assignment2

February 17, 2020

```
In [172]: import pandas as pd
          import numpy as np
          import seaborn as sns
          import matplotlib.pyplot as plt
          import geopandas as gpd
```

```
In [203]: %matplotlib inline
          plt.rcParams['figure.figsize'] = (15, 10)
```

0.1 Part 1

1 1.

In order to adress the problem at hand, it is important that the quality of our data is up to a certain standard for the purpose of drawing reasonable conclusions. In our police stops data we are concerned with some of the following attributes of police stops in order to adress the problem of unfair policing:

- `service_area`
- `subject_race`
- `subject_age`
- `date_time`

In order to perform any sort of analysis we will first need to assess the validity of the aforementioned attributes and make decisions regarding uncertainty in the measurements. Let us first adress the missingness of our data as a whole.

It became clear the the attributes `obtained_consent`, `property_seized`, and `contraband_found` are a majority `NaN` which makes sense. It is rare that an officer finds the need to search a citizen's vehicle as a result of some sort of equiptment or moving violation. Therefore, these attributes have been omitted from the analysis of missingness in the data.

Next, we see somewhat high rates of missingness in the residency status of those who were stopped by police but more important than that is the missingness of the `subject_age` attribute in our dataset. While not high at about 2.5% this is much larger than the relevant attrbutes listed above. In addition to some of the `subject_age` data being missing the accuracy and precision of the reported age of these citizens is fairly questionable. We expect most people on the road to be of ages anywhere as low as 15 and anywhere as high as 100. These are failry reasonable assumptions, especially for San Diego. However, in our dataset we see ages as low as 0 and as high as 2,005. We

also see some ages just above what we considered to be reasonable in our dataset which could be used as codes for something else in our dataset, perhaps. As for precisionof `subject_age` we see a majority of the subject's ages falling on every fifth year (e.g. 20, 25, 30, 35, etc...). The reporting that the officers are doing provide rough estimates but may introduce some unwanted bias once race/ethnicity is taken into consideration.

The next attribute that we need to assess is the `date_time` attrbute and all other time attributes that come along with it. There are very few datetime attrbutes missing from this dataset, but the times that do exist are distributed in an odd way. We see stop times occur most commonly on the hour followed by the half hour. After these two timestamps most reported stops occur every ten minutes. These inaccuracies make the analysis of the Veil of Darkness technique quite difficult. We do not know whether these times represent the beginning or end of a traffic stop nor do we know the duration of the traffic stop.

Lastly, the breakdown of `subject_race` is fairly predictable given the demographic of the greater San Diego area. We see a number of codes that we're not all too concerned with given how small of percentage they make up in the grand scheme of things. However, there may be some errors with precision that we are not yet aware of and may become apparent after undergoing our own Veil of Darkness analysis.

2 2.

`subject_race`

For the races of our citizens the top five by percentage are as follows:

- White (41.7%)
- Hispanic (30.6%)
- Black (11.4%)
- Other (7.5%)
- Other Asian (4.8%)

Less than 1% of the races in the dataset are missing. This is not all significant and the distribution of the times of day that these missing values occur is insignificant. Therefre, we should be satisified to omit those values that are missing in our dataset.

`subject_age`

As was mentioned before, the rate of missingness for the `subject_age` column is about 2.5%. The distribution of our ages is right-skewed with an average age of just over 37 years of age. This makes sense given that this age is representative of the people we'd expect to see on the road commuting to and from work.

The average age of white drivers who were stopped (38.8) is larger than the ages of both the hispanic (35.2) and black (35.9) drivers that were stopped. The largest average age of stopped drivers came from those who were identified as being filipino (39.8) and the lowest average age of stopped drivers came from the cambodian population (33.2).

`date_time`

Again, a very small number of timestamps were not recorded in the police stops data making temporal measurements fairly complete. The number of records that occur either on the hour or on the half hour make up just below 17% of the dataset making these two minutes out of the hour severly overrepresented in the collection of police stops. When performing bivariate analyses and assessing something like the Veil of Darkness the temporal data that shows up near the border of whatever time range we're dealing with may need to be examined further.

