In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
data = pd.read_csv('acs_ny.csv')
```

Looking at our visualizations that we did first, we use pearson's R to compare continuous variables and sperman's r to compare categorical variables.

In [3]:
```
data.columns
```

Out[3]:
```
Index(['Acres', 'FamilyIncome', 'FamilyType', 'NumBedrooms', 'NumChildren',
       'NumPeople', 'NumRooms', 'NumUnits', 'NumVehicles', 'NumWorkers',
       'OwnRent', 'YearBuilt', 'HouseCosts', 'ElectricBill', 'FoodStamp',
       'HeatingFuel', 'Insurance', 'Language'],
      dtype='object')
```

I am going to check which categorical variables have more impact on FamilyIncome

In [6]:
```
print (data[["Acres", "FamilyIncome"]].groupby(['Acres'], as_index=False).mean
())
```

```
    Acres   FamilyIncome
0     10+   97389.628486
1  10-Jan  111636.585909
2   Sub 1  110671.252776
```

In [9]:
```
print (data[["FamilyType", "FamilyIncome"]].groupby(['FamilyType'], as_index=F
alse).mean())
```

```
    FamilyType   FamilyIncome
0  Female Head   60850.583282
1    Male Head   74263.688638
2      Married  121356.840118
```

In [10]:
```
print (data[["NumBedrooms", "FamilyIncome"]].groupby(['NumBedrooms'], as_index
=False).mean())
```

```
   NumBedrooms   FamilyIncome
0            0   74552.121212
1            1   61602.025974
2            2   74946.326669
3            3   97750.189341
4            4  131370.870893
5            5  163146.053687
6            8  174883.902033
```

In [11]:
```
print (data[["NumChildren", "FamilyIncome"]].groupby(['NumChildren'], as_index
=False).mean())
```

```
    NumChildren   FamilyIncome
0             0  106584.894997
1             1  108440.171665
2             2  119009.694826
3             3  122769.560856
4             4  103504.732824
5             5   99076.380952
6             6  100900.648649
7             7   44404.666667
8             8   52500.000000
9             9   35200.000000
10           10   50915.000000
11           12   81600.000000
```

In [12]:
```
print (data[["NumPeople", "FamilyIncome"]].groupby(['NumPeople'], as_index=Fal
se).mean())
```

```
    NumPeople   FamilyIncome
0           2   97046.045656
1           3  108412.038269
2           4  123143.581521
3           5  123610.667961
4           6  117470.004444
5           7  111522.473846
6           8  111375.826923
7           9  102453.061728
8          10  105157.105263
9          11   94638.888889
10         12  106592.500000
11         13  108900.000000
12         15   68300.000000
13         16  200200.000000
14         18  157960.000000
```

In [13]:
```
print (data[["NumUnits", "FamilyIncome"]].groupby(['NumUnits'], as_index=False
).mean())
```

```
          NumUnits   FamilyIncome
0      Mobile home   47483.238558
1  Single attached  100504.674507
2  Single detached  113771.442499
```

In [14]:
```
print (data[["NumVehicles", "FamilyIncome"]].groupby(['NumVehicles'], as_index
=False).mean())
```

```
   NumVehicles   FamilyIncome
0            0   64546.568828
1            1   74806.600494
2            2  113359.209624
3            3  128760.206499
4            4  146387.449630
5            5  157359.886792
6            6  150355.096774
```

In [15]:
```
print (data[["NumWorkers", "FamilyIncome"]].groupby(['NumWorkers'], as_index=F
alse).mean())
```

```
   NumWorkers   FamilyIncome
0           0   51550.829940
1           1   89827.271090
2           2  121339.056694
3           3  138042.471258
```

In [16]:
```
print (data[["OwnRent", "FamilyIncome"]].groupby(['OwnRent'], as_index=False).
mean())
```

```
    OwnRent   FamilyIncome
0  Mortgage  117122.223467
1  Outright  112983.694268
2    Rented   55143.930000
```

In [18]:
```
print (data[["YearBuilt", "FamilyIncome"]].groupby(['YearBuilt'], as_index=Fal
se).mean())
```

