**Importing packages**[**¶**](#gjdgxs)

In [1]:

**from** **pandas** **import** Series, DataFrame  
**import** **pandas** **as** **pd**  
%**pylab** inline  
*# Special packages*  
**import** **statsmodels.api** **as** **sm**  
**from** **patsy** **import** dmatrices

Populating the interactive namespace from numpy and matplotlib

C:\Users\ericl\Anaconda2\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The pandas.core.datetools module is deprecated and will be removed in a future version. Please use the pandas.tseries module instead.  
 from pandas.core import datetools

**base data**[**¶**](#30j0zll)

In [2]:

df = pd.read\_csv('Mass Shootings Dataset.csv')  
df[:5]

Out[2]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **S#** | **Title** | **Location** | **Date** | **Summary** | **Fatalities** | **Injured** | **Total victims** | **Mental Health Issues** | **Race** | **Gender** | **Latitude** | **Longitude** |
| **0** | 1 | Las Vegas Strip mass shooting | Las Vegas, NV | 10/1/2017 | NaN | 58 | 515 | 573 | Unclear | NaN | NaN | NaN | NaN |
| **1** | 2 | San Francisco UPS shooting | San Francisco, CA | 6/14/2017 | Jimmy Lam, 38, fatally shot three coworkers an... | 3 | 2 | 5 | Yes | Asian | M | NaN | NaN |
| **2** | 3 | Pennsylvania supermarket shooting | Tunkhannock, PA | 6/7/2017 | Randy Stair, a 24-year-old worker at Weis groc... | 3 | 0 | 3 | Unclear | White | M | NaN | NaN |
| **3** | 4 | Florida awning manufacturer shooting | Orlando, Florida | 6/5/2017 | John Robert Neumann, Jr., 45, a former employe... | 5 | 0 | 5 | Unclear | NaN | M | NaN | NaN |
| **4** | 5 | Rural Ohio nursing home shooting | Kirkersville, Ohio | 5/12/2017 | Thomas Hartless, 43, shot and killed a former ... | 3 | 0 | 3 | Yes | White | M | NaN | NaN |

**parses through location to get state**[**¶**](#1fob9te)

In [3]:

**def** get\_state(f):  
 state = ''  
 **try**:  
 state = f.split(",")[1]  
 **except**:  
 *#state = f.split(",")[0]*  
 state = " "  
 **return** state  
   
df['State'] = df['Location'].apply(get\_state)  
*#df['State'].dropna()*  
mask = df['State'] != ' '  
df = df[mask]  
df[:3]

Out[3]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **S#** | **Title** | **Location** | **Date** | **Summary** | **Fatalities** | **Injured** | **Total victims** | **Mental Health Issues** | **Race** | **Gender** | **Latitude** | **Longitude** | **State** |
| **0** | 1 | Las Vegas Strip mass shooting | Las Vegas, NV | 10/1/2017 | NaN | 58 | 515 | 573 | Unclear | NaN | NaN | NaN | NaN | NV |
| **1** | 2 | San Francisco UPS shooting | San Francisco, CA | 6/14/2017 | Jimmy Lam, 38, fatally shot three coworkers an... | 3 | 2 | 5 | Yes | Asian | M | NaN | NaN | CA |
| **2** | 3 | Pennsylvania supermarket shooting | Tunkhannock, PA | 6/7/2017 | Randy Stair, a 24-year-old worker at Weis groc... | 3 | 0 | 3 | Unclear | White | M | NaN | NaN | PA |

**making abbreviations to non abbreviated state names**[**¶**](#3znysh7)

In [4]:

state\_df = pd.read\_csv('states.csv', header = None)  
state\_df[:10]  
stateList = state\_df[0].tolist()  
stateList2 = state\_df[1].tolist()  
stateList3 = []  
**for** i **in** range (len(stateList2)):  
 stateList3.append([stateList[i], stateList2[i]])  
   
