MATH2715 R Report

Eleanor Sleightholm

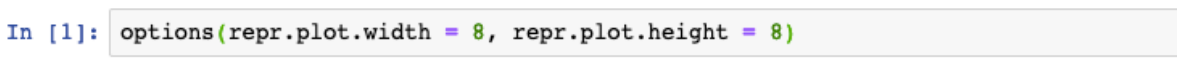
SID: 201214895

QUESTION 1:

In Question One we are asked to show that the sample mean of a Poisson Distribution with lambda equal to 2 approaches a Normal Distribution, for a large sample size n. Essentially, we are trying to prove the Central Limit Theorem for the Poisson Distribution with lambda equal to 2.

In order to answer this question, we must first write code in R that compares the Poisson Sample Mean Distribution to the Normal Distribution. This will allow us to graphically prove the Central Limit Theorem and observe whether our Poisson Distribution approaches the Normal Distribution. Throughout, I will be using code I have inputted from R Jupyter Notebooks.

Since we are dealing with large sample sizes, we must ensure our graph is easy to interpret and has a plotting area of suitable size. Using 'repr.plot.width' and 'repr.plot.height' sets the width and height of the plot in inches. Therefore, we input the following command:



```
In [1]: options(repr.plot.width = 8, repr.plot.height = 8)
```

Following on from this, we then create a function, **outputplotCLT**, which produces a comparison plot between the Poisson Sample Mean Distribution and the Normal Distribution. We start by defining the function, the number of sample means and the sample size which, in this case, is an inputted value of the reader's choice. We then use the command 'cases' so we can sample according to the selected distribution – the Poisson Distribution with lambda equal to 2.

In the code, we then define a matrix with the number of rows corresponding to the number of sample means and number of columns corresponding to the sample size. We then calculate the mean and standard deviation for the approximate normal. For a Poisson Distribution with parameter lambda, we know that the Sample Mean Distribution gives a mean of lambda and variance of lambda divided by n. In summary,
The sample mean is a discrete random variable given by

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The mean and variance of this distribution are:

$$\mu_{\overline{x}} = \lambda, \quad \sigma^2_{\overline{x}} = \frac{\lambda}{n}$$

So, for our distribution with lambda equal to 2, our mean is 2 and variance is sqrt(2/n) as defined in our code below.

Finally, we input code that calculates the mean for our sample size, finds the fit for the Normal Distribution and then plots our Poisson Distribution against the Normal for our particular sample size, n. All of the above condenses into the following, simple code:

For convenience, we have written some information about which code does what in hashtags.

```
In [2]: outputplotCLT <- function(input){
          #define number of sample means
          samp = 1000
          #define sample size
          n = input$rvs
          #in the following we sample according to the selected distribution
          #the number of rows is the number of sample means,
          #the number of columns is the sample size
          #calculate the mean and sd of the approximate normal in both cases
         cases=c("[1] Pois(2)")
          if(input$select == 1){
            matrix = matrix(rpois(n*samp, 2), nrow = samp, ncol = n)
            mu = 2
            sd = sqrt(2)/sqrt(n)
          }

          #calculate the means from each sample
          means = rowMeans(matrix)
          #find the fit for the normal (estimate parameters)
          xfit = seq(from = min(means), to = max(means), by = .001)
          yfit = dnorm(xfit, mean = mu, sd = sd)
          #plot means, overlay Normal distribution
          plot(density(means), main = "", xlab = "",
               ylab = "Density", col = rgb(0, 1, 0, 3/4),
               lwd = 8, ylim = c(0, max(density(means)$y, yfit)))
          lines(xfit, yfit, col = rgb(0, 0, 1, 3/4), lwd = 8)
          title(main = paste(c("n=",input$rvs," Case=",cases[input$select]),collapse=''))
              legend("topright", legend = c("Sample Mean Density", "Estimated Normal Density"),
                 lty=c(1,1), lwd=c(2.5,2.5),
                 col=c(rgb(0, 1, 0, 3/4), rgb(0, 0, 1, 3/4)))
        }
```

