Distributions

X. Pouwels

2022-03-02

Contents

Install and load packages

First, the required packages need to be installed and loaded.

```
rm(list = ls()) # clear environment

# Install
#install.packages("fitdistrplus")
#install.packages("mvtnorm")
#install.packages("rriskDistributions")
#install.packages("doSNOW")
#install.packages("parallel")
#install.packages("ggplot2")
```

```r
#install.packages("survival")
#install.packages("survminer")
#install.packages("flexsurv")

# Load
library(fitdistrplus)
library(mvtnorm)
library(rriskDistributions)
library(doSNOW)
library(parallel)
library(ggplot2)
library(survival)
library(survminer)
library(flexsurv)
```

Continuous non-censored data

Inspect data set

```r
data("groundbeef") #load data
names(groundbeef) # display names columns
```

```
## [1] "serving"
```

You can check the names of columns and the first columns by using the next commands. We will use the `groundbeef` data set. This data set contains one variable `serving` (N = 254) which represents the serving sizes of ground beef patties consumed by children under 5 years old, collected in a French survey. This data set is provided within the *fitdistrplus* package.

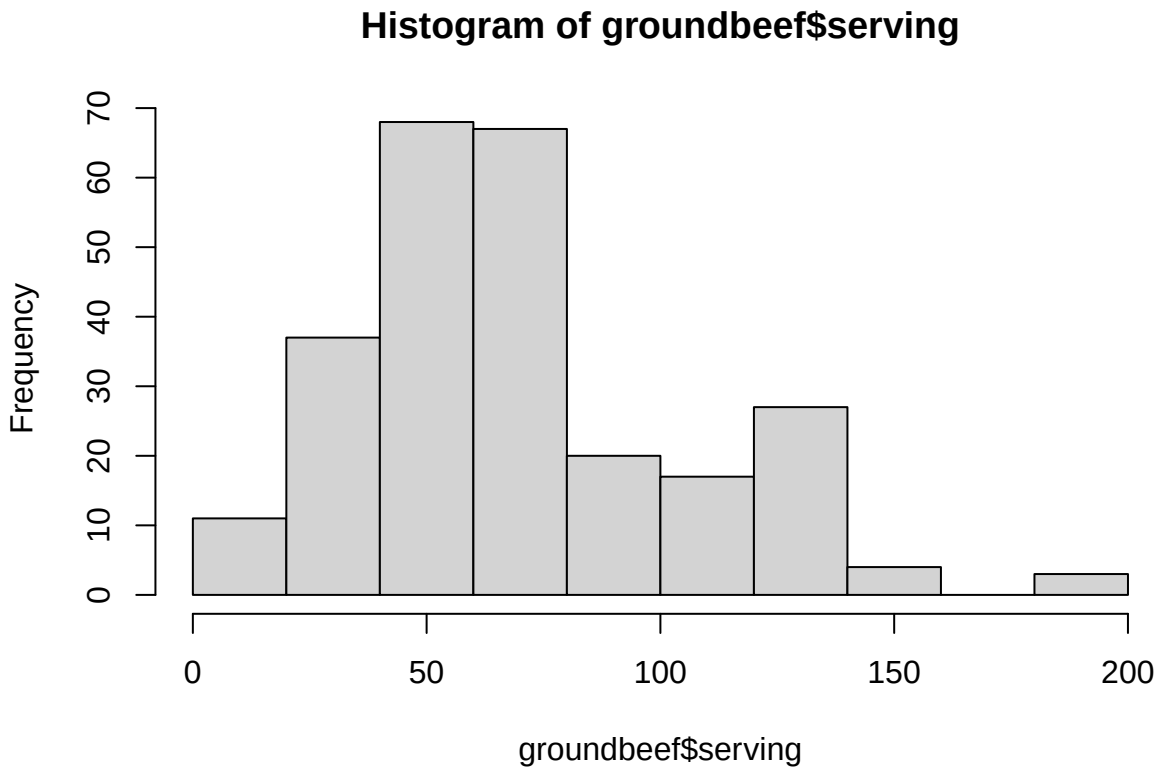
```r
head(groundbeef) # show first 6 rows
```

```
##   serving
## 1      30
## 2      10
## 3      20
## 4      24
## 5      20
## 6      24
```

```r
summary(groundbeef) # provide summary statistic
```

```
##     serving
##  Min.   : 10.00
##  1st Qu.: 50.00
##  Median : 79.00
##  Mean   : 73.65
##  3rd Qu.:100.00
##  Max.   :200.00
```

```r
#plot(danishuni$Loss) # provide plot per entry
hist(groundbeef$serving) # provide histogram
```



Fit distribution

Exponential

```r
dist_exp <- fitdist(groundbeef$serving, distr = "exp")
dist_exp
```

```
## Fitting of the distribution ' exp ' by maximum likelihood
## Parameters:
##        estimate   Std. Error
## rate 0.01357853 0.0008473423
```

Weibull

```r
dist_weib <- fitdist(groundbeef$serving, distr = "weibull")
dist_weib
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## shape  2.185885  0.1045755
## scale 83.347679  2.5268626
```

Gamma

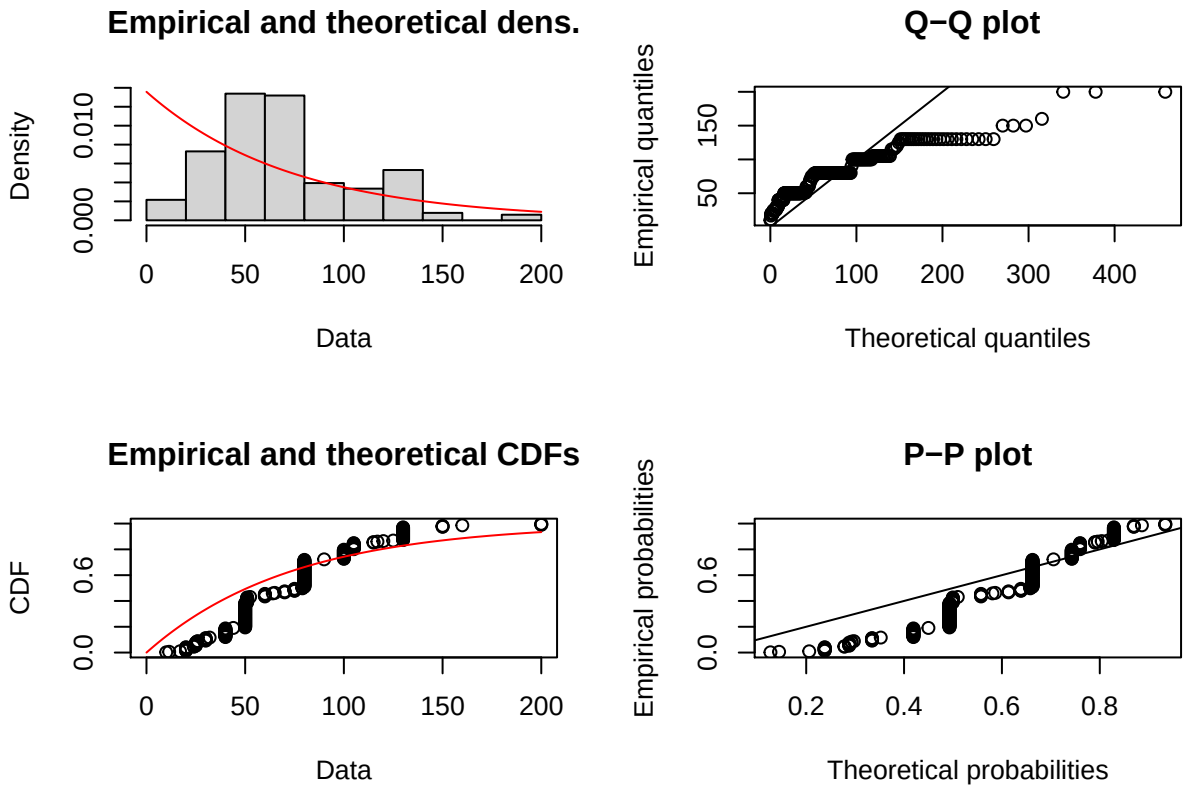
```
dist_gamma <- fitdist(groundbeef$serving, distr = "gamma")
dist_gamma
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##         estimate  Std. Error
## shape 4.00825257 0.341336046
## rate  0.05441911 0.004935468
```

