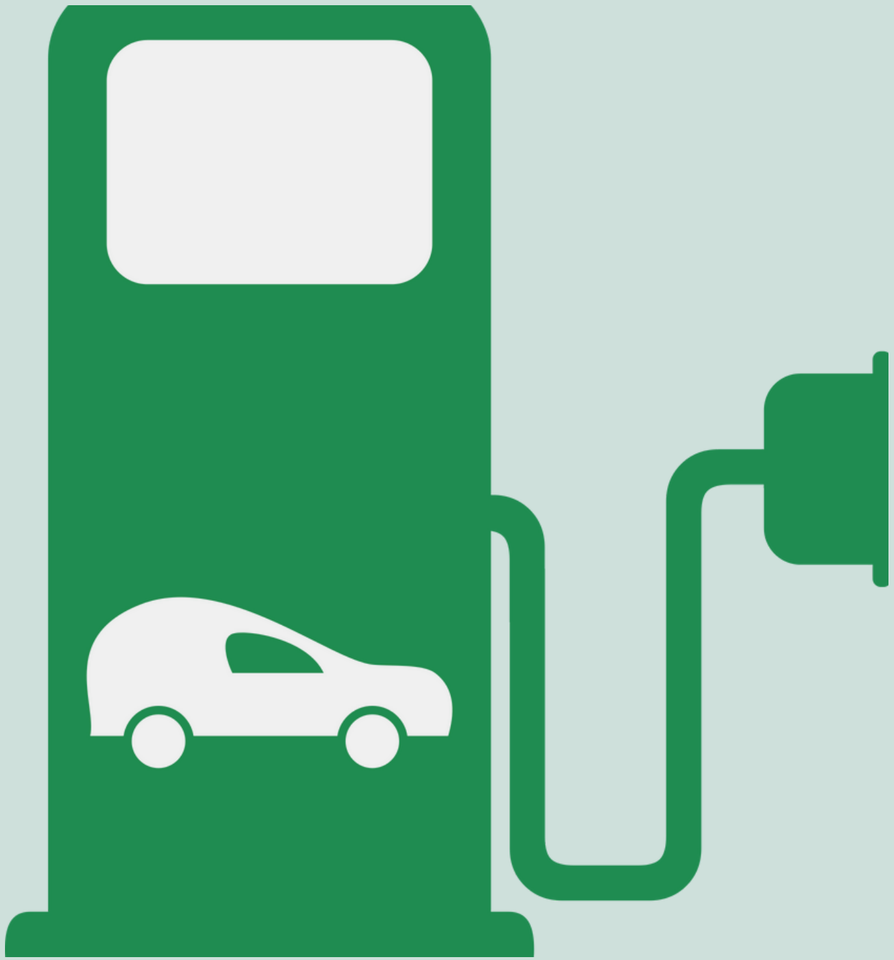
Recharging the Future:

Does Income Determine the Ability to Move Forward?

Jordan Morales
Julia Richard
Kristin Scholten
Hope Youngblood



Overview

- Research questions
- Project scope
- Process of using "Big Data"
- What we learned
- Limitations & project issues

What's the Big Idea?

- There were more than 1.1 million registered electric vehicles on the road in the U.S in 2020.
- The average price of an electric vehicle is $55K v $36K for a traditional vehicle.

Resarch Questions

- Where are electric vehicle charging stations located across the Western US?
- Geographically, what does that look like?
- Economically, what does that look like?

Hyoptheses

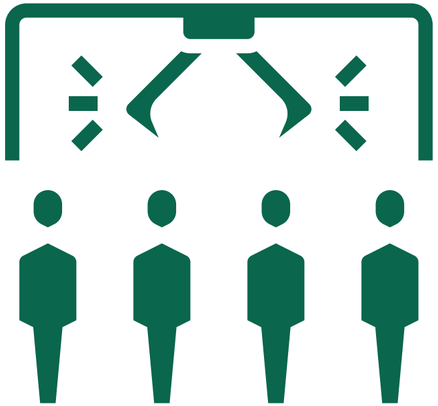
- There is a positive correlation between EV stations & household income
- EV stations will be located close to metro areas
- There are more EV stations within higher-income zip codes

Project Scope

- For data management purposes, we narrowed down our data search to only access the location of EV charging stations in the Western United States (WA, OR, CA, NV, UT, AZ, NM, ID, WY, MT, and CO).
- We used public APIs to access 2 large public databases: US Census Bureau and National Renewable Energy Laboratory (a division of US Department of Energy).
- Through data analysis, we compared the location of the charging stations to predetermined population demographics.
- We summarized the demographics that seemed to be strongly correlated to site location of the charging stations.

