Advanced statistic

Anne STAFF, Bianca BOI

Description of the project

• in the first part, you will obtain the maximum likelihood estimator for the Log-logistic distribution and implement a Newton-Raphson algorithm.

• in the second part, you will study the real data set using descriptive statistics and an R-package for maximum likelihood estimation. You will also obtain bootstrap-based confidence intervals.

Maximum Likelihood estimation

We are interested in estimating the parameters of a Log-logistic distribution. $X \sim LL(\alpha, \beta)$ if X has a probability density function :

$$f_X(x) = \frac{(\frac{\beta}{\alpha})(\frac{x}{\alpha})^{\beta-1}}{[1 + (\frac{x}{\alpha})^{\beta}]^2}, x, \alpha, \beta > 0$$

We consider an i.i.d n-sample $(X_1, \ldots, X_n)$ where the X_i's arise from a Log-logistic distribution subject to random right censoring.

Corresponding statistical model

The corresponding statistical model for the given data, which follows a log-logistic distribution subject to random right censoring, is the censored log-logistic regression model.

This model allows for the estimation of the parameters of the log-logistic distribution, taking into account the censoring of some observations at a pre-specified threshold value.

The density probability is :

$$f_X(x_i; \alpha, \beta) = \frac{(\frac{\beta}{\alpha})(\frac{x_i}{\alpha})^{\beta-1}}{[1 + (\frac{x_i}{\alpha})^{\beta}]^2}, x, \alpha, \beta > 0$$

The cumulative distribution function is the following :

$$F(X_i; \alpha, \beta) = \frac{1}{1 + (\frac{X_i}{\alpha})^{\beta}}$$

The non-parametric survival function $S(t; \alpha, \beta)$ is given by:

$$S(t; \alpha, \beta) = 1 - F(t) = [1 + (\frac{t}{\alpha})^{\beta}]^{-}1$$

where t is the time, and α and β are the parameters of the log-logistic distribution.

$X \sim LL(\alpha, \beta)$ and we consider an i.i.d n-sample $(X_1, \ldots, X_n)$ where the X_i's arise from a Log-logistic distribution so the sampling hypothesis is admitted.

The c.d.f of X belongs well to a parametric family F, such that :

$$F = \{F_\theta; \theta \in \Theta\}$$

Where

$$\Theta \subseteq R^d$$

is a parameter space, so the parametric hypothesis is admitted.

$$F(\theta; \alpha, \beta) = \frac{1}{1 + (\frac{\theta}{\alpha})^{\beta}} = F(\theta'; \alpha, \beta) = \frac{1}{1 + (\frac{\theta'}{\alpha})^{\beta}} \iff \theta = \theta'$$

So the identifiability is admitted.

There exist well a measure μ on the Borel sets of R^m s.t F_θ admits a density $f(x; \theta)$ w.r.t μ. In our case μ is the Lebesgue measure on R^p because we are in the continuous case.

Log Likelihood

$$MLE(x_i) = \prod_{i=1}^{n} \frac{(\frac{\beta}{\alpha})(\frac{X_i}{\alpha})^{\beta-1}}{[1 + (\frac{X_i}{\alpha})^{\beta}]^2}$$

$$logL(Xi; \alpha, \beta) = \sum_{i=1}^{n} log\left(\frac{(\frac{\beta}{\alpha})(\frac{Xi}{\alpha})^{\beta-1}}{[1 + (\frac{Xi}{\alpha})^{\beta}]^{2}} \right)$$

$$logL(Xi; \alpha, \beta) = nlog(\beta) - n\beta log(\alpha) + (\beta - 1)\sum_{i=1}^{n} log(X_i) - 2\sum_{i=1}^{n} log[1 + (\frac{X_i}{\alpha})^{\beta}]$$

Score equations and Hessian matrix

To calculate the score equations we have to derivate the logL over α or β.

$$Score(\alpha) = \frac{\partial logL}{\partial \alpha} = -\frac{n\beta}{\alpha} + 2\sum_{i=1}^{n} \frac{(\frac{\beta}{X_i})(\frac{X_i}{\alpha})^{\beta-1}}{[1 + (\frac{X_i}{\alpha})^{\beta}]}$$

$$Score(\beta) = \frac{\partial logL}{\partial \beta} = \frac{n}{\beta} - nlog(\alpha) + \sum_{i=1}^{n} log(X_i) + 2\sum_{i=1}^{n} \frac{(\frac{X_i}{\alpha})^{\beta} log(\frac{X_i}{\alpha})}{[1 + (\frac{X_i}{\alpha})^{\beta}]}$$

Hessian Matrix:

$$\begin{pmatrix} \frac{\partial^2 logL}{\partial \alpha^2} & \frac{\partial^2 logL}{\partial \alpha \partial \beta} \\ \frac{\partial^2 logL}{\partial \beta \partial \alpha} & \frac{\partial^2 logL}{\partial \beta^2} \end{pmatrix}$$

with :

$$\frac{\partial^2 logL}{\partial \alpha \partial \beta} = \frac{\partial^2 logL}{\partial \beta \partial \alpha}$$

, we then have :

$$\begin{pmatrix} \frac{n\beta}{\alpha^2} - 2\sum_{i=1}^{n} \frac{\frac{\beta^2}{X_i^2}(\frac{X_i}{\alpha})^{\beta-1}}{[1+(\frac{X_i}{\alpha})^{\beta}]^2} & -\frac{n}{\alpha} + 2\sum_{i=1}^{n} \frac{X_i^{\beta} log(\frac{X_i}{\alpha})}{[1+(\frac{X_i}{\alpha})^{\beta}]^2} \\ -\frac{n}{\alpha} + 2\sum_{i=1}^{n} \frac{X_i^{\beta} log(\frac{X_i}{\alpha})}{[1+(\frac{X_i}{\alpha})^{\beta}]^2} & -\frac{n}{\beta^2} - 2\sum_{i=1}^{n} \frac{X_i^{\beta} log(\frac{X_i}{\alpha})}{[1+(\frac{X_i}{\alpha})^{\beta}]^2} \end{pmatrix}$$

Newton Raphson algorithm

Goal: find an approximation of the root (x axis intersection) of a function, works for real and complex functions

1. pick a random point x of the function ("first guess")
2. find the tangent to the curve at that point x
3. find the intersection of the tangent with the x axis and make it your new x
4. repeat until you're close enough (delta-y below a certain threshold epsilon)

Input: Function f(x), initial guess x0, tolerance epsilon, maximum number of iterations max_iterations

1. Set $x = x_0$
2. Set iteration = 0
3. Repeat the following steps until convergence or reaching the maximum number of iterations:
    1. Set $f\_x = f(x)$
    2. Set f_prime_x = derivative of $f(x)$
    3. Set $delta\_x = f\_x / f\_prime\_x$
    4. Set $x = x - delta\_x$
    5. Increment iteration by 1
    6. If abs($delta\_x$) < epsilon or iteration >= $max\_iterations$, exit the loop
4. Output the final approximation x as the root of the function f(x)

Newton Raphson algorithm on R

Preparation (dependencies & data):

```
library(fitdistrplus)
```

```
## Loading required package: MASS
```

```
## Loading required package: survival
```

```r
gpigs <- read.table("surv.gpigs.txt", header = T, sep = ";")
gpigs.noncensored <- gpigs[gpigs$censored == 0,]
gpigs.mc <- gpigs.noncensored$lifetime[gpigs.noncensored$regime == "M_C"]
gpigs.mc.with_censored <- gpigs$lifetime[gpigs$regime == "M_C"]
gpigs.m43 <- gpigs.noncensored$lifetime[gpigs.noncensored$regime == "M_4.3"]
```

```r
score_alpha <- function(x, alpha, beta) {
  n <- length(x)
  score <- -((n * beta) / alpha) + 2 * sum((beta / x) * ((x / alpha)^(beta - 1)) / (1 + (x / 
alpha)^beta))
  return(score)
}

score_beta <- function(x, alpha, beta) {
  n <- length(x)
  score <- (n / beta) - (n * log(alpha)) + sum(log(x)) + 2 * sum(((x / alpha)^beta) * log(x / 
alpha) / (1 + (x / alpha)^beta))
  return(score)
}

loglogistic_hessian <- function(x, n, alpha, beta) {
  hessian <- matrix(0, nrow = 2, ncol = 2)
  xi <- x
  hessian[1, 1] <- (n * beta / alpha^2) - (2 * sum((beta^2 / xi^2) * ((xi / alpha)^(beta - 
1)) / ((1 + (xi / alpha)^beta)^2)))
  hessian[2, 2] <- -(n / beta^2) - (2 * sum(- (xi^beta * log(xi / alpha)) / ((1 + (xi / alph
a)^beta)^2)))
  hessian[2, 1] <- -(n / alpha) + (2 * sum((2 * xi^beta * log(xi / alpha)) / ((1 + (xi / alph
a)^beta)^2)))
  hessian[1, 2] <- sum((2 * xi^beta * log(xi / alpha)) / ((1 + (xi / alpha)^beta)^2))
  return(hessian)
}


NR_fit_LogLogistic_hessian <- function(x, params_0, eps = 0.00001)
{
  params <- params_0
  n <- length(x)

  sumlogx <- sum(log(x))

  diff <- Inf
  alpha <- params_0[1]
  beta <- params_0[2]

  while (diff > eps)
  {
    params.old <- params
    # Gradient vector
    s <- c(score_alpha(x, alpha, beta), score_beta(x, alpha, beta))

    # Calculate Hessian matrix
    H <- loglogistic_hessian(x, n, alpha, beta)

    # Update parametres using hessian matrix
    params <- params - solve(H) %*% s
    alpha <- params[1]
    beta <- params[2]

    diff <- abs(sum(params - params.old))
  }
  list(params = params, H = H)
}

NR_fit_LogLogistic_hessian(gpigs.m43, c(153, 3))
```

```
## $params
##             [,1]
## [1,] 160.270238
## [2,]   2.914891
##
## $H
##                [,1]     [,2]
## [1,] -9.561138e-03 -1887540
## [2,] -3.775081e+06 -1887549
```

