Introduction

This notebook contains the final dataset and all data cleaning and preprocessing steps. After data cleaning, it shows

Using this notebook

You will need to use the data for development, not production (smaller dataset) as the production dataframe is too la located in the `lyft_no_data` folder (the project's repository folder. Change the directories in the `pd.read_csv` calls t

Table of contents

Initial Setup

```
import pandas as pd, numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
# Read in the dataset
dev_df = pd.read_csv('/content/SF_DF.csv')

# joining the regions df so that I can filter by SF region

regions = pd.read_csv('/content/region_ids.csv')

dev_df = pd.merge(dev_df, regions, 'inner', left_on='start_station_id', right_on='station_id'
```

Data Cleaning / Preprocessing

```
# first things first, filter the master df down to SF region to simplify things:

dev_df = dev_df[dev_df['region_id'] == 3]

# Convert times to datetime
```

```
dev_df['start_time'] = pd.to_datetime(dev_df['start_time'])
dev_df['end_time'] = pd.to_datetime(dev_df['end_time'])

# Get the day of the rides too
dev_df['day_of_ride'] = dev_df['start_time'].apply(lambda x: x.day)


# Remove member_birth_year
dev_df.drop('member_birth_year', axis=1, inplace=True)


# Remove the 31st day since it's not in every month
dev_df = dev_df[dev_df['day_of_ride'] < 31]
```

EDA

The purpose of this section is to get started on the EDA unit of DSC on my dev df. I still need to crunch the whole df, b at least I'll know what plots I will want.

General questions to explore

Some questions I'll be exploring in this notebook in turn include:

1. What is the general distribution of variables? When I see this, are there any more outliers to account for?
2. Looking at bivariate data, what are the distributions of age by gender for example?
3. What is the correlation between cost per ride and duration of rides? How are costs and durations distributed?
4. Is there patterning in the bikes that are most vs. least profitable?
5. Is there a relationship between cost, ride length, and gender? User type?
6. How can I visualize the data on maps?

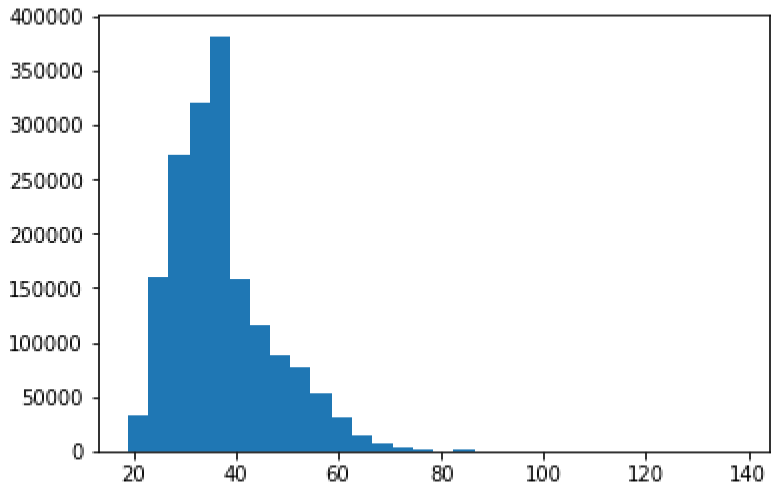
Questions to analyze further and business cases for analyzing them:

1. Given a departure station, what is the revenue from the individual ride? Reason: to learn what the most profitab supply, etc. at that/those stations.
2. Given a rider's age and gender, how likely are they to do an above average value ride. Reason: target marketing demographic to increase revenue.
3. What is the most popular time of day and day of week to ride the bikes? Reason: campaigns and push notificat them to take a bike. After this, you can do an A/B test to measure the efficacy of those efforts (i.e. did ridership to random chance or not [statistical test]?)