US Census Request

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import requests
from census import Census
from config import api_key

c = Census(api_key, year=2019)
```



```
census_data = c.acs5.get(("NAME", "B01003_001E", "B19013_001E", "B01003_001E", "B01002_001E", "B11001_001E",
                          "B19301_001E", "B08303_001E", "B08015_001E",
                          "B17001_002E"), {'for': 'zip code tabulation area:*'})

# # Convert to DataFrame
census_pd = pd.DataFrame(census_data)
census_pd.head()
```

	NAME	B01003_001E	B19013_001E	B01002_001E	B11001_001E	B19301_001E	B08303_001E	B08015_001E	B17001_002E	state	zip code tabulation area
0	ZCTA5 00601	17113.0	14361.0	41.9	5509.0	7493.0	3504.0	3115.0	10552.0	72	00601
1	ZCTA5 00602	37751.0	16807.0	42.9	12740.0	9694.0	10525.0	8645.0	18653.0	72	00602
2	ZCTA5 00603	47081.0	16049.0	42.1	19228.0	11259.0	11479.0	10460.0	23691.0	72	00603
3	ZCTA5 00606	6392.0	12119.0	44.3	1946.0	6093.0	1228.0	1085.0	4185.0	72	00606
4	ZCTA5 00610	26686.0	19898.0	42.7	8795.0	10572.0	8065.0	6800.0	12204.0	72	00610

Rename & Add Columns

```
# Column Reordering
census_pd = census_pd.rename(columns={"NAME": "State ID",
                                      "state": "State ID",
                                      "B01003_001E": "Population",
                                      "B11001_001E": "Number of Households",
                                      "B19013_001E": "Household Income",
#                                       "B19301_001E": "Per Capita Income",
                                      "B17001_002E": "Poverty Count",
                                      "B08303_001E": "Total Commute Time",
                                      "B08015_001E": "Total Vehicles",
                                      "NAME": "Name", "zip code tabulation area": "Zipcode"})

census_pd.head()
```

	State ID	Zipcode	Population	Number of Households	Household Income	Poverty Count	Poverty Rate	Total Commute Time	Average Commute Time	Total Vehicles	Average Vehicles per Household
0	72	00601	17113.0	5509.0	14361.0	10552.0	61.660726	3504.0	0.636050	3115.0	0.565438
1	72	00602	37751.0	12740.0	16807.0	18653.0	49.410612	10525.0	0.826138	8645.0	0.678571
2	72	00603	47081.0	19228.0	16049.0	23691.0	50.319662	11479.0	0.596994	10460.0	0.543998
3	72	00606	6392.0	1946.0	12119.0	4185.0	65.472466	1228.0	0.631038	1085.0	0.557554
4	72	00610	26686.0	8795.0	19898.0	12204.0	45.731844	8065.0	0.916998	6800.0	0.773167

```
census_pd["Poverty Rate"] = 100 * \
    census_pd["Poverty Count"].astype(
        int) / census_pd["Population"].astype(int)

census_pd["Average Vehicles per Household"] = census_pd["Total Vehicles"] / census_pd["Number of Households"]

census_pd["Average Commute Time"] = census_pd["Total Commute Time"] / census_pd["Number of Households"]

census_pd = census_pd[["State ID", "Zipcode", "Population", "Number of Households",
                       "Household Income", "Poverty Count", "Poverty Rate",
                       "Total Commute Time", "Average Commute Time",
                       "Total Vehicles", "Average Vehicles per Household"]]

# Visualize
print(len(census_pd))
census_pd.head()
```

```
33120
```

	State ID	Zipcode	Population	Number of Households	Household Income	Poverty Count	Poverty Rate	Total Commute Time	Average Commute Time	Total Vehicles	Average Vehicles per Household
0	72	00601	17113.0	5509.0	14361.0	10552.0	61.660726	3504.0	0.636050	3115.0	0.565438
1	72	00602	37751.0	12740.0	16807.0	18653.0	49.410612	10525.0	0.826138	8645.0	0.678571
2	72	00603	47081.0	19228.0	16049.0	23691.0	50.319662	11479.0	0.596994	10460.0	0.543998
3	72	00606	6392.0	1946.0	12119.0	4185.0	65.472466	1228.0	0.631038	1085.0	0.557554
4	72	00610	26686.0	8795.0	19898.0	12204.0	45.731844	8065.0	0.916998	6800.0	0.773167

Sort by State

```
states = ["04", "06", "08", "16", "30", "32", "35", "41", "49", "53", "56"]

state_name = {"04": "AZ",
              "06": "CA",
              "08": "CO",
              "16": "ID",
              "30": "MT",
              "32": "NV",
              "35": "NM",
              "41": "OR",
              "49": "UT",
              "53": "WA",
              "56": "WY"}

state_census_df = census_pd.loc[census_pd["State ID"].isin (states)]
state_census_df

state_census_df["State"] = [state_name[state] for state in state_census_df["State ID"]]
state_census_df
```

