**Ford BikeGo Bicycle Sharing Analysis**

**Monica De La Torre, Kent Lum, Jason Zelaya**

**Core Audience:**

The groups that would benefit greatly from our analysis are the companies who are interested in entering the bicycle sharing market. Our analysis provides information on the categories of people who are not taking advantage of the bicycle sharing platforms.

**Hypothesis:**

Males under 30 years old will have the highest number of users.

Areas of high traffic congestion will demographically have the most usage.

Beneficial target category would be females.

**Core Message:**

Bike sharing is the solution to prevent the anxiety and emotional stress caused by the “last mile” experience.

**Questions:**

The questions this analysis focused on were the following:  
1) What are the demographics of the individuals who use Ford GoBike’s service?  
2) Who should Ford GoBike target for growth areas?

3) Are there any potential areas of growth?

**Motivation:**

Virtually everyone has experienced the stress caused by the “last mile” before, and the three of us had considered trying out a bicycle sharing platform at one point. We decided that this would be the most efficient way of learning about the pros and cons of a bicycle sharing service than if we did simple research on our own, and at the same time we would be able to share our potential solution with others to prevent future “last mile” anxiety for them as well.

**How We Found Our Data**To find the most ideal dataset for our analysis we did a google search of bicycle sharing platforms around the nation. After going over the information each database covered, as well as the quantity of information provided, we determined that Ford GoBike’s database would enable us to get the most accurate results. The data was retrieved from <https://s3.amazonaws.com/fordgobike-data/index.html>.

**Data Exploration**For our data exploration we both read and opened a database to confirm that there 16 columns of data for us to use with each month of 2018 available. The amount of data was more than sufficient as well; roughly 2 million rows of data was at our disposal after merging all 12 months since they were separated by month in zip files. Furthermore, the data was for multiple cities rather than just one like many of the databases we found had. As a bonus, since the data was gathered from the Bay Area the analysis and results would apply to us.

The cleanup process involved merging all 12 sets of data, adding columns, removing columns, and filtering out misleading or useless data.

The first step was to create a path to all 12 zipped sets of data and merge them by utilizing the “glob” dependency.

The second step was to customize the dataframe for our analysis. We achieved this by adding the columns “city”, “zip\_code”, and “age.” We also removed the column “bike\_share\_for\_all\_trip” since it was not applicable to answering our questions.

**Analysis Process Explanation**

We started by adding values to the blank columns we created in the data frame.

To calculate the age of each rider we assigned the variable “thisyear” to “2019.” We then assigned the age column of the data frame that we named “clean\_df” to a mathematical operation using “thisyear” and subtracting the values in the “member\_birth\_year” column. To ensure compatibility we converted both columns used to integers using “.astype(int).” To confirm everything processed the way we wanted it to we took a look at the first few rows using “.head().” We previewed every change in our dataframes with “.head().”

For the “age” column we decided to exclude everyone over the age of 65 since people tend to become more inactive the older they get. To do this we created two variables and assigned “max\_age” to 65 and max\_age\_df to the “age” column in “clean\_df” and used a comparison operator, “<,” to return only the riders were were 65 years old and younger.

To determine the number of each gender out of clean\_df we simply used .groupby() on the “member\_gender” column and used “.size()” to return the count of each gender.

We then created a pie chart to display the percentage of riders by gender. Males (73.3%) far out-weighed female (25.1%) and other (1.5%). We used explode to make it easier to view the “other” slice since it was so narrow. We also made sure to use the traditional “lightcoral,” “lightskyblue,” and “gold” colors for our graphs.

To obtain the count of each gender for each user type, customers and subscribers, we created a new data frame called “cus\_df” to contain the “user\_type” and “member\_gender” columns that we acquired using .groupby(). To retrieve the count we used “.size()” again.

Once again we chose a pie chart for our visual. The male subscribers (65.71%) were the largest category and tripled the second largest category. The second largest was female subscribers (21.47%), followed by male customers (7.63%). The other three categories, female customer, male customer, and other subscriber amounted to slightly over 5%.