1. What is the general distribution of variables? When I see this, are there any more outliers to account for?

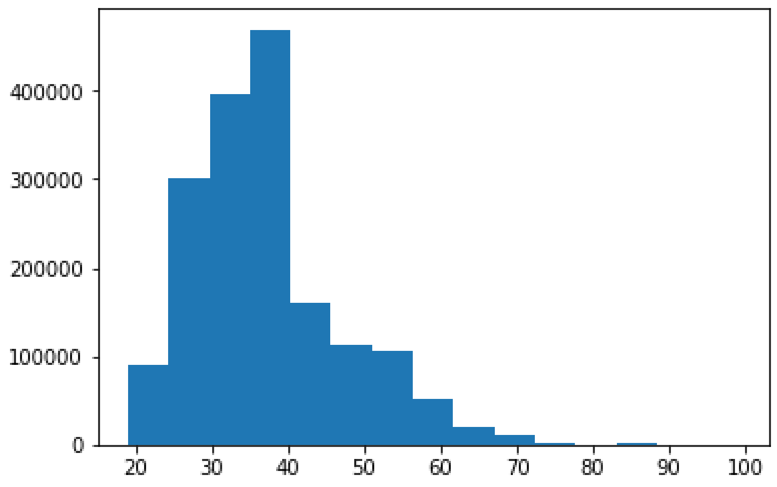
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
plt.hist(dev_df['age'], bins=30);
```

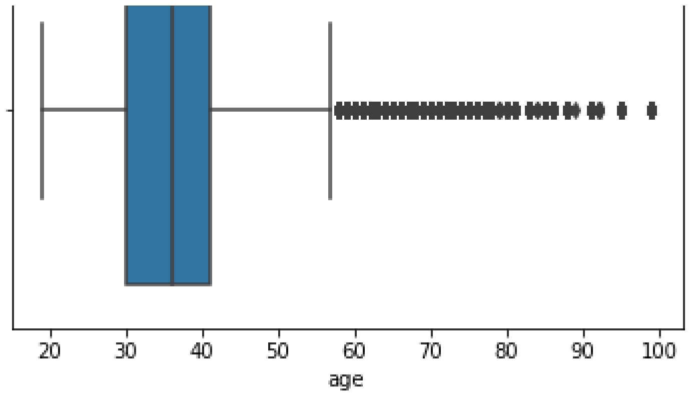


Based on the histogram above, I've decided to remove values >= 100

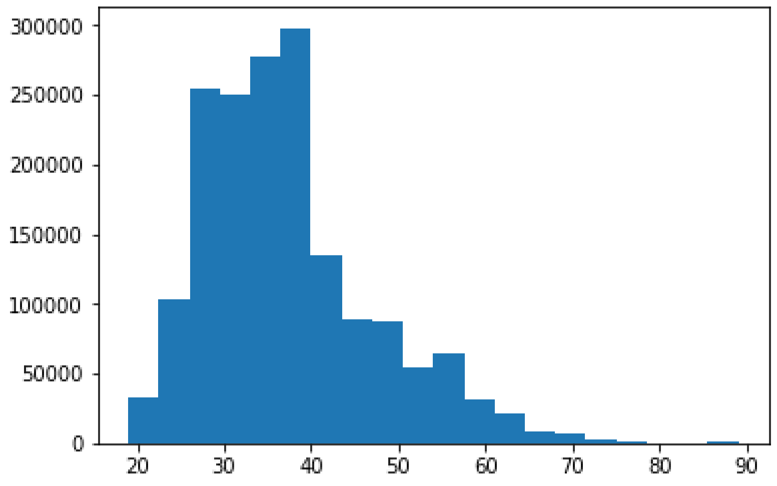
```
# simplify the name of the df
dev_df = dev_df[dev_df['age'] < 100]

import matplotlib.pyplot as plt
%matplotlib inline

age = dev_df['age']

plt.hist(x=age, bins = 15);
```



```
import seaborn as sns

sns.boxplot(age);
```



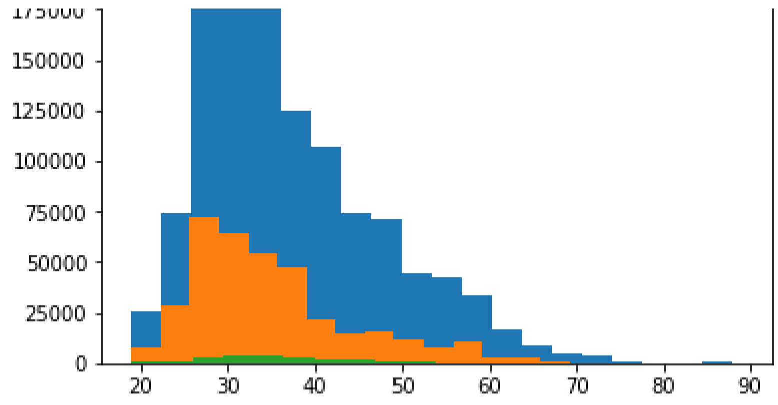
```
dev_df = dev_df[dev_df['age'] < 90]

age = dev_df['age']

