Linear Regression Homework 1

UNI: qy2205
Name: Quan Yuan
Email: quan.yuan@columbia.edu

1. Chapter 1 problem 19

a. Obtain the least square estimate of β_0 and β_1, and state the estimated reg function

Solution:
For any questions like:

$$y_i = \beta_1 + \beta_2 x_{2i} + \ldots + \beta_k x_{ki} + \epsilon_i$$

where $i \in (1, 2, .., n)$

We set y = $\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ $X = \begin{pmatrix} 1 & x_{21} & \cdots & x_{k1} \\ 1 & x_{22} & \cdots & x_{k2} \\ \vdots & & & \\ 1 & x_{2n} & \cdots & x_{kn} \end{pmatrix}$ $\beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix}$ $\epsilon = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$

Thus, the function can be written as:

$$y = X\beta + \epsilon$$

We define the error as

$$S = \sum_i^n \epsilon_i^2 = \epsilon_i^T \epsilon_i = (y - X\beta)^T (y - X\beta) = y^T y - y^T X\beta - \beta X^T y + \beta^T X^T X\beta$$

Then we calculate the first order derivatives of the error term, we get

$$\frac{\partial S}{\partial \beta} = -2X^T y + 2X^T X\beta$$

Let the first order derivatives equals to zero, we can get

$$\beta = (X^T X)^{-1} X^T y$$

And we also need to check the second order derivatives

$$\frac{\partial^2 S}{\partial \beta^2} = 2X^T X$$

which is a positive matrix.

The following is the `python` code for this problem

In [104]:

```
# import packages
import pandas as pd
import numpy as np
from sklearn import linear_model
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(color_codes=True)
%matplotlib inline
params = {'legend.fontsize': 5,
          'figure.figsize': (14, 6),
          'axes.labelsize': 15,
          'axes.titlesize': 18,
         'xtick.labelsize': 12,
         'ytick.labelsize': 12,
         'font.family': 'Times New Roman'}
plt.rcParams.update(params)

# clean data
CH01PR19 = pd.read_table('CH01PR19.txt', header = None, sep = ' ')[[1, 5]]
CH01PR19.columns = ['GPA', 'ACT']
CH01PR19.head(3)
```

Out[104]:

	GPA	ACT
0	3.897	21
1	3.885	14
2	3.778	28

In [105]:

```
# define our linear regression
class LinearReg:
    def __init__(self, data):
        '''data: type: pandas dataframe'''
        self.data = data
        self.length = len(data)
    def ols(self, x, y):
        '''
        x: column name
        y: column name
        '''
        X = np.matrix(np.vstack([np.ones(self.length), self.data[x].values]).T)
        y = np.matrix(self.data[y].values).T
        beta = np.linalg.inv(X.T*X)*X.T*y
        return beta
    def sklearn_ols(self, x, y):
        X = np.matrix(self.data[x].values).T
        y = np.matrix(self.data[y].values).T
        # Create linear regression object
        OLS = linear_model.LinearRegression()
        # Train the model using the training sets
        OLS.fit(X, y)
        return np.vstack([OLS.intercept_, OLS.coef_])
    def visual(self, x, y, step = 0.01):
        para = self.ols(x, y)
        X = self.data[x]
        Y = self.data[y]
        min_x, max_x = min(X), max(X)
        # x is also matrix
        func = lambda x: x*para
        x_sim = np.arange(min_x, max_x, step)
        xm = np.vstack([np.ones(len(x_sim)), x_sim]).T
        y_sim = func(xm)
        self.data.plot.scatter(x,y)
        plt.plot(x_sim, y_sim)
        plt.title('The Relationship Between {0} and {1}'.format(x, y))
```

In [106]:

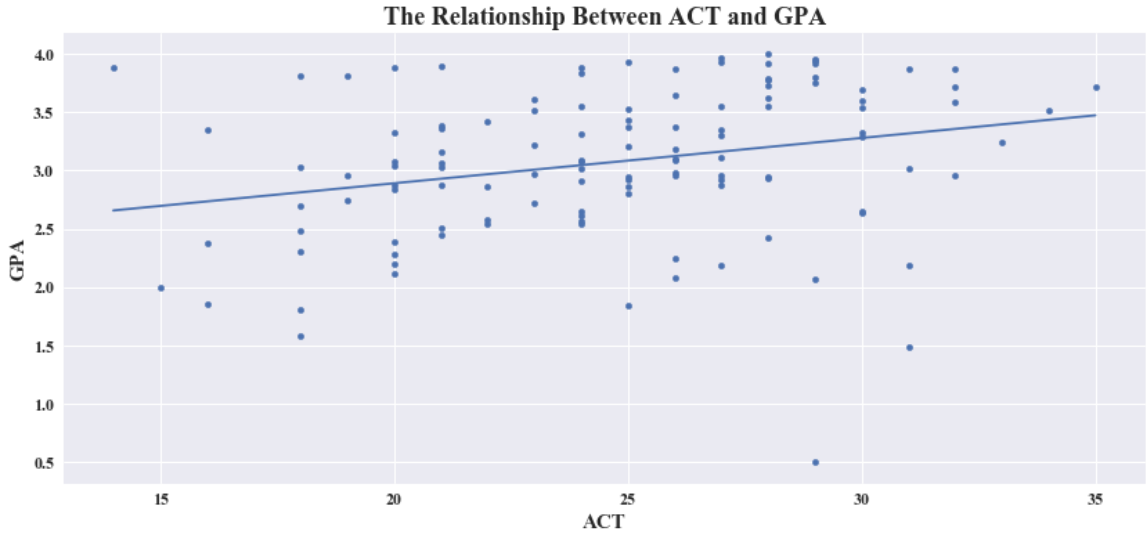
```
# Run regression
LReg19 = LinearReg(data = CH01PR19)
beta_ols = LReg19.ols(x = 'ACT', y = 'GPA')
beta_sklearn = LReg19.sklearn_ols(x = 'ACT', y = 'GPA')
print('OLS coded by myself \\beta_0 is {0}, \\beta_1 is {1}'.format(round(float(beta_ols[0]), 3
), round(float(beta_ols[1]), 3)))
print('OLS by sklearn \\beta_0 is {0}, \\beta_1 is {1}'.format(round(float(beta_sklearn[0]), 3),
 round(float(beta_sklearn[1]), 3)))
print('The line is y = {0}x + {1}'.format(round(float(beta_ols[1]), 3), round(float(beta_ols[0
]), 3)))
```

```
OLS coded by myself \beta_0 is 2.114, \beta_1 is 0.039
OLS by sklearn \beta_0 is 2.114, \beta_1 is 0.039
The line is y = 0.039x + 2.114
```

b. Plot the regression function and data

In [107]:

```
# visualization
LReg19.visual(x = 'ACT', y = 'GPA')
```



c. point estimate

In [148]:

```
meanx30 = 0.039*30 + 2.114
print('point estimate of the mean freshman GPA for student with ACT test score X = 30 is {0}'.format(meanx30))
```

```
point estimate of the mean freshman GPA for student with ACT test score X = 30 is
3.284
```

d. point estimate

The point estimate of change in the mean response would increase by 0.0388

2. Chapter 1 problem 29

Refer to regression model (1.1) $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$. Assume that X = 0 is within the scope of the model. What is the implication for the regression function if $\beta_0 = 0$ so that the model is $Y_i = \beta_1 X_i + \epsilon_i$? How would the regression function plot on a graph?

Answer: The implication of the model is that the regression line always goes through the origin. The model only depends on slope.

3. Chapter 1 problem 30

Answer: The implication of the model is that the regression line always parallel with x axis. The model only depends on intercept.

