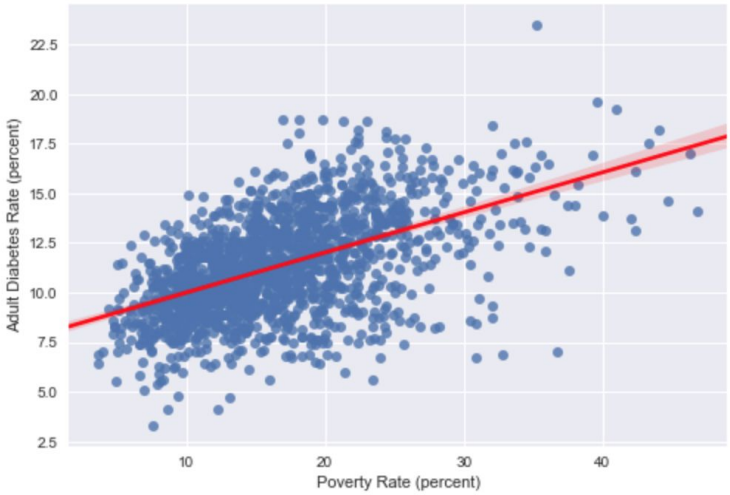
DSD Project 1 - Food by Counties Dataset

Team Members: Neelesh Dodda, Yongqi Gan, Pransu Dash

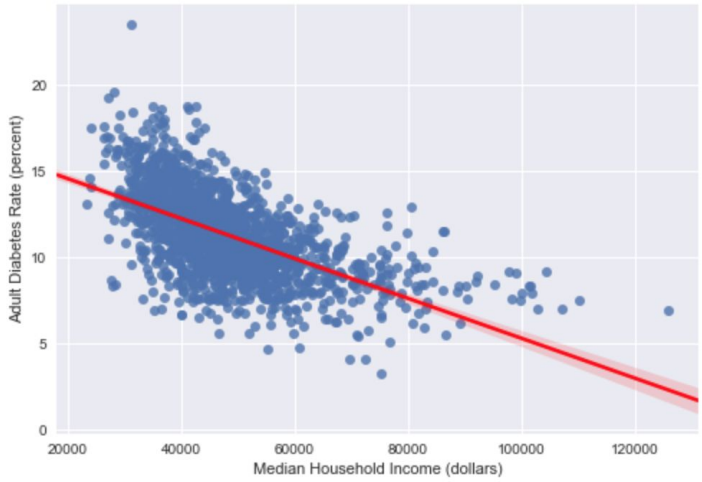
Our project involved the vast and very segmented Food Environment Atlas. The dataset provides many figures on agriculture, food options, health levels, and other socioeconomic factors within each county in the United States. The first step was to clean and merge the data. Since the data was presented in many different csv sheets, we had to merge the columns of all the tables together by county and FIPS. This led us to recognize that are many pitfalls when working with such huge datasets since many entries would have extra spaces which made comparing values nearly impossible until we recognized the errors in the data. Cleaning data was by far the most important part of this project as it allowed us to perform fast computations on the data without having to worry about losing integrity of data. We chose to analyze three possible relations within the data.

The first concerns the poverty rates in the counties compared to their adult diabetes rates. The original goal was to fit a linear model to some measure of socioeconomic status as it relates to a measure of health. Looking at the measured socioeconomic factors, the 2 that make the most sense to analyze are “Median Household Income” and “Poverty Rate” and their relation to either the “Adult Obesity Rate” or “Adult Diabetes Rate” by county. Finding a set of variables that showed a possible linear relation was also a big challenge due to the vast number of possible combinations of variables in the dataset. After some educated trial and error, we settled on selecting a better predictor of Adult Diabetes Rate in a county: Median Income or Poverty Rate. After performing a linear regression on both combinations, we got lower R-values than we hoped for but nonetheless got a relatively low mean squared error (4.2 and 4.4 respectively). The correlations are shown in the graphs below:

```
r-value:  0.528108380817
```




```
r-value:  0.574991248313
```



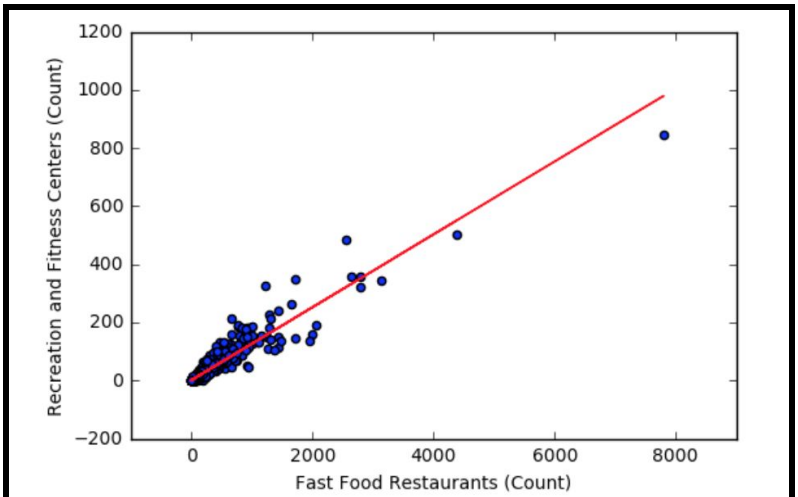
Overall, there appears to be some correlation between the median income and poverty rate with the adult diabetes rate in each county. However, there is definitely room for improvement and one of these improvements could be, if time permitted, a multivariate regression with income and poverty rate as explanatory variables for adult diabetes rate. We could also incorporate other explanatory variables such as grocery store availability versus convenience store availability. This shows how linearity restricts regression by looking for a strictly decreasing or increasing relation which limits what sorts of data can be subjected to this kind of analysis. Nonetheless it is still powerful for showing some types of relations such as this

