cogs9_proj

December 18, 2020

1 Cogs9 Project: For the SDPD in 2019, is there a significant difference in the likelihood of someone being stopped according to their race?

Group Name: CV GANG
By Andrew Cheng

1.1 Analysis

For our analysis method, we'll be using python pandas - link to repository-https://github.com/AndrewCheng2002/Cogs-9-Project

[85]:
```
#Modules
import numpy as np
import pandas as pd
import geopandas as gpd
#import geoplot as geoplot
import math
import matplotlib.pyplot as plt
import datetime

%matplotlib inline
plt.style.use('fivethirtyeight')
```

```
        ␣
↪---------------------------------------------------------------------------

    ModuleNotFoundError                       Traceback (most recent call␣
↪last)

    <ipython-input-85-f6c182b9deb5> in <module>
      3 import pandas as pd
      4 import geopandas as gpd
----> 5 import geoplot as geoplot
      6 import math
```

```
      7 import matplotlib.pyplot as plt
```

```
ModuleNotFoundError: No module named 'geoplot'
```

Data Collection: How do we get data to preform analysis on?

When answering a data science question, our first tasks is to gather the data itself. However not just any data will suffice and in order to get an accurate, unbiased and ethical model, we need ensure that our data is sufficient and representative of the population.(San Diego in 2019) We needed to find a source that was reliable to ensure fairness and correct represenation as well as one that was large enough to not be affected by outliers. After spending lots time and deliberation we settled on using RIPA (An act passed by the government that requires police departments to publish their data regarding stops, arrests, etc.) to get data on race and stop information. We simply downloaded the data off the RIPA website which was conviently in csv format. In addition to race and stop data we also need data of specific race demographics. In this case, we decided to use the US census which has information about the race distribution of the San Diego County in 2019. Even though we are analyzing the SDPD in 2019, which not a census year(every decade), we felt that the predictive algorithms used by the US government were trustworthy and sufficient enough to get an accerate representation of the population demographic. Because there's was no quick download function and the fact that we're only analyzing 7 races, we decided it was easier to just webscrapped the information and put it in a table.

[2]:
```
#We made pandas read the RIPA csv files we downloaded
data_race_raw = pd.read_csv('ripa_race_datasd.csv')
data_stops_raw = pd.read_csv('ripa_stops_datasd.csv', low_memory = False)

#For the census we webscrapped the information and manually inserted in a table
data_census_race = pd.DataFrame({'percentage of population': [45,5.5,34.1,.9,12.
 ↪6,1.3,.6]},
index = ['White','Black/African American','Hispanic/Latino/a','Middle Eastern␣
 ↪or South Asian','Asian','Native American','Pacific Islander'])
```

[3]:
```
data_race_raw
```

[3]:

	stop_id	pid	race
0	2443	1	White
1	2444	1	White
2	2447	1	Hispanic/Latino/a
3	2447	2	Hispanic/Latino/a
4	2448	1	White
…	…	…	…
394970	356019	1	Black/African American
394971	356025	1	Black/African American
394972	356080	1	White
394973	356300	1	Black/African American
394974	356303	1	Black/African American

```
[394975 rows x 3 columns]
```

