yuelinzhou@berkeley.edu-zhangdean@berkeley.edu-wealth-final-project

August 13, 2020

[1]:
```
# modules for research report
from datascience import *
import numpy as np
import random
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')

# module for YouTube video
from IPython.display import YouTubeVideo

# okpy config
from client.api.notebook import Notebook
ok = Notebook('wealth-final-project.ok')
_ = ok.auth(inline=True)
```

```
=====================================================================
Assignment: Final Project: Household Wealth in the United States
OK, version v1.12.5
=====================================================================

Successfully logged in as zhangdean@berkeley.edu
```

1 Family & Household Wealth in the United States (2009-2013)

This dataset was originally published as part of household-specific data from the American Community Survey (ACS) — a yearly, ongoing survey conducted by the U.S. Census Bureau — Public Use Microdata Sample (PUMS). This dataset is taken from the ACS 5-year PUMS, spanning the years 2009-2013, and it has been cleaned for your convenience. All observations and variables not of interest have been removed, and a random sample of the data containing 1,200 entries has been provided. A brief description of the dataset is provided below.

NB: You may not copy any public analyses of this dataset. Doing so will result in a zero.

1.1 Summary

The origins of the ACS can be traced back to the mid-20th century. That is, the postwar period in the United States that saw massive population growth and swiftly changing household, urban and rural demographics. Beginning in the 1960s, lawmakers, unable to get actionable data about their rapidly changing communities from the once-every-ten-years Census, started to look for ways to get more immediate information about the people in their districts.

It wasn't until the 1990s, however, that plans to get more frequent Census-type data came to fruition. Congress, seeing a drop in Census response rates as a result of its burdensome length, directed the U.S. Census Bureau to develop different ways to get the much needed information. The U.S. Census Bureau, in concert with statisticians from other organizations, eventually developed a yearly survey now known as the ACS. The ACS was officially launched in 2005.

According to the U.S. Census Bureau website: >"American Community Survey (ACS) data are tabulated for a variety of different geographic areas ranging in size from broad geographic regions to cities, towns, county subdivisions, and block groups."

1.2 Disclaimer

At the time this data was collected, the U.S. Census Bureau and the ACS only considered binary, opposite-sex couples in the context of marital status; any time this dataset mentions a married pair or a spouse, it is referring to an opposite-sex couple or partner. This dataset covers the period 2009-2013, when same-sex marriage had not yet been legalized under U.S. federal law, though some states maintained legal same-sex marriage during this time period.

1.3 Data Description

This dataset utilizes the following abbreviations: * `GQ`: group quarters * `Non-Family`: a household that is not associated with any family. If referring to an individual, the respondent is considered a member of group quarters. If referring to a property, the property is considered vacant.

This dataset contains three tables, included in the `data` folder: 1. `families_data` provides information about characteristics of each household. 2. `resources_data` provides information about the resources available to each household. 3. `states_data` provides the full names and abbreviations of U.S. states as strings, as well as an integer code for each state that corresponds to the integer codes found in the previous two tables.

A description of each table's variables is provided below: 1. `families data`: * `ID`: a unique identifier for each household * `REGION`: region of the United States * `DIVISION`: division of the United States, more specific than region * `STATE`: state of the United States * `FAMILY INCOME`: yearly income by family, not adjusted for inflation * `HOUSEHOLD LANGUAGE`: description of the geographic area from which the main household language is originally. The categories for Household Language are as follows: English only, and Other Non-English. * `HOUSEHOLD INCOME`: yearly income by household, not adjusted for inflation * `WORKERS IN FAMILY`: number of workers in each family * `PERSONS IN FAMILY`: number of persons in each family

2. `resources_data`:
    - `ID`: a unique identifier for each household
    - `REGION`: region of the United States
    - `DIVISION`: division of the United States, more specific than region

- `STATE`: state of the United States
- `MONTHLY RENT`: the monthly rent each renting household is paying. If the property is owned by the household, this value is the string "Owner".
- `GROSS MONTHLY RENT`: the gross rent (monthly amount) each renting household is paying. If the property is owned by the household, this value is the string "Owner".
- `OCCUPANCY STATUS`: description of the occupancy status for a particular property; for example: "owned free and clear" or "rented."
- `NUMBER OF VEHICLES`: number of vehicles a particular household has access to.
- `HOUSEHOLD TELEPHONE`: a binary variable, whether or not a household has access to a telephone.
- `PROPERTY VALUE`: The value of property in dollars ($).

3. `states_data`:
   - `CODE`: number for reference in original table
   - `FULL NAME`: full state name
   - `ABRV`: abbreviation

1.4 Note about Non-Family Values

Many values in the table below are categorized as "non-family" if the census respondent is not part of a family unit (e.g. the respondent resides in group quarters). Our data contains a lot of family-specific data, like `FAMILY INCOME` or `PERSONS IN FAMILY`; it wouldn't make sense for a person who is not part of a family to have responses to those.

Non-family respondents, however, account for a significant portion of census data — roughly between 15%-30%, depending on the sample. If you would like to work with the non-family respondent data, we encourage you to consider non-family-dependent variables, like `HOUSEHOLD INCOME` or `PROPERTY VALUE`, among others. If, however, you do not want to work with these variables, we encourage you to filter out these variables (in a process called "data cleaning") from your data before you get started.

