EE 379K: Data Science Lab

Lab 2 - 9/18/17

Rachel Chen and Kevin Yee

rjc2737 and kjy252

1. Correlations

(a)

When given a data matrix, an easy way to tell if any two columns are correlated is to look at a scatter plot of each column against each other column. For a warm up, do this: Look at the data in DF1 in Lab2.zip. Which columns are (pairwise) correlated? Figure out how to do this with Pandas, and also how to do this with Seaborn.

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

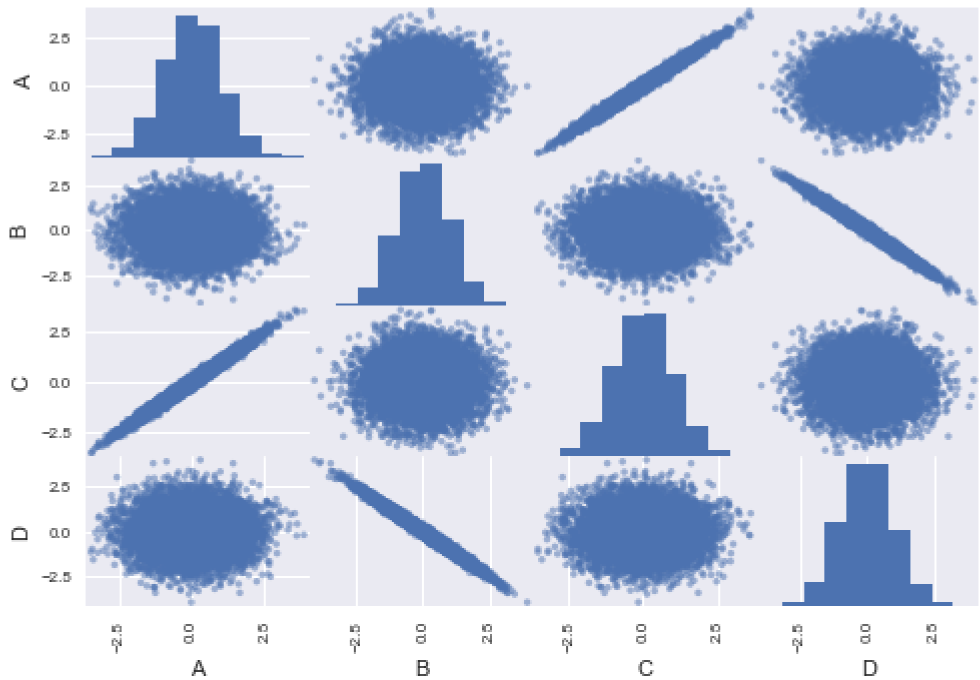
In [2]:

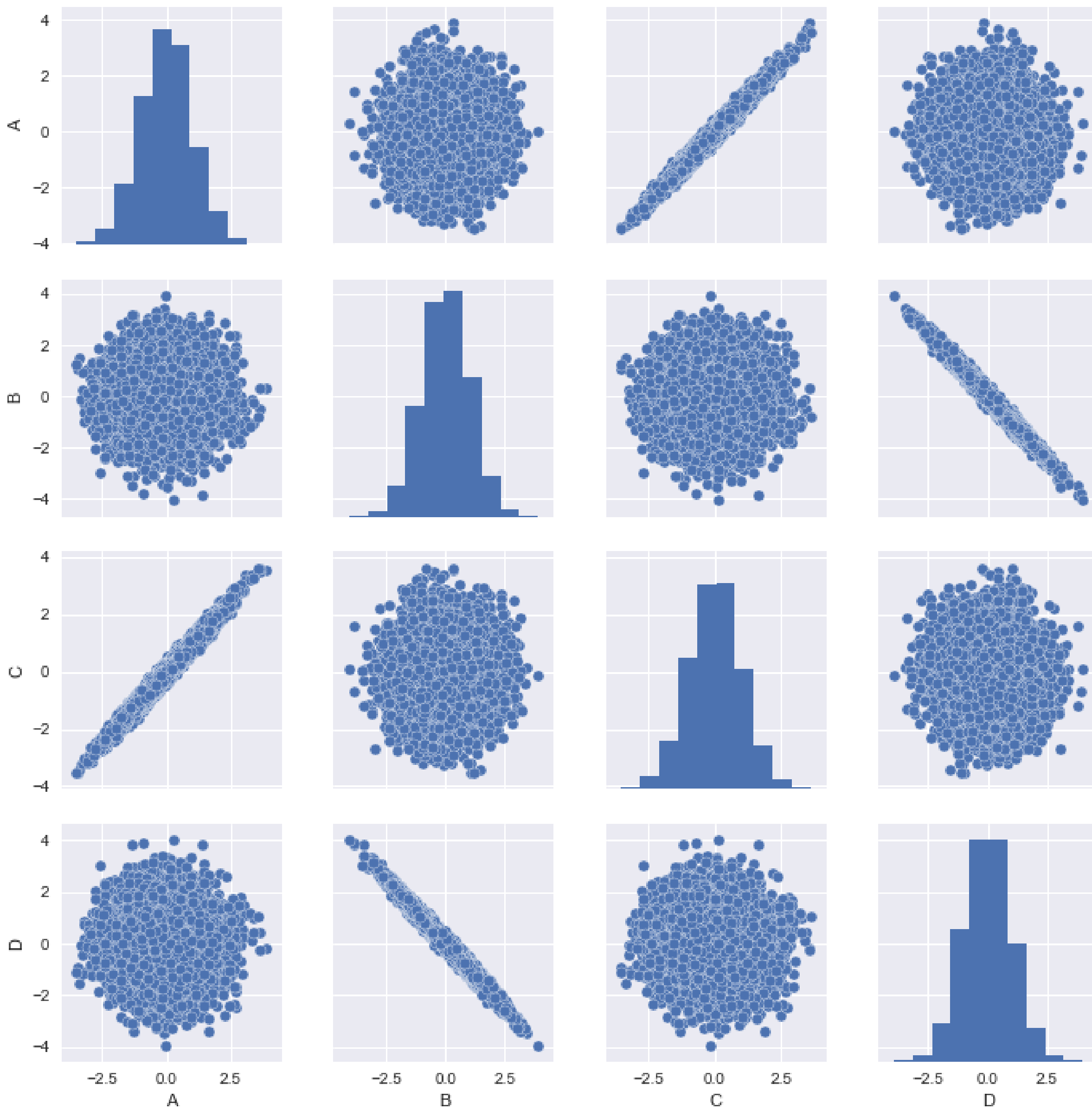
```
#Read csv into pandas
colNames = pd.Series(['A',' B','C', 'D'])
df = pd.DataFrame.from_csv("Lab2_Data/DF1")
df.columns = colNames;

#verify data is correct
df.head()
```

Out[2]:

	A	B	C	D
0	1.038502	0.899865	0.835053	-0.971528
1	0.320455	-0.647459	0.149079	0.352593
2	0.055480	2.234771	0.271672	-2.108739
3	-0.007260	-0.524299	-0.126550	0.670827
4	-1.237390	-1.377017	-1.049932	1.342079

In [3]:
```
scatter_matrix(df);

sns.pairplot(df);
sns.plt.show();
```


A
B
C
D
2.5
0.0
-2.5



(b)

Compute the covariance matrix of the data. Write the explicit expression for what this is, and then use any command you like (e.g., np.cov) to compute the 4 □ 4 matrix. Explain why the numbers that you get t with the plots you got.

In [84]:
```
df.cov()
```

Out[84]:

	A	B	C	D
A	1.001558	-0.004012	0.991624	0.004125
B	-0.004012	1.005378	-0.004099	-0.995457
C	0.991624	-0.004099	1.001589	0.004081
D	0.004125	-0.995457	0.004081	1.005168

$$\sum = \begin{bmatrix} \phi_1^2 & \phi_{12} & \phi_{13} & \phi_{14} \\ \phi_{21} & \phi_2^2 & \phi_{23} & \phi_{24} \\ \phi_{31} & \phi_{32} & \phi_3^2 & \phi_{34} \\ \phi_{41} & \phi_{42} & \phi_{43} & \phi_4^2 \end{bmatrix}$$

The covariance matrix is defined above, where the covariance between

$$Cov_{ij}$$

designates the correlation between the two columns.

