Project LSTAT2185

Mathieu Graf and Victor Dujardin

Exercise 1.

Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be the function $f(x) = \frac{exp(x)}{1+exp(x)}$

In order to derive the third order Taylor expansion, we need the three derivatives of $f(x)$.

$$f'(x) = \frac{e^x}{(1+e^x)^2}$$

$$f''(x) = \frac{e^x(1-e^x)}{(1+e^x)^3}$$

$$f^{(3)}(x) = \frac{(e^{2x} - 4e^x + 1)e^x}{(1+e^x)^4}$$

Using the formula :

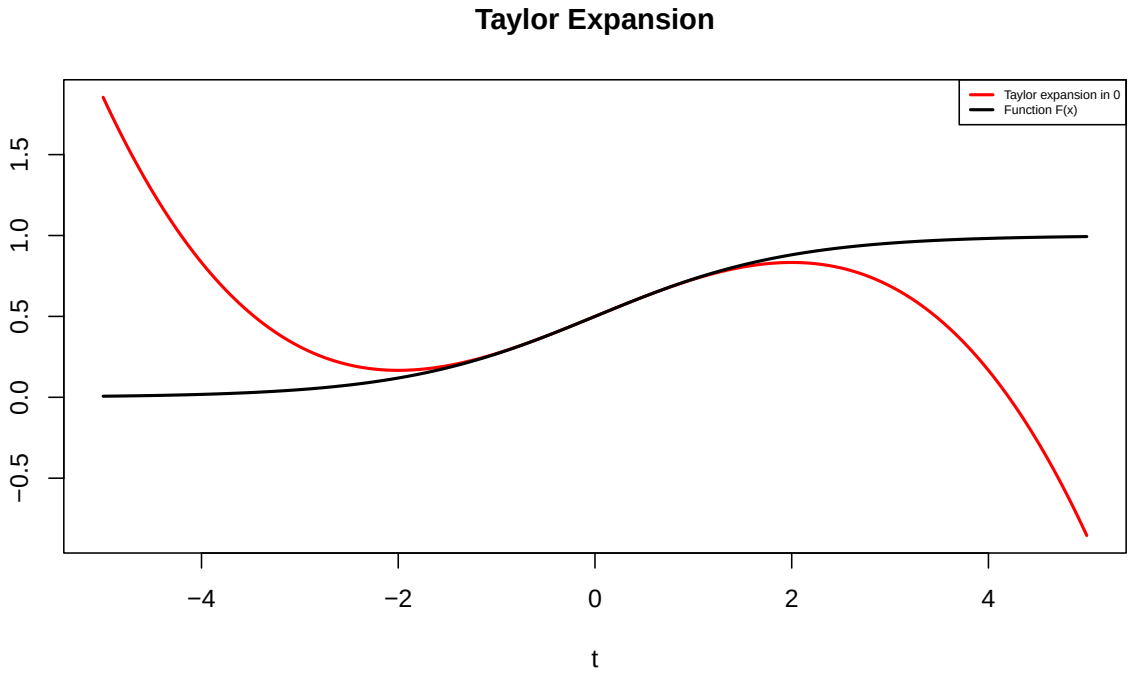
$$f(x) \approx T(x) = \sum_{k=0}^{K} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

The Taylor expansion of f around the point $x = 0$ is

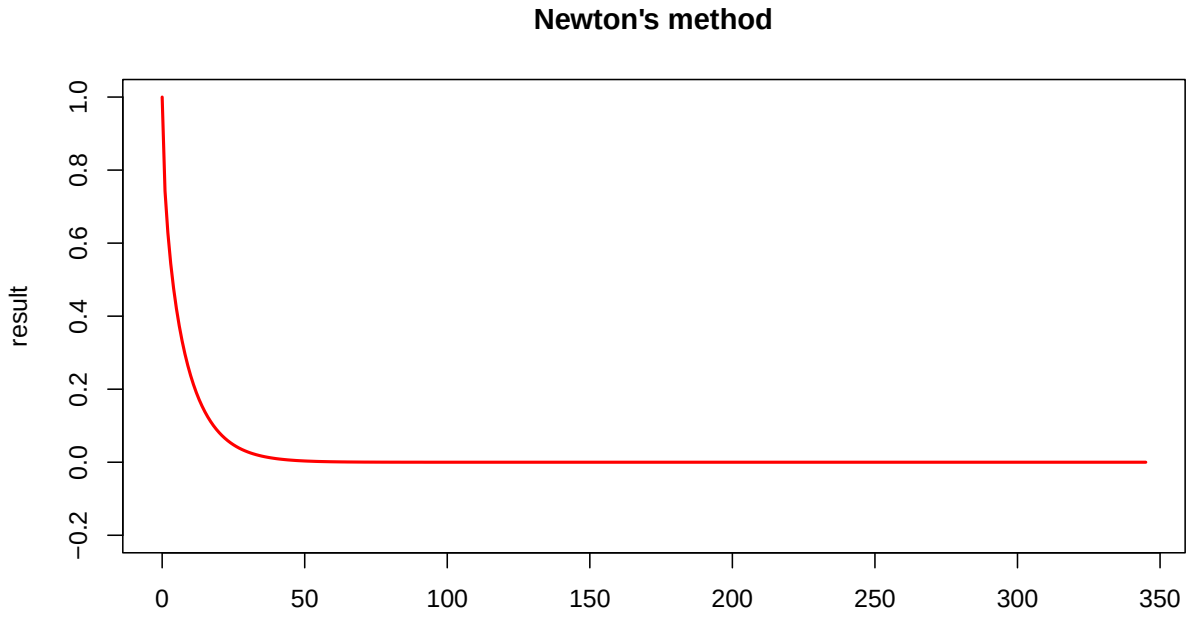
$$T(x) = \frac{e^0}{1+e^0} + \frac{e^0}{(1+e^0)^2}(x-0) - \frac{(e^0-1)e^0}{(1+e^0)^3}\frac{(x-0)^2}{2!} + \frac{(e^{20} - 4e^0 + 1)e^0}{(1+e^0)^4}\frac{(x-0)^3}{3!}$$