Hint: if you want to clean your data so that non-family values do not appear, consider filtering using `.where`.

1.5 Inspiration

A variety of exploratory analyses, hypothesis tests, and predictions problems can be tackled with this data. Here are a few ideas to get you started:

1. Is there a relationship between property value and English-only-speaking households?
2. Is there a significant difference in monthly rent for households in the West region of the United States compared to the Northeast region?
3. How do rows containing 'Non-Family' data compare to responses filled by heads of Families?
4. Where is household telephone access limited? How is this associated with various measures of wealth?
5. What states and regions have higher or lower vehicle ownership? See `NUMBER OF VEHICLES`.

If you'd like to learn more about wealth in the United States, check out the following resources: 1. Where does your net worth rank in the United States? Data visualization by the *New York Times*. 2. Consider taking Wealth & Poverty (PUBPOL C103) with former U.S. Secretary of Labor and current UC Berkeley professor Robert Reich. 3. Consider taking Contemporary Theories of

Political Economy (POLECON 101) with Professor Khalid Kadir (it's Data 8 GSI Maya's favorite class at Berkeley!).

Credit to Prof. Lexin Li, and his course Big Data: A Public Health Perspective (PBHLTH 244), for introducing the Data 8 staff to this dataset. Feel free to ask him about your path to upper division and graduate level biostatistics courses at UC Berkeley.

Don't forget to review the Final Project Guidelines for a complete list of requirements.

1.6 Preview

[2]:
```
families_data = Table.read_table('data/families.csv').relabel(4, 'FAMILY␣
↪INCOME').relabel(6, 'HOUSEHOLD INCOME')
families_data
```

[2]:
```
ID            | REGION | DIVISION           | STATE | FAMILY INCOME | HOUSEHOLD
LANGUAGE | HOUSEHOLD INCOME | WORKERS IN FAMILY | PERSONS IN FAMILY
2009000089236 | South  | East South Central | 1     | Non-Family    | English
only       | 9.81             | Non-Family        | Non-Family
2009000311177 | South  | East South Central | 1     | 10.44         | English
only       | 10.44            | 1                 | 2
2009000426047 | South  | East South Central | 1     | Non-Family    | English
only       | 9.88             | Non-Family        | Non-Family
2009000448671 | South  | East South Central | 1     | Non-Family    | English
only       | 11.53            | Non-Family        | Non-Family
2009000523654 | South  | East South Central | 1     | Non-Family    | English
only       | 10.12            | Non-Family        | Non-Family
2009000742214 | South  | East South Central | 1     | 12.37         | English
only       | 12.37            | 1                 | 2
2009001335518 | South  | East South Central | 1     | 11.15         | English
only       | 11.15            | 3+                | 3
2010000145521 | South  | East South Central | 1     | 10.11         | English
only       | 10.11            | 0                 | 4
2010000560424 | South  | East South Central | 1     | 11.78         | English
only       | 11.78            | 2                 | 3
2010000853150 | South  | East South Central | 1     | Non-Family    | English
only       | 11.69            | Non-Family        | Non-Family
… (1190 rows omitted)
```

[3]:
```
resources_data = Table.read_table('data/resources.csv').relabel(4, 'PROPERTY␣
↪VALUE').relabel(5, 'MONTHLY RENT').relabel(6, 'GROSS MONTHLY RENT')
resources_data
```

[3]:
```
ID            | REGION | DIVISION           | STATE | PROPERTY VALUE | MONTHLY
RENT | GROSS MONTHLY RENT | OCCUPANCY STATUS | NUMBER OF VEHICLES | HOUSEHOLD
TELEPHONE
2009000089236 | South  | East South Central | 1     | 10860          | Owner
| Owner              | Owned            | 1                  | True
```

```
2009000311177 | South  | East South Central | 1     | 43441          | Owner
| Owner              | Owned with loan  | 2                  | True
2009000426047 | South  | East South Central | 1     | Renter         | 238
| 238                | Rented           | 0                  | True
2009000448671 | South  | East South Central | 1     | Renter         | 543
| 705                | Rented           | 2                  | True
2009000523654 | South  | East South Central | 1     | 21720          | Owner
| Owner              | Owned            | 2                  | True
2009000742214 | South  | East South Central | 1     | 141184         | Owner
| Owner              | Owned with loan  | 1                  | True
2009001335518 | South  | East South Central | 1     | 32580          | Owner
| Owner              | Owned with loan  | 0                  | True
2010000145521 | South  | East South Central | 1     | 74787          | Owner
| Owner              | Owned with loan  | 2                  | True
2010000560424 | South  | East South Central | 1     | 347228         | Owner
| Owner              | Owned with loan  | 3                  | True
2010000853150 | South  | East South Central | 1     | 48077          | Owner
| Owner              | Owned            | 1                  | True
… (1190 rows omitted)
```

[4]:
```
states_data = Table.read_table('data/states.csv')
states_data
```

[4]:
```
CODE | FULL NAME            | ABRV
1    | Alabama              | AL
2    | Alaska               | AK
4    | Arizona              | AZ
5    | Arkansas             | AR
6    | California           | CA
8    | Colorado             | CO
9    | Connecticut          | CT
10   | Delaware             | DE
11   | District of Columbia | DC
12   | Florida              | FL
… (42 rows omitted)
```

2 Research Report

2.1 Introduction

We have analyzed the Family & Household Wealth in the United States dataset (2009-2013) for our final project. This dataset describes information of any family and household in the period of 2009-2013 from a random sample of 1200 families and households in the US. This data was collected by the U.S. Census Bureau and the American Community Survey (ACS). The dataset is divided into three tables: `families_data`, `resources_data` and `states_data`.

