Solutions exam 18/19

gherardo varando

Load the data

```
rm(list = ls()) ## clean workspace
load("exam1819.RData") ## change the path to your location
ls()
```

```
## [1] "bodyfat"   "radiation" "wind"
```

Problem 1

We look at the data with a contingency table

```
table(radiation)
```

```
## , , dead = 0
##
##      treatment
## dose   0  1
##   201 25 32
##   220 31 76
##   243  8 43
##   260  3  6
##
## , , dead = 1
##
##      treatment
## dose   0  1
##   201 10  3
##   220 89 44
##   243 83 48
##   260 12  9
```

Q1.1

To estimate probabilities of death we use empirical frequencies (MLE estimates for Bernoulli random variables). We can do it with the `table` function.

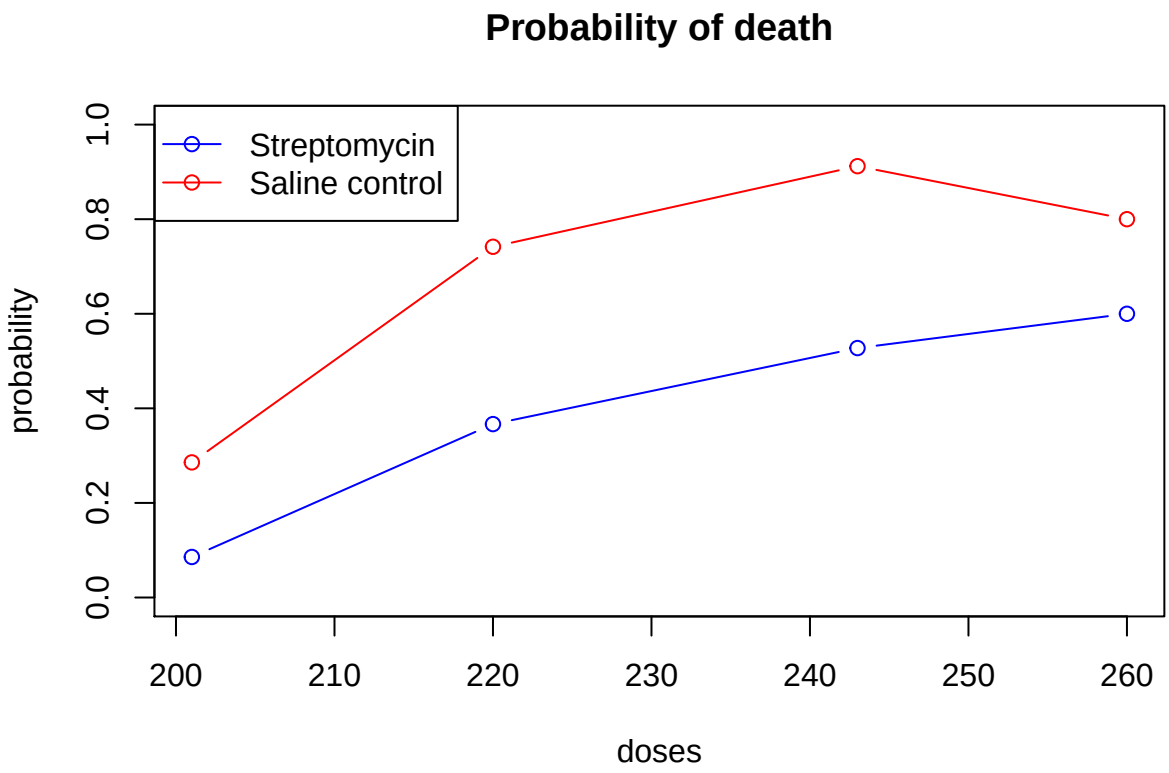
```
probs_death <- table(radiation)[,,2] /
  (table(radiation)[,,1] + table(radiation)[,,2])
probs_death
```

```
##      treatment
## dose           0          1
```

```
##   201 0.28571429 0.08571429
##   220 0.74166667 0.36666667
##   243 0.91208791 0.52747253
##   260 0.80000000 0.60000000
```

We can now plot everything

```
doses <- dimnames(probs_death)$dose
plot(doses, probs_death[, 1], type = "b", col = "red", ylim = c(0,1),
     ylab = "probability", main = "Probability of death")
lines(doses, probs_death[, 2], type = "b", col = "blue")
legend("topleft", legend = c("Streptomycin", "Saline control"),
       col = c("blue", "red"), lty = 1, pch = 1)
```



From the plot we can observe that mice treated with Streptomycin have lower estimated probabilities of dying. If this difference is statistical significant is the subject of the next two questions.

