Computational Stats Deliverable 2

António Coelho, Gonçalo Arsénio, Sara Vigário, Tiago dos Santos

2018-11-09

Contents

1 Problem 1

Consider the following sample:

```
c(7.0,3.5,11.9,8.9,10.1,1.2,1.1,7.9,12.9,1.3,5.2,5.1,3.9,2.5,10.4,6.2,-3.9)
```

1.1 Normal variance MLE

Assume that the given sample originated from a random variable with a normal distribution whose parameter σ^2 is unknown. Use the sample to determine the maximum-likelihood estimator of the parameter σ^2.

Maximum normal distribution Likelihood estimation.

Assuming that our sample follows a normal distribution as said on the exercise question, we can say that the probability density function follows this format

$$f_X(x_j) = (2.\pi.\sigma_0^2).e^{-\frac{1}{2}.\frac{(xj-\mu)^2}{\sigma_0^2}}$$

And to optain the maximum likelihood estimation, we need to partially derivate the likelihood in order to the two parameters, μ and σ^2, and discover the zeros.

$$L(\mu, \sigma^2, x_1, ..., x_n) = \prod_{j=1}^{n} f_X(x_j; \mu, \sigma^2)$$

$$= \prod_{j=1}^{n} (2\pi\sigma^2)^{-1/2}.e^{-\frac{1}{2}.\frac{(xj-\mu)^2}{\sigma^2}}$$

$$= (2\pi\sigma^2)^{-n/2}.e^{-\frac{1}{2\sigma^2}\sum_{j=1}^{n}(xj-\mu)^2}$$

Likelihood function:

$$l(\mu,\sigma^2;x_1,...,x_n) = -\frac{n}{2}ln(2\pi) - \frac{n}{2}ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{n}(x_j-\mu)^2$$

Derivates equals to zero

$$\frac{\partial}{\partial\mu}l(\mu,\sigma^2;x_1,...,x_n) = 0$$

$$\frac{\partial}{\partial\mu}l(\mu,\sigma^2;x_1,...,x_n) = \frac{\partial}{\partial\mu}\left(-\frac{n}{2}ln(2\pi) - \frac{n}{2}ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{n}(x_j-\mu)^2\right) = \frac{1}{\sigma^2}(\sum_{j=1}^{n}x_j - n\mu)$$

$$\frac{1}{\sigma^2}(\sum_{j=1}^{n}x_j - n\mu) = 0 \equiv \left(\sum_{j=1}^{n}x_j - n\mu\right) = 0$$

Then:

$$\hat{\mu} = \frac{1}{n}\left(\sum_{j=1}^{n}x_j\right)$$

$$\frac{\partial}{\partial\sigma^2}l(\mu,\sigma^2;x_1,...,x_n) = 0$$

$$\frac{\partial}{\partial\sigma^2}l(\mu,\sigma^2;x_1,...,x_n) = \frac{\partial}{\partial\mu}\left(-\frac{n}{2}ln(2\pi) - \frac{n}{2}ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{n}(x_j-\mu)^2)\right) = \frac{1}{2\sigma^2}\left[\frac{1}{\sigma^2}\sum_{j=1}^{n}(x_j-\mu)^2 - n\right]$$

$$\frac{1}{2\sigma^2}\left[\frac{1}{\sigma^2}\sum_{j=1}^{n}(x_j-\mu)^2 - n\right] = 0 \equiv \frac{1}{\sigma^2}\sum_{j=1}^{n}(x_j-\mu)^2 - n = 0$$

Then:

$$\widehat{\sigma^2} = \frac{1}{n}\sum_{j=1}^{n}(x_j-\mu)^2$$

```r
sample<-c(7.0,3.5,11.9,8.9,10.1,1.2,1.1,7.9,12.9,1.3,5.2,5.1,3.9,2.5,10.4,6.2,-3.9)

norm.maximLikelihoodEst.var <- function(sample){
  n <- length(sample)
  sample.mean <- mean(sample)
  sum((sample - sample.mean)^2)/n
}

norm.maximLikelihoodEst.var(sample)
```

```
## [1] 18.84941
```

1.2 Jackknife bias and variance

Determine the Jackknife bias and variance of the estimator obtained in the item (a), and discuss the quality of the estimator.

We want to estimate the bias and variability of the estimator, $\theta = t(F)$. Since a good estimator not sensible to the sample is often of interest to know it faces minor pertubations in F. We can define:

$$L_t(y;F) = \lim_{\epsilon \to 0} \frac{t[(1-\epsilon)F + \epsilon H_y] - t(F)}{\epsilon}$$

$$H_y = \begin{cases} 1, u \leq y \\ 0, u \geq y \end{cases} \tag{2.0}$$

as the influence fuction of t at F.

Using (2.0) with the empirical function we get the empirical influence function:

$$I(y) = L_t(y; \hat{F}) \tag{2.1}$$

Applying an extension of Taylor's Theorem to $t(\hat{F})$:

$$t(\hat{F}) \simeq t(F) + \frac{1}{n}\sum_{j=1}^{n} I_j \tag{2.2}$$

So:

$$\theta - \hat{\theta} = -\frac{1}{n}\sum_{j=1}^{n} I_j \tag{2.3}$$

If we take $\epsilon = -\frac{1}{n-1}$ we get a distribution with no weight on the point x_j and weight $\frac{1}{n-1}$ on the rest of the sample, $\hat{F_{-j}}$. In practice this is having a sample of size $n-1$ by omitting x_j from the original sample.

