DS-GA 1002 - Homework 8

Eric Niblock

November 3rd, 2020

1. (a) Suppose $X_1, ..., X_n \overset{iid}{\sim} F$, where F has density

$$f(x;\alpha) = \begin{cases} \alpha x^{\alpha-1}, & \textbf{for } x \in [0,1] \\ 0, & \textbf{otherwise} \end{cases} \tag{1}$$

where $\alpha > 0$ is unknown. Determine the MLE for α.

We find that the joint distribution for $x_1, ..., x_n$ is given by,

$$f(x_1, ..., x_n; \alpha) = \prod_{i=1}^{n} \alpha x_i^{\alpha-1} \tag{2}$$

Then, the log-likelihood function is given by,

$$\begin{aligned} \mathcal{L}(\alpha; x_1, ..., x_n) &= \sum_{i=1}^{n} ln(\alpha) + (\alpha - 1) ln(x_i) \\ &= n \cdot ln\alpha + (\alpha - 1) \sum_{i=1}^{n} ln(x_i) \end{aligned} \tag{3}$$

Then, taking the derivative yields,

$$\frac{\partial \mathcal{L}}{\partial \alpha}(\alpha; x_1, ..., x_n) = \frac{n}{\alpha} + \sum_{i=1}^{n} ln(x_i) \tag{4}$$

And setting the derivative equal to zero, we find the maximum value for $\hat{\alpha}$,

$$\hat{\alpha} = -\frac{n}{\sum_{i=1}^{n} ln(x_i)} = -\frac{1}{\overline{ln(X)}} \tag{5}$$

Thus this is the result for the MLE estimate of α.

(b) Let $X_1, ..., X_n \overset{iid}{\sim} \mathcal{N}(\mu, \sigma^2)$. Let τ be the 0.95 percentile, i.e., $P(X < \tau) = 0.95$. Find the MLE for τ . [Hint: Write an expression for τ in terms of μ, σ, and ϕ and apply equivariance.]

First, we have the following relationship,

$$P(X < \tau) = 0.95 = P(\frac{X - \mu}{\sigma} < \frac{\tau - \mu}{\sigma}) \tag{6}$$

And furthermore, we have that on a standard normal distribution ($\mathcal{N}(0, 1)$), the 0.95 percentile is given by $\tau_0 = 1.64485$. Therefore, we have that,

$$\tau_0 = \frac{\tau - \mu}{\sigma} \implies \tau = 1.64485\sigma + \mu \tag{7}$$

And, furthermore,

$$\hat{\tau} = 1.64485\hat{\sigma} + \hat{\mu} \tag{8}$$

Now, the MLE estimation of μ and σ^2 for $X_1, ..., X_n \overset{iid}{\sim} \mathcal{N}(\mu, \sigma^2)$ begins with the joint distribution of $X_1, ..., X_n$:

$$f(x_1, ..., x_n; \mu, \sigma) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x_i - \mu}{\sigma})^2} \tag{9}$$

Then, the log-likelihood equation becomes,

$$\begin{aligned} \mathcal{L}(\mu, \sigma; x_1, ..., x_n) &= \sum_{i=1}^{n} ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{1}{2}\left(\frac{x_i - \mu}{\sigma}\right)^2 \\ &= n \cdot ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2 \end{aligned} \tag{10}$$

Then taking the derivative,

$$\frac{\partial \mathcal{L}}{\partial \mu}(\mu, \sigma; x_1, ..., x_n) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} -2(x_i - \mu) \tag{11}$$

Setting this derivative equal to zero yields,

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{12}$$

Similarly, taking the derivative with respect to σ and using the maximum likelihood estimate $\hat{\mu}$ yields,

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2 \tag{13}$$

Then, using these estimates, it is clear that our estimate for τ becomes,

$$\hat{\tau} = 1.64485 \sqrt{\sum_{i=1}^{n} \frac{(x_i - \hat{\mu})^2}{n}} + \frac{1}{n} \sum_{i=1}^{n} x_i \tag{14}$$

2. A training program claims their teaching methods will improve their students' quantitative GRE scores. They have provided an anonymized list of quantitative GRE scores from all 75 students that completed their most recent course and also attend the nearby university. They were also able to obtain a sample (roughly random) of 100 quantitative GRE scores from students at the nearby university that did not enroll in the training program. The data on these 175 students is found in *gre_data.csv*. They claim that this data provides strong evidence that their program works. We model the training program scores as independent draws from $\mathcal{N}(\mu_T, \sigma^2)$ and the remaining scores as independent draws from $\mathcal{N}(\mu_U, \sigma^2)$.