After creating the pie chart we set our sights on grouping the number of users by age. We used the bins method to bin the data by age groups and a list to provide the age ranges. We continued to bin our data by using pd.cut() on “age\_grp” and “age\_bracket” columns from our database. We then used group by to find the count of the age group which was grouped by the “age\_grp” and “user\_type” columns.

We decided to use a bar chart

Our next goal was to find zip codes using latitude and longitude coordinates. This was tricky because the function returned a list of SimpleZipcode() functions with 24 parameters, and each parameter included the variable as well as the variable’s assigned value within the same parameter.

To start off, we used set a variable, “coordinates,” to a “search.by\_coordinates() function using the uszipcode dependency to print out the first element so we could determine where the value of “zipcode” was located. After taking note of its position we assigned the variable “coordinates\_list” to an empty list. We also set a variable, “clean\_small,” to the first 50 rows of “clean\_df” so we could run the code. Since our data frame had roughly 1.7 million rows after cleaning the original data frame it would have taken several hours to finish the loop before we could move on, We zipped the columns “start\_station\_latitude” and “start\_station\_longitude” and passed “lat, lng” through them using a for loop. We used printed out “lat, lng” to confirm that it was looping properly. We then used the “search\_by\_coordinates()” function again and set the parameter “returns” to 0 so it would print out all of the applicable latitude and longitude coordinates from the data frame. Finally, we ended the loop by appending each pair of coordinates into a list.

Now that we had our list of tuples we could begin making the value of “zipcode” accessible. To do this we used a list comprehension to unpack the rest of the values in “coordinates.” We used zipcode = [coordinates\_list[i][0].items()[0][1] for i, x in enumerate(coordinates\_list)]. [i] allowed us to set an index to “coordinates\_list” so it would become an iterable. Now that we can use [num] to select values, we used the first used .items() on the first [0] so we could separate the variables from their values in the form of tuples. The second [0] selected the first tuple, and the [1] selected the value of zipcode

Next, we used the zipcodes we just obtained to find their corresponding cities. We set “city\_by\_zip\_list” to an empty list and used a for loop to enumerate() through the “zip\_code” column. We passed an additional value, “i,” in the for loop to create indices so we could run enumerate on a single argument. We then looked for the zipcode using the “seach\_by\_zipcode()” function.

The last column we calculated using zipcodes is the “num\_stations\_by\_zip.” To start we created a separate data frame with the columns “zip\_code,” “start\_station\_id,” and “end\_station\_id.” We then used .groupby() on the “zip\_code” column and running .size() and .reset\_index() on it. .size() returned the total number of elements while .reset\_index() recalculated the index for each row.

Next up is finding the number of rides per month. To do this we had to use the “datetime” dependency and convert “clean\_df” to “datetime.” The next step is to add a “month” column. Finally, we used .groupby() to run “.start\_time.count() to count each ride grouped by months.

Finally, we wanted to find the number of rides each gender took every month. To do this we created three separate data frames; one for female, one for male and one for other. We followed the same method as the block of code before and used groupby() to run the functions through. We ran start\_time.count() to get the number of rides for each gender categorized by month.

**Graphs:**Scatter Plot: The trend is that the zip codes with the most usage are in congested areas. San Francisco, Berkeley and Oakland had the largest bubbles. Related to our hypothesis of bike sharing utilization being higher in beach areas, San Mateo also had some of the larger bubbles corresponding to the zip codes near the beach.

Pie Chart for Gender: The trend is that males, regardless of categories of comparison, use Ford GoBike’s bicycle sharing platform more than females.

Bar chart: The trend is that the millennials are utilizing Ford GoBike’s bicycle sharing service out of the our included age groups.. If you add up the 35-60 year olds, they are roughly the same quantity as the millennial subscribers.

Subscriber vs Customer: Without a doubt subscribers use Ford GoBike’s platform more than customers. A whopping 87% of utilization is from subscribers compared to the meager 13% for customers.

**Numerical Summary:**-Males represent the highest number of consumers comprising 73.3% of the users.  
-Females second highest consumers comprising 25.1% of the users.  
-Everyone else combined is about represents roughly 5% of users.

-October is the peak season with 191,485 rides.  
-January is the lowest with 86,963 rides.