one where we can see that some measures of low health can be related to socioeconomic factors. Using methods like this, we can get a better understanding of how to recognize health disorders like diabetes by gauging a person's financial status.

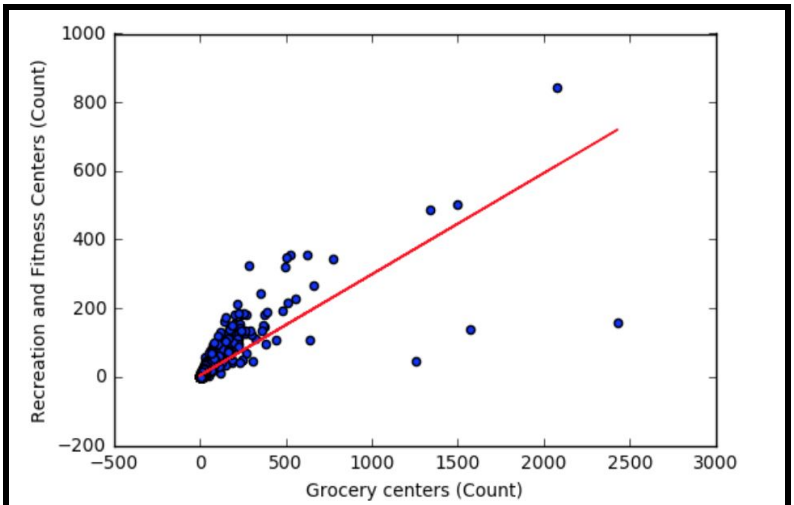
Our attempts at logistic regression were somewhat limited by the nature of the data. There were only a few columns of data that were binary response variables. Initially we considered forcing some data into a binary form, by imposing some arbitrary cutoff. However, we decided against this course of action since it would bring up credibility and validity issues with regards to where we decided to put our cutoff.

We decided to study the difference between urban and rural counties. Specifically, we wanted to see if we could use the density of stores and eating establishments (number per 1000 residents) in order to predict whether a given county was urban or rural. We were initially not expecting the predictor to predict very well, because the density is relative (per 1000), and one would expect that stores and restaurants would want to open in places with low density to capture the market. We were therefore surprised when the regression model was somewhat effective at predicting whether a county was urban or rural. Our overall prediction accuracy was 75%.

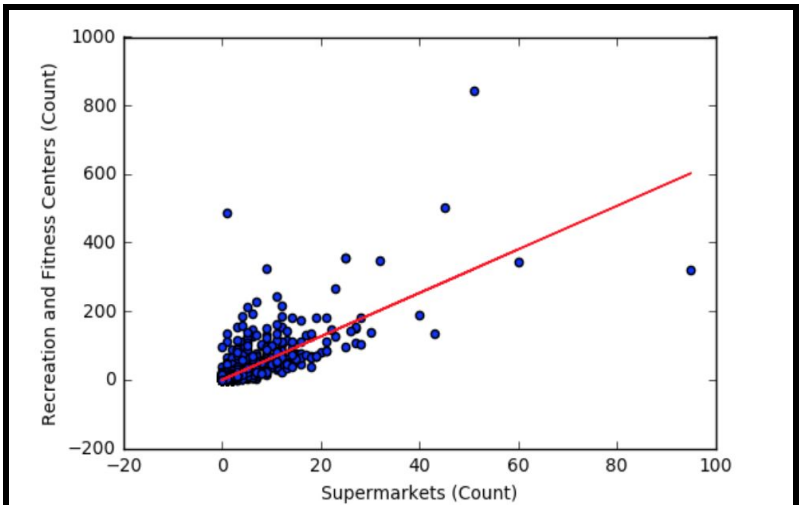
Lastly, we looked at the correlation between different three types of dining establishments and recreation/fitness centers. The idea was to see whether particular food places had greater association with rec centers than others. There were many options for food places, but we ended up selecting fast food restaurants, grocery stores, and supermarkets for the most variety. Each variable selected corresponds to a count of the particular place by each county in the year of 2014 (FFR14, GROC14, SUPERC14, RECFAC14). We performed a linear regression on each food place count with recreation/fitness center count. The graphs are shown below:



R-value: 0.951256158719



R-value: 0.793643614909



R-value: 0.755163330649

Overall, we see strong correlations between grocery centers/supermarkets and rec centers. However, there is an almost linear relationship between fast food restaurants and rec centers. This shows (much more definitively than the other dining establishments) that we can predict more recreation/fitness centers with more fast food restaurants. If counties can't stop the growth of the fast food industry, then the only solution is recreation and fitness centers!