(a) Compute an unbiased estimate of $\mu_T - \mu_U$ that uses all of the data.

We have the following estimator, S, for $\mu_T - \mu_U$,

$$S = \frac{1}{n}\sum_{i=1}^{n} x_{T_i} - \frac{1}{m}\sum_{j=1}^{m} x_{U_j} \tag{15}$$

Where $\{x_{T_1}, ..., x_{T_n}\}$ represent the n measurements of students in the training program and $\{x_{U_1}, ..., x_{U_m}\}$ represent the m measurements of students not in the training program. We know this estimator to be unbiased since

$$\begin{aligned} E[S] &= \frac{1}{n}E\left[\sum_{i=1}^{n} x_{T_i}\right] - \frac{1}{m}E\left[\sum_{j=1}^{m} x_{U_j}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} E[x_{T_i}] - \frac{1}{m}\sum_{j=1}^{m} E[x_{U_j}] \\ &= \frac{1}{n}\sum_{i=1}^{n} \mu_T - \frac{1}{m}\sum_{j=1}^{m} \mu_U \\ &= \mu_T - \mu_U \end{aligned} \tag{16}$$

Since $E[S] = \mu_T - \mu_U$, we know that S is unbiased. Furthermore, we find that,

$$S = 3.197 \tag{17}$$

(b) Before you reveal your results, your colleague says "your estimate is unbiased, so if it is positive, that means we have strong evidence to conclude the true μ_T is larger than the true μ_U." Assuming the model is correct, give a short explanation why more is needed conclude this.

We have only calculated one value for S, though S itself lies in a distribution with some standard error. If we estimate the standard error, and construct a confidence interval, we could better approach the colleague's statement. However, we ideally need to construct a plethora of confidence intervals in order to gain any real information about the true means.

(c) Compute the standard error and a 95% confidence interval for $\mu_T - \mu_U$ using your estimator.

We will need to know the variance of our estimator, which we find as,

$$\begin{aligned} Var(S) &= Var\left(\frac{1}{n}\sum_{i=1}^{n} x_{T_i}\right) + Var\left(\frac{1}{m}\sum_{j=1}^{m} x_{U_j}\right) \\ &= \frac{1}{n^2}\sum_{i=1}^{n} Var(x_{T_i}) + \frac{1}{m^2} Var(x_{U_j}) \\ &= \frac{nVar(x_{T_i})}{n^2} + \frac{mVar(x_{U_j})}{m^2} \\ &= \frac{Var(x_{T_i})}{n} + \frac{Var(x_{U_j})}{m} \end{aligned} \tag{18}$$

And furthermore, by use of the sample variance, we find that

$$\sigma_S = \sqrt{\frac{Var(x_{T_i})}{n} + \frac{Var(x_{U_j})}{m}} \tag{19}$$

So, the 95% confidence interval becomes,

$$[S - 1.96\sigma_S, S + 1.96\sigma_S] = [0.221, 6.173] \tag{20}$$

(d) Generate 5000 bootstrap estimates of $\mu_T - \mu_U$.

