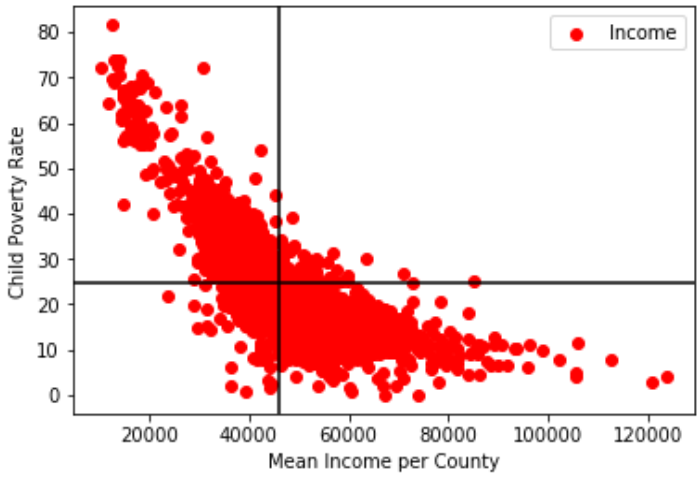
Taylor Choe
Radhika Dhomse
Shubham Gupta
Dhanush Patel

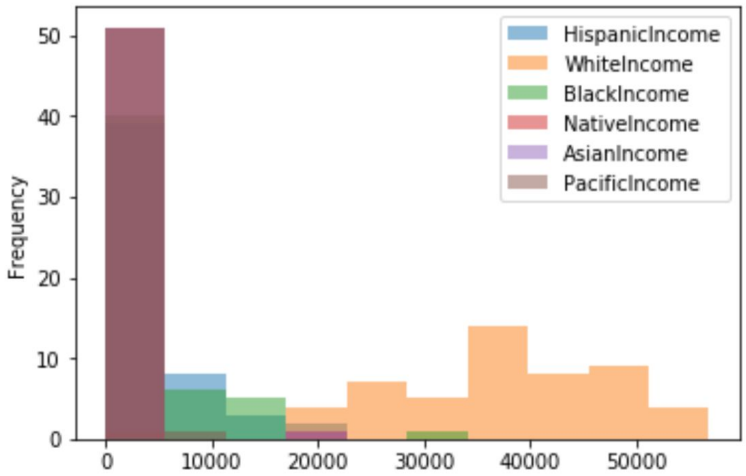
When we started the project, we faced some difficulty choosing financial indicators. We each came in with certain conceptions of how pairs of indicators would correlate and found that we were incorrect regarding some of them. We ended up trying different representations for a lot of different pairs of indicators and chose four sets that told us the most about the data. The four we ended with were race vs income, child poverty vs income, professional vs income, modes of transportation vs income.

Our first graph presents the percentage of individuals employed in management, business, science, and arts. We used a scatter plot for this as we only had two indicators that we wanted to compare directly. We found a fairly strong positively correlated relationship between the percentage of individuals involved in professional work and their income as the income increases with the percentage. We had no preconceptions regarding this relationship, but we chose these indicators as they showed the strongest correlation.

Our second idea was to try Race versus Child Poverty in which we made scatter plots — one that averaged over county and one that averaged over state. For both of those plots, we didn't observe any significant trends, so we decided to instead try Income versus Child Poverty, while taking taking the average of the censuses from each county. We were able to observe a negative correlation between Income and Child Poverty. We graphed both the average child poverty rate and the average of the mean income per county and observe that the average of the mean income per county is 46188.44 and the average child poverty rate is about 24.76%. The trend of the graph reflects what we expected about the relationship between income and child poverty rate.