In the dataset provided by DF1, we can see columns C and A have a positive correlation of 0.99 and columns B and D have a negative correlation of -0.99. This indicates that C and A and B and D are closely related

(c)

The above problem in reverse. Generate a zero-mean multivariate Gaussian random variable in 3 dimensions,

$$Z = (X1; X2; X3)$$

so that

$$(X1; X2)$$

and

$$(X1; X3)$$

are uncorre- lated, but

$$(X2; X3)$$

are correlated.

Specically: choose a covariance matrix that has the above correlations structure, and write this down. Then nd a way to generate samples from this Gaussian.

Choose one of the non-zero covariance terms (Cij , if C denotes your covariance matrix) and plot it vs the estimated covariance term, as the number of samples you use scales.

The goal is to get a visual representation of how the empirical covariance converges to the true (or family) covariance.

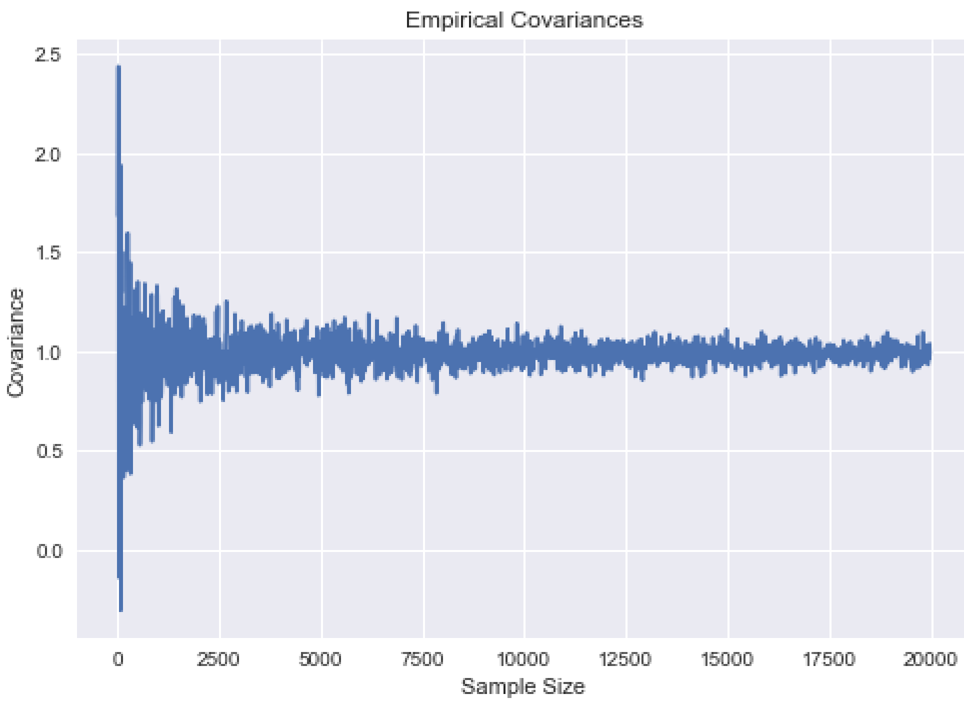
We can define the covariance matrix as follows:

$$\sum = \begin{bmatrix} \phi_1^2 & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_2^2 & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_3^2 \end{bmatrix}$$

In [85]:
```
#Generate zero mean multivariate
#X2 and X3 are correlated
mean = [0, 0, 0]
cov = [[5, 0, 0],
        [0, 5, 0.99],
        [0, 0.99, 5]]

#Generate random samples from this gaussian
rv = np.random.multivariate_normal(mean,cov,100000)
df = pd.DataFrame(rv, columns = ['X1', 'X2', 'X3'])


samples = []
covariances = []
#Plot vs estimated covariance term, as the number of samples you use scale
for numsamples in range(0, 20000, 10):
    rv = np.random.multivariate_normal(mean,cov,numsamples)
    covar23 = pd.DataFrame(rv, columns = ['X1', 'X2', 'X3']).cov().loc['X2']
['X3']

    samples.append(numsamples)
    covariances.append(covar23)



plt.title('Empirical Covariances')
plt.xlabel('Sample Size')
plt.ylabel('Covariance')
plt.plot(samples,covariances)
```

```
C:\Users\kevjy\Anaconda\lib\site-packages\numpy\lib\function_base.py:1110: Ru
ntimeWarning: Mean of empty slice.
  avg = a.mean(axis)
C:\Users\kevjy\Anaconda\lib\site-packages\pandas\core\frame.py:5000: RuntimeW
arning: Degrees of freedom <= 0 for slice
  baseCov = np.cov(mat.T)
```

Out[85]: [<matplotlib.lines.Line2D at 0x22b66a58>]



The above empirical covariance converges as n approaches infinity.

2. Outliers.

Consider the two-dimensional data in DF2 in Lab2.zip. Look at a scatter plot of the data. It contains two points that look like potential outliers. Which one is “more” outlying? Propose a transformation of the data that makes it clear that the point at (−1, 1) is more outlying than the point at (5.5, 5), even though the latter point is“farther away” from the nearest points. Plot the data again after performing this transformation. Provide discussion as appropriate to justify your choice of transformation. Hint: if y comes from a standard Gaussian in two dimensions (i.e., with covariance equal to the two by two identity matrix), and

$$Q = \begin{bmatrix} 2 & \frac{1}{2} \\ \frac{1}{2} & 2 \end{bmatrix}$$

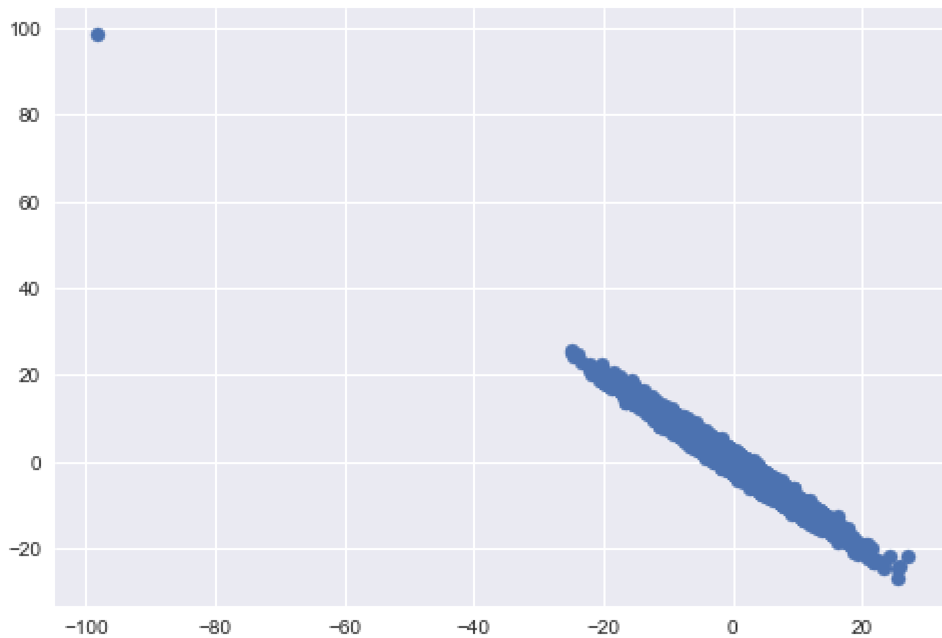
what is the covariance matrix of the random variable z = Qy? If you are given z, how would you create a random Gaussian vector with covariance equal to the identity, using z?

