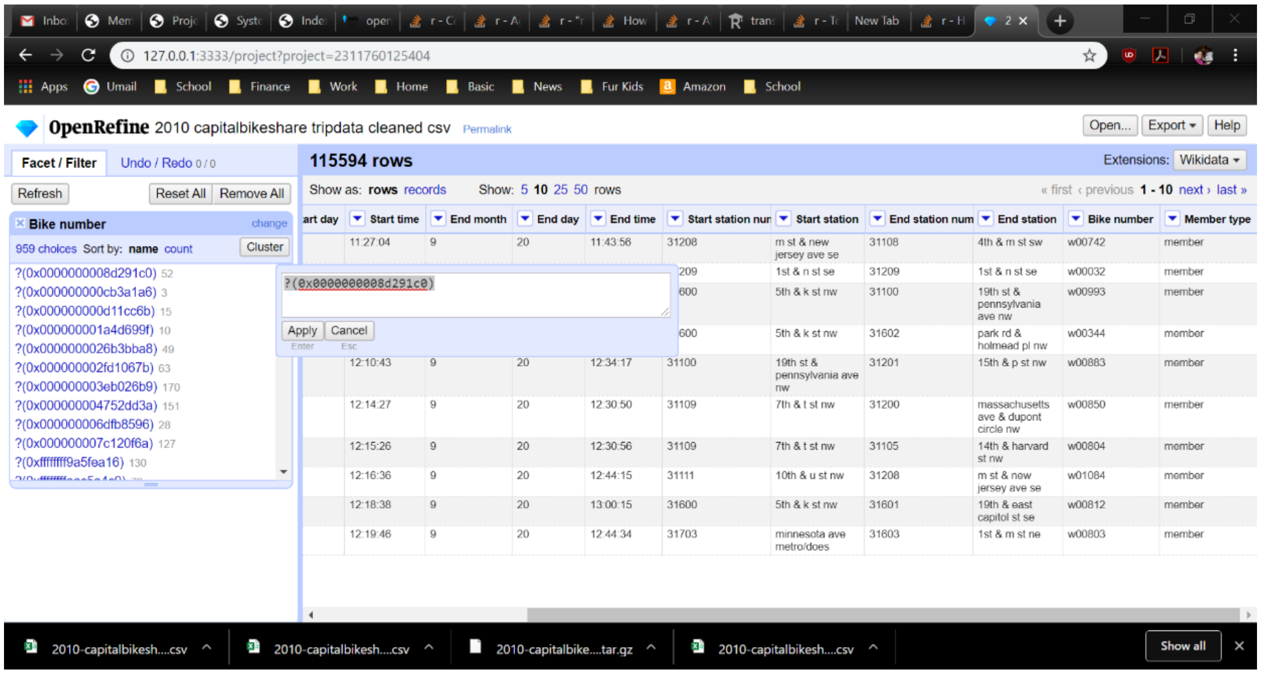
0 matching rows (115594 total)

Show as: rows records Show: 5 10 25 50 rows « first ‹ previous 1 - 0 next › last »

All	Duration	Start year	Start month	Start day	Start time	End year	End month	End day	End time	Start station number	Start

With those rows out of the dataset, all observations started and ended in 2010. Therefore, I removed the Start year and End year columns. Within the Bike number variable, there were 21 observations that looked like they had not been entered in the correct format and were incoherent. I edited those entries to have a Bike number value of “unknown”.

Leah Scherschel
I535 Big Data Project B Report


OpenRefine 2010 capitalbikeshare tripdata cleaned csv

115594 rows

Show as: **rows** records Show: 5 **10** 25 50 rows « first ‹ previous **1 - 10** next › last »