Monte-Carlo experiment

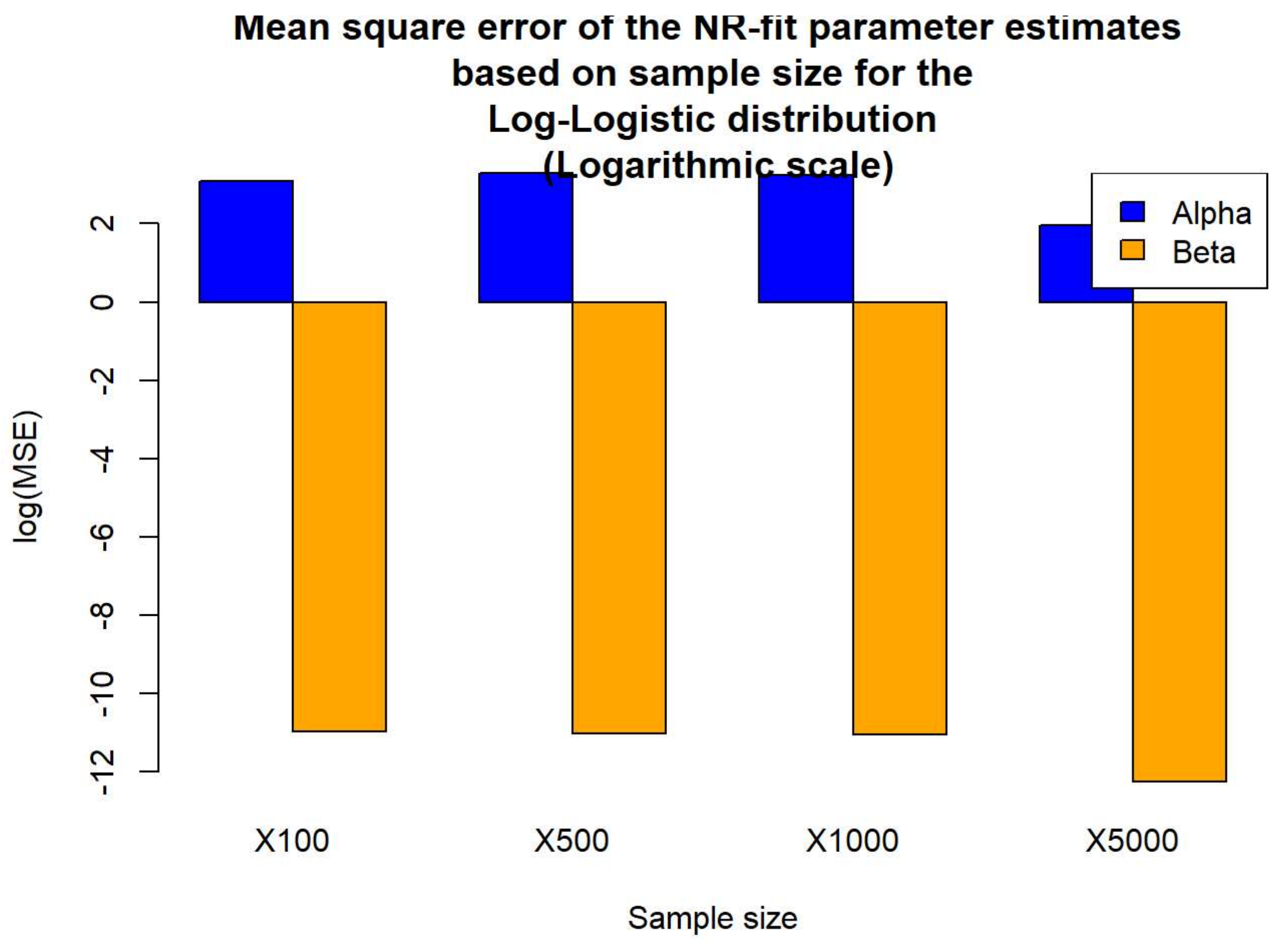
```
res <- read.table("mc_res_with_censoring_1k_iterations_452_2.csv",header=T)
res
```

	X100	X500	X1000	X5000
	<dbl>	<dbl>	<dbl>	<dbl>
1	2.199421e+01	2.687885e+01	2.550282e+01	7.139860e+00
2	1.713783e-05	1.642928e-05	1.589551e-05	4.856685e-06

2 rows

```r
rloglogis <- function(n, alpha, beta) {
  u <- runif(n)  # Generate n random numbers from a uniform distribution

  x <- alpha * ((1 / u - 1)^(-1 / beta))  # Transform the uniform random numbers using the in
verse CDF

  return(x)
}

set.seed(42)
#####################################
# some values needed for later
n_values <- c(100, 500, 1000, 5000, 10000) # sample sizes
I <- 1000 # no. of iterations
alpha0 <- 452.6
beta0 <- 2.2
#####################################
# censoring function, for a given quantile sets the max value and censors everything above
censor <- function(x, p = 0.9) {
  # for example the top 10% is everything above the 0.9 quantile
  top_ten <- quantile(x, p)
  res = x
  # censor that shit
  res[x > top_ten] = top_ten
  return(res)
}
#####################################
# the montecarlo iteration for a given sample size
mc <- function(n, I, alpha0, beta0) {
  # initialise the matrices that will track the estimates throughout the iterations
  est <- matrix(nrow = 0, ncol = 2)
  # iterate
  for (i in 1:I) {
    print(c(i, I))
    # create random data set that follows a log-logistic distribution with the given parametr
es
    dat <- rloglogis(n, alpha0, beta0)
    # censor the top 10%
    dat_censored <- censor(dat)
    est_temp <- NR_fit_LogLogistic_hessian(dat_censored, c(alpha0, beta0), 0.0001)$params
    est <- rbind(est, c(est_temp[1,1], est_temp[2,1]))
  }
  return(c(mean((est[,1] - alpha0)^2), mean((est[,2] - beta0)^2)))
}
# test_mc <- mc(100, I, alpha0, beta0)
#####################################
# here we iterated through different sample sizes but it took several hours
# so we saved the result as a csv and in further processing just load it from the file
#
# res <- matrix(nrow = 2, ncol = 0)
# for (i in 1:length(n_values)) {
#   res <- cbind(res, mc(n_values[i], I, alpha0, beta0))
# }
# colnames(res) <- n_values[1:4]
# write.table(res, "mc_res_with_censoring_1k_iterations_452_2.csv")
```

```
########################################
res <- as.matrix(read.table("mc_res_with_censoring_1k_iterations_452_2.csv",header=T))

barplot(log(res),
        col = c("blue", "orange"),
        beside = TRUE,
        main = "Mean square error of the NR-fit parameter estimates \n based on sample size f
or the \n Log-Logistic distribution \n (Logarithmic scale)", xlab = "Sample size", ylab = "lo
g(MSE)")


legend("topright",
       legend = c("Alpha", "Beta"),
       fill = c("blue", "orange"))
```



We chose a logarithmic scale to be able to see how both parametres evolve. Our observation is that the final parametre estimates vary a lot more based on the choice of initial parametres though. Same goes for computation time, it can be exorbitantly high depending on which initial parametres we put. Probably our NR-fit function isn’t very robust.

Real data analysis

Comparison of 2 treatments with descriptive analysis

In our data we have the lifetimes of guinea pigs in days and also their regime and if the value of the lifetime is censored or not.

We want to study the resistance of guinea pigs to Tubercle Bacili.

We take for the MC guinea pigs all of them containing censored ones and non censored ones.

```
summary(gpigs.mc.with_censored)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    18.0   214.0   621.0   500.9   735.0   735.0
```

```
summary(gpigs.m43)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    10.0   108.0   149.5   176.8   224.0   555.0
```

As we can see, the guinea pigs with MC regime are surviving longer than the ones with M43 regime.