Since we have used the code ‘cases’ we must then write a command that allows the reader to choose our Poisson Distribution by inputting the number 1. We do this as follows:

```
In [*]: select=as.integer(readline(prompt=paste(c("Enter 1 for Pois(2)\n"),collapse='')))

        Enter 1 for Pois(2)
```

After selecting the Poisson Distribution, we must now set up a command that allows the reader to choose a particular sample size. We do this as follows:

```
In [*]: rvs=as.integer(readline(prompt = "Enter a sample size\n"))

        Enter a sample size
```

Finally, we need code to specify the input and to plot the result. The following code allows us to plot a graph of the Poisson Sample Mean Distribution against the Normal Distribution for our inputted sample size value.

```
In [7]: input=NULL
        input$select=select
        input$rvs=rvs

        outputplotCLT(input)
```

Assume we choose a sample size of 10. We can use the following code to produce 10 separate graphs for n = 1 to n = 10. For each sample size we choose, we can input the value as n into the ‘seq(1,n,1)’ command. This will plot n comparison graphs which we can then observe.

```
In [9]: input=NULL
        input$select=1
        for (i in seq(1,10,1)){
            input$rvs=i
            outputplotCLT(input)
        }
```

Now we have sufficient, working code, we can observe whether the Poisson Sample Mean Distribution approaches that of a Normal Distribution. Since we are observing this as n approaches a large value, we will alter the sample sizes of n and note the outcome. First, we will enter sample sizes of n = 1, n = 10 and n = 100. Doing so, we get the following graphs:

Comparison between the Sample Mean of Poisson Distribution with the Normal Distribution with n = 1, n = 10 & n = 100

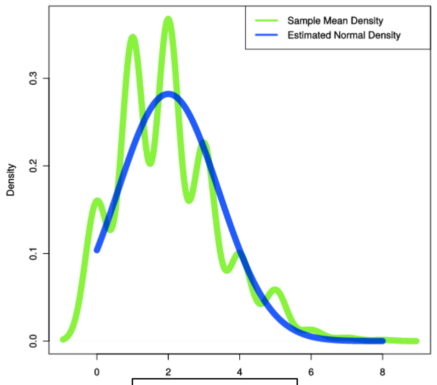


Fig 1, n = 1

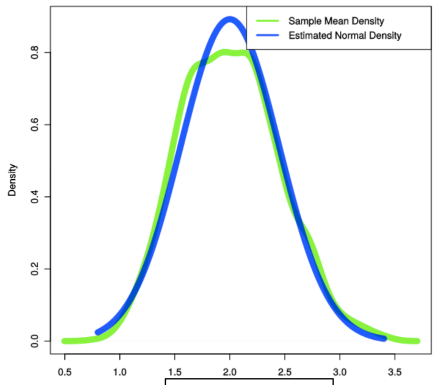


Fig 2, n = 10

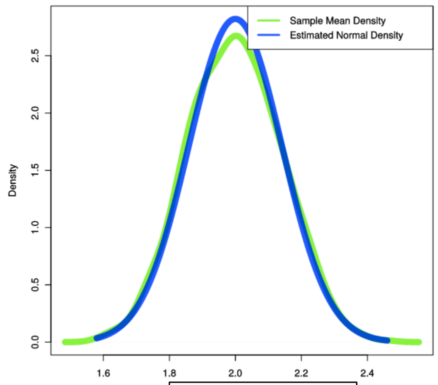


Fig 3, n = 100

The blue graph is the Estimated Normal Density and the green graph is the Poisson Sample Mean Density as denoted by the key in the upper right corner. Immediately, we can see the Poisson Sample Mean Density gradually resembles that of the Estimated Normal Density as the value of our sample size increases from n = 1 (Fig 1) to n = 10 (Fig 2) to n = 100 (Fig 3).

By observing our graphs, we can say for definite that with n=1, the Sample Mean of Pois(2) does *not* approach the Normal Distribution. The irregularity of the green graph along with the steep peaks and troughs in comparison to the symmetric Normal graph allows us to see that the two distributions differ greatly. This confirms our belief that for n = 1, the Sample Mean of Pois(2) does not approach the Normal Distribution.