The `families_data` table provides information on characteristics of each household in the United States. For example, we can roughly pinpoint the location of a particular household, see statistics

about the family, and what their predominant spoken language at home is. In particular, we are interested in "Household Language", "State" and "Region". Households can either speak English, some other language, or give no response, which will be marked as "non-family" in the data. The "State" and "Region" variables give us more detail about a household's location.

The `resources_data` table provides information on the overall resources that are available to each household. This table keeps track of notable additional variables such as rent, terms of occupancy, property value, etc. This table also keeps track of the location of households, which we want to continue making note of. But we are particularly interested in "Property Value" and "Number of Vehicles" as this will allow us to determine the value of the property of a particular household as well as how many vehicles a particular household owns.

Lastly, we have the `states_data` table, which helps us identify state names. We noticed that a common characteristic between all three tables was location. We can use the "State" variable in tables `families_data` and `resources_data` and the "Code" variable in `states_data` to link the three tables together. This way, we can answer more ideas, test hypothesis, and be able to make accurate predictions.

2.2 Hypothesis Testing and Prediction Questions

Please bold your hypothesis testing and prediction questions.

By combining the information contained in all three tables of this dataset, we are interested in studying if a possible relationship exists between property value and the spoken language of American households. We also want to examine whether certain households in different states and regions have higher or lower vehicle ownership.

In particular, we want to find out if the property value of a household will be significantly different if the household speaks a language other than English as its primary language. Our null hypothesis is that the relationship between property value for non-English-speaking households and English-speaking households is the same, and that any differences is due to chance. Our alternative hypothesis is that the households whose primary language is not English will have a significantly different property value than households whose primary language is English.

We also want to determine which states and regions of households will have higher or lower vehicle ownership. We will use the K-Nearest Neighbors algorithm to tackle this problem.

2.3 Exploratory Data Analysis

You may change the order of the plots and tables.

Table Requiring a Join Operation:

First, we will join `families_data`, `resources_data`, and `states_data` in order to create the `full_data` table, which is the aggregate of the entire dataset.

Now, we will use the information in `full_data` in order to pick two variables out: "PROPERTY VALUE" and "HOUSEHOLD LANGUAGE". We do this so that we can see whether or not there is a relationship between property value and English-only-speaking households. Further analysis will be made in the *Hypothesis Testing* section.

We have decided to remove "Renters" and "Non-Family" from the dataset because a renter does not tell us anything about a household's property value. Some fair assumptions might be that a renter might be renting from a low property value residence, but this is not safe to assume in our analysis. "Renters" is a categorical variable, which is very different from a low, numeric property value. Also, "Non-Family" indicates that a household has no response, which again, tells us nothing about property value or a household's spoken language.

[5]:
```
# Use this cell to join datasets
full_data = families_data.join("ID", resources_data).join("STATE", states_data,␣
 ↪"CODE")

# Datasets used for A/B Testing
hypothesis_data = full_data.select("PROPERTY VALUE","HOUSEHOLD LANGUAGE")

# We need to eliminate all non-family and renters from the table.
hypothesis_data_filtered = hypothesis_data.where("PROPERTY VALUE", are.
 ↪not_equal_to("Non-Family")).where("HOUSEHOLD LANGUAGE",are.
 ↪not_equal_to("Non-Family")).where("PROPERTY VALUE", are.
 ↪not_equal_to("Renter"))

# We need to turn the PROPERTY VALUE column into ints (they're in strings right␣
 ↪now)
hypothesis_data_filtered = hypothesis_data_filtered.with_column(
    "PROPERTY VALUE", hypothesis_data_filtered.apply(int, "PROPERTY VALUE")
)
# Finally, create the desired grouped table for hypothesis question
hyp_grouped = hypothesis_data_filtered.group("HOUSEHOLD LANGUAGE", np.mean)
hyp_grouped
```

[5]:
```
HOUSEHOLD LANGUAGE | PROPERTY VALUE mean
English only       | 244086
Other Non-English  | 283217
```

Aggregated Data Table:

We want to keep track of location and the number of vehicles for each household so we can use k-nearest neighbors to answer our prediction question. This information won't be used until the *Prediction* section, but nonetheless, this information is still useful.

Again, in the "Number of Vehicles" column, some people chose to not respond, which is indicated as "Non-Family". This will be filtered out to make analyzing the prediction model easier. We have also decided to include "Property Value" as a feature for our k-nearest neighbors classifier. This column requires filtering out "Renters" and "Non-Family", as explained above.