In [86]:
```
#Read from CSV file
df = pd.DataFrame.from_csv("Lab2_Data/DF2")

#plot of untransformed data
plt.scatter(df['0'],df['1'])
plt.title("Outlier Point")
```

Out[86]: <matplotlib.text.Text at 0x18b05b38>

In [87]:
```
#covariance
cov = df.cov()

#inverse covariance
cov_inv = pd.DataFrame(np.linalg.pinv(cov.values))

# Find Y = A^-1 * X where Y has the the identity matrix
# Where Cov = A * A^T

# Transform the data and plot again
data = pd.DataFrame(np.dot(df,cov_inv))

colnames = ['A','B']
data.columns = colnames

plt.scatter(data['A'],data['B'])
```

Out[87]: <matplotlib.collections.PathCollection at 0x13287278>



From the graph above, we can see clearly that (-1, 1) is the clear outlier.

The Mahalanobis distance was used to scale distances so that distances along a direction where the dataset is very spread out are scaled down, and distances along directions where the dataset is tighly packed are scaled up.

As a result, a value of much higher variance will appear farther away with this transformation

Why the above statement works?

dataset.

The Mahalanobis distance uses covariance to scale distances so that distances along a direction where the dataset is very spread out are scaled down, and distances along directions where the dataset is tightly packed are scaled up. For example, in Figure 5.15(b)[221] the Mahalanobis distance between B and A will be less than the Mahalanobis distance between C and A, whereas in Figure 5.15(c)[221] the opposite will be true. The Mahalanobis distance is defined as

$Mahalanobis(\mathbf{a}, \mathbf{b}) =$

(5.16)

Let's step through Equation (5.16)[222] bit by bit. First, this equation computes a distance between two instances **a** and **b**, each with m descriptive features. The first big term we come to in the equation is $[\mathbf{a}[1] - \mathbf{b}[1], \ldots, \mathbf{a}[m] - \mathbf{b}[m]]$. This is a row vector that is created by subtracting each descriptive feature value of instance **b** from the corresponding feature values of **a**. The next term in the equation, $\sum^{-1}$, represents the **inverse covariance matrix**[22] computed across all instances in the dataset. Multiplying the difference in feature values by the inverse covariance matrix has two effects. First, the larger the **variance** of a feature, the less weight the difference between the values for that feature will contribute to the distance calculation. Second, the larger the correlation between two features, the less weight they contribute to the distance. The final

22 We explain **covariance matrices** in Section 3.5.2[86]. The **inverse covariance matrix** is the matrix such that when the covariance matrix is multiplied by its inverse, the result is the **identity matrix**: $\sum \times \sum^{-1} = \mathbb{I}$. The identity matrix is a square matrix in which all the elements of the main diagonal are 1, and all other elements are 0. Multiplying any matrix by the identity matrix leaves the original matrix unchanged—this is the equivalent of multiplying by 1 for real numbers. So the effect of multiplying feature values be an inverse covariance matrix is to rescale the variances of all features to 1 and to set the covariance between all feature pairs to 0. Calculating the inverse of a matrix involves solving systems of linear equations and requires the use of techniques from linear algebra such as **Gauss-Jordan elimination** or **LU decomposition**. We do not cover these techniques here, but they are covered in most standard linear algebra textbooks such as Anton and Rorres (2010).

3. Even More Standard Error

3. **Even More Standard Error** (This is to be completed only after you've completed the last written exercise below). In one of the written exercises below, you derive an expression

for what is called the *Standard Error*: where β denotes the "truth," $\hat{\beta}$ denotes the value we compute using least squares linear regression, and Z and e are as in the exercise below, you find:

$$\hat{\beta} - \beta = Ze.$$

If we know the distribution of the noise (the distribution generating the noise vectors, e_i), then we know the distribution for the error, $(\hat{\beta} - \beta)$. This allows us to answer the question given in class: if we solve a regression and obtain value $\hat{\beta}$, how can we tell if it is statistically significant? The answer is: we compare the size of $\hat{\beta}$ to the spread introduced by the noise (i.e., the standard error), and we ask: what is the likelihood that the true $\beta = 0$, and what we observed was purely due to the noise.

If the noise is Gaussian (normal), i.e., $e_i \sim N(0, \sigma^2)$, and if the values of the x_i are normalized, then we expect error of the size $\sigma/\sqrt{n}$, as this is roughly the standard deviation of the expression for the error that you derive above. This means: if you have twice the data points, you should expect the error to be reduced by about 1.4 (the formula says that the standard deviation of the error would decrease by a factor of $1/\sqrt{2}$).

Compute this empirically, as follows: We will generate data for a regression problem, solve it, and see what the error is: Generate data as I did in the example from class: $x_i \sim N(0,1)$, $e_i \sim N(0,1)$. Generate y by $y_i = \beta_0 + x_i\beta + e_i$, where $\beta_0 = -3$ and $\beta = 0$. *Note that since $\beta = 0$, this means that y and x are unrelated! The question we are exploring here is as follows: when we solve a regression problem, we are not going to find $\hat{\beta} = 0$ – we will find that $\hat{\beta}$ takes some other values, hopefully close to zero. How do we know if the value of $\hat{\beta}$ we get is statistically meaningful?*

- By creating fresh data and each time computing $\hat{\beta}$ and recording $\hat{\beta} - \beta$, compute the *empirical standard deviation* of the error for $n = 150$ (the number we used in class). In class, in the exercise where I tried to find a linear regression of y vs. noise, we found $\hat{\beta} = -0.15$. Given your empirical computation of the standard deviation of the error, how significant is the value -0.15?
- Now repeat the above experiment for different values of n. Plot these values, and on the same plot, plot $1/\sqrt{n}$. How is the fit?

Answer

Part 1

$$y_i = \beta_0 + x_i\beta + e_i$$

where $\beta_0 = -3$ and $\beta = 0$ $X_i = N(0,1)$

In [88]:
```
def stdError(n):
    errors = []
    for i in range(1000):
        #defining the constants
        betanot = -3
        beta = 0
        Xnorm = np.random.normal(0, 1, n)
        Enoise = np.random.normal(0,1, n)

        #plugging into the formula
        Y = betanot + beta*Xnorm + Enoise

        #Formula derived in written section
        betahat = np.dot(Xnorm, Y)/np.dot(Xnorm,Xnorm)

        #calculate error
        error = betahat - beta
        errors.append(error)
    return np.std(errors)
```

In [89]:
```
stdError(150)
```

Out[89]: 0.26638038764360272

Given that n = 150, the standard deviation is ~0.244 and the computed value $\hat{\beta}$ is ~0.-15.

We recognize that the calcualted value of $\hat{\beta}$ is within one standard deviation of 0.

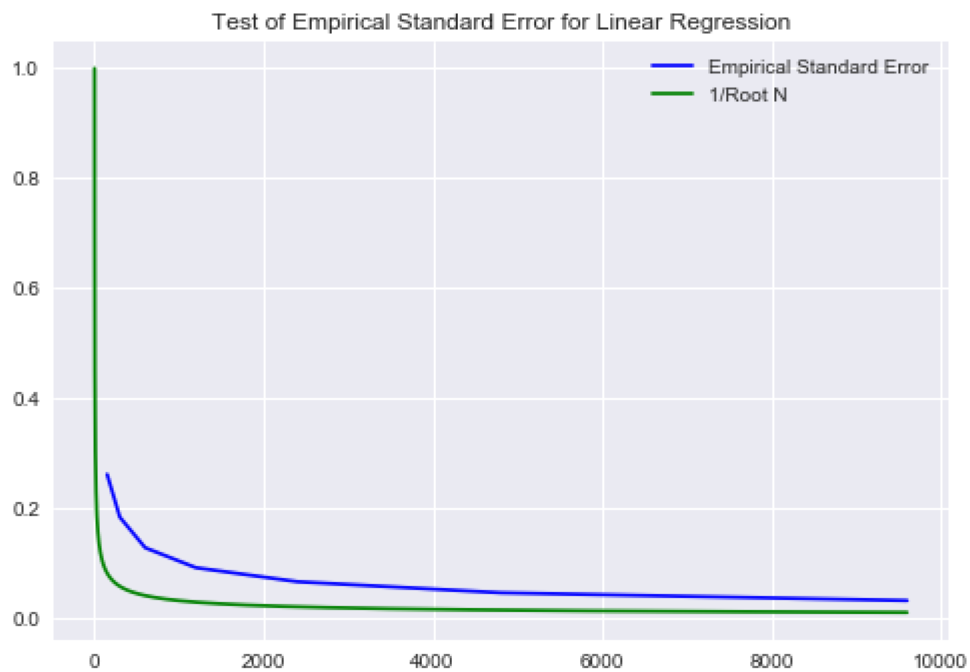
$$-0.266 < \hat{\beta} < 0.266$$

the above expression indicates a good estimate. As a result, -0.15 is not significant given the standard deviation