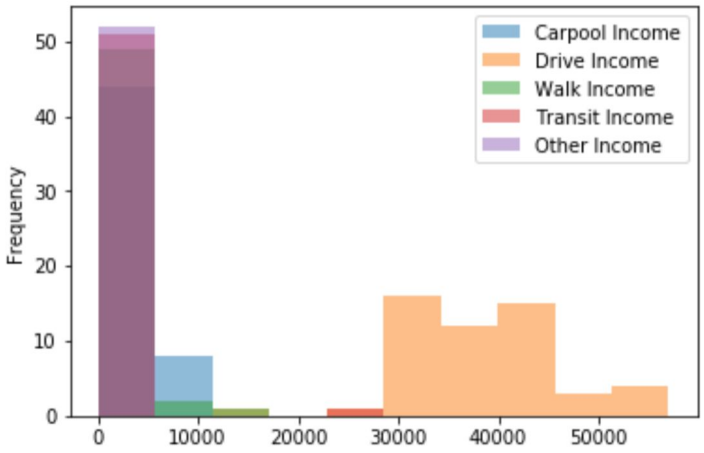


Our third graph presents race vs income. We decided to use a representation that consists of multiple overlaid histograms. This only confirmed our preconceptions that the range of incomes for white individuals falls much higher than those of other races. The minorities were much, much lower just as we expected with Black, Native, and Pacific race individuals having ranges on the far left of the graph with very low incomes and a few outliers. The average Asian range of incomes was somewhere in the middle which was also very consistent with what we had believed prior to analyzing the data.

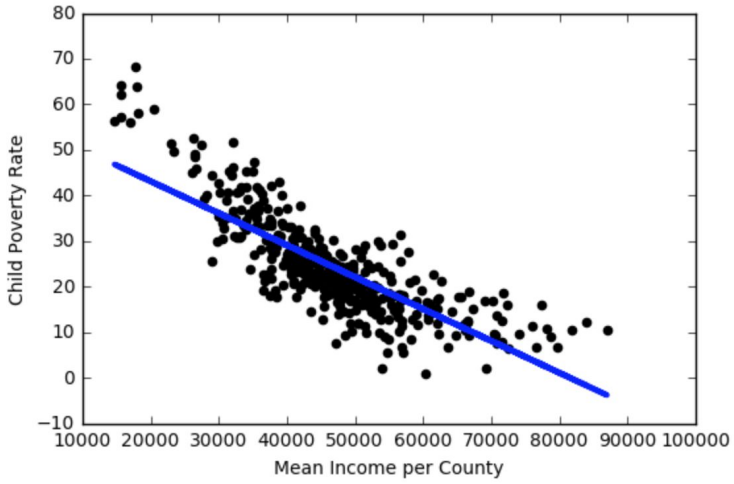


For our fourth graph on modes of transportation vs income, we initially had tried a scatter plot with different colors per mode of transportation, but this representation did not clearly reveal anything about the data so we decided to transition to a graphical representation that overlays multiple histograms for income based on each mode of transportation. We found that those who drive to work generally have a much range of

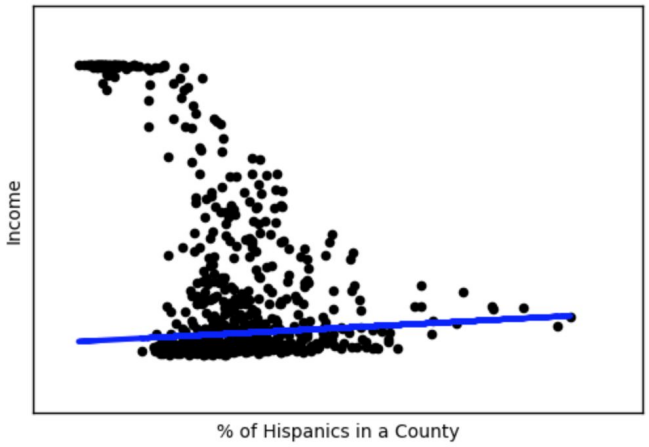
incomes than those who use other modes of transportation. This was not a huge surprise to us; however, we were surprised to find that those who use transit have a higher range of incomes than those who carpool. The outcome for walking and other forms of income were not too surprising.