plt.hist(x=age, bins = 20);
plt.show()
```



Honestly I could cut this down to a max age of 75, it doesn't really matter - bikesharing is such a milennial thing.

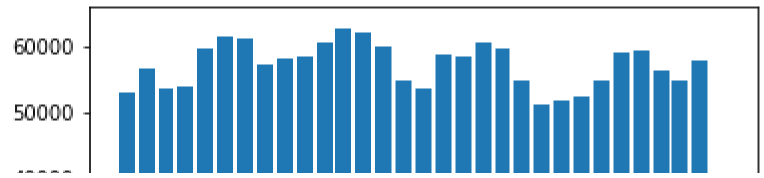
2. Looking at bivariate data, what are the distributions of age by gender for example?

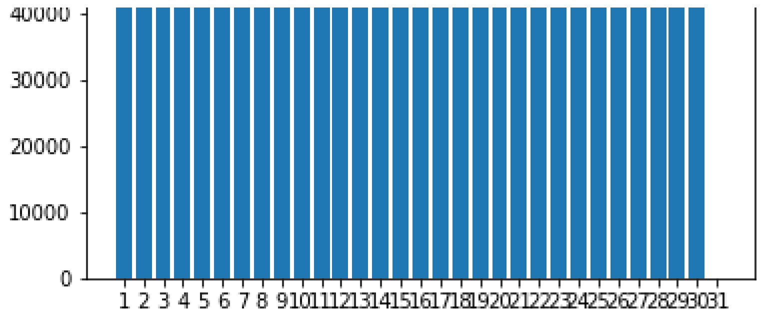
```
# take a look at the distribution by gender below

men = dev_df[dev_df['member_gender'] == 0.0]['age']
women = dev_df[dev_df['member_gender'] == 1.0]['age']
other = dev_df[dev_df['member_gender'] == 2.0]['age']

plt.hist(x=men, bins = 20)
plt.hist(x=women, bins = 20)
plt.hist(x=other, bins = 20)
plt.show();
```



3. What is the correlation between cost per ride and duration of rides? How are costs and durations distributed?

```
mins = dev_df['duration_min']

dev_df['start_time'] = pd.to_datetime(dev_df['start_time'])
dev_df['end_time'] = pd.to_datetime(dev_df['end_time'])

start_hours = dev_df['start_time'].apply(lambda x: x.hour)

end_hours = dev_df['end_time'].apply(lambda x: x.hour)

days = dev_df['end_time'].apply(lambda x: x.day)

days_list = days.value_counts(ascending=True).index.tolist()

counts = days.value_counts(ascending=True).tolist()

d= {'days':days_list,
    'counts':counts}

day_counts = pd.DataFrame(d)

day_counts = day_counts.sort_values(['days'])


plt.bar(x=day_counts.days, height=day_counts.counts);
plt.xticks(day_counts.days.to_list());
```

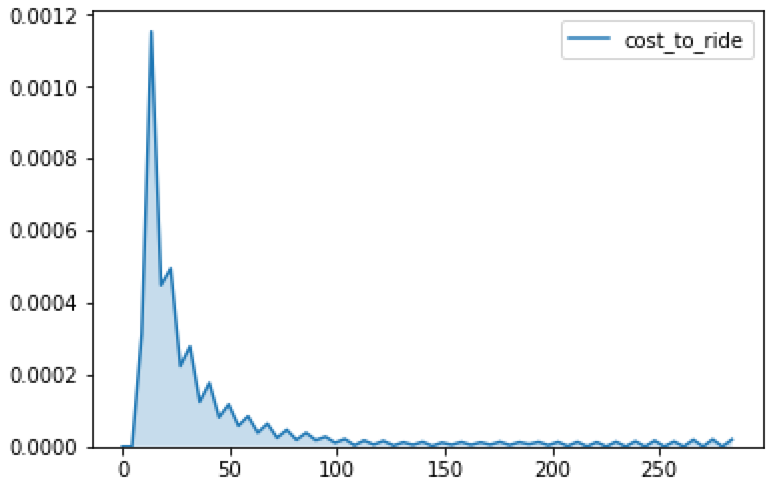




There's an interesting cyclic pattern in that the peaks are around the beginning of weeks, with higher peaks progressi
rides being at the end.

