Project 2
Computational Numeric Statistics

João Camacho, N. 56861, Analysis and Engineering of Big Data
Ana Mendes, N. 57144, Analysis and Engineering of Big Data
Lia Schmid , N. 57629, Cienceas Cognitivas
Emanuele Vivoli, N. 57284, Engineering Informatics
Simão Gonçalves, N. 54896, Analysis and Engineering of Big Data

CONTENTS

I. Theoretical Notes

This section will hold some results that will provide useful information in the following exercises. They are detailed here so it doesn't distract us from the main goal in the solution of the exercises.

A. Binomial Distribution

This is a distribution of discrete random variables which models problems such as the following:

1) There are N individuals in a population where K have a specific characteristic and N-K don't have it.
2) We extract (without replacement) n random individuals and we want to know the probability that k of them have the characteristic in question.

This is a generalization of the Bernoulli trial which works under n=1 and k is either $\{0,1\}$. The Binomial distribution is, in fact, the sum of n independent and identically distributed Bernoulli random variables.

Let p be the probability that a single individual has the specific characteristic and n the number of randomly extracted individuals, the Binomial's probability density function is:

$$P(x = k) = f(k) = \binom{n}{k} p^k \cdot (1-p)^{n-k} \tag{1}$$

1) Expected Value of a Binomial RV: For a discrete random variable, the expected value is calculated as:

$$\sum_{i=0}^{n} x_i p_i \tag{2}$$

Therefore, for the Binomial random variable we have:

$$\begin{aligned}
E[X] &= \sum_{x=0}^{n} x \cdot \binom{n}{x} p^x \cdot (1-p)^{n-x} \\
&= \sum_{x=1}^{n} x \cdot \frac{n!}{x!(n-x)!} p^x \cdot (1-p)^{n-x} \\
&= \sum_{x=1}^{n} \cdot \frac{n!}{(x-1)!(n-x)!} p^x \cdot (1-p)^{n-x} \\
&= \sum_{x=1}^{n} \cdot \frac{n \cdot (n-1)!}{(x-1)!(n-x)!} p \cdot p^{x-1} \cdot (1-p)^{n-x} \\
&= np \cdot \sum_{x=1}^{n} \frac{(n-1)!}{(x-1)!(n-x)!} \cdot p^{x-1} \cdot (1-p)^{n-x}
\end{aligned}$$

Making the following substitution:

$$\begin{cases} a = x - 1 \\ b = n - 1 \end{cases}$$

we than obtain:

$$\begin{aligned}
E[X] &= np \cdot \sum_{x=1}^{n} \frac{(n-1)!}{(x-1)!(n-x)!} \cdot p^{x-1} \cdot (1-p)^{n-x} \\
&= np \cdot \sum_{a=0}^{b} \frac{b!}{a! \cdot (b-a)!} \cdot p^a \cdot (1-p)^{b-a} \\
&= np \cdot \sum_{a=0}^{b} \binom{b}{a} p^a \cdot (1-p)^{b-a} \\
&= np \cdot 1 \\
&= np
\end{aligned} \tag{3}$$

B. Bootstrap

Bootstrap is a statistical tool that allows, among other things, the estimation of bias or variance of a particular statistical estimator, or constructing approximate confidence intervals for parameters of interest, through the re-sampling of the available data.

We are usually interested in studying a particular parameter of a population P, however all we have available is a sample from it. Bootstrap assumes the sample represents approximately the distribution of the population and, from its empirical distribution, calculates estimates for properties such as bias or variance of estimators through the re-sample of the available sample.

C. Bootstrap - Estimating standard error of an estimator

Following the previous section, in order to estimate the standard error of an estimator $\hat{\theta}$, we pick a large number B and repeat for b=1,...B:

- draw a bootstrap sample $\{\tilde{z}\}_N^{(b)}$ from the original sample $\{z\}_N$
- compute the statistic $\tilde{\theta}^{(b)}$ for that sample

Note that,

- The notation $\{z\}_N$ represents the set of N observations, $z_1, ..., z_N$ of the available sample;
- The notation $\{\tilde{z}\}_N^{(b)}$ represent the b-th bootstrap sample which has N observations, $\tilde{z}_1^{(b)}, ..., \tilde{z}_N^{(b)}$ sampled from $\{z\}_N$.

Noting that the statistic $\hat{\theta}$ computed on any bootstrap sample b has the notation: $\tilde{\theta}^{(b)}$, an estimate of the standard error $SE(\hat{\theta})$ is then:

$$SE(\hat{\theta}) \approx \sqrt{\frac{1}{B} \sum_{b=1}^{B} \left(\tilde{\theta}^{(b)} - \frac{1}{B} \sum_{r=1}^{B} \tilde{\theta}^{(r)} \right)^2} \tag{4}$$

that is nothing but the sample standard deviation of the bootstrap statistic $\tilde{\theta}^{(1)}, \ldots \tilde{\theta}^{(B)}$.

D. Bootstrap - Estimating the bias of an estimator

Bootstrap is also used to estimate the bias of an estimator. It enables that using the following approximation:

$$\begin{aligned} Bias(\hat{\theta}) &= E(\hat{\theta}) - \theta \\ &\approx E(\tilde{\theta}) - \hat{\theta} \\ &\approx \frac{1}{B}\sum_{b=1}^{B}\tilde{\theta}^{(b)} - \hat{\theta} \end{aligned} \tag{5}$$

The first approximation is the kernel of the estimation. It remains a valid approximation as long as the distribution of $\hat{\theta} - \theta$ and $\tilde{\theta} - \hat{\theta}$ are close. In other words, this approximation assumes that the distribution of $\hat{\theta}$ is similar to the distribution of $\tilde{\theta}$ (which means the expected value of each would be similar), and that $\hat{\theta}$ is itself close to θ. Naturally, these approximations become increasingly valid the better our original sample represents its underlying distribution of the unknown population P. And that depends both on the original sample size and how truly random and independently the observations were sampled.

On a side note, a sufficient, but not necessary, condition for (5) to hold is that $\hat{\theta} - \theta$ be a pivotal (or approximately pivotal) quantity. A pivotal quantity is a quantity that depends on the data and population parameters, but its distribution does not depend on the population parameters.

E. Bootstrap - non parametric confidence intervals

The bootstrap allows the construction of confidence intervals. This means we can compute a $1-\alpha$ confidence interval for θ over the original sample $\{z\}_N$:

$$P(L \leq \theta \leq U) = 1 - \alpha. \tag{6}$$

An important consideration to take into account is that the lower and upper limits of the interval are random - they depend on $\{z\}_N$. And what is being considered in the probability above is just the randomness of the bounds, and θ is fixed.

1) Basic (or pivotal) bootstrap confidence interval: The basic bootstrap confidence interval for θ computes the bootstrap statistics $\{\tilde{\theta}\}_B$ and approximates the distribution of $\hat{\theta} - \theta$ by $\tilde{\theta} - \hat{\theta}$. It assumes essentially the same approximation as in (5) in I-D.

We compute the $\alpha/2$ and $1-\alpha/2$ quantiles of $\{\tilde{\theta}\}_B$, call them $q_{\alpha/2}$ and $q_{1-\alpha/2}$ and argue:

$$\begin{aligned} 1-\alpha &= P(q_{\alpha/2} \leq \tilde{\theta} \leq q_{1-\alpha/2}) \\ &= P(q_{\alpha/2} - \hat{\theta} \leq \tilde{\theta} - \hat{\theta} \leq q_{1-\alpha/2} - \hat{\theta}) \\ &\overset{(5)}{\approx} P(q_{\alpha/2} - \hat{\theta} \leq \hat{\theta} - \theta \leq q_{1-\alpha/2} - \hat{\theta}) \\ &= P(q_{\alpha/2} - 2\hat{\theta} \leq -\theta \leq q_{1-\alpha/2} - 2\hat{\theta}) \\ &= P(2\hat{\theta} - q_{1-\alpha/2} \leq \theta \leq 2\hat{\theta} - q_{\alpha/2}) \end{aligned} \tag{7}$$

Therefore, we use $[L, U] = [2\hat{\theta} - q_{1-\alpha/2}, 2\hat{\theta} - q_{\alpha/2}]$ as an approximate $(1-\alpha)$ confidence interval for θ.

2) Bootstrap percentile confidence interval: This is a very simple method to compute bootstrap confidence intervals. Essentially, it uses the empirical quantiles of the vector of bootstrap estimated $\theta's$, $\{\tilde{\theta}\}_B$, to compute the estimated confidence intervals for θ.

3) Bootstrap studentized confidence interval: This method relies on:

$$\frac{t-\theta}{\hat{se}(t)} \overset{d}{\sim} \frac{t^* - t}{se(t^*)} \tag{8}$$

4) Bootstrap: How bias affects confidence intervals: Let's say there is a bias in our bootstrap estimator of the population parameter θ ,

$$\bar{\theta}^* = \hat{\theta} + B \tag{9}$$

Where,

- $\bar{\theta}^*$ is the mean of the bootstrap estimations of θ, $\frac{1}{B}\sum_{b=1}^{B}\tilde{\theta}^{(b)}$, for each bootstrap sample b;
- $\hat{\theta}$ is the estimator of θ in the available sample of the population.

Let's consider $\bar{\theta}^* - \delta_1$ and $\bar{\theta}^* + \delta_2$ to be, respectively, the $\alpha\%$ and $(1-\alpha)\%$ quantiles of $\{\tilde{\theta}\}_B$. The percentile confidence interval is calculated as follows:

$$\begin{aligned} [L, U]_{percentile} &= [\bar{\theta}^* - \delta_1, \bar{\theta}^* + \delta_2] \\ &= [\hat{\theta} + B - \delta_1, \hat{\theta} + B + \delta_2] \end{aligned} \tag{10}$$

And the $\alpha\%$ and $(1-\alpha)\%$ quantile estimates of the CI by the basic/pivotal bootstrap are:

$$\begin{aligned} [L, U]_{basic} &= [2\hat{\theta} - q_{1-\alpha/2}, 2\hat{\theta} - q_{\alpha/2}] \\ &= [2\hat{\theta} - (\bar{\theta}^* + \delta_2), 2\hat{\theta} - (\bar{\theta}^* - \delta_1)] \\ &= [\hat{\theta} - B - \delta_2, \hat{\theta} - B + \delta_1] \end{aligned} \tag{11}$$

Looking at the CI's from both methods we see that it is possible for both CI's to be very affected by the bias and consequently flip on the CI limits around the biased center. An example of this happening can be found here: [9], where the the estimations on samples from the empirical distribution were by definition biased for the particular estimator used. In the extreme case this could mean the intervals won't even catch the estimate on the original sample, $\hat{\theta}$:

$$[L, U]_{basic} < \hat{\theta} < [L, U]_{percentile} \tag{12}$$

This raises the question of the reliability of bootstrap confidence intervals. There's a list of assumptions the bootstrap makes in order to work:

- The empirical distribution $\hat{F}$ has to represent well the population distribution F;

- Sampling any quantity from $\hat{F}$ will have the same distribution as sampling from F. In other words, bootstrapping assumes the quantity is pivotal;

Regarding the first assumption, we should try to averiguate how the initial sample was obtained.