$$= \frac{1}{2} + \frac{1}{4}x - \frac{1}{48}x^3$$

Here we can see the Taylor expansion in red which approximate the real function f.



We want to find the maximum of this function by implementing a Newton algorithm where $\theta_{t+1} = \theta_t - \frac{f'(x)}{f''(x)}$.




```
## [1] "The optimum is a maximum and there are 346 iteration and the optimum is 1.0322836008352e-16"
```

The algorithm converges to zero.

Exercise 2.

Poisson model

In order to derive the log-likelihood function of a poisson model we should start from its pmf which is

$$P(Y = y) = \frac{\lambda^y e^{-\lambda}}{y!}$$

We can derive the likelihood function

$$L(\lambda|y_1, ..., y_{240}) = \prod_{i=1}^{240} \frac{\lambda^{y_i} e^{-\lambda}}{y_i!}$$

The log-likelihood function of a poisson model is

$$LL(\lambda|y_1, ..., y_{240}) = ln(\lambda) \sum_{i=1}^{240} y_i - 240\lambda - \sum_{i=1}^{240} ln(y_i!)$$

Since the third term does not depend of λ we can get rid of it and optimize for the first two terms.

```
## [1] "optim gives us : max = 0.358336339066652 | obj =  -174.261079932635"
```

Here we can see that the optimal value of λ is 0.358.

Quasi-poisson model 1

The quasi-Poisson model's pmf is given as

$$P(Y = y) = \begin{cases} p + (1-p)\frac{\lambda^y e^{-\lambda}}{y!} & \text{if } y = 0 \\ (1-p)\frac{\lambda^y e^{-\lambda}}{y!} & \text{if } y \geq 1 \end{cases}$$

The likelihood and log-likelihood functions are

$$L(\lambda, p|y_1, ..., y_{240}) = (p + (1-p)\frac{\lambda^0 e^{-\lambda}}{0!})^{182} \times ((1-p)\frac{\lambda^1 e^{-\lambda}}{1!})^{41} \times ((1-p)\frac{\lambda^2 e^{-\lambda}}{2!})^{12} \times ((1-p)\frac{\lambda^3 e^{-\lambda}}{3!})^2$$
$$\times((1-p)\frac{\lambda^4 e^{-\lambda}}{4!})^2 \times (1-p)\frac{\lambda^7 e^{-\lambda}}{7!}$$

$$LL(\lambda, p|y_1, ..., y_{240}) = 182ln((p+(1-p)\frac{\lambda^0 e^{-\lambda}}{0!}))+41ln((1-p)\frac{\lambda^1 e^{-\lambda}}{1!})+12ln((1-p)\frac{\lambda^2 e^{-\lambda}}{2!})+2ln((1-p)\frac{\lambda^3 e^{-\lambda}}{3!})$$
$$+2ln((1-p)\frac{\lambda^4 e^{-\lambda}}{4!}) + ln((1-p)\frac{\lambda^7 e^{-\lambda}}{7!})$$

As in the last model, terms which does not include any of the parameters, lambda or p, are not relevant since we derive with respect to lambda and p. In our case, $\frac{d(-\ln(y_i!))}{d\lambda} = \frac{d(-\ln(y_i!))}{dp} = 0$

```
##      lambda         p     value iteration gradient convergence
## 1 0.8472774 0.5770769 -163.6544        29       10           0
```

```
##      lambda         p     value iteration gradient convergence
## 1 0.8472769 0.5770766 -163.6544       178       29           0
```

The optimal values for λ and p are 0.85 and 0.58.