[4]:

```
data_stops_raw
```

[4]:

```
        stop_id        ori agency  exp_years   date_stop time_stop  \
0          2443  CA0371100     SD         10  2018-07-01  00:01:37
1          2444  CA0371100     SD         18  2018-07-01  00:03:34
2          2447  CA0371100     SD          1  2018-07-01  00:05:43
3          2447  CA0371100     SD          1  2018-07-01  00:05:43
4          2448  CA0371100     SD          3  2018-07-01  00:19:06
…             …          …      …          …           …
391129   356019  CA0371100     SD          1  2020-09-30  23:05:00
391130   356025  CA0371100     SD          1  2020-09-30  23:38:00
391131   356080  CA0371100     SD         18  2020-09-30  15:31:00
391132   356300  CA0371100     SD         18  2020-09-30  19:30:00
391133   356303  CA0371100     SD          1  2020-09-30  19:37:52

        stopduration  stop_in_response_to_cfs  officer_assignment_key  \
0                 30                        0                       1
1                 10                        0                       1
2                 15                        1                      10
3                 15                        1                      10
4                  5                        0                       1
…                  …                        …                      …
391129             7                        1                       1
391130            30                        1                       1
391131             5                        0                       1
391132           180                        1                       1
391133            45                        0                       1

                                         assignment  …  \
0       Patrol, traffic enforcement, field operations  …
1       Patrol, traffic enforcement, field operations  …
2                                             Other  …
3                                             Other  …
4       Patrol, traffic enforcement, field operations  …
…                                                 …  …
391129  Patrol, traffic enforcement, field operations  …
391130  Patrol, traffic enforcement, field operations  …
391131  Patrol, traffic enforcement, field operations  …
391132  Patrol, traffic enforcement, field operations  …
391133  Patrol, traffic enforcement, field operations  …

                  beat_name pid isstudent perceived_limited_english  \
0         Pacific Beach 122   1         0                         0
1         Mission Beach 121   1         0                         0
```

```
2              El Cerrito 822    1         0                         0
3              El Cerrito 822    2         0                         0
4            Ocean Beach 614    1         0                         0
…                      …   …        …                         …
391129         Harborview 527    1         0                         0
391130      Core-Columbia 524    1         0                         0
391131            Unknown 999    1         0                         0
391132   Carmel Mountain 232    1         0                         0
391133        Golden Hill 517    1         0                         0

       perceived_age  perceived_gender gender_nonconforming gend  gend_nc  \
0                 25              Male                    0    1      NaN
1                 25              Male                    0    1      NaN
2                 30              Male                    0    1      NaN
3                 30            Female                    0    2      NaN
4                 23              Male                    0    1      NaN
…                …                 …                   …    …       …
391129            50            Female                    0    2      NaN
391130            35              Male                    0    1      NaN
391131            60              Male                    0    1      NaN
391132            25              Male                    0    1      NaN
391133            28              Male                    0    1      NaN

       perceived_lgbt
0                  No
1                  No
2                  No
3                  No
4                  No
…                 …
391129             No
391130             No
391131             No
391132             No
391133             No

[391134 rows x 29 columns]
```

[5]:

```
data_census_race
```

[5]:

```
                               percentage of population
White                                              45.0
Black/African American                              5.5
Hispanic/Latino/a                                  34.1
Middle Eastern or South Asian                       0.9
Asian                                              12.6
Native American                                     1.3
```

```
Pacific Islander                                    0.6
```

Data Wrangling: How do we make our data useable

As of now, our raw data has a lot of information that we don't really need to answer our data science question. (For the SDPD in 2019, is there is significant difference in the likely hood of someone being stopped according to their race?) The columns that we need consists of the **races** of the people stopped and the **date**. (to restrict our time interval) In order to do this and combine it into one table, we need to first drop all the columns in the stops data we aren't using and merge that table to the race table at the **stop_id**.

Then we have to remove duplicates that arise from the merge. The reason why the merge creates duplicates is because multiple people can be stopped at one stop_id represented by the **pid**(person id) which means that there will be an addition copy of the stop_id from the dates table. (Has unique stops and doesn't account for pid) In addition to removing duplicates, we have to set the final table to only contain stop_ids with a date in 2019. To this, we need to convert the dates in the date column to something we can read, such as a **datetime**. After converting the date column of strings to a datetime, we can restrict the table to only include stops from the year 2019.

[6]:
```python
data_race = data_race_raw.set_index('stop_id')
data_race
```

[6]:
```
         pid                     race
stop_id
2443       1                    White
2444       1                    White
2447       1       Hispanic/Latino/a
2447       2       Hispanic/Latino/a
2448       1                    White
…        …                       …
356019     1  Black/African American
356025     1  Black/African American
356080     1                    White
356300     1  Black/African American
356303     1  Black/African American

[394975 rows x 2 columns]
```

[7]:
```python
data_date = pd.DataFrame().assign(date = data_stops_raw.get('date_stop'),␣
↪stop_id = data_stops_raw.get('stop_id')).set_index('stop_id')
data_date
```

[7]:
```
               date
stop_id
2443     2018-07-01
2444     2018-07-01
2447     2018-07-01
2447     2018-07-01
```

```
2448      2018-07-01
…              …
356019    2020-09-30
356025    2020-09-30
356080    2020-09-30
356300    2020-09-30
356303    2020-09-30

[391134 rows x 1 columns]
```

[8]:
```python
#Merge race data set with the dates from the stop data set with the stop_id
data_merged = data_race.merge(data_date,left_index = True, right_index = True)
data_merged
```