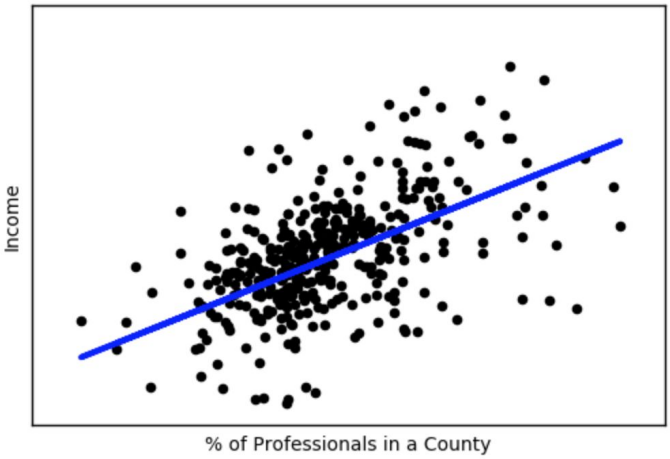
Using the linear relationship we saw between Child Poverty and County Income, we decided to use linear regression to be able to predict child poverty rate from county income. Using this technique, on our training set we were able to get a mean squared error of 42.10 and a variance score of 0.67. A visualization of the prediction for the test set is shown below:



We similarly used linear regression technique to predict income based upon the percentage of hispanics in the county. We got a mean squared error of 1505.57 and a variance score of 0.67.



Using this technique again to predict income based upon the percentage of professionals in the county, we got a mean squared error of 107947377.02 and a variance score of 0.26.



Using what we learned in our exploratory data analysis, we were able to determine which features formed a strong linear relationship to help us use Linear Regression to aid our predictive model.

Project 1: Data Cleaning, Visualization, and Mining

In [1]:
```
!pip install plotly --upgrade
!pip install sklearn --upgrade
# Add all of your import statements here
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import plotly
import plotly.plotly as py
from plotly.graph_objs import *
plotly.tools.set_credentials_file(username='dhanush123', api_key='H6asXrlTQsML
a7WQJzbg')
plotly.offline.init_notebook_mode(connected=True)

from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

import warnings
warnings.filterwarnings(action='once')
```

```
Collecting plotly
Collecting decorator>=4.0.6 (from plotly)
  Using cached decorator-4.2.1-py2.py3-none-any.whl
Requirement already up-to-date: six in /srv/app/venv/lib/python3.6/site-pack
ages (from plotly)
Collecting requests (from plotly)
  Using cached requests-2.18.4-py2.py3-none-any.whl
Requirement already up-to-date: nbformat>=4.2 in /srv/app/venv/lib/python3.
6/site-packages (from plotly)
Requirement already up-to-date: pytz in /srv/app/venv/lib/python3.6/site-pac
kages (from plotly)
Collecting certifi>=2017.4.17 (from requests->plotly)
  Using cached certifi-2018.1.18-py2.py3-none-any.whl
Requirement already up-to-date: urllib3<1.23,>=1.21.1 in /srv/app/venv/lib/p
ython3.6/site-packages (from requests->plotly)
Requirement already up-to-date: chardet<3.1.0,>=3.0.2 in /srv/app/venv/lib/p
ython3.6/site-packages (from requests->plotly)
Requirement already up-to-date: idna<2.7,>=2.5 in /srv/app/venv/lib/python3.
6/site-packages (from requests->plotly)
Requirement already up-to-date: traitlets>=4.1 in /srv/app/venv/lib/python3.
6/site-packages (from nbformat>=4.2->plotly)
Requirement already up-to-date: jsonschema!=2.5.0,>=2.4 in /srv/app/venv/li
b/python3.6/site-packages (from nbformat>=4.2->plotly)
Requirement already up-to-date: jupyter-core in /srv/app/venv/lib/python3.6/
site-packages (from nbformat>=4.2->plotly)
Requirement already up-to-date: ipython-genutils in /srv/app/venv/lib/python
3.6/site-packages (from nbformat>=4.2->plotly)
Installing collected packages: decorator, certifi, requests, plotly
  Found existing installation: decorator 4.1.2
    Uninstalling decorator-4.1.2:
      Successfully uninstalled decorator-4.1.2
  Found existing installation: certifi 2017.11.5
    Uninstalling certifi-2017.11.5:
      Successfully uninstalled certifi-2017.11.5
  Found existing installation: requests 2.12.4
    Uninstalling requests-2.12.4:
      Successfully uninstalled requests-2.12.4
Successfully installed certifi-2018.1.18 decorator-4.2.1 plotly-2.4.1 reques
ts-2.18.4
Collecting sklearn
Collecting scikit-learn (from sklearn)
  Using cached scikit_learn-0.19.1-cp36-cp36m-manylinux1_x86_64.whl
Installing collected packages: scikit-learn, sklearn
  Found existing installation: scikit-learn 0.19.0
    Uninstalling scikit-learn-0.19.0:
      Successfully uninstalled scikit-learn-0.19.0
Successfully installed scikit-learn-0.19.1 sklearn-0.0
```

Setup, Cleaning, Organizing, and Exploring the Data

Let's now try to read in the data, clean up the data by getting rid of NAs, and explore the data. Feel free to use any commands to do these parts. As always, report your steps in the writeup.