```
<ipython-input-9-43694d56ade5>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  state_census_df["State"] = [state_name[state] for state in state_census_df["State ID"]]
```

	State ID	Zipcode	Population	Number of Households	Household Income	Poverty Count	Poverty Rate	Total Commute Time	Average Commute Time	Total Vehicles	Average Vehicles per Household	State
20057	30	59001	1558.0	734.0	58482.0	14.0	0.898588	679.0	0.925068	545.0	0.742507	MT
20058	30	59002	127.0	47.0	37917.0	33.0	25.984252	52.0	1.106383	45.0	0.957447	MT

3330 rows x 12 columns

```
]: state_census_df.to_csv("state_census_df.csv", encoding="utf-8", index=False)
```

NREL Request

In [1]:
```
# Dependencies
import requests
import json
import pandas as pd
import numpy as np
import sklearn.datasets as dta
import scipy.stats as st
from config import get_key
```

In [ ]:
```
hopes_awesome_file = pd.read_csv("../Project Jupyter Notebook Files/resources/state_census_df.csv")
hopes_awesome_file.head()
```

In [3]:
```
# URL for GET requests to retrieve EV charging station data
url = f"https://developer.nrel.gov/api/alt-fuel-stations/v1.json?api_key={get_key}&fuel_type=ELEC&state=WA,OR,CA,ID,UT,
      NV,AZ,WY,MT,NM,CO&limit=all"
```

In [ ]:
```
# Pretty print JSON for all EV charging stations in WA,OR,CA,ID,UT,NV,AZ,WY,MT,NM and CO
response = requests.get(url).json()
print(json.dumps(response, indent=4, sort_keys=True))
```

In [5]:
```
#Create lists to cull out Zipcode, Latitude and Longitude response info:

stations = response['fuel_stations']

state = []
city = []
zip = []
latitude = []
longitude = []

for station in stations:

    state.append(station['state'])
    city.append(station['city'])
    zip.append(station['zip'])
    latitude.append(station['latitude'])
    longitude.append(station['longitude'])
```

Create & Merge Datasets

In [6]:
```
#Create a data frame from State, Zipcode, Latitude and Longitude:

EVstation_dict = {
                  "State": state,
                  "city": city,
                  "Zipcode": zip,
                  "latitude": latitude,
                  "longitude": longitude
                  }

EVstation_data = pd.DataFrame(EVstation_dict)
EVstation_data.head()
```

Out[6]:

	State	city	Zipcode	latitude	longitude
0	CA	Sun Valley	91352	34.248319	-118.387971
1	CA	Los Angeles	90024	34.052542	-118.448504
2	CA	Rosemead	91770	34.050745	-118.081014
3	CA	Los Angeles	90015	34.040539	-118.271387
4	CA	Los Angeles	90012	34.059133	-118.248589

In [7]:
```
len(EVstation_data)
```

Out[7]:
```
20562
```

In [8]:
```
EVstation_data.dtypes
EVstation_data['Zipcode']=EVstation_data['Zipcode'].astype(int)
```

In [9]:
```
Evmerged = pd.merge(hopes_awesome_file,EVstation_data,how = "inner", on="Zipcode")
```

Working with Large Datasets

```
In [9]: # Pretty print JSON for all EV charging stations in WA,OR,CA,ID,UT,NV,AZ,WY,MT,NM and CO
        response = requests.get(url).json()
        print(json.dumps(response, indent=4, sort_keys=True))
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

Solution:

jupyter notebook --NotebookApp.iopub_data_rate_limit=1.0e10

Industry Best Practices

Importing & Sharing Large Files

Importing

Import locally, especially large datasets

Git LFS - GitHub

Git LFS - large file storage
GitHub - primarily used to share code

Sharing

Files can be shared via Google Drive

Data by Zipcode

In [6]:
```
#Create a data frame from State, Zipcode, Latitude and Longitude:

EVstation_dict = {
                  "State": state,
                  "city": city,
                  "Zipcode": zip,
                  "latitude": latitude,
                  "longitude": longitude
                  }

EVstation_data = pd.DataFrame(EVstation_dict)
EVstation_data.head()
```

Out[6]:

	State	city	Zipcode	latitude	longitude
0	CA	Sun Valley	91352	34.248319	-118.387971
1	CA	Los Angeles	90024	34.052542	-118.448504
2	CA	Rosemead	91770	34.050745	-118.081014
3	CA	Los Angeles	90015	34.040539	-118.271387
4	CA	Los Angeles	90012	34.059133	-118.248589

In [7]:
```
len(EVstation_data)
```

Out[7]: 20562

In [8]:
```
EVstation_data.dtypes
EVstation_data['Zipcode']=EVstation_data['Zipcode'].astype(int)
```

In [9]:
```
Evmerged = pd.merge(hopes_awesome_file,EVstation_data,how = "inner",
```

In [12]:
```
# Income Grouped by Zipcode

income_zipcode = census_data[["Zipcode", "Household Income"]]
income_zipcode
```