So, the jackknife aproximation to the empirical influence value I_j is:

$$I_{jack;j} = (n-1)[t(\hat{F}) - t(\hat{F}_{-j})] = (n-1)(\theta - \hat{\theta}_{-j})$$

Consenquently:

$$b_{jack} = -\frac{1}{n}\sum_{j=1}^{n} I_{jack;j} \tag{2.1}$$

$$Var_{jack} = \frac{1}{n(n-1)}(\sum_{j=1}^{n} I^2_{jack;j} - nb^2_{jack}) \tag{2.2}$$

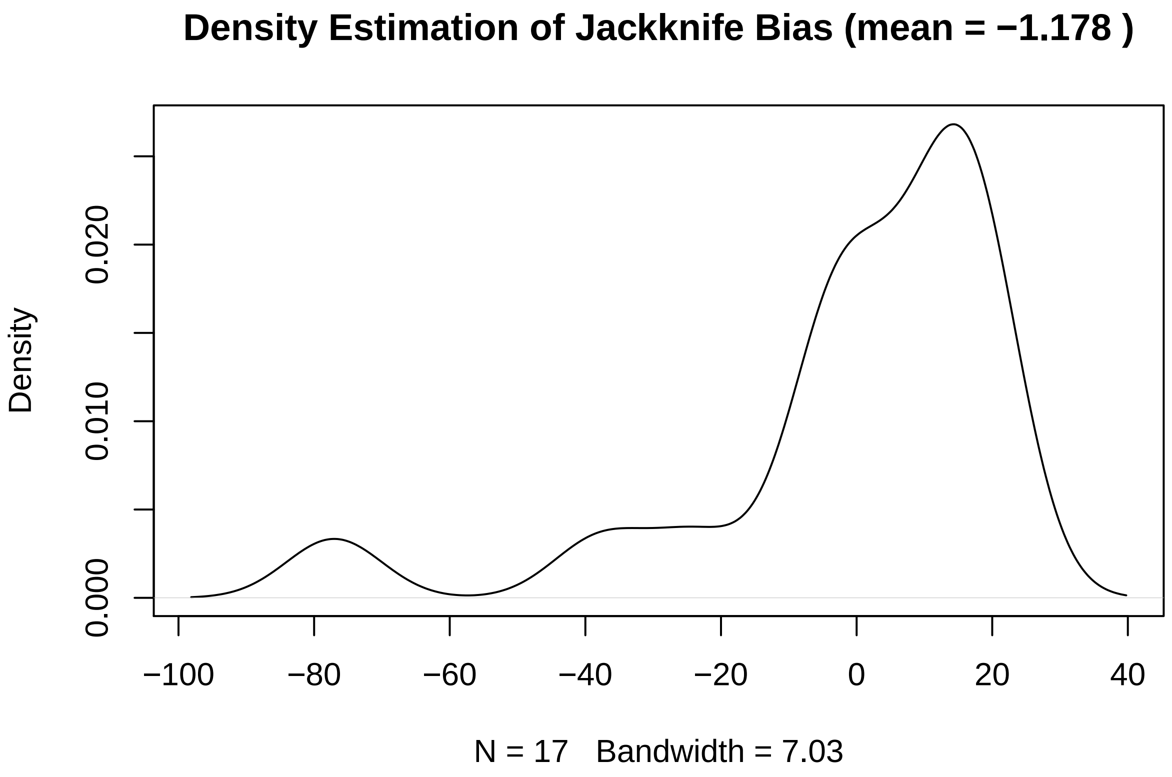
```
sample<-c(7.0,3.5,11.9,8.9,10.1,1.2,1.1,7.9,12.9,1.3,5.2,5.1,3.9,2.5,10.4,6.2,-3.9)
n <- length(sample)
sample.mean <- mean(sample)


ljack <- function(idx, sample, n){
  (n-1)*(norm.maximLikelihoodEst.var(sample) - norm.maximLikelihoodEst.var(sample[-idx]))
}

bias.jackknife <- -mean(sapply(1:n, ljack, sample, n))
bias.jackknife
```

```
## [1] -1.178088
```

```r
ljack.bias.distribution <- sapply(1:n, ljack, sample, n )

variance.jackknife <- 1/(n*(n-1))*sum(sapply(1:n,function(idx,sample,n){
  ljack(idx,sample,n)^2 - n*(bias.jackknife^2)
},sample,n))
variance.jackknife
```

```
## [1] 35.37843
```

```r
plot(
  density(-ljack.bias.distribution)
  , main=paste0("Density Estimation of Jackknife Bias (mean = ",round(bias.jackknife,3)," )")
)
```



1.3 Kolmogorov-Smirnov Test

Consider the null hypothesis H_0: The sample was generated from a random variable with a normal distribution with parameters $(\mu, \sigma^2) = (5, 5)$. Use the test statistic's empirical distribution function to estimate the Kolmogorov-Smirnov test statistic's p-value. Explain why the test statistic is adequate considering the type of null hypothesis we are trying to test.