In [2]:
```
df = pd.read_csv('acs2015_county_data.csv')
df.head()
```

Out[2]:

	CensusId	State	County	TotalPop	Men	Women	Hispanic	White	Black	Native
0	1001	Alabama	Autauga	55221	26745	28476	2.6	75.8	18.5	0.4
1	1003	Alabama	Baldwin	195121	95314	99807	4.5	83.1	9.5	0.6
2	1005	Alabama	Barbour	26932	14497	12435	4.6	46.2	46.7	0.2
3	1007	Alabama	Bibb	22604	12073	10531	2.2	74.5	21.4	0.4
4	1009	Alabama	Blount	57710	28512	29198	8.6	87.9	1.5	0.3

5 rows × 37 columns

Visualizing the Data

This is the most fun part of the project! Now you can go ahead and create your graphs (appropriately labeled of course), models, and other visualizations. Make sure you demonstrate your ability to create rich graphs by creating various different types of graphs, both for numerical as well as categorical data.

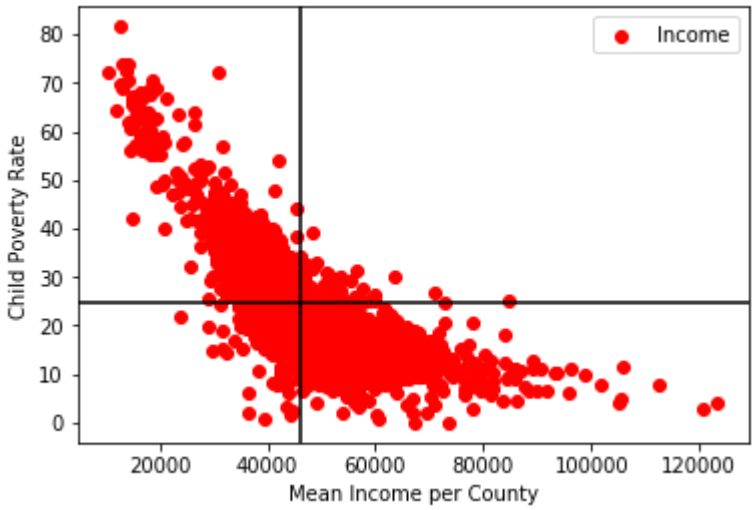
Here are some types of graphs that you will want to create for this project. Remember to browse the online documentation for many Python graphing and visualization packages such as plotly, seaborn, and matplotlib if you need to borrow any code or get some help!

1. Creating multiple graphs that plot important financial indicators (y-axis) like the median annual income, per capita income, poverty rate, childhood poverty rate, and/or unemployment rate present in a particular county AGAINST several features in the dataset (x-axis) that might affect these factors, such as the total population of the county; the racial demographics of the county; or the occupations of different workers (professionals, service workers, construction, manufacturing).
2. Using plotly as a graphing library to visualize these financial indicators in the 2015 American Community Survey data on a geographical map of the United States on a state-wide or county-based level (similar to what we did in class when we visualized global average temperatures for different countries on a global map).