[8]:
```
          pid                    race        date
stop_id
2443        1                   White  2018-07-01
2444        1                   White  2018-07-01
2447        1      Hispanic/Latino/a  2018-07-01
2447        1      Hispanic/Latino/a  2018-07-01
2447        2      Hispanic/Latino/a  2018-07-01
…       …                        …         …
356019      1  Black/African American  2020-09-30
356025      1  Black/African American  2020-09-30
356080      1                   White  2020-09-30
356300      1  Black/African American  2020-09-30
356303      1  Black/African American  2020-09-30

[595128 rows x 3 columns]
```

[9]:
```python
#Remove Duplicates
data_final = data_merged.drop_duplicates()

#Get the year from the date string
def to_year(date):
    dt = datetime.datetime.strptime(date,'%Y-%m-%d')
    return dt.year

#Include data within subjected time interval
data_final = data_final[data_final.get('date').apply(to_year) == 2019]
data_final
```

[9]:
```
          pid                           race        date
stop_id
84362       1              Hispanic/Latino/a  2019-01-01
84364       1                          White  2019-01-01
84369       1         Black/African American  2019-01-01
```

```
84372      2              Hispanic/Latino/a  2019-01-01
84376      1  Middle Eastern or South Asian  2019-01-01
…       …                              …          …
254761     8                          White  2019-12-31
254771     2                          White  2019-12-31
254776     1                Native American  2019-12-31
255002     4                          White  2019-12-31
255002     5                          White  2019-12-31

[8398 rows x 3 columns]
```

Exploratory/Descriptive Analysis: What does our data say?

Now that we have our data neated sorted and ready to use, lets use that to answer our question. In order to do that we started off by producing some basic descriptive analysis to get a general idea of the trend of our data. For the descriptive analysis, we decided to compare the percentages of each race stopped versus the percentages of each race's demographic to see if there was any significant difference between the two. To do this, we created a new table showing the percentage of each race stopped and then merged the demographic table to that table. Then generate basic statistics that comparet the two distributions such as the absolute mean difference, the standard deviation of the absolute differences, and the range of the absolute differences.

[10]:
```
#Generate Race Percentages Table from the data wrangled data_final
race_percentage = data_final.groupby('race').count()/data_final.shape[0]*100
race_percentage = race_percentage.drop(columns = ['date']).
 ↪rename(columns={'pid':'percentage stopped'})
race_percentage
```

[10]:
```
                               percentage stopped
race
Asian                                   10.431055
Black/African American                  20.159562
Hispanic/Latino/a                       22.552989
Middle Eastern or South Asian            8.001905
Native American                          3.346035
Pacific Islander                         6.215766
White                                   29.292689
```

Now we see that there's a difference between the two distributions in the following table below.

[11]:
```
#Now merge the census data and sort by lowest population to highest
race_census_percentage = race_percentage.merge(data_census_race,left_index =␣
 ↪True,right_index = True)
race_census_percentage = race_census_percentage.sort_values('percentage of␣
 ↪population', ascending = True)
race_census_percentage
```

[11]:

	percentage stopped	percentage of population
Pacific Islander	6.215766	0.6
Middle Eastern or South Asian	8.001905	0.9
Native American	3.346035	1.3
Black/African American	20.159562	5.5
Asian	10.431055	12.6
Hispanic/Latino/a	22.552989	34.1
White	29.292689	45.0

Basic Statistics generated below:

[12]:
```
difference = race_census_percentage.get('percentage␣
↪stopped')-race_census_percentage.get('percentage of population')
mean_abs_diff = abs(difference).mean()
std_abs_diff = np.std(abs(difference))
range_abs_diff = abs(difference).max()-abs(difference).min()
print('mean abs diff: '+ str(mean_abs_diff) +'%, std abs diff: ' +␣
↪str(std_abs_diff) + '%, range abs diff: ' + str(range_abs_diff)+'%')
```

```
mean abs diff: 8.406647841322764%, std abs diff: 5.22518917771041%, range abs
diff: 13.661276494403433%
```