Hypothesis testing whose hypothesis are:

$$H_0 : X \sim \mathcal{N}(5, 5) H_1 : X \nsim \mathcal{N}(5, 5) \tag{1.0}$$

Since we dont know the probability distribution function of the test statistic, $T(X) = D$, we will use the empirical distribution function under the null hypothesis, H_0, $\hat{F}(x)_{H_0}$,

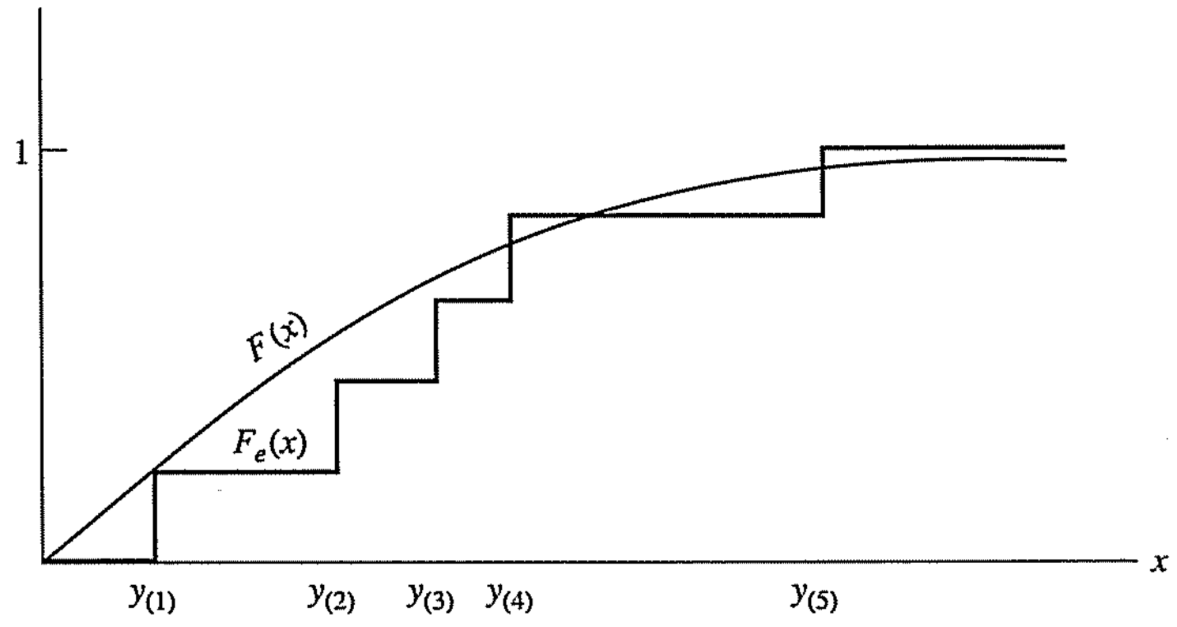


Figure 1: Graph showing the distances between continuous cdf and discrete empirical cdf (Ross, page 222, 4th edition)

$$\hat{F}(x)_{H_0} = \frac{\#\{i : x_i \leq x\}}{n} \tag{1.1}$$

with x_i being one of n observations of the sample

To do this, we need to use the Monte Carlo method.

So, we generate m samples of the random variable, X under H_0, $X \sim \mathcal{N}(5, 5)$, and, for each sample, compute the test statistic:

$$x_1 : x_1^1, ..., x_n^1 \rightarrow t_1 = T(x^1)...x_m : x_1^m, ..., x_n^m \rightarrow t_m = T(x^m) \tag{1.2}$$

To compare the samples with the reference probability distribution, $\mathcal{N}(5, 5)$, we will use the Kolmogorov-Smirnov statistic.

In short, this statistic determines the degree of similarity between two given probability distributions through distance. This distance, D, is the the largest distance between the two given cumulative distribution functions.

Being $F(x)$ the cumulative distribution function of the known distribution and $y(j)$ the discontinuity points of $\hat{F}(x)_{H_0}$:

$$\begin{aligned} D &= \max_x |\hat{F}(x)_{H_0} - F(x)| \\ &= \max_x\{\max_x\{\hat{F}(x)_{H_0} - F(x)\}, \max_x\{\hat{F}(x) - F(x)_{H_0}\}\} \\ &= \max_{j=1,...,n}\{\frac{j}{n} - F(y(j)), F(y(j)) - \frac{j-1}{n}\} \end{aligned} \tag{1.3}$$

This is well ilustrated in figure 1, where since both functions are monotonically increasing, a maximum distance D will only occur in the discontinuity points of $\hat{F}(x)_{H_0}$ or $y(j)$.