`service_area`

All of the service areas in our dataset are well known and the largest portion of stops come from serive area 310 which represents the eastern portion of San Diego. This service area also contains portion of Interstate 8 as well as Interstates 805 and 15. These are roads are major thoroughfares for people commuting to and from work. The same goes for the northern division represented by service area 120. This area contains a large portion of Interstate 5 including where it intersects with Interstate 8 that was mentioned earlier.

3 4.

The thought that sparked this investigation was that the proportion of minority drivers pulled over by police is greater during daylight hours when officers can more easily identify race.

Previously, statistical methods of looking for signs of racial profiling by police compared stops to population, however looking specifically at what is called the "veil of darkness" may prove to reveal the underlying issue.

This method looks at stops around the time of sunrise and sunset and assumes an officer can better observe a driver's skin color when the sun is out. By limiting the analysis to stops that happened during dawn and dusk inter-twilight hours, we are able to normalize the effects of a variable driving population. For example, 6 p.m. during the winter is dark and 6 p.m. during the summer is light, but in general the driving population should be similar at that time. The proportion of stops during the two seasons should theoretically be constant.

When we analyze the two time periods we see the following trends. During the non-inter-twilight time periods we see the following stop rates:

- White (42.4%)
- Hispanic (30.0%)
- Black (11.3%)
- Other (7.6%)
- Other Asian (4.8%)

During the inter-twilight period we see the following stop rates:

- White (37.3%)
- Hispanic (34.0%)
- Black (12.6%)
- Other (6.9%)
- Other Asian (4.6%)

From these results we can see that the black population saw a 1.3% increase in stops and the hispanic population saw a 4.0% increase in stops. Additionally, white drivers saw a significant decrease of 5.1% which leads us to believe that departments pull over minorities at a higher rate during these inter-twilight hours rather than in daylight.

3.0.1 EDA

```
In [106]: stops_2014 = pd.read_csv('data/vehicle_stops_2014_datasd_v1.csv')
          stops_2015 = pd.read_csv('data/vehicle_stops_2015_datasd_v1.csv')
          stops_2016 = pd.read_csv('data/vehicle_stops_2016_datasd_v1.csv')
```

```
stops_2017 = pd.read_csv('data/vehicle_stops_2017_datasd_v1.csv')
stops_final = pd.read_csv('data/vehicle_stops_final_datasd_v1.csv')
```

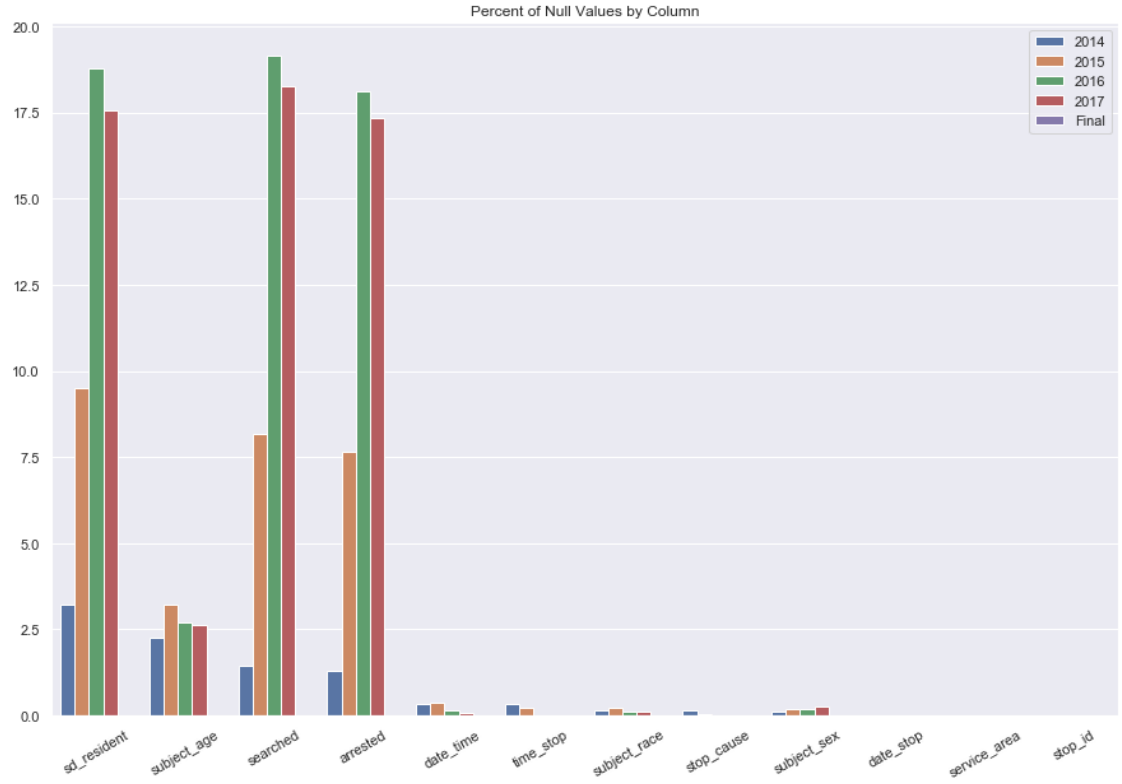
In [107]:
```
stops_final = stops_final.drop(['action', 'search_basis', 'search_type'], axis=1)
```