DSD Project 1

Pransu Dash, Neelesh Dodda, Yongqi Gan

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from collections import defaultdict
import datetime as dt
import matplotlib.dates as mdates
import folium
import seaborn as sns
import missingno as msno
%matplotlib inline

import sklearn

from pandas import Series, DataFrame
from pylab import rcParams
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report

from scipy import stats
rcParams['figure.figsize'] = 10, 8
sns.set_style('whitegrid')

import scipy
from sklearn.feature_selection import RFECV
from sklearn.linear_model import LinearRegression
```

```
/usr/local/lib/python3.5/dist-packages/sklearn/cross_validation.py:41:
 DeprecationWarning: This module was deprecated in version 0.18 in favo
r of the model_selection module into which all the refactored classes a
nd functions are moved. Also note that the interface of the new CV iter
ators are different from that of this module. This module will be remov
ed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [2]:
```
#importing the dataframes in preparation for cleaning
d1 = pd.read_csv('1.csv', index_col=False)
d2 = pd.read_csv('2.csv', index_col=False)
d3 = pd.read_csv('3.csv', index_col=False)
d4 = pd.read_csv('4.csv', index_col=False)
d5 = pd.read_csv('5.csv', index_col=False)
d6 = pd.read_csv('6.csv', index_col=False)
d7 = pd.read_csv('7.csv', index_col=False)
d8 = pd.read_csv('8.csv', index_col=False)
d9 = pd.read_csv('9.csv', index_col=False)
```

In [3]:
```
#merging the first nine dataframes, casting FIPS to float
result = pd.merge(d1, d2, on=['County','State','FIPS'])
result = pd.merge(result, d3, on=['County','State','FIPS'])
result = pd.merge(result, d4, on=['County','State','FIPS'])
result = pd.merge(result, d5, on=['County','State','FIPS'])
result = pd.merge(result, d6, on=['County','State','FIPS'])
result = pd.merge(result, d7, on=['County','State','FIPS'])
result = pd.merge(result, d8, on=['County','State','FIPS'])
result = pd.merge(result, d9, on=['County','State','FIPS'])
temp = result['FIPS']
temp = [float(t) for t in temp]
result['FIPS'] = temp
```

In [4]:
```
#importing the last two dataframes and merging with each other
d10 = pd.read_csv('10.csv', index_col=False)
d11 = pd.read_csv('11.csv', index_col=False)
#merging by state
d_merge = d10.merge(d11, left_on = 'State', right_on = 'State', how = "l
eft")
```

In [5]:
```
d_merge.head()
```

Out[5]:

	FIPS	State	County	2010 Census Population	Population Estimate, 2011	Population Estimate, 2012	Population Estimate, 2013	Population Estimate, 2014	I
0	1001.0	Alabama	Autauga	54571.0	55255.0	55027.0	54792.0	54977.0	5
1	1003.0	Alabama	Baldwin	182265.0	186653.0	190403.0	195147.0	199745.0	2
2	1005.0	Alabama	Barbour	27457.0	27326.0	27132.0	26938.0	26763.0	2
3	1007.0	Alabama	Bibb	22915.0	22736.0	22645.0	22501.0	22511.0	2
4	1009.0	Alabama	Blount	57322.0	57707.0	57772.0	57746.0	57621.0	5

5 rows × 49 columns

In [6]:
```
#dropping the county and state columns
d_merge = d_merge.drop (['County', 'State'], axis = 1)
```

In [7]:
```
#merging with the rest of the data using FIPS
result = pd.merge(result, d_merge, on = "FIPS")
```

In [8]:
```
result.to_csv('fuck.csv')
```

Regression & Analysis

Linear Regression

In [9]:
```
# Select columns needed from DataFrame
data = result[['PCT_OBESE_ADULTS13', 'PCT_DIABETES_ADULTS13', 'POVRATE1
5', 'MEDHHINC15']]
```

In [10]:
```
# Drop NaN values as method of cleaning
data = data.dropna(axis=0)
data.shape
```