4. Chapter 1 problem 33

Refer to the regression model $Y_i = \beta_0 + \epsilon_i$. Derive the least squares estimator of β_0 for this model

Based on the regression model, we could drive the SSE is

$$SEE = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$SSE = \sum_{i=1}^{n}(Y_i - \beta_0) = n\beta_0^2 - 2\beta_0\sum_{i=1}^{n}Y_i + \sum_{i=1}^{n}{Y_i}^2 = n(\beta_0^2 - \frac{2}{n}\beta_0\sum_{i=1}^{n}Y_i + \frac{1}{n}\sum_{i=1}^{n}{Y_i}^2)$$

$$= n((\beta_0 - \frac{1}{n}\sum_{i=1}^{n}Y_i)^2 + (\frac{1}{n}\sum_{i=1}^{n}{Y_i}^2 - \frac{1}{n^2}(\sum_{i=1}^{n}Y_i)^2))$$

Thus, the least squares estimator of β_0 for this model is

$$\frac{1}{n}\sum_{i=1}^{n}Y_i$$

5. Chapter 1 problem 34

show β_0 is unbiased

$$E(\beta_0) = \frac{1}{n}\sum_{i=1}^{n}E(Y_i) = \frac{1}{n}\sum_{i+1}^{n}\frac{\sum_{i=1}^{n}\beta_0 + \epsilon_i}{n}$$

Since $\sum_{i=1}^{n}\epsilon_i = 0$
Therefore,

$$E(\beta_0) = \beta_0$$

Another: show β_1 is unbiased (for general one factor linear regression)

For this question, we can simply write the formula of β_1 as

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})y_i - (x_i - \bar{x})\bar{y}}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})y_i}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

So,

$$E(\beta_1) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})E(y_i)}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(\beta_0 + \beta_1 x_i)}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$= \frac{\beta_0 \sum x_i - n\bar{x}\beta_0 + \beta_1 \sum x_i^2 - \bar{x}\beta_1 \sum x_i}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$= \frac{\beta_1(\sum x_i^2 - n\bar{x}^2)}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Since

$$\sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum x_i^2 - 2\bar{x}\sum x_i + \sum \bar{x}^2 = \sum x_i^2 - n\bar{x}^2$$

Therefore, the final result is:

$$E(\beta_1) = \frac{\beta_1(\sum x_i^2 - n\bar{x}^2)}{\sum x_i^2 - n\bar{x}^2} = \beta_1$$

So, β_1 is unbiased

6. Chapter 1 problem 39

(a) show the least square reg line fitted to three points is the same as fiitted to six points

We can get the coefficients of the fitted line with 6 points as:

$$\beta_1 = \frac{\sum_{i=1}^{6}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{6}(x_i - \bar{x})^2}$$

According to six points fitted line, we could get:

$$\beta_1 = \frac{-5(y_{11} - \bar{y}) - 5(y_{12} - \bar{y}) + 5(y_{31} - \bar{y}) + 5(y_{32} - \bar{y})}{2 \times (5^2 \times 5^2)}$$

According to three points fitted line, we could get:

$$\beta_1 = \frac{5(y_{31} + y_{32} - y_{11} - y_{12})}{2 \times (5^2 \times 5^2)}$$

Thus the two fitted lines' coefficients are equal. And we could also draw a conclusion that β_0 of two lines are also equal since

$$\hat{\beta_0} = \hat{y} - \hat{\beta_1}\bar{x}$$

(b) could the error term variance σ^2 be estimated without fitting a regression line?

Yes. since if the regression line hold, the vaiance of error term should be constant. So we can simply calculate the variance in fixed x point. $\sigma_j = E(Y_{1i} - \hat{Y_1})^2$. So the variance σ^2 can be estimated like $E(\sigma_j)$

7. Chapter 1 problem 41

Refer to the regression model $Y_i = \beta_1 X_i + \epsilon_i, i = 1, 2, 3...n$

(a) Find the least squares estimator of β_1

method1: Calculate the residuals:

$$Error = \sum (Y_i - \beta_1 X_i)^2$$
$$= \sum ({Y_i}^2 - 2\beta_1 \sum X_i Y_i + \beta_1^2 \sum X_i^2)$$

So minimize $f(\beta_1)$ is

$$\hat{\beta_1} = \frac{\sum X_i Y_i}{\sum X_i^2}$$

method2: We have already find the solution of β_1 in Chapter 1 Problem 19 with matrix form.

$$\beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} = (X^T X)^{-1} X^T y$$

Where the regression line is:

$$y_i = \beta_1 + \beta_2 x_{2i} + \ldots + \beta_k x_{ki} + \epsilon_i$$

So we can simply trun the first column of X to 0 and calculate the result, so result in this format is still general.