F. Likelihood

Let X_1,...,X_n be a population of $X \sim F(\theta)$. Let $f(x;\theta)$ be the density function of X depending on a parameter θ.

Then for each realization $(x_1, \ldots, x_n)$ of the random sample above, the function:

$$\mathcal{L}(\theta|\ x) = f(x;\theta) = P_\theta(X{=}x)$$

is considered as a function of θ, called likelihood function, given the outcome x of the random variable X.

Because $X_1, \ldots, X_n$ are independent random variables, then the likelihood function becomes:

$$\mathcal{L}(\theta|\ x) = f(x_1, ..., x_n;\theta) \overset{ind.}{=} \prod_{i=1}^{n} f(x_i;\theta)$$

If we apply the log function to the likelihood function it becomes the log-likelihood (and we can do it without changing the maximum of the original function because logarithm is a monotonically increasing function):

$$l\ (\theta|\ x) = \sum_{i=1}^{n} \log(f(x_i;\theta)) \tag{13}$$

It is important to note that $\mathcal{L}(\theta|x)$ should not be confused with $p(\theta|x)$; the likelihood is equal to the probability that a particular outcome x is observed when the true value of the parameter is θ , and hence it is equal to a probability density over the outcome x, not over the parameter θ.

The use of the probability density in specifying the likelihood function above is justified as follows. Given an observation x_j, the likelihood for the interval $[x_j, x_j + h]$, where $h > 0$ is a constant, is given by $\mathcal{L}(\theta \mid x \in [x_j, x_j + h])$.

Observe that

$$\begin{aligned}&\operatorname*{argmax}_\theta\ \mathcal{L}(\theta \mid x \in [x_j, x_j + h])\\ =\ &\frac{1}{h}\operatorname*{argmax}_\theta\ \mathcal{L}(\theta \mid x \in [x_j, x_j + h])\end{aligned}$$

since h is positive and constant.
Because:

$$\begin{aligned}&\frac{1}{h}\operatorname*{argmax}_\theta\ \mathcal{L}(\theta|x \in [x_j, x_j + h])\\ =&\frac{1}{h}\operatorname*{argmax}_\theta\ \mathcal{P}\nabla(x_j \le x \le x_j + h|\theta)\\ =&\frac{1}{h}\operatorname*{argmax}_\theta\ \int_{x_j+h}^{x_j} f\ (x|\theta)dx\end{aligned}$$

where $f(x \mid \theta)$ is the probability density function, it follows that:

$$\begin{aligned}&\frac{1}{h}\operatorname*{argmax}_\theta\ \mathcal{L}(\theta|x \in [x_j, x_j + h])\\ &\qquad = \frac{1}{h}\operatorname*{argmax}_\theta\ \int_{x_j+h}^{x_j} f\ (x|\theta)dx\end{aligned}$$

The first fundamental theorem of calculus and the l'Hôpital's rule together provide that:

$$\begin{aligned}\lim_{h\to 0^+} \frac{1}{h}\operatorname*{argmax}_\theta\ &\int_{x_j+h}^{x_j} f\ (x|\theta)dx =\\ &= \lim_{h\to 0^+} \frac{\frac{d}{dh}\int_{x_j}^{x_j+h} f(x|\theta)dx}{\frac{dh}{dh}}\\ &= \lim_{h\to 0^+} \frac{f(x_j + h|\theta)}{1}\\ &= f(x_j|\theta)\end{aligned} \tag{14}$$

We can see, applying what we did in (14), the following:

$$\begin{aligned}\operatorname*{argmax}_\theta\ \mathcal{L}(\theta|x_j) &= \operatorname*{argmax}_\theta\ [\lim_{h\to 0^+}\ L\ (\theta|x \in [x_j, x_j + h])]\\ &= \operatorname*{argmax}_\theta\ [\lim_{h\to 0^+} \frac{1}{h}\int_{x_j}^{x_j+h} f(x|\theta)dx]\\ &= \operatorname*{argmax}_\theta\ f(x_j|\theta)\end{aligned}$$

and so we conclude that maximizing the probability density at x_j amounts to maximizing the likelihood of the specific observation x_j.

For Θ = θ_1,...,θ_p the score function is calculated as :

$$S\ (\Theta) = \frac{\partial}{\partial\Theta}\ l\ (\Theta|\ x) = \begin{bmatrix}\frac{\partial}{\partial\theta_1} \cdots \frac{\partial}{\partial\theta_p}\end{bmatrix} \tag{15}$$

We can then obtain the Hessian matrix, which is given by the derivative of the Score function :

$$\begin{aligned}H\ (\theta) =&\frac{\partial}{\partial\theta}\ S\ (\theta) = \frac{\partial^2}{\partial\theta^2}\ l\ (\theta|\ x) =\\ =&\begin{bmatrix}\frac{\partial^2}{\partial\theta_1^2}\ l\ (\theta) & \frac{\partial^2}{\partial\theta_1\partial\theta_2}\ l\ (\theta) & \cdots & \frac{\partial^2}{\partial\theta_1\partial\theta_p}\ l\ (\theta)\\ \frac{\partial^2}{\partial\theta_2\partial\theta_1}\ l\ (\theta) & \frac{\partial^2}{\partial\theta_2^2}\ l\ (\theta) & \cdots & \frac{\partial^2}{\partial\theta_2\partial\theta_p}\ l\ (\theta)\\ & & \vdots & \\ \frac{\partial^2}{\partial\theta_p\partial\theta_1}\ l\ (\theta) & \frac{\partial^2}{\partial\theta_p\partial\theta_2}\ l\ (\theta) & \cdots & \frac{\partial^2}{\partial\theta_p^2}\ l\ (\theta)\end{bmatrix}\end{aligned} \tag{16}$$

More specifically, if the likelihood function is twice continuously differentiable on the k-dimensional parameter space Θ of all the $\theta_1, ..., \theta_p$, then there exists a unique maximum of the likelihood function with a set $\boldsymbol{\theta}_{max} \in \Theta$. Also the matrix $H(\theta)$ exists and is negatively defined at every $\theta \in \Theta$ for which gradient

$$\nabla\mathcal{L}_i = \frac{\partial\mathcal{L}}{\partial\theta_i}$$

vanishes and

$$\lim_{\theta\to\partial\Theta} \mathcal{L}(\theta) = 0$$

i.e. the likelihood function approaches a constant on the boundary of the parameter space Θ.

As a result the $\boldsymbol{\theta}_{max}$ approaches the MLE (Maximum likelihood estimator), as demonstrated in [5] and [4].

G. Fisher information

We can determine the Fisher information (if X is continuous) as:

$$\mathcal{I}_X(\theta) = \int_X (\frac{d}{d\theta} \log f(x|\theta))^2 p_\theta(x) dx$$

and if X is discrete:

$$\mathcal{I}_X(\theta) = \sum_X (\frac{d}{d\theta} \log f(x|\theta))^2 p_\theta(x)$$

The derivative $\frac{d}{d\theta} \log f(x|\theta)$ is the score function of x, defined in (15). This function describes how sensitive the model is to changes in θ at a particular θ.

If we use the score vector defined in (15) we can calculate the fisher information matrix as:

continuous

$$I_X\ (\theta)_{i,j} = -\int_X \frac{\partial^2}{\partial\theta_i \partial\theta_j} \log f(x|\theta) p_\theta(x) dx \tag{17}$$

discrete

$$I_X\ (\theta)_{i,j} = -\sum_X \frac{\partial^2}{\partial\theta_i \partial\theta_j} \log f(x|\theta) p_\theta(x) \tag{18}$$

but in both cases they can be also written as:

$$\begin{aligned} I_X\ (\theta)_{i,j} &= E\left[\Big(\frac{\partial}{\partial\theta_i}\ l\ (\theta_i)\Big)\Big(\frac{\partial}{\partial\theta_j}\ l\ (\theta_j)\Big)^T\right] \\ &= -\ E\left[H\ (\theta)\right] \end{aligned} \tag{19}$$

For so-called regular parametric models, the MLE for vector-valued parameters θ converges in distribution to a multivariate normal distribution, that is,

$$\sqrt{n}(\theta - \theta*) \rightarrow^D N_d\ (0, I_X^{-1}(\theta*)), \text{as n} \rightarrow \inf \tag{20}$$

$\theta*$ is the MLE and the θ is the estimated parameter,N_d is a d-dimensional multivariate normal distribution and $I_X^{-1}\ (\theta*)$ is the inverse Fisher information matrix at the true value of $\theta*$.

For n large enough, we can thus, approximate the sampling distribution of the "error" of the MLE by a normal distribution.

$$(\theta - \theta*) \approx^D N_d\ (0, \frac{1}{n} I_X^{-1}(\theta*)) \tag{21}$$

In practice, we fix n and replace the true sampling distribution by this normal distribution. Hence, we incur an approximation error that is only negligible whenever n is large enough, as demonstrated in [3].

H. Optimization

In cases we have a non-linear equation in some parameters and therefore cannot solve it analytically, one can use numerical methods. These are usually iterative methods which consecutively approximate the solution by improving every step. One usually needs a starting value, the first estimator, which can either be detected graphically or methods of the moments. Furthermore, an update equation is needed for the next iteration. At last, one stops the optimization algorithm when some stopping rule is achieved.

*1) **Convergence Criteria**:* In exercise 3 the *absolute convergence criterion* is used, so the iterative procedure stops when

$$|\theta^{(t+1)} - \theta^{(t)}| < \epsilon \tag{22}$$

Exercises 4 uses the *relative convergence criterion*. The methods stop when

$$\frac{|\theta^{(t+1)} - \theta^{(t)}|}{|\theta^{(t)}|} < \epsilon \tag{23}$$

ϵ is usually a very small value named the tolerance or precision constant.

*2) **Bisection method**:* We can imagining this method to partition an interval into two parts until finding a root. Therefore, one uses the following theorem.

Theorem I.1 (Bolzano theorem)**.** *Let f be a continuous function in the limited interval [a, b] $\in \mathbb{R}$ such that:*

$$f(a)f(b) \leq 0$$

***then** f has at least one root $x^* \in]a, b[$.*

Idea of Proof

In order to give an idea of the proof, we want to show that with the updating method introduced in (25) , we will surely obtain a root and (I.1) holds.