As previously stated, D is applied to every Monte Carlo sample. So, appling (1.3) to (1.1) we get:

$$
\begin{gathered}
T(x^1) = D_1 = \max_{j=1,\ldots,n} \{\frac{j}{n} - F(y_1(j)), F(y_1(j)) - \frac{j-1}{n}\} \\
\ldots \\
T(x^2) = D_2 = \max_{j=1,\ldots,n} \{\frac{j}{n} - F(y_2(j)), F(y_2(j)) - \frac{j-1}{n}\}
\end{gathered} \tag{1.4}
$$

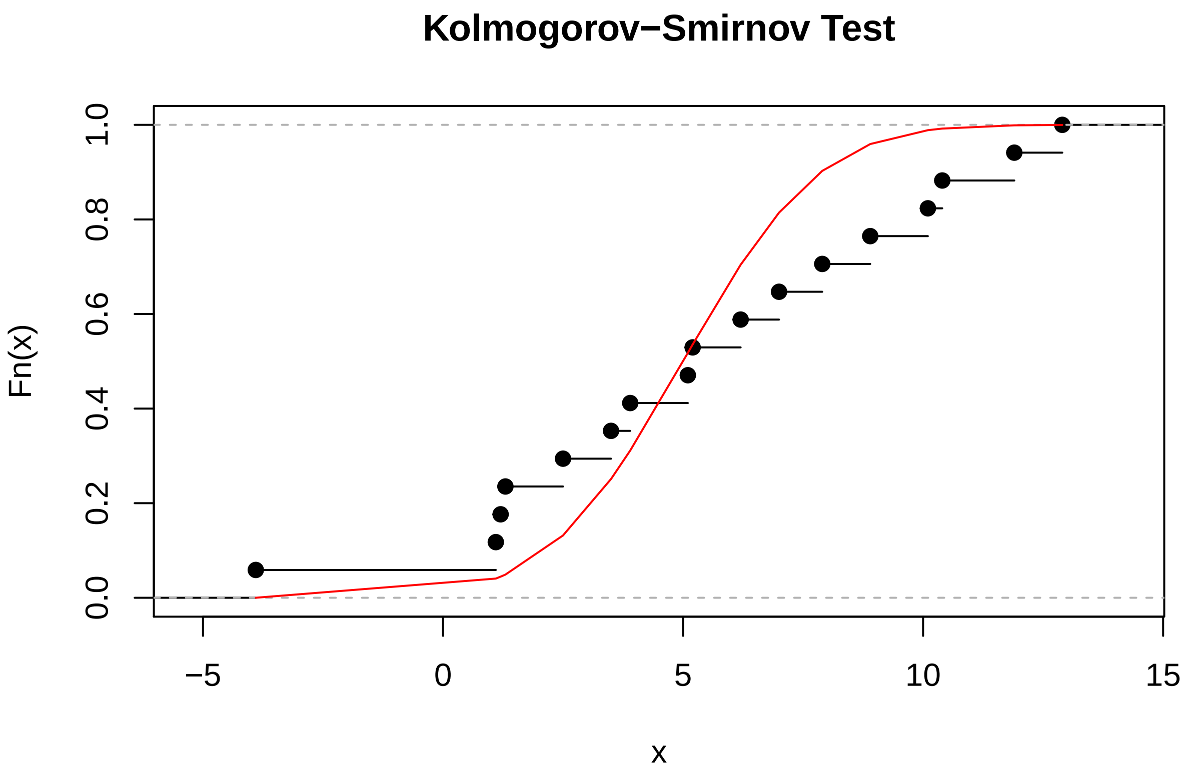
and to the original sample:

$$
T(x) = d = \max_{j=1,\ldots,n} \{\frac{j}{n} - F(y(j)), F(y(j)) - \frac{j-1}{n}\} \tag{1.5}
$$

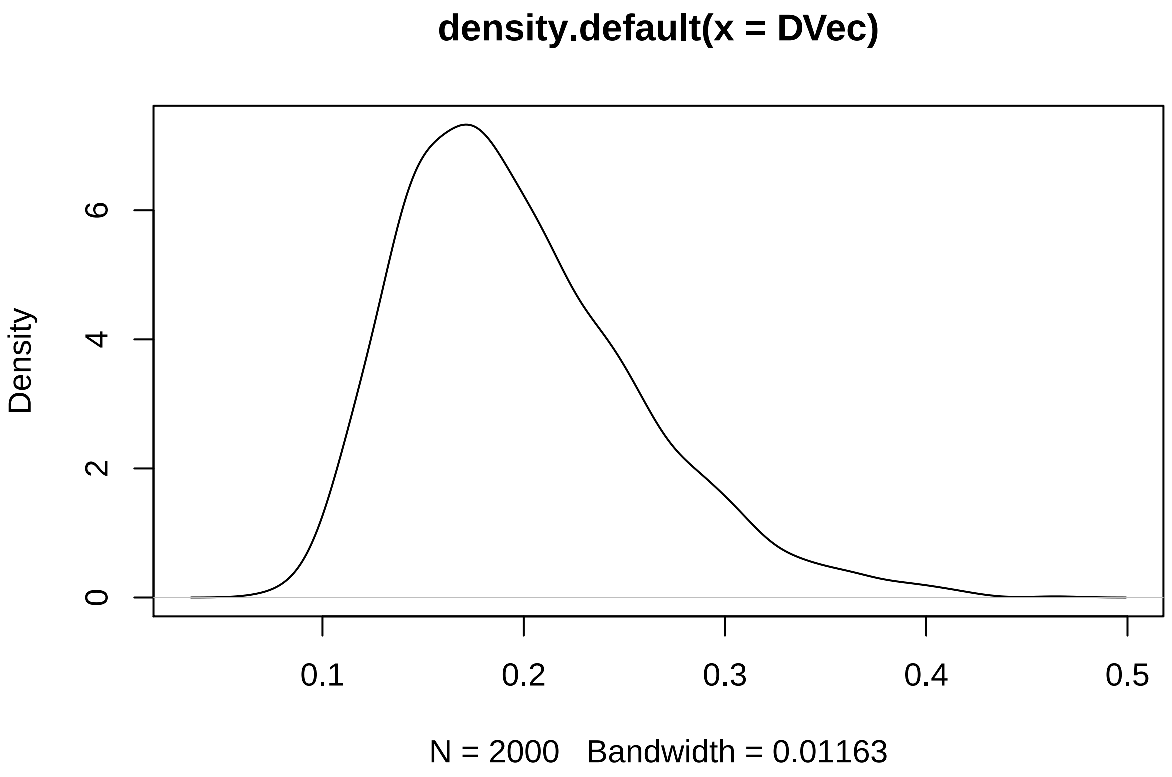
Finally, the $p - \hat{value}$ is estimated:

$$
p - \hat{value} = P(D \geq d|H_0) = \frac{\#\{D_k : D_k \geq d\}}{n} \tag{1.6}
$$

```
KS_statDist<-function(sample,mean,sd)
{
  sample<-sort(sample)
  n<-length(sample)
  DVecA<-rep(0, n)
  DVecB<-rep(0, n)
  for(j in c(1:n))
  {
    DVecA[j]<-(j/n)-pnorm(sample[j],mean,sd)
    DVecB[j]<-pnorm(sample[j],mean,sd)-((j-1)/n)
  }
  DVecAA <- sapply(1:n,function(idx){(j/n)-pnorm(sample[j],mean,sd)})
  DVecBB <- sapply(1:n,function(idx){pnorm(sample[j],mean,sd)-((j-1)/n)})
  return(max(c(DVecA,DVecB)))
}

#Kolmogorov-Smirnov test
sample.sort <- sort(sample,decreasing = F)
d<-KS_statDist(sample.sort,5,sqrt(5))

#plot
EmpCDF<-ecdf(sample.sort)
NormCDF<-pnorm(sample.sort,5,sqrt(5))
plot(EmpCDF,main="Kolmogorov-Smirnov Test")
lines(sample.sort,NormCDF,col="red")
```



```r
MCsampleNum<-2000
m<-length(sample.sort)
DVec<-rep(0, m)
pValueNumerator<-0
for (k in c(1:MCsampleNum))
{
  #D calculation of the generated MC samples
  DVec[k]<-KS_statDist(rnorm(m,5,sqrt(5)),5,sqrt(5))
  if(DVec[k]>=d)
    pValueNumerator<-pValueNumerator+1
}

DVec.df <- data.frame(
  pts = DVec
)
library(ggplot2)
p <- ggplot2::ggplot(DVec.df, aes(x=pts)) +
  geom_density()
plot(density(DVec))
```



```r
#p-value calculation
pValue<-(pValueNumerator+1)/(MCsampleNum+1)
pValue
```

```
## [1] 0.1649175
```

2 Problem 2

2.1 Exponential parameter MLE

Assume that the given sample originated from a random variable with a exponential distribution whose parameter λ is unknown. Use the sample to determine the maximum-likelihood estimator of the parameter λ

Maximum normal distribution Likelihood estimation.

Assuming that our sample follows a exponencial distribution as said on the exercise question, we can say that the probability density function follows this format

$$f(x;\lambda) = \lambda e^{-\lambda x}$$

And to optain the maximum likelihood estimation, we need to derivate the likelihood function in order to λ and discover the zeros.

$$L(\lambda, x_1, ..., x_n) = \prod_{j=1}^{n} f_X(x_j;\lambda)$$

$$= \prod_{j=1}^{n} \lambda e^{-\lambda x_j}$$

$$= \lambda e^{-\lambda \sum_{j=1}^{n} x_j}$$

$$l(\lambda; x) = n\ ln(\lambda) - \lambda(\sum_{j=1}^{n} x_j)$$

$$\frac{d}{d\lambda} l(\lambda; x) = 0 \equiv \frac{n}{\lambda} - \sum_{j=1}^{n} x_j = 0 \equiv \lambda = \frac{1}{\frac{\sum_{j=1}^{n} x_j}{n}}$$

which means

$$\hat{\lambda} = \frac{1}{\overline{X}}$$

```r
sample <- c(0.05,0.03,0.19,0.14,0.12,0.03,0.08,0.19,0.07,0.01,0.24,0.10,0.03,0.31)

exp.maximLikelihoodEst.lambda <- function(sample){
  1/mean(sample)
}

exp.maximLikelihoodEst.lambda.estimated <- exp.maximLikelihoodEst.lambda(sample)
exp.maximLikelihoodEst.lambda.estimated
```

```
## [1] 8.805031
```

2.2 Bootstrap bias and variance

Determine the Bootstrap bias and variance of the estimator obtained in the item (a).