In [108]:
```
def percent_missing(df):
    df1 = df.drop(['obtained_consent', 'property_seized', 'contraband_found'], axis=
    missing_value_df = (df1.isnull().sum() * 100 / len(df1)).sort_values(ascending=Fa
    return missing_value_df

missing_2014 = percent_missing(stops_2014)
missing_2015 = percent_missing(stops_2015)
missing_2016 = percent_missing(stops_2016)
missing_2017 = percent_missing(stops_2017)
missing_final = percent_missing(stops_final)
```

Missngness by Column by Year

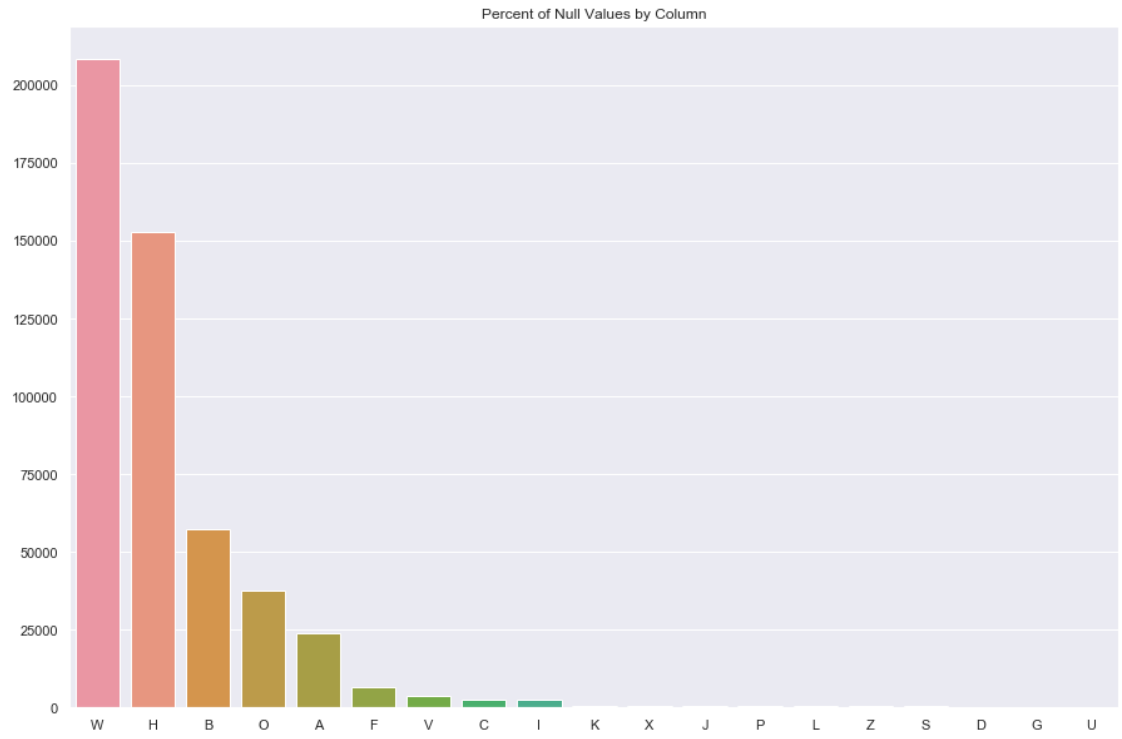
In [171]:
```
column_missingness = pd.concat([missing_2014, missing_2015, missing_2016, missing_201
                                keys=['2014','2015','2016','2017','Final'])
p = sns.barplot(x=column_missingness.index.get_level_values(1),
                y=column_missingness.values,
                hue=column_missingness.index.get_level_values(0))
p.set_xticklabels(p.get_xticklabels(), rotation=30)
sns.set(rc={'figure.figsize':(15,10)})
p.set_title('Percent of Null Values by Column')
plt.show()
```