```
sns.kdeplot(dev_df['cost_to_ride'], shade=True, gridsize=40)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66ade22dd8>
```



The cost of rides is left skewed too and most rides are under 50 or even 10 dollars.

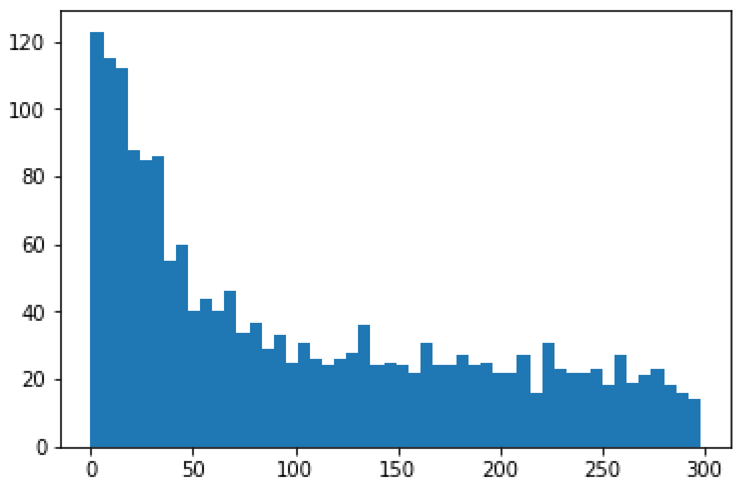
5. Is there patterning in the bikes that are most vs. least profitable?

```
# pivot and melt:
bike_revenue = dev_df.pivot_table(values='cost_to_ride', columns='bike_id', aggfunc='sum').me

# sorted best performer to least performer
bike_revenue = bike_revenue.sort_values('value', ascending=False)
bike_revenue[:10]
```

	bike_id	value
869	1112	2044
2320	2852	2038

2638	3235	2026
52	73	1962
2297	2829	1922
2235	2766	1878
2029	2543	1868
978	1298	1866
2425	2970	1864
442	579	1835

```
# There are not many bikes that make more than $300 so filtering down here

bike_revenue = bike_revenue[bike_revenue['value'] < 300]

# distribution of value by bike
plt.hist(bike_revenue.value, bins=50);
```



```
#bike 3328 has accrued the most value

bike_revenue[bike_revenue['value'] > 250][:10].sort_values('value', ascending=False)
```

	bike_id	value
3328	4424	298
1781	2264	298

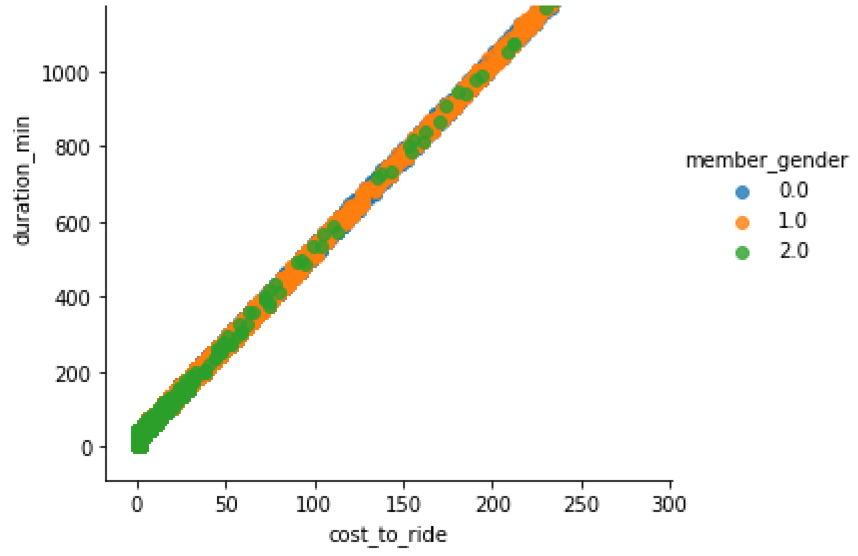
3118	4000	298
239	288	296
737	932	296
727	918	296
1549	2015	296
1471	1933	296
3220	4305	295
2076	2591	295

More data viz and mapping

6. Is there a relationship between cost, ride length, and gender? User type?

```
men = dev_df[dev_df['member_gender'] == 0.0]
women = dev_df[dev_df['member_gender'] == 1.0]
other = dev_df[dev_df['member_gender'] == 2.0]