Let w.l.o.g. $f(a) \leq 0 \leq f(b)$

One define: $a_0 = a, b_0 = b, I_0 = [a_0, b_0], x^{(0)} = \frac{a_0+b_0}{2}$.

There are now three cases to be considered, where the first iteration has $n = 0$:

$$\begin{cases} f(x^{(n)}) = 0 : \text{set } x^* = x^{(n)} \text{ done.} \\ f(x^{(n)}) > 0 : \text{set } a_{n+1} = a_n, b_{n+1} = x^{(n)} \\ f(x^{(n)}) < 0 : \text{set } a_{n+1} = x^{(n)}, b_{n+1} = b_n \end{cases} \tag{24}$$

This procedure is continued until finding the root or fulfilling a convergence rule (see (22, 23)). So, the general update rule is as follows:

$$[a_{n+1}, b_{n+1}] = \begin{cases} [a_n, x^{(n)}] \text{ if } f(a_n)f(x^{(n)}) \leq 0, \\ [x^{(n)}, b_n] \text{ if } f(a_n)f(x^{(n)}) > 0 \end{cases} \tag{25}$$

using $x^{(t+1)} = \frac{a_{t+1}+b_{t+1}}{2}$

To show that the bisection method actually works, we can create two sequences which bound each other:

1) $(a_n)_{n\geq 0} : a_0 \leq a_1 \leq a_2 \leq ... \leq b_0 = b$
   $\rightarrow (a_n)_{n\geq 0}$ as a monotonic increasing sequence

2) $(b_n)_{n\geq 0} : b_0 \geq b_1 \geq b_2 \geq ... \geq a_0 = a$
   $\rightarrow (b_n)_{n\geq 0}$ as a monotonic decreasing sequence

Because of the Monotone convergence theorem both sequences converge.

Furthermore, we have the size of the interval in step n as $b_n - a_n = \frac{b_0 - a_0}{2^n} \quad \forall n \in \mathbb{N}_0$. This is simple to **Proof** by induction.

Base Case $(B.C.) : n = 0 :$

$$b_0 - a_0 = \frac{b_0 - a_0}{2^0} \Leftrightarrow b_0 - a_0 = \frac{b_0 - a_0}{1} \Leftrightarrow b_0 - a_0 = b_0 - a_0 \checkmark$$

Induction Hypotheses $(I.H.)$: *step with* n

$$b_n - a_n = \frac{b_0 - a_0}{2^n}$$

Inductive Step $(I.S.)$ $n \to n+1$

As shown in (24), we have two cases, or the a_n increases or the b_n decreases. For this proof lets suppose (w.l.o.g.) the case in which $f(x^{(n)}) > 0$ (for the opposite case the basic idea is the same). So for this case the $a_{n+1} = a_n$ and the b_{n+1} is equals to $x^{(n+1)}$, so $b_{n+1} = \frac{(b_n + a_n)}{2}$. Using this definition of a_{n+1} and b_{n+1} we can substitute in the $n+1$ formulation:

$$\begin{aligned} b_{n+1} - a_{n+1} &= \frac{(b_n + a_n)}{2} - a_n \\ &= \frac{b_n}{2} + \frac{a_n}{2} - \frac{2}{2}a_n \\ &= \frac{b_n}{2} - \frac{a_n}{2} \\ &= \frac{1}{2}(b_n - a_n) \\ &\overset{I.H.}{=} \frac{1}{2}\frac{(b_0 - a_0)}{2^n} \\ &= \frac{b_0 - a_0}{2^{n+1}} \quad \square \end{aligned} \tag{26}$$

We showed until now that the sequences $(a_n)_{n\geq 0}$ and $(b_n)_{n\geq 0}$ are monotonically convergent to the same point (because of (26), with $n \to \infty$, the distance between the limit point of (b_n) and (a_n) is 0). As the next of x_n is calculated with $x_{n+1} = \frac{a_{n+1} + b_{n+1}}{2}$ we can see $(x_n)_{n\geq 0}$ as sequence of midpoints. The sequence $(x_n)_{n\geq 0}$ is itself lower and upper bounded from $(a_n)_{n\geq 0}$ and $(b_n)_{n\geq 0}$, that force all to converge at the same point x^*. Because of our precondition $f(a_n) \leq 0$ and $f(b_n) \geq 0 \ \forall n \in \mathbb{N}_0$ one knows that $(a_n) \leq (x_n) \leq (b_n)$ using the Squeeze theorem $lim_{n\to\infty} a_n = lim_{n\to\infty} x_n = lim_{n\to\infty} b_n = x^*$. What left now is to show that $f(x^*) = 0$, and it is given from the hypothesis condition of the Theorem I.1 that is always guaranteed: $f(a_n)f(b_n) \leq 0$. It is also true in the limit point, so we we have:

$$\begin{aligned} f(x^*)f(x^*) \leq 0 &\Leftrightarrow [f(x^*)]^2 \leq 0 \\ &\Leftrightarrow f(x^*) = 0 \end{aligned} \tag{27}$$

We can also notice that the error at each iteration is bounded from the value that we had previously demonstrated in (26), and as the interval is always partitioned in half, so is the absolute error in each iteration:

$$\underbrace{|x^* - x^{(n+1)}|}_{|\epsilon_{n+1}|} \leq \frac{b_{n+1} - a_{n+1}}{2} = \frac{1}{2}\frac{b_n - a_n}{2}$$

$$\underbrace{|x^* - x^{(n)}|}_{|\epsilon_n|} \leq \frac{1}{2}(b_n - a_n)$$

then:

$$|\epsilon_{n+1}| \approx \frac{1}{2}|\epsilon_n|$$

Hence the bisection method converges linearly.

To reach the desired precision ϵ (or maximum error admitted) one needs a number of steps corresponding to:

$$n = log_2(\frac{\epsilon_0}{\epsilon}) \tag{28}$$

At this point it is important to notice, that in case there is more than one root in the initial interval, the bisection method will just find one of them, but not the others. And in case we choose an interval were the function has the same sign (so the interval doesn't bracket a root) we end up with no solutions. In addition to this, the convergence of the Bisection method is just linear, and maybe we want to speed up a bit.

*3) **Newton Raphson Method**:* Let x_0 be an initial guess to the root x^* with $f(x^*) = 0$. Let h be the correction i.e. $x^* = x_0 + h$. Then $f(x^*) = 0$ implies $f(x_0 + h) = 0$. Now assuming h **small** and f **twice continuously differentiable**, we find:

$$\begin{aligned} 0 &= f(x^*) \\ &= f(x_0) + (x^* - x_0)f'(x_0) + \frac{(x^* - x_0)^2}{2}f''(\xi_0) \end{aligned} \tag{29}$$

with $\xi_0 \in I(x_0, x^*)$.

This is the second order Taylor expansion in the neighbourhood of x_0, and it's for this quadratic approximation of the function f in the neighbourhood of the root that we need h small and f twice continuously differentiable.

Neglecting quadratic and higher order term ($\frac{h^2}{2}f''(\xi)$) and assuming that x^* is a simple root, we find:

$$h = -\frac{f(x_0)}{f'(x_0)} \tag{30}$$

and than a point obtained as:

$$\begin{aligned} x_1 &= x_0 + h \\ &= x_0 - \frac{f(x_0)}{f'(x_0)} \end{aligned}$$

is a better approximation of x^* than x_0, and still in the neighbourhood. The method of Newton-Raphson suggests the process of finding the unique root as an iterative method with this idea of updating the recent point into a point that better approximates the root x^*, using a step:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

This step from the current point x_n to the next point x_{n+1} can be also visualized as drawing a tangent in x_n, and choosing the next point x_{n+1} as the zero point of the tangent (x_{n+1}, 0), nether less than the root of a linear approximation of the function f in x_n. This method allows us to obtain a significant speed up with respect to the Bisection method, but everything comes with a price: in fact we are supposed to have a single root function at least twice continuosly differentiable and starting from a good approximation (in order to use Taylor expansion).

Lets see how a convergence proof can be obtained (under restricted hypothesis) by proving the following theorem.

Theorem I.2. *If $f \in \mathcal{C}^2$, f'' is continuous and x^* is a simple root of f* ***then*** *there exists a neighborhood of x^* for which Newton's and Raphson's method converges to x when started from any x_0 in that neighborhood.*

Idea of Proof
Suppose:

- x^* simple root
- f continuous derivative
- $f'(x^*) \neq 0$

Since $f'(x^*) \neq 0$ and f' continuous in x^*, than there exist a neighbourhood of x^* in which $f'(x) \neq 0$, $\forall x$. Consider the following calculus restricted to this neighbourhood.

Lets define the error at one iteration $\epsilon_n = x_n - x^*$; a Taylor expansion gives (as in (29)):

$$\begin{aligned} 0 &= f(x^*) \\ &= f(x_n) + (x^* - x_n)f'(x_n) + \frac{(x^* - x_n)^2}{2} f''(\xi_n) \end{aligned}$$

with $\xi_n \in (x_n, x^*)$. Rearranging we obtain:

$$x_n + h_n - x^* = (x^* - x_n)^2 \frac{f''(\xi_n)}{2f'(x_n)}$$

where h_n is the Newton update as in (30). Now, for the error ϵ_n definition, and since the left part $x_n + h_n = x_{n+1}$, we conclude

$$\epsilon_{n+1} = (\epsilon_n)^2 \frac{f''(\xi_n)}{2f'(x_n)} \tag{31}$$

Consider now a neighbourhood of x^*, $\mathcal{N}_\delta(x^*) = [x^* - \delta, x^* + \delta]$, for $\delta > 0$. Let

$$c(\delta) = \max_{a,b \in \mathcal{N}_\delta} \left| \frac{f''(a)}{2f'(b)} \right|$$

Since

$$c(\delta) \to \left| \frac{f''(x^*)}{2f'(x^*)} \right|$$

for $\delta \to 0$, it follows that $\delta c(\delta) \to 0$ as $\delta \to 0$. One can then choose δ such as $\delta c(\delta) < 1$.

If $x_n \in \mathcal{N}_\delta(x^*)$, then (31) implies that:

$$\left| c(\delta)\epsilon_{n+1} \right| \leq \left(c(\delta)\epsilon_n \right)^2 \tag{32}$$

Suppose that the starting value is not too bad, $|\epsilon_0| = |x_0 - x^*| < \delta$, then the (32) implies that:

$$\left| \epsilon_n \right| \leq \frac{\left(c(\delta)\delta \right)^{2^n}}{c(\delta)}$$

which converges to 0 as $n \to \infty$. Hence, $x_n \to x^*$.

