DS-GA 1002 - Homework 11

Eric Niblock

November 22nd, 2020

1. **We perform a least squares fit on a dataset $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^2$ with at least 3 distinct x_i-values. Suppose (x_n, y_n) lies on the fitted line. If we omit (x_n, y_n) from the dataset and perform least squares again, does the fitted line change? Either prove it doesn't change, or give an example where it does.**

   Given the inclusion of (x_n, y_n), assume we have found some fitted line such that,

$$\hat{y} = B_0 + B_1 x \tag{1}$$

   With the further condition that,

$$y_n = B_0 + B_1 x_n \tag{2}$$

   With the least squares estimators B_0, B_1 being generated via,

$$B_0, B_1 = \underset{b_0, b_1 \in \mathbb{R}}{argmin} \sum_{i=1}^{n} (Y_i - b_0 - b_1 x_i)^2 \tag{3}$$

   However, since $y_n - B_0 - B_1 x_n = 0$, we see that,

$$B_0, B_1 = \underset{b_0, b_1 \in \mathbb{R}}{argmin} \sum_{i=1}^{n} (Y_i - b_0 - b_1 x_i)^2 = \underset{b_0, b_1 \in \mathbb{R}}{argmin} \sum_{i=1}^{n-1} (Y_i - b_0 - b_1 x_i)^2 \tag{4}$$

And thus the removal of (x_n, y_n) does not change our estimators B_0, B_1, and the fitted line remains the same.

2. Let $(1, Y_1), ..., (50, Y_{50})$ be draws from a simple linear model where $Y_i \sim N(\beta_0 + \beta_1 i, 3^2)$for $i = 1, ..., 50$.

(a) Find $r \in \mathbb{R}$ such that $P(B_0 \in (\beta_0 - r, \beta_0 + r)) = 0.95$, where B_0 is our least squares estimate of β_0.

We have that,

$$SE(\beta_0) = \sqrt{\frac{\sigma^2}{n}\left(1 + \frac{\overline{x}_n^2}{\hat{\sigma}_x^2}\right)} \tag{5}$$

And using $\overline{x}_n = 25.5$, and $\sigma^2 = 3^2$, we have,

$$SE(\beta_0) = \sqrt{\frac{9}{50}\left(1 + \frac{25.5}{208.25}\right)} = 0.44900 \tag{6}$$

Then the 95% confidence interval is given by,

$$[\beta_0 - 1.96SE(\beta_0), \beta_0 + 1.96SE(\beta_0)] \tag{7}$$
$$[\beta_0 - 0.88005, \beta_0 + 0.88005] \tag{8}$$

So, this implies that $r = 0.88005$.

(b) Find $r \in \mathbb{R}$ such that $P(B_1 \in (\beta_1 - r, \beta_1 + r)) = 0.95$, where B_1 is our least squares estimate of β_1.

We have that,

$$SE(\beta_1) = \sqrt{\frac{\sigma^2/n}{\hat{\sigma}_x^2}} = \sqrt{\frac{\sigma^2}{\sum_{i=1}^n (x_i - \overline{x}_n)^2}} \tag{9}$$

From simple calculation, we know that $\overline{x}_n = 25.5$, and we are given that $\sigma^2 = 3^2$, so,

$$SE(\beta_1) = \sqrt{\frac{9}{\sum_{i=1}^{n}(x_i - 25.5)^2}} = \sqrt{\frac{9}{10412.5}} = 0.029400 \tag{10}$$

Then the 95% confidence interval is given by,

$$[\beta_1 - 1.96SE(\beta_1), \beta_1 + 1.96SE(\beta_1)] \tag{11}$$
$$[\beta_1 - 0.05762, \beta_1 + 0.05762] \tag{12}$$

So, this implies that $r = 0.05762$.

(c) Find $r \in \mathbb{R}$ such that $P(B_0 + 3B_1 \in (\beta_0 + 3\beta_1 - r, \beta_0 + 3\beta_1 + r)) = 0.95$, where B_0, B_1 are our least squares estimates.