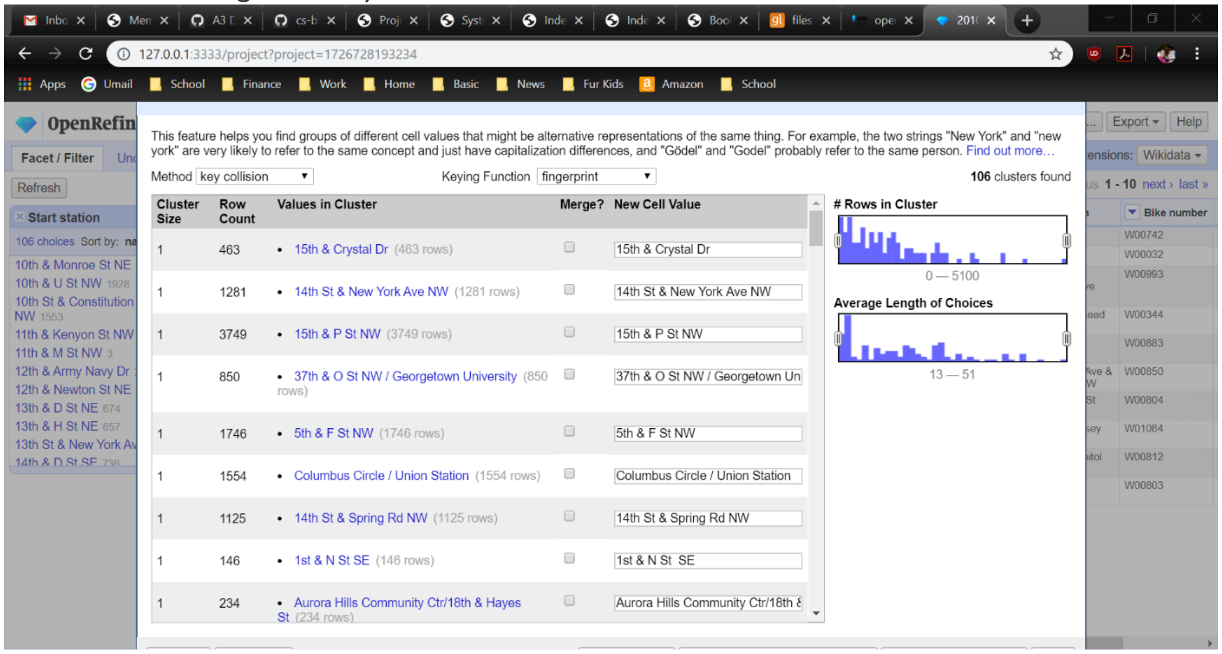
Bike number — 959 choices Sort by: **name** count

- ?(0x0000000008d291c0) 52
- ?(0x000000000cb3a1a6) 3
- ?(0x000000000d11cc6b) 15
- ?(0x000000001a4d699f) 10
- ?(0x0000000026b3bba8) 49
- ?(0x000000002fd1067b) 63
- ?(0x000000003eb026b9) 170
- ?(0x000000004752dd3a) 151
- ?(0x000000006dfb8596) 28
- ?(0x000000007c120f6a) 127
- ?(0xffffffff9a5fea16) 130

```
?(0x0000000008d291c0)
```

art day	Start time	End month	End day	End time	Start station num	Start station	End station num	End station	Bike number	Member type
	11:27:04	9	20	11:43:56	31208	m st & new jersey ave se	31108	4th & m st sw	w00742	member
					209	1st & n st se	31209	1st & n st se	w00032	member
					600	5th & k st nw	31100	19th st & pennsylvania ave nw	w00993	member
					600	5th & k st nw	31602	park rd & holmead pl nw	w00344	member
	12:10:43	9	20	12:34:17	31100	19th st & pennsylvania ave nw	31201	15th & p st nw	w00883	member
	12:14:27	9	20	12:30:50	31109	7th & t st nw	31200	massachusetts ave & dupont circle nw	w00850	member
	12:15:26	9	20	12:30:56	31109	7th & t st nw	31105	14th & harvard st nw	w00804	member
	12:16:36	9	20	12:44:15	31111	10th & u st nw	31208	m st & new jersey ave se	w01084	member
	12:18:38	9	20	13:00:15	31600	5th & k st nw	31601	19th & east capitol st se	w00812	member
	12:19:46	9	20	12:44:34	31703	minnesota ave metro/does	31603	1st & m st ne	w00803	member