Breakup by Race

```
In [417]: by_race = pd.concat([stops_2014.subject_race,
                               stops_2015.subject_race,
                               stops_2016.subject_race,
                               stops_2017.subject_race,
                               stops_final.subject_race],
                               keys=['2014',
                                     '2015',
                                     '2016',
                                     '2017',
                                     'Final'])
```

```
p = sns.barplot(x=by_race.value_counts().index, y=by_race.value_counts().values)
sns.set(rc={'figure.figsize':(15,10)})
p.set_title('Percent of Null Values by Column')
plt.show()
```

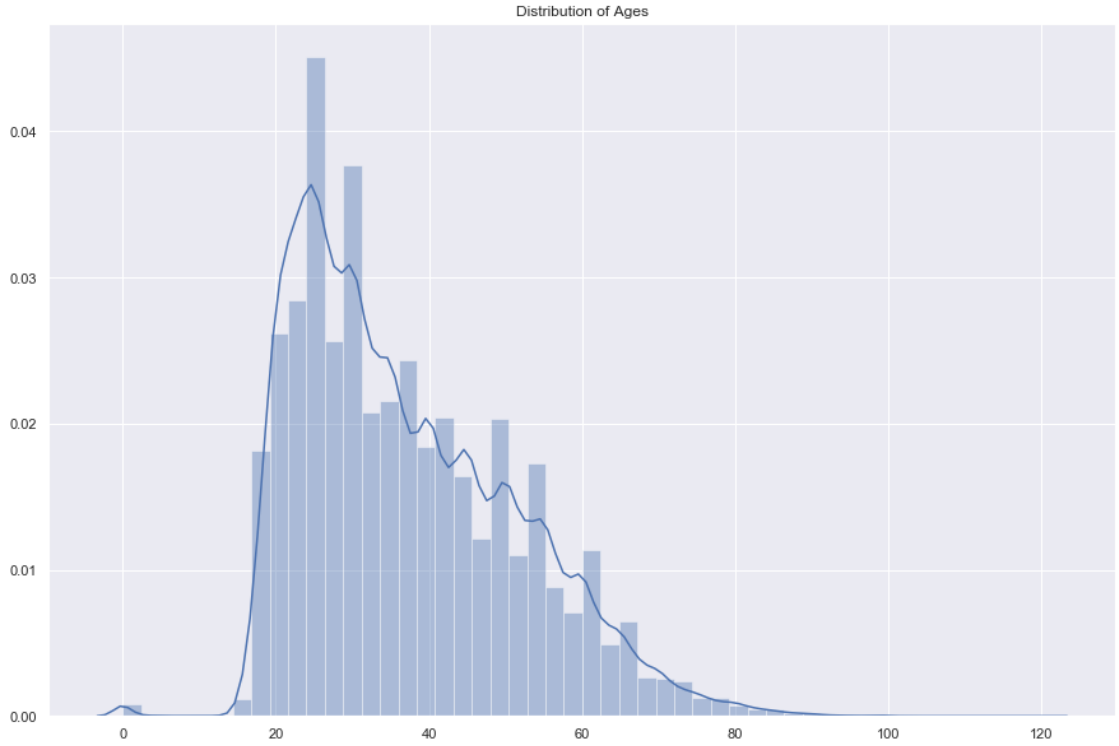


Breakup of Age

```
In [306]: by_age = pd.concat([stops_2014.subject_age,
                              stops_2015.subject_age,
                              stops_2016.subject_age,
                              stops_2017.subject_age,
                              stops_final.subject_age],
                              keys=['2014',
                                    '2015',