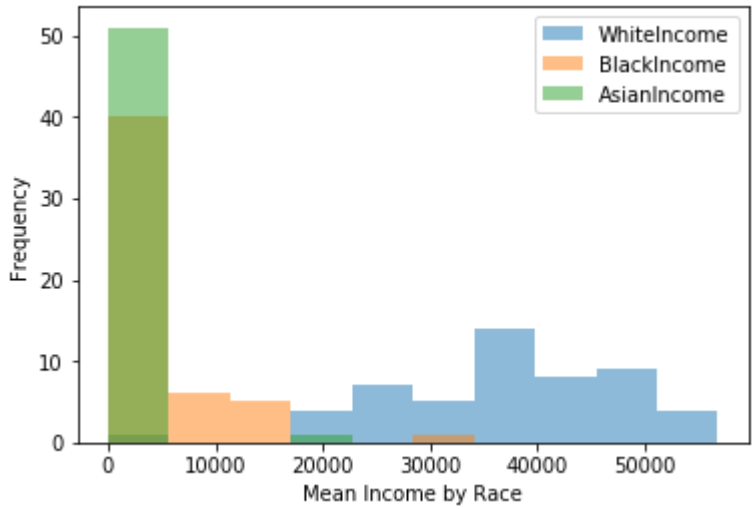
In [3]:
```
df = df.dropna(axis=0, how='any')
################################################################################
##########
childPovIncome = df[["County", "Income", "ChildPoverty"]]
childPovIncome = childPovIncome.groupby("County").mean()
meanIncome = childPovIncome["Income"].mean()
meanChildPov = childPovIncome['ChildPoverty'].mean()
plt.scatter(childPovIncome['Income'], childPovIncome['ChildPoverty'], c='r')
plt.legend(["Income", "ChildPoverty"])
```

```
plt.xlabel("Mean Income per County")
plt.ylabel("Child Poverty Rate")
plt.axvline(x=meanIncome, color='k')
plt.axhline(y=meanChildPov, color='k')
plt.show()
##########################################################################
##########
graph1DF = df[["Income","TotalPop", "Hispanic", "White", "Black", "Native",
"Asian", "Pacific"]]

hispanicPop = np.array(graph1DF["Hispanic"]) * np.array(graph1DF["Income"])
/ 100
whitePop = np.array(graph1DF["White"]) * np.array(graph1DF["Income"]) / 100
blackPop = np.array(graph1DF["Black"]) * np.array(graph1DF["Income"]) / 100
nativePop = np.array(graph1DF["Native"]) * np.array(graph1DF["Income"]) / 10
0
asianPop = np.array(graph1DF["Asian"]) * np.array(graph1DF["Income"])  / 100
pacificPop = np.array(graph1DF["Pacific"]) *  np.array(graph1DF["Income"]) /
 100

d ={"State":df["State"], "HispanicIncome": hispanicPop, "WhiteIncome": white
Pop, "BlackIncome": blackPop, "NativeIncome": nativePop, "AsianIncome":asian
Pop, "PacificIncome":pacificPop}
newDF = pd.DataFrame(data = d)
newDF = newDF.dropna(axis=0, how='any')
newDF = newDF.groupby("State").mean()

plt.figure();
newDF[["WhiteIncome", "BlackIncome", "AsianIncome"]].plot.hist(alpha=0.5)

plt.xlabel("Mean Income by Race")
plt.show()
##########################################################################
###########
incomeTransportation = df[["County", "Income", "Carpool", "Drive", "Walk",
"Transit", "OtherTransp"]]

carpoolPop = np.array(incomeTransportation["Carpool"]) * np.array(incomeTran
sportation["Income"]) / 100
drivePop = np.array(incomeTransportation["Drive"]) * np.array(incomeTranspor
tation["Income"]) / 100
walkPop = np.array(incomeTransportation["Walk"]) * np.array(incomeTransporta
tion["Income"]) / 100
transitPop = np.array(incomeTransportation["Transit"]) * np.array(incomeTran
sportation["Income"]) / 100
otherPop = np.array(incomeTransportation["OtherTransp"]) * np.array(incomeTr
ansportation["Income"])  / 100

d ={"State":df["State"], "Carpool Income": carpoolPop, "Drive Income": drive
Pop, "Walk Income": walkPop, "Transit Income": transitPop, "Other Income":ot
herPop}
newDF = pd.DataFrame(data = d)
newDF = newDF.dropna(axis=0, how='any')
newDF = newDF.groupby("State").mean()

plt.figure()
newDF[["Carpool Income", "Drive Income", "Walk Income", "Transit Income", "O
```