We now observe the graph for a slight increase in the sample size with n=10. Immediately, we can see a great different between n = 1 (Fig 1) and n=10 (Fig 2). The graphs overlap much more considerably, and the Poisson Sample Mean Density is lacking in peaks and troughs. It appears symmetric and resembles the Estimated Normal Density better than in Fig 1. From this, we see that a slight increase in the sample size has given the Poisson Sample Mean Distribution a distribution that slightly resembles that of the Normal.

For our final graph, Fig 3, we see a further increase in the similarity between the Poisson Sample Mean and Normal Distribution. The two graphs appear to be almost identical, with the Poisson having a lower peak. Comparing to Fig 1 we see a great improvement in the symmetricity of the distribution and so, we can conclude that increasing the sample size allows the Poisson Sample Mean with lambda equal to 2 to approach a Normal Distribution.

When observing our graphs, we must consider the Central Limit Theorem (CLT). The Central Limit Theorem states that for large n, the distribution of the sample mean approaches a Normal distribution. This is a very powerful result because it holds no matter what the distribution of the underlying random variables is. Therefore, we know that eventually, our sample mean will approach a Normal distribution but for *large* n. Therefore, before we can completely arrive at a conclusion, we must consider *larger* values of n.

The largest value we have considered is n = 100. Therefore, to improve the results of our findings, and confirm the CLT, we will create a sequence of plots to further justify our conclusion. We will use the ln[9] command as seen above but with the sequence from 1 to 100,000 that increases by a sample size of 20,000 between each plot (so *i in seq(1, 100000, 20000)*). This will produce 5 plots as follows:

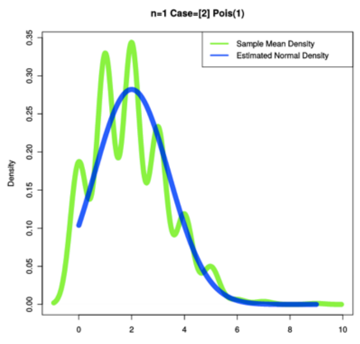


Fig 4. n = 1

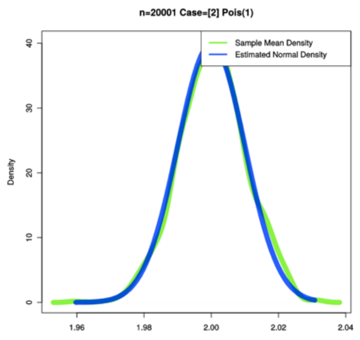


Fig 5. n = 20001

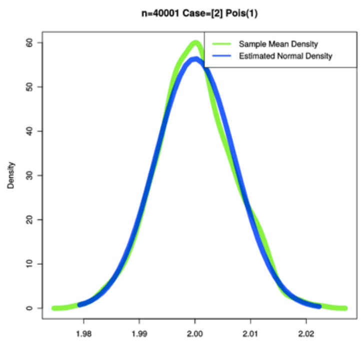


Fig 6. n = 40001

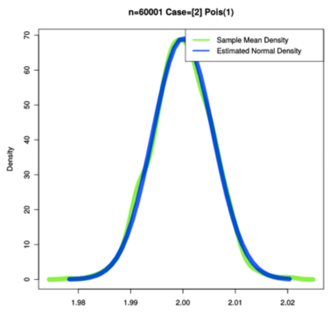


Fig 7. n = 60001

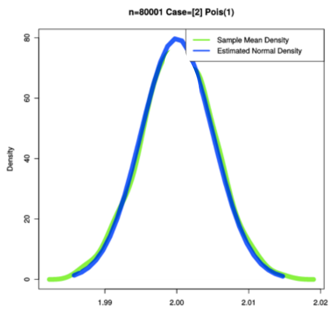


Fig 8. n = 80001

We can see that, as the sample size increases, the Sample Mean Density plot resembles that of the Normal Distribution. The sequence of plots above confirms both the Central Limit Theorem our belief so we can conclude that the sample mean of a Poisson Distribution with lambda equal to 2 approaches a Normal Distribution, for a large sample size n.