                                    '2016',
                                    '2017',
                                    'Final'])

          by_age = by_age.apply(lambda age: pd.to_numeric(age, errors='coerce')).dropna()
          by_age = by_age[by_age < 125]
          p = sns.distplot(by_age.values)
          sns.set(rc={'figure.figsize':(15,10)})
          p.set_title('Distribution of Ages')
          plt.show()
```

```
/Users/colerichmond/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarn
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



Service Area

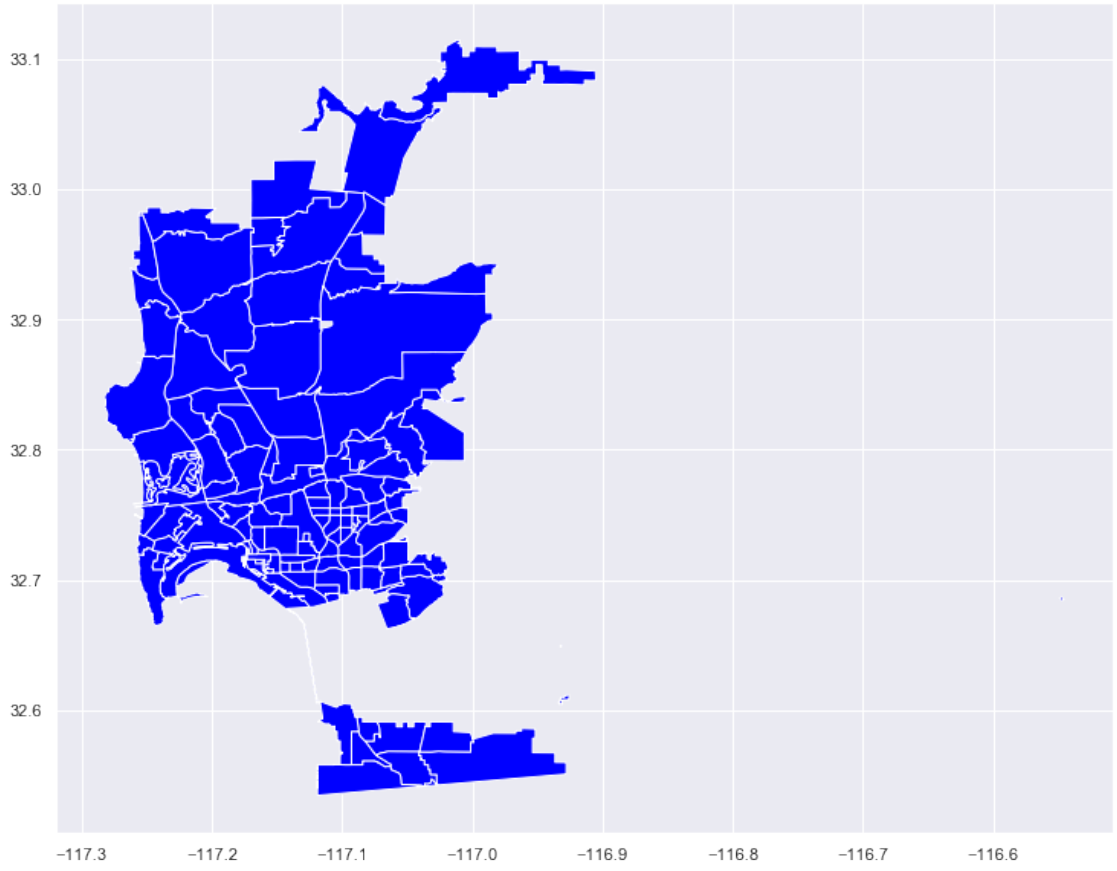
```
In [253]: df_beats = gpd.read_file('data/pd_beats_datasd.geojson')
          df_beats.head()
```

Out[253]:

	objectid	beat	div	serv	name	geometry
0	3	935	9	930	NORTH CITY	(POLYGON ((-117.20425008 32.96202036, -117.204...
1	7	0	0	0	SAN DIEGO	(POLYGON ((-117.22524988 32.70268691, -117.225...
2	9	511	5	510		(POLYGON ((-117.22524988 32.70268691, -117.224...
3	10	722	7	720	NESTOR	POLYGON ((-117.087138 32.583822, -117.08695 32...
4	11	314	3	310	BIRDLAND	POLYGON ((-117.1537739 32.78023503, -117.15424...