Part 2

In [90]:
```
#Gather data for Empirical Covariances
yErrors = []
numSamples = [150, 300, 600, 1200, 2400, 4800, 9600]
for i in range(len(numSamples)):
    yErrors.append(stdError(numSamples[i]))

#Gather data for 1/root(i)
xSqrt = []
ySqrt = []
for i in range(1,9600):
    xSqrt.append(i)
    ySqrt.append(1/np.sqrt(i))

#plot
plt.plot(numSamples, yErrors, color = 'b', label = 'Empirical Standard Error')
plt.plot(xSqrt, ySqrt, color = 'g', label = '1/Root N')
plt.legend()
plt.title('Test of Empirical Standard Error for Linear Regression')
```

Out[90]: <matplotlib.text.Text at 0x10cea278>



4. Names and Frequencies

The goal of this exercise is for you to get more experience with Pandas, and to get a chance to explore a cool data set. Download the file Names.zip from Canvas. This contains the frequency of all names that appeared more than 5 times on a social security application from 1880 through 2015.

- Write a program that on input k and XXXX, returns the top k names from year XXXX.
- Write a program that on input Name returns the frequency for men and women of the name Name.
- It could be that names are more diverse now than they were in 1880, so that a name may be relatively the most popular, though its frequency may have been decreasing over the years. Modify the above to return the relative frequency.
- Find all the names that used to be more popular for one gender, but then became more popular for another gender.
- (Optional) Find something cool about this data set.

In [8]:
```
k = int(input("Enter the number of top names: "))
year = int(input("Enter the year would like to search: "))
while(year < 1880 or year > 2015):
    year = int(input("Please enter another year between 1880 and 2015"))
df = pd.read_csv("Names/yob"+str(year)+".txt", sep=',', names=['name', 'gende
r', 'freq'])

#print
print(df.sort_values('freq', ascending=False).head(k))
```

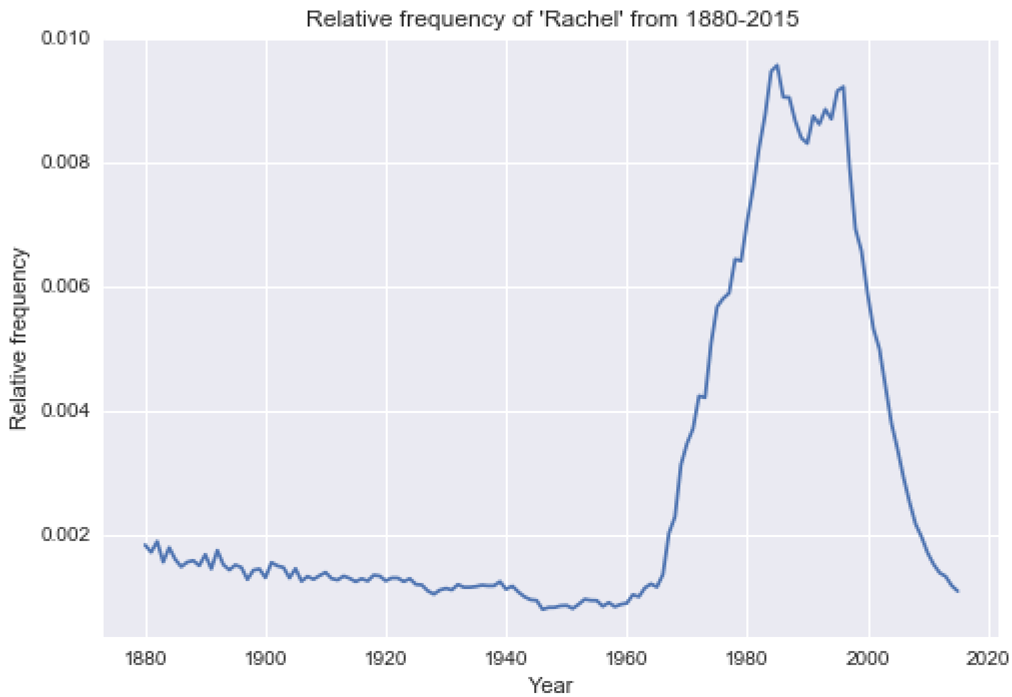
```
Enter the number of top names: 5
Enter the year would like to search: 1999
          name gender   freq
16938    Jacob      M  35346
16939  Michael      M  33906
16940  Matthew      M  30417
16941   Joshua      M  27254
0        Emily      F  26537
```

In [12]:
```
name = raw_input("Enter the name: ")
filenames = ['Names/yob%s.txt' % year for year in range(1880, 2016)]
dfs = []
for filename in filenames:
    dfs.append(pd.read_csv(filename, sep=',', names=['name', 'gender',
'freq']))
name_count_female = 0
name_count_male = 0

for df in dfs:
    df_name = df.loc[df['name'] == name]
    females = df_name.loc[df_name['gender'] == 'F'].freq
    males = df_name.loc[df_name['gender'] == 'M'].freq
    if len(females) > 0:
        name_count_female += females.values[0]
    if len(males) > 0:
        name_count_male += males.values[0]
print 'Female: %d, Male: %d' % (name_count_female, name_count_male)
```

```
Enter the name: James
Female: 23215, Male: 5120990
```

In [8]:
```
name =  raw_input("Enter a name: ")
filenames = ['Names/yob%s.txt' % year for year in range(1880, 2016)]
dfs = []
for filename in filenames:
    dfs.append(pd.read_csv(filename, sep=',', names=['name', 'gender',
'freq']))
df_freq = pd.DataFrame()
year = 1880
name_count_female = 0
name_count_male = 0
for df in dfs:
    df_name = df.loc[df['name'] == name]
    females = df_name.loc[df_name['gender'] == 'F'].freq
    males = df_name.loc[df_name['gender'] == 'M'].freq
    if len(females) > 0:
        name_count_female = females.values[0]
    if len(males) > 0:
        name_count_male = males.values[0]
    #get specific name's male and female count

    df_female = df.loc[df['gender'] == 'F'].sum()
    df_male = df.loc[df['gender'] == 'M'].sum()
    total_count_female = df_female[2]
    total_count_male = df_male[2]
    #get total male and female count

    rel_freq_female = name_count_female / float(total_count_female)
    rel_freq_male = name_count_male / float(total_count_male)
    #calculate relative freq of name

    df_name_freq = pd.DataFrame({'Year' : year,'F_freq' : [rel_freq_female],
'M_freq' : [rel_freq_male]})
    df_freq = df_freq.append(df_name_freq)
    year += 1

    #reset
    name_count_female = 0
    name_count_male =0

plt.title("Relative frequency of '"+ name + "' from 1880-2015")
plt.xlabel("Year")
plt.ylabel("Relative frequency")
plt.plot(df_freq.Year, (df_freq.M_freq + df_freq.F_freq))
plt.show()


print(df_freq)
```