Quasi-poisson model 2

The quasi-Poisson model's pmf is given as

$$P(Y=y)=\begin{cases} p & \text{if } y=0 \\ (1-p)\frac{\lambda^y e^{-\lambda}}{y!(1-e^{-\lambda})} & \text{if } y\geq 1 \end{cases}$$

The likelihood and log-likelihood functions are

$$L(\lambda,p|y_1,...,y_{240})=p^{182}\times((1-p)\frac{\lambda^1 e^{-\lambda}}{(1-e^{-\lambda})1!})^{41}\times((1-p)\frac{\lambda^2 e^{-\lambda}}{(1-e^{-\lambda})2!})^{12}\times((1-p)\frac{\lambda^3 e^{-\lambda}}{(1-e^{-\lambda})3!})^{2}\times((1-p)\frac{\lambda^4 e^{-\lambda}}{(1-e^{-\lambda})4!})^{2}$$
$$\times(1-p)\frac{\lambda^7 e^{-\lambda}}{(1-e^{-\lambda})7!}$$

$$LL(\lambda,p|y_1,...,y_{240})=182ln(p)+41ln((1-p)\frac{\lambda^1 e^{-\lambda}}{1!(1-e^{-\lambda})})+12ln((1-p)\frac{\lambda^2 e^{-\lambda}}{2!(1-e^{-\lambda})})+2ln((1-p)\frac{\lambda^3 e^{-\lambda}}{3!(1-e^{-\lambda})})$$
$$+2ln((1-p)\frac{\lambda^4 e^{-\lambda}}{4!(1-e^{-\lambda})})+ln((1-p)\frac{\lambda^7 e^{-\lambda}}{7!(1-e^{-\lambda})})$$

```
##      lambda         p     value iteration gradient convergence
## 1 0.8472784 0.7583324 -163.6544        37       10           0
```

```
##       lamda         p     value iteration gradient convergence
## 1 0.8472775 0.7583333 -163.6544       438       95           0
```

The optimal values for λ and p are :

$$Quasi-Poisson1=\begin{cases} \lambda=0.85 \\ p=0.76 \end{cases}$$

$$Quasi-Poisson2=\begin{cases} \lambda=0.85 \\ p=0.58 \end{cases}$$

In this study, we applied two quasi-Poisson models to a dataset in order to estimate the probability of observing a count of 0 (p) and the intensity parameter (lambda). The first model defines p as the probability of observing a count of 0, while the second model defines p as the probability of observing any count less than 1. These different definitions of p led to different estimates for p in the two models, even though the estimates for lambda were similar.

To verify our results, we attempted to solve the system of equations by hand. However, we encountered an equation of the form $xe^x = 1$ that we were unable to solve analytically. As a result, we had to rely on numerical methods to obtain a solution. Despite this challenge, we trust the validity of our numerical results.

Exercise 3.

Question 1.

One possible estimator for the coefficient of variation of X, denoted as θ, is the sample coefficient of variation. The sample coefficient of variation is defined as the ratio of the sample standard deviation to the sample mean, and is calculated as follows: $\hat{\theta} = S/\bar{X} = \frac{\sqrt{\sum_i (X_i - \bar{X})^2 n}}{\sqrt{(n-1)}\sum_i X_i}$

Therefore, to estimate θ using the method of moments, we can calculate the sample mean and sample standard deviation of the sample X1, ..., Xn, and then use these estimates to calculate the sample coefficient of variation $\hat{\theta}$.

Question 2.

Firstly we can estimate the bias and the variance of our estimator via

$$Bias^*(T) = Bias_{F_n}(T) = E_{F_n}T^2(X^*) - \theta(F_n)$$

and

$$Var^*(T) = Var_{F_n}(T) = E_{F_n}T^2(X^*) - (E_{F_n}T(X^*))^2$$

```
## [1] "Bias: -0.008"
```

```
## [1] "Variance: 0.002"
```

We now give a confidence interval for θ using a traditional asymptotic method, the basic bootstrap method, the percentile bootstrap method and the t-bootstrap method.

$$\theta \in \left[\hat{\theta} - z_{1-\frac{\alpha}{2}}\hat{\sigma}, \hat{\theta} - z_{\frac{\alpha}{2}}\hat{\sigma}\right]$$

```
## [1] "Confidence interval for the assymptotic method: 0.219 0.464"
```

$$\theta \in \left[\hat{\theta} - u^*(1-\frac{\alpha}{2}), \hat{\theta} - u^*(\frac{\alpha}{2})\right]$$

```
## [1] "Confidence interval for the basic bootstrap method: 0.258 0.438"
```

$$\theta \in \left[\hat{\theta} + u^*(\frac{\alpha}{2}), \hat{\theta} + u^*(1-\frac{\alpha}{2})\right]$$

```
## [1] "Confidence interval for the percentile bootstrap method: 0.245 0.424"
```

$$\theta \in \left[\theta - n^{-1/2}\hat{s}_{1-\alpha/2}, \hat{\theta} - n^{-1/2}s\hat{y}_{\alpha/2}\right]$$