# plotting cost and duration colored by gender
sns.lmplot(x='cost_to_ride', y='duration_min', hue='member_gender', data=dev_df, fit_reg=Fals
```

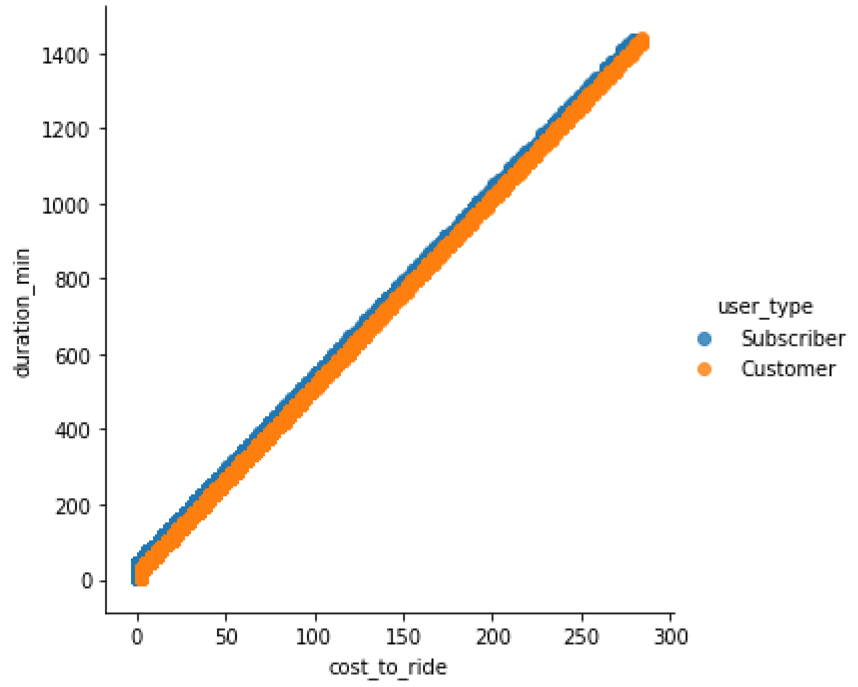
```
<seaborn.axisgrid.FacetGrid at 0x7f66975ae6d8>
```



Naturally duration of ride and cost are very correlated, but interestingly we see a bunching of lower cost and shorter r
respectively).

```
# cost and duration colored by user type
sns.lmplot(x='cost_to_ride', y='duration_min', hue='user_type', data=dev_df, fit_reg=False)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f6696c75e10>
```



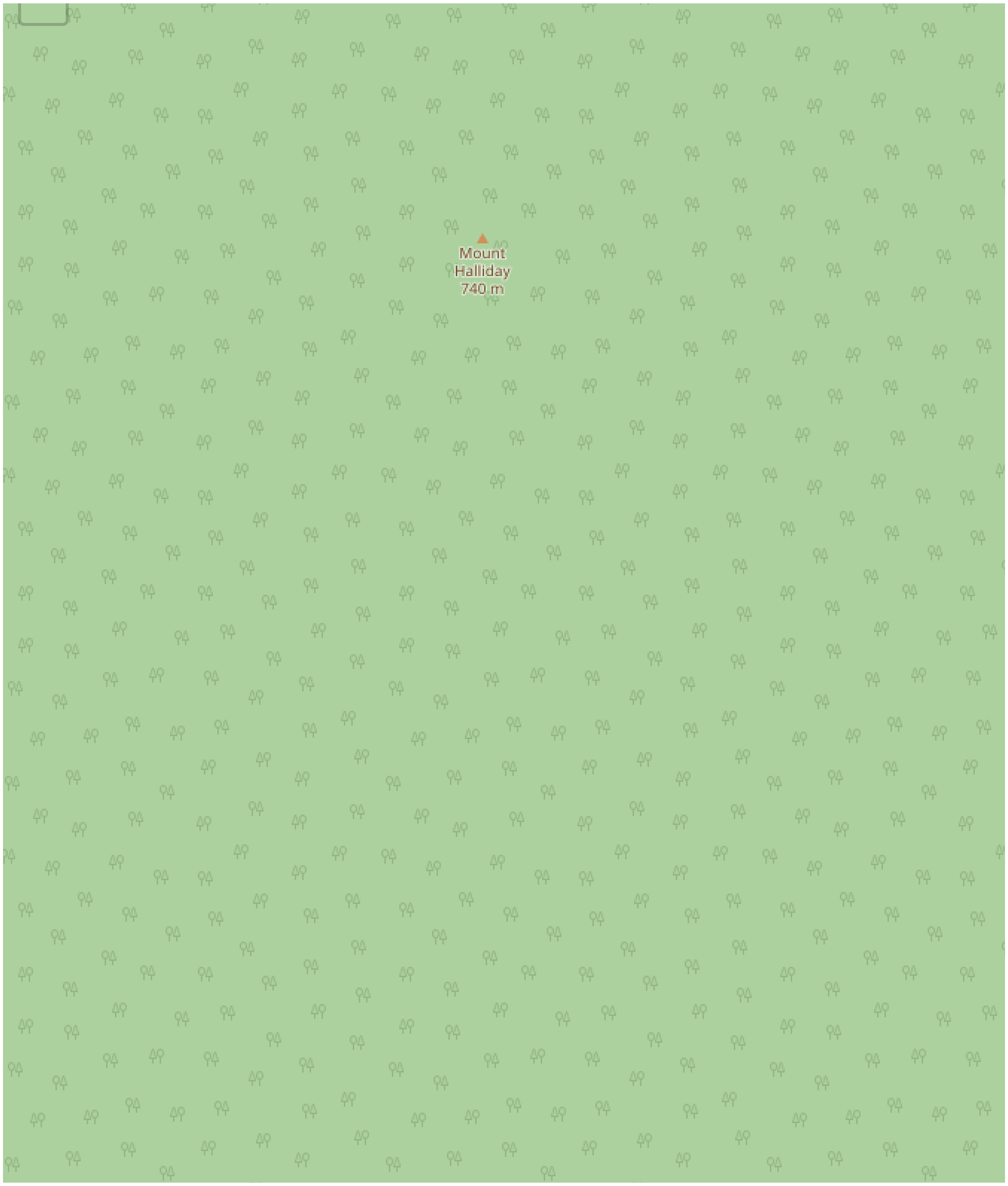
It doesn't seem like there are any differecnces really between subscribers and customers.

7. Map visualizations