Enter a name: Rachel



```
      F_freq    M_freq  Year
0   0.001824  0.000000  1880
0   0.001707  0.000000  1881
0   0.001882  0.000000  1882
0   0.001549  0.000000  1883
0   0.001783  0.000000  1884
0   0.001593  0.000000  1885
0   0.001474  0.000000  1886
0   0.001548  0.000000  1887
0   0.001573  0.000000  1888
0   0.001491  0.000000  1889
0   0.001670  0.000000  1890
0   0.001439  0.000000  1891
0   0.001738  0.000000  1892
0   0.001508  0.000000  1893
0   0.001422  0.000000  1894
0   0.001502  0.000000  1895
0   0.001463  0.000000  1896
0   0.001268  0.000000  1897
0   0.001418  0.000000  1898
0   0.001390  0.000047  1899
0   0.001264  0.000033  1900
0   0.001542  0.000000  1901
0   0.001488  0.000000  1902
0   0.001462  0.000000  1903
0   0.001293  0.000000  1904
0   0.001444  0.000000  1905
0   0.001239  0.000000  1906
0   0.001318  0.000000  1907
0   0.001271  0.000000  1908
0   0.001331  0.000000  1909
..       ...       ...   ...
0   0.009010  0.000031  1986
0   0.009000  0.000034  1987
0   0.008620  0.000032  1988
0   0.008331  0.000053  1989
0   0.008275  0.000019  1990
0   0.008718  0.000015  1991
0   0.008589  0.000013  1992
0   0.008829  0.000013  1993
0   0.008674  0.000013  1994
0   0.009125  0.000015  1995
0   0.009193  0.000013  1996
0   0.007921  0.000012  1997
0   0.006905  0.000009  1998
0   0.006554  0.000009  1999
0   0.005878  0.000006  2000
0   0.005297  0.000004  2001
0   0.004969  0.000008  2002
0   0.004369  0.000007  2003
0   0.003767  0.000017  2004
0   0.003364  0.000007  2005
0   0.002909  0.000003  2006
0   0.002511  0.000003  2007
0   0.002160  0.000002  2008
0   0.001952  0.000003  2009
0   0.001698  0.000000  2010
```

```
0   0.001512  0.000000  2011
0   0.001372  0.000004  2012
0   0.001318  0.000000  2013
0   0.001168  0.000003  2014
0   0.001080  0.000000  2015

[136 rows x 3 columns]
```

In [2]:
```
filenames = ['Names/yob%s.txt' % year for year in range(1880, 2016)]
dfs = []
initialyear = 1880
for filename in filenames:
    currentDF = pd.read_csv(filename, sep=',', names=['name', 'gender',
'freq'])
    currentDF['Year'] = pd.Series(initialyear, index=currentDF.index)
    dfs.append(currentDF)
    initialyear += 1
bigDF = pd.concat(dfs)

#find all values that appear more than the 136 (expected number names if it wa
s just one gender)
dfpiv = pd.pivot_table(bigDF,index=['name'],aggfunc='count')
multiNamePiv = dfpiv[dfpiv['freq'] > (2016-1880)]
print multiNamePiv.index
```

```
Index([u'Aaron', u'Abbie', u'Abby', u'Abel', u'Abigail', u'Abraham', u'Ada',
       u'Adair', u'Adam', u'Addie',
       ...
       u'Yancy', u'Yolanda', u'Young', u'Yvette', u'Yvonne', u'Zachary',
       u'Zane', u'Zelma', u'Zoe', u'Zola'],
      dtype='object', name=u'name', length=1672)
```

In [76]:
```
#WARNING: Long Running Code
femaleToMale = [];
maleToFemale = [];
for name in multiNamePiv.index:
    male1948 = bigDF[(bigDF['name'] == name) & (bigDF['Year'] < 1948) &
(bigDF['gender'] == 'M')].freq.sum()
    female1948 = bigDF[(bigDF['name'] == name) & (bigDF['Year'] < 1948) & (big
DF['gender'] == 'F')].freq.sum()

    male2015 = bigDF[(bigDF['name'] == name) & (bigDF['Year'] > 1948) &
(bigDF['gender'] == 'M')].freq.sum()
    female2015 = bigDF[(bigDF['name'] == name) & (bigDF['Year'] > 1948) & (big
DF['gender'] == 'F')].freq.sum()

    if(male1948 > female1948 and male2015 < female2015):
        femaleToMale.append(name)
        print "Male name before 1948 but Female name after " + name
    if(female1948 > male1948 and female2015 < male2015):
        maleToFemale.append(name)
        print "Female name before 1948 but Male name afterwards " + name
```

```
Male name before 1948 but Female name after Allison
Female name before 1948 but Male name afterwards Alpha
Male name before 1948 but Female name after Alva
Male name before 1948 but Female name after Arden
Female name before 1948 but Male name afterwards Artie
Male name before 1948 but Female name after Ashley
Male name before 1948 but Female name after Aubrey
Female name before 1948 but Male name afterwards Audie
Male name before 1948 but Female name after Avery
Male name before 1948 but Female name after Blair
Male name before 1948 but Female name after Carlie
Male name before 1948 but Female name after Charley
Male name before 1948 but Female name after Clair
Male name before 1948 but Female name after Courtney
Male name before 1948 but Female name after Dee
Male name before 1948 but Female name after Elisha
Female name before 1948 but Male name afterwards Frankie
Male name before 1948 but Female name after Gale
Male name before 1948 but Female name after Hollie
Male name before 1948 but Female name after Ivey
Male name before 1948 but Female name after Ivory
Male name before 1948 but Female name after Jackie
Female name before 1948 but Male name afterwards Jessie
Male name before 1948 but Female name after Jodie
Male name before 1948 but Female name after Kelly
Male name before 1948 but Female name after Lacy
Male name before 1948 but Female name after Lavern
Female name before 1948 but Male name afterwards Lavon
Male name before 1948 but Female name after Leigh
Female name before 1948 but Male name afterwards Lennie
Male name before 1948 but Female name after Lesley
Male name before 1948 but Female name after Leslie
Male name before 1948 but Female name after Lindsay
Male name before 1948 but Female name after Lindsey
Female name before 1948 but Male name afterwards Maxie
Male name before 1948 but Female name after Morgan
Female name before 1948 but Male name afterwards Ocie
Female name before 1948 but Male name afterwards Ollie
Male name before 1948 but Female name after Paris
Female name before 1948 but Male name afterwards Pat
Female name before 1948 but Male name afterwards Robbie
Male name before 1948 but Female name after Rosario
Male name before 1948 but Female name after Stacy
Male name before 1948 but Female name after Sydney
Female name before 1948 but Male name afterwards Toby
Male name before 1948 but Female name after Tracy
Female name before 1948 but Male name afterwards Trinidad
```

Male name before 1948 but Female name after Allison Female name before 1948 but Male name afterwards Alpha Male name before 1948 but Female name after Alva Male name before 1948 but Female name after Arden Female name before 1948 but Male name afterwards Artie Male name before 1948 but Female name after Ashley Male name before 1948 but Female name after Aubrey Female name before 1948 but Male name afterwards Audie Male name before 1948 but Female name after Avery Male name before 1948 but Female name after Blair Male name before 1948 but Female name after Carlie Male name before 1948 but Female name after Charley Male name before 1948 but Female name after Clair Male name before 1948 but Female name after Courtney Male name before 1948 but Female name after Dee Male name before 1948 but Female name after Elisha Female name before 1948 but Male name afterwards Frankie Male name before 1948 but Female name after Gale Male name before 1948 but Female name after Hollie Male name before 1948 but Female name after Ivey Male name before 1948 but Female name after Ivory Male name before 1948 but Female name after Jackie Female name before 1948 but Male name afterwards Jessie Male name before 1948 but Female name after Jodie Male name before 1948 but Female name after Kelly Male name before 1948 but Female name after Lacy Male name before 1948 but Female name after Lavern Female name before 1948 but Male name afterwards Lavon Male name before 1948 but Female name after Leigh Female name before 1948 but Male name afterwards Lennie Male name before 1948 but Female name after Lesley Male name before 1948 but Female name after Leslie Male name before 1948 but Female name after Lindsay Male name before 1948 but Female name after Lindsey Female name before 1948 but Male name afterwards Maxie Male name before 1948 but Female name after Morgan Female name before 1948 but Male name afterwards Ocie Female name before 1948 but Male name afterwards Ollie Male name before 1948 but Female name after Paris Female name before 1948 but Male name afterwards Pat Female name before 1948 but Male name afterwards Robbie Male name before 1948 but Female name after Rosario Male name before 1948 but Female name after Stacy Male name before 1948 but Female name after Sydney Female name before 1948 but Male name afterwards Toby Male name before 1948 but Female name after Tracy Female name before 1948 but Male name afterwards Trinidad