As shown in the [1], we have just shown the Theorem I.2. In fact, when f is twice continuously differentiable ($\mathcal{C}^2$), is convex, and has a root, then the method converges to the root from any starting point.

If we desired to start from a point somewhere in an interval $\mathcal{I} = [a, b]$, we need to guaranteed that:

- $f'(x) \neq 0$ in $\mathcal{I}$
- $f''(x)$ doesn't change sign in $\mathcal{I}$
- $f(a)f(b) < 0$
- $f(a)/f'(a) < b - a$ and $f(b)/f'(b) < b - a$

if all is guaranteed, so the Newton Raphson method will converge $\forall\ x_0 \in \mathcal{I}$.

Regarding the speed of convergence, the (31) clearly define a relation between the error at step $n + 1$ and at step n as:

$$\frac{\epsilon_{n+1}}{(\epsilon_n)^2} < \frac{f''(\xi_n)}{2f'(x_n)} \tag{33}$$

The formula of the convergence of order p says that

$$\lim_{n \to \infty} \frac{|\epsilon_{n+1}|}{|\epsilon_n|^p} < M \tag{34}$$

with $M > 0$, and if $p = 2$ so it's quadratic convergence.

So in case it converges, it has a quadratic ($p = 2$) speed of convergence, and the right part of (33) tends to

$$\frac{f''(x^*)}{2f'(x^*)}$$

that is the M parameter of (34).

In the practical exercises, as we were looking for the max of log-likelihood, the f function we referred to in this section can be intended as the score function $\mathcal{S}$ obtain as (15), that makes the update rule for Newton Raphson method as:

$$x_{n+1} = x_n - \frac{\mathcal{S}(x_n)}{\mathcal{H}_n}$$

where $\mathcal{H}_n$ is the hessian matrix calculated at the n-th iteration, given by the derivative of the score function, as in (16).

4) ***Fisher Scoring Method****:* This method is an alternative approach to Newton Raphson method, that uses for the update rule the expected values of the second derivative of the $l\ (\theta|\ x)$ described in (13). In fact can be shown, as in [2], that Fisher Information Matrix is defined as the covariance of score function. It is a curvature matrix and has interpretation as the negative expected Hessian of log likelihood function. Thus the immediate application of F is as drop-in replacement of H in second order optimization methods.

With this change, the update rule for *Fisher scoring*, also called *Iteratively reweighted least squares*, is the following:

$$x_{n+1} = x_n + \mathcal{S}(x_n) * \mathcal{I}_n^{-1}$$

where $\mathcal{I}_n^{-1}$ is the information matrix calculated at the n-th iteration, given by (minus sign) the expectation of the derivative of the score function ($\mathcal{H}_n$), as in [1].

Fisher scoring and Newton's method both have the same asymptotic properties, but for individual problems one may be computationally or analytically easier than the other. Generally, Fisher scoring works better in the beginning to make rapid improvements, while Newton's method works better for refinement near the end.

II. Exercise 1 - Bootstrap and Jacknife: Batteries problem

We are given a sample of 20 independent battery survival times (time elapsed before having to be replaced). And we consider the following random variable X = "The number of batteries that lived more than 1200 hours in n inspected batteries." X follows a Binomial distribution with parameters Bin(n,p) where p is the probability of surviving more than 1200 hours. Furthermore we define the following statistic:

$$P = \frac{X}{n} \tag{35}$$

A. Show that P is an unbiased and consistent estimator of p

1) Bias:

$$\begin{aligned} E[P] &= E\Big[\frac{X}{n}\Big] \\ &= \frac{1}{n} \cdot E[X] \\ &\overset{\text{I-A}}{=} \frac{1}{n} \cdot np \\ &= p \end{aligned}$$

P is therefore an unbiased estimator of the distribution parameter *p*.

2) Consistency:

$$\begin{aligned} \lim_{n\to\infty} E[P] = \\ &= \lim_{n\to\infty} p \\ &= p \end{aligned}$$

P is also consistent.

3) Estimate P and V(P): The number of batteries that lasted longer than 1200 hours in our sample is 13. And the size of the sample is 20. And since P is an unbiased and consistent estimator of the distribution parameter p, an estimate of p is:

$$\hat{p} = P = \frac{X}{n} = 13/20 = 0.65 \tag{36}$$

The variance of the estimator is:

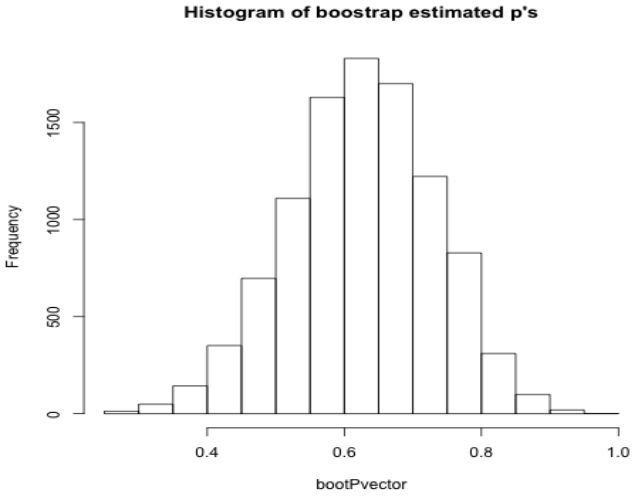
$$\begin{aligned} V(P) = V(\frac{X}{n}) &= \frac{1}{n^2} \cdot V(X) \\ &= \frac{1}{n^2} \cdot np(1-p) = \frac{p(1-p)}{n} \\ &\approx \frac{P(1-P)}{n} = \frac{0.65(1-0.65)}{20} \\ &= 1.1375 \times 10^{-2} \end{aligned}$$

B. Non-Parametric Bootstrap

We are going to estimate the variance and bias of P using the *non-parametric* bootstrap technique. For that we will generate B=10000 samples with replacement of size n (n=20) from the original sample. The bootstrap estimate of p, P^*, and the bias and variance estimates of our estimator P, respectively, are the following:

$$\begin{cases} P^* = \frac{1}{B}\sum_{b=1}^{B} P_b^* \\ bias(P) = E[T-\theta] \approx \frac{1}{B}\sum_{b=1}^{B}(P_b^*) - P = P^* - P \\ Var(P-p) = V(P) \approx \frac{1}{B-1}\sum_{b=1}^{B}(P_b^* - P^*)^2 \end{cases}$$

After generating 10.000 samples we computed the estimated P's for each sample and present the histogram below. We expect the histogram to be centered around the initial estimated P=0.65, with a certain variance.



With this vector of bootstrap estimated P's we can now compute an estimate of the bias and variance of our estimator P. The results are:

$$\begin{cases} P^* = 0.6502 \\ bias(P) = 2.0 \times 10^{-04} \\ Var(P-p) = 1.154 \times 10^{-02} \end{cases}$$

We notice that both the estimated bias and variance of the estimator are relatively low compared to the absolute value of P. Finally, since we now have an estimate of the bias of the estimator P, we can correct P with its bias:

$$P_{biascorrected} = P - bias(P) = 0.6498$$

C. Confidence Intervals for P

We are going to compute the confidence intervals for p using two techniques:

- bootstrap non-parametric **percentile** confidence interval
- bootstrap non-parametric **studentized** confidence interval

The calculation for these intervals can be found in the *Theoretical Notes*. The percentile interval was obtained using the following code:

Listing 1: Bootstrap Percentile interval in R

```
ci_percentile = quantile(bootPvector,
    c(.025,.975))
ci_percentile
# 2.5% 97.5%
# 0.45  0.85
```

and the studentized interval:

Listing 2: Bootstrap studentized interval in R

```
sd_boot_P = c()
set.seed(123)
M = 250
for (boot_sample_idx in
    1:nrow(boot_samples))
{
  boot_sample
    = boot_samples[boot_sample_idx,]

  # vector will contain the estimated P's
  boot_P = c()
  for (m in 1:M){
    boot_sample_ception = sample(
        as.numeric(boot_sample),
        length(boot_sample), replace = TRUE)
    pi = length(boot_sample_ception
        [boot_sample_ception > 1200])
        / length(boot_sample)
    boot_P = c(boot_P, pi)
  }
  # compute sd
  sd_P = sd(boot_P)
  sd_boot_P = c(sd_boot_P, sd_P)
}
# 2
deltastar = (bootPvector - P) / sd_boot_P;
# 3
d = quantile(deltastar, c(.025,.975));
# 4
ci = P - c(d[2],d[1])*sd(bootPvector); ci
#97.5%      2.5%
# 0.381 0.852
```

This result is surprising because the bootstrap studentized interval is usually more precise than the percentile one. Because of this, we decided to test this interval by varying the sample size of each bootstrap between 100 and 1000 to see if the interval could still change significantly (we didn't use a bigger sample size due to computational time). The following are the results of this test:

TABLE I: Studentized interval for different sample sizes

bootstrap sample size	100	250	1000
studentized interval	[0.38, 0.86]	[0.38, 0.85]	[0.38, 0.85]

The intervals didn't change much despite the large increase in sample size.

D. Hipothesis thesting for the population mean

We now intend to test the following hypothesis for the population mean,

$$H_0 : \mu \leq 1000 \text{ vs. } H_1 : \mu > 1000$$

We are using non parametric bootstrap, which means we are not assuming anything about the population.

If we consider the empirical distribution of the population to be the data we have available we easily see that it is not an appropriate estimation, because it does not obey H_0 (the sample mean is $1478.8 \neq 1000$). However if we compute the following transformation z_i for each point in our available data, we notice that the transformed sample as mean $\mu_0 = 1000$.

$$z_i = x_i - \bar{x} + \mu_0$$

Now the test will be based on the approximate distribution of the statistic:

$$t = \frac{\bar{x} - \mu_0}{s} \tag{37}$$

With s being the sample standard deviation. We will approximate the distribution of the test by generating bootstrap samples of the transformed initial sample, $\{z\}_N$, computing the statistic t and finally take the frequency of all the generated samples whose statistic t was less than μ_0.

Listing 3: Hypothesis testing in R

```
set.seed(123)
B=10000;
alpha = 0.05
n = length(A); n
mu0 = 1000;
sd0 = sd(A);
t.obs = (rowMeans(A)-mu0)/sd0
t.npb=numeric(B)
z=A-rowMeans(A)+mu0

for(i in 1:B){
  x.npb=sample(z,n,replace=T)
  sd.npb = sd(x.npb)
```

```
  t.npb[i]=(rowMeans(x.npb)-mu0)/sd.npb
}
# left tailed test
p.value <- sum(t.npb>=t.obs)/B; p.value
# 0.0053
ifelse(p.value < alpha,'Reject H0',
    "Don't reject H0")
# reject H0
```

The p-value obtained was below the confidence level $\alpha = 0.05$, which means we have statistical evidence to reject H_0.