```
In [211]: ax = df_beats.plot(color='blue', column='beat')
```

```
/Users/colerichmond/anaconda3/lib/python3.6/site-packages/geopandas/plotting.py:393: UserWarni
  "'color'.", UserWarning)
```



```
In [363]: by_service = pd.concat([stops_2014.service_area,
                                  stops_2015.service_area,
                                  stops_2016.service_area,
                                  stops_2017.service_area,
                                  stops_final.service_area],
                                  keys=['2014',
                                        '2015',
                                        '2016',
                                        '2017',
                                        'Final'])

          (by_service.value_counts() / len(by_service)).sort_values(ascending=False)
```

```
Out[363]: 310        0.087038
          120        0.081696
          520        0.075034
          710        0.070643
          240        0.067069
          110        0.062980
          930        0.062411
          620        0.051000
          230        0.049871
          610        0.046669
          320        0.045728
          720        0.045258
          430        0.042008
          510        0.039788
          440        0.036704
          810        0.035630
          830        0.033174
          820        0.032696
          Unknown    0.031387
          530        0.002139
          130        0.000510
          630        0.000494
          840        0.000074
          Name: service_area, dtype: float64
```

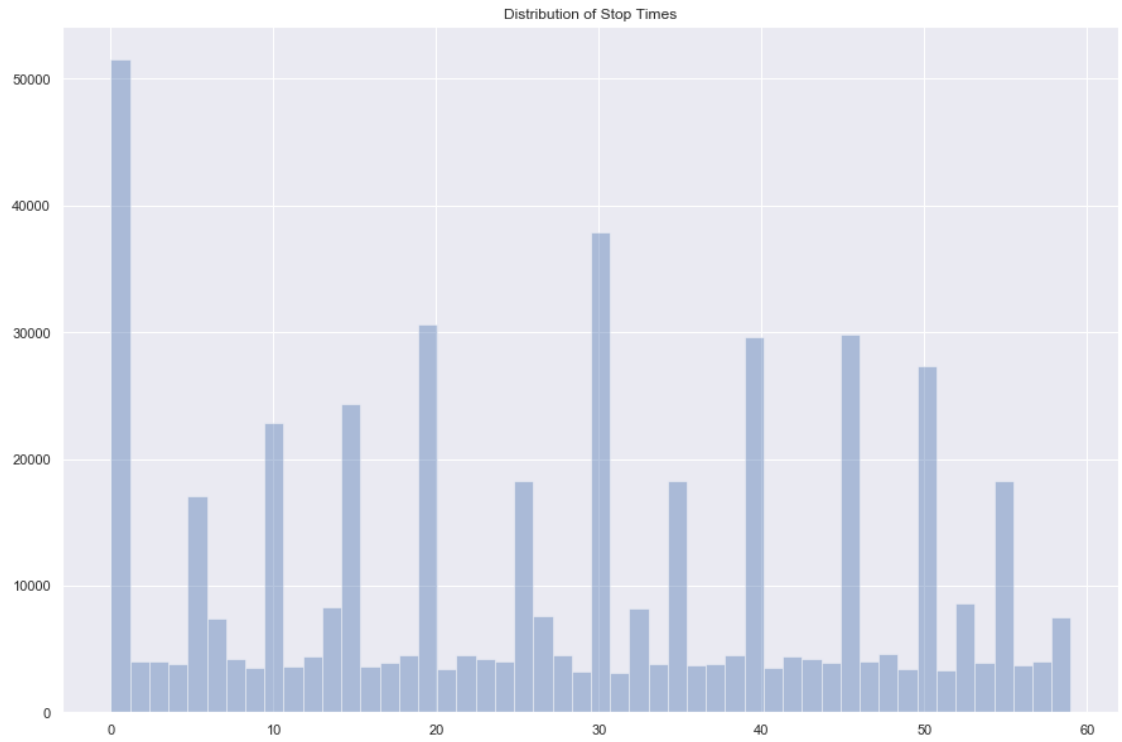
Time of Stop