```
      YearBuilt   FamilyIncome
0            15   89607.333333
1     1940-1949  100841.288040
2     1950-1959  113841.226828
3     1960-1969  114866.615635
4     1970-1979  106516.868085
5     1980-1989  114882.158690
6     1990-1999  114773.528079
7     2000-2004  122748.533463
8          2005  136434.094595
9          2006  129836.977901
10         2007  128283.935484
11         2008  117252.000000
12         2009  111362.457831
13         2010  156745.918367
14  Before 1939  102600.280216
```

In [19]:
```
print (data[["FoodStamp", "FamilyIncome"]].groupby(['FoodStamp'], as_index=Fal
se).mean())
```

```
  FoodStamp   FamilyIncome
0        No  115476.583961
1       Yes   48754.160361
```

In [20]:
```
print (data[["HeatingFuel", "FamilyIncome"]].groupby(['HeatingFuel'], as_index
=False).mean())
```

```
   HeatingFuel   FamilyIncome
0         Coal   77704.096154
1  Electricity   97564.523438
2          Gas  111222.965247
3         None   98398.260870
4          Oil  117499.955730
5        Other   88582.659574
6        Solar  125660.000000
7         Wood   79155.990818
```

In [21]:
```
print (data[["Language", "FamilyIncome"]].groupby(['Language'], as_index=False
).mean())
```

```
         Language   FamilyIncome
0   Asian Pacific  126038.225124
1         English  108620.361166
2           Other  134087.292254
3  Other European  125834.625864
4         Spanish   98355.357783
```

What I see here are how the averages of the categories have an effect on FamilyIncome.

So far, I am willing to drop these variables but will further analyze them:

Acres and YearBuilt

In [22]:
```
data.corr() #uses pearson's r to calculate correlation.
```

Out[22]:

	FamilyIncome	NumBedrooms	NumChildren	NumPeople	NumRooms	I
FamilyIncome	1.000000	0.243360	0.034841	0.079866	0.277138	(
NumBedrooms	0.243360	1.000000	0.173064	0.324647	0.617191	(
NumChildren	0.034841	0.173064	1.000000	0.679914	0.105397	-
NumPeople	0.079866	0.324647	0.679914	1.000000	0.164520	(
NumRooms	0.277138	0.617191	0.105397	0.164520	1.000000	(
NumVehicles	0.208376	0.189318	-0.086460	0.158397	0.204736	1
NumWorkers	0.230320	0.137775	-0.003459	0.308212	0.107081	(
HouseCosts	0.465851	0.258925	0.146380	0.180947	0.214482	(
ElectricBill	0.245494	0.240851	0.108400	0.224866	0.191467	(
Insurance	0.409448	0.270862	0.000276	0.046895	0.255811	(

The above correlation is only for numeric variables. Here, I can see that None of the variables really correlate with the independent variable. HouseCosts has the most correlation, followed by insurance.

For the dependent variables, The most correlated variables are NumChildren and NumPeople, NumRooms and NumBedrooms. So, we will delete one of each to take care of multicollinearity. However, since they are categorical variables, we will use spearman's r first.

Looking at the relationship with FamilyIncome, we are going to delete NumChildren and NumBedrooms because they are least correlated.

We are still going to do more data wrangling before we delete anything.

In [26]:
```
from scipy.stats import spearmanr
spearmanr_coefficient, p_value = spearmanr(data['NumChildren'],data['NumPeopl
e'])
print(spearmanr_coefficient)
```

```
0.678027299248
```

In [29]:
```
spearmanr_coefficient, p_value = spearmanr(data['NumRooms'],data['NumBedrooms'
])
print(spearmanr_coefficient)
```

```
0.664319110005
```

Spearmanr also gave us a high correlation between the independent variables.

Now, we check the independent vs dependent variables for categorical variables.

In [30]:
```
spearmanr_coefficient, p_value = spearmanr(data['Acres'],data['FamilyIncome'])
print(spearmanr_coefficient)
```

```
0.0488871572277

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\scipy\stats\stats.py:253: RuntimeWarning: The input array could not b
e properly checked for nan values. nan values will be ignored.
  "values. nan values will be ignored.", RuntimeWarning)
```

In [31]:
```
spearmanr_coefficient, p_value = spearmanr(data['FamilyType'],data['FamilyInco
me'])
print(spearmanr_coefficient)
```

```
0.328725070153

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\scipy\stats\stats.py:253: RuntimeWarning: The input array could not b
e properly checked for nan values. nan values will be ignored.
  "values. nan values will be ignored.", RuntimeWarning)
```

In [32]:
```
spearmanr_coefficient, p_value = spearmanr(data['NumBedrooms'],data['FamilyInc
ome'])
print(spearmanr_coefficient)
```