Out[12]:

	Zipcode	Household Income
0	59001	58482.0
1	59002	37917.0
2	59003	46838.0
3	59006	44750.0
4	59007	105625.0
...	...	...
5351	99363	38011.0
5352	99371	53750.0
5353	99401	67917.0
5354	99402	55595.0
5355	99403	53079.0

5356 rows × 2 columns

Merge & Drop

In [15]:
```
# Merged Income and Charge Zipcodes

merged_income_chargers = pd.merge(zipcode_df, income_zipcode, on="Zipcode", how="left")
merged_income_chargers.head()
```

Out[15]:

	Zipcode	counts	Household Income
0	00987	1	NaN
1	55301	1	NaN
2	59011	1	49754.0
3	59020	1	36875.0

In [17]:
```
# Dropping NA Datapoints

clean_income_chargers = merged_income_chargers.dropna()
clean_income_chargers
```

Out[17]:

	Zipcode	counts	Household Income
2	59011	1	49754.0
3	59020	1	36875.0
4	59024	1	53136.0
5	59030	1	51694.0
6	59035	1	47292.0
...	...	...	...
2418	99352	6	81410.0
2419	99354	46	73369.0
2420	99361	1	69118.0
2421	99362	15	56665.0
2422	99403	1	53079.0

Sort & Clean

In [19]:
```
# Sorted Clean Dataframe without NA
sorted_clean_income_chargers = clean_income_chargers.sort_values("Household Income", ascending=False)
sorted_clean_income_chargers
```

Out[19]:

	Zipcode	counts	Household Income
1490	94027	3	250001.0
1488	94024	6	250001.0
1491	94028	3	234091.0
1487	94022	28	223859.0
1881	95837	4	219063.0
...	...	...	...
1986	97135	1	-666666666.0
245	81330	1	-666666666.0
1019	91608	16	-666666666.0
978	91330	8	-666666666.0
752	89425	1	-666666666.0

In [20]:
```
# Dropped Household incomes less than $0

clean_income_chargers_v2 = clean_income_chargers[clean_income_chargers["Household Income"] > 0]

clean_income_chargers_v2
```

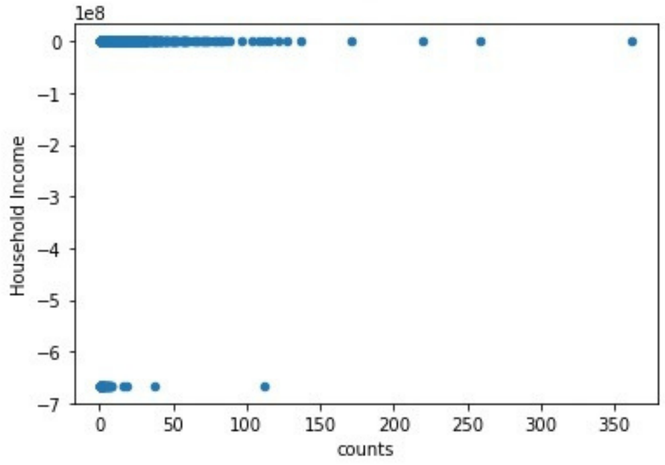
Out[20]:

	Zipcode	counts	Household Income
2	59011	1	49754.0
3	59020	1	36875.0
4	59024	1	53136.0
5	59030	1	51694.0
6	59035	1	47292.0
...	...	...	...
2418	99352	6	81410.0
2419	99354	46	73369.0
2420	99361	1	69118.0
2421	99362	15	56665.0
2422	99403	1	53079.0

Excluding Data

```
In [21]: #Scatter Plot v1 Including incomes less than 0

clean_income_chargers.plot.scatter(x = "counts",
                                   y = "Household Income",)
```

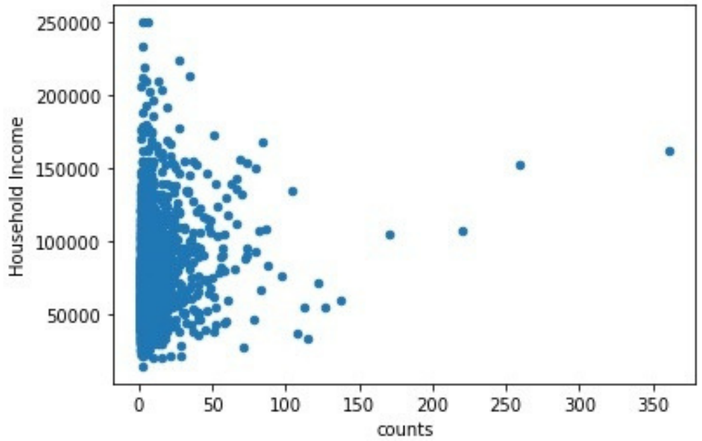
```
Out[21]: <AxesSubplot:xlabel='counts', ylabel='Household Income'>
```