In [78]:
```
#this code works, but take's forever to analyze. commented out
'''
runningCountMales = 0
runningCountFemales = 0

moreFemaleThanMale =False;
swapHappened = False;

femaleToMaleList = []
maleToFemaleList = []
for name in multiNamePiv.index:
    for year in range(1880,2016):
        df_name = bigDF.loc[(bigDF['name'] == name) & (bigDF['Year'] == year)]
        females = df_name.loc[df_name['gender'] == 'F'].freq
        males = df_name.loc[df_name['gender'] == 'M'].freq
    if (len(females) > 0 and len(males) > 0):
        if(females.values[0] > males.values[0]):
            moreFemaleThanMale = True
        else:
            moreFemaleThanMale = False
        if(moreFemaleThanMale and not swapHappened and males.values[0] > femal
es.values[0]):
            swapHappened = True;
            femaleToMaleList.append(name);
        if(not moreFemaleThanMale and not swapHappened and females.values[0] >
 males.values[0]):
            swapHapened = True;
            maleToFemaleList.append(name);
print femaleToMaleList
'''
```

Out[78]:
```
"\nrunningCountMales = 0\nrunningCountFemales = 0\n\nmoreFemaleThanMale =Fals
e;\nswapHappened = False;\n\nfemaleToMaleList = []\nmaleToFemaleList = []\nfo
r name in multiNamePiv.index:\n    for year in range(1880,2016):\n        df_
name = bigDF.loc[(bigDF['name'] == name) & (bigDF['Year'] == year)]\n
 females = df_name.loc[df_name['gender'] == 'F'].freq\n        males = df_nam
e.loc[df_name['gender'] == 'M'].freq\n    if (len(females) > 0 and len(males)
> 0):\n        if(females.values[0] > males.values[0]):\n            moreFema
leThanMale = True\n        else:\n            moreFemaleThanMale = False\n
     if(moreFemaleThanMale and not swapHappened and males.values[0] > female
s.values[0]):\n            swapHappened = True;\n            femaleToMaleLis
t.append(name);\n        if(not moreFemaleThanMale and not swapHappened and f
emales.values[0] > males.values[0]):\n            swapHapened = True;\n
      maleToFemaleList.append(name);\nprint femaleToMaleList   \n"
```

5. Visualization Tools and Missing/Hidden Values.

Visualization is important both for exploring the data, as well as for explaining what you have done. There are a huge number of such tools now available. This exercise walks through various functionalities of matplotlib and pandas.

- The first part of this exercise was created by Dataquest. Run through the commands given in this tutorial: https://www.dataquest.io/blog/matplotlib-tutorial/ (https://www.dataquest.io/blog/matplotlib-tutorial/) and under- stand the code.
- Suppose that you would now like to plot some of the results by state. As you will see, the state information is sometimes missing, and other times it comes in varying forms. Figure out how to aggregate the results by state. The challenge here: how many of the tweets can you (correctly) assign to a state? Note: depending on how well you want to do (i.e., how many tweets you want to correctly assign to their state), this is not an easy problem!