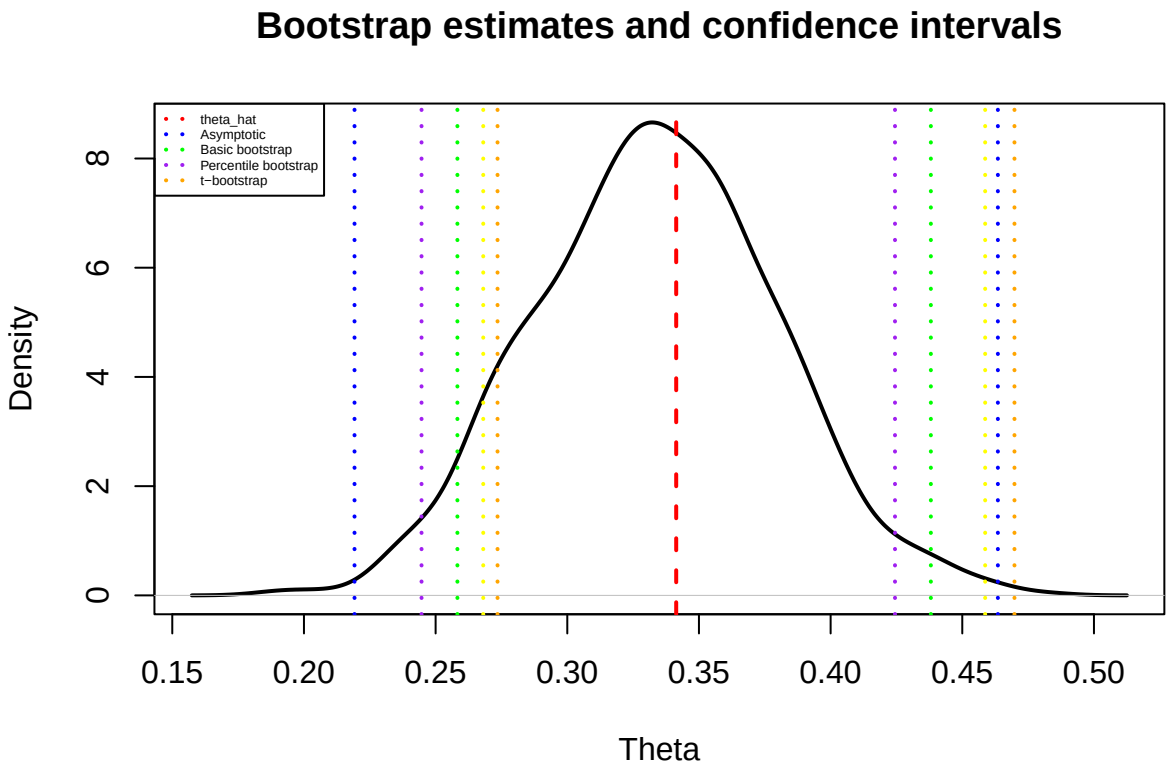
where $\hat{y}(a)$ is the a-quantile of $\hat{K}(x)$ the empirical bootstrap distribution of $U^* = \sqrt{n}(\frac{(\hat{\theta}^* - \hat{\theta})}{S^*})$

```
## [1] "Confidence interval for the t-bootstrap method: 0.274 0.47"
```

We will aslo use an iterated bootstrap procedure for the t-bootstrap method.

```
## [1] "Confidence interval for the iterated t-bootstrap method: 0.268 0.459"
```

Here is a plot of all the confidence interval :



In this analysis, we evaluated the performance of four different methods for constructing confidence intervals for the estimator theta: traditional asymptotic method, basic bootstrap, percentile bootstrap, and iterative bootstrap. Our results showed that all of the confidence intervals enclose the true value of theta, indicating that each method is able to provide valid confidence intervals. In the next step of our analysis, we will compare the performance of these different methods by examining the length of the confidence intervals and the coverage probability, which should be close to the nominal level of 95%. This will allow us to determine the most appropriate method for constructing confidence intervals in this context.

Question 3.

We can perform a hypothesis test based on a bootstrap procedure for the parameter of interest, θ.