The mean for the MC guinea pigs is around 500 wich is almost three times bigger than the mean for M43 guinea pigs (176.8)

fit a distribution to the "lifetime" with Exponential distribution

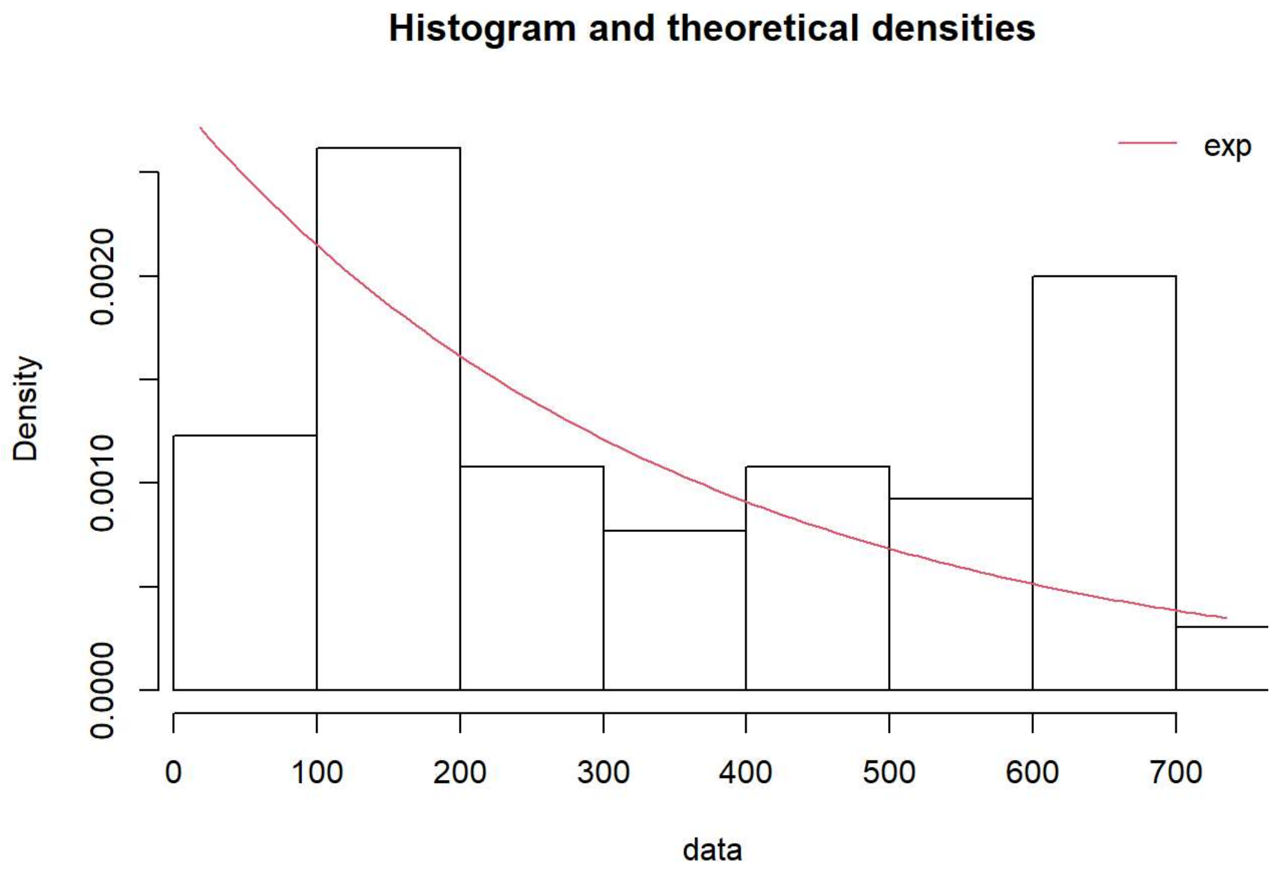
for MC group

```
fe <- fitdist(gpigs.mc, "exp")
fe$estimate
```

```
##        rate 
## 0.002859783
```

This rate is pretty low so the mean distribution will be pretty high. Which is maybe not what we expected, The Histrogram will give more precision about it.

```
denscomp(list(fe))
```



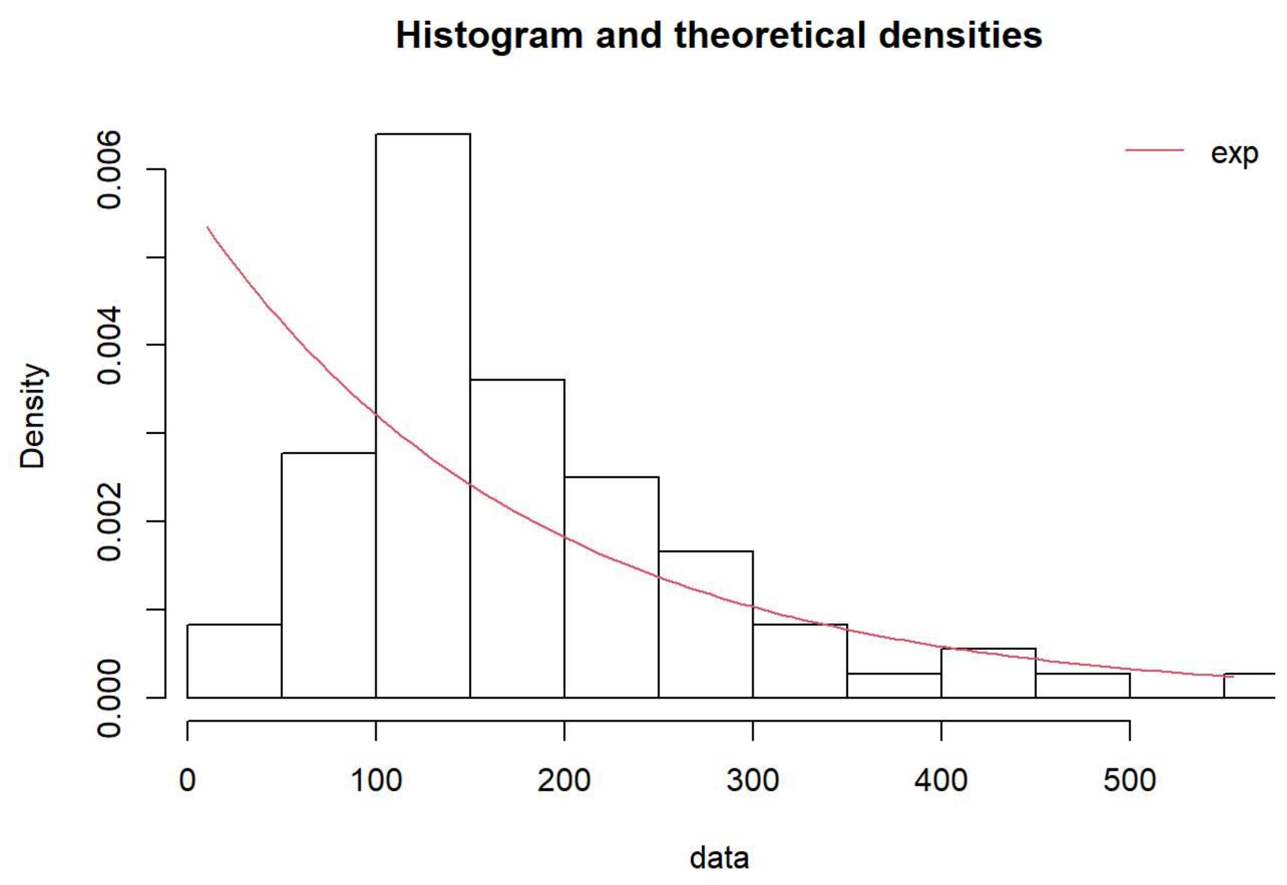
As we can see the exponential function is truly not adapted to our data. The curve is not following well the true values.

for M43 group

```
fe <- fitdist(gpigs.m43, "exp")
fe$estimate
```

```
##        rate
## 0.005655487
```

```
denscomp(list(fe))
```



As we can see the fitting is a bit better with these data but still not good enought.