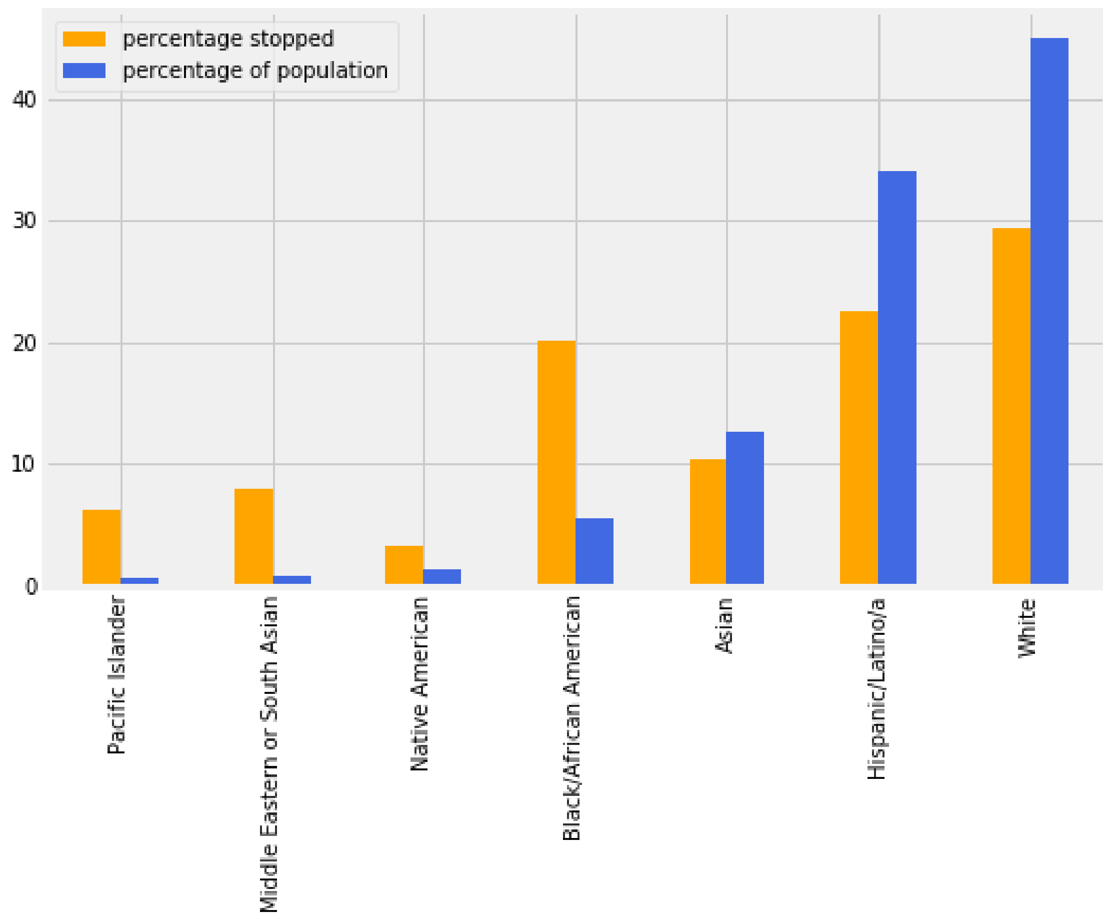
Our descriptive analysis shows that there's a decent chance that race is correlated with the likelyhood of being stopped by the police in San Diego with the mean difference from the demographic being 8.41%. Assuming that these results are realiable(proven later in our statistical analysis), we can then use this to justify or disprove our hypothesis that blacks and hispanics are stopped more frequently by the police compared to other races.

Data Visualization: What does the data show us

Once we completed some basic analysis, the next step is to visualize the data and see how each races compares to it's demographic. With Pandas it's really easy and simple to perform since we already have a completed table from our exploratory analysis and data wrangling we did. We decided to make a grouped bar graph to compare the distribution of two categorical variables. We decided to choose blue and orange as the colors and make the graph a bit wider to make it easier on the eyes. As you can see, miniorities are stopped much more often than the rest of the races, especially in Black/African American and much less in Asian and White which supports our hypothesis. However, Hispanics/Latinos are surprisingly stopped less relative to their population according to the visualization which goes against our hypothesis.