The first test is :

$$\begin{cases} H_0 : \theta \geq 1/3 \\ H_1 : \theta < 1/3 \end{cases}$$

```
## [1] "p-value : 0.633366633366633"
```

```
## [1] "Fail to reject null hypothesis"
```

The second test is :

$$\begin{cases} H_0 : \theta \geq 0.75 \\ H_1 : \theta < 0.75 \end{cases}$$

```
## [1] "p-value : 0.000999000999000999"
```

```
## [1] "Reject null hypothesis"
```

These two tests are consistent with a visual analysis of the theta estimator. From the plots, it is clear that the theta estimator is consistently around 1/3. The results of the statistical tests further support the visual analysis of the theta estimator, as both tests indicate that the null hypothesis of the theta estimator being either superior to 1/3 or to 3/4 can be respectively rejected and not rejected. This suggests that the theta estimator is significantly different from zero, and is therefore an important predictor of the response variable. It is interesting to use these tests to reaffirm or confirm previous results, as they provide a more rigorous and quantitative approach to data analysis. Overall, the results of these tests support the conclusion that the theta estimator is a reliable predictor of the response variable.

Question 4.

In order to evaluate the performance of different methods for obtaining confidence intervals, we conducted a Monte Carlo study using samples of size n drawn B times, where $n \in \{10, 50, 100, 500\}$. We used the coverage probability and the width of the interval as measures of evaluation.

```
## [1] "             n = 3              n = 10            n = 20             n = 30
```

```
##                length coverage    length coverage     length coverage
## As.Normal   0.3632655    0.915 0.1807893    0.985 0.12911499    0.988
## Basic Boot. 0.2290878    0.799 0.1249469    0.915 0.09173132    0.926
## Perc.Boot.  0.2290878    0.674 0.1249469    0.892 0.09173132    0.920
## t-Boot.     0.3826387    0.919 0.1355990    0.933 0.09518367    0.937
##                 length coverage
## As.Normal   0.05827017    0.981
## Basic Boot. 0.04266203    0.944
## Perc.Boot.  0.04266203    0.936
## t-Boot.     0.04293546    0.942
```

It can be observed that each confidence interval (CI) improves with n. For n=30, the basic bootstrap has the smallest size and the closest coverage to 95%. The percentile bootstrap has similar results. However, for small n, the iterative bootstrap method is very effective compared to the others. The asymptotic method, on the other hand, is too wide and covers too many estimators, exceeding the expected 95% confidence.

Appendix.

$$f(x) = \frac{exp(x)}{1 + exp(x)}$$

$$f'(x) = (\frac{e^x}{(1+e^x)})' = \frac{(e^x)'(1+e^x) - e^x(1+e^x)'}{(1+e^x)^2} = \frac{e^x(1+e^x) - e^{2x}}{(1+e^x)^2} = \frac{e^x + e^{2x} - e^{2x}}{(1+e^x)^2} = \frac{e^x}{(1+e^x)^2}$$

$$f''(x) = (\frac{e^x}{(1+e^x)^2})' = \frac{(e^x)'(1+e^x)^2 - e^x((1+e^x)^2)'}{(1+e^x)^4} = \frac{e^x(1+e^x)^2 - e^x 2(1+e^x)e^x}{(1+e^x)^4}$$

$$= \frac{e^x(1+e^x) - e^x 2e^x}{(1+e^x)^3} = \frac{e^x + e^{2x} - 2e^{2x}}{(1+e^x)^3} = \frac{e^x - e^{2x}}{(1+e^x)^3} = \frac{e^x(1-e^x)}{(1+e^x)^3}$$

$$f'''(x) = (\frac{e^x(1-e^x)}{(1+e^x)^3})' = \frac{(e^x(1-e^x))'(1+e^x)^3 - e^x(1-e^x)((1+e^x)^3)'}{(1+e^x)^6} = \frac{(e^x - 2e^{2x})(1+e^x)^3 - e^x(1-e^x)3(1+e^x)^2 e^x}{(1+e^x)^6}$$

$$= \frac{(e^x - 2e^{2x})(1+e^x) - e^x(1-e^x)3e^x}{(1+e^x)^4} = \frac{e^x - 2e^{2x} + e^{2x} - 2e^{3x} - 3e^{2x} + 3e^{3x}}{(1+e^x)^4}$$