Do the same for log logistic distribution

Define the distribution function

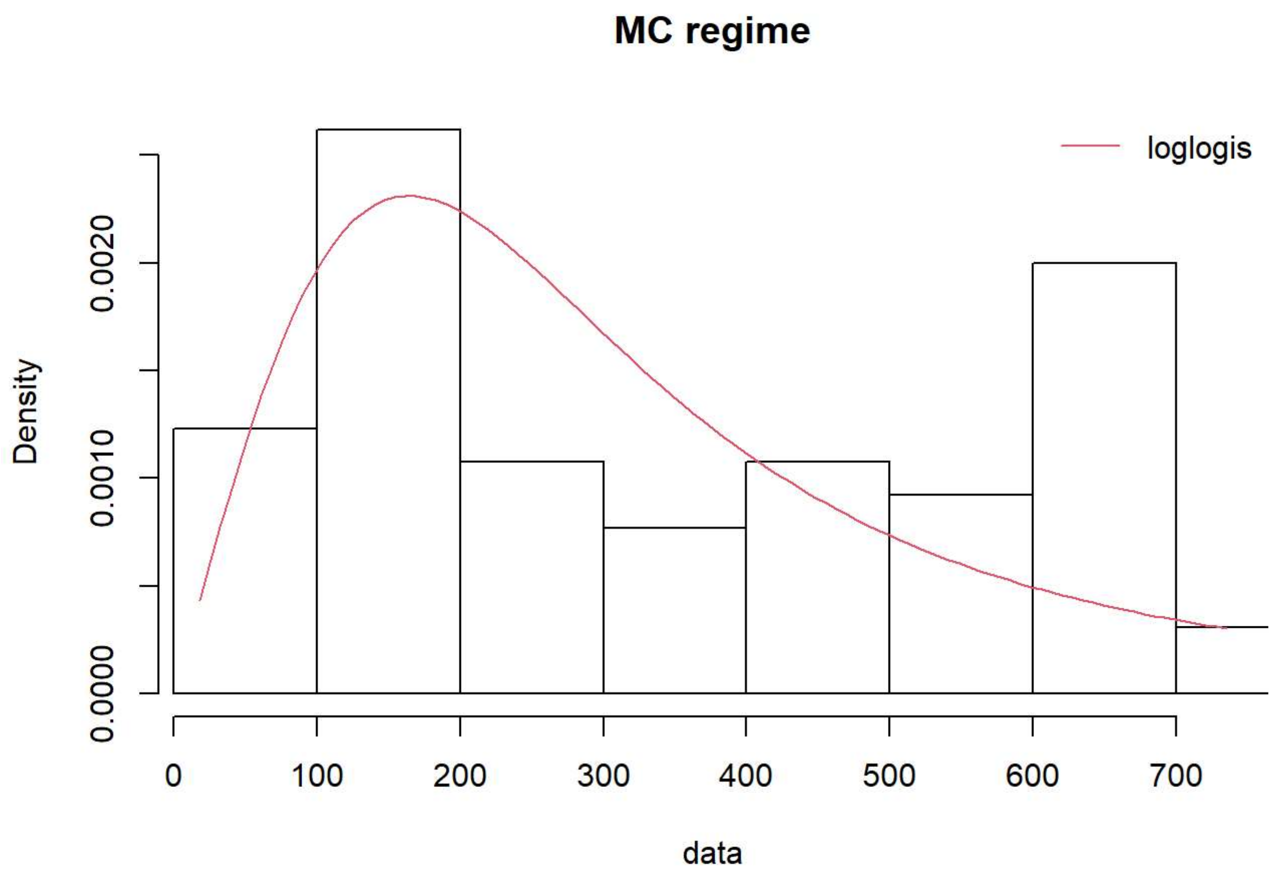
```
dloglogis <- function(x, alpha, beta) {
  res <- (beta/alpha) * (x/alpha)^(beta-1)*(1+(x/alpha)^beta)^-2
  return(res)
}
ploglogis <- function(q, alpha, beta) {
  res <- 1 / (1 + (q/alpha)^beta)
  return(res)
}
qloglogis <- function(p, alpha, beta) {
  res <- alpha * ((1/p) - 1)^(1/beta)
  return(res)
}
```

Control group :

```
# MC regime (without censored data for the moment)
fll <- fitdist(gpigs.mc, "loglogis", method = "mle", start=list(alpha=10, beta=5))
fll$estimate
```

```
##      alpha       beta
## 282.086118   2.014417
```

```
denscomp(list(fll), main = "MC regime")
```



This graph represent the lifetimes of guinea pigs for animals with MC regime but without the censored data. As we can see the curve is following pretty well the data especially for the beginning.

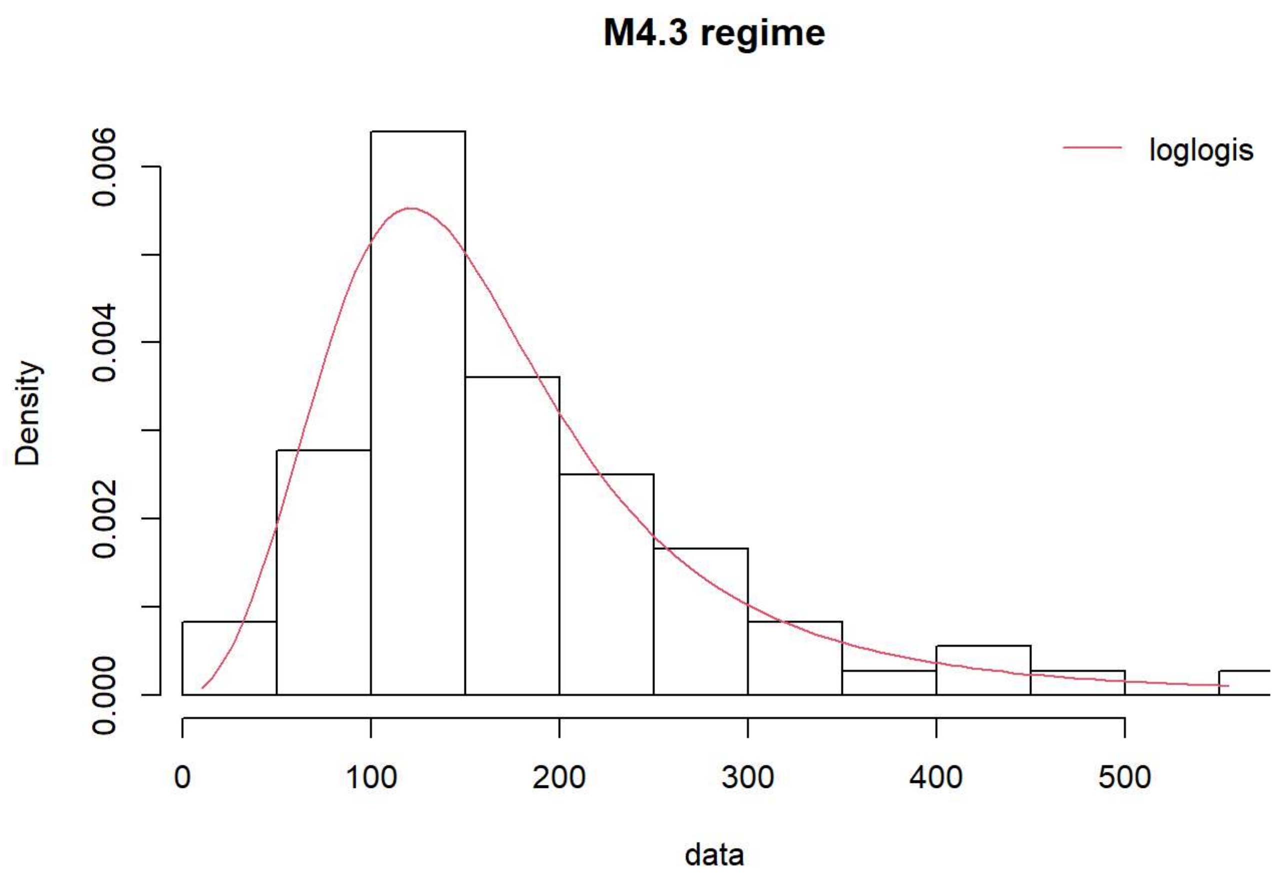
M4.3-Regime group :

```
# M4.3 regime
fll <- fitdist(gpigs.noncensored$lifetime[gpigs.noncensored$regime == "M_4.3"], "loglogis", m
ethod = "mle", start=list(alpha=10, beta=5))
fll$estimate
```

```
##      alpha       beta
## 152.389696   3.012415
```

The estimation of alpha and beta are a bit different for the M43 guinea pigs than for the MC guinea pigs.

```
denscomp(list(fll), main = "M4.3 regime")
```