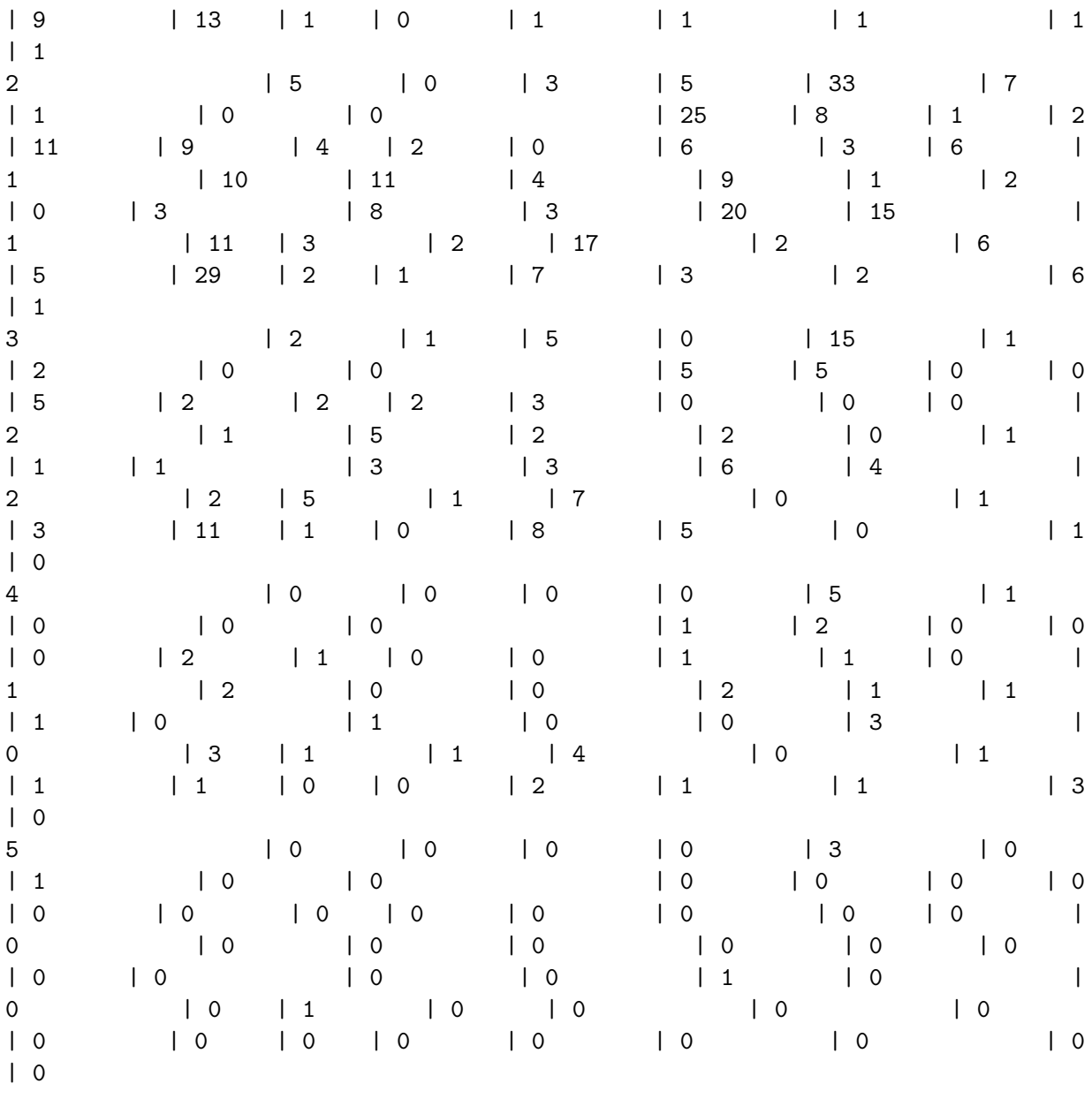
NOTE: There was one household in Missouri, Midwest with a property value of $182,615 that had 6+ vehicles, and there was no easy way to convert that to an int, so we have removed this household from the dataset. Considering that this is one household out of 720, removing this household should have little to no impact on our classifier.

[6]:
```
# Below code is for k-nearest neighbors
prediction_data = full_data.select("STATE", "FULL NAME", "REGION", "NUMBER OF␣
 ↪VEHICLES", "PROPERTY VALUE", "ID").where("NUMBER OF VEHICLES", are.
 ↪not_equal_to("Non-Family")).where("PROPERTY VALUE", are.
 ↪not_equal_to("Renter")).where("PROPERTY VALUE", are.
 ↪not_equal_to("Non-Family"))
prediction_data = prediction_data.where("NUMBER OF VEHICLES", are.
 ↪not_equal_to("6+"))
prediction_data_filtered = prediction_data.with_columns(
    "STATE", prediction_data.apply(int, "STATE"),
    "NUMBER OF VEHICLES", prediction_data.apply(int, "NUMBER OF VEHICLES"),
    "PROPERTY VALUE", prediction_data.apply(int, "PROPERTY VALUE"),
    "ID", prediction_data.column("ID")
)
prediction_pivot = prediction_data_filtered.pivot("FULL NAME", "NUMBER OF␣
 ↪VEHICLES")
prediction_pivot
```


```
/opt/conda/lib/python3.6/site-packages/datascience/tables.py:496:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
  values = np.array(tuple(values))
```