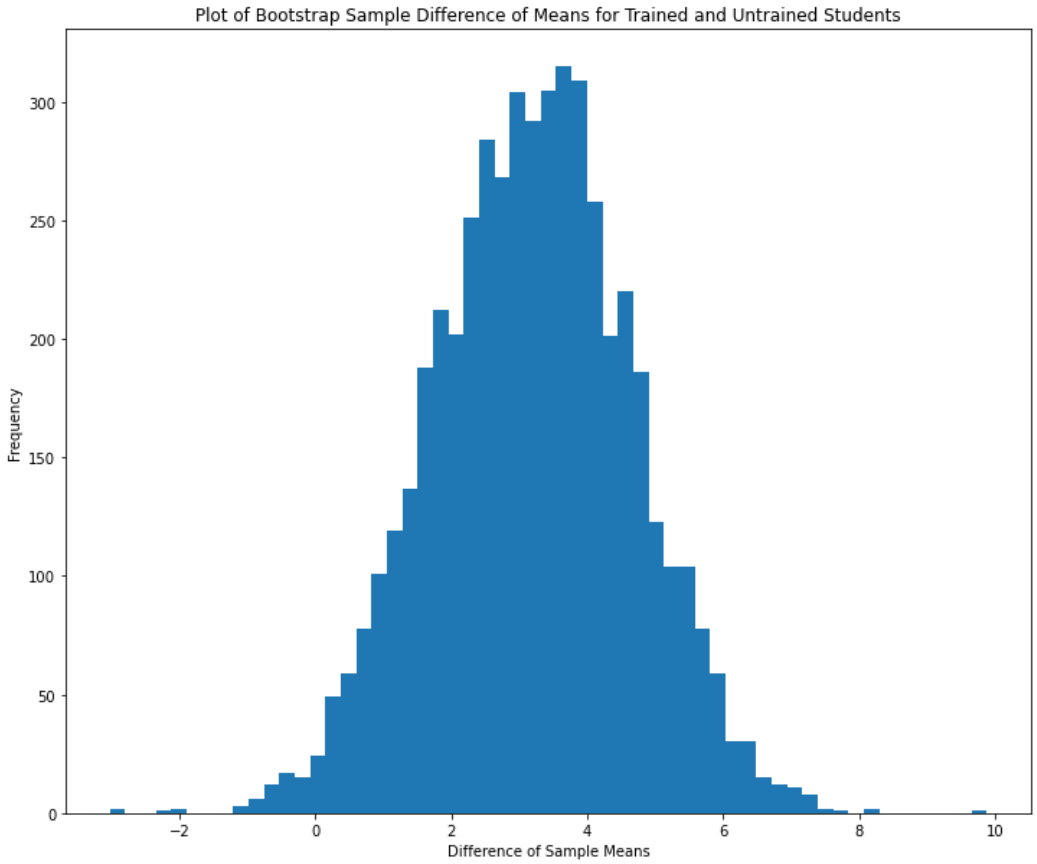
i. **Plot a histogram of your bootstrap estimates. [Hint: Use bins = 'auto' to autoselect the number of bins.]**

ii. **Use your bootstrap estimates to compute a 95% confidence interval for your estimate of $\mu_T - \mu_U$.**

Below, we plotted the histogram of our estimates of $\mu_T - \mu_U$. We also used the following code to produce the confidence interval,

```
import pandas as pd
import numpy as np
A = pd.read_csv(r'gre_data.csv')
nt = np.array(A['no_training'])
t = np.array(A['training'])
t = t[~np.isnan(t)]
T = np.random.choice(t, replace=True, size=(5000,75))
NT = np.random.choice(nt, replace=True, size=(5000,100))
L,U = np.quantile(means_T - means_NT, [0.025,0.975])
L,U
```

```
(0.3496666666666614, 5.966666666666669)
```



So, the resulting 95% confidence interval for $\mu_T - \mu_U$ becomes,

$$[0.350, 5.967] \tag{21}$$

(e) Assuming the model is correct, explain why this data is not conclusive evidence that the training program raises student grades as claimed. [Hint: The confidence interval suggests that μ_T really is larger than μ_U, so that isn't the reason.]

We can never obtain conclusive evidence about the true means (or their difference), unless we somehow have access to the population data, and can calculate them directly.

3. Consider the following data model:

$$X_1, ..., X_{15} \overset{iid}{\sim} \mathcal{N}(\mu, \sigma^2)$$

In this problem we will use simulation to analyze and better understand the joint distribution of $\hat{\mu}$ and $\hat{\sigma}$, the MLEs for μ and σ, respectively.

Simulate 1000 datasets drawn from the model using the values $\mu = 1$ and $\sigma^2 = 4$.