We have that,

$$SE(\beta_0 + x\beta_1; x) = \sqrt{\frac{\sigma^2}{n}\left(1 + \left(\frac{x - \overline{x}_n}{\hat{\sigma}_x}\right)^2\right)} \tag{13}$$

From simple calculation, we know that $\overline{x}_n = 25.5$, and we are given that $\sigma^2 = 3^2$ and $x = 3$, so,

$$SE(\beta_0 + 3\beta_1; 3) = \sqrt{\frac{9}{50}\left(1 + \frac{(3 - 25.5)^2}{208.25}\right)} = 0.78586 \tag{14}$$

Then the 95% confidence interval is given by,

$$[\beta_0 + 3\beta_1 - 1.96SE(\beta_0 + 3\beta_1), \beta_0 + 3\beta_1 + 1.96SE(\beta_0 + 3\beta_1)] \tag{15}$$
$$[\beta_0 + 3\beta_1 - 1.5403, \beta_0 + 3\beta_1 + 1.5403] \tag{16}$$

So, this implies that $r = 1.5403$.

(d) **Give a test statistic and rejection region for a test that $\beta_1 \neq 0$ with size 0.01 ($\sigma = 3$ is known).**

The test statistic for a test that $\beta_0 \neq 0$ is given by,

$$T = \frac{0 - \beta_1}{\sqrt{\frac{S^2/n}{\hat{\sigma}_x^2}}} = \frac{-\beta_1}{0.0294} \sim \mathcal{N}(0,1) \tag{17}$$

Where the denominator is simply the standard error calculated in (b). From the standard normal, we know that $2P_T(T \geq t) = 0.01$ implies that $t = 2.576$. Then the rejection region becomes,

$$R = [-\infty, -2.576] \cup [2.576, \infty] \tag{18}$$

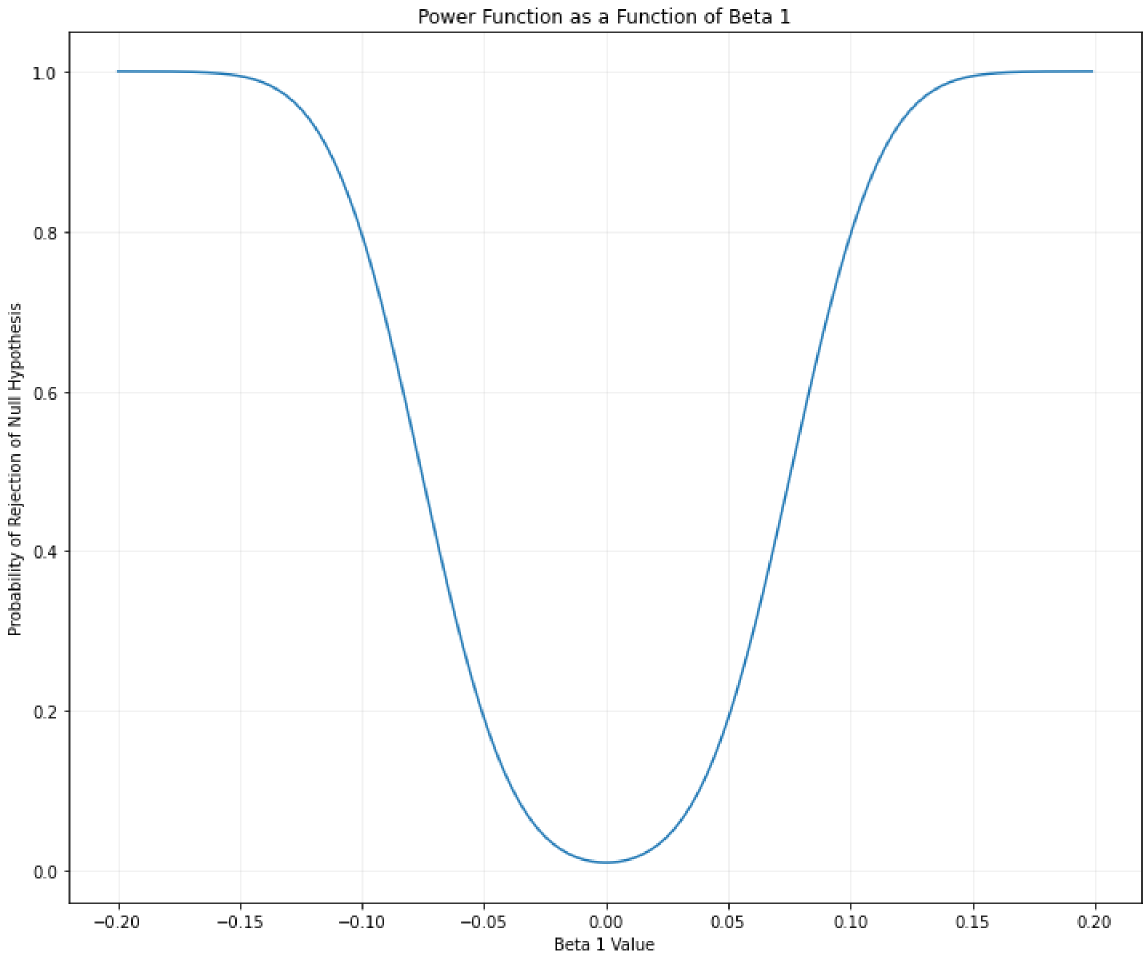
(e) **Plot the power function for your test as a function of β_1. Make sure your plot is large enough to include most interesting values (i.e., it doesn't need to include a ton of values where the power is nearly 1).**

