# *F21SA - Statistical Modeling and Analysis - Coursework Report - Aisha Aijaz, H00305254*

## I. INTRODUCTION

In this project, I have tackled a real-world statistical problem using the R programming language. The data has been summarized and the proposed predictions are realized based on probability. Some useful calculations were made to further implement answering questions about the statistical model. Robustness was also verified with the error possibilities in the value of the estimator. We found that our model is highly sensitive to erroneous constants and thus it will be affected by inconsistent values.

## II. MATHEMATICAL CALCULATIONS

***Likelihood Function***

We begin by calculating the likelihood function. For this, we first identify the function which we are going to be working with. Then we have to create a product function for all the data points in our datasets. As specified, the number of records (N) is 458. After this, we will have to calculate the Probability function where $X_i$ is the predicted instance and $x_i$ is the actual occurring instance.

$$f(x,\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \mathrm{x}_1{}^{-(\alpha+1)} \exp\left\{-\frac{\beta}{\mathrm{x}}\right\}$$

$$f(x_1,\, x_2,\, x_3, \ldots, x_{458}) =$$

$$\frac{\beta^\alpha}{\Gamma(\alpha)} \mathrm{x}_1{}^{-(\alpha+1)} \exp\left\{-\frac{\beta}{\mathrm{x}_1}\right\} \times \frac{\beta^\alpha}{\Gamma(\alpha)} \mathrm{x}_2{}^{-(\alpha+1)} \exp\left\{-\frac{\beta}{\mathrm{x}_2}\right\} \times \ldots \times \frac{\beta^\alpha}{\Gamma(\alpha)} \mathrm{x}_{458}{}^{-(\alpha+1)} \exp\left\{-\frac{\beta}{\mathrm{x}_{458}}\right\}$$

$$= \prod_{i=1}^{458} \frac{\beta^\alpha}{\Gamma(\alpha)} x_i{}^{-(\alpha+1)} \exp\left\{-\frac{\beta}{\mathrm{x_i}}\right\}$$

$$P(X_1 = x_1,\, X_2 = x_2,\, X_3 = x_3, \ldots, X_N = x_N) \;=\; \prod_{i=1}^{N} \frac{\beta^\alpha}{\Gamma(\alpha)} x_i{}^{-(\alpha+1)} \exp\left\{-\frac{\beta}{\mathrm{x_i}}\right\} \;=\; \boldsymbol{L}$$

Here ***L*** is the Likelihood function that we will use in the next step.

***Log-Likelihood Function***

To find the maximum likelihood, it is imperative that the Likelihood function is derived with respect to $\beta$ while the result is equated to 0. However, finding this partial derivative for a product function can be very complex. So, to decrease the time and effort into the maximize step, we take the log of *L* to convert the product function to a summation function instead. This will allow the derivation step to become much simpler and would not affect our required result.

$$\boldsymbol{l} = \log(L)$$

$$= \log\left(\prod_{i=1}^{N} \frac{\beta^\alpha}{\Gamma(\alpha)} x_i{}^{-(\alpha+1)} \exp\left\{-\frac{\beta}{\mathrm{x_i}}\right\}\right)$$

$$= \sum_{i=1}^{N} \log\left(\frac{\beta^\alpha}{\Gamma(\alpha)} x_i{}^{-(\alpha+1)} \exp\left\{-\frac{\beta}{\mathrm{x_i}}\right\}\right)$$

$$= N\left(\alpha\log(\beta) - \log\left(\Gamma(\alpha)\right) - (\alpha+1)\sum_{i=1}^{N}\log(x_i) - \beta\left(\sum_{i=1}^{N}\frac{1}{x_i}\right)\right)$$

***Maximize the function (apply derivative and equate to zero)***

Now, we have to take the derivative of $l(\beta)$ and then equate it to zero to find the unknown parameter. In our case, it will be $\hat{\beta}$ which it will now be called. It is the estimator for our required function and will be useful to us in the next steps and the overall market analysis.

$$\frac{\partial l(\beta)}{\partial \beta} = \frac{\alpha}{\beta} - \sum_{i=1}^{N} \frac{1}{x_i} = 0$$

$$\frac{\alpha}{\hat{\beta}} = \sum_{i=1}^{N} \frac{1}{x_i}$$

$$\boldsymbol{\hat{\beta}} = \frac{N\alpha}{\sum_{i=1}^{N} \frac{1}{x_i}}$$

***Fisher Information for $\boldsymbol{\hat{\beta}}$***

In simplest terms, the Fisher information tells us what the expected score of the observed information is. For this, we have calculated first the unbiased estimator $U$ which is just the derivative of our initial estimator. Then we take a second derivative of this with respect to $\beta$ to get the expected value of observed data. This will give us the Fisher's Information.