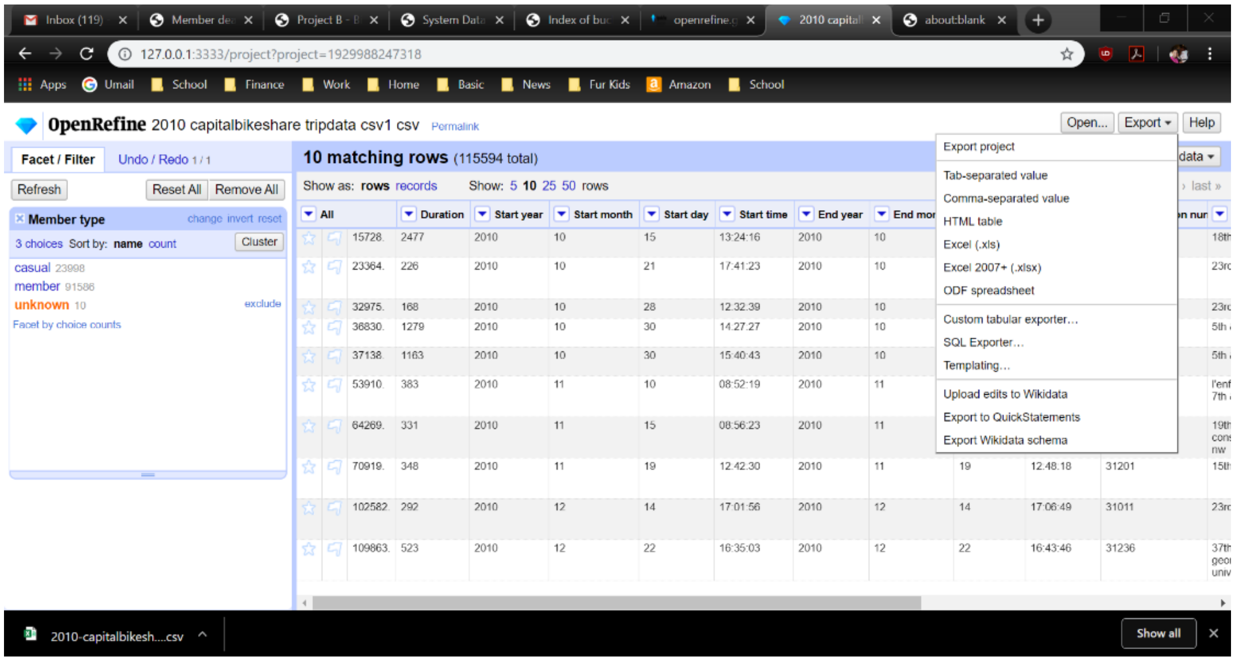
As a final check for inconsistent entries in the text fields Start station, End station, and Member type, I used the “Facet -> Text facet” feature from the drop down menu of those columns, then selected “Cluster”. After browsing the heuristic function results in both the “key collision” and “nearest neighbor” methods, I concluded that there were no values in those columns that needed to be merged to tidy the data.


This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more…

Method: key collision Keying Function: fingerprint **106** clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
1	463	15th & Crystal Dr (463 rows)	☐	15th & Crystal Dr
1	1281	14th St & New York Ave NW (1281 rows)	☐	14th St & New York Ave NW
1	3749	15th & P St NW (3749 rows)	☐	15th & P St NW
1	850	37th & O St NW / Georgetown University (850 rows)	☐	37th & O St NW / Georgetown Un
1	1746	5th & F St NW (1746 rows)	☐	5th & F St NW
1	1554	Columbus Circle / Union Station (1554 rows)	☐	Columbus Circle / Union Station
1	1125	14th St & Spring Rd NW (1125 rows)	☐	14th St & Spring Rd NW
1	146	1st & N St SE (146 rows)	☐	1st & N St SE
1	234	Aurora Hills Community Ctr/18th & Hayes St (234 rows)	☐	Aurora Hills Community Ctr/18th &

Finally, the main cleaning of the data was complete, the data had been processed, and it could be exported using “Export -> Comma-separated value”.



127.0.0.1:3333/project?project=1929988247318

OpenRefine 2010 capitalbikeshare tripdata csv1 csv Permalink

Member type
3 choices Sort by: name count
casual 23998
member 91586
unknown 10

10 matching rows (115594 total)

Show as: rows records Show: 5 10 25 50 rows

All	Duration	Start year	Start month	Start day	Start time	End year	End mor			n nur	
15728.	2477	2010	10	15	13:24:16	2010	10				18th
23364.	226	2010	10	21	17:41:23	2010	10				23r
32975.	168	2010	10	28	12:32:39	2010	10				23r
36830.	1279	2010	10	30	14:27:27	2010	10				5th
37138.	1163	2010	10	30	15:40:43	2010	10				5th
53910.	383	2010	11	10	08:52:19	2010	11				l'enf 7th
64269.	331	2010	11	15	08:56:23	2010	11				19th cons nw
70919.	348	2010	11	19	12:42:30	2010	11	19	12:48:18	31201	15th
102582.	292	2010	12	14	17:01:56	2010	12	14	17:06:49	31011	23r
109863.	523	2010	12	22	16:35:03	2010	12	22	16:43:46	31236	37th geo univ

The next step in the data pipeline was to analyze the now-cleaned dataset. For this, I used RStudio and wrote code in an R markdown file. I imported the cleaned 2010 Capital Bikeshare dataset into R from the 2010-capitalbikeshare-tripdata-cleaned.csv file, along with useful packages:

```
library(dplyr)

library(tidyr)
library(varhandle)
library(ggplot2)
library(readr)

X2010CBS <- read_csv("~/Classes/Big Data/Project B/2010-capitalbikeshare-tripdata-cleaned.cs
v")
```

I viewed the data in RStudio, ensuring that it had loaded correctly and was in the form I expected.

```
View(X2010CBS)
head(X2010CBS)
```

Next, I provided a summary of the data, utilizing both built in functions and those I wrote myself. The summary() function provided the Minimum, 1st Quartile, Median, Mean, and Maximum for each variable with type “dbl”. It also provided the class type and total number of observations for the variables with class “hms” (hours:minutes:seconds) or “character”.

```
summary(X2010CBS)
```