Viés:

$$E(\hat{\Psi} - \Psi) \approx b_{boot}(\hat{\Psi}) = \frac{1}{R} . \sum_{r=1}^{R} .(\hat{\Psi}_r^* - \hat{\Psi})$$

```r
R <- 999
set.seed(1)
bias.bootstrap <- mean(sapply(1:R, function(idx,sample,sample.psi){
  sample.bootstrap <- rexp(length(sample),sample.psi)
  sample.bootstrap.psi <- exp.maximLikelihoodEst.lambda(sample.bootstrap)

  sample.bootstrap.psi - sample.psi
},sample,exp.maximLikelihoodEst.lambda.estimated))
bias.bootstrap
```

```
## [1] 0.7442572
```

$$Var(\hat{\Psi} - \Psi) \approx Var_{boot} = \frac{1}{R-1} \sum_{r=1}^{R} *(\hat{\Psi}_r^* - \hat{\Psi}^*)^2$$

```r
set.seed(1)

bootstrap.var <- function(sample,R){
  estimated.bootstrap <- sapply(1:R, function(idx,sample,sample.psi){
    exp.maximLikelihoodEst.lambda(rexp(length(sample),sample.psi))
  },sample,exp.maximLikelihoodEst.lambda.estimated)
  sum((estimated.bootstrap - mean(estimated.bootstrap))^2)/(R-1)
}

variance.bootstrap <- bootstrap.var(sample,R)
variance.bootstrap
```

```
## [1] 8.171176
```

2.3 Bootstrap confidence interval

Construct a basic Bootstrap confidence interval for λ

Confidence interval

$$P(a_\alpha < \hat{\theta} - \theta < a_{1-\alpha})$$

The confidence Interval can be describe like this: $]\hat{\theta} - a_{1-\alpha}, \hat{\theta} - a_\alpha[$

But,as we do not know the distribution of $\hat{\theta} - \theta$ we will use the empirical distribution function of $\hat{\theta^*} - \hat{\theta}$ to estimate a_α and $a_{1-\alpha}$

That being said: $\hat{a_\alpha} = \theta^*_{((R+1)\alpha)} - \hat{\theta}$

$a_{\hat{\alpha-1}} = \theta^*_{((R+1)(\alpha-1))} - \hat{\theta}$

```
set.seed(1)

boot.ci <- function(sample, R, alphaG, exp.maximLikelihoodEst.lambda.estimated, variance.bootstrap){
  alpha <- alphaG/2
  distance.booststrap <- sapply(1:R, function(idx,sample,sample.phi){
    sample.bootstrap <- rexp(length(sample),sample.phi)
    sample.bootstrap.phi <- exp.maximLikelihoodEst.lambda(sample.bootstrap)

    sample.bootstrap.phi - sample.phi
  },sample,exp.maximLikelihoodEst.lambda.estimated)

  distance.booststrap <- c(distance.booststrap,0)
  distance.booststrap.sorted <- sort(distance.booststrap, decreasing = F)

  ahat_left <- distance.booststrap.sorted[(R+1)*alpha]
  ahat_right <- distance.booststrap.sorted[(R+1)*(1-alpha)]

  sample.bootstraps.phi <- distance.booststrap+exp.maximLikelihoodEst.lambda.estimated
  ci_left <- exp.maximLikelihoodEst.lambda.estimated-ahat_right
  ci_right <- exp.maximLikelihoodEst.lambda.estimated-ahat_left

  return(
    list(
      sample.bootstraps.phi = sample.bootstraps.phi,
      ci_left = ci_left,
      ci_right = ci_right
    )
  )
}
```