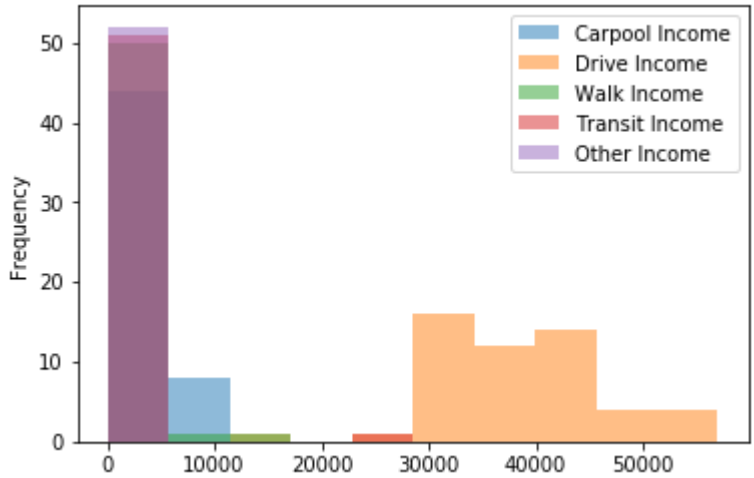
```
ther Income"]].plot.hist(alpha = 0.5)
plt.show()
##########################################################################
###########
```




```
<matplotlib.figure.Figure at 0x7ff70fcdaeb8>
```

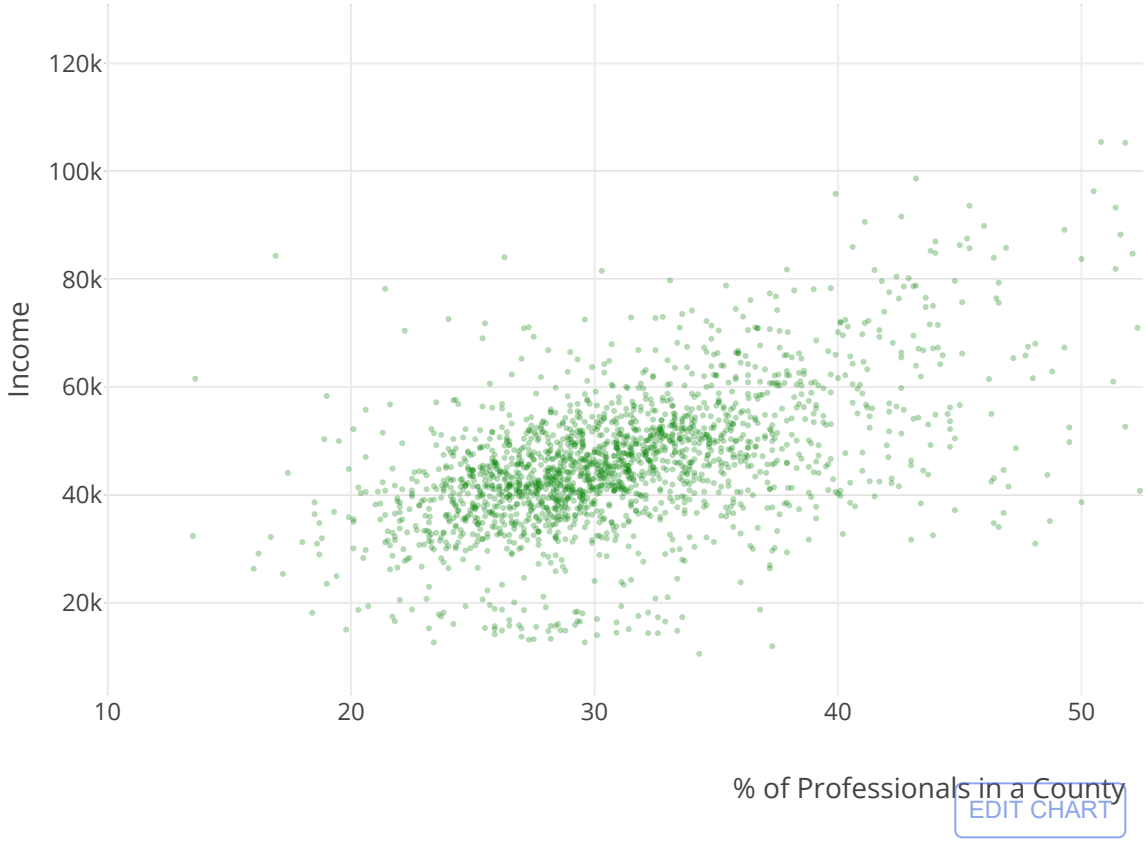



```
<matplotlib.figure.Figure at 0x7ff70fb8fb38>
```