```
0.262060072174
```

In [33]:
```
spearmanr_coefficient, p_value = spearmanr(data['NumChildren'],data['FamilyInc
ome'])
print(spearmanr_coefficient)
```

```
0.0131632659587
```

In [34]:
```
spearmanr_coefficient, p_value = spearmanr(data['NumPeople'],data['FamilyIncom
e'])
print(spearmanr_coefficient)
```

```
0.126566094469
```

In [35]:
```
spearmanr_coefficient, p_value = spearmanr(data['NumRooms'],data['FamilyIncom
e'])
print(spearmanr_coefficient)
```

```
0.304114737035
```

In [36]:
```
spearmanr_coefficient, p_value = spearmanr(data['NumUnits'],data['FamilyIncom
e'])
print(spearmanr_coefficient)
```

```
0.133558239642
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\scipy\stats\stats.py:253: RuntimeWarning: The input array could not b
e properly checked for nan values. nan values will be ignored.
  "values. nan values will be ignored.", RuntimeWarning)
```

In [37]:
```
spearmanr_coefficient, p_value = spearmanr(data['NumVehicles'],data['FamilyInc
ome'])
print(spearmanr_coefficient)
```

```
0.324239036965
```

In [38]:
```
spearmanr_coefficient, p_value = spearmanr(data['NumWorkers'],data['FamilyInco
me'])
print(spearmanr_coefficient)
```

```
0.403427713964
```

In [39]:
```
spearmanr_coefficient, p_value = spearmanr(data['OwnRent'],data['FamilyIncome'
])
print(spearmanr_coefficient)
```

```
-0.301953003731
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\scipy\stats\stats.py:253: RuntimeWarning: The input array could not b
e properly checked for nan values. nan values will be ignored.
  "values. nan values will be ignored.", RuntimeWarning)
```

In [40]:
```
spearmanr_coefficient, p_value = spearmanr(data['YearBuilt'],data['FamilyIncom
e'])
print(spearmanr_coefficient)
```

```
-0.0558708465521
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\scipy\stats\stats.py:253: RuntimeWarning: The input array could not b
e properly checked for nan values. nan values will be ignored.
  "values. nan values will be ignored.", RuntimeWarning)
```

In [41]:
```
spearmanr_coefficient, p_value = spearmanr(data['HouseCosts'],data['FamilyInco
me']) #continuous variable. So, pearson's R value is preffered.
print(spearmanr_coefficient)
```

```
0.437340893131
```

In [42]:
```
spearmanr_coefficient, p_value = spearmanr(data['ElectricBill'],data['FamilyIn
come']) #continuous variable. So, pearson's R value is preffered.
print(spearmanr_coefficient)
```

```
0.233889009602
```

In [43]:
```
spearmanr_coefficient, p_value = spearmanr(data['FoodStamp'],data['FamilyIncom
e'])
print(spearmanr_coefficient)
```

```
-0.281969502572
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\scipy\stats\stats.py:253: RuntimeWarning: The input array could not b
e properly checked for nan values. nan values will be ignored.
  "values. nan values will be ignored.", RuntimeWarning)
```

In [44]:
```
spearmanr_coefficient, p_value = spearmanr(data['HeatingFuel'],data['FamilyInc
ome'])
print(spearmanr_coefficient)
```

```
0.00566231554938
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\scipy\stats\stats.py:253: RuntimeWarning: The input array could not b
e properly checked for nan values. nan values will be ignored.
  "values. nan values will be ignored.", RuntimeWarning)
```

In [45]:
```
spearmanr_coefficient, p_value = spearmanr(data['Insurance'],data['FamilyInc
ome']) #continuous variable. So, pearson's R value is preffered.
print(spearmanr_coefficient)
```

```
0.409948285597
```

In [46]:
```
spearmanr_coefficient, p_value = spearmanr(data['Language'],data['FamilyIncom
e'])
print(spearmanr_coefficient)
```