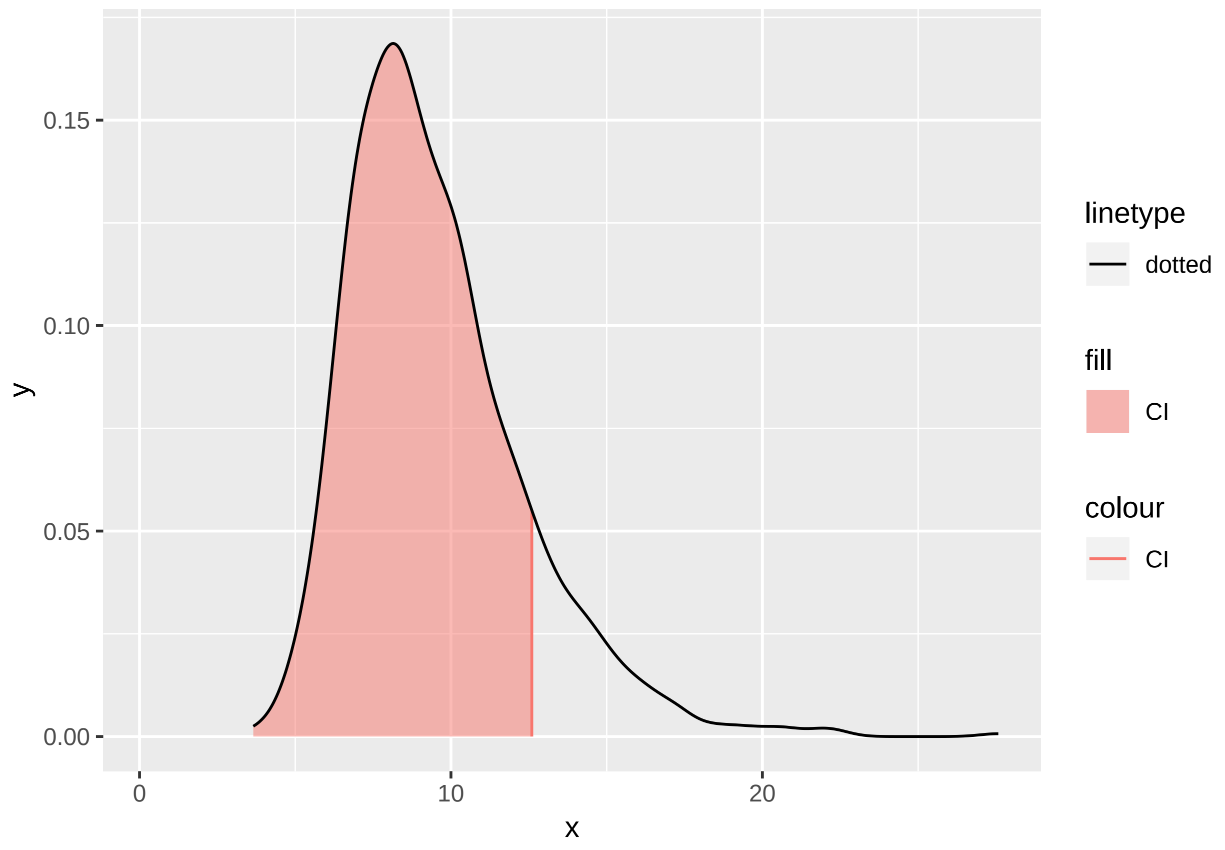


Figure 2: Confidence intervals of 0.025 in the estimate pdf of the estimator λ

```
boot_cis <- boot.ci(sample, R, 0.025, exp.maximLikelihoodEst.lambda.estimated, variance.bootstrap)
sample.bootstraps.phi <- boot_cis$sample.bootstraps.phi
```

```
## left value: 0.204453
## right value: 12.595924
```

2.4 Studentized-Bootstrap confidence interval

Construct a Studentized-Bootstrap confidence interval for λ (Use the Bootstrap variance estimator to estimate the Bootstrap samples's variance).

Studentized-Bootstrap confidence Interval can be describe like this:

$$] \ \hat{\theta} - z^*_{((R+1)(1-\alpha))}\sqrt{v} \ , \ \hat{\theta} - z^r_{((R+1)\alpha)}\sqrt{v} \ [$$

The pivolal statistic used is the following:

$$\hat{z} = \frac{\hat{\theta} - \theta}{\sqrt{var(\hat{\theta} - \theta)}}$$

assuming that $var(\hat{\theta} - \theta)$ can be estimated by $Var_{boot}(\hat{\theta})$ then

$$\hat{z}_r^* = \frac{\hat{\theta}_r^* - \hat{\theta}}{\sqrt{v_r^*}}$$

This method can be more accurate when compared to the basic bootstrap interval, while being more computational demanding[1].

```r
R <- 999
set.seed(1)
studentized.ci <- function(sample, R,alphaG,exp.maximLikelihoodEst.lambda.estimated,variance.bootstrap)
  alpha <- alphaG/2

  distance.booststrap.studantized <- sapply(1:R, function(idx,sample,sample.phi){

    sample.bootstrap <- rexp(length(sample),sample.phi)
    sample.bootstrap.phi <- exp.maximLikelihoodEst.lambda(sample.bootstrap)

    v <- bootstrap.var(sample.bootstrap,R)

    (sample.bootstrap.phi - sample.phi)/sqrt(v)
  },sample,exp.maximLikelihoodEst.lambda.estimated)

  distance.booststrap.studantized <- c(distance.booststrap.studantized,0)
  distance.booststrap.studantized.sorted <- sort(distance.booststrap.studantized, decreasing = F)

  ahat.stdzd_left <- distance.booststrap.studantized.sorted[(R+1)*alpha]*sqrt(variance.bootstrap)
  ahat.stdzd_right <- distance.booststrap.studantized.sorted[(R+1)*(1-alpha)]*sqrt(variance.bootstrap)


  ci_left.stdzd <- exp.maximLikelihoodEst.lambda.estimated-ahat.stdzd_right
  ci_right.stdzd <- exp.maximLikelihoodEst.lambda.estimated-ahat.stdzd_left
  return(
    list(
      ci_left = ci_left.stdzd,
      ci_right = ci_right.stdzd
    )
  )
}
studentized.ci.values <- studentized.ci(sample, R, 0.025, exp.maximLikelihoodEst.lambda.estimated, varia
```

```
## left value: 0.190189
## right value: 12.633102
```

2.5 Discussion

Discuss the results of items (c) and (d). Choose $\alpha = 0.025$ as the significance level for the confidence intervals.