(b) Estimate β_1 with maximum likelihood method

method 1:

$$Y_i \sim N(\beta_1 X_i, \sigma^2)$$
$$L(\beta_1; Y) = \frac{1}{n} \sum ln(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \beta_1 X_i)^2}{2\sigma^2}})$$
$$= \frac{1}{n} \sum (-\frac{(Y_i - \beta_1 X_i)^2}{2\sigma^2} + constant)$$
$$= |A_1| \sum -(Y_i - \beta_1 X_i)^2 + A_2$$

Thus we need to maximum

$$\sum -(Y_i - \beta_1 X_i)^2$$

which is exactly the same as we did before, so the β_1 estimator is the same.

method 2:
matrix operation
First we drive the likelihood function:

$$p(\epsilon_i) = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{\epsilon_i^2}{2\sigma^2})$$
$$p(y_i|x_i;\beta) = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(y_i-\beta^T x_i)^2}{2\sigma^2})$$
$$L(\theta) = \prod_{i=1}^{n} p(y_i|x_i;\beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(y_i-\beta^T x_i)^2}{2\sigma^2})$$
$$l(\theta) = logL(\theta) = log\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(y_i-\beta^T x_i)^2}{2\sigma^2})$$
$$= \sum_{i=1}^{n} log\frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(y_i-\beta^T x_i)^2}{2\sigma^2})$$
$$= mlog\frac{1}{\sqrt{2\pi\sigma}} - \frac{1}{2\sigma^2}\sum_{i=1}^{m}(y_i-\beta^T x_i)^2$$

Since $mlog\frac{1}{\sqrt{2\pi\sigma}}$ is a constant, so we only need to consider the minimum of

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

We can also write above formula as:

$$\frac{1}{2}(X\theta - y)^T(X\theta - y)$$

Therefore, the following thing is almost like we write in problem 19, calculate the first order derivatives and let it equals to zero

The final result is also

$$\beta = (X^T X)^{-1} X^T y$$

So the result is the same as the estimator when we used with least square method

(c) show that the MLE of β_1 is unbiased

$$E(\hat{\beta}_1) = \frac{\sum X_i E(Y_i)}{\sum X_i^2} = \frac{\sum X_i E(\beta_1 X_i + \epsilon_i)}{\sum X_i^2} = \beta_1$$

8. Chapter 1 problem 43

In [109]:

```
# load data
CDI = pd.read_excel('CDI.xlsx')
CDI.head()
```

Out[109]:

	id	country	state	land_area	total_population	percent_of_population18_34
0	1	Los_Angeles	CA	4060	8863164	32.1
1	2	Cook	IL	946	5105067	29.2
2	3	Harris	TX	1729	2818199	31.3
3	4	San_Diego	CA	4205	2498016	33.5
4	5	Orange	CA	790	2410556	32.6

(a) Regression

- (Y) number_of_active_physicians
- (X) total_population, number_of_hospital_beds, total_personal_income

For this problem, we use the Linear Regression class (LinearReg) to solve it.