```
-0.0140644729046

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\scipy\stats\stats.py:253: RuntimeWarning: The input array could not b
e properly checked for nan values. nan values will be ignored.
  "values. nan values will be ignored.", RuntimeWarning)
```

Thus, before when we used mean to check the relationship between independent variables and dependent variables, we found out that Acres and YearBuilt did not correlate with FamilyIncome.

With spearmanr, we can justify that and thus remove it.

Here, we also found out that Language, HeatingFuel and NumChildren had less than 0.1 for correlation, so we delete those too.

Overall, none of the variables really had any significant correlation with the FamilyIncome, so a different method of Regression should be used. Since, we are only focusing on Multiple Regression for this exercise, we will carry on after deleting the following variables.

Acres, YearBuilt, Language, HeatingFuel, NumChildren, NumBedrooms.

The number to beat is 0.30

In [47]:
```
data = data.drop(['Acres', 'YearBuilt', 'Language', 'HeatingFuel', 'NumChildre
n', 'NumBedrooms'],axis = 1)
```

In [48]:
```
data.head()
```

Out[48]:

	FamilyIncome	FamilyType	NumPeople	NumRooms	NumUnits	NumVehicles	NumW
0	150	Married	3	9	Single detached	1	0
1	180	Female Head	4	6	Single detached	2	0
2	280	Female Head	2	8	Single detached	3	1
3	330	Female Head	2	4	Single detached	1	0
4	330	Male Head	2	5	Single attached	1	0

In [50]:
```
y = data.iloc[:, 0:1]
x = data.iloc[:, 1:]
```

In [53]:
```
encoded_x = pd.get_dummies(x, drop_first = True) #to get rid of dummy variable
 trap
```

In [56]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(encoded_x, y, test_size =
0.2, random_state = 0)
```

In [57]:
```
import statsmodels.api as sm
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\statsmodels\compat\pandas.py:56: FutureWarning: The pandas.core.datet
ools module is deprecated and will be removed in a future version. Please use
the pandas.tseries module instead.
  from pandas.core import datetools
```

In [58]:
```
X = encoded_x
Y = y
## fit a OLS model with intercept on TV and Radio
X = sm.add_constant(X)
est = sm.OLS(Y, X).fit()

est.summary()
```

Out[58]:

OLS Regression Results

Dep. Variable:	FamilyIncome	**R-squared:**	0.339
Model:	OLS	**Adj. R-squared:**	0.339
Method:	Least Squares	**F-statistic:**	832.5
Date:	Sat, 14 Apr 2018	**Prob (F-statistic):**	0.00
Time:	12:42:53	**Log-Likelihood:**	-2.8953e+05
No. Observations:	22745	**AIC:**	5.791e+05
Df Residuals:	22730	**BIC:**	5.792e+05
Df Model:	14		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-5.283e+04	3906.574	-13.524	0.000	-6.05e+04	-4.52e+04
NumPeople	-4945.9979	435.753	-11.350	0.000	-5800.103	-4091.892
NumRooms	5496.4229	250.747	21.920	0.000	5004.941	5987.905
NumVehicles	6414.4310	652.424	9.832	0.000	5135.636	7693.226
NumWorkers	1.757e+04	759.716	23.125	0.000	1.61e+04	1.91e+04
HouseCosts	27.5695	0.573	48.104	0.000	26.446	28.693
ElectricBill	46.0162	5.617	8.193	0.000	35.007	57.025
Insurance	18.3831	0.675	27.222	0.000	17.059	19.707
FamilyType_Male Head	8009.7917	2812.207	2.848	0.004	2497.673	1.35e+04
FamilyType_Married	3.052e+04	1657.173	18.416	0.000	2.73e+04	3.38e+04
NumUnits_Single attached	5020.7158	3565.433	1.408	0.159	-1967.776	1.2e+04
NumUnits_Single detached	4802.5017	3175.383	1.512	0.130	-1421.466	1.1e+04
OwnRent_Outright	5.881e+04	6762.220	8.697	0.000	4.56e+04	7.21e+04
OwnRent_Rented	6444.8143	2002.114	3.219	0.001	2520.534	1.04e+04
FoodStamp_Yes	-1.378e+04	2253.118	-6.116	0.000	-1.82e+04	-9363.823

Omnibus:	16174.698	**Durbin-Watson:**	0.559
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	427169.290
Skew:	3.110	**Prob(JB):**	0.00
Kurtosis:	23.299	**Cond. No.**	2.77e+04

We use backward elimination with a p-value <= 0.05 criteria.