$$= \frac{e^{3x} - 4e^{2x} + e^x}{(1+e^x)^4} = \frac{e^x(e^{2x} - 4e^x + 1)}{(1+e^x)^4}$$

```
f = function(x){
  return(exp(x) / (1 + exp(x)))
}

df1 = function(x){
  return(exp(x)/(1 + exp(x))^2 )
}

df2 = function(x){
  return(-(exp(x)*(exp(x) - 1))/(1 + exp(x))^3)
}


df3 = function(x){
  return((exp(x)*(-4*exp(x) + exp(2*x) + 1))/(exp(x) + 1)^4)
}

taylor_exp = function(x){
  return((0.5) * 1 + (0.25) * x + (-1/48) * x^3)
}
```

```
t = seq(-5,5,0.01)
plot(t,taylor_exp(t), type = 'l', col = 'red', lwd = 2, main = 'Taylor Expansion', ylab = '')
lines(t, f(t), col = 'black', lwd = 2 )
legend("topright", col = c('red', 'black'), lty = c(1,1), lwd = c(2,2),
       cex = 0.5, legend = c("Taylor expansion in 0","Function F(x)"))
```

```
Newton_method = function(x){
  j = 0
  i0 = x
  i = 0
  result = c(i0)
  time = c(0)
  while(j != 1){
    i = i + 1
    i1 = i0 - df2(i0)/df3(i0)*0.1
    result = c(result,i1)
    time = c(time,i)
    if(i1 == i0){
      j = 1
    }
    else {
      j = 0
      i0 = i1
    }
  }
  data.frame(time,result)
  plot(time, result, type = 'l', ylim = c(-0.2,1),
       col = 'red', lwd = 2, main = "Newton's method", xlab = '')
  optimum <- result[length(time )-1]
  if (df2(optimum) > 0) {
    print(paste("The optimum is a minimum and there are",length(time), "iteration and the optimum is",
  } else {
    print(paste("The optimum is a maximum and there are",length(time), "iteration and the optimum is",
  }
}

x0 = 1
Newton_method(x0)
```

```
mouvements_nb = c(0,1,2,3,4,5,6,7)
counts = c(182,41,12,2,2,0,0,1)
X =  c(rep(0, 182), rep(1, 41), rep(2,12), rep(3, 2), rep(4,2), c(7))

lambda_hat = t(mouvements_nb)%*%counts/240
print(paste("lambda_hat =",lambda_hat))
# 2.1
LL1 = function(lambda, X) { log(lambda) * sum(X)-length(X) * lambda}
res = optimize(f = LL1, X = X, lower = 0.001, upper = 99, maximum =TRUE)
paste("optim gives us : max =",res$maximum,"| obj = ", res$objective)
```

```
#2.2
X2 =  c(rep(1, 41), rep(2,12), rep(3, 2), rep(4,2), c(7))

LL2 = function(para){
  lambda = para[1]
  p = para[2]
  return(length(X2) * log(1-p) + sum(X2*log(lambda)) - lambda * length(X2)+
           182* log(p + (1-p) * exp(-lambda)))}
```

```
BFGS = optim(par =c(0.5,0.5), fn = LL2, method = "BFGS",control =list(fnscale =-1))
BFGS_res = c(BFGS$par,BFGS$value, BFGS$counts, BFGS$convergence)
BFGS_res = data.frame(matrix(BFGS_res, nrow = 1))
colnames(BFGS_res) = c('lambda', 'p', 'value', 'iteration','gradient','convergence')
BFGS_res

CG = optim(par =c(0.5,0.5), fn = LL2, method = "CG", control =list(fnscale =-1))
CG_res = c(CG$par,CG$value, CG$counts, CG$convergence)
CG_res = data.frame(matrix(CG_res, nrow = 1))
colnames(CG_res) = c('lambda', 'p', 'value', 'iteration','gradient','convergence')
CG_res
```

```
#2.3

LL3 = function(para){
  lambda = para[1]
  p = para[2]

  return( length(X2) * log(1-p) + sum(X2 * log(lambda)) - lambda  * length(X2)
          - length(X2)* log(1- exp(-lambda)) + 182 * log(p)  )
}


BFGS = optim(par = c(0.5,0.5), fn = LL3, method = "BFGS",control =list(fnscale =-1))
BFGS_res = c(BFGS$par,BFGS$value, BFGS$counts, BFGS$convergence)
BFGS_res = data.frame(matrix(BFGS_res, nrow = 1))
colnames(BFGS_res) = c('lambda', 'p', 'value', 'iteration','gradient','convergence')
BFGS_res

CG = optim(par = c(0.5,0.5), fn = LL3, method = "CG",control =list(fnscale =-1))
CG_res = c(CG$par,CG$value, CG$counts, CG$convergence)
CG_res = data.frame(matrix(CG_res, nrow = 1))
colnames(CG_res) = c('lamda', 'p', 'value', 'iteration','gradient','convergence')
CG_res
```

```
a = 9
c = 1
n = 30
X = rgamma(n = n, shape = a, scale = c)




theta_hat = ((n-1)/n*sd(X))/mean(X)



theta_boot = c()
sigmastar = c()

for (i in 1:1000){