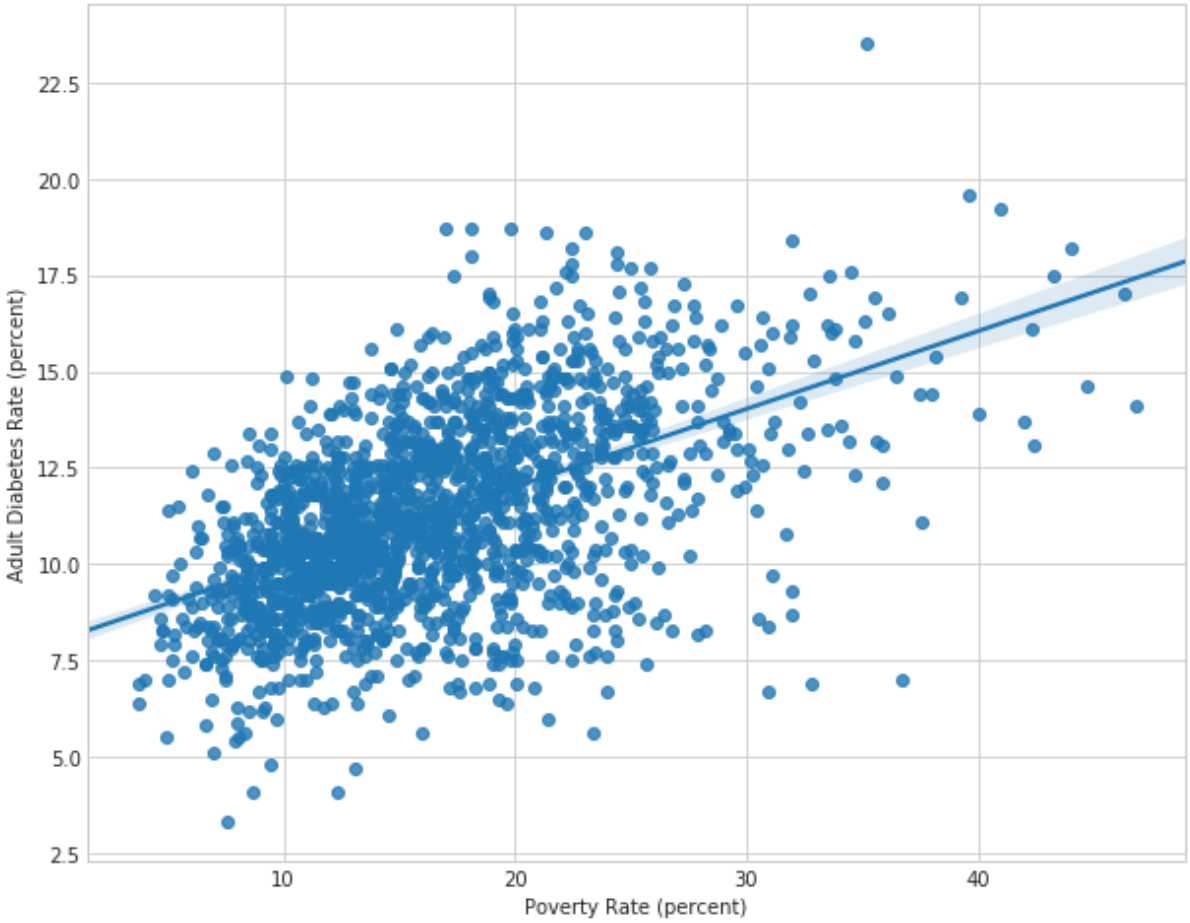
Out[10]: (3139, 4)

In [11]:
```
# Define our first set of train and validate data arrays using x and y a
s the columns we are correlating
x = 'POVRATE15'
y = 'PCT_DIABETES_ADULTS13'
x_train, x_valid, y_train, y_valid = train_test_split(data[x], data[y],
test_size=0.45, random_state=42)
```

In [12]:
```
# Plots scatter plot and regression line using Seaborn and calculates R-value using scipy
train = pd.DataFrame({'Poverty Rate (percent)': x_train, 'Adult Diabetes Rate (percent)': y_train})
sns.regplot('Poverty Rate (percent)', 'Adult Diabetes Rate (percent)', train)
slope, intercept, r_value, p_value, std_err = stats.linregress(x_train,y_train)
print("r-value: ", abs(r_value))

def model(val):
    global slope
    global intercept
    return slope * val + intercept
```

```
r-value:  0.528108380817
```