[6]:

NUMBER OF VEHICLES	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connecticut	Delaware	District of Columbia	Florida	Georgia	Hawaii	Idaho	Illinois	Indiana	Iowa	Kansas	Kentucky	Louisiana	Maine	Maryland	Massachusetts	Michigan	Minnesota	Mississippi	Missouri	Montana	Nebraska	Nevada	New Hampshire	New Jersey	New Mexico	New York	North Carolina	North Dakota	Ohio	Oklahoma	Oregon	Pennsylvania	Rhode Island	South Carolina	Tennessee	Texas	Utah	Vermont	Virginia	Washington	West Virginia	Wisconsin	Wyoming
0	1	1	0	0	2	0	0	0	1	3	0	0	0	3	1	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	0	0	2	0	1	0	2	0	0	0	0	0	1	3	0	0	2	0	1	0	1
1	4	0	2	1	19	4	2	2	1	16	5	1	1	6	5	2	1	2	3	2	7	3	8	2	2	1	0	4	4	3	2	0	12	7	0	8	4	4	9	0	7									



Quantitative Plot:

The folowing data is related to our hypothesis testing question. Here, we draw an histogram between English-only speaking households and other non-English-speaking households. The distribution of their property value is shown, and the shape of both histograms are skewed to the right. Most of the data falls between 0 to 1,000,000 dollars.

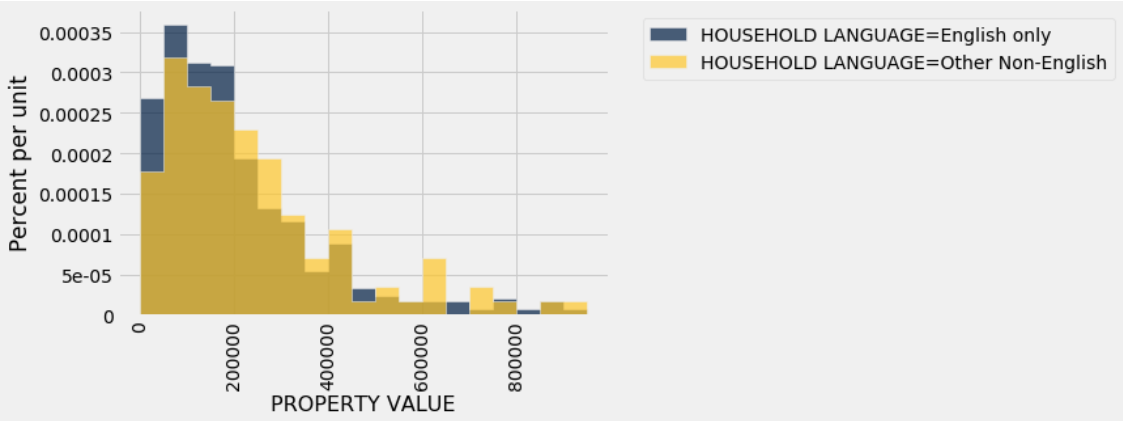
We can see that both histograms' distributions are actually very similar. So far, it appears as if property values don't differ all that much no matter what predominant language a household speaks. But we cannot conclude this just yet. Further analysis will be made in the *Hypothesis Testing* section.

We have decided to limit our histogram to these bins to get the best visualization. There are outlier households with greater property values than 1 million, but we have excluded them from

the histogram.

[7]:
```
eng_and_non_eng = hypothesis_data_filtered.group("HOUSEHOLD LANGUAGE")
hypothesis_data_filtered.hist("PROPERTY VALUE", group = "HOUSEHOLD LANGUAGE",␣
 ↪bins=np.arange(0,1000000, 50000))
```

```
/opt/conda/lib/python3.6/site-packages/numpy/core/_asarray.py:83:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
```



Qualitative Plot:

We want to see if the predominant languages spoken by households are accurately represented in our dataset. Because our dataset is a random sample of 1200 families in the United States, we should expect our calculated proportions to be accurate to the true proportion of spoken household languages in the United States taken by the United States census.