After viewing these initial descriptive statistics, I decided that the Duration column in units of seconds wasn't very useful for understanding the data at a glance. I created a new column with Duration in minutes called Duration_min.

```
X2010CBS2 <- X2010CBS%>%
  mutate(Duration_min = X2010CBS$Duration/60)
View(X2010CBS2)
```

I then provided basic descriptive statistics of the Duration_min variable by using functions I wrote, my_mode() and basic_stat(). I also found the range of the Duration_min variable.

```
my_mode<-function(v){
  p<-table(v)
  df<-data.frame(p)
  colnames(df)<-c("Variable", "Frequency")
  which.max(df$Frequency)
  df$Actual<-unfactor(df$Variable)
  df$Actual[which.max(df$Frequency)]
}
basic_stat<-function(v){
  a<-c("Mean", "Median", "Mode", "Range", "Variance", "Std Dev")
  b<-c(mean(v, na.rm = T), median(v, na.rm = T), my_mode(v), max(v, na.rm = T)-min(v, na.rm =
T), var(v, na.rm = T), sd(v, na.rm = T))
  df<-data.frame("Measurements of Central Tendency"=a, "Values"=b)
  df
}
basic_stat(X2010CBS2$Duration_min)

range(X2010CBS2$Duration_min)
```

To investigate the frequency that each bike was used, excluding where Bike number was "unknown", I created a new table that provided the number of records for each Bike number. I also calculated the number of different bikes in use.

```
BikeUseDf <- X2010CBS2 %>%
  select(`Bike number`)%>%
  mutate(Bike_use_freq = frequency(`Bike number`))
BikeUseFreqDf <- aggregate(x = BikeUseDf$Bike_use_freq, by = list(BikeUseDf$`Bike number`),
FUN = sum)
colnames(BikeUseFreqDf)<-c("Bike number", "Frequency")
BikeUseFreqDf <- subset(BikeUseFreqDf, BikeUseFreqDf$`Bike number`!="unknown")
View(BikeUseFreqDf)
TotalNumofBikes <- length(BikeUseFreqDf$`Bike number`)
TotalNumofBikes
```

Following that, I looked at the total duration of the time each of the individual bikes was used, excluding where Bike number was “unknown”.

```
BikeUseDurationDf <- X2010CBS2 %>%
  select(`Bike number`, Duration_min)
BikeUseTotalDurDf <- aggregate(x = BikeUseDurationDf$Duration_min, by = list(BikeUseDuratio
nDf$`Bike number`), FUN = sum)
colnames(BikeUseTotalDurDf)<-c("Bike_number", "Total_Duration_of_Use")
BikeUseTotalDurDf <- subset(BikeUseTotalDurDf, BikeUseTotalDurDf$Bike_number!="unknown
")
View(BikeUseTotalDurDf)
basic_stat(BikeUseTotalDurDf$Total_Duration_of_Use)

range(BikeUseTotalDurDf$Total_Duration_of_Use)
```

The next step of my analysis was to visualize the data I had been summarizing and investigating. To begin, I plotted a histogram of Bike Use Frequency showing the number of trips each individual bike was taken on.

```
hist(BikeUseFreqDf$Frequency, freq = T, main = "Frequency at which Bikes were Used", xlab = "
Number of Times a Bike was Used", col = "blue")
```

Next, I plotted the total duration of use for each unique, known bike.

```
ggplot(BikeUseTotalDurDf, aes(x=reorder(Bike_number, Total_Duration_of_Use), y=Total_Dura
tion_of_Use))+geom_bar(stat = "identity", color = "darkred")+xlab("Bikes")+ylab("Total Duratio
n of Use")+ggtitle("Total Duration of Use for Each Bike")
```

Finally, I plotted the total duration all bikes were used by month. Months were referenced by their number: 9 = “September”, 10 = “October”, 11 = “November”, 12 = “December”.

```
ggplot(X2010CBS2, aes(x=`Start month`, y=Duration_min, color=`Start month`))+geom_col()
```

The last step I took while using RStudio was to knit the R markdown file with all outputs into a Word document. This knitted output was copied and pasted into this report as needed.