  samp = sample(x = as.matrix(X), size = n, replace = TRUE)
  theta_boot = append(theta_boot, ((n-1)/n*sd(samp)/mean(samp)))
```

```
  sigmastar = append(sigmastar, sd(samp))
}

Bias= mean(theta_boot) -theta_hat
variance = var(theta_boot)

print(paste("Bias:", round(Bias,3)))
print(paste("Variance:", round(variance,3)))
```

```
born_inf_norm = theta_hat - qnorm(p = 0.975)* theta_hat/sqrt(n)
born_sup_norm =theta_hat + qnorm(p = 0.975)*theta_hat/sqrt(n)
print(paste("Confidence interval for the assymptotic method:",
            c(round(born_inf_norm,3)),round(born_sup_norm,3)))
```

```
born_inf_basic = theta_hat - quantile(theta_boot-theta_hat, probs = 0.975)
born_sup_basic = theta_hat - quantile(theta_boot-theta_hat, probs = 0.025)
print(paste("Confidence interval for the basic bootstrap method:",
            c(round(born_inf_basic,3)),round(born_sup_basic,3)))
```

```
born_inf_perc = theta_hat  + quantile(theta_boot-theta_hat, probs = 0.025)
born_sup_perc = theta_hat  + quantile(theta_boot-theta_hat, probs = 0.975)
print(paste("Confidence interval for the percentile bootstrap method:",
            c(round(born_inf_perc,3)),round(born_sup_perc,3)))
```

```
U = sqrt(n) * (theta_boot-theta_hat)/sigmastar
born_inf_t = theta_hat-sd(X) *quantile(U, probs = 0.975)/sqrt(n)
born_sup_t = theta_hat-sd(X) *quantile(U, probs = 0.025)/sqrt(n)
print(paste("Confidence interval for the t-bootstrap method:",
            c(round(born_inf_t,3)),round(born_sup_t,3)))
```

```
T <- function(X) {
  return(sd(X) / mean(X))
}


bootstrap <- function(X) {
  return(sample(X, replace = TRUE))
}




B1 <- 2000

T_star <- numeric(B1)


for (b1 in 1:B1) {

  X_star <- bootstrap(X)
```

```
  T_star[b1] <- T(X_star)
}


B2 <- 200


var_star_star <- numeric(B1)

for (b1 in 1:B1) {

  T_star_star <- numeric(B2)

  for (b2 in 1:B2) {

    X_star_star <- bootstrap(X_star[b1])


    T_star_star[b2] <- T(X_star_star)
  }

  var_star_star[b1] <- var(T_star_star)
}

sigma_star <- sqrt(n * var(T_star) / B1)



W_star <- sqrt(n) * (T_star - T(X)) / sigma_star




lower <- T(X) - quantile(W_star, p = 1 - 0.05 / 2, na.rm = TRUE) * sigma_star / sqrt(n)
upper <- T(X) - quantile(W_star, p = 0.05 / 2, na.rm = TRUE) * sigma_star / sqrt(n)
print(paste("Confidence interval for the iterated t-bootstrap method:",
            c(round(lower,3)),round(upper,3)))
```

```
plot(density(theta_boot), main = "Bootstrap estimates and confidence intervals",
     xlab = "Theta", ylab = "Density", lwd = 2)
abline(v = theta_hat, lty = 2, col = "red", lw = 2)
abline(v = c(born_inf_norm, born_sup_norm), lty = 3, col = "blue", lw = 2)
abline(v = c(born_inf_basic, born_sup_basic), lty = 3, col = "green", lw = 2)
abline(v = c(born_inf_perc, born_sup_perc), lty = 3, col = "purple", lw = 2)
abline(v = c(born_inf_t, born_sup_t), lty = 3, col = "orange", lw = 2)
abline(v = c(lower, upper), lty = 3, col = 'yellow', lw = 2)
legend("topleft", c("theta_hat","Asymptotic", "Basic bootstrap", "Percentile bootstrap", "t-bootstrap")
       lty = c(3, 3, 3, 3, 3, 3),lw = c(2,2,2,2,2,2), col = c("red", "blue", "green", "purple", "orange
```

```
set.seed(1234)
a= 9
c=1
n = 30
```

```r
X = rgamma(n = n, shape = a, scale = c)

null_hypothesis <- 1/3
alternative_hypothesis <- "!="


theta_hat <- ((n - 1) / n * sd(X)) / mean(X)
Xtilde = X-mean(X)+3*sd(X)

B <- 1000
bootstrap_samples <- lapply(1:B, function(i) sample(x = Xtilde, size = n, replace = TRUE))



theta_estimators <- lapply(bootstrap_samples, function(bootstrap_sample) {
  mean_bootstrap_sample <- mean(bootstrap_sample)
  var_bootstrap_sample <- var(bootstrap_sample)
  theta_estimator <- sqrt(var_bootstrap_sample) / mean_bootstrap_sample
  return(theta_estimator)})