III. Exercise 2 - Bootstrap and Jacknife: Geometric distribution

Let X be a random variable and $X \sim Geom(p)$, which has probability mass function (p.m.f.) given by:

$$f(x;p) = p(1-p)^x \quad x = 0, 1, ... \quad p \in [0, 1] \tag{38}$$

A. Maximum Likelihood Estimator of p

The maximum likelihood estimator (MLE) is is defined as:

$$\hat{\theta}_{MLE} = maxL(\theta|\mathbf{x}) = \prod_{i=1}^{n} f(x_i|\theta) \tag{39}$$

We also can define the MLE with log-likelihood:

$$\begin{aligned} \hat{\theta}_{MLE} &= maxL(\theta|\mathbf{x}) = maxl(\theta|\mathbf{x}) = \\ &= max \sum_{i=1}^{n} log(f(x_i|\theta)) \end{aligned} \tag{40}$$

In our case, the likelihood function of the sample is given by:

$$\begin{aligned} L(p) = L(p|x) &= \prod_{i=1}^{n} f(x_i|p) = \\ &= \prod_{i=1}^{n} p(1-p)^{x_i} \\ &= p^n (1-p)^{\sum_{i=1}^{n} x_i} \end{aligned} \tag{41}$$

And the log-likelihood function of the sample can be written as:

$$\begin{aligned} l(p) = l(p|x) &= logL(p) = \\ &= log(p^n) + log(1-p)^{\sum_{i=1}^{n} x_i} = \\ &= nlog(p) + log(1-p) \sum_{i=1}^{n} x_i \end{aligned} \tag{42}$$

To find the maximum, we need to find the score function, which is a first derivative of the log-likelihood function, is:

$$s(p) = l'(p) = \frac{d}{dp} l(p) = \frac{n}{p} - \frac{1}{1-p} \sum_{i=1}^{n} x_i \tag{43}$$

And solve the equation $s(p) = 0$:

$$\begin{aligned} s(p) = 0 &\Leftrightarrow \frac{n}{p} - \frac{1}{1-p} \sum_{i=1}^{n} x_i = 0 \Leftrightarrow \\ &\Leftrightarrow \frac{n}{p} = \frac{1}{1-p} \sum_{i=1}^{n} x_i \Leftrightarrow \\ &\Leftrightarrow \frac{p \sum_{i=1}^{n} x_i}{n} = 1 - p \Leftrightarrow \\ &\Leftrightarrow p \sum_{i=1}^{n} x_i = n - np \Leftrightarrow \\ &\Leftrightarrow p \left(\sum_{i=1}^{n} x_i + n \right) = n \Leftrightarrow \\ &\Leftrightarrow p = \frac{n^{-1}}{n^{-1}} \cdot \frac{n}{n + \sum_{i=1}^{n} x_i} \Leftrightarrow \\ &\Leftrightarrow p = \frac{1}{1 + \bar{x}} \end{aligned}$$

Therefore, $p = \frac{1}{1+\bar{x}}$ is our ML estimate candidate. Now, we will calculate the derivate of score function to prove that $\frac{1}{1+\bar{x}}$ is the maximum.

$$\begin{aligned} s'(p)\Big|_{p=\frac{1}{1+\bar{x}}} &= l''(p)\Big|_{p=\frac{1}{1+\bar{X}}} = \frac{d}{dp} s(p)\Big|_{p=\frac{1}{1+\bar{X}}} = \\ &= \left(\frac{n}{p} - \frac{1}{1-p} \sum_{i=1}^{n} x_i \right)' \Big|_{p=\frac{1}{1+\bar{X}}} = \\ &= \left(\frac{n}{p} - \frac{\bar{x}}{\frac{1-p}{n}} \right)' \Big|_{p=\frac{1}{1+\bar{X}}} = \\ &= \left(\frac{n}{p} - \frac{n\bar{x}}{1-p} \right)' \Big|_{p=\frac{1}{1+\bar{X}}} = \\ &= - \frac{n}{p^2} - \frac{n\bar{x}}{(1-p)^2} \Big|_{p=\frac{1}{1+\bar{X}}} = \\ &= - \frac{n}{(\frac{1}{1+\bar{x}})^2} - \frac{n\bar{x}}{(1-\frac{1}{1+\bar{x}})^2} = \\ &= - \frac{n}{(1+\bar{x})^{-2}} - \frac{n\bar{x}}{(\frac{\bar{x}}{1-\bar{x}})^2} = \\ &= - n(1+\bar{x})^2 - \frac{n\bar{x}(1-\bar{x})^2}{\bar{x}^2} \\ &= - n(1+\bar{x}^2) \left(1 + \frac{1}{\bar{x}} \right) < 0 \end{aligned} \tag{44}$$

How the second derivative captures the steepness of the curvature around the point p. A more negative second derivative implies that the function is more steeply concave down around the point p.

Therefore, we can conclude that:

$$\hat{p}_{MLE} = \frac{1}{1 + \bar{X}}$$

B. Estimate p

We have an observed sample size of 25 from X and we are going to estimate p with maximum likelihood estimator.

Using the R, we have:

```
p.mle = 1/(1+mean(x))
# p.mle = 0.4310345
```

Therefore, we can conclude for a sample of 25 observations from a geometric distribution when the observed sample mean is $\bar{x} = 1.32$ the MLE of p is 0.431.

We can confirm this estimate graphically, through the likelihood, log-likelihood or score function. Below we have the graphic representations of these three functions.

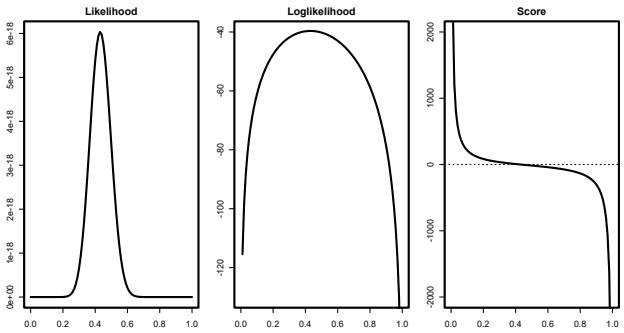


Fig. 1: Likelihood, Log-likelihood and Score functions of Geometric Distribution

C. Jackknife Method

Now, we are going to use the Jackknife Method to calculate the estimate of p and the respective standard error and bias.

In Jackknife method, we generated 25 estimates to p, where the i^{th} jackknife replication is

$$p_i = \frac{1}{1 + \sum_{j \neq i} x_j}$$

which is calculated from the 24 values (without x_i) in the i^{th} jackknife sample.

Let $\hat{p}_{(.)} = \sum_{i=1}^{n} \hat{p}_{(i)}/n$, the jackknife estimate of standard error and bias of our estimator p, respectively, are the following:

$$\begin{cases} \widehat{bias}(\hat{p}) = (n-1)(\hat{p}_{(.)} - \hat{p}) \\ \widehat{var}(\hat{p}) = \frac{n-1}{n} \sum_{i=1}^{n} (\hat{p}_{(i)} - \hat{p}_{(.)})^2 \end{cases}$$

And the code used in R is the following:

```
p.jack=numeric(n)

for(i in 1:n){
    p.jack[i] = 1/(1+mean(x[-i]))
}

# Estimative p
Jack.p = mean(p.jack)
# Standard error of the MLE
se.jack = sqrt((n-1)*
            mean((p.jack-Jack.p)^2))
# Bias of the MLE
bias.jack = (n-1)*(Jack.p-p.mle)
```

D. Bootstrap Method

For the same sample, we will calculate the standard error and bias, but this time with the bootstrap method. Let

$$\begin{cases} \widehat{bias}(\hat{p}) = \hat{p}_{(.)} - \hat{p} \\ \widehat{var}(\hat{p}) = \frac{1}{B-1} \sum_{b=1}^{B} (\hat{p}_b - \hat{p})^2 \end{cases}$$

where $\hat{p}_{(.)} = \frac{1}{B} \sum_{b=1}^{B} \hat{p}(b)$

The code is the following:

```
B=5000
p.boot=numeric(B)

for(i in 1:B){
  x.boot=sample(x,n,replace=T)
  p.boot[i]=(1/(1+mean(x.boot)))
}

# Estimative p
boot.p = mean(p.boot)
se.boot = sd(p.boot)
# Bias of the MLE
bias.boot = mean(p.boot)-p.mle
```

E. Comparison of standard error and bias between Jackknife and Bootstrap Methods

After executing the two methods, we got the following results:

TABLE II: Standard error and Bias of Jackknife and Bootstrap Methods

	Jackknife	Bootstrap
Estimative	0.431	0.438
Standard error	0.057	0.056
Bias	0.007	0.007

We can conclude that in both methods the p estimate value is very close to the true value, and the values of the standard error and bias are the same for Jackknife and Bootstrap methods. So both methods work well for this sample.

IV. Exercise 3 - Optimization: Beta distribution

Let $X \sim Beta(\alpha, 1)$, which has probability density function (p.d.f.) given by:

$$f(x;\alpha) = \frac{x^{\alpha-1}}{B(\alpha,1)}, \alpha > 0, x \in [0,1]$$

Furthermore, we know

$$B(\alpha,1) = \frac{\Gamma(\alpha)\Gamma(1)}{\Gamma(\alpha+1)} = \frac{\Gamma(\alpha)\overbrace{\Gamma(1)}^{=1}}{\alpha\Gamma(\alpha)} = \frac{1}{\alpha}$$

So the simplified p.d.f. is

$$f(x;\alpha) = \frac{x^{\alpha-1}}{\frac{1}{\alpha}} = \alpha x^{\alpha-1}, \alpha > 0, x \in [0,1] \tag{45}$$

A. Derivation of the likelihood, log-likelihood and score functions

The likelihood function of the sample is given by:

$$\begin{aligned} L(\alpha) = L(\alpha|x) &= \prod_{i=1}^{n} f(x_i|\alpha) \\ &= \prod_{i=1}^{n} \alpha x_i^{\alpha-1} = \alpha^n \prod_{i=1}^{n} x_i^{\alpha-1} \end{aligned}$$

The log-likelihood function of the sample can be written as:

$$\begin{aligned} l(\alpha) = l(\alpha|x) &= logL(\alpha) = \\ &= log\left(\alpha^n \prod_{i=1}^{n} x_i^{\alpha-1}\right) \\ &= log(\alpha^n) + log\left(\prod_{i=1}^{n} x_i^{\alpha-1}\right) \\ &= nlog(\alpha) + \sum_{i=1}^{n} log(x_i^{\alpha-1}) \\ &= nlog(\alpha) + \sum_{i=1}^{n} (\alpha-1) log(x_i) \\ &= nlog(\alpha) + (\alpha-1)\sum_{i=1}^{n} log(x_i) \end{aligned}$$