The following code was used to produce the resulting power function of β_1,

```python
from scipy import stats
import matplotlib.pyplot as plt
import numpy as np
```

```python
height = []
for beta1 in np.arange(-0.2,0.2,0.001):
    serr = 0.0294
    T = 2.576*serr
    above = 1 - stats.norm.cdf(T,-1*beta1,serr)
    below = stats.norm.cdf(-1*T,-1*beta1,serr)
    height.append(above+below)
```

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(12,10))
plt.plot(np.arange(-0.2,0.2,0.001),height)
plt.grid(alpha=0.2)
plt.title('Power Function as a Function of Beta 1')
plt.xlabel('Beta 1 Value')
plt.ylabel('Probability of Rejection of Null Hypothesis')
```


Power Function as a Function of Beta 1
Probability of Rejection of Null Hypothesis
Beta 1 Value
1.0
0.8
0.6
0.4
0.2
0.0
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20

3. Data was collected on the proportion of male births in four industrialized countries (i.e., the fraction of male births to total births). The data is contained in *maleprop.csv*.

(a) Fit four simple linear models (one for each country) where the proportion of male births is the response and the year is the input. Report your estimated coefficients for each model.

The following code was used to calculate the estimated coefficients β_0, β_1 for each of the four countries,

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
males = pd.read_csv(r'maleprop.csv')
data = {'USA': [np.array(males['Year'][20:-4]), np.array(males['USA'][20:-4])],
        'Canada': [np.array(males['Year'][20:-4]), np.array(males['Canada'][20:-4])],
        'Denmark': [np.array(males['Year']), np.array(males['Denmark'])],
        'Netherlands': [np.array(males['Year']), np.array(males['Netherlands'])]}
```