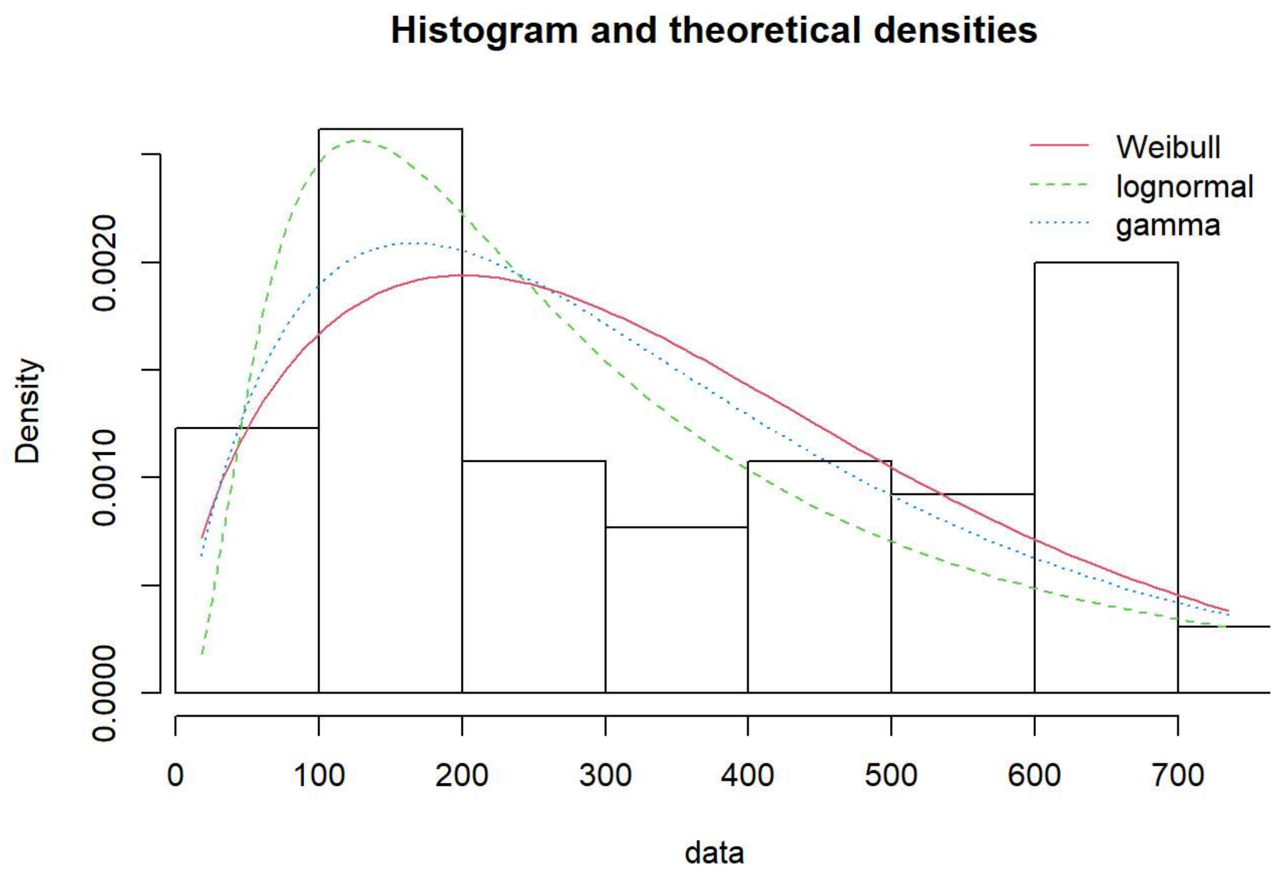


In this case the loglogistic distribution fit perfectly with the data as we can see on the graph.

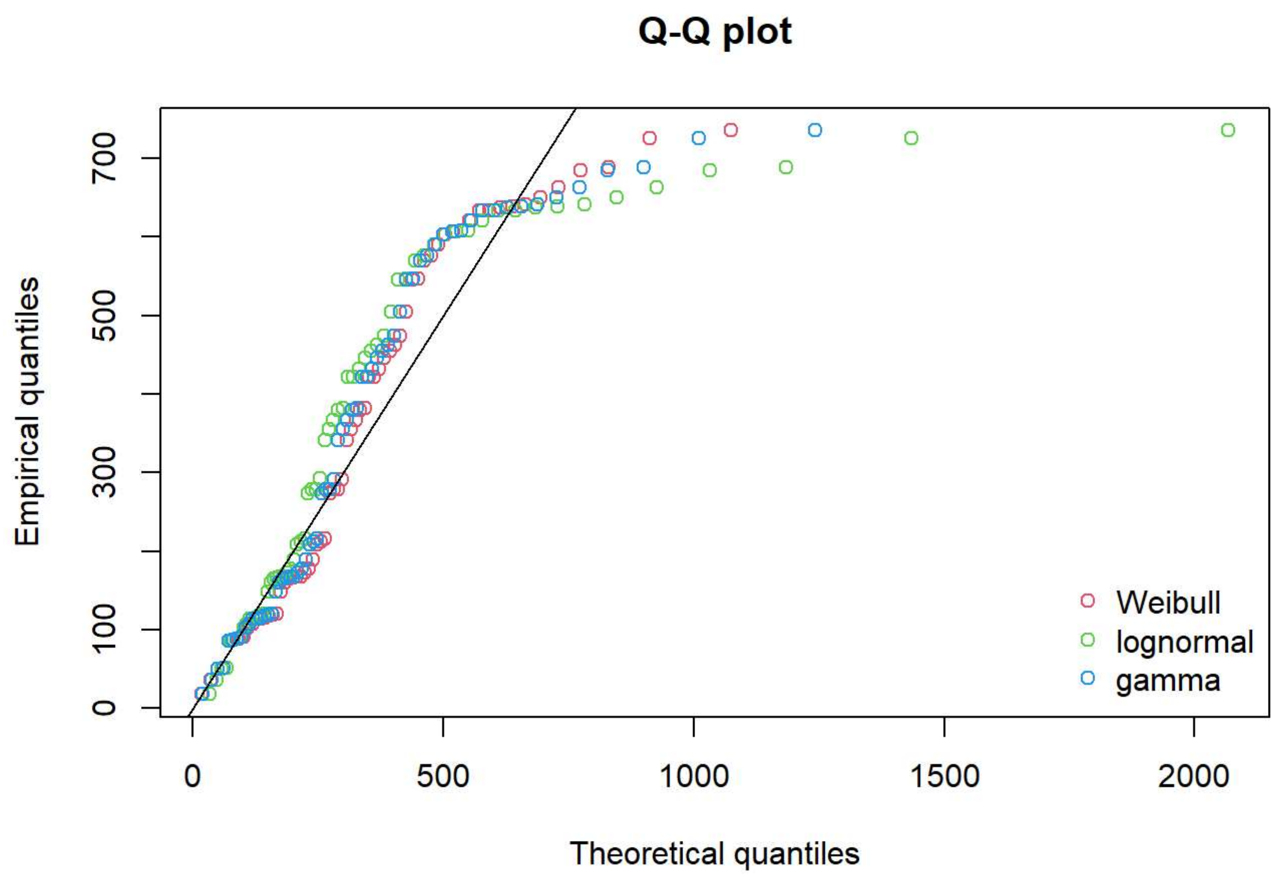
Try other distributions

###Weibull, Gamma, log-Normal fit on the control group

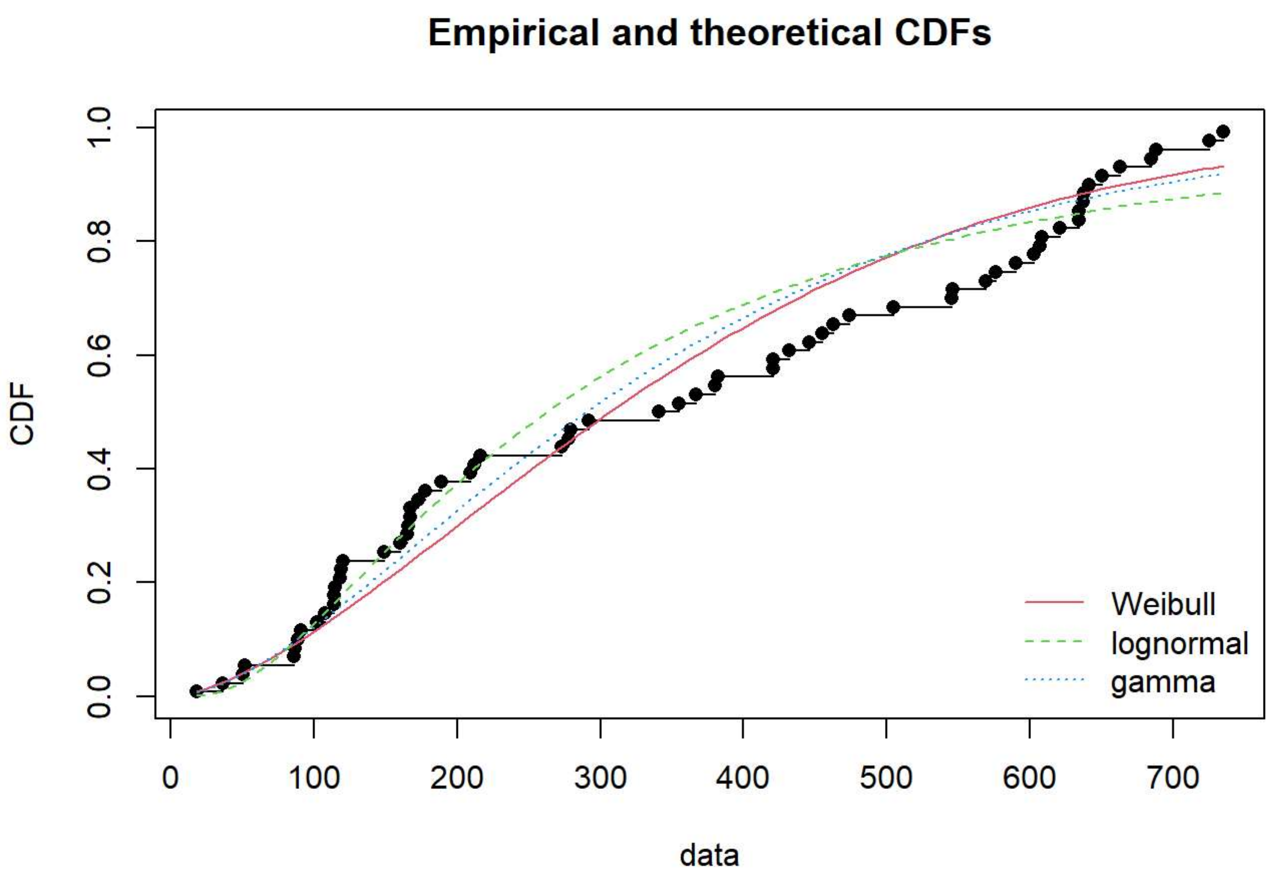
```
fw <- fitdist(gpigs.mc, "weibull")
fg <- fitdist(gpigs.mc, "gamma")
fln <- fitdist(gpigs.mc, "lnorm")
plot.legend <- c("Weibull", "lognormal", "gamma")
denscomp(list(fw, fln, fg), legendtext = plot.legend)
```