Results

The initial view of the dataset provided a tab in RStudio of the table (not shown) and this list of the first few observations in the dataset.

```
View(X2010CBS)
head(X2010CBS)

## # A tibble: 6 x 13
##   Duration `Start month` `Start day` `Start time` `End month` `End day`
##      <dbl>         <dbl>       <dbl> <time>             <dbl>     <dbl>
## 1     1012             9          20 11:27:04               9        20
## 2      61           9        20 11:41:22            9       20
## 3     2690           9        20 12:05:37            9       20
## 4     1406           9        20 12:06:05            9       20
## 5     1413           9        20 12:10:43            9       20
## 6      982           9        20 12:14:27            9       20
## # ... with 7 more variables: `End time` <time>, `Start station
## #   number` <dbl>, `Start station` <chr>, `End station number` <dbl>, `End
## #   station` <chr>, `Bike number` <chr>, `Member type` <chr>
```

The dataset contained 115,594 observations of 13 variables: Duration (dbl), Start month (dbl), Start day (dbl), Start time (S3: hms), End month (dbl), End day (dbl), End time (S3: hms), Start station number (dbl), Start station (chr), End station number (dbl), End station (chr), Bike number (chr), and Member type (chr). Below is the output of the summary() function.

```
summary(X2010CBS)

##    Duration      Start month      Start day      Start time
##  Min.   :   60   Min.   : 9.00   Min.   : 1.00   Length:115594
##  1st Qu.:  403   1st Qu.:10.00   1st Qu.: 9.00   Class1:hms
##  Median :  665   Median :11.00   Median :17.00   Class2:difftime
##  Mean   : 1254   Mean   :10.86   Mean   :16.26   Mode  :numeric
##  3rd Qu.: 1120   3rd Qu.:11.00   3rd Qu.:23.00
##  Max.   :85644   Max.   :12.00   Max.   :31.00
##    End month       End day       End time       Start station number
##  Min.   : 9.00   Min.   : 1.00   Length:115594     Min.   :31000
##  1st Qu.:10.00   1st Qu.: 9.00   Class1:hms        1st Qu.:31110
##  Median :11.00   Median :17.00   Class2:difftime   Median :31213
##  Mean   :10.86   Mean   :16.26   Mode  :numeric    Mean   :31266
##  3rd Qu.:11.00   3rd Qu.:23.00                     3rd Qu.:31301
##  Max.   :12.00   Max.   :31.00                     Max.   :31805
##  Start station      End station number End station
##  Length:115594      Min.   :31000      Length:115594
##  Class :character   1st Qu.:31111      Class :character
##  Mode  :character   Median :31214      Mode  :character
##                     Mean   :31268
##                     3rd Qu.:31238
##                     Max.   :31805
##  Bike number        Member type
##  Length:115594      Length:115594
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

The next output from R showed the Measurements of Central Tendency and range of the Duration_min variable.

```
##   Measurements.of.Central.Tendency     Values
## 1                             Mean   20.89508
## 2                           Median   11.08333
## 3                             Mode    6.30000
## 4                            Range 1426.40000
## 5                         Variance 2348.05341
## 6                          Std Dev   48.45672

## [1]    1.0 1427.4
```

The average trip lasted about 21 minutes. The most frequent trip duration was 6.3 minutes. The shortest trip was 1 minute, and the longest was 1427.4 minutes (or 23.79 hours). Excluding "unknown" Bike numbers, there were 938 bikes. Below are the Measurements of Central Tendency and range of the total duration of use for each bike.

```
##   Measurements.of.Central.Tendency       Values
## 1                             Mean 2.546719e+03
## 2                           Median 2.526717e+03
## 3                             Mode 4.916667e+00
## 4                            Range 7.347867e+03
## 5                         Variance 1.592574e+06
## 6                          Std Dev 1.261972e+03