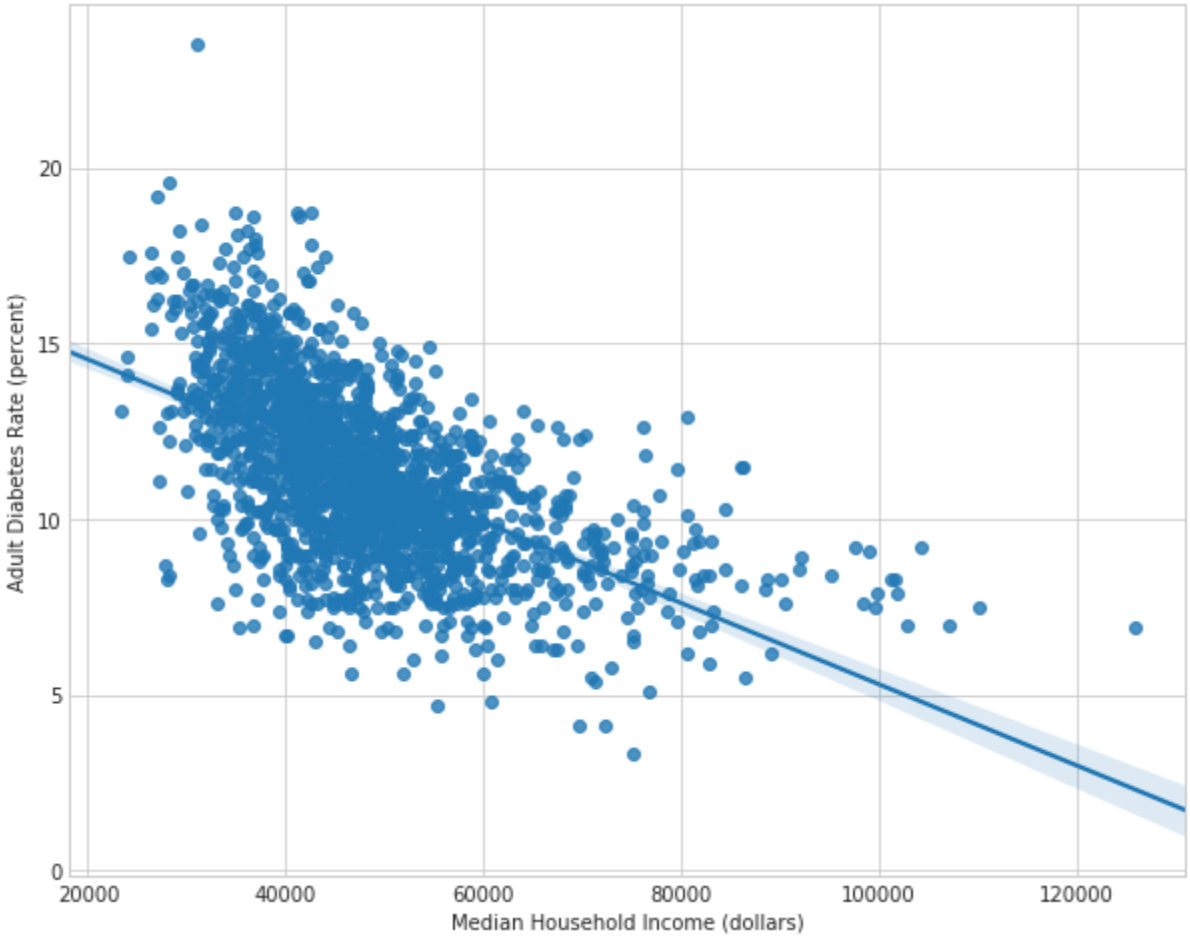
In [13]:
```
# Calculates MSE for our model
y_hat = [model(x) for x in x_valid]
mse = 1/len(y_valid)*np.dot((y_valid - y_hat),(y_valid - y_hat))
print("The MSE for the " + x + " to " + y + " is:", mse)
```

```
The MSE for the POVRATE15 to PCT_DIABETES_ADULTS13 is: 4.46768835252
```

In [14]:
```
# Define our second set of train and validate data arrays using x and y
 as the columns we are correlating
x = 'MEDHHINC15'
y = 'PCT_DIABETES_ADULTS13'
x_train, x_valid, y_train, y_valid = train_test_split(data[x], data[y],
test_size=0.45, random_state=42)
```

In [15]:
```
# Plots scatter plot and regression line using Seaborn and calculates R-
value using scipy
train = pd.DataFrame({'Median Household Income (dollars)': x_train, 'Adu
lt Diabetes Rate (percent)': y_train})
sns.regplot('Median Household Income (dollars)', 'Adult Diabetes Rate (p
ercent)', train)
slope, intercept, r_value, p_value, std_err = stats.linregress(x_train,y
_train)
print("r-value: ", abs(r_value))
```

```
r-value:  0.574991248313
```



In [16]:
```
# Calculates MSE for our model
y_hat = [model(x) for x in x_valid]
mse = 1/len(y_valid)*np.dot((y_valid - y_hat),(y_valid - y_hat))
print("The MSE for the " + x + " to " + y + " is:", mse)
```

```
The MSE for the MEDHHINC15 to PCT_DIABETES_ADULTS13 is: 4.17551811949
```

Logistic Regression

We wanted to look at the difference between urban and rural areas. Specifically, we wanted to find out if the densities of grocery stores, supermarkets, convenience stores, specialized food stores, and restaurants could be used to predict whether a county is urban or rural.

In [17]:
```
#subsetting the necessary data to create a new dataframe
df_logit = result[['GROCPTH09', 'SUPERCPTH09', 'CONVSPTH09', 'SPECSPTH0
9', 'FFRPTH09', 'FSRPTH09', 'METRO13']]
```

In [18]:
```
df_logit.head()
```

Out[18]:

	GROCPTH09	SUPERCPTH09	CONVSPTH09	SPECSPTH09	FFRPTH09	FSRPTH09	MI
0	0.110834	0.018472	0.535698	0.036945	0.554170	0.628059	1
1	0.133775	0.033444	0.663300	0.117053	0.624282	1.125938	1
2	0.180786	0.000000	0.506201	0.072314	0.759301	0.433887	0
3	0.261540	0.043590	0.828211	0.000000	0.305131	0.261540	1
4	0.104637	0.017440	0.540625	0.034879	0.418549	0.331351	1

In [19]:
```
#checking for null values
df_logit.isnull().sum()
```

Out[19]:
```
GROCPTH09      0
SUPERCPTH09    0
CONVSPTH09     0
SPECSPTH09     0
FFRPTH09       0
FSRPTH09       0
METRO13        0
dtype: int64
```

In [20]:
```
#separating the endogeneous (experimental) variable from the exogeneous
 (explanatory) variables
X = df_logit.ix[:,(0,1,2,3,4,5)].values
y = df_logit.ix[:,6].values
```

```
/usr/local/lib/python3.5/dist-packages/ipykernel_launcher.py:2: Depreca
tionWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is
-deprecated
```