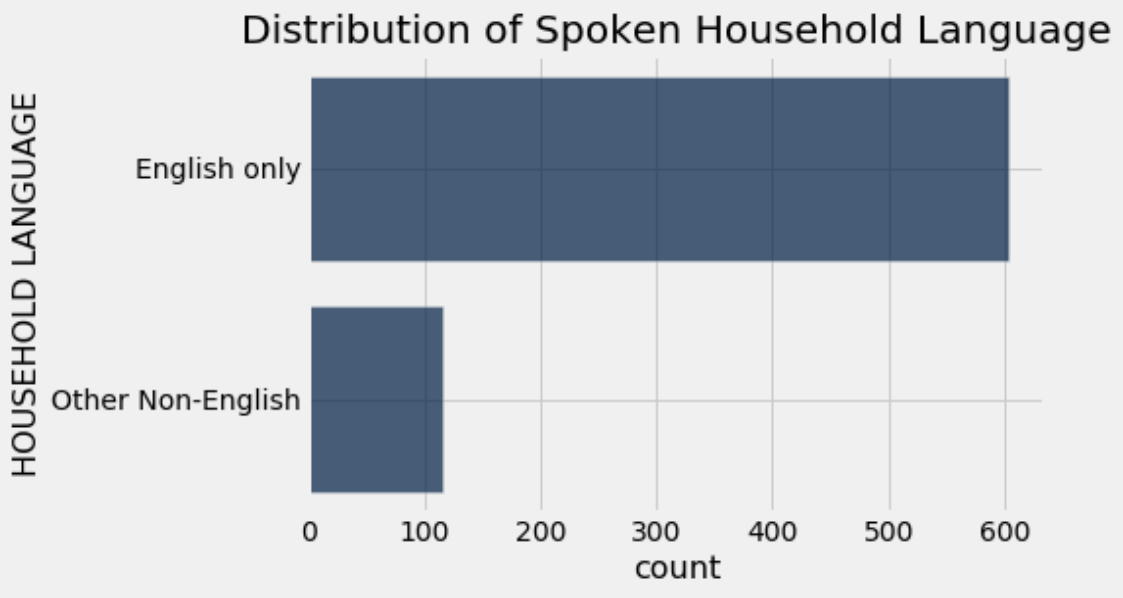
We will calculate the proportions by counting the number of both English-only speaking households and non-English-speaking households dividing each by the total number of houesholds. Then we will generate a bar chart to see the proportion of a certain household language.

We see that our calculations are fairly consistent with the actual data from the U.S. census (~80% English-speaking households).

[8]:
```
prop_eng_only = eng_and_non_eng.column(1).item(0) / sum(eng_and_non_eng.
 ↪column(1))
prop_non_eng = eng_and_non_eng.column(1).item(1) / sum(eng_and_non_eng.
 ↪column(1))
print("Proportion of English-only speaking households: ", prop_eng_only)
print("Proportion of Other Non-English speaking households: ", prop_non_eng)
eng_and_non_eng.barh("HOUSEHOLD LANGUAGE")
plots.title("Distribution of Spoken Household Language")
```

```
Proportion of English-only speaking households:  0.8388888888888889
Proportion of Other Non-English speaking households:  0.16111111111111112
```

[8]:
```
Text(0.5, 1.0, 'Distribution of Spoken Household Language')
```



2.4 Hypothesis Testing

Do not copy code from demo notebooks or homeworks! You may split portions of your code into distinct cells. Also, be sure to set a random seed so that your results are reproducible.

Now, we will actually perform our hypothesis testing procedure to determine if there really is a difference between the property value and which household language a household speaks. Our null hypothesis is the property value is the same for households that speak English only and households that speak a non-English language. Our alternative hypothesis is that households who speaks a language other than English will have a significantly different property value than households that speak English only.

We will be using A/B test to test our hypothesis. Our test statistic will be the difference in average property values of non-English speaking households and English-only speaking households. Extreme (small or big) values of the test statistic will favor the alternative hypothesis. We will be using a significance cutoff of 5%.

[9]:
```
# set the random seed so that results are reproducible
np.random.seed(420)

means_table = hyp_grouped
observed_difference = means_table.column(1).item(1) - means_table.column(1).
 ↪item(0)
```