In [4]:
```
dataIP = df.groupby("County")[["Income","Professional"]].mean().reset_index
()
graphX = []
graphY = []
for row in dataIP.itertuples():
    graphX.append(row.Professional)
    graphY.append(row.Income)

layoutz = Layout(xaxis=dict(
            title='% of Professionals in a County',
            titlefont=dict(
                size=18,
                color='#ad1616'
                )
            ),
          yaxis=dict(
                title='Income',
                titlefont=dict(
                    size=18,
                    color='#ad1616'
                )
            ))
py.iplot({"data":[Scatter(x=graphX,
                  y=graphY,
                  mode='markers',
                  marker=Marker(color='green', size=3, opacity=0.3),
                 )],
          "layout":{
                 "xaxis":{"title":"% of Professionals in a County"},
                 "yaxis":{"title": "Income"}
          }})
```

Out[4]:



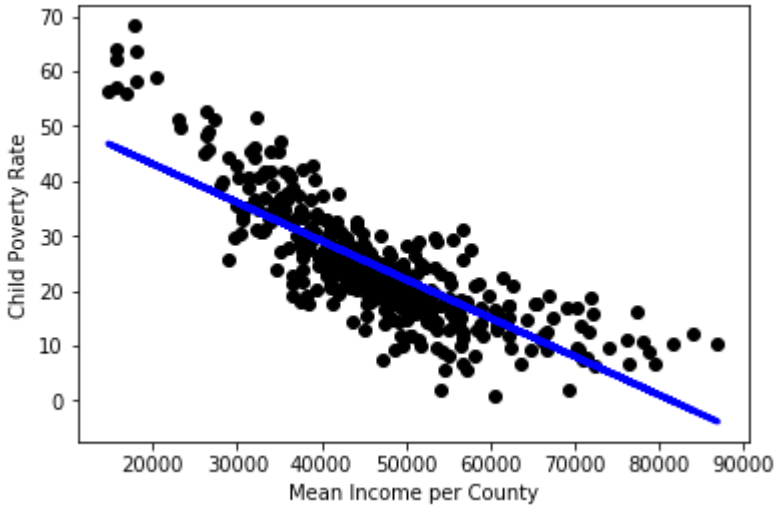
Linear Regression

Make sure you also demonstrate your ability to create regressions on your data. In this part of the project, you will pick 3 financial indicators (like the median annual income, per capita income, poverty rate, childhood poverty rate, and/or unemployment rate) in the dataset. You will then train 3 linear regression models in scikit-learn that attempt to predict these 3 financial indicators of a county from the rest of the data collected in the dataset. You should analyze the accuracy of your linear regression model and report important predictors or features in your dataset that had the highest effect size on these 3 financial indicators (as discovered by your regression model).

Optional: You can try adding regularization to see if you can get better results.