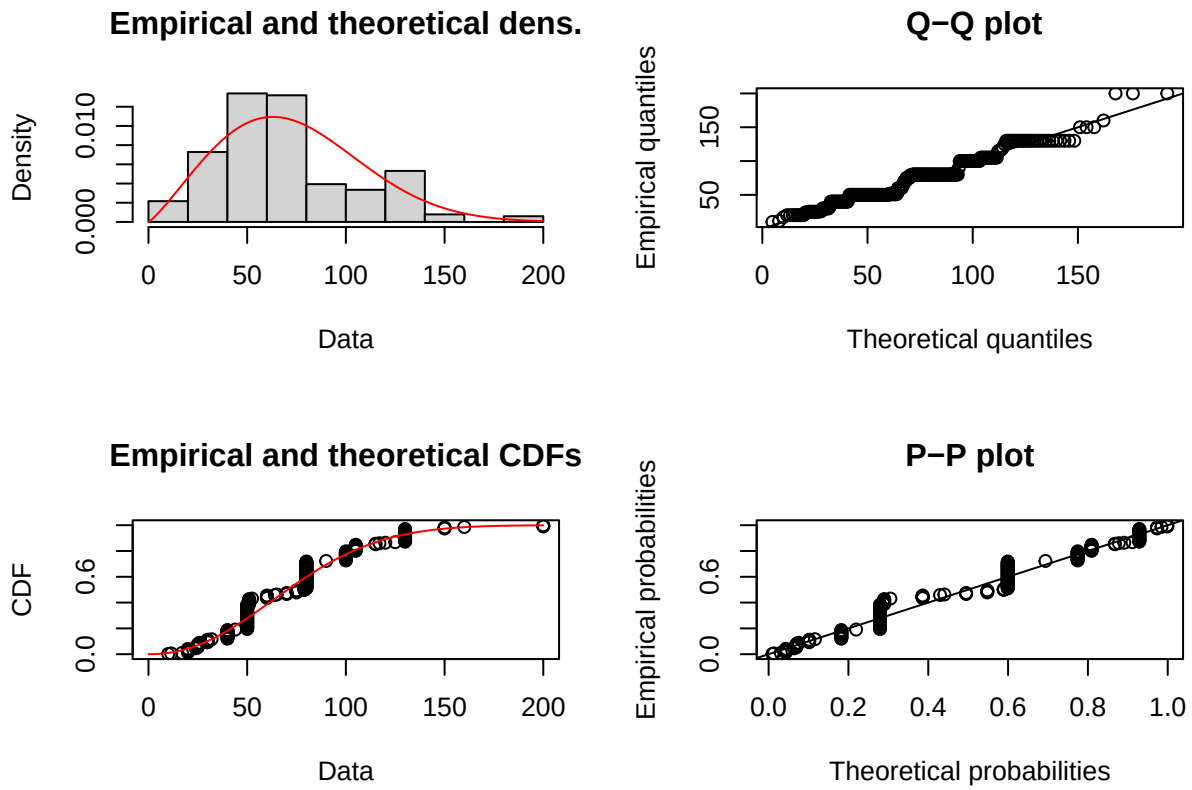
Compare goodness of fit

Visually, individual

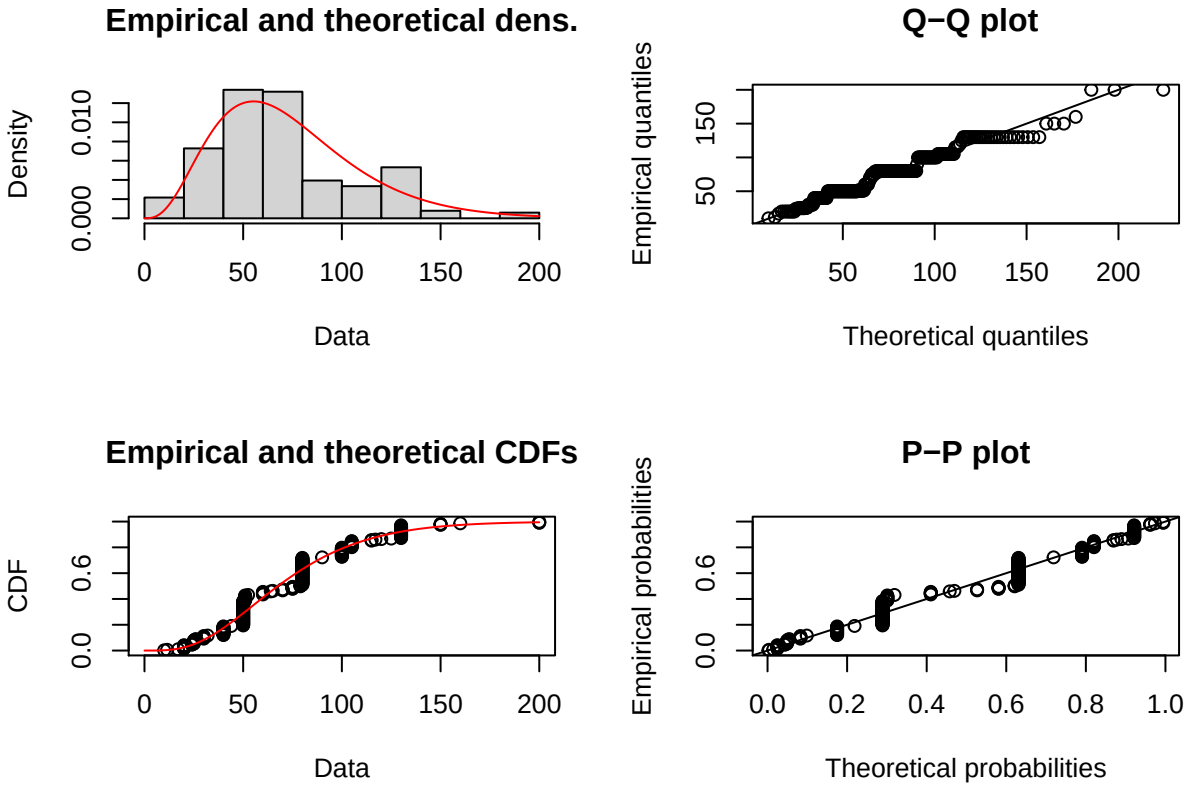
```
plot(dist_exp)
```



```
plot(dist_weib)
```



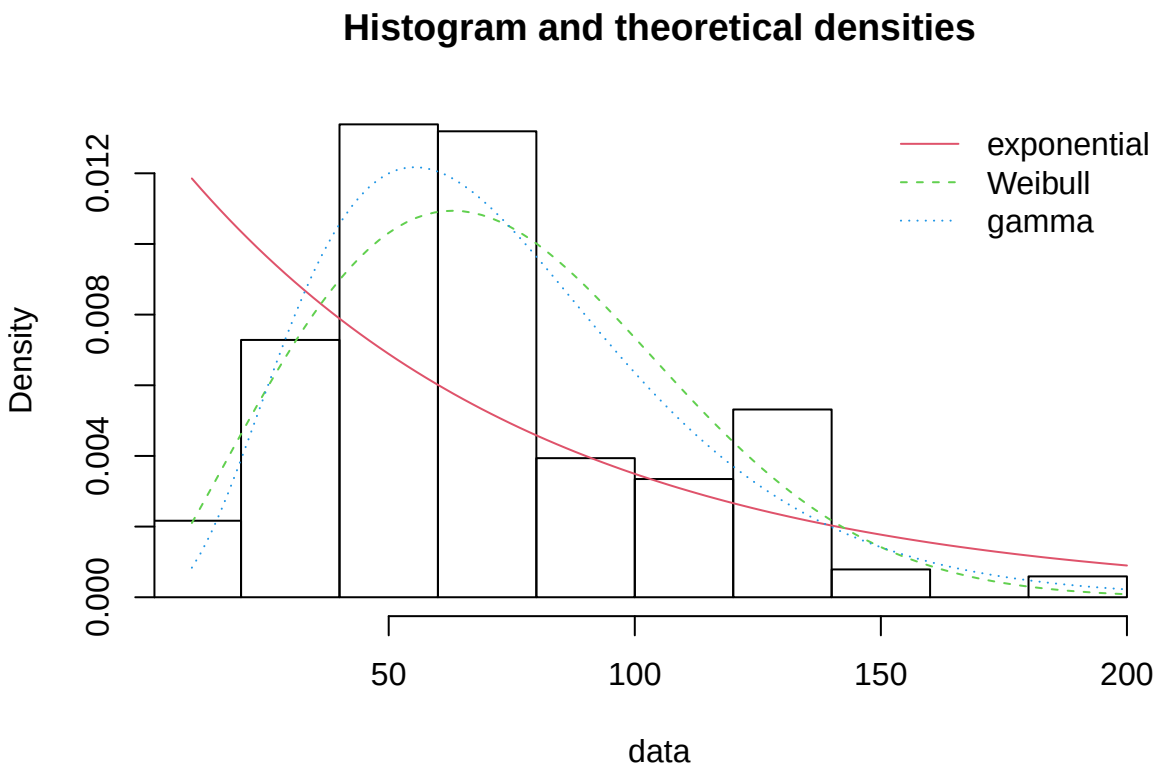

```
plot(dist_gamma)
```



Visually, against each other

The comparison of these density functions against the hystogram shows that the Weibull and gamma distributions tend to fit better to the data than the exponential function.

```
denscomp(list(dist_exp, dist_weib, dist_gamma), legendtext=c("exponential", "Weibull", "gamma"))
```