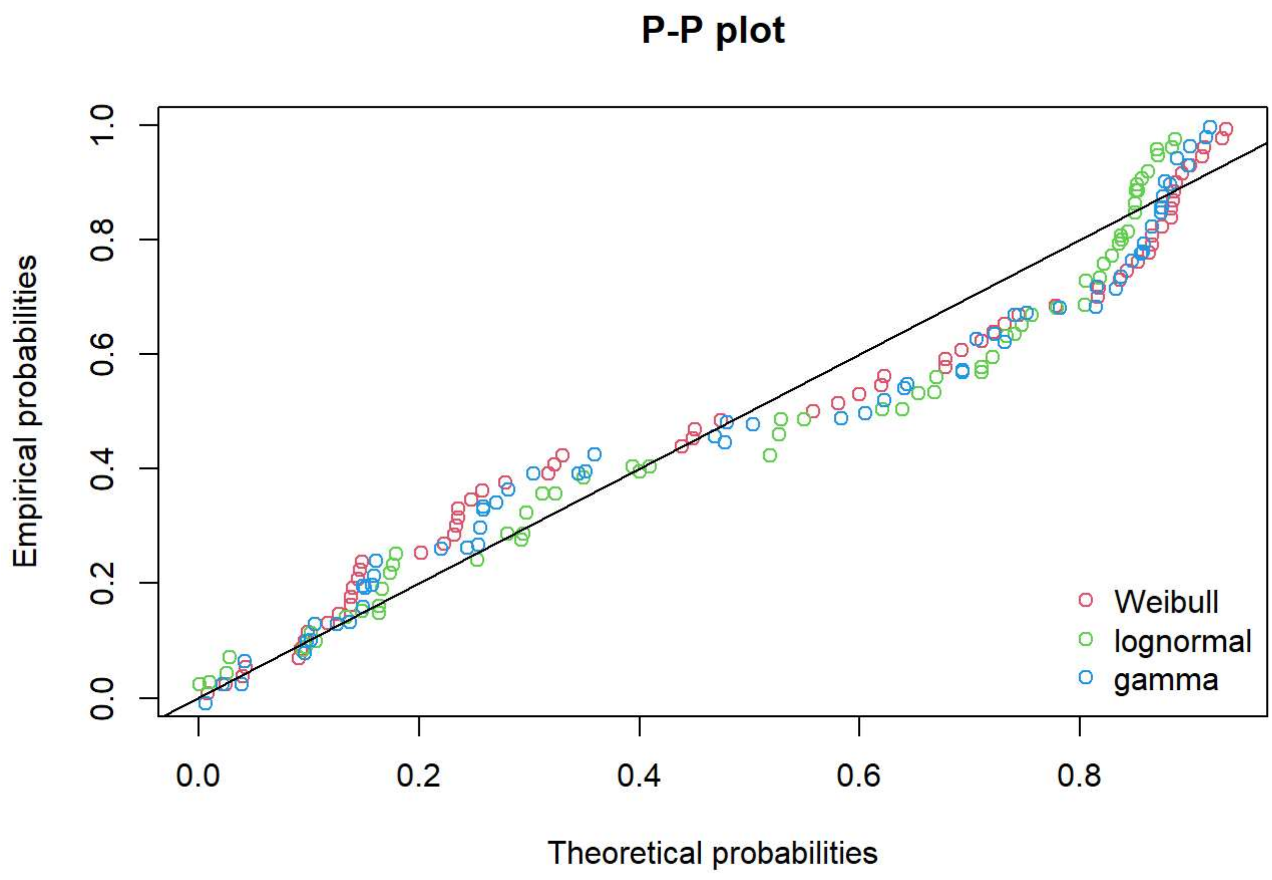
```
qqcomp(list(fw, fln, fg), legendtext = plot.legend)
```



```
cdfcomp(list(fw, fln, fg), legendtext = plot.legend)
```



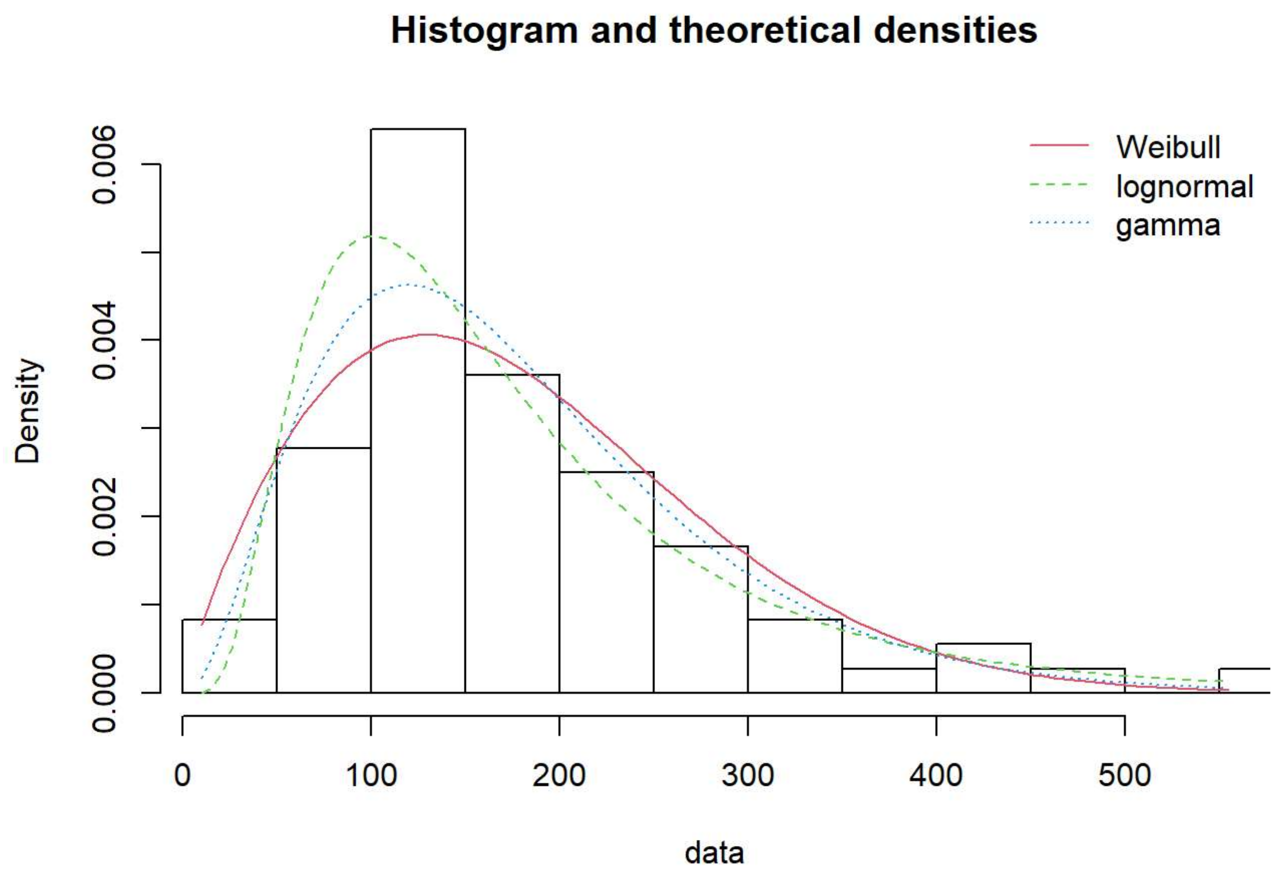
```
ppcomp(list(fw, fln, fg), legendtext = plot.legend)
```