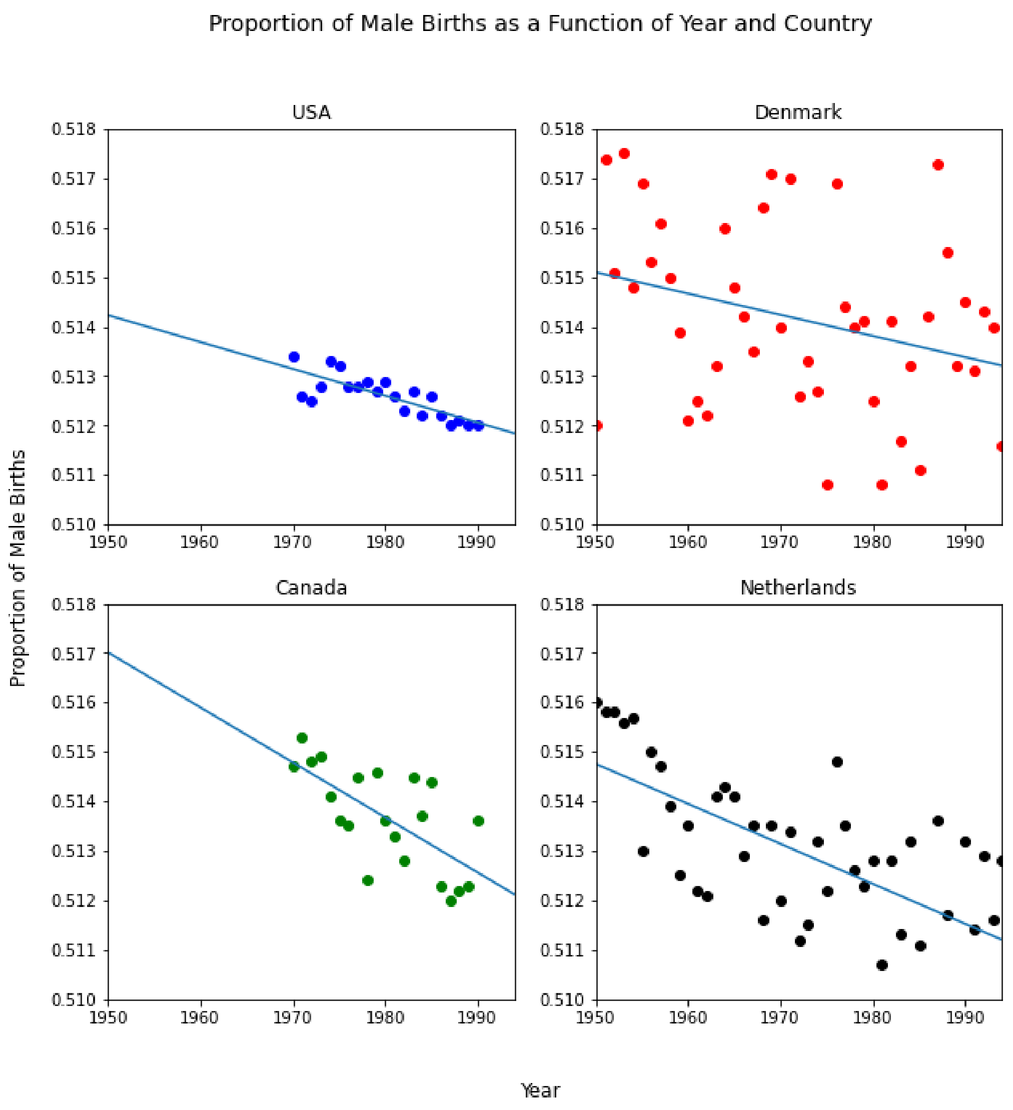
```
def b_1(xs,ys):
    xbar = np.mean(xs)
    ybar = np.mean(ys)
    tottop = 0
    totbot = 0
    for i in range(len(xs)):
        top = xs[i]*(ys[i] - ybar)
        bot = xs[i]*(xs[i] - xbar)
        tottop += top
        totbot += bot
    return(tottop/totbot)
def b_0(xs,ys,b1):
    xbar = np.mean(xs)
    ybar = np.mean(ys)
    return(ybar - b1*xbar)
```

```
b1s = {}
b0s = {}
for name in names:
    b1 = b_1(data[name][0],data[name][1])
    b0 = b_0(data[name][0],data[name][1],b1)
    b1s[name] = b1
    b0s[name] = b0
    print(name, ': ', 'b0: ', round(b0,4), '  b1: ', round(b1,8))
    print('_____________________________')
```

```
USA :  b0:  0.6201   b1:  -5.429e-05
_____________________________
Denmark :  b0:  0.5987   b1:  -4.289e-05
_____________________________
Canada :  b0:  0.7338   b1:  -0.00011117
_____________________________
Netherlands :  b0:  0.6724   b1:  -8.084e-05
_____________________________
```