Goodness-of-fit statistics

All at once

The observation that the Weibull and gamma fit the data better is confirmed by the goodness-of-fit statistics. For instance, the Akaike Information Criterion (the lower, the better) of the gamma distribution is the lowest for the gamma distribution, followed by the Weibull, and finally the exponential.

```
gofstat(list(dist_exp, dist_weib, dist_gamma), fitnames = c("exponential", "Weibull", "gamma"))
```

```
## Goodness-of-fit statistics
##                              exponential   Weibull     gamma
## Kolmogorov-Smirnov statistic   0.3009692 0.1396646 0.1281246
## Cramer-von Mises statistic     6.1004290 0.6840994 0.6934112
## Anderson-Darling statistic    31.9421438 3.5736460 3.5660192
##
## Goodness-of-fit criteria
##                                exponential  Weibull    gamma
## Akaike's Information Criterion    2694.027 2514.449 2511.250
## Bayesian Information Criterion    2697.564 2521.524 2518.325
```

Specific (e.g. AIC)

```r
c('exponenital'= dist_exp$aic, 'Weibull'= dist_weib$aic, 'gamma'= dist_gamma$aic)
```

```
## exponenital     Weibull       gamma
##    2694.027    2514.449    2511.250
```

Draw random value from distributions

In a discrete event simulation, drawing random values from this type of distributions allows to capture stochastic uncertainty.

```r
rexp(n = 1, rate = dist_exp$estimate["rate"]) # 1 random value
```

```
## [1] 92.85669
```

```r
rweibull(1, scale = dist_weib$estimate["scale"], shape = dist_weib$estimate["shape"]) # 1 random value
```

```
## [1] 26.88971
```

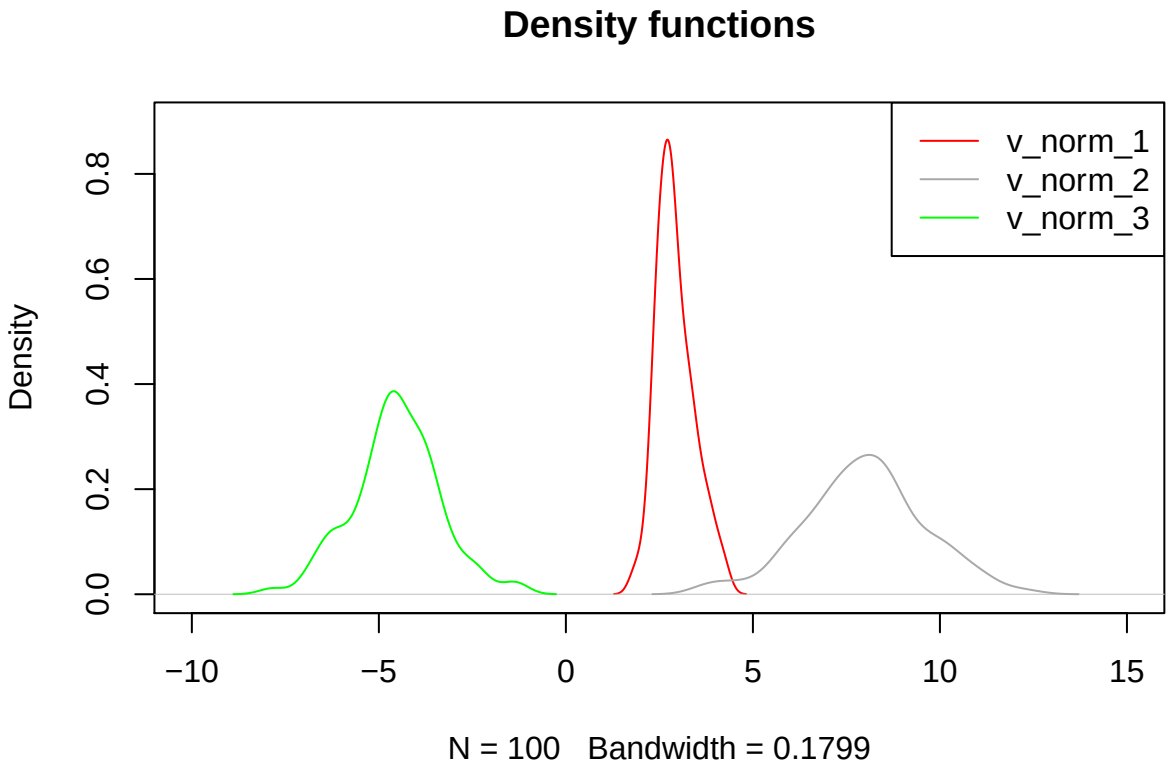
```r
rgamma(1, shape = dist_gamma$estimate["shape"], rate = dist_gamma$estimate["rate"]) # 1 random value
```

```
## [1] 36.97635
```

Multivariate normal distribution

Three normally distributed variables

```r
set.seed(1234)
#create three variables which are correlated
v_norm_1 <- rnorm(100, 3, 0.5)
v_norm_2 <- v_norm_1 + rnorm(100, 5, 1.5)
v_norm_3 <- 1/2 * v_norm_1 - 3/4 * v_norm_2

m_vect <- cbind(v_norm_1,
                v_norm_2,
                v_norm_3) # create matrix from random variables


plot(density(v_norm_1), col = 'red', xlim = c(-10, 15), ylim = c(0, 0.9), main = 'Density functions') #
lines(density(v_norm_2), col = 'darkgray')
lines(density(v_norm_3), col = 'green')
legend("topright",                    # Add legend to plot
       legend = c("v_norm_1", "v_norm_2", "v_norm_3"),
       col = c('red', 'darkgray', 'green'),
       lty = 1)
```




```
v_means <-  c(mean(v_norm_1), mean(v_norm_2), mean(v_norm_3)) # create vector of means
m_cov <- cov(m_vect) # create covariance matrix

v_means # shows vector of means
```

```
## [1]  2.921619  7.983484 -4.526803
```

```
cor(m_vect) # shows correlations matrix
```

```
##             v_norm_1   v_norm_2    v_norm_3
## v_norm_1  1.00000000  0.2865347 -0.08248227
## v_norm_2  0.28653469  1.0000000 -0.97843931
## v_norm_3 -0.08248227 -0.9784393  1.00000000
```

```
m_cov # shows covariance matrix
```

```
##             v_norm_1  v_norm_2    v_norm_3
## v_norm_1  0.25220750  0.232471 -0.04824953
## v_norm_2  0.23247104  2.609909 -1.84119601
## v_norm_3 -0.04824953 -1.841196  1.35677224
```

```
rmvnorm(n = 5, mean = v_means, sigma = m_cov) # simultaneous random draw for each variable
```

```
##          [,1]     [,2]      [,3]
## [1,] 3.286840 8.826255 -4.976271
## [2,] 3.372062 7.833647 -4.189205
## [3,] 3.970091 9.758291 -5.333673
## [4,] 2.445974 8.701008 -5.302769
## [5,] 3.280841 6.495582 -3.231266
```

Fitting distributions based on summary statistics (method of moments)

Often, you will only dispose of the summary statistics (mean and standard deviation) of a distribution instead of the individual patient level data to fit these statistical distributions. In those case, the method of moments can be used to calculate the parameters of the distribution based on these summary statistics. Of course, if you dispose of the patient level data, you would use the method described in the previous section!