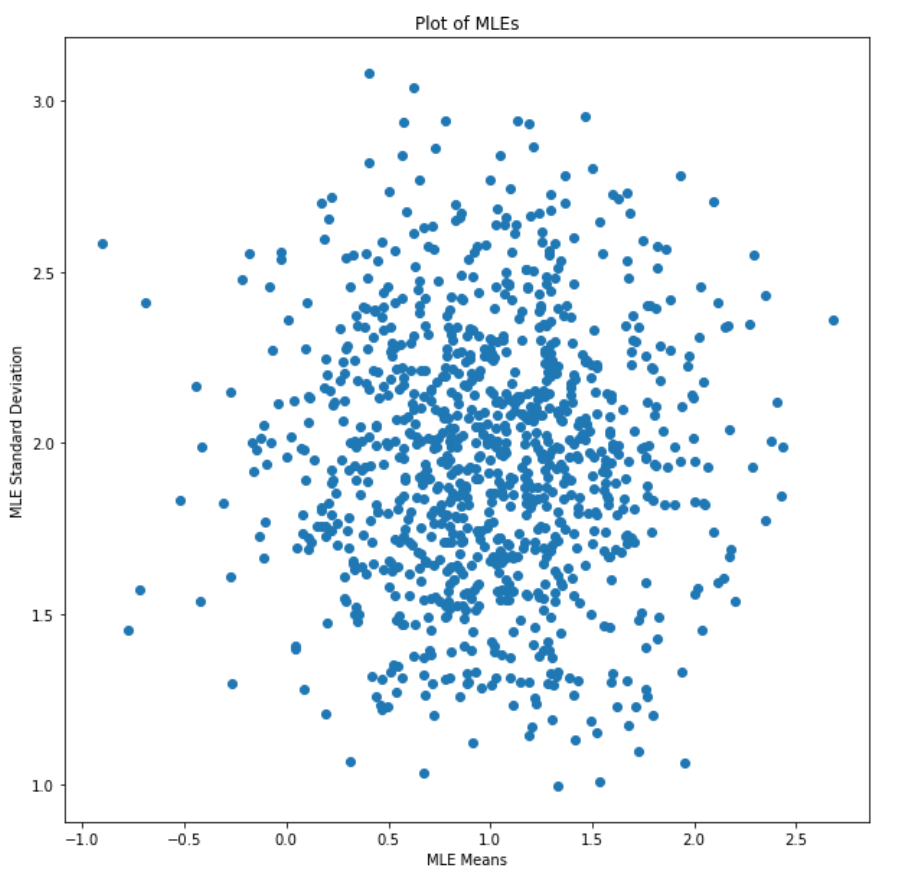
(a) Using each simulated dataset, compute the MLEs for μ and σ. Give a scatter plot of your 1000 MLEs (each pair of MLEs is a single point on your plot).

We follow the previously outlined MLEs for normal distributions given by,

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{22}$$

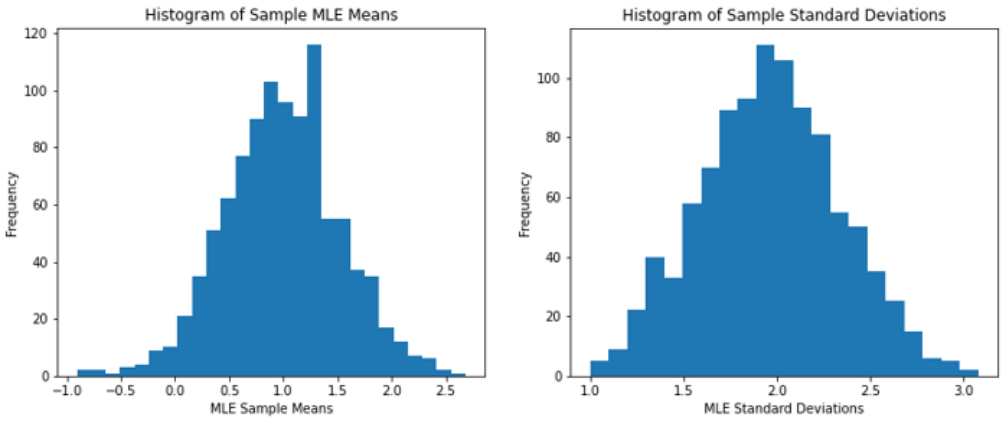
$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2 \tag{23}$$

Where we have instead used the sample standard deviation. Observe the following plot of MLEs,



(b) Plot two histograms of your 1000 MLEs (one for μ and one for σ, with bins ='auto').