[13]:
```
race_census_percentage.plot(kind = 'bar', figsize = [8,5],color =␣
↪['orange','royalblue'])
```

[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2c927fe358>
```



Statistical Analysis: How do we know that our results are trustworthy

After performing exploratory analysis to generate basic statistics and producing a visualization, we need to make sure that these results are valid and significant. In order to achieve this we will perform a hypothesis test and calculate a confidence interval and p-value for our statistic.

- Null: There is no significant difference between the percentage of races stopped respective to their demographic
- Alternate: There is a significant difference between the percentage of race stopped respective to their demographic

[14]:
```
#Test Statistic will be the Mean Difference
test_stat = mean_abs_diff
test_stat
```

[14]: 8.406647841322764

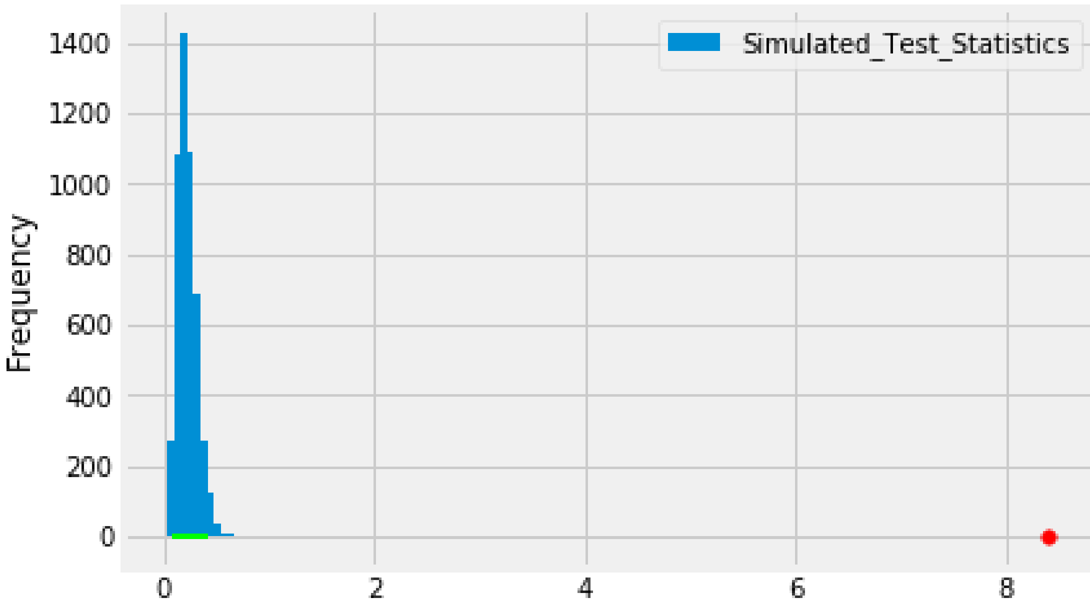
[15]:
```
#We'll generate about 5000 sample test stats using the census data to create a␣
↪95% confidence interval
```

```
num_repetitions = 5000
population = data_final.shape[0]

simulated_test_stats = np.array([])

for i in range(num_repetitions):
    model_proportions = race_census_percentage.get('percentage of population')/
 ↪100
    sample = np.random.multinomial(population, model_proportions)/population
    sim_test_stat = abs(model_proportions-sample).mean()*100
    simulated_test_stats = np.append(simulated_test_stats, sim_test_stat)

simulated_test_stats
```

[15]:
```
array([0.27462321, 0.16178002, 0.26500868, …, 0.1989045 , 0.1827646 ,
       0.25130473])
```

[16]:
```
#Lets look at the distribution and generate the 95% confidence interval
t = pd.DataFrame().assign(Simulated_Test_Statistics = simulated_test_stats)
t.plot(kind='hist')

confidence_interval = [np.percentile(simulated_test_stats,2.5),np.
 ↪percentile(simulated_test_stats,97.5)]
plt.scatter(test_stat, 0, color='red', s=30);
plt.plot(confidence_interval,[0,0], color = 'lime', linewidth = 2)
print('Confidence Interval: [' + str(confidence_interval[0]) +', ' +␣
 ↪str(confidence_interval[1]) + ']')
```

```
Confidence Interval: [0.08422379478106999, 0.4320547409247092]
```



[17]:
```
#Now lets generate a p value
p_value = np.count_nonzero(simulated_test_stats >= test_stat)/
 ↪simulated_test_stats.shape[0]
print('p value: ' + str(p_value))
```