The score function, which is a first derivative of the log-likelihood function, is:

$$s(\alpha) = l'(\alpha) = \frac{d}{d\alpha} l(\alpha) = \frac{n}{\alpha} - \sum_{i=1}^{n} log(x_i) \tag{46}$$

Furthermore, we compute the derivative of the score function (46) which is the second derivative of the loglikelihood because it is later needed.

$$\begin{aligned} s'(\alpha;x) = l''(\alpha) &= \frac{d}{d\alpha} s(\alpha) \\ &= \left(\frac{n}{\alpha} - \sum_{i=1}^{n} log(x_i)\right)' = -\frac{n}{\alpha^2} \end{aligned} \tag{47}$$

B. Graphical display of the likelihood, log-likelihood and score functions in order to locate the ML estimate of α.

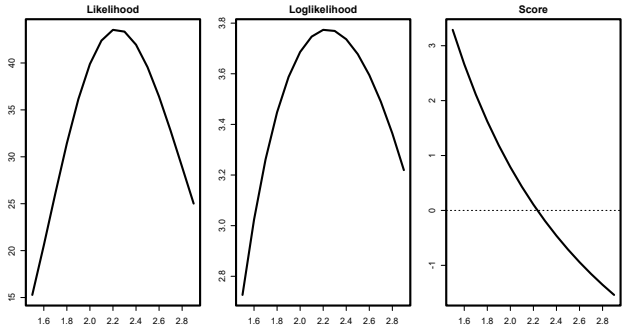
First, we want to try to compute $\hat{\alpha}$ for our maximum likelihood analytically. So, we compute the necessary and sufficient condition of the likelihood that are $l'(\alpha;x) = s(\alpha;x) = 0$ and $l''(\alpha;x) = s'(\alpha;x) < 0$, respectively to obtain a maximum.

$$\begin{aligned} s(\alpha;x) = 0 &\Leftrightarrow \frac{n}{\alpha} + \sum_{i=1}^{n} log(x_i) = 0 \\ &\Leftrightarrow \frac{n}{\alpha} = -\sum_{i=1}^{n} log(x_i) \\ &\Leftrightarrow \alpha = -\frac{n}{\sum_{i=1}^{n} log(x_i)} \end{aligned} \tag{48}$$

In our case (48) is possible to solve and we have $\hat{\alpha} = 2.235083$. So $\hat{\alpha}$ is the candidate for our maximum. Considering $s''(\alpha;x) = -\frac{n}{\alpha^2}$ is always < 0, our candidate corresponds to a maximum. Computing $L(\hat{\alpha}) = L(2.235083) = 43.61213$.

An other approach is the numerical one. Therefore, we first take a look at the plots to get a first graphical estimation for a candidate for α.

Fig. 2: $f(x;\alpha) = \frac{x^{\alpha-1}}{B(\alpha,1)}$



The plots of the likelihood, loglikelihood and score function indicate that the best alpha value can be found at $\alpha \approx 2.2$ This is our first Maximum-Likelihood estimator.

C. Approximation of the ML estimate α using the R function maxLik().

When using the maxLik() function performs a Maximum Likelihood estimation. As we did not specify the method to be used, it uses the Newton Raphson method as default. This is why an initial estimator is needed. For this purpose we used the estimation for α provided by the plots in (2).

```
# graphical maximum likelihood estimator
# using the loglikelihood function
mme.graphical = 2.2
maxLik(logLik=loglik, start=mme.graphical)
#Maximum Likelihood estimation
#Newton-Raphson maximisation, 3 iterations
#Return code 1: gradient close to zero
```

```
#Log-Likelihood: 3.775335 (1 free parameter)
#Estimate(s): 2.235083.
```

As can be seen in the output our graphical estimation was quite close to the one computed by the MaxLik()-function using the Newton-Raphson maximisation which needed three iterations to estimate $\alpha^* = 2.235083$

D. Algorithms of bisection, Newton-Raphson and Fisher scoring to approximate the ML estimate of α using the provided sample. Justification of the choice of the initial estimates.

When trying to find the maximum likelihood estimator one can try to solve

1) *T*here are different approaches for the initial values of the interval.
   - Graphical ML estimator: Looking at the plots in (2) our first ML estimator was 2.2. For our initial values for a and b we chose one value smaller that mme.graphical and the other one bigger, so a.init = 2 and b.init = 2.5. To make use of Bolzano's Theoreme, we tested whether $s(a) \cdot s(b) < 0$.

     ```
     bolz <- s(a.init)*s(b.init); bolz
     # -0.5609915 < 0
     ```

     As this was the case, as shown in (I.1), the method converges in 19 iterations. The convergence is listed in table (V) in the appendix.
   - Start at extreme values: As $\alpha > 0$ we tested the initial value of the left interval very close to 0, so we chose ϵ which is 0.000001. The upper interval value was simple set to b = 5. As $s(a) \cdot s(b) < 0$, again the method converges, this time using 23 iterations (see table (VI) attached).
   - Using values close to the MaxLik() estimator $\alpha^* = 2.235083$. Cutting the last digit our initial values were a = 2.23508 and b = 2.23509. Although so close to the value computed by the function four iterations were needed, as can be seen in table (VII).
2) Newton-Raphson:

   - For the initial estimate for the NR algorithm one can use the moments estimates from the method of moments.
     Using (45) and as $x \in [0, 1]$ we have

$$\begin{aligned}\mathbb{E}(X) &\underset{def.}{=} \int_0^1 x f(x;\alpha)dx = \int_0^1 x\alpha x^{\alpha-1}dx \\ &= \int_0^1 \alpha x^{\alpha}dx = \frac{\alpha}{\alpha+1}x^{\alpha+1}\Big|_0^1 \\ &= \frac{\alpha}{\alpha+1} = \bar{x}\end{aligned} \tag{49}$$

     Now, one can iteratively approximate α by choosing different values and compare, for which α the sample mean is approximated best.

```
mean(x)
# 0.676118

alpha <- seq(1.5,2.9,0.1)
k = 1
for(i in alpha){
  print(c(k, alpha[k],
          (alpha[k]/(alpha[k]+1))))
  k= k+1}
```

As can be seen in table (VIII) the mean is approximated best using $\alpha_{mme.mean} = 2.1$. Using this initial estimator the Newton Raphson method needs 5 steps (see table X).
An other way to get an estimator we solve (49) for α.

$$\begin{aligned}&\frac{\alpha}{\alpha+1} = \bar{x} \Leftrightarrow \frac{\alpha+1}{\alpha} = \frac{1}{\bar{x}} \Leftrightarrow 1 + \frac{1}{\alpha} = \frac{1}{\bar{x}} \Leftrightarrow \frac{1}{\alpha} = \frac{1-\bar{x}}{\bar{x}} \\ &\Leftrightarrow \alpha = \frac{\bar{x}}{1-\bar{x}} = \alpha_{MME}\end{aligned} \tag{50}$$

Below, there is the implementation of the method in R which is analogous used in the other exercises, as well. In case the method diverges the method stops but still shows the last iteration when the value gets negative.

```
NR      <- function(alpha0,eps){
  diff = |alpha_(t+1)-alpha_(t)| <= eps
  alpha.it     = vector()
  alpha.it[1] = alpha0
  k            = 1
  diff         = 1
  broke = FALSE
  # to see whether method diverges
  while(!broke && diff>eps){
    alpha.it[k+1] = alpha.it[k]
    -s(alpha.it[k])/s.prime(alpha.it[k])
    if (alpha.it[k+1] > 0){
      diff
= abs(alpha.it[k+1]-alpha.it[k])
    }else{
      broke = TRUE
    }
    k              = k+1
  }
  result = as.matrix(alpha.it)
  colnames(result)<-"iterates"
  rownames(result)<-1:length(alpha.it)
  result
}
```

Table (XI) shows the iterations using the Newton Raphson method starting with (50), the estimation of this α.

3) To use the Fisher scoring method as shown in ...

$$\alpha^{(t+1)} = \alpha^{(t)} + \frac{l'(\alpha^{(t)})}{I(\alpha)^{(t)}}$$

As shown in ... the Fisher information is computed as follows and we can use the derivative of the score function (47).

$$I(\alpha) = -\mathbb{E}\left[s'(\alpha; x)\right] = -\mathbb{E}\left[\underbrace{-\frac{n}{\alpha^2}}_{const.}\right]$$
$$= -(-\frac{n}{\alpha^2}) = \frac{n}{\alpha^2}$$

The implementation in R is quite the same as the Newton Raphson method just with the update rule

```
alpha.it[k+1] = alpha.it[k]
    +s(alpha.it[k])/I(alpha.it[k])
```

When using the graphical initial estimate the scoring fisher method needed four steps, with the approximation of the mean and the expected value five steps are needed (see tables (XIV) (XV), (XVI)).

E. Sensitivity of the methods to the initial estimates?

First, it is important to notice, that our Newton Raphson method and our Scoring Fisher method do exactly the same. As the Fisher Information $I(\alpha) = \frac{n}{\alpha^2} = -s'(\alpha)$ the update rule in both methods is the following.

$$\alpha_{t+1} = \alpha_t + \frac{\frac{n}{\alpha_t} - \sum_{i=1}^{n} log(x_i)}{\frac{n}{\alpha_t^2}}$$
$$= \alpha_t + (\frac{n}{\alpha_t} - \sum_{i=1}^{n} log(x_i)) * \frac{\alpha_t^2}{n}$$
$$= \alpha_t + \alpha_t - \frac{\alpha_t^2}{n}\sum_{i=1}^{n} log(x_i)$$
$$= 2\alpha_t - \alpha_t^2 \bar{x}$$

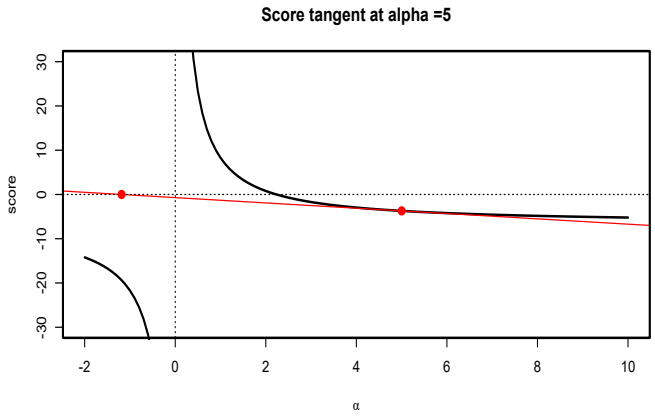
Table (III) compares how many iterations were needed in the different methods.