```
In [272]: by_time = pd.concat([stops_2014.date_time,
                               stops_2015.date_time,
                               stops_2016.date_time,
                               stops_2017.date_time,
                               stops_final.date_time],
```

```
                                keys=['2014',
                                      '2015',
                                      '2016',
                                      '2017',
                                      'Final'])

        by_time = by_time.apply(lambda time: pd.to_datetime(time).minute)
        by_time = by_time.dropna()
        p = sns.distplot(by_time.values, kde=False)
        sns.set(rc={'figure.figsize':(15,10)})
        p.set_title('Distribution of Stop Times')
        plt.show()
```

```
/Users/colerichmond/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarn
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



Veil of Darkness (5:09pm to 8:29pm)

```
In [413]: by_time = pd.concat([stops_2014.date_time,
                               stops_2015.date_time,
                               stops_2016.date_time,
                               stops_2017.date_time,
                               stops_final.date_time],
```

```
                                keys=['2014',
                                      '2015',
                                      '2016',
                                      '2017',
                                      'Final'])

          by_time = by_time.apply(lambda time: pd.to_datetime(time))
          by_time = by_time.dropna()
          times = by_time.apply(lambda times: times.time())
```

```
In [414]: times = by_time.apply(lambda times: times.time())
          intertwilight = (times >= pd.to_datetime('17:09', format='%H:%M').time()) & (times <=
```

```
In [423]: # inter-twilight
          by_time[intertwilight].to_frame().join(by_race.to_frame())['subject_race'].value_cou
```

```
Out[423]: W    0.373750
          H    0.339935
          B    0.126303
          O    0.068961
          A    0.046010
          F    0.012981
          V    0.008049
          C    0.007141
          I    0.006975
          X    0.001967
          P    0.001634
          K    0.001241
          J    0.001195
          L    0.001120
          D    0.000802
          S    0.000741
          Z    0.000590
          U    0.000333
          G    0.000272
          Name: subject_race, dtype: float64
```

```
In [427]: non_intertwilight = (times <= pd.to_datetime('17:09', format='%H:%M').time()) | (time
```

```
In [429]: # non-inter-twilight
          by_time[non_intertwilight].to_frame().join(by_race.to_frame())['subject_race'].value_
```

```
Out[429]: W    0.424024
          H    0.300340
          B    0.112613
          O    0.076457
          A    0.047722
          F    0.013410
          V    0.007167
```

```
C    0.005122
I    0.004775
K    0.001475
X    0.001348
J    0.001253
P    0.001170
Z    0.000760
L    0.000696
S    0.000649
D    0.000508
G    0.000261
U    0.000250
Name: subject_race, dtype: float64
```