To conclude, we have observed through data and graphs that the sample mean of a Poisson Distribution with lambda equal to 2 approaches a Normal Distribution as you increase the sample size. We have been able to observe this through increasing n values, considering the Central Limit Theorem and plotting the corresponding graphs against that of a Normal Distribution.

QUESTION 2:

In order to implement the Maximum Likelihood Estimator for the exponential distribution in R we must follow a two-step process. The first step is to declare the log-likelihood function and the second step is to optimise the log-likelihood function which is done in terms of a particular data set. Since we are considering the exponential distribution, we must consider the log-likelihood function for this distribution. Therefore, the log-likelihood function for the exponential distribution is defined as follows:

$$L(\mu) = nln(\mu) - \mu \sum_i y_i$$

We can now program this function with the following syntax:

```
In [1]: exponential.lik<-function(y,par){
            n<-length(y)
            theta = par
            logl<-n*log(theta)-theta*sum(y)
            return(-logl)
        }
```

In this code, exponential.lik is the name of the log-likelihood function for the exponential distribution which we will later use when optimising the function. Furthermore, the unknown parameter is theta since the exponential distribution only has one parameter. This needs to be estimated and, in this case, y is a placeholder for the data. Since the log-likelihood function requires knowledge of the sample size, we can obtain this by using the command 'n<-length(y)'. Finally, we must ask R to return -1 times the log-likelihood function since the R function 'optim' which we will use later minimises rather than maximises and we are wanting the function to be maximised.

Now we have declared the exponential.lik function, we can now invoke the 'optim' command. When using the optim command we must be aware of the specification for this command. It must include a vector of starting values, the log-likelihood function we created and data which declares the data for the estimation. So, we begin by defining our data:

```
In [2]: data=rexp(100)
        data
        length(data)
```

Here, we use the 'rexp' command with n = 100 to generate 100 random values from the exponential distribution. We can call back the data to observe the generated values and check that there are 100 of them by using the 'length(data)' command. Now we have our data and log-likelihood function we can use the 'optim' command. Below, 1 is the starting value for the algorithm. Furthermore, since the log-likelihood function refers to generic data objects as y, it is important that the vector data is equated with y. Hence, we construct the optim command to look as follows:

```
In [3]: resExp = optim(par=1,exponential.lik,y=data,method="BFGS")
        resExp
```

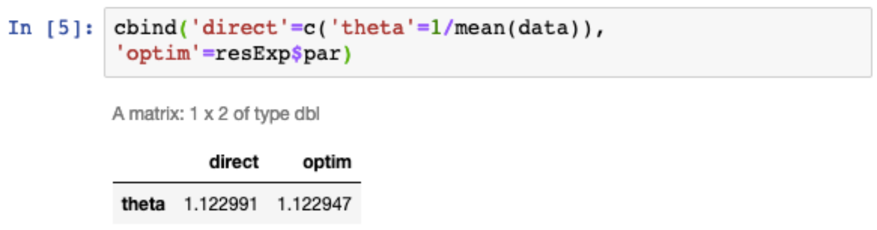
```
$par
1.12294695023081
$value
88.4004101267611
$counts
            function   11
            gradient   4

$convergence
0
$message
NULL
```

We decided to call back 'resExp' to observe what output we get. Above, '$par' gives the best set of parameters found (our optimal value) and '$value' gives the corresponding value of our function, 'exponential.lik', to '$par'. Furthermore, '$counts' gives the number of calls to our function and gradient respectively.