**def** mod\_state(f):  
 *#print f*  
 *#print len(stateList3)*  
 **for** i **in** range(len(stateList3)):  
 *#print repr(str(f)), repr(str(stateList3[i][1]))*  
 **if** f.lstrip() == stateList3[i][1]:  
 **return** stateList3[i][0]  
 **elif** f.lstrip() == stateList3[i][0]:  
 **return** stateList3[i][0]  
   
df['Mod\_state'] = df['State'].apply(mod\_state)  
df[:3]

Out[4]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **S#** | **Title** | **Location** | **Date** | **Summary** | **Fatalities** | **Injured** | **Total victims** | **Mental Health Issues** | **Race** | **Gender** | **Latitude** | **Longitude** | **State** | **Mod\_state** |
| **0** | 1 | Las Vegas Strip mass shooting | Las Vegas, NV | 10/1/2017 | NaN | 58 | 515 | 573 | Unclear | NaN | NaN | NaN | NaN | NV | Nevada |
| **1** | 2 | San Francisco UPS shooting | San Francisco, CA | 6/14/2017 | Jimmy Lam, 38, fatally shot three coworkers an... | 3 | 2 | 5 | Yes | Asian | M | NaN | NaN | CA | California |
| **2** | 3 | Pennsylvania supermarket shooting | Tunkhannock, PA | 6/7/2017 | Randy Stair, a 24-year-old worker at Weis groc... | 3 | 0 | 3 | Unclear | White | M | NaN | NaN | PA | Pennsylvania |

**States and Total victims of shootings**[**¶**](#2et92p0)

In [5]:

*#def makeInt (f):*  
 *#print (str(f))*  
 *#return int(f.strip())*   
  
*#df['Total victims'] = df['Total victims'].apply(makeInt)*  
state = pd.pivot\_table(df,   
 index='Mod\_state',  
 columns=None,  
 values='Total victims',  
 aggfunc='sum')  
state

Out[5]:

|  |  |
| --- | --- |
|  | **Total victims** |
| **Mod\_state** |  |
| **Alabama** | 51 |
| **Alaska** | 4 |
| **Arizona** | 84 |
| **Arkansas** | 33 |
| **California** | 535 |
| **Colorado** | 275 |
| **Connecticut** | 90 |
| **Florida** | 317 |
| **Georgia** | 107 |
| **Hawaii** | 14 |
| **Idaho** | 4 |
| **Illinois** | 106 |
| **Indiana** | 5 |
| **Iowa** | 13 |
| **Kansas** | 43 |
| **Kentucky** | 70 |
| **Louisiana** | 53 |
| **Maine** | 3 |
| **Massachusetts** | 27 |
| **Michigan** | 54 |
| **Minnesota** | 54 |
| **Mississippi** | 47 |
| **Missouri** | 27 |
| **Montana** | 8 |
| **Nebraska** | 38 |
| **Nevada** | 611 |
| **New Jersey** | 12 |
| **New Mexico** | 7 |
| **New York** | 146 |
| **North Carolina** | 71 |
| **Ohio** | 65 |
| **Oklahoma** | 57 |
| **Oregon** | 105 |
| **Pennsylvania** | 41 |
| **South Carolina** | 55 |
| **South Dakota** | 5 |
| **Tennessee** | 29 |
| **Texas** | 373 |
| **Utah** | 22 |
| **Vermont** | 4 |
| **Virginia** | 125 |
| **Washington** | 155 |
| **West Virginia** | 4 |
| **Wisconsin** | 87 |
| **Wyoming** | 4 |

**List of countries by firearm-related death rate, Wikipedia**[**¶**](#tyjcwt)

**Data Cleaning With python and apply functions**[**¶**](#3dy6vkm)

In [6]:

df\_all = pd.read\_csv('firearms\_country.csv')  
  