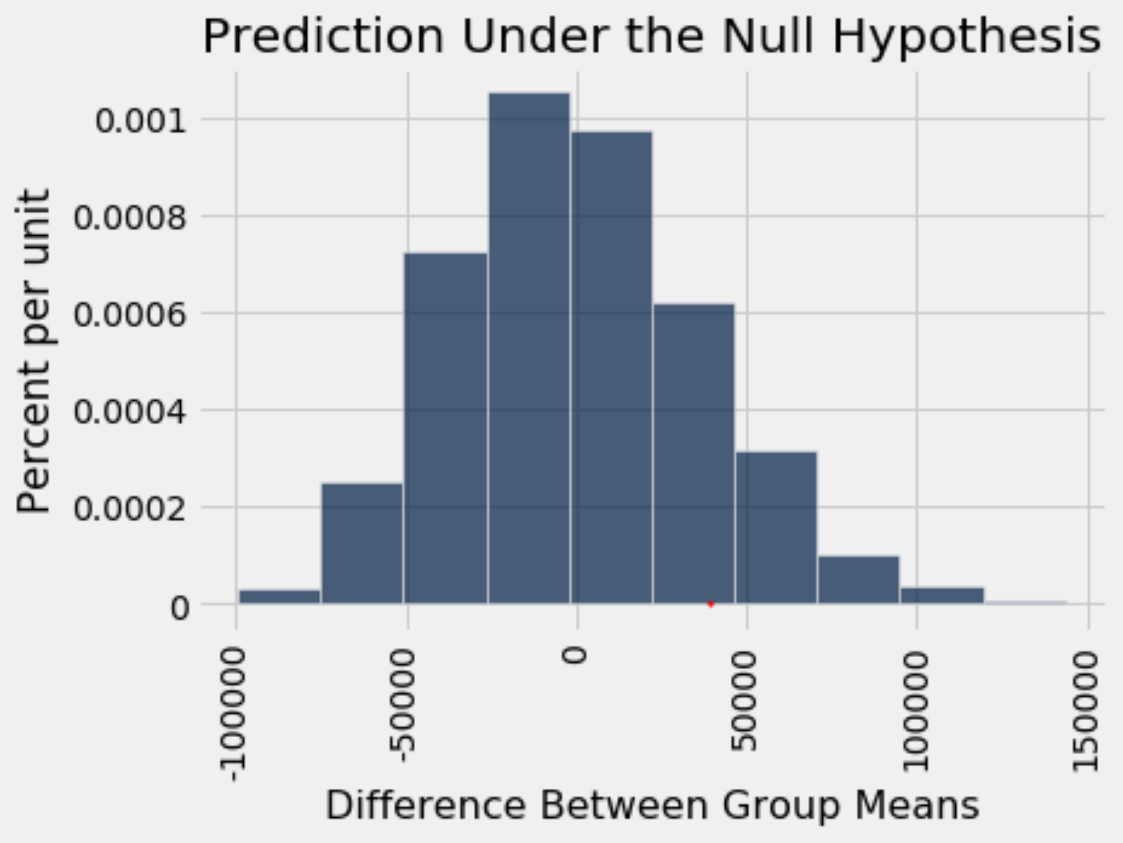
```
# mean different function
def mean_diff(table, label, group_label):
    means_table = table.select(label, group_label).group(group_label, np.
 ↪average)
    means = means_table.column(1)
    return means.item(1) - means.item(0)

# one simulated function
def one_simulate(table, label, group_label):
    shuf_labels = table.sample(with_replacement = False).column(group_label)
    tbl_labels = table.select(label).with_column('Shuffled Label', shuf_labels)
    return mean_diff(tbl_labels, label, 'Shuffled Label')

# repeat 10000 times
differences = make_array()
for i in np.arange(10000):
    one = one_simulate(hypothesis_data_filtered, "PROPERTY VALUE", "HOUSEHOLD␣
 ↪LANGUAGE")
    differences = np.append(differences, one)

# distribution under the null
Table().with_column('Difference Between Group Means', differences).hist()
print('Observed Difference:', observed_difference)
plots.title('Prediction Under the Null Hypothesis')
plots.scatter(observed_difference, 0, color = "red", s=3, zorder =2);
```

```
Observed Difference: 39131.31057319022
```



[10]:
```
# Calculate the p-value and percentiles
print(percentile(2.5, differences))
print(percentile(97.5, differences))
p_value = np.count_nonzero(differences <= observed_difference)/10000
p_value
```

```
-63634.11760675954
78754.45284311485
```

[10]: 0.8526

We see that our observed difference value, 39131, is within the middle 95% confidence interval. We also see that the p_value is well above 0.05. Therefore, we should accept the null hypothesis.

[11]:
```
# on 5% significant level
p_value > 0.05
```

[11]: True

Our observed test statistic is equal to about 40000. Under the null assumption, we have evidence that supports that property value is the same for all households, no matter what language a household may speak, English or not. Our data is consistent with the null hypothesis. So, we

accept the null hypothesis and assume that there is no relationship between property value and English-speaking households.

2.5 Prediction

Be sure to set a random seed so that your results are reproducible.

2.6 K-Nearest Neighbors

We will implement a k-nearest neighbors classifier that will determine what states and regions have higher or lower vehicle ownership. First, we will divide up our dataset into a training and test set.

As of right now, we expect households located in states and regions that are more populated to have more vehicles (i.e. California, Texas, etc.) based on the aggregated data table from the *Exploratory Data Analysis* section.

We will change the data in the "Number of Vehicles" column to 0-2 (for low number of vehicles) and 3-5 (for high number of vehicles).

[12]:
```
# Helper function for converting the "Number of Vehicles" column
def turn_to_string(number):
    b = ""
    if number >= 0 and number <= 2:
        b = "0-2"
    else:
        b = "3-5"
    return b
```

[13]:
```
np.random.seed(420)

new_prediction_data = prediction_data_filtered.with_columns(
    "NUMBER OF VEHICLES", prediction_data_filtered.apply(turn_to_string,␣
 ↪"NUMBER OF VEHICLES")
)
shuffle_data = new_prediction_data.sample(with_replacement=False)
training_set = shuffle_data.take(np.arange(342))
test_set = shuffle_data.take(np.arange(342, 455))

print("Training set:\t", training_set.num_rows, "examples")
print("Test set:\t", test_set.num_rows, "examples")
training_set.show(5),  test_set.show(5);
```

```
Training set:    342 examples
Test set:        113 examples
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

Here, we will define some helper functions to use for our classifier.

[14]:
```
# Helper functions for our classifier
def distance(arg1, arg2):
    return np.sqrt(sum((arg1 - arg2) ** 2))