In [59]:
```
encoded_x = encoded_x.drop(['NumUnits_Single attached'], axis = 1)
X = encoded_x
Y = y
## fit a OLS model with intercept on TV and Radio
X = sm.add_constant(X)
est = sm.OLS(Y, X).fit()

est.summary()
```

Out[59]:

OLS Regression Results

Dep. Variable:	FamilyIncome	**R-squared:**	0.339
Model:	OLS	**Adj. R-squared:**	0.339
Method:	Least Squares	**F-statistic:**	896.4
Date:	Sat, 14 Apr 2018	**Prob (F-statistic):**	0.00
Time:	12:46:16	**Log-Likelihood:**	-2.8953e+05
No. Observations:	22745	**AIC:**	5.791e+05
Df Residuals:	22731	**BIC:**	5.792e+05
Df Model:	13		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-4.908e+04	2857.476	-17.176	0.000	-5.47e+04	-4.35e+04
NumPeople	-4935.4520	435.698	-11.328	0.000	-5789.450	-4081.454
NumRooms	5498.8402	250.747	21.930	0.000	5007.359	5990.321
NumVehicles	6316.9860	648.758	9.737	0.000	5045.377	7588.595
NumWorkers	1.763e+04	758.518	23.241	0.000	1.61e+04	1.91e+04
HouseCosts	27.6643	0.569	48.605	0.000	26.549	28.780
ElectricBill	46.0556	5.617	8.200	0.000	35.046	57.065
Insurance	18.3947	0.675	27.241	0.000	17.071	19.718
FamilyType_Male Head	7914.0288	2811.446	2.815	0.005	2403.403	1.34e+04
FamilyType_Married	3.046e+04	1656.600	18.384	0.000	2.72e+04	3.37e+04
NumUnits_Single detached	1011.8917	1684.329	0.601	0.548	-2289.508	4313.291
OwnRent_Outright	5.997e+04	6711.968	8.935	0.000	4.68e+04	7.31e+04
OwnRent_Rented	6423.3120	2002.099	3.208	0.001	2499.061	1.03e+04
FoodStamp_Yes	-1.395e+04	2250.006	-6.199	0.000	-1.84e+04	-9537.896

Omnibus:	16167.427	**Durbin-Watson:**	0.560
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	426638.972
Skew:	3.109	**Prob(JB):**	0.00
Kurtosis:	23.286	**Cond. No.**	2.75e+04

In [60]:
```
encoded_x = encoded_x.drop(['NumUnits_Single detached'], axis = 1)
X = encoded_x
Y = y
## fit a OLS model with intercept on TV and Radio
X = sm.add_constant(X)
est = sm.OLS(Y, X).fit()

est.summary()
```

Out[60]:

OLS Regression Results

Dep. Variable:	FamilyIncome	**R-squared:**	0.339
Model:	OLS	**Adj. R-squared:**	0.339
Method:	Least Squares	**F-statistic:**	971.0
Date:	Sat, 14 Apr 2018	**Prob (F-statistic):**	0.00
Time:	12:46:54	**Log-Likelihood:**	-2.8953e+05
No. Observations:	22745	**AIC:**	5.791e+05
Df Residuals:	22732	**BIC:**	5.792e+05
Df Model:	12		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-4.84e+04	2620.529	-18.468	0.000	-5.35e+04	-4.33e+04
NumPeople	-4945.2743	435.385	-11.358	0.000	-5798.659	-4091.890
NumRooms	5521.3471	247.929	22.270	0.000	5035.389	6007.305
NumVehicles	6373.5992	641.868	9.930	0.000	5115.494	7631.705
NumWorkers	1.761e+04	758.106	23.234	0.000	1.61e+04	1.91e+04
HouseCosts	27.6568	0.569	48.604	0.000	26.541	28.772
ElectricBill	46.0190	5.616	8.194	0.000	35.011	57.027
Insurance	18.3967	0.675	27.244	0.000	17.073	19.720
FamilyType_Male Head	7912.7812	2811.405	2.815	0.005	2402.235	1.34e+04
FamilyType_Married	3.046e+04	1656.509	18.391	0.000	2.72e+04	3.37e+04
OwnRent_Outright	5.926e+04	6608.528	8.968	0.000	4.63e+04	7.22e+04
OwnRent_Rented	6275.9427	1986.987	3.159	0.002	2381.313	1.02e+04
FoodStamp_Yes	-1.399e+04	2249.131	-6.218	0.000	-1.84e+04	-9576.636

Omnibus:	16166.927	**Durbin-Watson:**	0.560
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	426661.504
Skew:	3.108	**Prob(JB):**	0.00
Kurtosis:	23.287	**Cond. No.**	2.70e+04

There was no any significant improvement of this model too.

For Model 5, we will try PCA and MFA to see if they have any significant changes.