Gamma distribution

Let's use our `groundbeef` example data set again.

```
dist_gamma # shows parameters of the distribution fitted using maximum likelyhood estimation on the ind
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##         estimate  Std. Error
## shape 4.00825257 0.341336046
## rate  0.05441911 0.004935468
```

Let's first calculate the mean and standard deviation of `serving` variable because these are the two elements that you need to compute the parameters of the gamma distribution.

```
mean_serving <- mean(groundbeef$serving)
sd_serving <- sd(groundbeef$serving)
```

For a gamma distribution, the scale (a) and rate (b) parameters can be calculated as follows. Both parameters of the gamma distribution are strictly positive.

$$a = mean^2/variance$$

$$b = mean/variance$$

When filling in the equations with the mean and standard deviation we just calculated, we obtain the following results. The estimated parameters are often slightly biased when using the methods of moments.

```
shape_mom <- mean_serving^2/sd_serving^2
rate_mom <- mean_serving/sd_serving^2

cbind("Fitted parameters" = dist_gamma$estimate,
      "Estimated parameters" = c(shape_mom, rate_mom))
```

```
##       Fitted parameters Estimated parameters
## shape        4.00825257           4.21183859
## rate         0.05441911           0.05719058
```

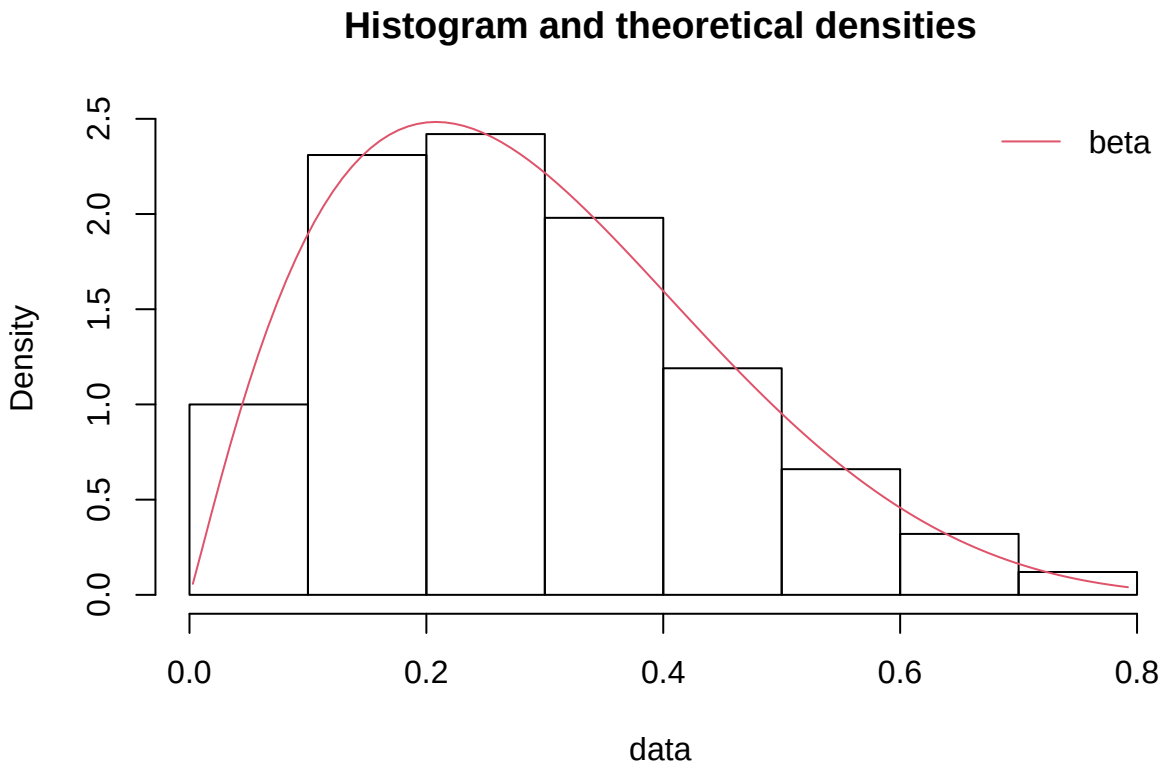
Beta distribution

For the beta distribution, the method of moments can also be used to estimate the parameters of the distribution. Imagine we have the following beta distribution with shape1 (a) = 2 and shape2 (b) = 5. Let's take 1000 random draws of this ditribution first.

```
v_rbeta <- rbeta(1000, shape1 = 2, shape2 = 5) # random beta variable
dist_beta <- fitdist(v_rbeta, distr = 'beta') # fit beta distribution to random vector
dist_beta$estimate # parameters of the fitted distribution
```

```
##   shape1   shape2
## 2.098566 5.180405
```

```
denscomp(list(dist_beta), legendtext = c('beta'))
```



The moments of the beta distribution can be calculated as follows.

$$a = ((mean * (1 - mean)/variance) - 1) * mean$$

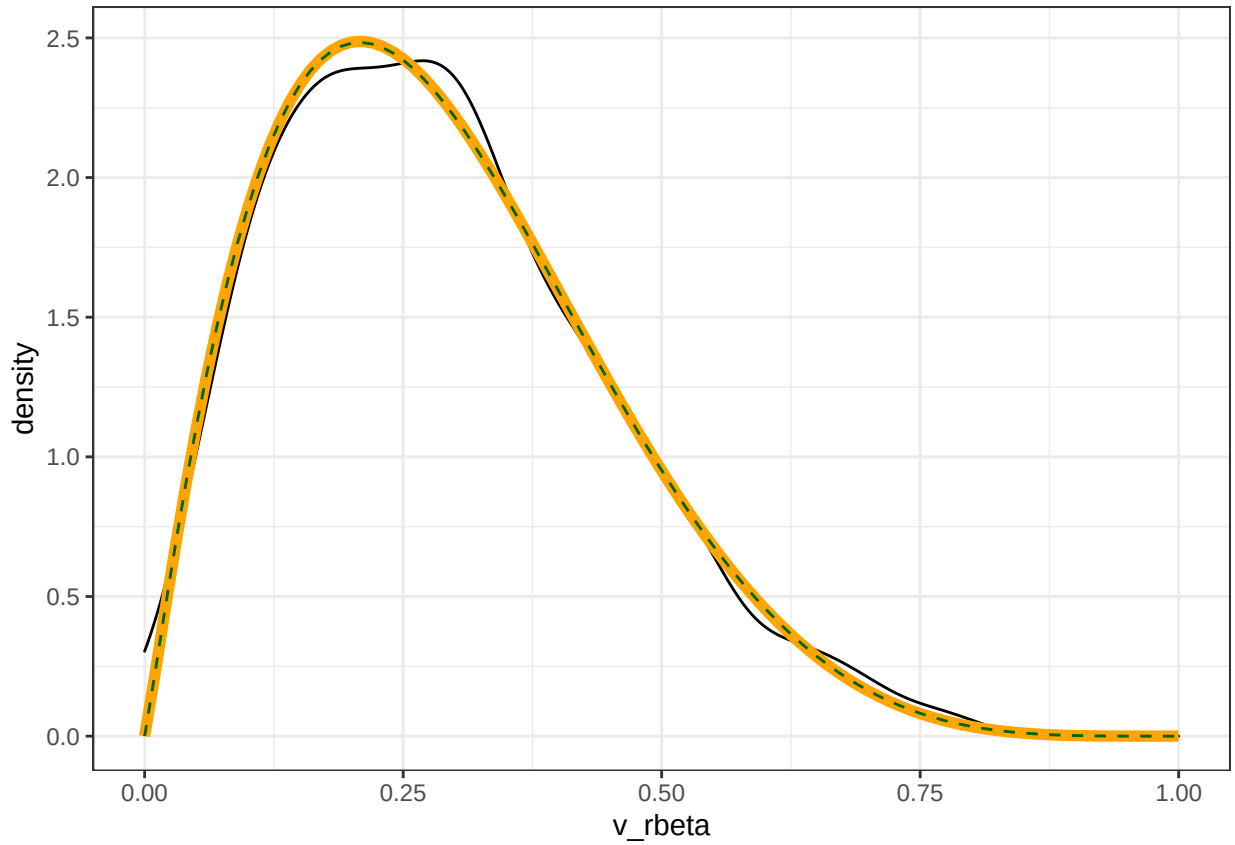
$$b = ((mean * (1 - mean)/variance) - 1) * (1 - mean)$$