```
p value: 0.0
```

We reject the null, therefore the difference in the percentage of races being stopped is statistically significant. With this test, we are confident to use the results as justification to prove if our hypothesis is correct or incorrect.

Geospatial Analysis: Is there any bias in our data due to the location of the stops

In this analysis we compare the total distribution of the San Diego police stops to the total demographic of San Diego to see if there's any difference between the two. However, the police may not uniformly partol each district of San Diego. For example, the police might be more active in urban areas where minorities may make up more of the population as opposed to suburbs where most of the populations are whites, hispanics, and asians. We'll perform a geospatial analysis to see if the frequency a police stops at a location has an effect on the mean difference of races to demographic stopped. This will tell us if there's any bias in our data and how severe it is.

The process will involving using geopandas to read a geodataframe in the form of a shp file. The shape file data we will be using is from ripa which contains a beat map of the san diego county. (A beat is a territory an officer partols) To start off with our geospatial analysis, we need to do a bit of addition data wrangling by adding the beat data to our data final. After adding all the neccessary columns (The frequency of stops at each beat and the mean absolute difference of race to demographic of each beat), we'll use the beat map to create visualizations to look for possible corrations, hence bias in our data. More specifically, we choose to make a cartogram that represented how much each beat was affected by the police according to the frequency of stops and a choropleth that compared the mean absolute difference of each beat.

[18]:
```python
#Data Wrangling

#We'll add the location to our final data set
data_loc = pd.DataFrame().assign(stop_id = data_stops_raw.get('stop_id'),beat =␣
↪data_stops_raw.get('beat'))
data_loc = data_loc.set_index('stop_id')
data_geo = data_final.merge(data_loc,left_index = True, right_index = True)
data_geo = data_geo.drop_duplicates()
data_geo
```

[18]:
```
        pid                         race        date  beat
stop_id
84362     1            Hispanic/Latino/a  2019-01-01   839
84364     1                        White  2019-01-01   124
84369     1       Black/African American  2019-01-01   614
84372     2            Hispanic/Latino/a  2019-01-01   122
84376     1  Middle Eastern or South Asian  2019-01-01   122
…       …                            …         …   …
254761    8                        White  2019-12-31   521
254771    2                        White  2019-12-31   512
254776    1              Native American  2019-12-31   121
255002    4                        White  2019-12-31   614
255002    5                        White  2019-12-31   614

[8398 rows x 4 columns]
```

[19]:
```python
#This is the frequency of stops for each beat
geo_final = data_geo.groupby('beat').count()
geo_final = geo_final.assign(frequency = geo_final.get('pid')).
↪drop(columns=['pid','date','race'])
geo_final
```

[19]:
```
      frequency
beat
111          67
112          26
113          23
114          50
115          53
…           …
934          61
935          19
936          10
937          20
999          69
```

[122 rows x 1 columns]

[20]:
```
#This is the returns absolute mean difference of each beat
def get_mean_diff(races):
    race_percentages = races/races.sum()*100
    return abs(race_percentages.get('frequency')- data_census_race.
 ↪get('percentage of population')).mean()
```

[22]:
```
#Generating the final table with mean abs difference and frequency
geo_diff = data_geo.groupby(['beat','race']).count().unstack(fill_value=0).
 ↪stack()
geo_diff = geo_diff.assign(frequency = geo_diff.get('pid')).
 ↪drop(columns=['pid','date'])

mean_diff = np.array([])
for i in geo_final.index:
    mean_diff = np.append(mean_diff, get_mean_diff(geo_diff.loc[i]))

geo_final = geo_final.assign(mean_diff = mean_diff)
geo_final
```

[22]:

beat	frequency	mean_diff
111	67	10.233262
112	26	9.791209
113	23	7.959006
114	50	11.171429
115	53	11.772507
…	…	…
934	61	10.422014
935	19	7.278195
936	10	7.571429
937	20	14.200000
999	69	10.591718

[122 rows x 2 columns]

[33]:
```
beat_map_raw = gpd.read_file('pd_beats_datasd/pd_beats_datasd.shp')
beat_map_raw
```

[33]:

	objectid	beat	div	serv	name \
0	3	935	9	930	NORTH CITY
1	7	0	0	0	SAN DIEGO
2	8	511	5	510	None
3	9	722	7	720	NESTOR
4	10	314	3	310	BIRDLAND

```
..          …   …   …    …                     …
135         610   243    2   240                MIRAMAR
136         616   937    9   930    BLACK MOUNTAIN RANCH
137         617   936    9   930        TORREY HIGHLANDS
138         618   233    2   230      RANCHO PENASQUITOS
139         619   235    2   230             SAN PASQUAL