p_value = 0
for(i in 1:length(bootstrap_samples)){
  if (theta_estimators[i]<theta_hat){p_value  = p_value+1}}
p_value = (p_value +1)/(B+1)
print(paste("p-value :",p_value))

if (p_value < 0.05) {
  print("Reject null hypothesis")
} else {
  print("Fail to reject null hypothesis")}
```

```r
set.seed(1234)
a= 9
c=1
n = 30
X = rgamma(n = n, shape = a, scale = c)

null_hypothesis <- 0.75
alternative_hypothesis <- "<="


theta_hat <- ((n - 1) / n * sd(X)) / mean(X)
Xtilde = X-mean(X)+4*sd(X)/3

B <- 1000
bootstrap_samples <- lapply(1:B, function(i) sample(x = Xtilde, size = n, replace = TRUE))



theta_estimators <- lapply(bootstrap_samples, function(bootstrap_sample) {
  mean_bootstrap_sample <- mean(bootstrap_sample)
```

```
  var_bootstrap_sample <- var(bootstrap_sample)
  theta_estimator <- sqrt(var_bootstrap_sample) / mean_bootstrap_sample
  return(theta_estimator)})

p_value = 0
for(i in 1:length(theta_estimators)){
  if (theta_estimators[i]<theta_hat){p_value  = p_value+1}}
p_value = (p_value +1)/(B+1)

print(paste("p-value :",p_value))


if (p_value < 0.05) {
  print("Reject null hypothesis")
} else {
  print("Fail to reject null hypothesis")}
```

```
resulttot = matrix(nrow = 4, ncol = 2,dimnames = list(c( "As.Normal", "Basic Boot.", "Perc.Boot.", "t-B
M = 1000
a= 9
c=1
n = 30

for (n in c(10,50,100,500)){
  count_t = 0
  count_norm = 0
  count_basic = 0
  count_perc = 0
  count_t_length = 0
  count_norm_length = 0
  count_basic_length = 0
  count_perc_length = 0
  for(j in 1:M){


    X =  rgamma(n = n, shape = a, scale = c)
    result = matrix(nrow = 4, ncol = 2,dimnames = list(c( "As.Normal", "Basic Boot.", "Perc.Boot.", "t-


    teta_hat = ((n-1)/n*sd(X))/mean(X)


    born_inf_norm = teta_hat -qnorm(p = 0.975)*teta_hat/sqrt(n)
    born_sup_norm =teta_hat+qnorm(p = 0.975)*teta_hat/sqrt(n)
    born_inf_norm
    born_sup_norm


    if (1/3>born_inf_norm && 1/3< born_sup_norm){count_norm = count_norm + 1}


    length_norm = born_sup_norm - born_inf_norm
```

```
count_norm_length = count_norm_length + length_norm
result[1,1] = count_norm_length/M



X_boot = c()
sigmastar = c()

for (i in 1:1000){
  samp = sample(x = as.matrix(X), size = n, replace = TRUE)
  X_boot = append(X_boot, ((n-1)/n*sd(samp)/mean(samp)))
  sigmastar = append(sigmastar, sd(samp))
}



born_inf_basic=2*teta_hat - quantile(X_boot, probs = 0.975)
born_sup_basic =2*teta_hat - quantile(X_boot, probs = 0.025)
born_inf_basic
born_sup_basic

if (1/3>born_inf_basic && 1/3< born_sup_basic){count_basic = count_basic + 1}


length_basic = born_sup_basic - born_inf_basic
count_basic_length = count_basic_length + length_basic
result[2,1] = count_basic_length/M


born_inf_perc=  quantile(X_boot, probs = 0.025)
born_sup_perc =  quantile(X_boot, probs = 0.975)
born_inf_perc
born_sup_perc

if (1/3>born_inf_perc && 1/3< born_sup_perc){count_perc = count_perc + 1}


length_perc = born_sup_perc - born_inf_perc
count_perc_length = count_perc_length + length_basic
result[3,1] = count_perc_length/M


S = sqrt(n) * (X_boot-teta_hat)/sigmastar


born_inf_t = teta_hat-sd(X) *quantile(S, probs = 0.975)/sqrt(n)
born_sup_t = teta_hat-sd(X) *quantile(S, probs = 0.025)/sqrt(n)
born_inf_t
born_sup_t


if (1/3>born_inf_t && 1/3< born_sup_t){count_t = count_t + 1}
```

```
    length_t = born_sup_t - born_inf_t
    count_t_length = count_t_length + length_t
    result[4,1] = count_t_length/M


  }


  result[1,2] = count_norm/M
  result[2,2] = count_basic/M
  result[3,2] = count_perc/M
  result[4,2] = count_t/M
  resulttot = cbind(resulttot, result)
}
print('              n = 3               n = 10            n = 20              n = 30                 ')
resulttot[,c(3:10)]
```