Now, we can use these equations to calulate the parameters of the distribution based on the mean and standard deviation of the random variable `v_rbeta`.

```
mean_beta <- mean(v_rbeta)
sd_beta <- sd(v_rbeta)
shape1_mom <- ((mean_beta * (1-mean_beta)/sd_beta^2)-1)*mean_beta # calculate first parameter of the di
shape2_mom <- ((mean_beta * (1-mean_beta)/sd_beta^2)-1)*(1-mean_beta) # calculate second parameter of t
cbind("Fitted parameters" = dist_beta$estimate,
      "Estimated parameters" = c(shape1_mom, shape2_mom))
```

```
##        Fitted parameters Estimated parameters
## shape1          2.098566             2.105815
## shape2          5.180405             5.200200
```

```
# plot the different density functions
## black = empirical
## orange = methods of moment
## dark green = beta distribution fitted to vector 'v_rbeta'

x <- seq(from = 0, to = 1, by = 0.01) # x-values for density function
y <- dbeta(seq(from = 0, to = 1, by = 0.01), shape1 = shape1_mom, shape2 = shape2_mom) # y-values densi
df <- as.data.frame(cbind(x,y))

y <- dbeta(seq(from = 0, to = 1, by = 0.01), shape1 = dist_beta$estimate["shape1"], shape2 = dist_beta$
df2 <- as.data.frame(cbind(x,y))

ggplot(aes(x = v_rbeta), data = as.data.frame(v_rbeta)) +
  geom_density() +
  geom_line(data = df, aes(x = x,
                y = y),
            colour = "orange",
            size = 2) +
  geom_line(data = df2, aes(x = x,
                y = y),
            colour = "darkgreen",
            linetype = 2) +
  theme_bw()
```

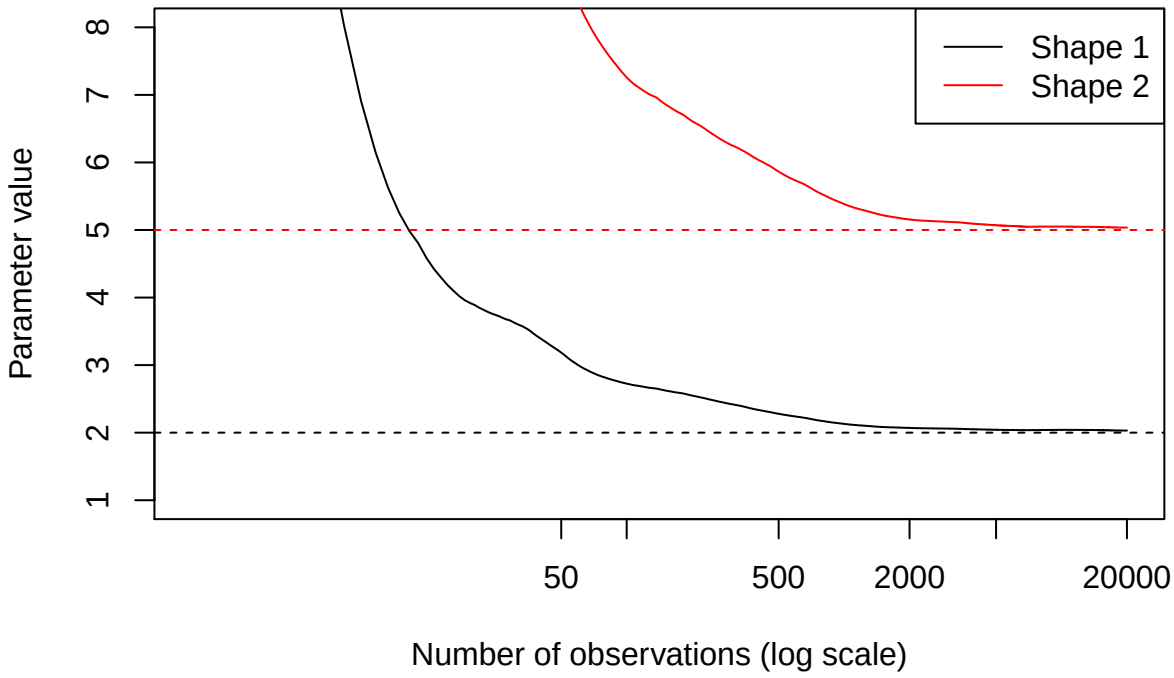


Relation between number of observations, estimated and real parameters

The following graph shows how the value of the estimated parameters based on the method of moments varies based on the number of observation that you have. For this example we again use a beta distribution with shape1 = 2 and shape2 = 5. We draw 20,000 random value of this distribution and for each new draw we re-restimated the parameters of the distribution.

```
v_rbeta2 <- rbeta(20000, shape1 = 2, shape2 = 5) # random beta variable

#m_params <- matrix(NA, nrow = 2,
 #                  ncol = length(v_rbeta2),
  #                 dimnames = list(c("shape1", "shape2"),
   #                                  c(1:length(v_rbeta2))))

m_params <- sapply(1:length(v_rbeta2), function (x){ # matrix of estimated parameters

  mean_beta <- mean(v_rbeta2[c(1:x)])  # calculate mean
  sd_beta <- sd(v_rbeta2[c(1:x)]) # calculate sd
  shape1_mom <- ((mean_beta * (1-mean_beta)/sd_beta^2)-1)*mean_beta # calculate first parameter of the
  shape2_mom <- ((mean_beta * (1-mean_beta)/sd_beta^2)-1)*(1-mean_beta) # calculate second parameter of

  return(c(shape1_mom, shape2_mom))
})

shape1_mean <- cumsum(m_params[1,2:ncol(m_params)])/c(2:ncol(m_params)) # calculate average parameter o
```

```r
shape2_mean <- cumsum(m_params[2,2:ncol(m_params)])/c(2:ncol(m_params)) # calculate average parameter o

df_params <- cbind(Num_obs = c(1:length(v_rbeta2)),
                   Shape_1 = c(NA, shape1_mean),
                   Shape_2 = c(NA, shape2_mean))

plot(log(df_params[, "Num_obs"]), df_params[,"Shape_1"], type = 'l', xlab = 'Number of observations (log
     ylim = c(1, 8))
lines(log(df_params[, "Num_obs"]), df_params[,"Shape_2"], col = 'red')
abline(h = 2, col = 'black', lty = 2)
abline(h = 5, col = 'red', lty = 2)
axis(1, at = c(log(50), log(100), log(500), log(2000), log(5000), log(20000)), labels=c(50, 100, 500, 2
legend("topright",                    # Add legend to plot
       legend = c("Shape 1", "Shape 2"),
       col = c('black', 'red'),
       lty = 1)
```