## [1]    4.916667 7352.783333
```

The average bike was used about 2546.7 minutes, or about 42 hours. The bike used the least only had a total trip duration of 4.9 minutes, while the most-used bike had a total trip duration of 7352.8 minutes, or about 123 hours. Since the standard deviation of this variable is 1262.0 minutes, the shortest total use is only 2.0 standard deviations below the mean, while the longest total trip duration is 3.8 standard deviations above the mean.

Figure 1 below is the histogram of bike use frequency.

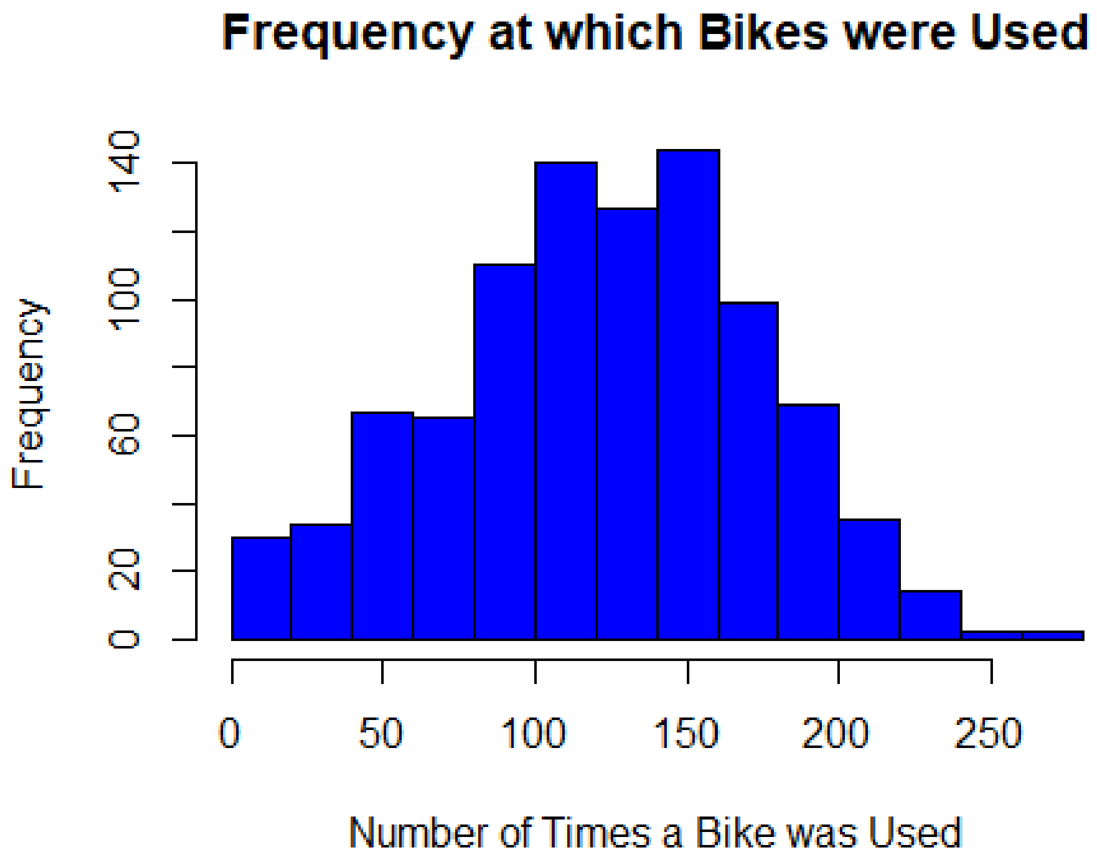


Figure 1.

The second figure is a plot of the total duration of use for each bike.

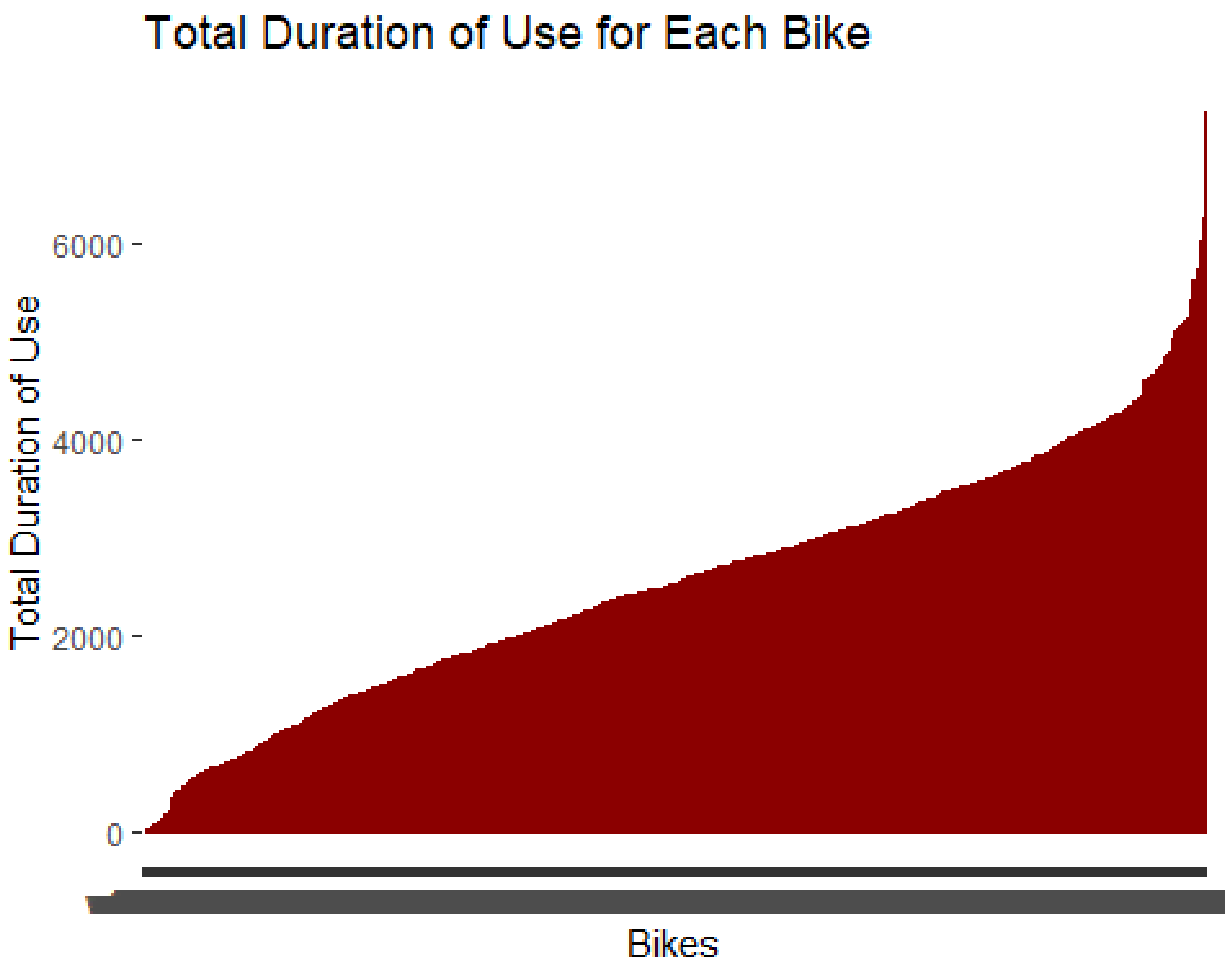


Figure 2.

The final figure below illustrates the total duration of bike use by month.

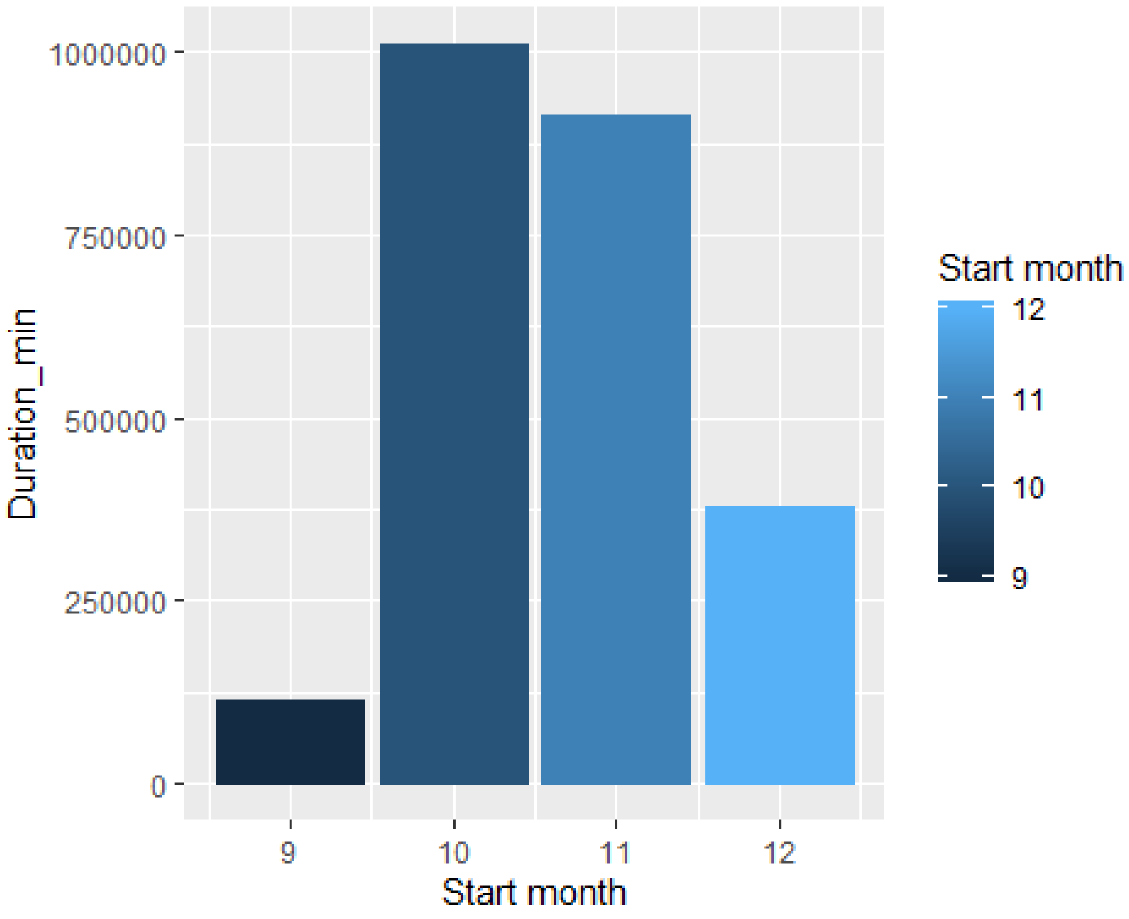


Figure 3.

Discussion

From my analysis of the dataset, I began to draw conclusions about how members used the Capital Bikeshare program during its first quarter of operation. The average member used a bike for 21 minutes each trip, but there was a potential outlier of a 23.79 hour trip. While a 21 minute bike trip across the Washington, DC metro region seems very likely considering the Capital Bikeshare website states that an additional charge is incurred by trips over 30 minutes, the almost 24 hour "trip" may have been a case of a user accidentally forgetting to end their session. Another hypothesis could be that the user wanted to reserve the bike for use later in the day, so they kept the trip going to prevent another member from using the bike. The best way to address this question would be with a member survey that investigated the way they used the bikes and why they chose to end the trip. Regardless of cause, this datapoint is almost 4 standard deviations away from the mean, which is considerable since the minimum is only 2 standard deviations below the mean. It may be skewing the data towards longer duration trips.

When investigating the use of individual bikes, I was able to answer some of my questions, particularly through my use of data visualizations. Figure 1 shows that the frequency at which each bike was used for a trip has a very normal looking distribution. If each bike were being used the same number of times, I would expect a much more uniform distribution. In addition, Figure 2 illustrated the great variation in