In [10]:
```
tweets = pd.read_csv("Lab2_data/tweets.csv")
```

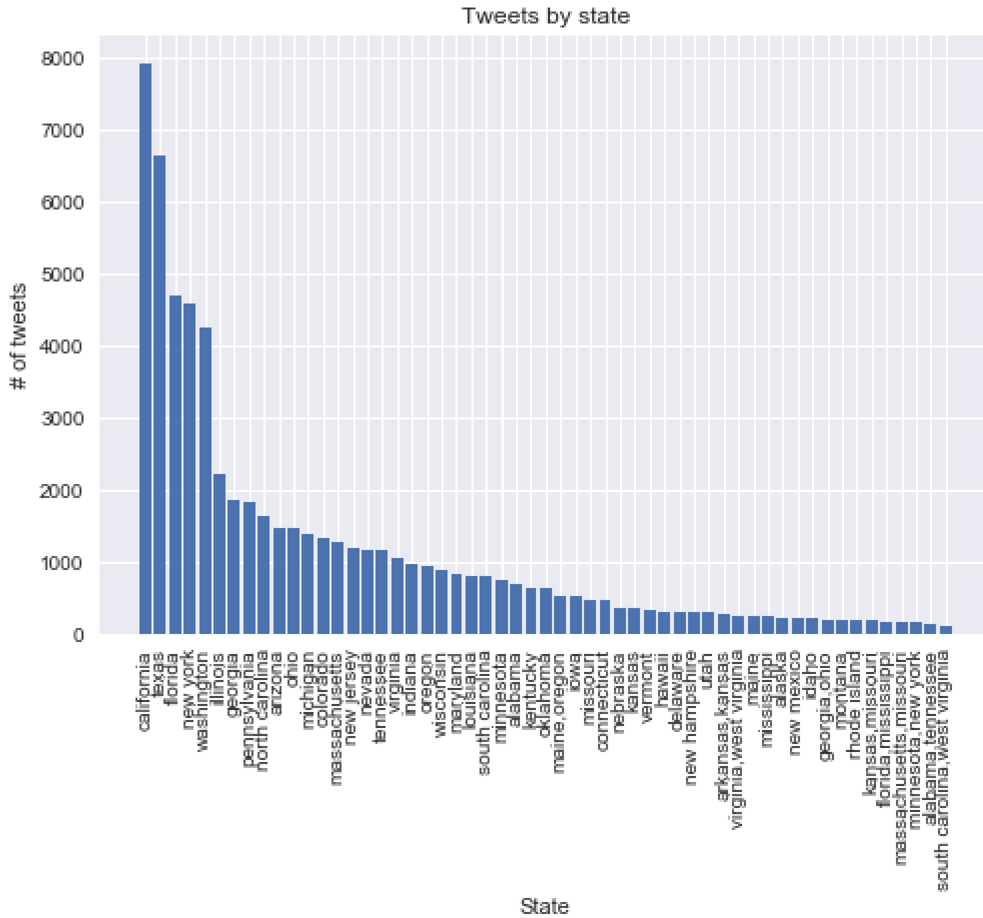
In [11]:
```
def get_state(row):
    state = []
    text = row["user_location"].lower()

    if "alabama" in text or r"\bal\b" in text or "birmingham" in text or "mobi
le" in text or "huntsville" in text:
        state.append("alabama")
    if "alaska" in text or r"\bak\b" in text or "anchorage" in text or "fairba
nks" in text or "juneau" in text:
        state.append("alaska")
    if "arizona" in text or r"\baz\b" in text or "phoenix" in text or "tuscon"
 in text or "mesa" in text:
        state.append("arizona")
    if "arkansas" in text or r"\bar\b" in text or "little rock" in text or "fo
rt smith" in text or "fayetteville" in text:
        state.append("arkansas")
    if "california" in text or r"\bca\b" in text or "los angeles" in text or
"san diego" in text or "san jose" in text or "san francisco" in text:
        state.append("california")
    if "colorado" in text or r"\bco\b" in text or "enver" in text or "colorado
 springs" in text or "aurora" in text:
        state.append("colorado")
    if "connecticut" in text or r"\bct\b" in text or "bridgeport" in text or
"new haven" in text or "hartford" in text:
        state.append("connecticut")
    if "delaware" in text or r"\bde\b" in text or "wilmington" in text or "dov
er" in text or "newark" in text:
        state.append("delaware")
    if "florida" in text or r"\bfl\b" in text or "jacksonville" in text or "mi
ami" in text or "tampa" in text:
        state.append("florida")
    if "georgia" in text or r"\bga\b" in text or "atlanta" in text or "august
a" in text or "columbus" in text:
        state.append("georgia")
    if "hawaii" in text or r"\bhi\b" in text or "honolulu" in text or "hilo" i
```

```
n text or "kailua" in text:
        state.append("hawaii")
    if "idaho" in text or r"\bid\b" in text or "boise" in text or "nampa" in t
ext or "idaho falls" in text:
        state.append("idaho")
    if "illinois" in text or r"\bil\b" in text or "chicago" in text or "auror
a" in text or "rockford" in text:
        state.append("illinois")
    if "indiana" in text or r"\bin\b" in text or "indianapolis" in text or "fo
rt wayne" in text or "evansville" in text:
        state.append("indiana")
    if "iowa" in text or r"\bia\b" in text or "des moines" in text or "cedar r
apids" in text or "davenport" in text:
        state.append("iowa")
    if "kansas" in text or r"\bks\b" in text or "wichita" in text or "overland
 park" in text or "kansas city" in text:
        state.append("kansas")
    if "kentucky" in text or r"\ky\b" in text or "louisville" in text or "lexi
ngton" in text or "owensboro" in text:
        state.append("kentucky")
    if "louisiana" in text or r"\bla\b" in text or "new orleans" in text or "s
hreveport" in text or "baton rouge" in text:
        state.append("louisiana")
    if "maine" in text or r"\bme\b" in text or "portland" in text or "lewisto
n" in text or "bangor" in text:
        state.append("maine")
    if "maryland" in text or r"\bmd\b" in text or "baltimore" in text or "fred
erick" in text or "gaithersburg" in text:
        state.append("maryland")
    if "massachusetts" in text or r"\bma\b" in text or "boston" in text or "wo
rcester" in text or "springfield" in text:
        state.append("massachusetts")
    if "michigan" in text or r"\bmi\b" in text or "detroit" in text or "grand
 rapids" in text or "warren" in text:
        state.append("michigan")
    if "minnesota" in text or r"\bmn\b" in text or "minneapolis" in text or "s
aint paul" in text or "rochester" in text:
        state.append("minnesota")
    if "mississippi" in text or r"\bms\b" in text or "jackson" in text or "gul
fport" in text or "biloxi" in text:
        state.append("mississippi")
    if "missouri" in text or r"\bmo\b" in text or "kansas city" in text or "sa
int louis" in text or "springfield" in text:
        state.append("missouri")
    if "montana" in text or r"\bmt\b" in text or "bilings" in text or "missoul
a" in text or "great falls" in text:
        state.append("montana")
    if "nebraska" in text or r"\bne\b" in text or "omaha" in text or "lincoln"
 in text or "bellevue" in text:
        state.append("nebraska")
    if "nevada" in text or r"\bnv\b" in text or "las vegas" in text or "reno"
in text or "henderson" in text:
        state.append("nevada")
    if "new hampshire" in text or r"\bnh\b" in text or "machester" in text or
"nashua" in text or "concord" in text:
        state.append("new hampshire")
    if "new jersey" in text or r"\bnj\b" in text or "newark" in text or "jerse
```

```
y city" in text or "paterson" in text:
        state.append("new jersey")
    if "new mexico" in text or r"\bnm\b" in text or "albuquerque" in text or
"las cruces" in text or "rio rancho" in text:
        state.append("new mexico")
    if "new york" in text or r"\bny\b" in text or "new york city" in text or
"buffalo" in text or "rochester" in text:
        state.append("new york")
    if "north carolina" in text or r"\bnc\b" in text or "charlotte" in text or
 "raleigh" in text or "greensboro" in text:
        state.append("north carolina")
    if "north dakota" in text or r"\bnd\b" in text or "fargo" in text or "bism
arck" in text or "grand forks" in text:
        state.append("north dakota")
    if "ohio" in text or r"\boh\b" in text or "columbus" in text or "clevelan
d" in text or "cincinnati" in text:
        state.append("ohio")
    if "oklahoma" in text or r"\bok\b" in text or "oklahoma city" in text or
"tulsa" in text or "norman" in text:
        state.append("oklahoma")
    if "oregon" in text or r"\bor\b" in text or "portland" in text or "salem"
in text or "eugene" in text:
        state.append("oregon")
    if "pennsylvania" in text or r"\bpa\b" in text or "philadelphia" in text o
r "pittsburgh" in text or "allentown" in text:
        state.append("pennsylvania")
    if "rhode island" in text or r"\bri\b" in text or "providence" in text or
"warwick" in text or "cranston" in text:
        state.append("rhode island")
    if "south carolina" in text or r"\bsc\b" in text or "charleston" in text o
r "columbia" in text or "north charleston" in text:
        state.append("south carolina")
    if "south dakota" in text or r"\bsc\b" in text or "sioux falls" in text or
 "rapid city" in text or "aberdeen" in text:
        state.append("south dakota")
    if "tennessee" in text or r"\btn\b" in text or "memphis" in text or "mobil
e" in text or "huntsville" in text:
        state.append("tennessee")
    if "texas" in text or r"\btx\b" in text or "austin" in text or "houston" i
n text or "dallas" in text:
        state.append("texas")
    if "utah" in text or r"\but\b" in text or "salt lake city" in text or "wes
t valley city" in text or "provo" in text:
        state.append("utah")
    if "vermont" in text or r"\bvt\b" in text or "burlington" in text or "sout
h burlington" in text or "rutland" in text:
        state.append("vermont")
    if "virginia" in text or r"\bva\b" in text or "virginia beach" in text or
"norfolk" in text or "chesapeake" in text:
        state.append("virginia")
    if "washington" in text or r"\bwa\b" in text or "seattle" in text or "spok
ane" in text or "tacoma" in text:
        state.append("washington")
    if "west virginia" in text or r"\bwv\b" in text or "charleston" in text or
 "huntington" in text or "parkersburg" in text:
        state.append("west virginia")
    if "wisconsin" in text or r"\bwi\b" in text or "milwaukee" in text or "mad
```

```
ison" in text or "green bay" in text:
        state.append("wisconsin")
    if "wyoming" in text or r"\bwy\b" in text or "cheyenne" in text or "caspe
r" in text or "laramie" in text:
        state.append("wyoming")

    return ",".join(state)

tweets["user_location"] = tweets["user_location"].astype(str)
tweets["state"] = tweets.apply(get_state,axis=1)

#word boundary doesn't work
```

In [17]:
```
counts = tweets["state"].value_counts()
counts = counts.drop("") #drop the count with no state attributed

countsDF = pd.DataFrame(counts)
countsDF.columns = ['Count']
countsDF = countsDF[countsDF['Count'] > 100]
tempXVals = []
for i in range(len(countsDF.index)):
  tempXVals.append(i)
plt.bar(tempXVals, countsDF.Count)
plt.xticks(tempXVals, countsDF.index, rotation = 90)
plt.title("Tweets by state")
plt.xlabel("State")
plt.ylabel("# of tweets")
plt.show()

print(counts)
```


Tweets by state
# of tweets
8000
7000
6000
5000
4000
3000
2000
1000
0
california
texas
florida
new york
washington
illinois
georgia
pennsylvania
north carolina
arizona
ohio
michigan
colorado
massachusetts
new jersey
nevada
tennessee
virginia
indiana
oregon
wisconsin
maryland
louisiana
south carolina
minnesota
alabama
kentucky
oklahoma
maine,oregon
iowa
missouri
connecticut
nebraska
kansas
vermont
hawaii
delaware
new hampshire
utah
arkansas,kansas
virginia,west virginia
maine
mississippi
alaska
new mexico
idaho
georgia,ohio
montana
rhode island
kansas,missouri
florida,mississippi
massachusetts,missouri
minnesota,new york
alabama,tennessee
south carolina,west virginia
State

```
california                        7926
texas                             6630
florida                           4705
new york                          4581
washington                        4261
illinois                          2222
georgia                           1856
pennsylvania                      1815
north carolina                    1638
arizona                           1469
ohio                              1459
michigan                          1390
colorado                          1320
massachusetts                     1270
new jersey                        1200
nevada                            1164
tennessee                         1161
virginia                          1046
indiana                            958
oregon                             937
wisconsin                          875
maryland                           821
louisiana                          799
south carolina                     798
minnesota                          743
alabama                            697
kentucky                           641
oklahoma                           627
maine,oregon                       516
iowa                               514
                                  ...
georgia,maryland                     1
south dakota,texas                   1
georgia,texas                        1
arkansas,kansas,texas                1
california,oklahoma                  1
nebraska,texas                       1
alabama,georgia                      1
ohio,washington                      1
louisiana,wisconsin                  1
california,minnesota                 1
nevada,utah                          1
indiana,nebraska                     1
new york,oregon                      1
california,ohio                      1
indiana,new mexico,ohio              1
alabama,colorado                     1
michigan,tennessee                   1
california,new jersey                1
ohio,wisconsin                       1
indiana,michigan                     1
arizona,new york                     1
arkansas,georgia,kansas              1
maryland,pennsylvania                1
georgia,illinois,ohio                1
delaware,new jersey,ohio             1
alabama,california                   1
```

```
arkansas,kansas,oklahoma,texas       1
illinois,tennessee                   1
illinois,new york                    1
illinois,massachusetts               1
Name: state, Length: 312, dtype: int64
```

1. Standard Error:

It is important to develop an intuition for how much error we should expect" when we solve a particular statistical problem. As the number of sample increase, we should expect the error to decrease. But by how much? In the rst lab, you generated samples from a univariate (Problem 3) and multivariate (Problem 4) Gaussian with given parameters, and then you were asked to estimate those parameters from the data you generated. In this exercise, we derive explicitly the relationship that you (should have) observed doing those exercises.