(b) Plot four scatter plots (one for each country). Overlay your fitted line on each scatter plot.

```
names = ['USA', 'Denmark', 'Canada', 'Netherlands']
color = ['b','r','g','k']

i=0
fig, ax = plt.subplots(nrows=2, ncols=2, figsize =(10,10))
fig.suptitle('Proportion of Male Births as a Function of Year and Country',fontsize=14)
fig.text(0.5, 0.04, 'Year', ha='center',fontsize=12)
fig.text(0.04, 0.5, 'Proportion of Male Births', va='center', rotation='vertical',fontsize=12)
for row in ax:
    for col in row:
        col.scatter(males['Year'],males[names[i]],c=color[i])
        col.set_title(names[i])
        col.set_xlim(1950,1994)
        col.set_ylim(0.51,0.518)
        col.plot(males['Year'], b0s[names[i]]+np.array(males['Year'])*b1s[names[i]])
        i+=1
```



The above subplots display the scatter plots of the proportion of male births as a function of year and country, each being associated with its own linear fit.

(c) For each model, estimate the standard error of the slope coefficient, and give a 95% confidence interval.

Below is the code that was used to estimate the standard error, and provide the 95% confidence intervals,

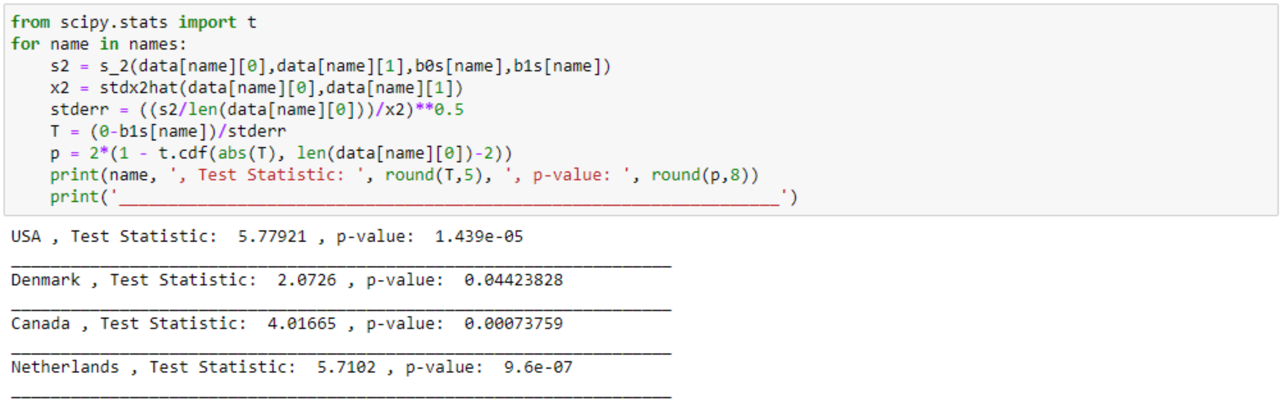
```
def s_2(xs,ys,b0,b1):
    c = 1/(len(xs)-2)
    add = 0
    for i in range(len(xs)):
        a = (ys[i] - b0 - b1*xs[i])**2
        add += a
    return(c*add)

def stdx2hat(xs,ys):
    xbar = np.mean(xs)
    add = 0
    for x in xs:
        add+= (x-xbar)**2
    return(add/len(xs))
```

```
for name in names:
    s2 = s_2(data[name][0],data[name][1],b0s[name],b1s[name])
    x2 = stdx2hat(data[name][0],data[name][1])
    stderr = ((s2/len(data[name][0]))/x2)**0.5
    print(name,', Standard Error: ', stderr)
    print(name, ', 95% Confidence Interval: ', [b1s[name]-(1.96*stderr), b1s[name]+(1.96*stderr)])
    print('___________________________________________________________________')
```

```
USA , Standard Error:  9.393272642713075e-06
USA , 95% Confidence Interval:  [-7.269652866857003e-05, -3.5874899909134775e-05]
___________________________________________________________________
Denmark , Standard Error:  2.0691600324236305e-05
Denmark , 95% Confidence Interval:  [-8.344091212980611e-05, -2.32983885879979e-06]
___________________________________________________________________
Canada , Standard Error:  2.767698308473909e-05
Canada , 95% Confidence Interval:  [-0.00016541571801492166, -5.692194432274444e-05]
___________________________________________________________________
Netherlands , Standard Error:  1.4157695517348635e-05
Netherlands , 95% Confidence Interval:  [-0.00010859229796979835, -5.3094131541791704e-05]
___________________________________________________________________
```

(d) For each model, test if the slope coefficient is non-zero, and report the associated p-values.

