Homeless Predictions

In [2]:

```
# Import Packages
from ipywidgets import interact, interactive, fixed
from IPython.html.widgets.widget_int import IntSlider
import ipywidgets as widgets
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.cross_validation import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np
import warnings
warnings.simplefilter(action = "ignore")

# Read data
df = pd.read_csv('data/homeless_df_forpred.csv')

# Make variables and train/test split
X = df[['parks','shelter_beds','building_permits','foreclosures']]
y = df['total_people'].values
# X = StandardScaler().fit_transform(X,y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

# Linear Regression
lr = LinearRegression(normalize=True, n_jobs=-1)
lr.fit(X_train, y_train)
# svr_lin = SVR(kernel='linear', C=1e3)
# svr_lin.fit(X_train, y_train)

# Create functions for multiplier -- min and max
def miner(x):
    return x/1.9

def maxer(x):
    return x*1.9

# Pass values through the multiplier functions for each feature
min_parks = miner(X['parks'].values)
max_parks = maxer(X['parks'].values)

min_shbeds = miner(X['shelter_beds'].values)
max_shbeds = maxer(X['shelter_beds'].values)
```

```
min_bp = miner(X['building_permits'].values)
max_bp = maxer(X['building_permits'].values)

min_fc = miner(X['foreclosures'].values)
max_fc = maxer(X['foreclosures'].values)

# Slider Values
parks_slide = IntSlider(min=min(min_parks), max=max(max_parks), step=1,
value=X.parks.mean())
shelter_beds_slide = IntSlider(min=min(min_shbeds), max=max(max_shbeds), step=1,
value=X.shelter_beds.mean())
building_permits_slide = IntSlider(min=min(min_bp), max=max(max_bp), step=1,
value=X.building_permits.mean())
foreclosures_slide = IntSlider(min=min(min_fc), max=max(max_fc), step=1,
value=X.foreclosures.mean())

# Make predictions
def predictor(parks_slide, shelter_beds_slide, building_permits_slide, foreclosures_slide):
    stuff = np.array([parks_slide.value, shelter_beds_slide.value,
building_permits_slide.value, foreclosures_slide.value])
    y_preds = lr.predict(stuff)
#     y_preds = svr_lin.predict(stuff)
    return round(sum(y_preds))

# Print Values and Predictions
def printer(parks, shelter_beds, building_permits, foreclosures):
    print 'Parks:', parks, '\nShelter Beds:', shelter_beds, '\nBuilding Permits',
building_permits, '\nForeclosures', foreclosures, '\n\nTOTAL HOMELESS:',
predictor(parks_slide, shelter_beds_slide, building_permits_slide, foreclosures_slide)

interact(printer,parks=parks_slide, shelter_beds=shelter_beds_slide,
building_permits=building_permits_slide, foreclosures=foreclosures_slide)
```

```
Parks: 7
Shelter Beds: 12
Building Permits 3141
Foreclosures 58

TOTAL HOMELESS: 18.0
```

Out[2]:

```
<function __main__.printer>
```