In [21]:
```
#creating a training and test dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
.3, random_state=25)
```

In [22]:
```
#creating the logistic regression model and training it on the training
 set
LogReg = LogisticRegression()
LogReg.fit(X_train, y_train)
```

Out[22]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=
1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.00
01,
          verbose=0, warm_start=False)
```

In [23]:
```
#predicting the y-values of the test dataset
y_pred = LogReg.predict(X_test)
```

In [24]:
```
#creating the confusion matrix to view the accuracy of our predictions
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
confusion_matrix
```

Out[24]:
```
array([[501,  98],
       [131, 213]])
```

In [25]:
```
#showing the classification report
print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

          0       0.79      0.84      0.81       599
          1       0.68      0.62      0.65       344

avg / total       0.75      0.76      0.75       943
```

We can see that the factors we considered were reasonably good at predicting whether a county was metropolitain. We had an overall prediction success of 75%. It is worth noting that there were more counties incorrectly classified as rural than metropolitain.

Linear Regression</h>

In [26]:
```
#Select and rename the columns we want
good = result[['FFR09', 'FFR14', 'GROC09', 'GROC14', 'SUPERC09', 'SUPERC
14', 'PCT_OBESE_ADULTS08', 'PCT_OBESE_ADULTS13', 'RECFAC09', 'RECFAC14',
 'PCT_DIABETES_ADULTS08', 'PCT_DIABETES_ADULTS13', 'POVRATE15']]
good.columns = ['fast9', 'fast14', 'groc9', 'groc14', 'super9', 'super1
4', 'obese9', 'obese14', 'rec9', 'rec14', 'dia9', 'dia14', 'pov']
good.head()
```

Out[26]:

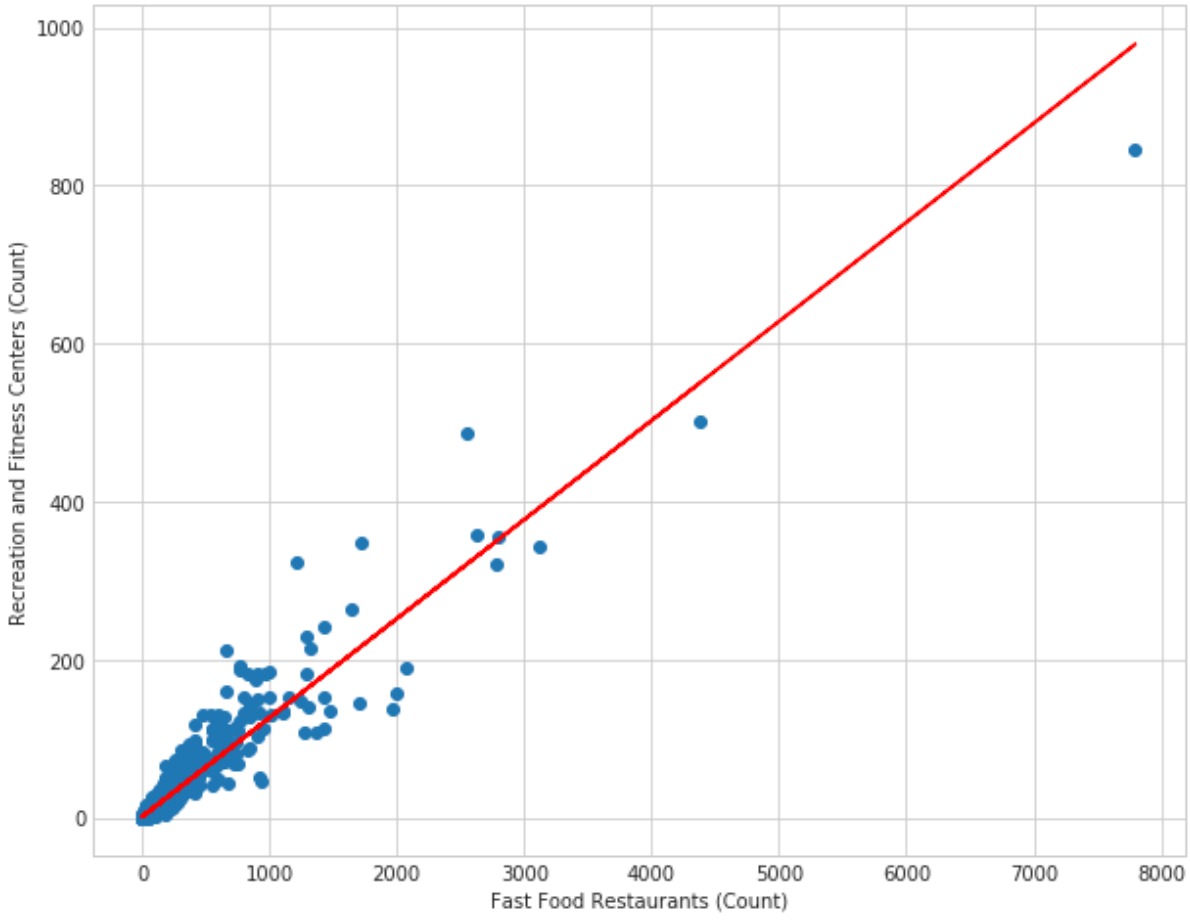
	fast9	fast14	groc9	groc14	super9	super14	obese9	obese14	rec9	rec14	dia9	dia1
0	30	36	6	4	1	1	31.5	34.1	4	5	11.4	13.0
1	112	132	24	29	6	6	26.2	27.4	18	25	9.8	10.4
2	21	22	5	5	0	1	37.6	44.4	1	0	13.6	18.4
3	7	5	6	5	1	1	32.3	40.3	1	1	11.1	14.8
4	24	21	6	6	1	1	31.9	34.6	3	3	11.4	14.1

In [27]:
```
#drop rows with missing data
good = good.dropna()
good.shape
```

Out[27]:
```
(3134, 13)
```

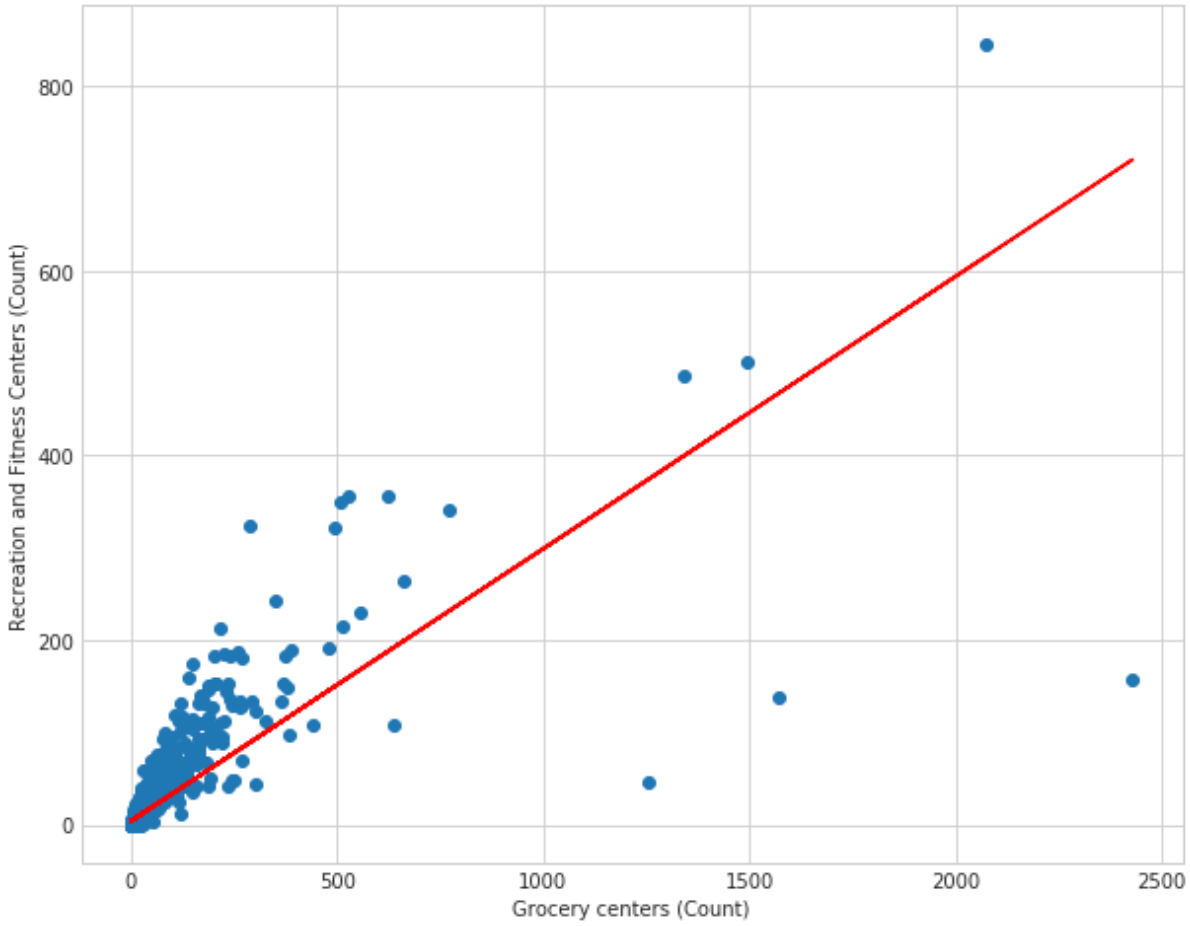
In [28]:
```
#Get the columns
fast9 = good['fast9']
fast14 = good['fast14']
groc9 = good['groc9']
groc14 = good['groc14']
super9 = list(good['super9'])
super14 = list(good['super14'])
obese9 = good['obese9']
obese14 = good['obese14']
rec9 = good['rec14']
rec14 = good['rec14']
dia9 = good['dia9']
dia14 = good['dia14']
pov = good['pov']
```

In [29]:
```
#Fast food restaurants vs Rec/Fitness Centers
current = fast14
measure = rec14
plt.figure(1)
plt.scatter(current,measure)