the total duration each bike was used. The bikes at the far right of Figure 2 (with the greatest total minutes of use) could be targeted for inspection and maintenance by the Capital Bikeshare team since they have likely been ridden the most. On the other end of the chart, the bikes that were least used may also be good targets for inspection. If something is wrong with a bike, then it would likely not be used during a trip, instead being passed over for a functional bike. By using this information generated by my pipeline, the Capital Bikeshare team could potentially be more efficient with the time they dedicate to bike upkeep.

The next interesting analysis I performed considered the total duration all bikes were used across the months of the first quarter of the program. As shown in Figure 3, there was minimal use of the bikes during the first month, September, but it was followed by the month with the greatest total duration of trips across all bikes. As the quarter progressed, the total time the bikes were used each month decreased slightly from October to November and greatly from November to December. It is unclear whether the Capital Bikeshare program began at the start of September or later in the month. This could be a reason for the minimal bike use in the first month. Another reason could be a lack of public knowledge or familiarity with the new system. The spike of use in October could be due to people wanting to "try out" the new program, and they may have lost interest in it afterward. The dip in December could also be due to winter weather. Users may have opted to take other, warmer forms of transportation rather than ride a bike, or they could have taken much shorter trips when they did choose to ride a bike. It would be interesting to use this data pipeline to continue to monitor these trends over time.

As for the technologies and skills I acquired from this course, the two that I used seem to be the most useful in many applications and built nicely on my previous knowledge. This class taught me the importance of a well-developed data pipeline. Although the one I implemented here was simple, with the basic lifecycle steps Plan, Acquire, Process, Analyze, and Share, it gave structure to my data handling which in the end I believe increased the quality of my final product. A little bit of forethought goes a long way in the use and management