TABLE III: Comparison of the amount of iterations needed using the Bisection method, Newton Raphson method and Scoring Fisher.

Init. Estim.	**[a, b] Bisection**	**NR**	**SF**
graphical	[2, 2.5] 19	4	4
mean	-	5	5
expectation	[mme $\pm 10^{-5}$]4	5	5
5	-	$\emptyset$	$\emptyset$
epsilon	[epsilon, 5] 23	29*	29*

Lets do some considerations.
In our practical example, in Newton Raphson and Fisher scoring, we check every iteration of this method if the value of alpha is still possible, so if it is $\alpha > 0$ and $\alpha < \infty$. In the code we just check if it is bigger than 0 because our score function doesn't allow the methods to diverge to ∞.
As explained above the amount of iterations of the Newton Raphson and Fisher Scoring method is the same. One can see that the bisection method in general needs much more steps to converge at it is just a linear improvement of the best root. $\emptyset$ in the table means that there was no solution found. The only advantage of the bisection method can be seen when using epsilon and 5 as interval, and epsilon as starting point for NR and SF. This is because with this starting point the last two methods do at the beginners steps very small steps that allow those to converge slowly. We also changed in this example (that appears in the table (III) with a *) the epsilon value to be smaller than the initial epsilon. So we conclude that the Newton Raphson method (ans as well the Scoring Fisher method) is much more sensitive to the initial estimations as depending on the starting point the algorithm either converges very fast or possibly does not converge at all. In the Bisection method one just has to confirm that indeed there is a root between the limits, afterwards convergence is assured, but much slower as it is a linear process.

Fig. 3: Score function with the tangent of $\alpha = 5$



V. Exercise 4 - Optimization: Exponential distribution

Let $X \sim Exp(\lambda)$, which has probability density function (p.d.f.) given by:

$$f(x; \lambda) = \lambda e^{-\lambda x}, \lambda > 0, x \geq 0 \tag{51}$$

The likelihood function of the sample is given by:

$$L(\lambda) = L(\lambda|x) = \prod_{i=1}^{n} f(x_i|\lambda) = \prod_{i=1}^{n} \lambda e^{-\lambda x_i}$$
$$= \lambda^n e^{-\lambda \sum_{i=1}^{n} x_i}$$

The log-likelihood function of the sample can be written as:

$$\begin{aligned} l(\lambda) = l(\lambda|x) &= logL(\lambda) = \\ &= log(\lambda^n e^{-\lambda \sum_{i=1}^n x_i}) \\ &= log(\lambda^n) + log(e^{-\lambda \sum_{i=1}^n x_i}) \\ &= nlog(\lambda) - \lambda \sum_{i=1}^n x_i \end{aligned}$$

The score function, which is a first derivative of the log-likelihood function, is:

$$s(\lambda) = l'(\lambda) = \frac{d}{d\lambda} l(\lambda) = \frac{n}{\lambda} - \sum_{i=1}^n x_i$$

To analytically detect the maximum we solve the equation $s(\lambda) = 0$

$$\begin{aligned} & s(\lambda) = 0 \Leftrightarrow \frac{n}{\lambda} - \sum_{i=1}^n x_i = 0 \\ \Leftrightarrow & \frac{n}{\lambda} = \sum_{i=1}^n x_i \Leftrightarrow \frac{1}{\lambda} = \frac{1}{n} \sum_{i=1}^n x_i \\ \Leftrightarrow & \frac{1}{\lambda} = \overline{x} \Leftrightarrow \lambda = \frac{1}{\overline{x}} \end{aligned} \tag{52}$$

To see whether the candidate of (52) is really a maximum we compute the second derivative of the loglikelihood which is the derivative of the score function and check whether it is < 0.

$$\begin{aligned} s'(\lambda)\Big|_{\lambda=\frac{1}{\overline{x}}} &= l''(\lambda)\Big|_{\lambda=\frac{1}{\overline{x}}} = \frac{d}{d\lambda} s(\lambda)\Big|_{\lambda=\frac{1}{\overline{x}}} \\ &= \left(\frac{n}{\lambda} - \sum_{i=1}^n x_i\right)'\Big|_{\lambda=\frac{1}{\overline{x}}} = -\frac{n}{\lambda^2}\Big|_{\frac{1}{\overline{x}}} \\ &= -\frac{n}{(\frac{1}{\overline{x}})^2} = -n\overline{x} < 0 \end{aligned}$$

Using this result we know that there really is a maximum and the maximum likelihood can be calculated. $L(\hat{\lambda}) = L(1.645161) = 3.530575 \cdot 10^{-06}$. Furthermore, we display the likelihood, log likelihood and score function to use these estimates for the following exercises. These are shown in (4).

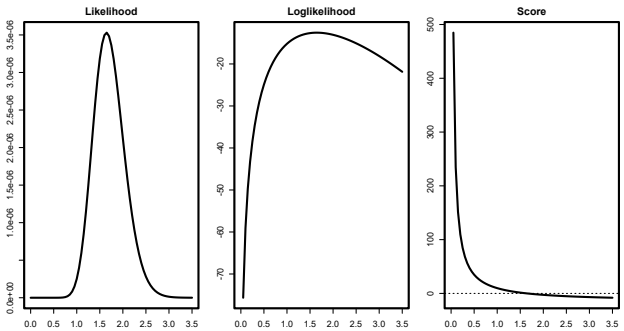


Fig. 4: $f(x;\lambda) = \lambda e^{-\lambda x}$

A. Newton-Raphson method to approximate the ML estimate of λ

For the initial estimate for the NR algorithm one can use the moments estimates as from the method of moments. Using (51) and $x \geq 0$ we have

$$\begin{aligned} \mathbb{E}(X) &\underset{def.}{=} \int_0^\infty x f(x;\lambda) dx \\ &= \int_0^\infty x\lambda e^{-\lambda x} dx = \lambda \int_0^\infty x e^{-\lambda x} dx \\ &= \cancel{\lambda}\left(x(-\frac{1}{\cancel{\lambda}} e^{-\lambda x})\Big|_0^\infty - \int_0^\infty -\frac{1}{\cancel{\lambda}} e^{-\lambda x} dx\right) \\ &= \left(-x e^{-\lambda x}\Big|_0^\infty - \int_0^\infty -e^{-\lambda x} dx\right) \\ &= \left(-x e^{-\lambda x} - \frac{1}{\lambda} e^{-\lambda x}\right)\Big|_0^\infty \\ &= \left(-e^{-\lambda x}(x + \frac{1}{\lambda})\right)\Big|_0^\infty = 0 - (-\frac{1}{\lambda}) = \frac{1}{\lambda} = \bar{x} \end{aligned} \tag{53}$$

Now, one can iteratively approximate λ by choosing different values and compare, for which λ the sample mean is approximated best.

```
mean(x)
# 0.6078434

lambda <- seq(1,2.4,0.1)
k = 1
for(i in lambda){
  print(c(k,lambda[k],
          (1/(lambda[k])))))
  k= k+1
}
```

As can be seen in table (XIX) the mean is approximated best using $\lambda_{mme.mean} = 1.7$. Using this initial estimator the Newton Raphson method needs five steps (see table XXI).
An other way to get an estimator is to solve (53) for λ.

$\Rightarrow \lambda_{MME} = \frac{1}{\bar{x}}$. In this case the algorithm is very fast and only needs two iterations (see table (XXII)). A divergence occurs when using λ = 5 (table (XXIII)).

The method needs a lot of iterations when the initial value is close to 0, in this case 27 steps are needed (table XXIV)).

B. Reparametrization

Let b = log(λ). Because $\lambda > 0$, one has that b $\in \mathbb{R}$. So, b is unrestricted. We obtain the reparametrized loglikelihood function as follows:

$$l(b) = nb - e^b \sum_{i=1}^n x_i$$

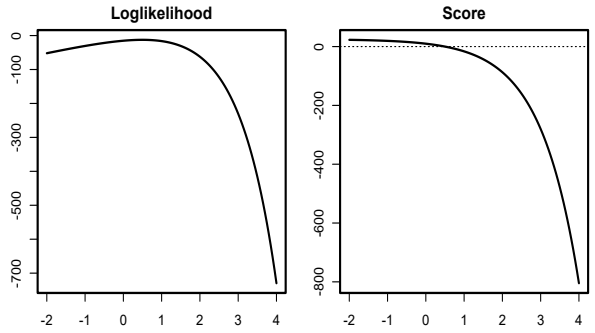
The new score function is the following:

$$s(b) = l'(b) = \frac{d}{db} l = n - e^b \sum_{i=1}^n x_i$$

And the derivative of the score function as it is also needed for the Newton Raphson algorithm.

$$s'(b) = l''(b) = \frac{d}{db}s = -e^b \sum_{i=1}^{n} x_i$$

Fig. 5: $f(x;\lambda) = \lambda e^{-\lambda x}$ using the reparametrization $b = log(\lambda)$



As a first estimate we can again use the MaxLik()- function. From the plots we can guess a first estimate for the b - value. Here, one has to convert the result back to get the λ-value.

```
mme.graph.reparam = 1
maxLik(logLik=loglik.b, start=mme.graph.b)
# Maximum Likelihood estimation
# Newton-Raphson maximisation, 4 iterations
# Return code 1: gradient close to zero
# Log-Likelihood: -12.55405
# (1 free parameter)
# Estimate(s): 0.497838
```

$$\Rightarrow b = log(\lambda) \Leftrightarrow \lambda = e^b = e^{0.497838} = 1.645161$$

This corresponds to the maximum likelihood estimator of exercise a). The update rule is now as follows. This time, the relative convergence criterion is used.

```
b.it[k+1] = b.it[k]
    -s(b.it[k])/s.prime(b.it[k])
    lambda.it[k+1] = exp(b.it[k+1])
    if (lambda.it[k+1] > 0){
      diff = abs(b.it[k+1]-b.it[k])
        /abs(b.it[k])
```

C. Sensitivity to initial values of both approaches.

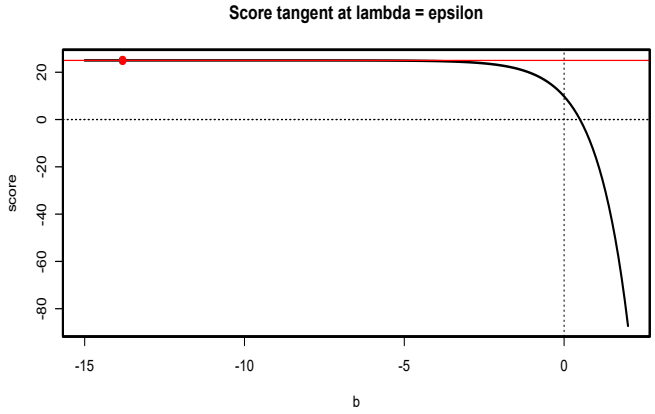
Table (IV) compares the number of iterations needed by the normal Newton Raphson and the one using reparametrization.