Next, we must compare our optimal value found above with our direct estimation for the exponential distribution. Since we are working with the exponential, we know that our direct estimation, theta, will be 1 divided by the mean data. For this reason, we input the following command:

```
In [4]: theta = 1/mean(data)
```

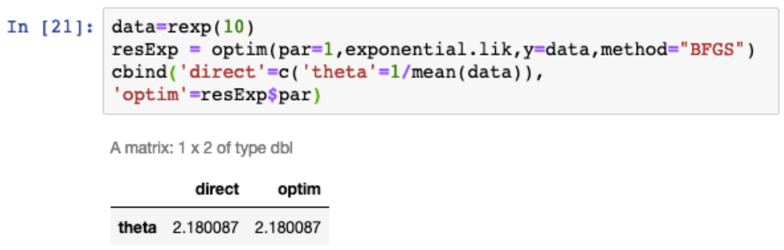
We can now form a table in R to compare the direct estimation, theta, with the optimal value we found earlier. To do this, we input the following command:



```
In [5]: cbind('direct'=c('theta'=1/mean(data)),
        'optim'=resExp$par)
```

```
A matrix: 1 x 2 of type dbl

         direct    optim
theta  1.122991  1.122947
```

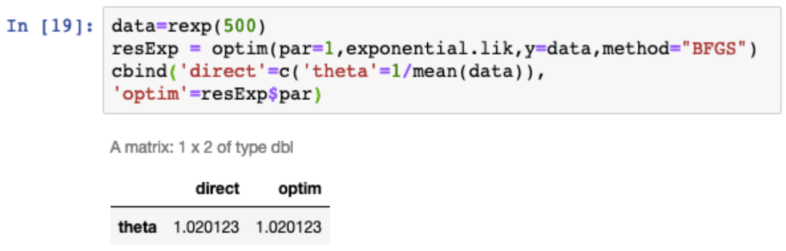
From this table we see that there is little difference between the direct estimation and the optimal value. In fact there is a 0.000044 difference between these values (calculated by the command 'theta – resExp$par'). Before we arrive at a final conclusion, we must test different data ranges and observe the Maximum Likelihood Estimator in each of these cases. We coded this as follows:



```
In [21]: data=rexp(10)
         resExp = optim(par=1,exponential.lik,y=data,method="BFGS")
         cbind('direct'=c('theta'=1/mean(data)),
         'optim'=resExp$par)
```

```
A matrix: 1 x 2 of type dbl

         direct    optim
theta  2.180087  2.180087
```

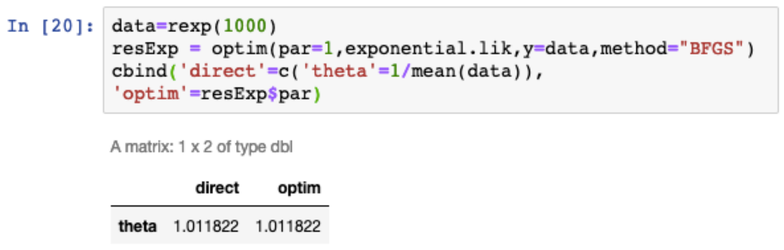
10 Data Values



```
In [19]: data=rexp(500)
         resExp = optim(par=1,exponential.lik,y=data,method="BFGS")
         cbind('direct'=c('theta'=1/mean(data)),
         'optim'=resExp$par)
```

```
A matrix: 1 x 2 of type dbl

         direct    optim
theta  1.020123  1.020123
```

500 Data Values



```
In [20]: data=rexp(1000)
         resExp = optim(par=1,exponential.lik,y=data,method="BFGS")
         cbind('direct'=c('theta'=1/mean(data)),
         'optim'=resExp$par)
```

```
A matrix: 1 x 2 of type dbl

         direct    optim
theta  1.011822  1.011822
```

1000 Data Values

In each of the tables we see that the direct estimation of the mean is identical to the optimal value that we have found. Therefore, we can conclude that the average of our exponential distribution sample is also the value of our Maximum Likelihood Estimator. We can observe that MLE estimators often lead to intuitive estimators, as seen in our example.

MLE theory states that, under certain conditions, the MLE will be approximately the best estimator. We can therefore say that, since the MLE and direct estimation of the mean are identical, this is an exceptionally good estimator and is very accurate. We are able to use MLE in order to obtain more robust parameter estimates.

To conclude, we have observed that implementing the MLE for the exponential distribution parameter allows us to observe that the MLE is identical to the direct estimation for the mean. Thus, the direct estimator of the mean for the exponential distribution is very accurate and is a suitable parameter estimate.