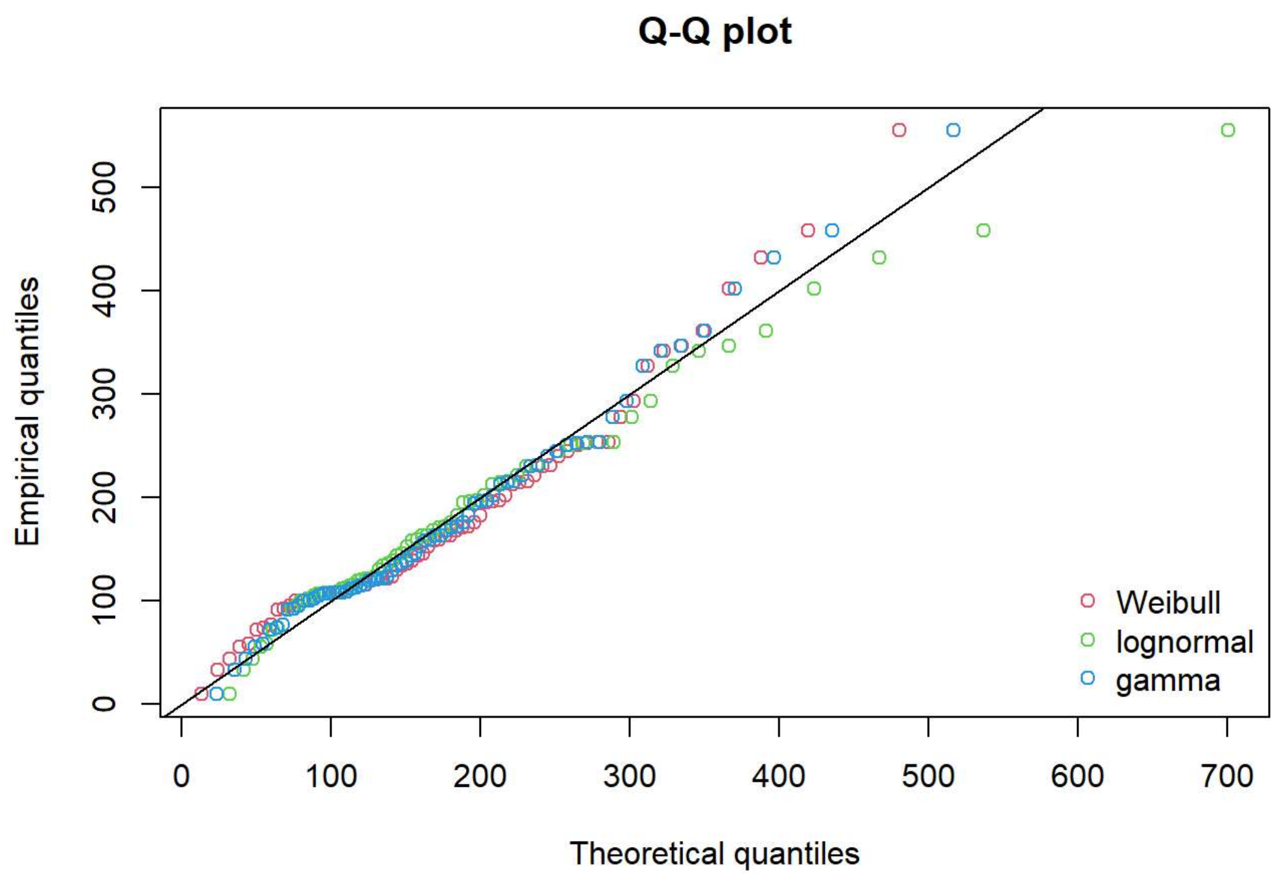
The three other distribution are fitting pretty well to our data, but the closest distribution is the gamma distribution.

Weibull, Gamma, log-Normal fit on the 4.3 group

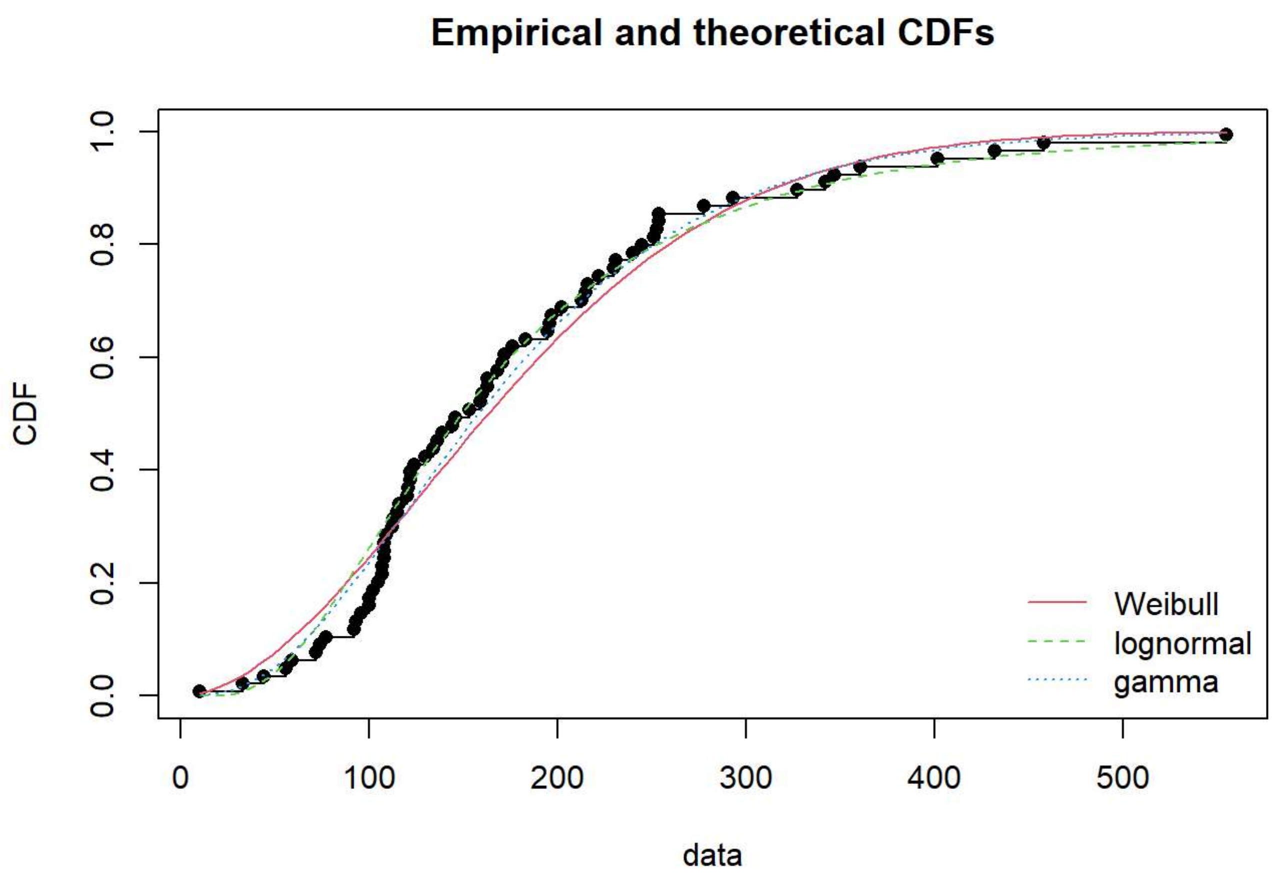
```
fw <- fitdist(gpigs.m43, "weibull")
fg <- fitdist(gpigs.m43, "gamma")
fln <- fitdist(gpigs.m43, "lnorm")
plot.legend <- c("Weibull", "lognormal", "gamma")
denscomp(list(fw, fln, fg), legendtext = plot.legend)
```