The advantage of the unrestricted parameter b has a price. Because of the logarithm the method in general needs more iterations than without the reparametrization. One can call the Newton Raphson algorithm using the reparametrization with the same values as before, so the graphical estimation, the approximation of the mean and the expected value. There are seven, eight and eight iterations needed, respectively (see tables (XXV), (XXVI),(XXVII). When using an initial value further away, say $\lambda_{MME} = 5$, the algorithm needs 4 steps (see table (XXIII)). Using a value with λ_{MME} almost 0, say ϵ, the method crashes (table (XXIX)) as it falls into a pitfall of the Newton Raphson method as there is a zero slope and the solution shoots off. This is illustrated in figure (6).

TABLE IV: Comparison of the amount of iterations needed using the Newton Raphson method with and without reparametrization.

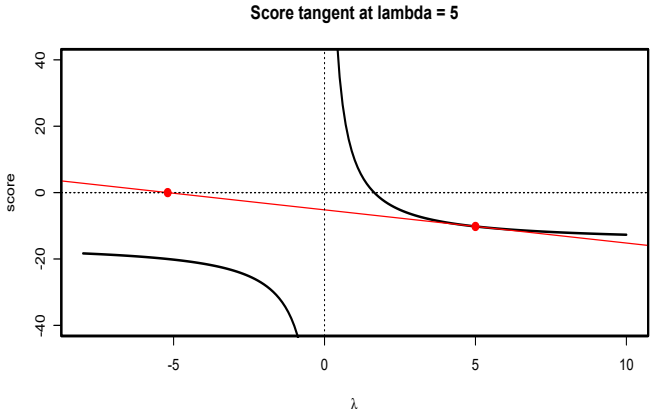
Init. Estim.	NR	Reparam.
graphical	6	7
mean	5	8
expectation	2	8
5	$\emptyset$	4
epsilon	27	$\emptyset$

Fig. 6: $f(x;\lambda) = \lambda e^{-\lambda x}$ using the reparameterization $b = log(\lambda)$



When using $\lambda = 5$ as initial estimation the reparameterized method works quite well as $log(5)$ is close to the real root to obtain the maximum Likelihood. So, in this case it really depends on the initial value which method works better here. On the other hand, a zero slope is achieved when not using the reparametrization (see figure (7)). Nevertheless, the reparameterized method is not so sensitive to the initial values as one can see that the amount of iterations does not change a lot depending on the starting point.

Fig. 7: Score function of $f(x;\lambda) = \lambda e^{-\lambda x}$ with tangent at $\alpha = 5$



REFERENCES

[1] Paolo Giudici, Geof H Givens, and Bani K Mallick. *Wiley Series in Computational Statistics, second edition*. Wiley Online Library, 2013.

[2] Agustinus Kristiadi. Fisher information matrix – by Agustinus Kristiadi's blog. https://wiseodd.github.io/techblog/2018/03/11/fisher-information/. [Accessed: 2019-11-24].

[3] Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul PPP Grasman, and Jan Wagenmakers, Eric. A tutorial on fisher information. *Journal of Mathematical Psychology*, 80:40–55, 2017.

[4] W. F. Mascarenhas. A mountain pass lemma and its implications regarding the uniqueness of constrained minimizers. *Optimization*, 60(8–9):1121–1159, 2011.

[5] Klaus; Styan George P. H. Mäkeläinen, Timo; Schmidt. On the existence and uniqueness of the maximum likelihood estimate of a vector-valued parameter in fixed-size samples. *Annals of Statistics*, 9(4):758–767, 1981.

Appendices

TABLE V: Ex. 3: Bisection with a.init = 2, b.init = 2.5, 19 steps were needed

iterates	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x*	2.25	2.125	2.1875	2.21875	2.234375	2.242188	2.238281	2.236328	2.235352	2.234863
iterates	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	
x*	2.235107	2.234985	2.235046	2.235077	2.235092	2.235085	2.235081	2.235083	2.235084	

TABLE VI: Ex. 3: Bisection with a.init = epsil, b.init = 5, 23 steps were needed

iterates	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
x*	2.505	1.2575	1.88125	2.193125	2.349062	2.271094	2.232109	2.251602	2.241855	2.236982	2.234546	2.235764
iterates	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	
x*	2.235155	2.23485	2.235003	2.235079	2.235117	2.235098	2.235088	2.235084	2.235081	2.235082	2.235083	

TABLE VII: Ex. 3: Bisection with a.init = mme, b.init = mme, 4 steps were needed

iterates	**1**	**2**	**3**	**4**
x*	2.505	1.2575	1.88125	2.193125

TABLE VIII: Ex. 3: Iterative search for α

iteration	alpha_i	E(X)_alpha_i
1	1.5	0.6
2	1.6	0.6153846
3	1.7	0.6296296
4	1.8	0.6428571
5	1.9	0.6551724
6	2	0.6666667
7	2.1	0.6774194
8	2.2	0.6875
9	2.3	0.6969697
10	2.4	0.7058824
11	2.5	0.7142857
12	2.6	0.7222222
13	2.7	0.7297297
14	2.8	0.7368421
15	2.9	0.7435897

TABLE IX: Ex. 3: Newton with graphical estimator $\alpha_{MME.graphical} = 2.2$

iterates	1	2	3	4
x*	2.2	2.234532	2.235083	2.235083

TABLE X: Ex. 3: Newton with $\alpha_{MME.mean} = 2.1$

iterates	1	2	3	4	5
x*	2.1	2.226919	2.235053	2.235083	2.235083

TABLE XI: Ex. 3: Newton with $\alpha_{MME.E} = \frac{\bar{x}}{1-\bar{x}}$

iterates	1	2	3	4	5
x*	2.087544	2.225344	2.23504	2.235083	2.235083

TABLE XII: Ex. 3: Newton with α_{MME} = epsilon

iterates	1	2
x*	1e-06	2e-06

TABLE XIII: Ex. 3: Newton with α_{MME} = 5

iterates	1	2
x*	5	-1.185267

TABLE XIV: Ex. 3: Scoring Fisher with $\alpha_{MME.graphical}$ = 2.2

iterates	1	2	3	4
x*	2.2	2.234532	2.235083	2.235083

TABLE XV: Ex. 3: Scoring Fisher with $\alpha_{MME.mean}$ = 2.1

iterates	1	2	3	4	5
x*	2.1	2.226919	2.235053	2.235083	2.235083

TABLE XVI: Ex. 3: Scoring Fisher with $\alpha_{MME.E} = \frac{\bar{x}}{1-\bar{x}}$

iterates	1	2	3	4	5
x*	2.087544	2.225344	2.23504	2.235083	2.235083

TABLE XVII: Ex. 3: SF with α_{MME} = epsilon

iterates	1	2
x*	1e-06	2e-06

TABLE XVIII: Ex. 3: SF with α_{MME} = 5

iterates	1	2
x*	5	-1.185267

TABLE XIX: Ex. 4: Iterative search for λ

iteration	alpha_i	E(X)_alpha_i
1	1	1
2	1.1	0.9090909
3	1.2	0.8333333
4	1.3	0.7692308
5	1.4	0.7142857
6	1.5	0.6666667
7	1.6	0.625
8	1.7	0.5882353
9	1.8	0.5555556
10	1.9	0.5263158
11	2	0.5
12	2.1	0.4761905
13	2.2	0.4545455
14	2.3	0.4347826
15	2.4	0.4166667

TABLE XX: Ex. 4: Newton with $\lambda_{MME.graphical} = 2$

iterates	1	2	3	4	5	6
x*	2	1.568626	1.6416	1.645153	1.645161	1.645161

TABLE XXI: Ex. 4: Newton with $\lambda_{MME.mean} = 1.7$

iterates	1	2	3	4	5
x*	1.7	1.643333	1.645159	1.645161	1.645161

TABLE XXII: Ex. 4: Newton with $\lambda_{MME.E} = \frac{1}{\bar{x}}$

iterates	1	2
x*	1.645161	1.645161

TABLE XXIII: Ex. 4: Newton with λ_{MME} = 5

iterates	1	2
x*	5	-5.196085

TABLE XXIV: Ex. 4: Newton with λ_{MME} = epsilon

iterates	1	2	3	4	5	6	7	8	9
x*	1e-06	1.999999e-06	3.999996e-06	7.999983e-06	1.599993e-05	3.19997e-05	6.399877e-05	0.0001279951	0.0002559802
iterates	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
x*	0.0005119205	0.001023682	0.002046726	0.004090907	0.00817164	0.01630269	0.03244383	0.06424785	0.1259866
iterates	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
x*	0.2423252	0.448957	0.7753956	1.185332	1.516637	1.63512	1.645099	1.645161	1.645161

TABLE XXV: Ex. 4: Reparametrization with $\lambda_{MME.graphical} = 2$

iterates	**1**	**2**	**3**	**4**	**5**	**6**	**7**
b_it	0.6931472	1.094254	1.460681	1.624474	1.644900	1.645161	1.645161
lambda_it	2.0000000	2.986954	4.308895	5.075749	5.180494	5.181842	5.181842

TABLE XXVI: Ex. 4: Reparametrization with $\lambda_{MME.mean} = 1.7$

iterates	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
b_it	0.5306283	0.8901083	1.298627	1.572167	1.641922	1.645154	1.645161	1.645161
lambda_it	1.7000000	2.4353933	3.664261	4.817077	5.165087	5.181809	5.181842	5.181842

TABLE XXVII: Ex. 4: Reparametrization with $\lambda_{MME.E} = \frac{1}{\bar{x}}$

iterates	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
b_it	0.497838	0.8450265	1.256010	1.553110	1.640010	1.645144	1.645161	1.645161
lambda_it	1.645161	2.3280394	3.511384	4.726146	5.155222	5.181758	5.181842	5.181842

TABLE XXVIII: Ex. 4: Reparametrization with λ_{MME} = 5

iterates	**1**	**2**	**3**	**4**
b.it	1.609438	1.644385	1.64516	1.645161
lambda.it	5.000000	5.177824	5.18184	5.181842

TABLE XXIX: Ex. 4: Reparametrization λ_{MME} = epsilon

iterates	**1**	**2**	**3**
b.it	-13.815511	-1.436491e+02	-12830.18
lambda.it	0.000001	4.111487e-63	0.00