                                               geometry
0     MULTIPOLYGON (((6268975.465 1931147.469, 62689…
1     MULTIPOLYGON (((6261648.576 1836846.672, 62616…
2     MULTIPOLYGON (((6261640.429 1836823.561, 62616…
3     POLYGON ((6302781.000 1793246.001, 6302905.000…
4     POLYGON ((6284667.652 1874418.895, 6284694.392…
..                                                  …
135   POLYGON ((6295777.084 1908435.558, 6295790.126…
136   POLYGON ((6295209.132 1952772.422, 6295208.647…
137   POLYGON ((6287497.113 1936631.116, 6287462.453…
138   POLYGON ((6306176.245 1943421.146, 6306158.416…
139   POLYGON ((6338763.812 1971222.216, 6338683.658…

[140 rows x 6 columns]
```

[55]:
```
beat_visual= beat_map_raw.merge(geo_final, left_on = 'beat',right_index = True,
    how = 'left')
beat_visual
```

[55]:
```
     objectid  beat  div  serv                    name  \
0           3   935    9   930              NORTH CITY
1           7     0    0     0               SAN DIEGO
2           8   511    5   510                    None
3           9   722    7   720                  NESTOR
4          10   314    3   310                BIRDLAND
..          …   …   …    …                     …
135         610   243    2   240                MIRAMAR
136         616   937    9   930    BLACK MOUNTAIN RANCH
137         617   936    9   930        TORREY HIGHLANDS
138         618   233    2   230      RANCHO PENASQUITOS
139         619   235    2   230             SAN PASQUAL

                                               geometry  frequency  mean_diff
0     MULTIPOLYGON (((6268975.465 1931147.469, 62689…       19.0   7.278195
1     MULTIPOLYGON (((6261648.576 1836846.672, 62616…        NaN        NaN
2     MULTIPOLYGON (((6261640.429 1836823.561, 62616…      117.0  11.084737
3     POLYGON ((6302781.000 1793246.001, 6302905.000…       27.0  13.114286
4     POLYGON ((6284667.652 1874418.895, 6284694.392…       21.0  20.136054
..                                                  …        …         …
135   POLYGON ((6295777.084 1908435.558, 6295790.126…       23.0  12.662112
```

```
136  POLYGON ((6295209.132 1952772.422, 6295208.647…       20.0  14.200000
137  POLYGON ((6287497.113 1936631.116, 6287462.453…       10.0   7.571429
138  POLYGON ((6306176.245 1943421.146, 6306158.416…       39.0  13.915018
139  POLYGON ((6338763.812 1971222.216, 6338683.658…        2.0  15.714286