**def** gun\_per\_100(f):   
 **return** f.split(" ")[0]  
**def** gun\_per\_100\_2(f):   
 **return** f.split("-")[0]  
**def** gun\_per\_100\_3(f):   
 f\_list = list(str(f))  
 **for** i **in** range (len(f\_list)):  
 **if** f\_list[i] **not** **in** ['1','2','3','4','5','6','7','8','9','0','.','?']:  
 f\_list[i] = ' '  
 **if** f\_list[0] == '?':  
 **return** np.nan  
 joinedtogether = "".join(f\_list)  
 **return** float(joinedtogether.split(" ")[0])  
**def** country\_func(f):  
 f\_list = list(str(f))  
 **for** i **in** range (len(f\_list)):  
 **if** **not** f\_list[i].isalpha():  
 f\_list[i] = ''  
 joinedtogether = "".join(f\_list)  
 **return** joinedtogether  
  
df\_all['guns\_per\_100'] = df\_all["Guns per 100 inhabitants"].apply(gun\_per\_100).apply(gun\_per\_100\_2).apply(gun\_per\_100\_3)  
df\_all["Country"] = df\_all["Country"].apply(country\_func)  
df\_all["Total"] = df\_all["Total"].apply(gun\_per\_100).apply(gun\_per\_100\_2).apply(gun\_per\_100\_3)  
df\_all[:5]

Out[6]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Country** | **Year** | **Total** | **Homicides** | **Suicides** | **Unintentional** | **Undetermined** | **Guns per 100 inhabitants** | **guns\_per\_100** |
| **0** | Argentina | 2014 | 6.93 | 3.07 | 1.2 | 0.08 | 2.57 | 8.8 (2010) | 8.8 |
| **1** | Australia | 2016 | 1.04 | 0.18 | 0.8 | 0.02 | 0.01 | 13.7 | 13.7 |
| **2** | Austria | 2014 | 2.90 | 0.12 | 2.69 | 0.02 | 0.09 | 30.4 (2007?) | 30.4 |
| **3** | Azerbaijan | 2007 | 0.07 | 0.27 (2010) | 0.01 | 0.02 | ? | 3.5 | 3.5 |
| **4** | Barbados | 2011 | 6.60 | 3.12 (2013) | ? | 0.35 | 4.88 | 3.5 (2007) | 3.5 |

**Regression Total Gun vs Guns per 100 inhabitants**[**¶**](#1t3h5sf)

In [7]:

y, X = dmatrices('Total ~ guns\_per\_100', data=df\_all, return\_type='dataframe')  
  
model = sm.OLS(y, X) *# Set up the model*  
result = model.fit() *# Fit model (find the intercept and slopes)*  
**print** result.summary()

OLS Regression Results   
==============================================================================  
Dep. Variable: Total R-squared: 0.001  
Model: OLS Adj. R-squared: -0.014  
Method: Least Squares F-statistic: 0.03828  
Date: Mon, 23 Apr 2018 Prob (F-statistic): 0.845  
Time: 15:10:26 Log-Likelihood: -273.81  
No. Observations: 69 AIC: 551.6  
Df Residuals: 67 BIC: 556.1  
Df Model: 1   
Covariance Type: nonrobust   
================================================================================  
 coef std err t P>|t| [0.025 0.975]  
--------------------------------------------------------------------------------  
Intercept 7.2671 2.210 3.289 0.002 2.856 11.678  
guns\_per\_100 -0.0212 0.108 -0.196 0.845 -0.238 0.195  
==============================================================================  
Omnibus: 59.924 Durbin-Watson: 1.893  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 245.661  
Skew: 2.786 Prob(JB): 4.52e-54  
Kurtosis: 10.376 Cond. No. 28.9  
==============================================================================  
  
Warnings:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

**Plot of regression**[**¶**](#4d34og8)

In [8]:

x = df\_all['guns\_per\_100']  
y\_actual = df\_all['Total']  
y\_expected = result.params['Intercept'] + result.params['guns\_per\_100'] \* x  
  
scatter(x=x, y=y\_actual, label='Total versus guns\_per\_100')  
plot(x, y\_expected, color='red', linewidth=2, label='Regression')  
xlabel('guns\_per\_100')  
ylabel('Total')  
legend(loc='best')

Out[8]:

<matplotlib.legend.Legend at 0xff3bac8>

**First World Countries Data Mask**[**¶**](#2s8eyo1)

In [9]:

*#countries\_list = "Belgium, Canada, Denmark, France, Germany, Greece, Iceland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Turkey, United Kingdom, United States, Israel, Japan, South Korea, Taiwan, Australia, New Zealand, Austria, Finland, Ireland, Sweden, Switzerland, Yugoslavia, Brazil, Russia, India, China"*  
countries\_list = "Belgium, Canada, Denmark, France, Germany, Greece, Iceland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Turkey, United Kingdom, United States, Israel, Japan, South Korea, Taiwan, Australia, New Zealand, Austria, Finland, Ireland, Sweden, Switzerland, Yugoslavia"  
  
countries\_list = countries\_list.replace(' ', '')  
countries\_list = countries\_list.split(',')  
countries\_list  
  
mask = df\_all['Country'].isin (countries\_list)  
df\_all\_masked = df\_all[mask]  
df\_all\_masked[:5]

Out[9]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Country** | **Year** | **Total** | **Homicides** | **Suicides** | **Unintentional** | **Undetermined** | **Guns per 100 inhabitants** | **guns\_per\_100** |
| **1** | Australia | 2016 | 1.04 | 0.18 | 0.8 | 0.02 | 0.01 | 13.7 | 13.70 |
| **2** | Austria | 2014 | 2.90 | 0.12 | 2.69 | 0.02 | 0.09 | 30.4 (2007?) | 30.40 |
| **6** | Belgium | 2013 | 1.24 | 0.14 | 1.03 | 0.01 | 0.05 | 6.86 (2015) | 6.86 |
| **10** | Canada | 2011 | 2.05 | 0.45 (2014) | 1.52 | 0.05 | 0.02 | 25.33 | 25.33 |
| **18** | Denmark | 2011 | 1.28 | 0.22 | 1.09 | 0.04 (2008) | 0.02 (2010) | 12 | 12.00 |

**Regression Total Gun vs Guns per 100 inhabitants, First World Countries**[**¶**](#17dp8vu)

In [10]:

y, X = dmatrices('Total ~ guns\_per\_100', data=df\_all\_masked, return\_type='dataframe')  
  
model = sm.OLS(y, X) *# Set up the model*  
result = model.fit() *# Fit model (find the intercept and slopes)*  
**print** result.summary()

OLS Regression Results   
==============================================================================  
Dep. Variable: Total R-squared: 0.763  
Model: OLS Adj. R-squared: 0.752  
Method: Least Squares F-statistic: 73.94  
Date: Mon, 23 Apr 2018 Prob (F-statistic): 1.22e-08  
Time: 15:10:26 Log-Likelihood: -37.876  
No. Observations: 25 AIC: 79.75  
Df Residuals: 23 BIC: 82.19  
Df Model: 1   
Covariance Type: nonrobust   
================================================================================  
 coef std err t P>|t| [0.025 0.975]  
--------------------------------------------------------------------------------  
Intercept -0.1978 0.327 -0.606 0.551 -0.873 0.478  
guns\_per\_100 0.1003 0.012 8.599 0.000 0.076 0.124  
==============================================================================  
Omnibus: 2.561 Durbin-Watson: 1.837  
Prob(Omnibus): 0.278 Jarque-Bera (JB): 1.529  
Skew: -0.600 Prob(JB): 0.466  
Kurtosis: 3.170 Cond. No. 39.9  
==============================================================================  
  
Warnings:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

**Plot of regression, First World Countries**[**¶**](#3rdcrjn)

In [11]:

x = df\_all\_masked['guns\_per\_100']  
y\_actual = df\_all\_masked['Total']  
y\_expected = result.params['Intercept'] + result.params['guns\_per\_100'] \* x  
  
scatter(x=x, y=y\_actual, label='Total versus guns\_per\_100')  
plot(x, y\_expected, color='red', linewidth=2, label='Regression')  
xlabel('guns\_per\_100')  
ylabel('Total')  
legend(loc='best')

Out[11]:

<matplotlib.legend.Legend at 0x10252a20>