In [175]:

```
# run regression
target = 'number_of_active_physicians'
var_list = ['total_population', 'number_of_hospital_beds', 'total_personal_income']
CDI_reg = LinearReg(CDI)
beta = []
for i, j in zip(var_list, [target]*3):
    beta.append(CDI_reg.ols(x = i, y = j))

# print result
for i, j in zip(var_list, beta):
    print('For the relationship between {0} and {1}'.format(i, target))
    print('y = {0}x {1}'.format(round(float(j[1]), 3), round(float(j[0]), 3)))
```

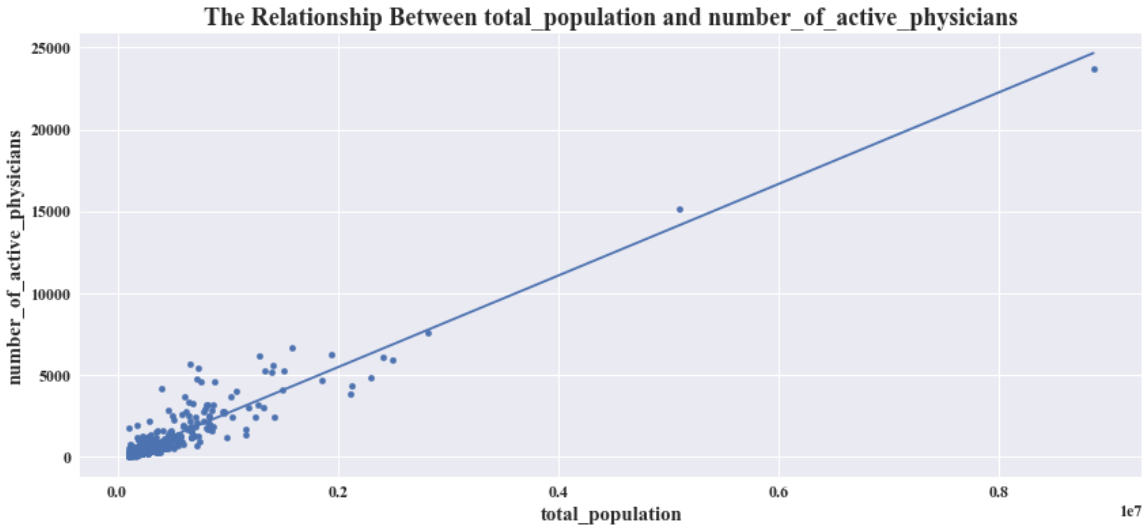
```
For the relationship between total_population and number_of_active_physicians
y = 0.003x -110.635
For the relationship between number_of_hospital_beds and number_of_active_physicia
ns
y = 0.743x -95.932
For the relationship between total_personal_income and number_of_active_physicians
y = 0.132x -48.395
```

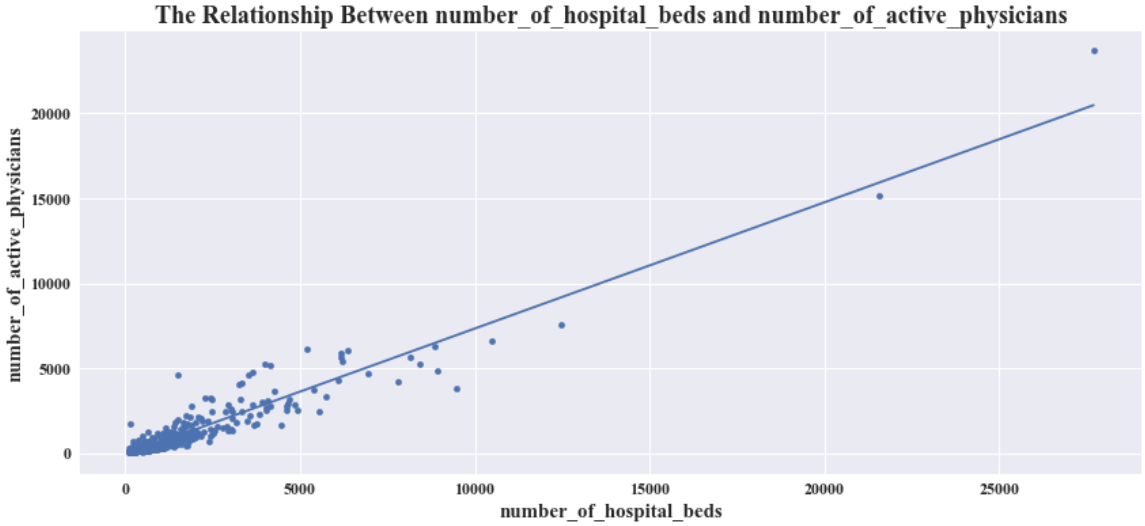
(b) plot three estimated regression functions and data on separate graphs.