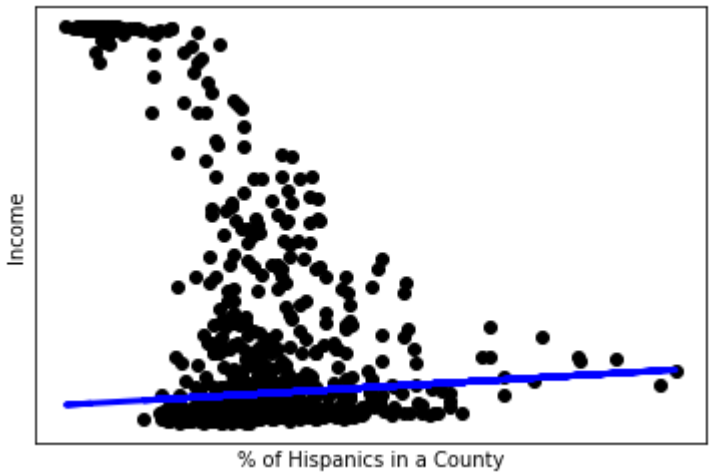
In [5]:
```
PI_Xtrain = childPovIncome["Income"][:int(len(childPovIncome)*0.8)]
PI_Xtest = childPovIncome["Income"][int(len(childPovIncome)*0.8):]
PI_Ytrain = childPovIncome["ChildPoverty"][:int(len(childPovIncome)*0.8)]
PI_Ytest = childPovIncome["ChildPoverty"][int(len(childPovIncome)*0.8):]

lr = linear_model.LinearRegression()
lr.fit(PI_Xtrain.values.reshape(-1, 1), PI_Ytrain.values.reshape(-1, 1))
PI_Ypred = lr.predict(PI_Xtest.values.reshape(-1, 1))

# The coefficients
print('Coefficients: \n', lr.coef_)
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(PI_Ytest, PI_Ypred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(PI_Ytest, PI_Ypred))

# Plot outputs
plt.scatter(PI_Xtest, PI_Ytest,  color='black');
plt.plot(PI_Xtest, PI_Ypred, color='blue', linewidth=3);

plt.xticks()
plt.yticks()
plt.xlabel("Mean Income per County")
plt.ylabel("Child Poverty Rate")

plt.show()
```

```
Coefficients:
 [[-0.00070052]]
Mean squared error: 42.10
Variance score: 0.67
```



In [6]:
```
IR_Xtrain = graph1DF["Income"][:int(len(graph1DF)*0.8)]
IR_Xtest = graph1DF["Income"][int(len(graph1DF)*0.8):]
IR_Ytrain = graph1DF["Hispanic"][:int(len(graph1DF)*0.8)]
IR_Ytest = graph1DF["Hispanic"][int(len(graph1DF)*0.8):]

lr = linear_model.LinearRegression()
lr.fit(IR_Xtrain.values.reshape(-1, 1), IR_Ytrain.values.reshape(-1, 1))
IR_Ypred = lr.predict(IR_Xtest.values.reshape(-1, 1))

# The coefficients
print('Coefficients: \n', lr.coef_)
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(IR_Ytest, IR_Ypred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(PI_Ytest, PI_Ypred))

# Plot outputs
plt.scatter(IR_Xtest, IR_Ytest,  color='black');
plt.plot(IR_Xtest, IR_Ypred, color='blue', linewidth=3);

plt.xticks(());
plt.yticks(());
plt.xlabel("% of Hispanics in a County");
plt.ylabel("Income");

plt.show();
```

```
Coefficients:
 [[  7.81782946e-05]]
Mean squared error: 1505.57
Variance score: 0.67
```



In [7]:
```
dataIP_Xtrain = dataIP["Professional"][:int(len(dataIP)*0.8)]
dataIP_Xtest = dataIP["Professional"][int(len(dataIP)*0.8):]
dataIP_Ytrain = dataIP["Income"][:int(len(dataIP)*0.8)]
dataIP_Ytest = dataIP["Income"][int(len(dataIP)*0.8):]

lr = linear_model.LinearRegression()
lr.fit(dataIP_Xtrain.reshape(-1, 1),dataIP_Ytrain.reshape(-1, 1))
dataIP_Ypred = lr.predict(dataIP_Xtest.reshape(-1, 1))

# The coefficients
print('Coefficients:', lr.coef_)
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(dataIP_Ytest, dataIP_Ypred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(dataIP_Ytest, dataIP_Ypred))

# Plot outputs
plt.scatter(dataIP_Xtest, dataIP_Ytest,  color='black');
plt.plot(dataIP_Xtest, dataIP_Ypred, color='blue', linewidth=3);

plt.xticks(());
plt.yticks(());
plt.xlabel("% of Professionals in a County");
plt.ylabel("Income");

plt.show();
```

```
Coefficients: [[ 1209.02927755]]
Mean squared error: 107947377.02
Variance score: 0.26
```

```
/srv/app/venv/lib/python3.6/site-packages/ipykernel_launcher.py:7: FutureWarn
ing:

reshape is deprecated and will raise in a subsequent release. Please use .val
ues.reshape(...) instead
```