Fitting distributions based on percentiles

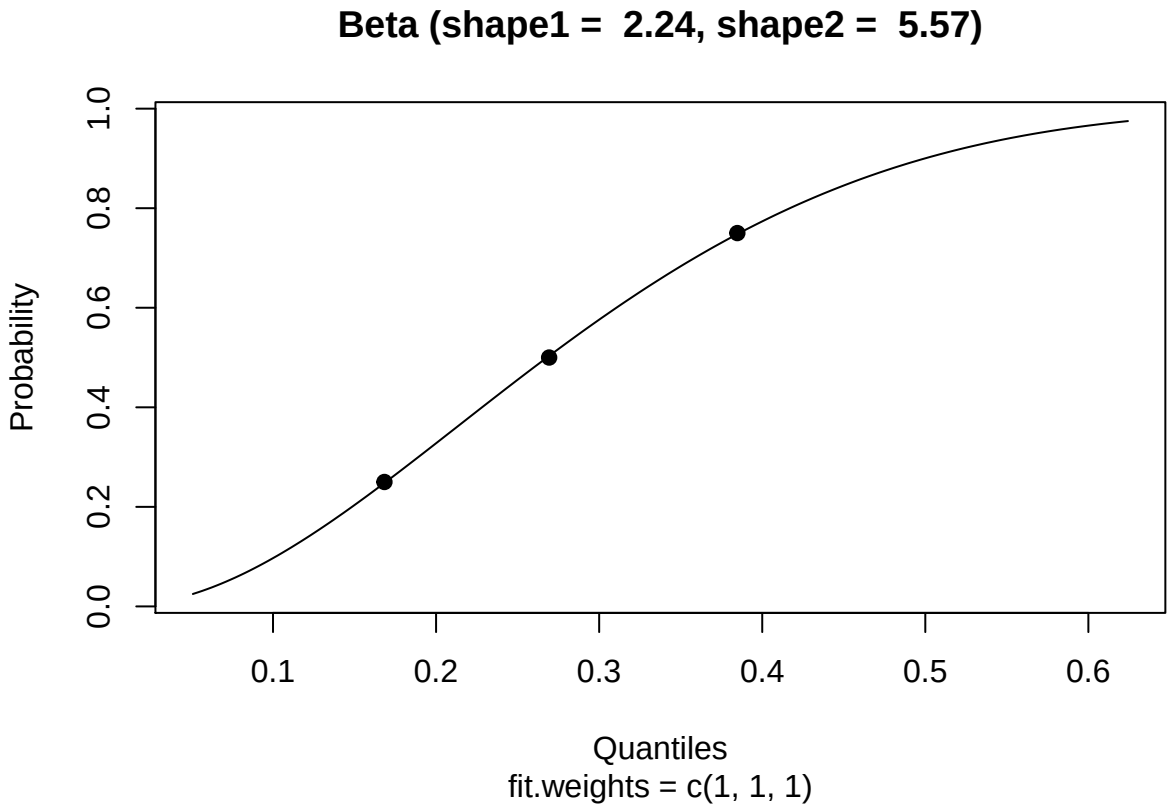
And what if you only have the median and interquartile range values, how could you estimate a distribution based on that? For that, you can use the `rriskDistributions` package. Let's use our beta distribution again. In this case, I have used the `get.beta.par` function because I was interested in getting parameters for a beta distribution. However, if you do not know which type of distribution is asociated with a certain

parameter, you could use the `fit.perc` function.
This can also be done using the `optim` package and function.

```
v_quantiles <- quantile(x = v_rbeta, probs = c(0.25, 0.5, 0.75))# compute first quantile, median, and t
v_quantiles
```

```
##       25%       50%       75%
## 0.1683180 0.2693383 0.3847358
```

```
dist_beta <- get.beta.par(p = c(0.25, 0.5, 0.75), q = v_quantiles, show.output = FALSE) # fit beta dist
```

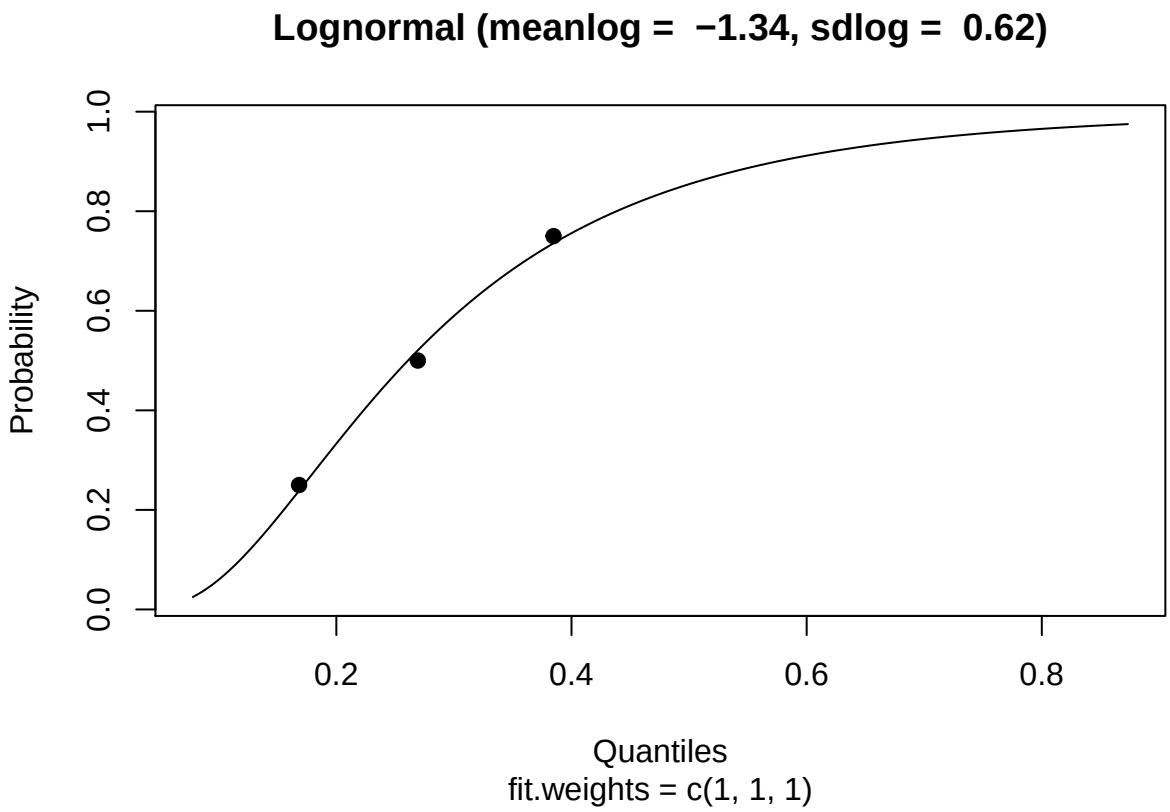



```
dist_beta
```

```
##   shape1   shape2
## 2.240356 5.568768
```

The above-fitted distribution matches the three percentiles quite nicely, because we have controlled the data generating process, by using a beta distribution. However, sometimes, you may not know which type of distribution is underlying the percentiles that you have identified in the literature. So, what would happen in case you do not fit an appropriate distribution, let's have a look! In the below code, we'll fit a lognormal distribution to the three percentiles

```
dist_2 <- get.lnorm.par(p = c(0.25, 0.5, 0.75), q = v_quantiles, show.output = FALSE) # fit lognormal d
```

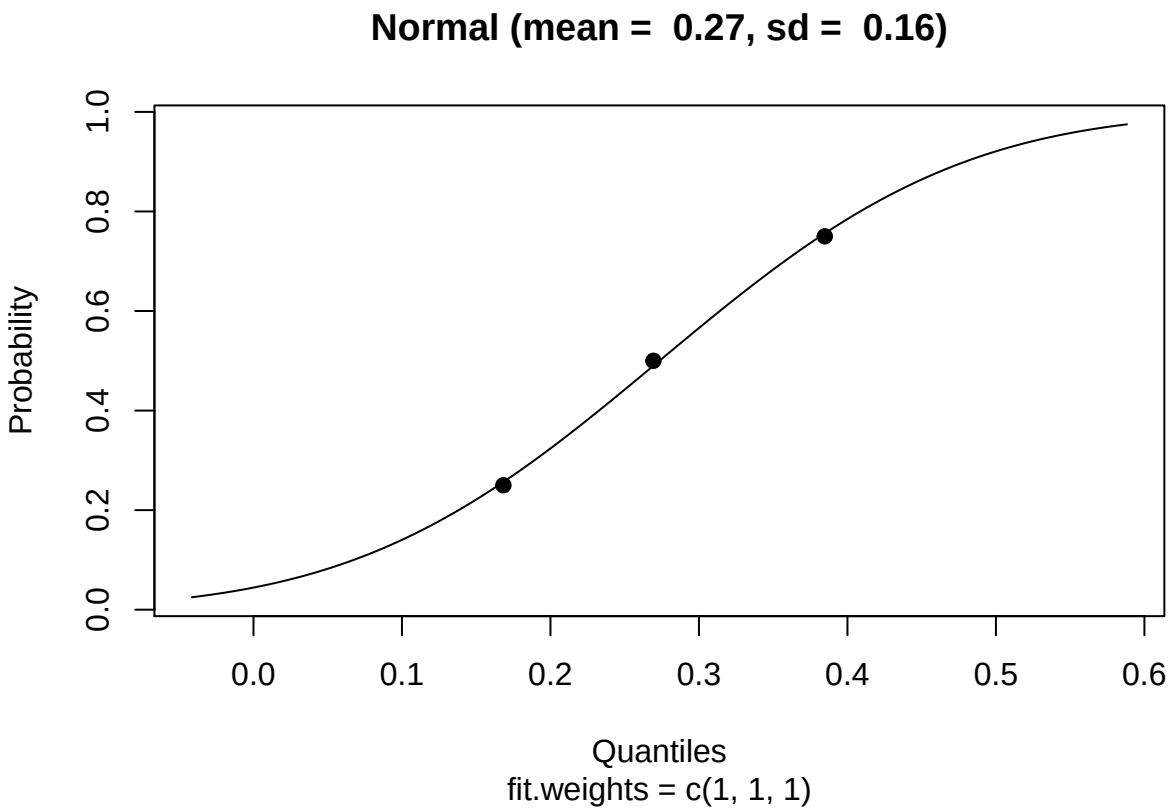



```
dist_2
```

```
##    meanlog      sdlog
## -1.3436439  0.6164074
```