Here we plot two histograms concerning the MLE sample means and standard deviations,



(c) Using the 1000 MLE pairs you computed, estimate the covariance matrix for the joint distribution of $\hat{\mu}$ and $\hat{\sigma}$. [Hint: *np.cov.*]

The code below not only provides the covariance matrix, but also shows the data generating procedure used throughout this question. Furthermore, it shows the output of standard errors for the following question,

```
mu, sigma = 1, 2
X = np.random.normal(mu, sigma, size=(1000,15))
M = np.mean(X, axis=1)
SD = np.std(X, axis=1, ddof=1)

C = np.array([list(M), list(SD)])
cov = np.cov(C)

print('Covariance Matrix: ',cov)
print('')
print('Standard Error of the MLE Mean: ', (cov[0,0])**0.5)
print('Standard Error of the MLE STD: ', (cov[1,1])**0.5)
```

```
Covariance Matrix:  [[0.27424044 0.00992081]
 [0.00992081 0.14491724]]

Standard Error of the MLE Mean:  0.5236797082351984
Standard Error of the MLE STD:  0.3806799761285935
```

(d) Estimate the standard errors of $\hat{\mu}$ and $\hat{\sigma}$ using your answer to the previous part.

The diagonals of the covariance matrix are associated with the variance of $\hat{\mu}$ and $\hat{\sigma}$. We then simply find the standard error by taking the square root. We find,

$$\sigma_{\hat{\mu}} = 0.524 \tag{24}$$

$$\sigma_{\hat{\sigma}} = 0.381 \tag{25}$$

(e) Using (only in this part) that our data is drawn from a distribution with variance 4, what is the true standard error of $\hat{\mu}$? [Hint: Check that this value is close to your answer to the previous part (within 0.05 let's say).]

Given that we know the population variance, it is clear that,

$$\sigma_{\hat{\mu}} = \frac{\sigma}{\sqrt{n}} = \frac{2}{\sqrt{15}} = 0.5164 \tag{26}$$

(f) Using a normal-approximation and the standard errors we estimated above (using our 1000 MLEs), what do you expect the widths of the 95% confidence intervals for μ and σ to be? (If your interval is $[a, b]$ the width is $b - a$.)

Since approximately 95% confidence intervals are given by,

$$[\hat{\mu} - 1.96\sigma_{\hat{\mu}}, \hat{\mu} + 1.96\sigma_{\hat{\mu}}] \tag{27}$$

$$[\hat{\sigma} - 1.96\sigma_{\hat{\sigma}}, \hat{\sigma} + 1.96\sigma_{\hat{\sigma}}] \tag{28}$$

We know that the width of these intervals is given by,

$$width(\hat{\mu}) = 3.92\sigma_{\hat{\mu}} = 2.095 \tag{29}$$

$$width(\hat{\sigma}) = 3.92\sigma_{\hat{\sigma}} = 1.523 \tag{30}$$

(g) Use the empirical quantiles of the values in your 1000 pairs to compute 95% confidence intervals for μ and σ. [Hint: Check that the widths here line up with those computed in the preceding part.]

The following code produces the empirical quantiles, and also shows their width,

```
L,U = np.quantile(M, [0.025,0.975])
print('Confidence Interval for u (quantile): ', [L,U])
print('Accompanying Width: ', U-L)
print('')
L,U = np.quantile(SD, [0.025,0.975])
print('Confidence Interval for std (quantile): ', [L,U])
print('Accompanying Width: ', U-L)
```

```
Confidence Interval for u (quantile):  [-0.010533183272434108, 2.0211853408255913]
Accompanying Width:  2.0317185240980256

Confidence Interval for std (quantile):  [1.2858573466886993, 2.7612414299151347]
Accompanying Width:  1.4753840832264353
```

The resulting width of these intervals is very similar to those calculated previously, which serves as a good check.