```
import folium # Use the Folium Javascript Map Library
import folium.plugins
```

```
# for some reason folium doesn't work for more than 66000 data points, so I will sample for r
sample = dev_df[:4000]

SF_COORDINATES = (37.77, -122.35)
sf_map = folium.Map(location=SF_COORDINATES, zoom_start=11)
locs = sample[['end_station_latitude', 'end_station_longitude']].astype('float').dropna().as_
heatmap = folium.plugins.HeatMap(locs.tolist(), radius = 10)
sf_map.add_child(heatmap)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: FutureWarning: Method .a
  if __name__ == '__main__':
```



Here's the locations of the end stations for each ride.

Stats Tests

The purpose of this section is to look into statistical significance, hypothesis tests, and tests to verify that the empiric
theoretical distributions I see these variables as.

I'll perform the following steps:

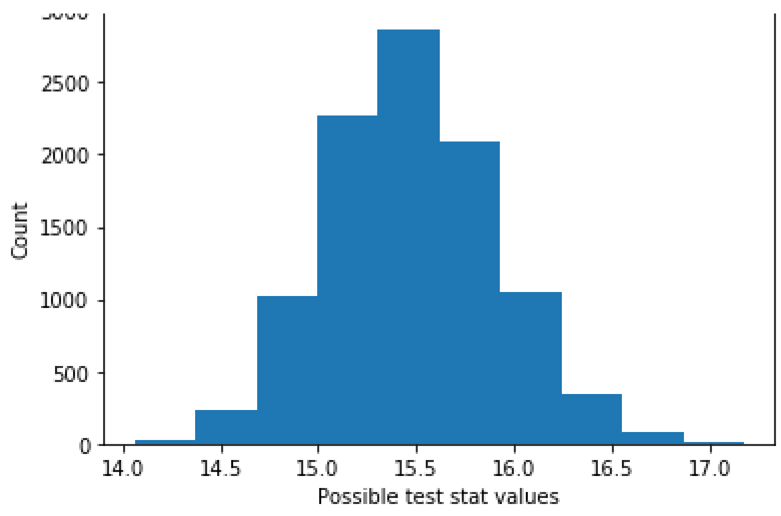
1. A little more EDA as needed and data cleaning that results from that
2. Joining the region IDs and filtering down as I've done for the master DF
3. Bootstrap resampling on various discrete and continuous variables to see that my samples reasonably came f
4. Bayesian tests as necessary.