$$U(\hat{\beta}) = \frac{\partial l(\hat{\beta})}{\partial \hat{\beta}}$$

$$U(\hat{\beta}) = N\left[\frac{\alpha}{\beta} - \sum_{i=1}^{N} \frac{1}{x_i}\right]$$

$$\frac{\partial\left(U(\hat{\beta})\right)}{\partial \hat{\beta}} = N\alpha \frac{\partial(\hat{\beta}^{-1})}{\partial \hat{\beta}}$$

$$= \frac{-N\alpha}{\hat{\beta}^2}$$

$$\boldsymbol{I(\hat{\beta})} = E\left(-\frac{\partial\left(U(\hat{\beta})\right)}{\partial \hat{\beta}}\right)$$

$$= E\left(\frac{N\alpha}{\hat{\beta}^2}\right)$$

$$= \frac{N\alpha}{\hat{\beta}^2}$$

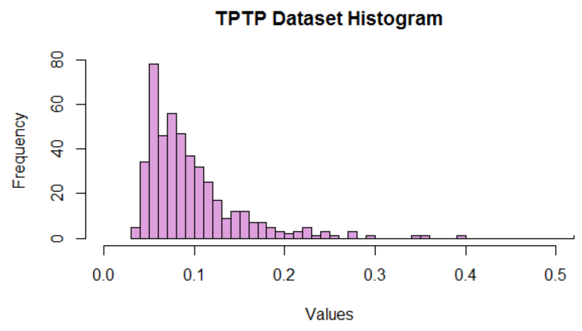
***Approximation of the distribution of $\boldsymbol{\hat{\beta}}$***

The approximation of the normal distribution will be specified by two parameters, the mean (μ) and the standard deviation (σ). We know that the standard deviation is the inverse of Fisher's Information, so the final result will be as follows.

$$\hat{\beta} \sim N\left(\beta, \frac{1}{I(\beta)}\right)$$

$$\boldsymbol{\hat{\beta} \sim N\left(\beta, \frac{\beta^2}{N\alpha}\right)}$$

## III. RESULTS AND CONCLUSIONS



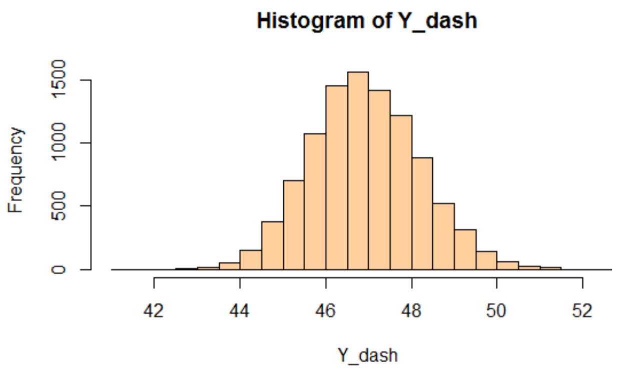
This is the frequency Histogram for the given TPTP dataset *(only the most occurring bins are shown when breaks = 100)*. As we can see, most of the values lie between the ranges of 0 to 0.2 Million £.

A statistical summary of the prices is also shown *(see table below)*. *The mean and median vary quite a bit. The median is lower than the mean which tells us that the data itself is skewed towards the right and is thus not symmetrical.*

This is a general summary and both the dataset histogram and their summary was created using a simple code *(this is a sample of the code, for further perusal see appendix)*.

```
#print the statistical summary
print(summary(myData))
#calculate betaCap
betaCap = (n * alpha) / denom
#calculate confidence interval
beta_L = betaCap - 1.96 * (betaCap / sqrt(n*alpha))
beta_U = betaCap + 1.96 * (betaCap / sqrt(n*alpha))
```

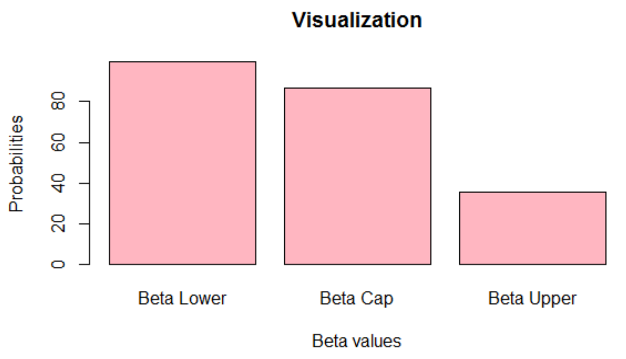
| | |
|---|---|
| **Minimum** | 0.03096 |
| **1st Quartile** | 0.05944 |
| **Median** | 0.08179 |
| **Mean** | 0.10133 |
| **3rd Quartile** | 0.11305 |
| **Maximum** | 1.31238 |