Below is the code used to test whether the the slope coefficient, β_1, is non-zero. The associated p-value is provided,



```
from scipy.stats import t
for name in names:
    s2 = s_2(data[name][0],data[name][1],b0s[name],b1s[name])
    x2 = stdx2hat(data[name][0],data[name][1])
    stderr = ((s2/len(data[name][0]))/x2)**0.5
    T = (0-b1s[name])/stderr
    p = 2*(1 - t.cdf(abs(T), len(data[name][0])-2))
    print(name, ', Test Statistic: ', round(T,5), ', p-value: ', round(p,8))
    print('______________________________________________________________________')
```

```
USA , Test Statistic:  5.77921 , p-value:  1.439e-05
______________________________________________________________________
Denmark , Test Statistic:  2.0726 , p-value:  0.04423828
______________________________________________________________________
Canada , Test Statistic:  4.01665 , p-value:  0.00073759
______________________________________________________________________
Netherlands , Test Statistic:  5.7102 , p-value:  9.6e-07
______________________________________________________________________
```

Using a threshold of $\alpha = 0.05$, we find that there is significant evidence in every case to reject the null hypothesis of the slope coefficient being equal to zero.

(e) Why does the US have the largest t-statistic but only the third largest slope (in absolute value)?

The US has the smallest standard error, which therefore inflates the test statistic. The US standard error is small because the residuals are comparatively smaller than the other nations. In other words, the data falls closer to the fitted line, and hence the error between the expected and predicted proportions of male births is lower. Due to this, the standard error is small, and the test statistic is large, because we have more confidence in the fitted line.

(f) Why is the standard error for the estimated slope smaller for the US than Canada?

Again, the data for the US has smaller associated residuals when compared to Canada. In other words, the difference between the projected and actual values for male birth rates is smaller concerning the US data than it is for Canada. This inflates Canada's standard error when compared to the US.

(g) Why does each country appear to have different values for σ^2, the noise term variance? [Hint: Each datapoint is a proportion, i.e., an average.]

Again, σ^2 is a function of the residuals. Since every country has a different fit, and a different spread of data, it is not odd for the sums of the residuals to be different in each case, therefore leading to a different σ^2.

4. **In the (tab-delimited) file *cars.txt* you can find data on a variety of car models.**

   (a) **Assume the data satisfies a simple linear model with MPG as the response and Horsepower as the input. Determine least squares estimates for the coefficients and σ^2.**

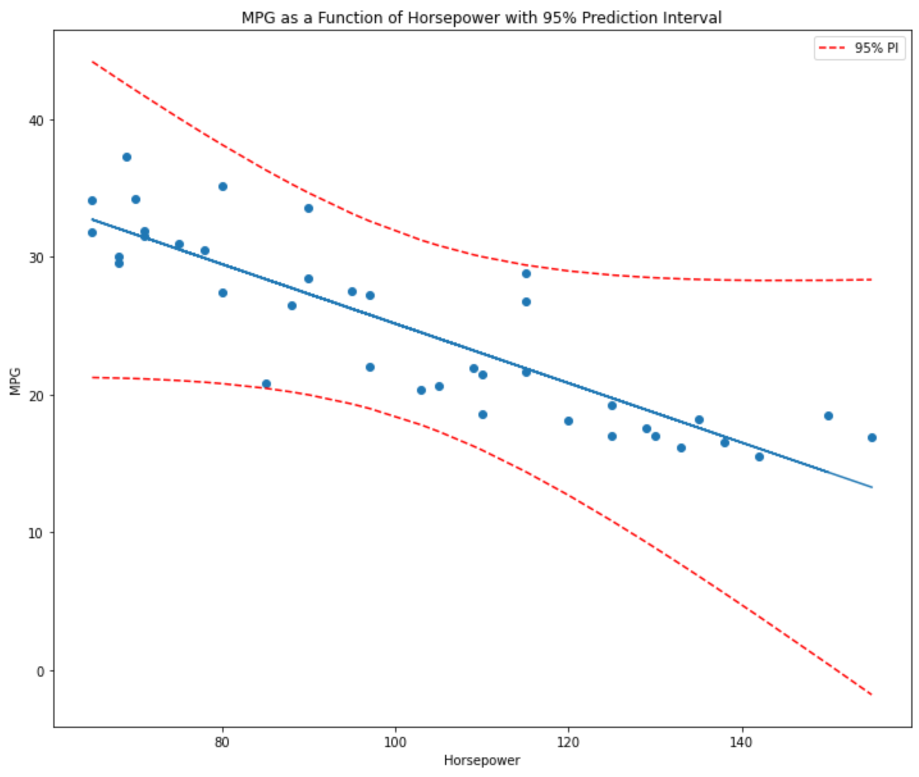
```
cars = pd.read_csv('cars.txt', sep='\t')

ys = cars['MPG']
xs = cars['Horsepower']

b1 = b_1(xs,ys)
b0 = b_0(xs,ys,b1)
s2 = s_2(xs,ys,b0,b1)

print('Value of b0: ', b0)
print('Value of b1: ', b1)
print('Value of s^2: ', s2)
```

```
Value of b0:  46.70664846877335
Value of b1:  -0.2157145995378654
Value of s^2:  10.612159571477124
```