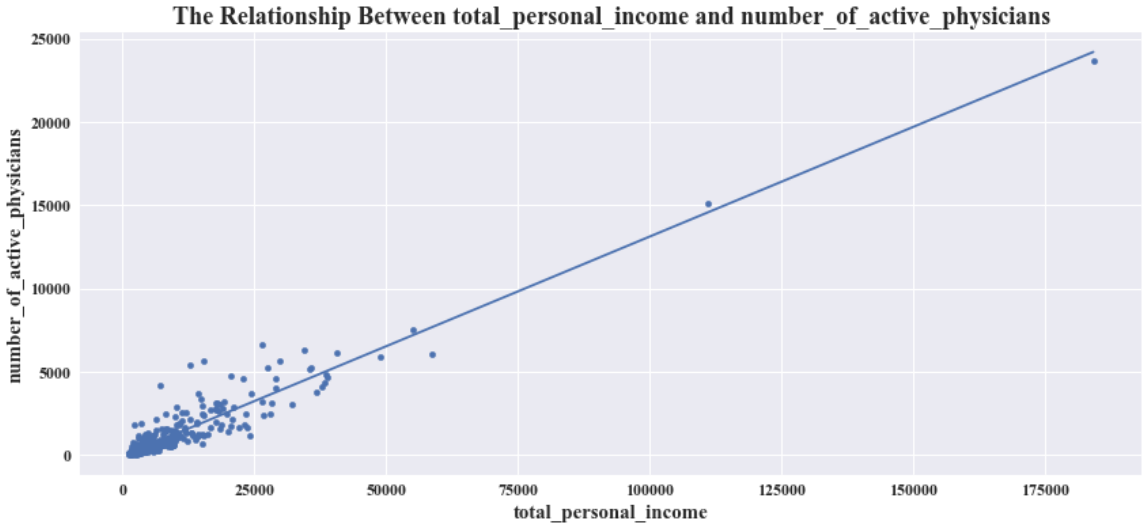
For this question, we also use LinearReg class

In [170]:

```
# visualzaition
step_list = [10000, 100, 1000]
for i, j, k in zip(var_list, [target]*3, step_list):
    CDI_reg.visual(x = i, y = j, step = k)
```