As you can see from this graph, the distribution does not fits as nicely as the beta. If we fit a normal or loglogistic distribution, we may also get negative values (see below), which may not be suitable if the distribution should represent a probability for instance. Hence, always think about the properties of your input parameters before fitting a distribution using percentiles.

```
dist_3 <- get.norm.par(p = c(0.25, 0.5, 0.75), q = v_quantiles, show.output = FALSE) # fit normal distr
```




```
dist_3
```

```
##      mean        sd
## 0.2733633 0.1606528
```

Censored data

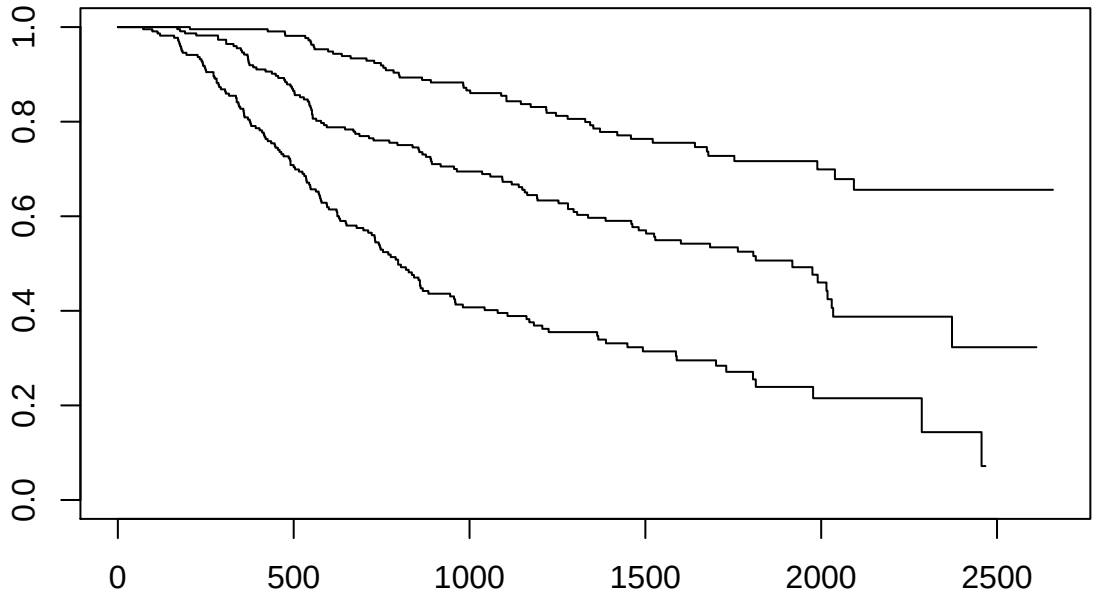
For censored time-to-event data, another package than the ones described above can be used, namely the `flexsurvreg` package. Censoring occurs when one does not observe whether a study participant experiences the event of interest, for instance death. Censored data is often represented through Kaplan-Meier curves and parametric survival curves can also be fitted to these censored data. This is often done to extrapolate the probability of experiencing an event beyond data collection as observed in the graph below.
For this example, we will use the `bc` (breast cancer) data set provided in the `flexsurv` package. in this data set the variable `censrec` represented whether study participant experience the event (1=dead, 0=censored) and rectime represents the time until death or censoring.

```
data("bc")
km <- Surv(time = bc$rectime, event = bc$censrec) # set up survival object
km_group <-survfit(km~group,data=bc,type='kaplan-meier',conf.type='log') # compute Kaplan-Meier curve p
km_group # show survival estimates per group
```

```
## Call: survfit(formula = km ~ group, data = bc, type = "kaplan-meier",
##     conf.type = "log")
```

```
##
##                n events median 0.95LCL 0.95UCL
## group=Good   229     51     NA      NA      NA
## group=Medium 229    103   1918    1502    2034
## group=Poor   228    145    797     722     956
```

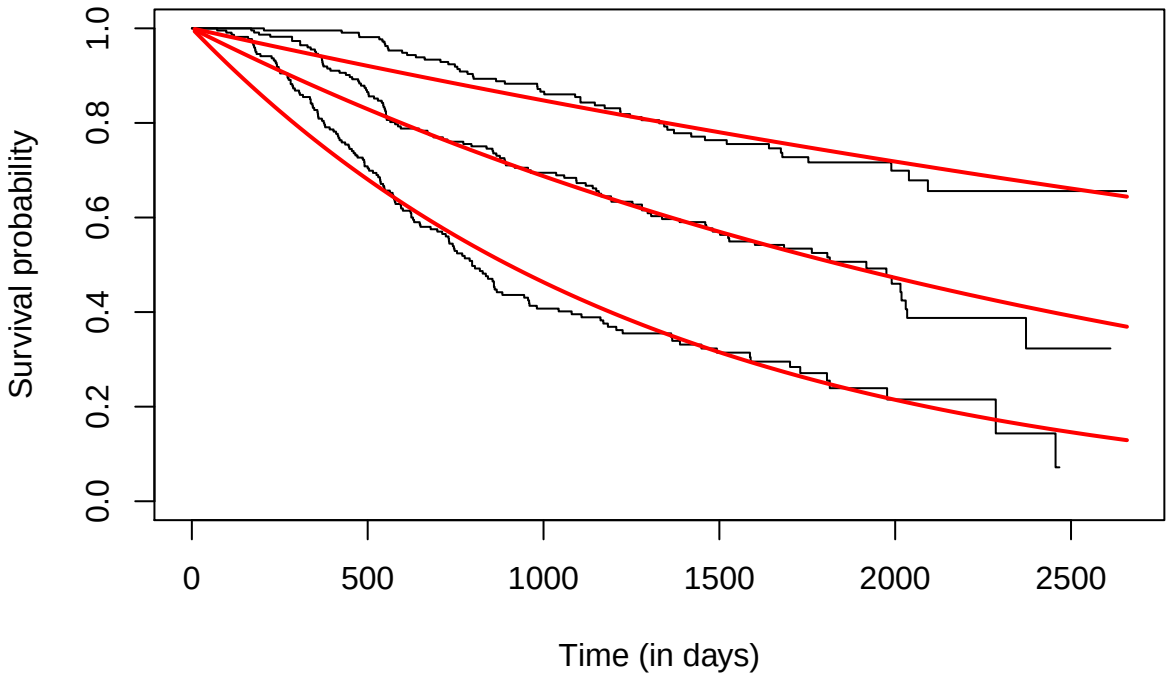
```
plot(km_group) # plot Kaplan-Meier curve per group
```