   (b) **Give a scatter plot of the data. For a contiguous range of Horsepower values containing the data, overlay a plot of your fitted line, and the boundaries of a 95% prediction interval.**

```
x2 = stdx2hat(xs,ys)
xbar = np.mean(xs)
lower = []
upper = []
for x in sorted(xs):
    rad = s2*(1+(1/len(xs))+((x-xbar)**2)/x2)
    lower.append(b0+b1*x-(stats.t.ppf(0.975,df=len(xs)-2)*rad**0.5))
    upper.append(b0+b1*x+(stats.t.ppf(0.975,df=len(xs)-2)*rad**0.5))
```

```
plt.figure(figsize=(12,10))
plt.scatter(xs,ys)
plt.plot(xs, b0+ b1*np.array(xs))
plt.plot(sorted(xs), lower, 'r--',label='95% PI')
plt.plot(sorted(xs), upper, 'r--')
plt.title('MPG as a Function of Horsepower with 95% Prediction Interval')
plt.xlabel('Horsepower')
plt.ylabel('MPG')
plt.legend()
```



(c) **Which car has the most leverage (with respect to the fit)?**

The following code was used to determine which car has the most leverage,

```
bs=[]
summer = 0
for x in xs:
    summer += (x-xbar)**2
for x in xs:
    b = ((len(xs)-1)/len(xs))*(x-xbar)/summer
    bs.append(b)
bs, xs = zip(*sorted(zip(bs, xs)))
print('Point with the Most Leverage:   x =',xs[-1])
```

```
Point with the Most Leverage:   x = 155
```

The point at $x = 155$ corresponds to the Buick Estate Wagon, which is the car with the most leverage.

(d) **Estimate 95% confidence intervals for the intercept and slope coefficients using 2000 draws from the parametric bootstrap. To generate a dataset you will resample, for each datapoint, the MPG from a normal**

distribution using the fitted value as the mean, and your estimate of σ_2 as the variance. Then use the bootstrap samples to estimate the confidence intervals.

The following code was used to calculate confidence intervals parametrically, using the same functions to calculate β_0 and β_1 as above.

```
para_b1s = []
para_b0s = []
means=[]

for i in range(2000):
    ynews = []
    for x in xs:
        ynews.append(np.random.normal(b0+x*b1,s2**0.5))
    b1_temp = b_1(xs,ynews)
    b0_temp = b_0(xs,ynews,b1_temp)
    para_b1s.append(b1_temp)
    para_b0s.append(b0_temp)
```

```
L1,U1 = np.quantile(para_b1s, [0.025,0.975])
L0,U0 = np.quantile(para_b0s, [0.025,0.975])
print('95% Confidence Interval for b1: ', (round(L1,5),round(U1,5)))
print('95% Confidence Interval for b0: ', (round(L0,5),round(U0,5)))
```

```
95% Confidence Interval for b1:  (-0.25457, -0.17805)
95% Confidence Interval for b0:  (42.56361, 50.95134)
```