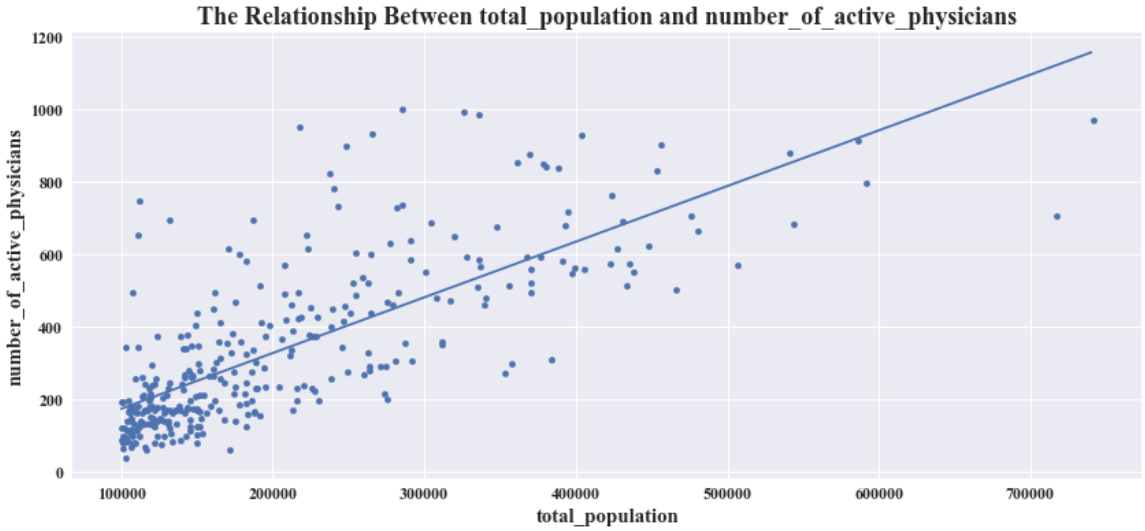


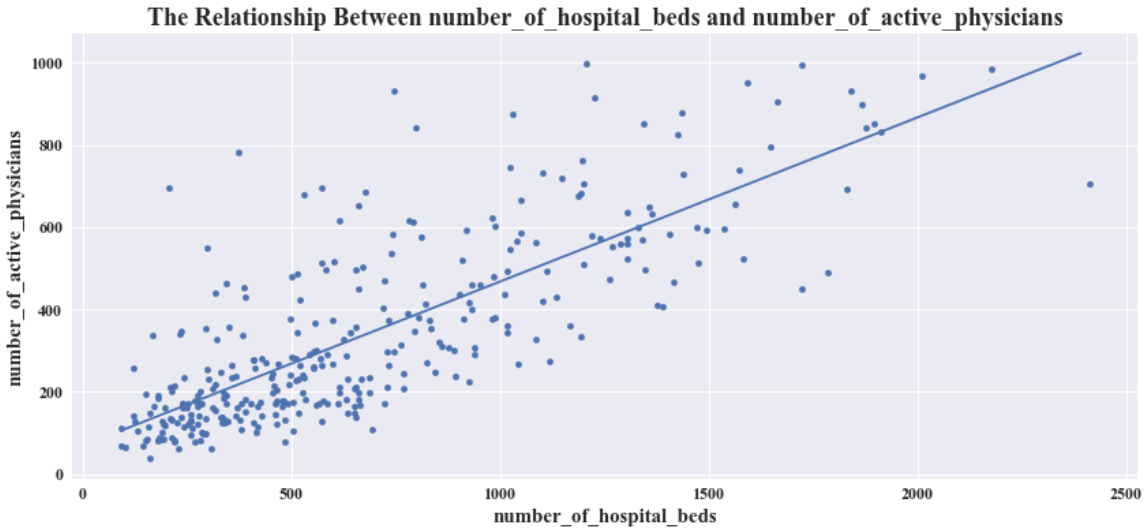


- According to the graphs above, we cannot find any patterns since the effect of extreme value, so we adjust the data for better visualization. For doing that, we simply set the limits to our dependent variable.
- only visualize the data which numbe of active physicians lower than 1000

In [171]:

```
CDI_visual = LinearReg(CDI[CDI['number_of_active_physicians'] < 1000])
for i, j, k in zip(var_list, [target]*3, step_list):
    CDI_visual.visual(x = i, y = j, step = k)
```


The Relationship Between total_population and number_of_active_physicians
number_of_active_physicians
total_population
0
200
400
600
800
1000
1200
100000
200000
300000
400000
500000
600000
700000


The Relationship Between number_of_hospital_beds and number_of_active_physicians
number_of_active_physicians
number_of_hospital_beds
0
200
400
600
800
1000
0
500
1000
1500
2000
2500

Answer: According to the graphs above, the regression line (relationship with hospital bed) seems provide a good fit. However, the other lines are not fit good since the variances are not constant at least when we set the limit that the number of active physician should less than 1000.