The histogram alongside shows the estimation of the distribution of Y' which is the collection of predicted TPTP for 10000 simulations. As expected, the graph is an almost perfectly normal distribution. *This is also proven by the fact that the median and mean are almost equal, which implies the graph must be symmetric.*

| Min | Q1 | Median | Mean | Q3 | Max |
|---|---|---|---|---|---|
| 42.78 | 46.05 | 46.87 | 46.91 | 47.76 | 51.47 |

Using the calculations from before, we used R to calculate the value for our Beta estimator and also then found the values for the confidence interval as well (Beta Lower and Beta Upper). As required by the coursework, we used these two values along with the original $\hat{\beta}$ to check for robustness. This is to see if even a slight change in the value of the estimator would affect the overall probability or not. As we can see, there is a lot of difference between the probability when using Beta Lower, Beta Cap, and Beta Upper. *This implies that our results are robust to errors when calculating the value of $\beta$ estimator.*



| | Beta Lower | Beta Cap | Beta Upper |
|---|---|---|---|
| **Probability** | 99.56% ↑ | 86.75% | 35.68% ↓ |

## IV. REFERENCES

[1] *A Tutorial on Fisher Information*, Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul Grasman, and Eric-Jan Wagenmakers, University of Amsterdam, October 2017.
[2] *Basics of Histograms*, Slawa Rokicki, R-Bloggers, December 2012.
[3] *Convert a Dataframe to a Matrix, RGB Color Specification*, R Documentation.
[4] *Descriptive Statistics*, Quick R.

## V. APPENDIX

### *R Code*

```
#read the data
myData = read.csv("TPTP-data-201718.csv")

#print the statistical summary
print(summary(myData))

#convert to a matrix and then plot it
new_Data = data.matrix(myData)
hist(new_Data,   xlim   =   c(0,   0.5),   breaks   =   100,   col   =
rgb(221,160,221, maxColorValue = 255), xlab = "Values", main =
"TPTP Dataset Histogram")

#initialize constants
n = 458
alpha = 5
denom = 0

#calculate betaCap denominator
for (i in 1:n)
  {
    denom = denom + (1/myData[i, 1])
  }
print(denom)

#calculate betaCap using formula betaCap = (n * alpha)/sum of all
from 1 to n(1/x_i)
betaCap = (n * alpha) / denom
print(betaCap)

#Confidence Interval for Beta
beta_L = betaCap - 1.96 * (betaCap / sqrt(n*alpha))
beta_U = betaCap + 1.96 * (betaCap / sqrt(n*alpha))

#create empty array for simulations, each with a different Y'
Y_dash = numeric(10000)

#initialize y
y = sum(myData)

#CASE 1: 10000 simulations where we use betaCap
count = 0
for (j in 1:10000)
{
  x = invgamma::rinvgamma(n, alpha, betaCap)
  Y_dash[j] = sum(x)

  if (Y_dash[j] < y) {
    count = count+1
  }
}

#calculate probability where we use betaCap
prob_betaCap = count/10000

#CASE 2: 10000 simulations where we use lower confidence interval
count = 0
for (j in 1:10000)
{
  x = invgamma::rinvgamma(n, alpha, beta_L)
  Y_dash[j] = sum(x)

  if (Y_dash[j] < y) {
    count = count+1
  }
}

#calculate probability where we used the lower confidence interval
prob_beta_L = count/10000

#CASE 3: 10000 simulations where we use upper confidence interval
count = 0
for (j in 1:10000)
{
  x = invgamma::rinvgamma(n, alpha, beta_U)
  Y_dash[j] = sum(x)

  if (Y_dash[j] < y) {
    count = count+1
  }
}

#plot the probabilities to show effect of different Beta Estimator
vector.numeric     =     c(prob_beta_L*100,     prob_betaCap*100,
prob_beta_U*100)
vector.numeric
barplot(vector.numeric, names.arg = c("Beta Lower", "Beta Cap",
"Beta Upper"), xlab = "Beta values", ylab = "Probabilities", main
= "Visualization", col = "lightpink")

#calculate probability where we used the upper confidence interval
prob_beta_U = count/10000

#required details

#specified break points
hist(Y_dash, breaks = c(41,41.5,42, 42.5, 43, 43.5, 44, 44.5, 45,
45.5,46, 46.5, 47, 47.5, 48, 48.5, 49, 49.5, 50, 50.5, 51, 51.5,
52, 52.5, 53),col = rgb(255, 207, 158, maxColorValue = 255))

#print details
summary(Y_dash)
cat(prob_betaCap * 100, "%", "\n" )
cat(prob_beta_L * 100, "%", "\n" )
cat(prob_beta_U * 100, "%", "\n" )
```