- Suppose $Z = N(\mu; \theta^2)$, i.e., Z is a univariate Gaussian (a.k.a. normal) random variable with mean μ and variance θ^2. Suppose that you see n samples from Z, i.e., you see data $z_1, \ldots, z_n$.Let
$$z_{avg} = \sum_{i=1}^{n} \frac{z_i}{n}$$
denote the sample mean.

We want to answer: how close is z_{avg} to μ?

Note that z_{avg} is a random variable so we need to quantify in a probabilistic way how close z_{avg} is to μ.

- Suppose Z ~ $N(0, 1)$. This is also called a standard normal random variable. For n = 10,000, compute the probability that z_{avg} > 0.1, z_{avg} > 0.01, and $z_{avg} > 0.001$
- Now for the general case: suppose $Z = N(\mu; \theta^2)$, and for general n, compute the probability that $z_{avg} > n^{\frac{-1}{3}}$, $z_{avg} > n^{\frac{-1}{2}}$, and $z_{avg} > n^{\frac{-2}{3}}$. For your calculations, you can let n scale if that makes things easier.

Answer

$$P(z_{avg} > c)$$

Re-arrange variables

$$P(S_n > c * n)$$

Using Central Limit Theorem

$$P(\frac{S_n}{\sqrt{n} * \theta} > \frac{0.1 * n}{\sqrt{n} * \theta})$$
$$P(Z_n > 0.1 * \sqrt{n})$$

Standard Normal

$$1 - P(Z_n < c * \sqrt{10000})$$

Replace:

c = 0.1

$$P(z_{avg} > 0.1)$$
$$1 - P(Z_n < 10) = 0$$

c = 0.01

$$P(z_{avg} > 0.01)$$
$$1 - P(Z_n < 1) = 0.16$$

c = 0.001

$$P(z_{avg} > 0.001)$$
$$1 - P(Z_n < 0.1) = 0.46$$

General Case

$$Z = N(\mu; \theta^2)$$
$$P(Z_{avg} > c)$$

Using Central Limit Theorem

$$P(\frac{S_n - (n*\mu)}{\sqrt{n}*\theta} > \frac{c*n - (n*\mu)}{\sqrt{n}*\theta})$$

For $c = n^{-1/3}$

$$P(Z_{avg} - \mu > n^{-1/3})$$
$$\hat{x} = n^{-1/3} + \mu$$
$$Z = \frac{n^{-1/3} + \mu - \mu}{\delta/\sqrt(n)}$$
$$Z = \frac{n^{-1/3}}{\delta}n^{1/2} = \frac{n^{1/6}}{\delta}$$
$$= P(Z < \frac{-n^{1/6}}{\delta})$$

For $c = n^{-1/2}$

$$P(Z_{avg} - \mu > n^{-1/2})$$
$$\hat{x} = n^{-1/2} + \mu$$
$$Z = \frac{n^{-1/2} + \mu - \mu}{\delta/\sqrt(n)}$$
$$Z = \frac{n^{-1/2}}{\delta}n^{1/2} = \frac{n^{-1/2}}{\delta}$$
$$= P(Z < \frac{-1}{\delta})$$

For $c = n^{-2/3}$

$$P(Z_{avg} - \mu > n^{-2/3})$$
$$\hat{x} = n^{-2/3} + \mu$$
$$Z = \frac{n^{-2/3} + \mu - \mu}{\delta/\sqrt(n)}$$
$$Z = \frac{n^{-2/3}}{\delta}n^{1/2} = \frac{n^{-1/6}}{\delta}$$
$$= P(Z < \frac{-n^{-1/6}}{\delta})$$

2. **More Standard Error** Consider a one dimensional regression problem, where the offset is zero. Thus, we are trying to fit a function of the form $h(x) = x \cdot \beta$. Suppose that the truth is a noisy version of this – that is, the true model according to which data are generated is:

$$y_i = x_i \cdot \beta + e_i.$$

Everything in the above equation is a scalar, i.e., $y_i, x_i, \beta, e_i \in \mathbb{R}$. Here, e_i represents independent noise that is not modeled by the linear relationship.

- When we have n data points, the least squares objective reads:

$$\min_{\beta} : \quad \frac{1}{n} \sum_{i=1}^{n} (x_i \beta - y_i)^2.$$

  Show that this is a quadratic function in β, that is, if we expand it, it has the form

$$A\beta^2 + B\beta + C.$$

- Compute A, B, and C explicity, i.e., as explicit functions of the data, $\{x_i, y_i\}$. Note that these should not be functions of β. Show that $A \geq 0$ regardless of the values of the data.
- Since $A \geq 0$, this is a quadratic function whose graph opens up. This means that it is convex, and therefore the solution is characterized as the solution obtained by setting the first derivative (w.r.t. β) equal to zero. Do this, and therefore explicitly solve for the solution $\hat{\beta}$. This is the one-dimensional form of what is known as the *normal equations*. *Hint: we did this problem in class.*
- Now using the one dimensional expression from the second part, and plugging in the relationship $y_i = x_i \cdot \beta + e_i$, write

$$\hat{\beta} = \beta + Z\mathbf{e},$$

  where $\mathbf{e}$ denotes the vector of all the errors, e_i, added in each stage, and where Z is a matrix of appropriate dimension. What is Z, explicitly?

(Bonus) Repeat the last two questions in the general case. That is, derive the normal equations and the standard error for the general (vector) case, where our model is

$$y_i = x_i^\top \beta + e_i,$$

where now $x_i, \beta \in \mathbb{R}^p$, and $x_i^\top \beta$ denotes the dot product.

Answer:

Part 1

When we have n data points, the least square objective reads:

$$min\beta = \frac{1}{n}\sum_{i=1}^{n}(x_i\beta - y_i)^2$$

We can show that β is a quadratic formula of the form $A\beta^2 + B\beta + C$

Multiplying out

$$min\beta = \frac{1}{n}[\sum_{i=1}^{n}(x_i^2\beta^2 - 2X_i\beta y_i + y_i^2]$$

Expand out

$$min\beta = \beta^2\frac{1}{n}\sum_{i=1}^{n}x_i^2 - \beta\frac{2}{n}\sum_{i=1}^{n}2x_iy_i + \frac{1}{n}\sum_{i=1}^{n}y_i^2$$

where

$$A = \frac{1}{n}\sum_{i=1}^{n}x_i^2$$
$$B = \frac{2}{n}\sum_{i=1}^{n}2x_iy_i$$
$$C = \frac{1}{n}\sum_{i=1}^{n}y_i^2$$

Notice A is a quadratic and $A \geq 0$

Part 2

Solve for β

$$\frac{d}{d\beta}(\frac{1}{n}\sum_{i=1}^{n}(X_i\beta - y_i)^2) = \frac{2}{n}\sum_{i=1}^{n}(X_i\beta - y)X_i) = 0$$
$$\sum_{i=1}^{n}(X_i^2\beta - X_iY_i) = 0$$
$$\therefore \beta = \frac{\sum_{i=1}^{n}X_iY_i}{\sum_{i=1}^{n}X_i^2}$$