of data. While paying attention to a data lifecycle and formulating a pipeline before beginning work with a dataset is a very useful general skill, the use of OpenRefine for cleaning data was one of the best specific technologies I was able to learn in this class. I did not know this tool existed, and I believe it to be a very powerful one. I am most comfortable performing data analysis (specifically statistical analysis) in R, but tidying data within that language can be very cumbersome. OpenRefine is user friendly and makes once tedious tasks much faster to complete. Since it can easily import and export datasets in a wide variety of formats, it makes going from the prior step in the pipeline and moving on to the next step even easier since transformations are so simple. In addition, its capability to hand big data is only limited by your own computer's memory. While there were other technologies and skills I learned in the class that I found useful and will continue to explore, formulating a data pipeline and using OpenRefine enabled me to answer the questions I had about the data without encountering any great difficulty.

Conclusion

Through implementation of a simple pipeline and data cleaning technology, valuable investigation of datasets can be accomplished. These skills and tools were used on the Capital Bikeshare dataset to answer various questions about how members used the bikes, how use varied across bikes, and how use varied over time. The developed data pipeline could be easily employed and adjusted to answer these and more questions as the program continues. The utilized skills and technologies learned from I535 were well suited for this use, and they will continue to be effective in the future.

References

1) General information on Capital Bikeshare: https://ddot.dc.gov/capitalbikeshare
2) Capital Bikeshare Dataset: https://www.capitalbikeshare.com/system-data
3) OpenRefine information and tutorials: https://openrefine.org/
4) Aggregating the frequency of bike use: https://stackoverflow.com/questions/10202480/aggregate-rows-by-shared-values-in-a-variable
5) Plotting a bar chart: https://ggplot2.tidyverse.org/reference/geom_bar.html
6) Sorting the bars in a bar chart: https://sebastiansauer.github.io/ordering-bars/
7) Coloring ggplot by variable: http://www.sthda.com/english/wiki/ggplot2-colors-how-to-change-colors-automatically-and-manually