4. Your firm manufactures devices that can overheat when used in hot temperatures. In *overheat.csv* you are given the data from an experiment conducted by your company. A device is left in a temperature controlled room for 10 minutes, and then several tests are run. The dataset contains the temperatures t_i (in Fahrenheit), and whether a failure occured Y_i (1 for failure, 0 for success). Consider the following model for failure:

$$Y_i \overset{iid}{\sim} Bernoulli\left(\frac{e^{a+bt_i}}{1+e^{a+bt_i}}\right)$$

where $a, b \in \mathbb{R}$ are unknown.

(a) Compute the log-likelihood function for the parameters a, b. Your result should be given in terms of Y_i and t_i (i.e., do not plug in the values from the file). Simplify your expression so that all logarithms that appear have the form

$$log(1 + e^{a+bt_i})$$

The log-likelihood function for Bernoulli random variables $Y_1, ..., Y_n$ is given by,

$$\begin{aligned}
\mathcal{L}(p; y_1, ...y_n) &= \sum_{i=1}^{n} ln(p^{y_i}) + ln((1-p)^{1-y_i}) \\
&= \sum_{i=1}^{n} y_i ln\left(\frac{e^{a+bt_i}}{1+e^{a+bt_i}}\right) + (1-y_i) ln\left(1 - \frac{e^{a+bt_i}}{1+e^{a+bt_i}}\right) \\
&= \sum_{i=1}^{n} y_i \left(ln(e^{a+bt_i}) - ln(1+e^{a+bt_i})\right) + (1-y_i) ln\left(\frac{1}{1+e^{a+bt_i}}\right) \\
&= \sum_{i=1}^{n} y_i(a+bt_i) - log(1+e^{a+bt_i})
\end{aligned} \tag{31}$$

(b) Using a numerical optimization library, compute the MLEs $\hat{a}, \hat{b}$ for a, b using the data in the file. [Hint: Use *np.logaddexp* to evaluate $log(1+e^{a+bt_i})$, and use *scipy.optimize.minimize* with method = 'Nelder-Mead' and $x_0 = [0, 0]$ along with the appropriate function to optimize.]

The following code was used to approximate $\hat{a}$ and $\hat{b}$ by numerical minimization methods. The results are given as a print out at the end of the code.

```
from scipy.optimize import minimize
import pandas as pd
import numpy as np

O = pd.read_csv(r'overheat.csv')
temp = np.array(O['Temperature'])
failure = np.array(O['Failure'])
```

```
def loglik(param, failure, temp):
    a=param[0]
    b=param[1]
    tot = 0
    for i in range(len(failure)):
        tot+= (failure[i]*(a+(b*temp[i]))) - np.logaddexp(0, a+b*temp[i])
    return(-1*tot)
```

```
res = minimize(loglik, [0,0], method='nelder-mead', args = (failure,temp),
               options={'maxiter':1000})
print('Result for a: ', round(res.x[0],4))
print('Result for b: ', round(res.x[1],4))
```