slope, intercept, r_value, p_value, std_err = scipy.stats.linregress(cur
rent, measure)
abline_values = [slope * i + intercept for i in current]
plt.plot(current, abline_values, 'r')
plt.ylabel("Recreation and Fitness Centers (Count)")
plt.xlabel("Fast Food Restaurants (Count)")
plt.show()
print("R-value: " + str(r_value))
```



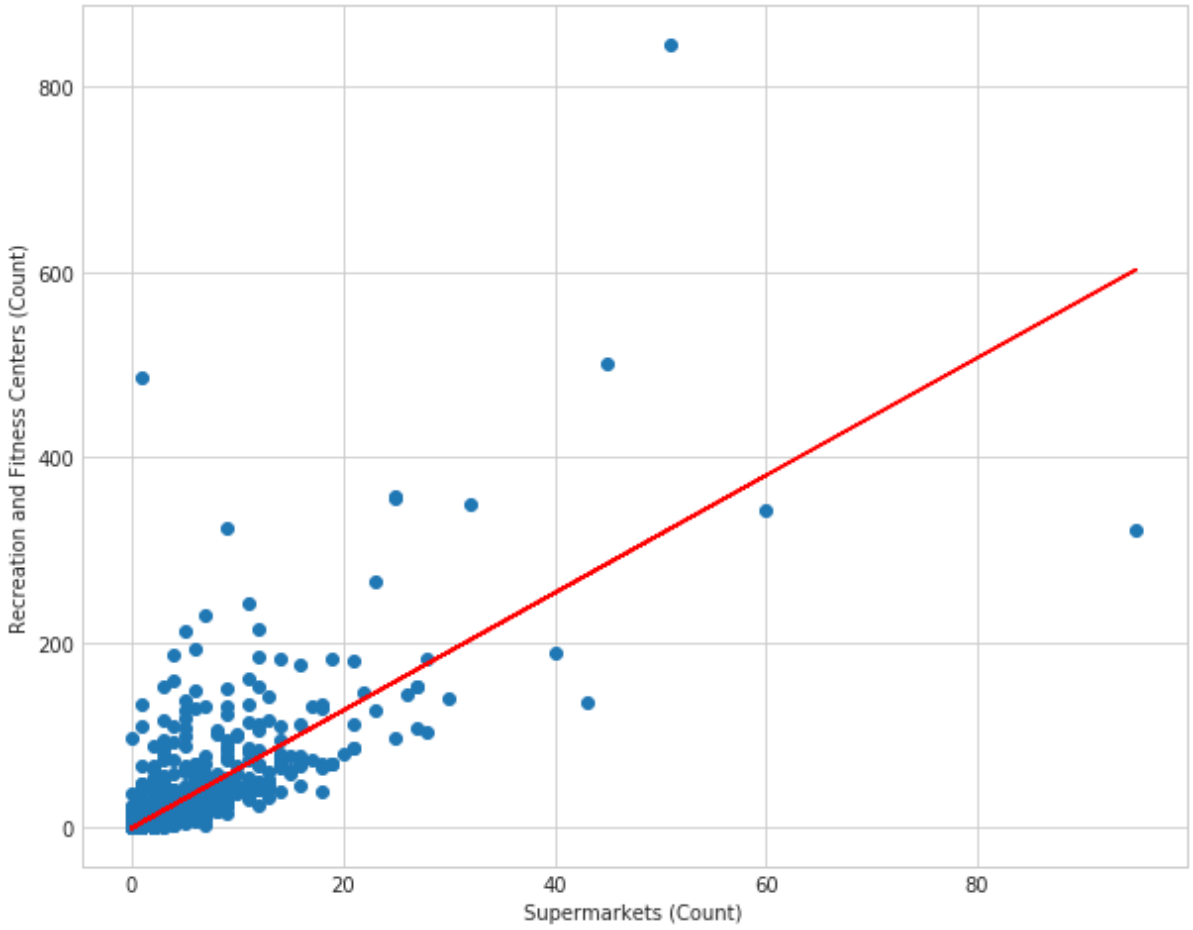

```
R-value: 0.951256158719
```

In [30]:
```
#Grocery centers vs Rec/Fitness Centers
current = groc14
measure = rec14
plt.figure(1)
plt.scatter(current,measure)
slope, intercept, r_value, p_value, std_err = scipy.stats.linregress(cur
rent, measure)
abline_values = [slope * i + intercept for i in current]
plt.plot(current, abline_values, 'r')
plt.ylabel("Recreation and Fitness Centers (Count)")
plt.xlabel("Grocery centers (Count)")
plt.show()
print("R-value: " + str(r_value))
```




```
R-value: 0.793643614909
```

In [31]:
```
#Supermarkets vs Rec/Fitness Centers
current = super14
measure = rec14
plt.figure(1)
plt.scatter(current,measure)
slope, intercept, r_value, p_value, std_err = scipy.stats.linregress(cur
rent, measure)
abline_values = [slope * i + intercept for i in current]
plt.plot(current, abline_values, 'r')
plt.ylabel("Recreation and Fitness Centers (Count)")
plt.xlabel("Supermarkets (Count)")
plt.show()
print("R-value: " + str(r_value))
```




```
R-value: 0.755163330649
```