Scatterplot including income less than zero

```
In [22]: #Scatter Plot v2 Household Incomes Greater than 0

clean_income_chargers_v2.plot.scatter(x = "counts",
                                      y = "Household Income",)
```

```
Out[22]: <AxesSubplot:xlabel='counts', ylabel='Household Income'>
```



Scatterplot including only income greater than zero

Regression & Correlation

In [24]:
```
# Regression and Correlation:

correlation = st.pearsonr(x_values,y_values)
print(f"The correlation between both factors is {round(correlation[0],2)}")
```

```
The correlation between both factors is 0.19
```

In [26]:
```
# Poverty Count and Zipcode

zipcode_poverty = census_data[["Zipcode", "Poverty Count"]]
zipcode_poverty.head()
```

Out[26]:

	Zipcode	Poverty Count
0	59001	14.0
1	59002	33.0
2	59003	362.0
3	59006	377.0
4	59007	13.0

Merge & Clean

In [29]:
```
# Merged Poverty Count and Charger Zipcode

merged_poverty_charger = pd.merge(zipcode_df, zipcode_poverty, on="Zipcode", how="left")
merged_poverty_charger
```

Out[29]:

	Zipcode	counts	Poverty Count
0	00987	1	NaN
1	55301	1	NaN
2	59011	1	179.0
3	59020	1	0.0
4	59024	1	35.0
...	...	...	...
2422	99403	1	2656.0
2423	89423	1	NaN
2424	90032	1	NaN
2425	94583	1	NaN
2426	95446	1	NaN

In [30]:
```
# Cleaned Merged Poverty Count and Charger Zipcode

clean_merged_poverty_charger = merged_poverty_charger.dropna()
clean_merged_poverty_charger
```

Out[30]:

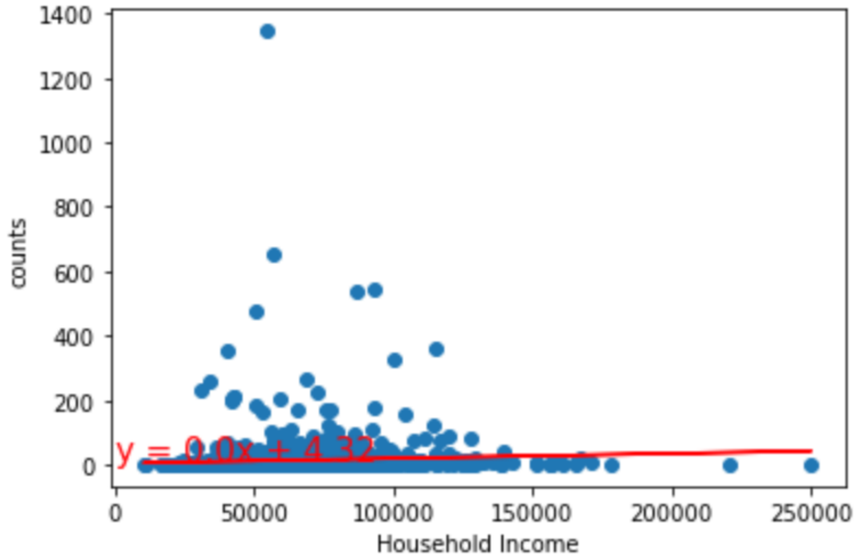
	Zipcode	counts	Poverty Count
2	59011	1	179.0
3	59020	1	0.0
4	59024	1	35.0
5	59030	1	140.0
6	59035	1	28.0
...	...	...	...
2418	99352	6	1640.0
2419	99354	46	3276.0
2420	99361	1	210.0
2421	99362	15	4683.0
2422	99403	1	2656.0

In [32]:
```
# Correlation

correlation = st.pearsonr(x2_values,y2_values)
print(f"The correlation between both factors is {round(correlation[0],2)}")
```

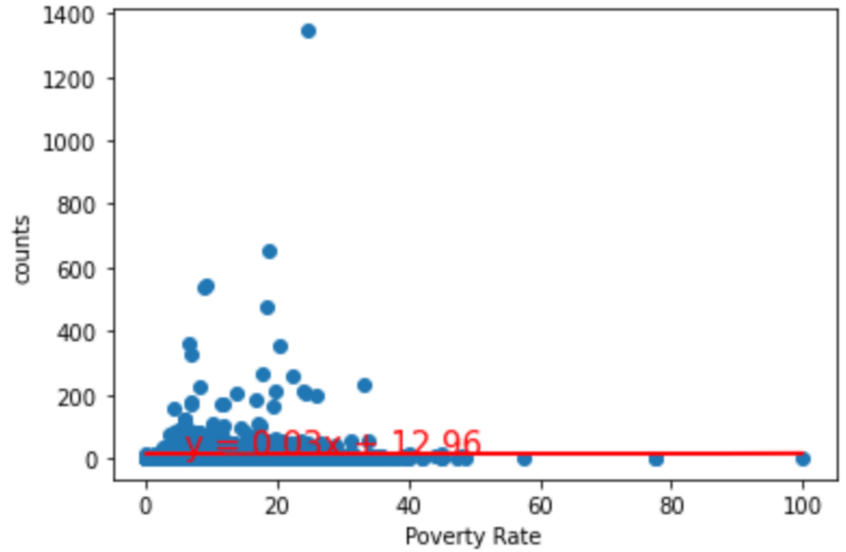
```
The correlation between both factors is 0.09
```