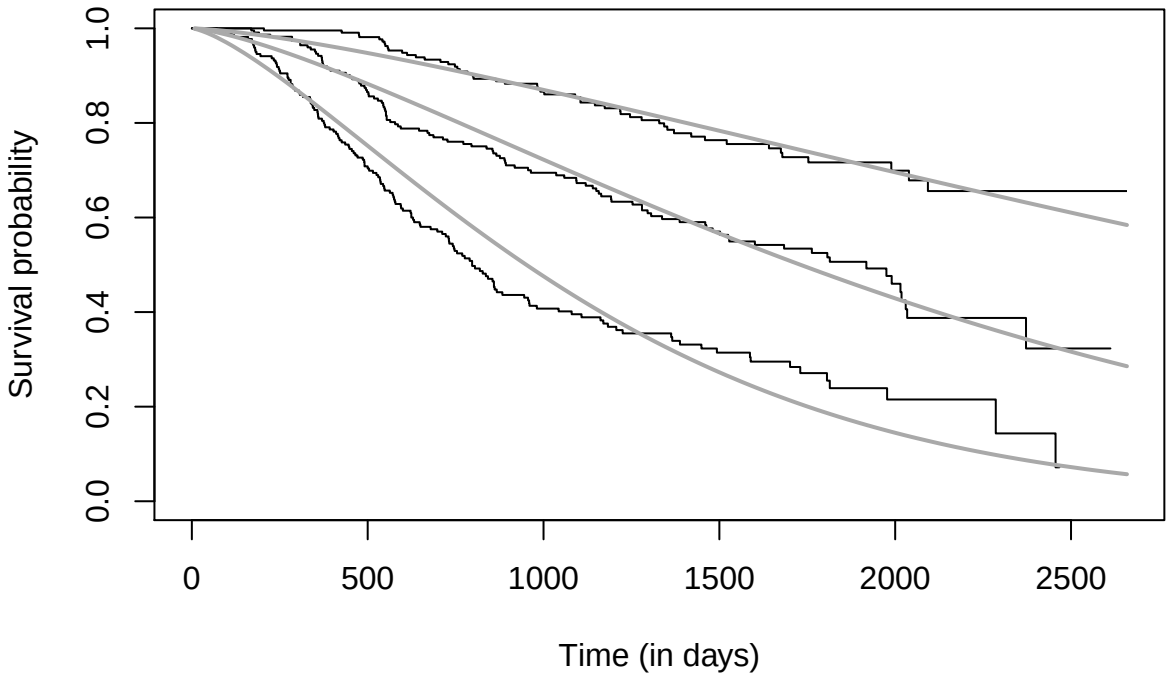

```
dist_surv_expo <- flexsurvreg(km~group, data = bc, dist = "expo")
dist_surv_weib <- flexsurvreg(km~group, data = bc, dist = "weibull")
dist_surv_gamma <- flexsurvreg(km~group, data = bc, dist = "gamma")

AIC(dist_surv_expo, dist_surv_weib, dist_surv_gamma) # compared Akaike Information Criterion (goodness-
```

```
##                 df      AIC
## dist_surv_expo   3 5196.351
## dist_surv_weib   4 5160.022
## dist_surv_gamma  4 5147.735
```

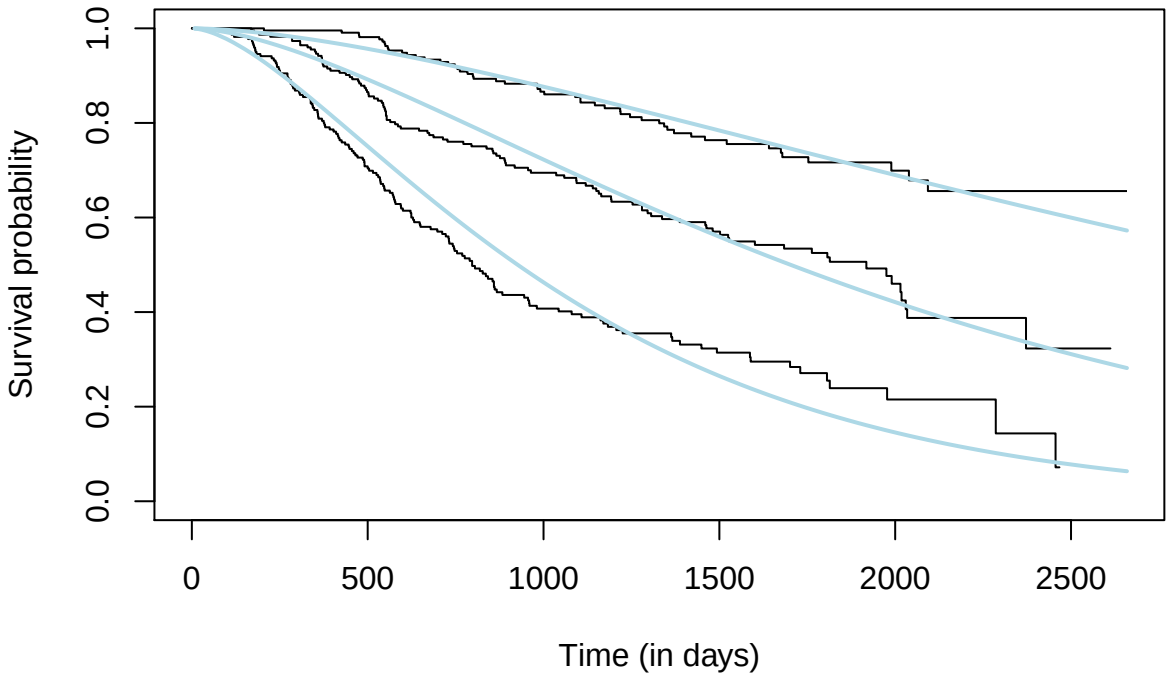
```
# plot in comparison with data
plot(dist_surv_expo, ylab = 'Survival probability', xlab = 'Time (in days)')
```



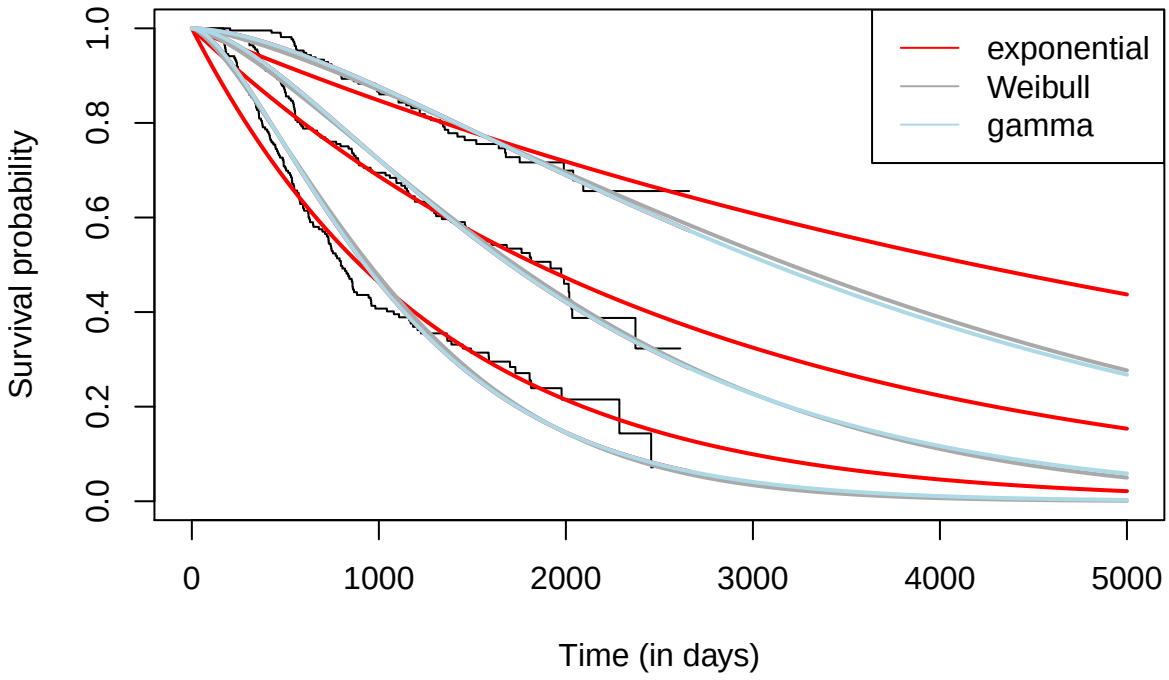

```
plot(dist_surv_weib, col = 'darkgray', ylab = 'Survival probability', xlab = 'Time (in days)')
```




```
plot(dist_surv_gamma, col = 'lightblue', ylab = 'Survival probability', xlab = 'Time (in days)')
```




```
plot(dist_surv_gamma, xlim = c(0, 5000), ylab = 'Survival probability', xlab = 'Time (in days)') # plot
lines(dist_surv_expo, t = c(0:5000))# plot extrapolation
lines(dist_surv_weib, t = c(0:5000), col = 'darkgray')# plot extrapolation
lines(dist_surv_gamma, t = c(0:5000), col = 'lightblue')# plot extrapolation
legend("topright",                    # Add legend to plot
       legend = c("exponential", "Weibull", "gamma"),
       col = c('red', 'darkgray', 'lightblue'),
       lty = 1)
```


exponential
Weibull
gamma
Survival probability
0.0
0.2
0.4
0.6
0.8
1.0
0
1000
2000
3000
4000
5000
Time (in days)