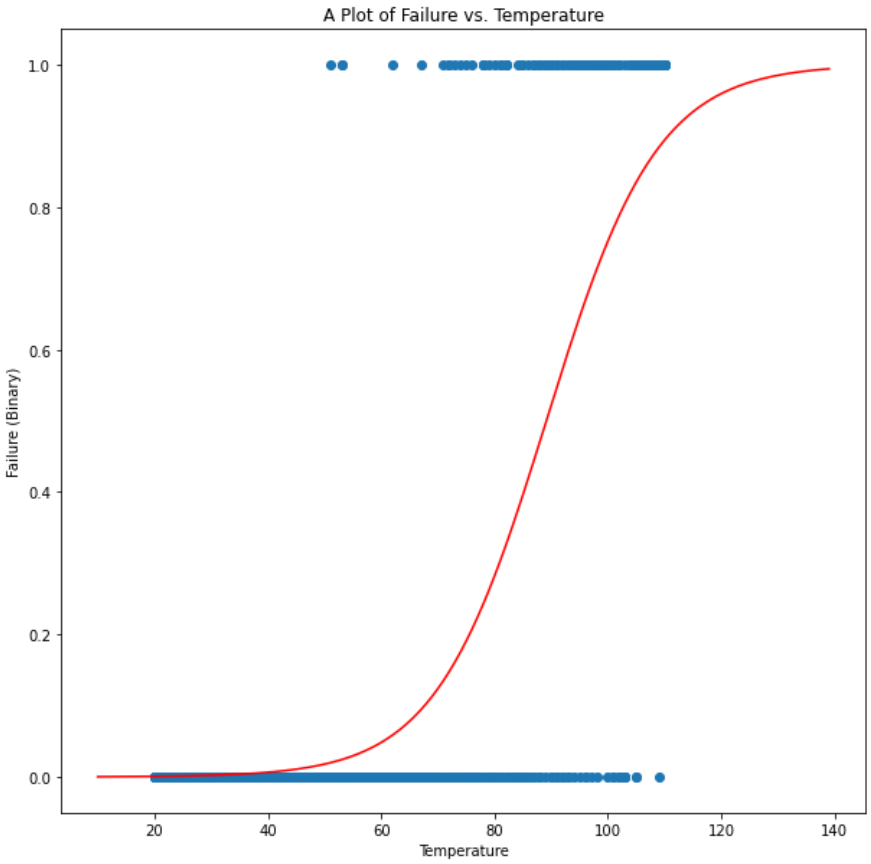
```
Result for a:  -9.1113
Result for b:  0.1022
```

(c) Plot the data as a scatter plot with temperature on the x-axis and whether failure occured on the y-axis. In the same figure, include a plot of the function

$$g(x) = \frac{e^{\hat{a}+\hat{b}x}}{1+e^{\hat{a}+\hat{b}x}}$$

Here we provide the code used to generate a plot of $g(x)$ as well as the scatter plot of the failure (binary) as a function of temperature.

```
import scipy
import matplotlib.pyplot as plt
ys = []
a = [-9.11129157,  0.10221834][0]
b = [-9.11129157,  0.10221834][1]
for i in list(range(10,140)):
    ys.append(scipy.special.expit(a+b*i))
plt.figure(figsize=(10,10))
plt.title('A Plot of Failure vs. Temperature')
plt.xlabel('Temperature')
plt.ylabel('Failure (Binary)')
plt.plot(list(range(10,140)),ys, c='r')
plt.scatter(temp, failure)
```



(d) **We are going to investigate the sampling distribution of our MLEs using the bootstrap.**

Generate 200 new datasets of size 455 by sampling 455 (t_i, Y_i) pairs with replacement from the given data. [Hint: You can sample indices from *range(455)* and then subselect a numpy array or use *pandas.DataFrame.sample* with *frac=1* and *replace=True*.]

i. **For each of the 200 generated datasets, estimate new MLEs for a, b. Create a scatter plot with all of your 200 estimates (one point for each pair of values).**
ii. **Use your 200 MLEs to compute a 95% bootstrap confidence interval for a.**
iii. **Use your 200 MLEs to compute a 95% bootstrap confidence interval for b.**

Below is the code used to generate the bootstrap and return the resulting confidence intervals for a and b.

```
indices = list(range(455))
a_list = []
b_list = []
for time in range(200):
    choice = np.random.choice(indices, 455, replace=True)
    samp_T = temp[choice]
    samp_F = failure[choice]
    res = minimize(loglik, [0,0], method='nelder-mead',args = (samp_F, samp_T),
               options={'maxiter':1000})
    a_list.append(res.x[0])
    b_list.append(res.x[1])
```

```
L,U = np.quantile(a_list, [0.025,0.975])
print('Confidence Interval for a: ', [L,U])
L,U = np.quantile(b_list, [0.025,0.975])
print('Confidence Interval for b: ', [L,U])
```

```
Confidence Interval for a:  [-11.262460581638507, -7.797749377632927]
Confidence Interval for b:  [0.0870030983106797, 0.12671843203350622]
```