```
# Get revenue by station

station_revenue = dev_df.pivot_table(values='cost_to_ride', columns='start_station_id', aggfu

station_revenue['total_value_of_station'] = station_revenue['value']

station_revenue.drop('value', axis=1, inplace=True)

dev_df = dev_df.merge(station_revenue, 'inner', left_on='start_station_id', right_on='total_v
```

```
from sklearn.utils import resample

# making necessary bootstrap functions to use over and over

def bootstrap_sample(var, func=np.mean, repetitions=10000, conf_interval=[2.5, 97.5]):
    test_stat = func(var)

    var_name = str(var)
    func_name = str(func)

    repetitions = repetitions

    variables = np.empty(repetitions)

    for i in range(repetitions):
        variables[i] = func(resample(var, n_samples = repetitions))

    lower_conf, upper_conf = np.percentile(variables, conf_interval)

    return print(f"Test stat = {test_stat}"), print('\n'), print(f"Confidence interval is bet
```

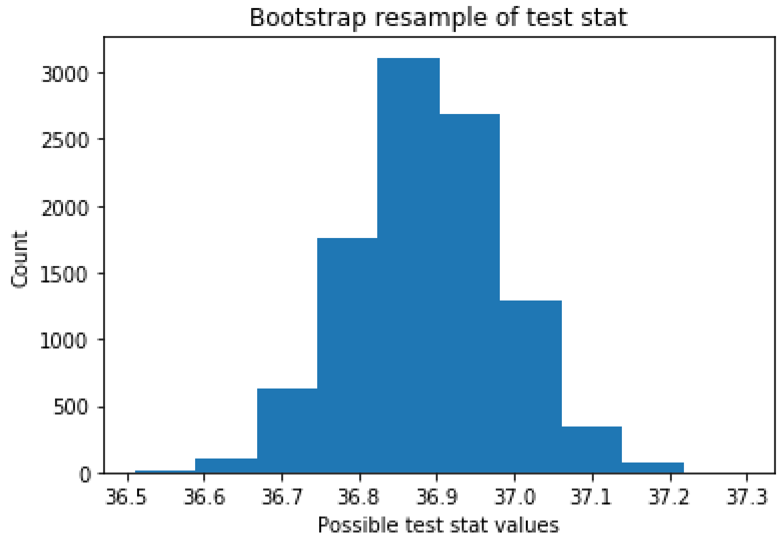
```
# duration of rides
bootstrap_sample(dev_df.duration_min, repetitions=10000);
```

```
Test stat = 15.476606254444144


Confidence interval is between 14.68185 and 16.36982
```



```
# mean age
bootstrap_sample(dev_df.age);
```

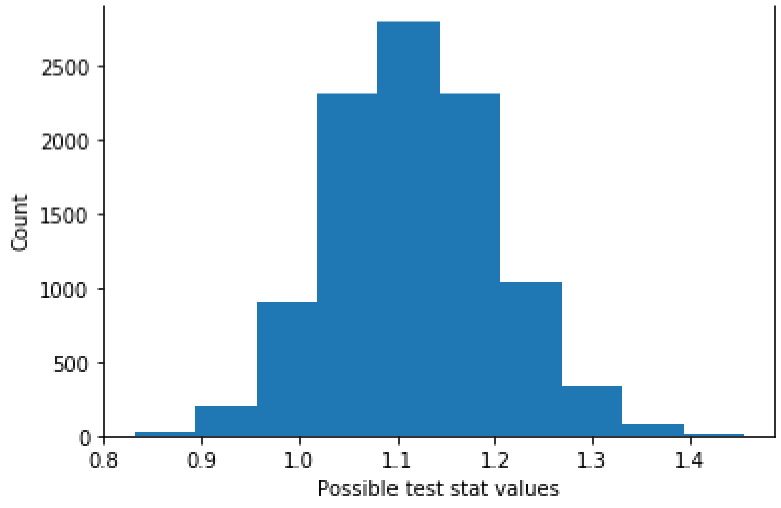
```
Test stat = 36.89112405635627


Confidence interval is between 36.6991975 and 37.0863
```



```
bootstrap_sample(dev_df.cost_to_ride);
```

```
Test stat = 1.1180562875513862


Confidence interval is between 0.9602975 and 1.2889
```

Bootstrap resample of test stat



Bootstrap Resampling of standard deviation differences

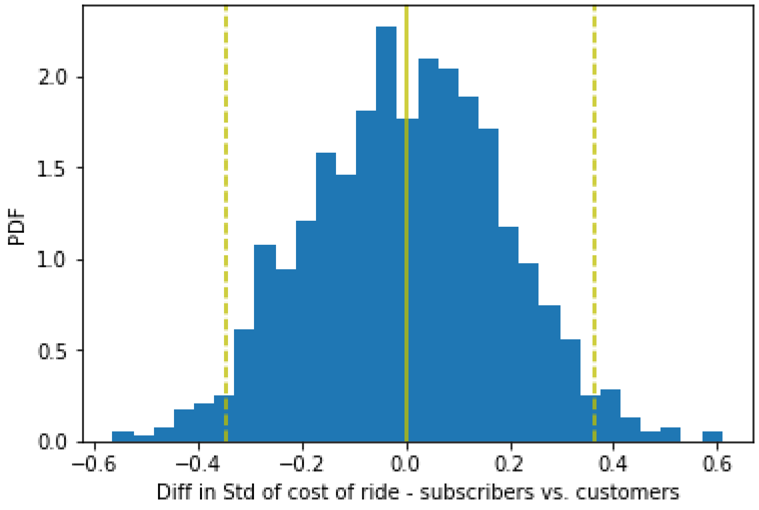
Checking the difference between subscriber and customer cost of ride standard deviations

```
def std_diff(data_1, data_2):
    """Difference in means of two arrays."""