Stations Locations v. Income Level

R squared: 0.004845489703172146
std dev: 5.5921526961077776e-05

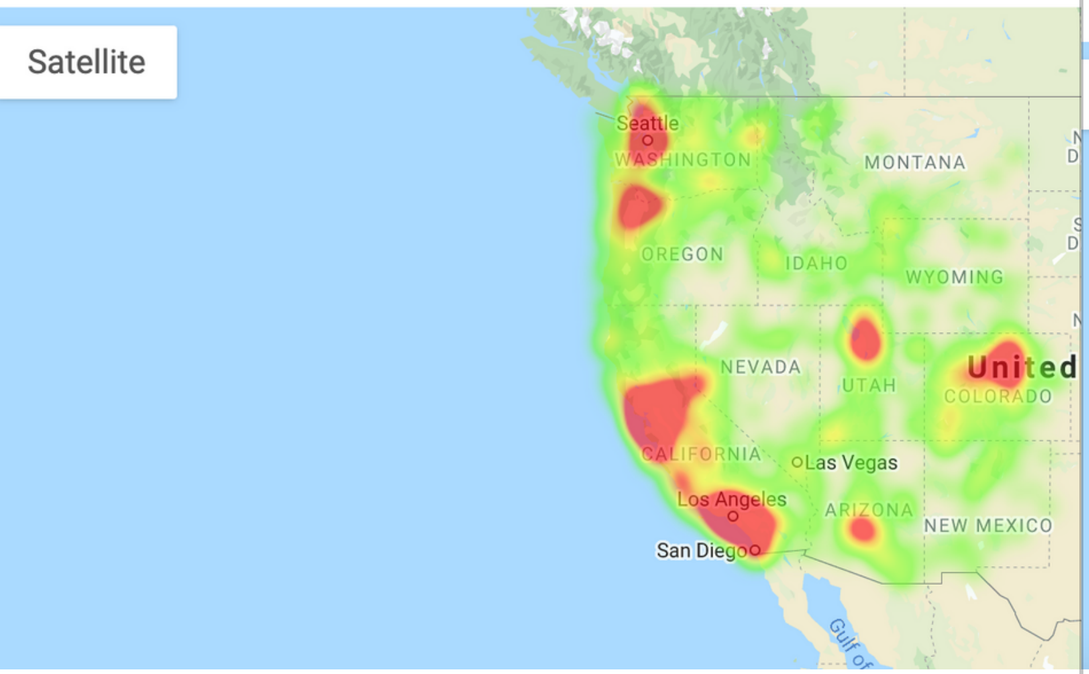


R squared: 2.6213348491923447e-05
std dev: 0.15986257328877568



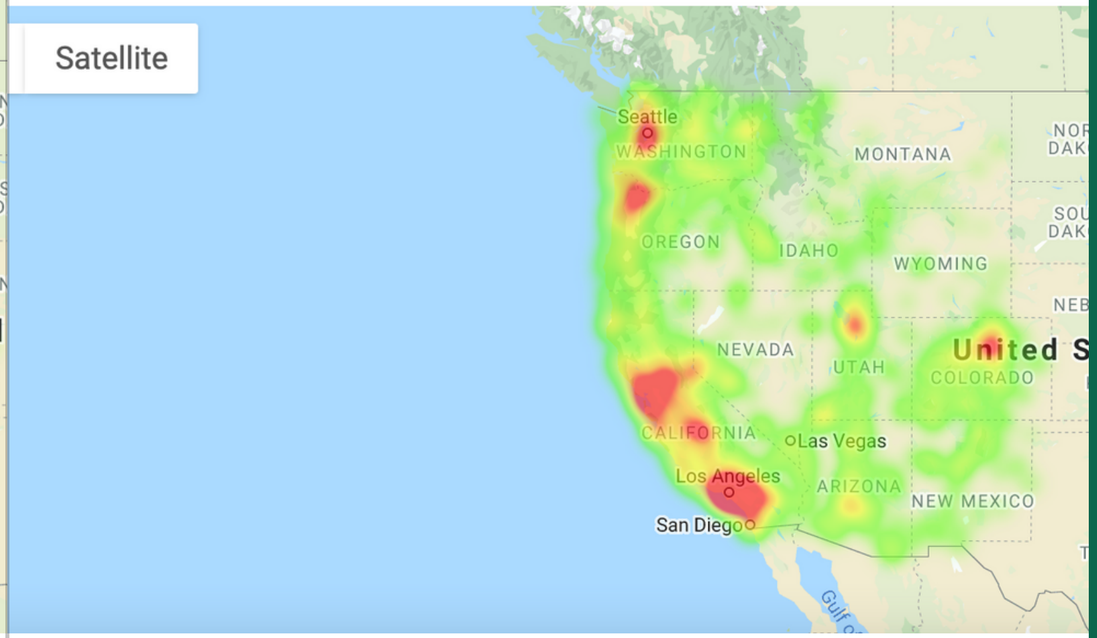
Economic Heatmaps

```
os.figure()

yer = gmaps.heatmap_layer(locations, weights=household_income/10000,
                          dissipating=False, max_intensity=100,
                          point_radius = 1)

ayer(income_layer)
```