(c) MSE

According to (a), we have three prediction functions and we use those to calculate MSE

In [176]:

```
# code function
f1_popu = lambda x: 0.0028*x - 110.635
f2_hospital = lambda x: 0.743*x - 95.932
f3_income = lambda x: 0.132*x - 48.395
```

In [177]:

```
# calculate prediction value
x1 = CDI['total_population']
y1 = x1.map(f1_popu)
x2 = CDI['number_of_hospital_beds']
y2 = x2.map(f2_hospital)
x3 = CDI['total_personal_income']
y3 = x3.map(f3_income)
# take real value
real = CDI['number_of_active_physicians']
# calculate MSE
def mse(pred, real):
    '''pred: series; real: series'''
    MSE = sum((pred - real).map(lambda x: pow(x, 2)))/(len(pred) - 2)
    return MSE
MSE1 = mse(y1, real)
MSE2 = mse(y2, real)
MSE3 = mse(y3, real)
print('MSE1 ({0}) is {1}'.format(var_list[0], MSE1))
print('MSE2 ({0}) is {1}'.format(var_list[1], MSE2))
print('MSE3 ({0}) is {1}'.format(var_list[2], MSE3))
```

```
MSE1 (total_population) is 372214.35467739595
MSE2 (number_of_hospital_beds) is 310191.983671758
MSE3 (total_personal_income) is 324559.80354953464
```

According to the result we calculate above, the predictor variable: number of hospital beds leads to the smallest variability.

9. Chapter 1 problem 44

(a) For each geographic region, regress per capita income in a CDI (Y) against the per-centage of individuals in a country having at least a bachelor's degree (X).

For this question, we use LinearReg class and mse function to finish it.

In [159]:

```
# select the columns we need to use in this question
newCDI = CDI[['per_capita_income', 'precent_bachelor_deg', 'geographic_region']]
print('geographic_region: {0}'.format(set(newCDI['geographic_region'])))
```

```
geographic_region: {1, 2, 3, 4}
```

In [163]:

```
# instantiation
LR1 = LinearReg(newCDI[newCDI['geographic_region'] == 1])
LR2 = LinearReg(newCDI[newCDI['geographic_region'] == 2])
LR3 = LinearReg(newCDI[newCDI['geographic_region'] == 3])
LR4 = LinearReg(newCDI[newCDI['geographic_region'] == 4])
LRlist = [LR1, LR2, LR3, LR4]
regionlist = [1, 2, 3, 4]
# run regression
result = []
for each_lr in LRlist:
    result.append(each_lr.ols(x = 'precent_bachelor_deg', \
                              y = 'per_capita_income'))
# print result
for i, j in zip(regionlist, result):
    print('For region {0}'.format(i))
    print('y = {0}x + {1}'.format(round(float(j[1]), 3), round(float(j[0]), 3)))
```

```
For region 1
y = 522.159x + 9223.816
For region 2
y = 238.669x + 13581.405
For region 3
y = 330.612x + 10529.785
For region 4
y = 440.316x + 8615.053
```

(b) Are the estimated regression functions similar for the four regions?

Answer: The regressoion functions similar for the four regions, for the slope, all four functions' slope around 400. For the intercept, all four functions' intercept around 10000

(c) MSE

In [162]:

```
# code function
f_region1 = lambda x: 522.159*x + 9223.816
f_region2 = lambda x: 238.669*x + 13581.405
f_region3 = lambda x: 330.612*x + 10529.785
f_region4 = lambda x: 440.316*x + 8615.053
# calculate prediction value
xlist = []
y_pred_list = []
y_real_list = []
for i in regionlist:
    xlist.append(newCDI[newCDI['geographic_region'] == i]['precent_bachelor_deg'])
    y_real_list.append(newCDI[newCDI['geographic_region'] == i]['per_capita_income'])
    y_pred_list.append(xlist[i-1].map(eval('f_region' + str(i))))
MSEresult = []
# calculate MSE
for i, j in zip(y_pred_list, y_real_list):
    MSEresult.append(mse(i, j))
for i, j in zip(regionlist, MSEresult):
    print('MSE (region {0}) is {1}'.format(i, j))
```

```
MSE (region 1) is 7335007.5742318705
MSE (region 2) is 4411341.031215522
MSE (region 3) is 7474349.403388513
MSE (region 4) is 8214317.882861176
```

Answer: The MSE for region 1, 3, 4 are almost the same (around 8000000). But for region 2, the variability is not around 8000000, just have half of it.