The values in these confidence intervals are pretty close, with the estimated value obtain in 1.1. From [1], the studentized-bootstrap can be more accurate when comparing to the bootstrap, but in our simulation this was not evident:

- range of bootstrap: 12.391
- range of studentized-bootstrap: 12.443

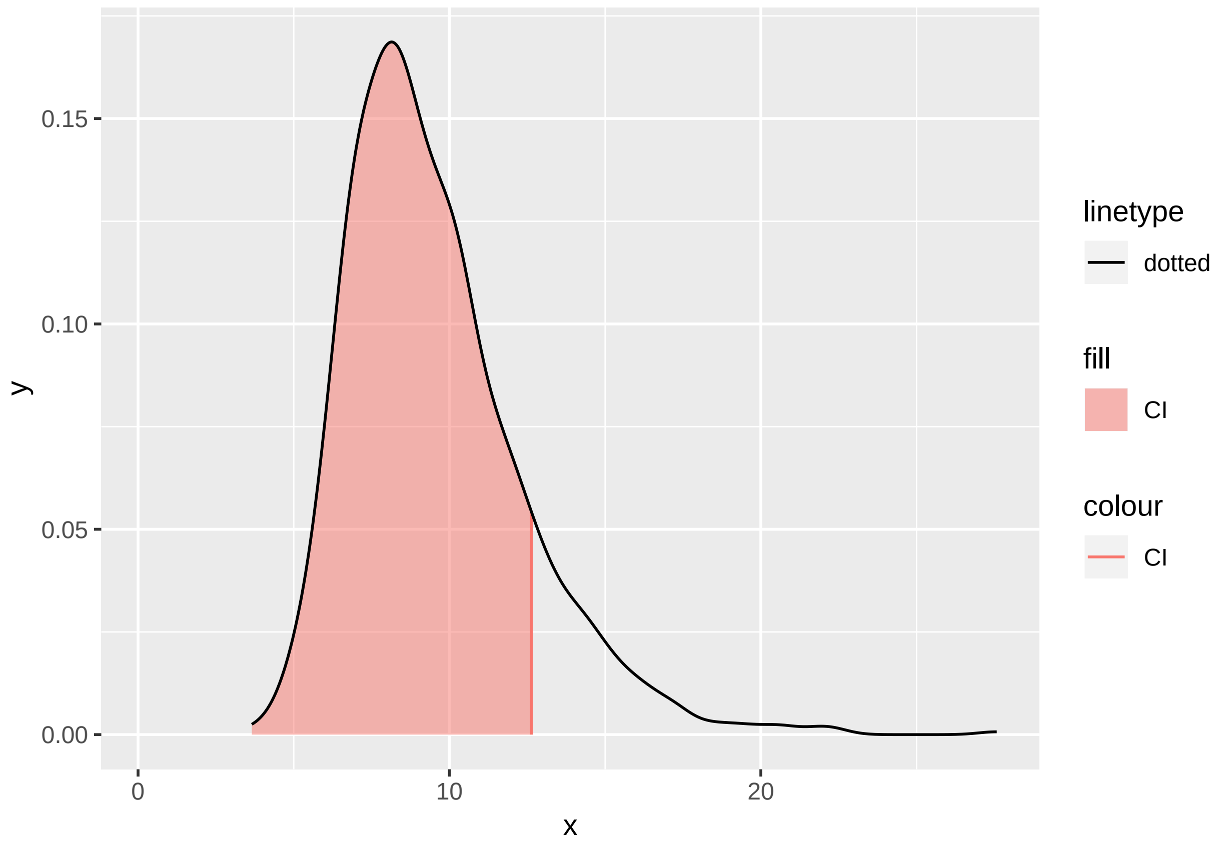


Figure 3: Studentized confidence intervals of 0.025 in the estimate pdf of the estimator λ

But by doing more simulations to assess this statement we can check this:

```
set.seed(100)

ranges <- sapply(1:100,function(idx){
  variance.bootstrap <- bootstrap.var(sample,R)
  boot_cis <- boot.ci(sample, R, 0.025, exp.maximLikelihoodEst.lambda.estimated, variance.bootstrap)
  std_cis <- studentized.ci(sample, R, 0.025, exp.maximLikelihoodEst.lambda.estimated, variance.bootstra

  return(
    list(
      range.boot = boot_cis$ci_right - boot_cis$ci_left,
      range.boot.t = std_cis$ci_right - std_cis$ci_left
    )
  )
})

ranges.unlist <- unlist(ranges)
```

```
boot.mean <- mean(ranges.unlist[seq(1,length(ranges.unlist),2)])
boot.t.mean <- mean(ranges.unlist[seq(2,length(ranges.unlist),2)])
```

boot.mean	boot.t.mean
12.35	12.22

Given the results, we can see that, in average, the Studentized-Bootstrap can indeed be more accurate than the Boostrap, but by a small value. This small value in difference can be because of two things:

1. Due to the small size of the sample (maybe they are not representative, which would inject a lot of uncertainty)
2. Due to the characteristics of the data. The Studentized Boostrap is not always more accurate than the Boostrap, in spite of there are cases where it is.

References

[1] Ryan Tibshirani. *The Bootstrap: Advanced Methods for Data Analysis (36-402/36-608), Spring 2014*. Carnegie Mellon University. 2014. URL: http://www.stat.cmu.edu/~ryantibs/advmethods/notes/bootstrap.pdf.