    # The difference of std of data_1, data_2: diff
    diff = np.std(data_1) - np.std(data_2)

    return diff

def permute_sample(data1, data2):
    """Generate a permutation sample from two data sets."""

    # Concatenate the data sets: data
    data = np.concatenate((data1, data2)) # tuple

    # Permute the concatenated array: permuted_data
    permuted_data = np.random.permutation(data)

    # Split the permuted array into two: perm_sample_1, perm_sample_2
    perm_samp_1 = permuted_data[:len(data1)]
    perm_samp_2 = permuted_data[len(data1):]

    return perm_samp_1, perm_samp_2

def draw_perm_reps(data_1, data_2, func, size=1):
    """Generate multiple permutation replicates."""

    # Initialize array of replicates: perm_replicates
    perm_replicates = np.empty(size)

    for i in range(size):
        # Generate permutation sample
        perm_samp_1, perm_samp_2 = permute_sample(data_1, data_2)

        # Compute the test statistic
        perm_replicates[i] = func(perm_samp_1, perm_samp_2)

    return perm_replicates


# split the data up by subscriber and customer
subscribers = dev_df[dev_df['user_type'] == 'Subscriber'].cost_to_ride
customers = dev_df[dev_df['user_type'] == 'Customer'].cost_to_ride

subscribers_std = np.std(subscribers)
customers_std = np.std(customers)

print(f'std for subscribers = {subscribers_std}, std for customers = {customers_std}')
```

```
std for subscribers = 4.130785981749726, std for customers = 17.252532376078598
```

```
N_rep = 1000

bs_sample_std_diff = np.empty(N_rep)

# Generate bootstrap sample: bs_sample
bs_sample_std_diff = draw_perm_reps(subscribers, customers, std_diff, N_rep)


import matplotlib.pyplot as plt
%matplotlib inline

bs_sample_std_diff = np.sort(bs_sample_std_diff)
bs_sample_std_mean = np.mean(bs_sample_std_diff)

bs_sample_std_perc_95 = np.percentile(bs_sample_std_diff, [2.5, 97.5])
print('Standard deviation mean diff: ' + str(bs_sample_std_mean))
print('95% percentile: lower ' + str(bs_sample_std_perc_95[0]) + ', upper ' + str(bs_sample_s
_ = plt.hist(bs_sample_std_diff, bins=30, normed=True)
_ = plt.xlabel('Diff in Std of cost of ride - subscribers vs. customers')
_ = plt.ylabel('PDF')
_ = plt.axvline(bs_sample_std_mean, color='y')
_ = plt.axvline(bs_sample_std_perc_95[0], color='y', linestyle='--')
_ = plt.axvline(bs_sample_std_perc_95[1], color='y', linestyle='--')
plt.show()
```

```
Standard deviation mean diff: 0.003166955446961012
95% percentile: lower -0.34542312208700393, upper 0.36356371897281875
/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py:6521: MatplotlibDeprecat
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'der
  alternative="'density'", removal="3.1")
```



H0 = there is a difference between the standard deviations of cost to ride for customers vs. subscribers H1 = there is
cost to ride for customers vs. subscribers

95% confidence interval is -7 to 7.97 standard deviation, so we lack sufficient evidence to state H0 that there is a diffe
meaning that the mean standard deviation of the cost to ride for customers and subscribers is within the confidence
observed by random chance.

```
# this is for rides by day

import numpy as np; np.random.seed(sum(map(ord, 'calmap')))
import pandas as pd
import calmap

all_days = pd.date_range('1/15/2014', periods=700, freq='D')
days = np.random.choice(all_days, 500)
events = pd.Series(np.random.randn(len(days)), index=days)

# in order to use this you need to first create an object similar to events above.
# it's a series with index of whatever datetime level you want
# so if you want it to be number of rides by day for example you need to group by lambda x:

calmap.yearplot(events, year=2015)
```