Income Heatmap

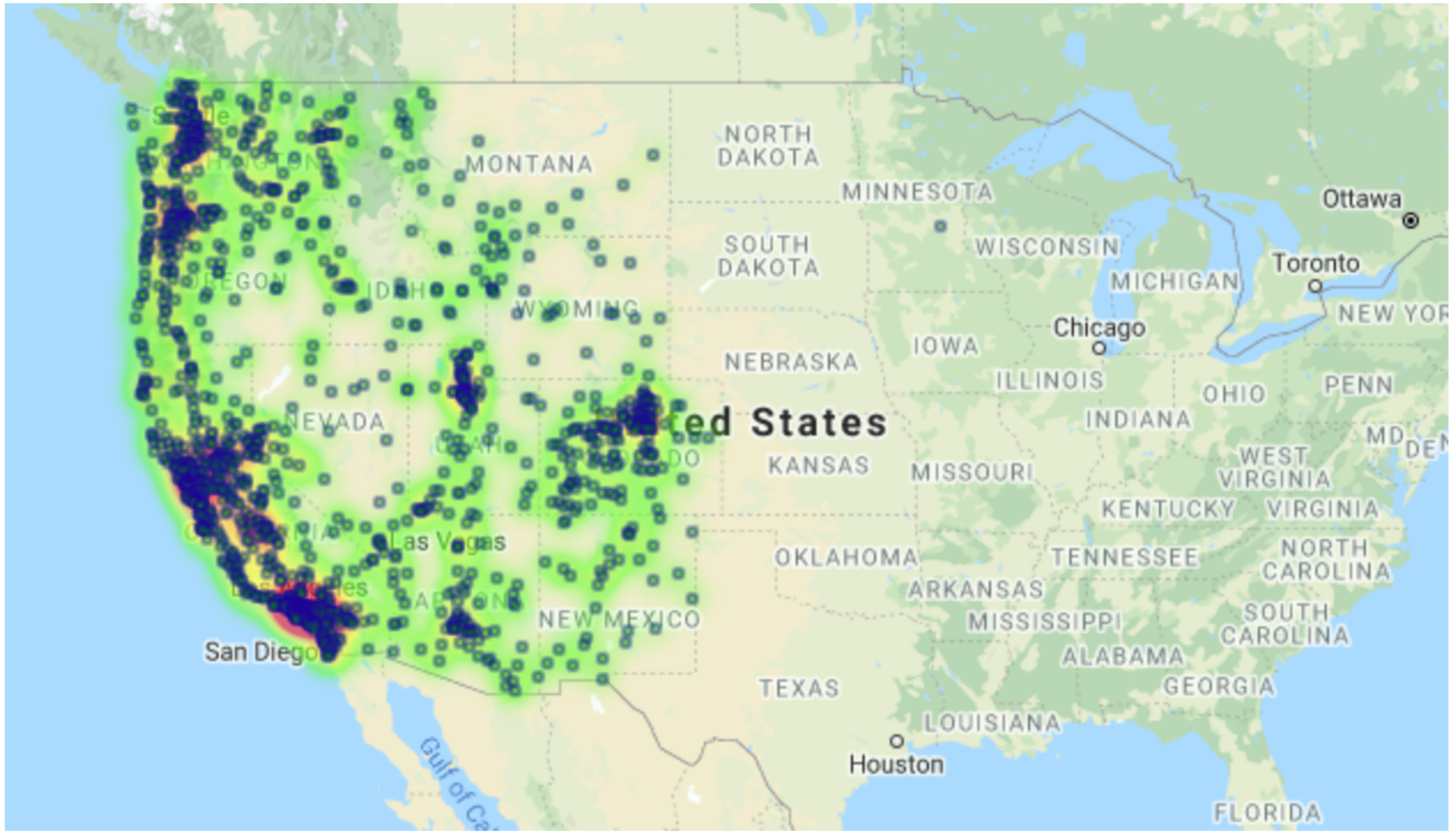
```
a poverty heatmap layer
s = census_data_complete[["latitude", "longitude"]].astype(float)
aps.figure()

er = gmaps.heatmap_layer(locations, weights=poverty_rate,
                         dissipating=False, max_intensity=400,
                         point_radius = 1)

layer(heat_layer)
```