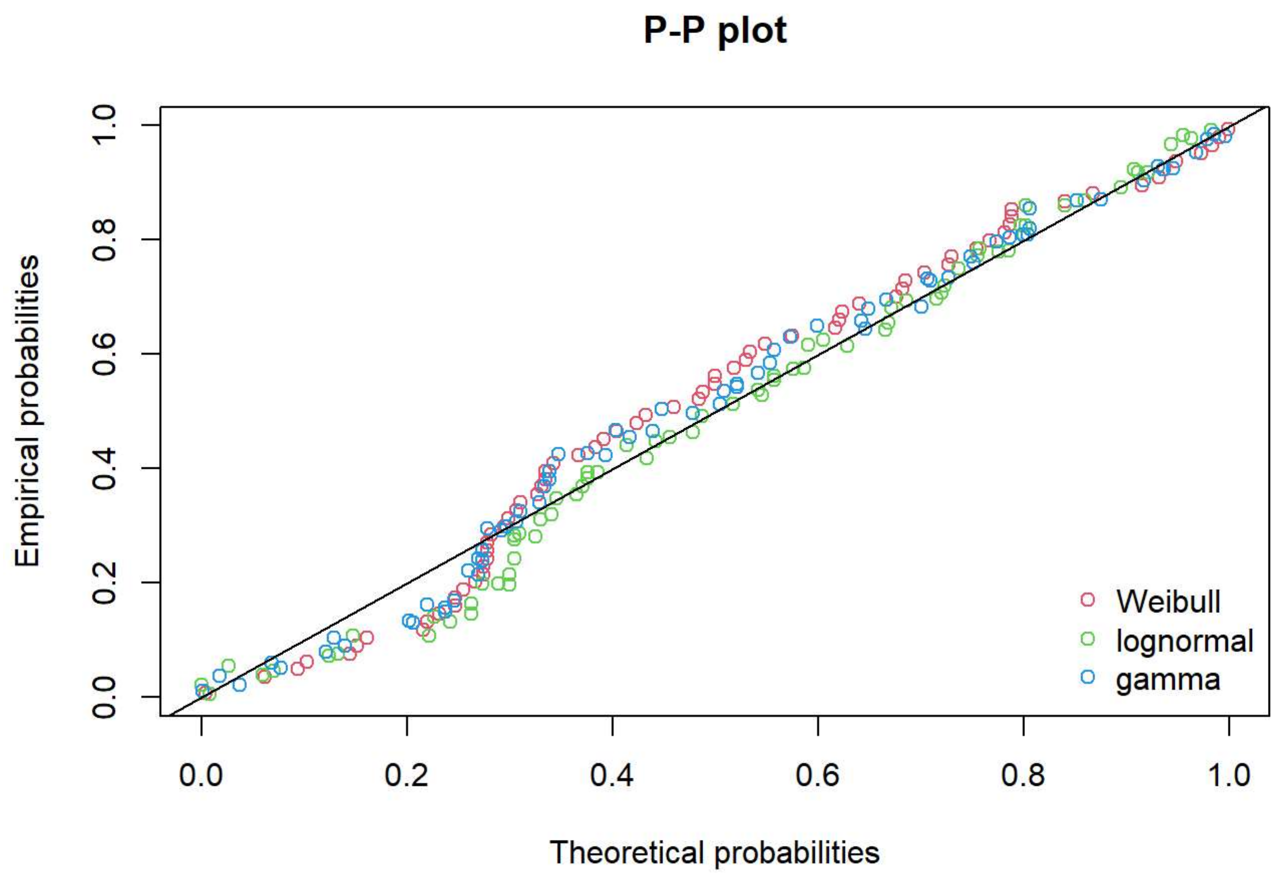
```
qqcomp(list(fw, fln, fg), legendtext = plot.legend)
```



```
cdfcomp(list(fw, fln, fg), legendtext = plot.legend)
```



```
ppcomp(list(fw, fln, fg), legendtext = plot.legend)
```



Same than previously the distributions are fitting well to the data of M43 guinea pigs and the closest is either the gamma distribution either the lognormal distribution.

We had estimated the $alpha = 152.4$ and $beta = 3.0$

```
fw$estimate
```

```
##      shape      scale 
##   1.825211 199.586910
```

For the weibull distribution the shape (beta) is to low and the scale much to high.

```
fg$estimate
```

```
##      shape       rate 
## 3.08241826 0.01743161
```

for the gamma distribution, the shape and the rate looks pretty good and we will confirm this at the end.

```
fln$estimate
```

```
##   meanlog     sdlog 
## 5.0042920 0.6290239
```

the meanlog and the sdlog are pretty fitting the data.

provide 95% basic bootstrap confidence interval

```r
loglogis <- function(x, alpha, beta) {
  res <- (beta/alpha) * (x/alpha)^(beta-1)*(1+(x/alpha)^beta)^-2
  return(res)
}
# and our data
data <- data.frame(x = gpigs$lifetime,
                   delta = gpigs$censored)
lifetimes <- gpigs$lifetime
# Function to calculate the statistic
calculate_statistic <- function(fitdist) {
  fit_R = fitdist(data = lifetimes, distr = "loglogis", method = "mle", start=list(alpha=10,
beta=5))
  return(fit_R)
}
fit_R = fitdist(data = lifetimes, distr = "loglogis", method = "mle", start=list(alpha=10, be
ta=5))

# Parameters
B <- 1000  # Number of bootstrap iterations
bootstrap_samples <- vector("numeric", B)

# Perform bootstrap iterations
for (i in 1:B) {
  # Randomly sample with replacement the pairs (xi, delta_i)
  bootstrap_sample <- data[sample(nrow(data), replace = TRUE), ]

  # Perform the desired analysis or calculation on the bootstrap sample
  bootstrap_statistic <- calculate_statistic(bootstrap_sample)

  # Store the result in the list of bootstrap samples
  bootstrap_samples[i] <- bootstrap_statistic
}

#display the results
npboot_CI = bootdist(fit_R ,bootmethod = "nonparam",niter = B)
summary(npboot_CI)
```

```
## Nonparametric bootstrap medians and 95% percentile CI
##           Median       2.5%      97.5%
## alpha 275.276211 237.963254 317.275492
## beta    1.891726   1.744956   2.049876
```

Use proper information criteria and goodness-of-fit plots to discuss the fit to the data.

```
fw <- fitdist(gpigs.mc, "weibull")
fg <- fitdist(gpigs.mc, "gamma")
fln <- fitdist(gpigs.mc, "lnorm")
fe <- fitdist(gpigs.mc, "exp")
fll <- fitdist(gpigs.mc, "loglogis", method = "mle", start=list(alpha=10, beta=5))

model_list <- list(fe, fll, fw, fg, fln)

# Noms des modèles
model_names <- c("exp", "loglogis", "weibull", "gamma","lnorm")

# Évaluer la qualité d'ajustement avec gofstat()
gof_stats <- gofstat(model_list, fitnames = model_names)

# Afficher les résultats
print(gof_stats)
```

```
## Goodness-of-fit statistics
##                                    exp    loglogis   weibull     gamma
## Kolmogorov-Smirnov statistic 0.1564934   0.9961019 0.1239208 0.1239852
## Cramer-von Mises statistic   0.4818175  22.4003702 0.2718269 0.2589622
## Anderson-Darling statistic   3.0970754 124.9865559 1.6639126 1.5902238
##                                  lnorm
## Kolmogorov-Smirnov statistic 0.1411335
## Cramer-von Mises statistic   0.2886472
## Anderson-Darling statistic   1.8220951
##
## Goodness-of-fit criteria
##                                     exp loglogis  weibull    gamma    lnorm
## Akaike's Information Criterion 893.4113 893.8985 879.8406 882.3291 891.7370
## Bayesian Information Criterion 895.5856 898.2473 884.1894 886.6778 896.0858
```

The most favorized distribution for the mc guinea pigs is the gamma one closely followed by the weibull one.

```r
fw <- fitdist(gpigs.m43, "weibull")
fg <- fitdist(gpigs.m43, "gamma")
fln <- fitdist(gpigs.m43, "lnorm")
fe <- fitdist(gpigs.m43, "exp")
fll <- fitdist(gpigs.m43, "loglogis", method = "mle", start=list(alpha=10, beta=5))

model_list <- list(fe, fll, fw, fg, fln)

# Noms des modèles
model_names <- c("exp", "loglogis", "weibull", "gamma","lnorm")

# Évaluer la qualité d'ajustement avec gofstat()
gof_stats <- gofstat(model_list, fitnames = model_names)

# Afficher les résultats
print(gof_stats)
```

```
## Goodness-of-fit statistics
##                                    exp    loglogis   weibull      gamma
## Kolmogorov-Smirnov statistic 0.2945495   0.9997269 0.1048345 0.09073485
## Cramer-von Mises statistic   1.4040565  23.7143219 0.1678917 0.09377564
## Anderson-Darling statistic   7.2647928 145.0213610 1.0069058 0.57934064
##                                  lnorm
## Kolmogorov-Smirnov statistic 0.1104098
## Cramer-von Mises statistic   0.1024276
## Anderson-Darling statistic   0.7380047
##
## Goodness-of-fit criteria
##                                     exp loglogis  weibull    gamma    lnorm
## Akaike's Information Criterion 891.2186 855.1982 858.7241 855.6027 862.1888
## Bayesian Information Criterion 893.4953 859.7516 863.2775 860.1560 866.7421
```

In the case of $m43$ guinea pigs the most favorized distribution is the gamma distribution for the Goodness-of-fit statistics and the loglogistic for the Goodness-of-fit criteria.