def row_to_array(row, variables):
    arr = make_array()
    for var in variables:
        arr = np.append(arr, row.item(var))
    return arr
```

This classifier chooses an arbitrary k (we have chosen k=20) and uses the properties of the `training_set` and `test_set` in order to make a prediction about how many vehicles belong to a household.

[15]:
```
# Classifier
properties = make_array("STATE", "PROPERTY VALUE", "ID")
def classify(row, k, train):
    test_row_features_array = row_to_array(row, properties)
    distances = make_array()
    for train_row in train.rows:
        train_row_features_array = row_to_array(train_row, properties)
        row_distance = distance(test_row_features_array,␣
 ↪train_row_features_array)
        distances = np.append(distances, row_distance)
    train_with_distances = train.with_columns('Distance', distances)
    nearest_neighbors = train_with_distances.sort('Distance').take(np.arange(k))
    most_common_label = nearest_neighbors.group('NUMBER OF VEHICLES').
 ↪sort('count', descending=True).column('NUMBER OF VEHICLES').item(0)
    return most_common_label
```

The code block below runs multiple classifications and we check for its accuracy.

[16]:
```
def classify_tests(table, pick_k):
    arr = make_array()
    for i in np.arange(test_set.num_rows):
        guess = classify(table.row(i), pick_k, training_set)
        arr = np.append(arr, guess)
    return arr

classified_tests = classify_tests(test_set, 20)
percent_correct = np.count_nonzero(classified_tests == test_set.column("NUMBER␣
 ↪OF VEHICLES")) / test_set.num_rows * 100
percent_correct
```

[16]:
```
76.10619469026548
```

We see that the percentage correct of our estimator is around 76%, which is decent for a classifier. In other words, we can guess correctly with our classifier whether a household has a high or low amount of vehicles 76% of the time.

The following classifier returns an array of states that are in alphabetical order and with no duplicates. This shows a list of states that the classifier thinks has households that have a high amount of vehicle ownership.

[17]:
```
def state_low_vehicle_classifier(table):
    states_low_vehicles = make_array()
    for i in np.arange(test_set.num_rows):
        if classify(table.row(i), 20, training_set) == '0-2':
            states_low_vehicles = np.append(states_low_vehicles, table.row(i).
 ↪item(1))
    return states_low_vehicles

lower = state_low_vehicle_classifier(test_set)
```

[18]:
```
lower = np.array(sorted(list(dict.fromkeys(lower))))
lower
```

[18]:
```
array(['Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut',
       'Florida', 'Georgia', 'Illinois', 'Indiana', 'Kansas', 'Louisiana',
       'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
       'Missouri', 'Nebraska', 'Nevada', 'New Jersey', 'New York',
       'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
       'Pennsylvania', 'South Carolina', 'Tennessee', 'Texas', 'Utah',
       'Virginia', 'Washington', 'West Virginia', 'Wisconsin'],
      dtype='<U14')
```

If we take the average of number of vehicles that each state has from the actual dataset and consider an average of 2- to be "lower", then we can compare the actual list of states with the shuffled dataset that our classifier returned, which will return our true accuracy of the classifier.

[19]:
```
average_num_vehicles_per_state = prediction_data_filtered.group("FULL NAME", np.
 ↪mean).select("FULL NAME", "NUMBER OF VEHICLES mean")
actual_states_low_vehicles = average_num_vehicles_per_state.where("NUMBER OF␣
 ↪VEHICLES mean", are.below_or_equal_to(2))
actual_lower = actual_states_low_vehicles.column(0)
```

[20]:
```
def actual_accuracy(expected, actual):
    # Calculate expected
    states_correct = 0
    for i in expected:
        for j in actual:
            if i == j:
                states_correct = states_correct + 1
    # Percent Error formula
```

```
    return 100 - (abs(len(actual) - states_correct) / len(actual)) * 100

actual_accuracy(lower, actual_lower)
```

[20]: 64.51612903225806

Our classifier can predict, with 64% accuracy, that the states (and their corresponding regions, which we have left out, but can easily figure out by looking at `full_data`) that have lower vehicle ownership are listed in the list "lower". It's not the best classifier, but we can make a reasonable prediction on which states and regions have high vehicle ownership.

In order to find out which states that have a high vehicle ownership (3-5 cars per household), we just have to see the states that are not on the "lower" list.

2.7 Conclusion

In this research report, we tried to see if a relationship exists between property value and a household's spoken language as well as try to predict which states and regions have higher or lower vehicle ownership. Through A/B testing, we examined property value vs. English-only speaking households as well as property value vs. non-English-speaking households, and both of these distributions have the same property value behavior no matter what predominant language a household speaks. Therefore, we concluded that there is no relationship between property value and English-only speaking households. We also built a classifier that comes up with a list of states which are considered to have low vehicle ownership with decent accuracy.

Unfortunately, we did have to filter out a lot of characteristics out of our dataset. We lost information about renters and people who chose not to report (indicated as "Non-Family"), which are important to understand if we wanted a more accurate answer to these inquiries in real life.

2.8 Presentation

In this section, you'll need to provide a link to your video presentation. If you've uploaded your presentation to YouTube, you can include the URL in the code below. We've provided an example to show you how to do this. Otherwise, provide the link in a markdown cell.

Link: *Replace this text with a link to your video presentation*

[21]:
```
# Full Link: https://www.youtube.com/watch?v=BKgdDLrSC5s&feature=emb_logo
# Plug in string between "v=" and ""&feature":
YouTubeVideo('AMjbvxRrHJo')
```

[21]:

3 Submission

Just as with the other assignments in this course, please submit your research notebook to Okpy. We suggest that you submit often so that your progress is saved.

[ ]:
```
# Run this line to submit your work
_ = ok.submit()
```

```
<IPython.core.display.Javascript object>
```

[ ]:
```

```