Poverty Heatmap

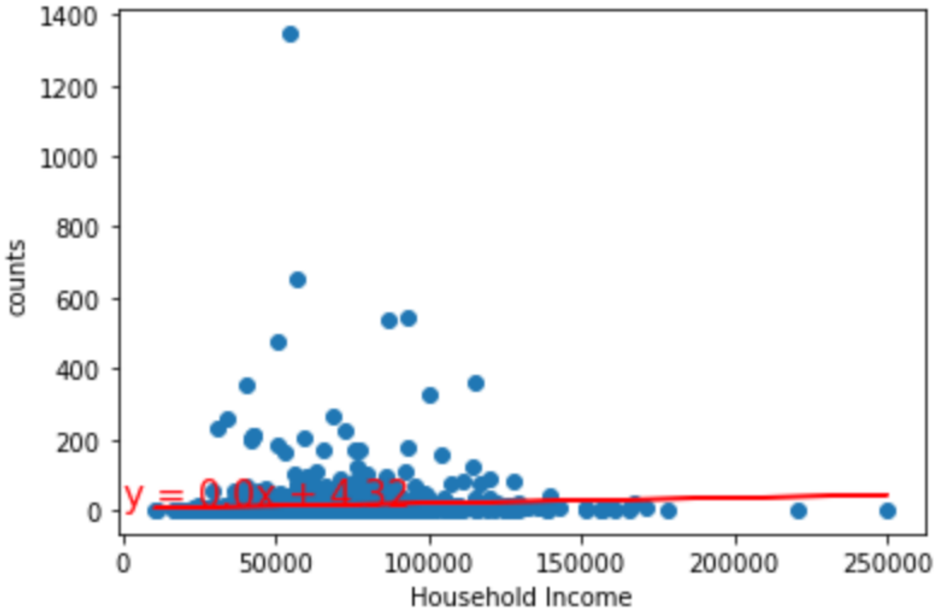
EV Station Locations

	State	counts
1	CA	13899
9	WA	1686
2	CO	1489
7	OR	939
8	UT	845
0	AZ	803
6	NV	449
5	NM	171
3	ID	136
4	MT	70
10	WY	64



Scatterplot and Regression

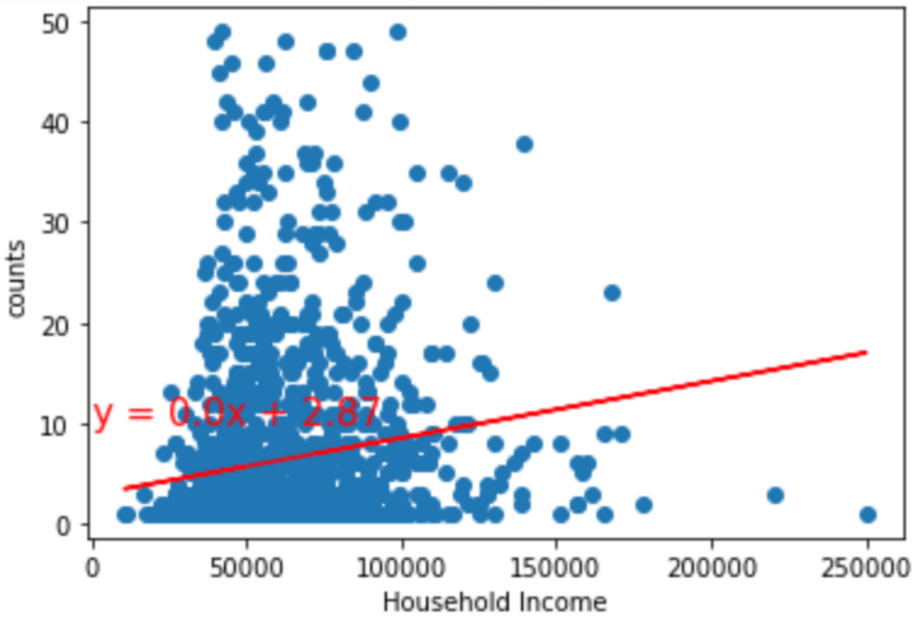
```
R squared: 0.004845489703172146
std dev: 5.5921526961077776e-05
```



Household Income vs. Counts

```
R squared: 0.025223423396642068
                                   -06
```

click to scroll output; double click to hide



Stations < 50 per zipcode

Results

Where are EV stations located?

CA has a greater number of EV stations than the other states studied

Geographically, what does that look like?

EV stations tend to be more concentrated around metro areas

Economically, what does that look like?

There was no correlation between EV stations and household income