[140 rows x 8 columns]
```

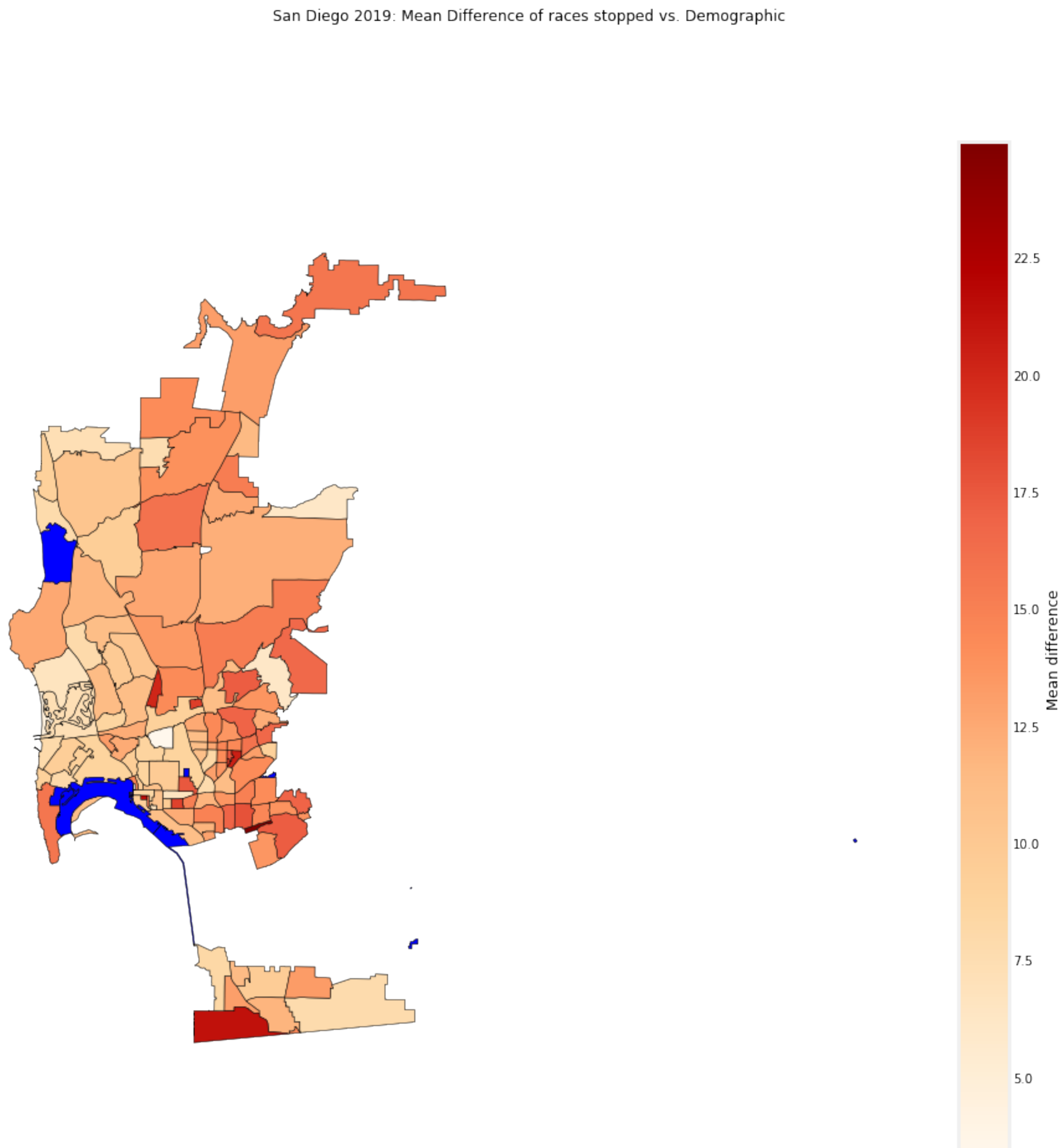
Cartogram: As you can see below, the downtown areas have a lot more police activity

[35]:
```
#ax = geoplot.cartogram(
#    beat_vis, scale='frequency', limits=(0.2, 1),
#    edgecolor='None', figsize=(7, 8)
#)
#geoplot.polyplot(beat_vis, edgecolor='gray', ax=ax
```

Choropleth: As you can see below, beats from the downtown area of San Diego County have higher mean differences compared to the suburbs more north.

[82]:
```
beat_visual.plot(column = 'mean_diff',figsize = [15,15], legend=True,␣
 ↪cmap='OrRd', edgecolor="black",
                 legend_kwds={'label': "Mean difference"},␣
 ↪missing_kwds={'color': 'blue'})
plt.axis('off')
print('Blue Represents that the beat has no stops')

plt.suptitle('San Diego 2019: Mean Difference of races stopped vs. Demographic')
plt.show()
```

```
Blue Represents that the beat has no stops
```



Overlay Visualization: Now lets see how similar the two graphs compare when we overlay them

[ ]:
```

```

Conclusion: Is our question answered?

Although we performed a hypothesis test and recieved a p-value of less than .05, our exploratory analysis may still contain biases and inaccuracies as shown in our geospatial analysis. However we believed that this isn't enough to completely invalidate our findings, but be conscious that our results maybe slighty skewed from not accounting for societal bias.(Police are more active in downtown areas which are typically inhabited by more minioritise) From our hypothesis test in our statistical analysis, we conclude that there's a relationship between someone's race and their likelyhood of being stopped by the people. Parts of our hypothesis were proven right and wrong.

We said that blacks/african americans were more likely of being stopped, which the data firmly suggested. However we also hypothesized that latinos/hispanics would also be targeted more, but according to our findings, this wasn't the case as it was even less likely to be stopped if one was a hispanic. In conclusion the trends to seems to be that minorities are stopped more often than other races in the demographic of San Diego in 2019.

[83]:
```
print('Congrats for making it to the END! :)')
```

```
Congrats for making it to the END! :)
```