Q1.2

We use here the Chi-squared test for independence to test if, under different radiation doses, the number of dead mice is independent of the treatment.

```
alpha <- 0.05
results <- t(sapply(doses, function(d){
  tmp <- chisq.test(table(radiation[radiation$dose == d, c(2,3)]),
                    simulate.p.value = F)
  return(list(statistic = tmp$statistic,
```

```
              pvalue = tmp$p.value,
              reject = tmp$p.value < alpha))
  }))

results
```

```
##     statistic pvalue       reject
## 201 3.40081   0.06516442   FALSE
## 220 32.64985  1.10348e-08  TRUE
## 243 31.49109  2.003571e-08 TRUE
## 260 0.6349206 0.4255561    FALSE
```

For `dose == 220` and `dose == 243` we can reject the null hypothesis of independence, and thus we can say that (at a level 0.05) there is a statistically significant dependence between `treatment` and `dead`. For the lowest and highest radiation dose we can not reject the null hypothesis of independence.

Q1.3

We want now to perform a one-sided Wald test using the statistic $\hat{\delta} = \hat{p}_1 - \hat{p}_0$. Where $\hat{p}_1$ is the estimated death probability under Streptomycin treatment and $\hat{p}_0$ is the estimated death probability under the saline control treatment. Both this estimated probability are the MLE estimations of the parameter p of two Bernoulli random variables (`dead` | (`treatment = 0/1`, `dose = ...`))

$\hat{\delta}$ is an asymptotically normal statistic and the standard error is

$$se(\hat{\delta}) = \sqrt{\mathbb{V}(\hat{p}_1 - \hat{p}_0)} = \sqrt{\mathbb{V}(\hat{p}_1) + \mathbb{V}(\hat{p}_0)}$$

Moreover we have that

$$\mathbb{V}(\hat{p}_i) = \frac{p_i(1-p_i)}{n_i} \approx \frac{\hat{p}_i(1-\hat{p}_i)}{n_i} \quad i = 0, 1$$

where n_i is the sample size used to estimate $\hat{p}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} X_j$.

Thus an approximation of the standard error is:

$$\hat{se}(\hat{\delta}) = \sqrt{\frac{\hat{p}_0(1-\hat{p}_0)}{n_0} + \frac{\hat{p}_1(1-\hat{p}_1)}{n_1}}$$

In our case $n_0 = n_1$ and thus the formula for the standard error could be simplified, but we keep the more general expression.

We now compute the value of $\hat{\delta}$ and the standard error for the different radiation doses.

```
Wald.stats <- t(sapply(doses, function(d){
  probs <- probs_death[d,]
  n1 <- sum(radiation$dose == d & radiation$treatment == 1)
  n0 <- sum(radiation$dose == d & radiation$treatment == 1) ## in this case they are equal
  delta <- probs[2] - probs[1] ## the order is inverted
  se <- sqrt(probs[1]*(1 - probs[1]) / n0 + probs[2] * (1 - probs[2]) / n1)
  return(list(delta = delta, se = se, W = delta / se))
}))
Wald.stats
```

```
##     delta      se         W
## 201 -0.2       0.08983302 -2.226353
## 220 -0.375     0.05942919 -6.31003
```

```
## 243 -0.3846154 0.06016724 -6.392438
## 260 -0.2       0.1632993  -1.224745
```

We want now to test the following hypothesis (be careful that probability of surviving = 1− probability of dying and)

$$H_0 : \hat{\delta} >= 0$$
$$H_1 : \hat{\delta} < 0$$

So $\hat{\delta}$ is extreme for H_0 if $\hat{\delta} << 0$. And assuming asymptotic normality of $\hat{delta}$, we have that $\hat{\delta}$ is extreme at a significance level of α if $W = \frac{\hat{\delta}}{\hat{se}(\hat{\delta})} < Q_Z(\alpha)$ where Q_Z is the standard Gaussian quantile function, observe that $Q_Z(\alpha) < 0$ for small α values which makes sense. We can thus implement the Wald test:

```
alpha <- 0.05
z <- qnorm(alpha)
t(sapply(doses, function(d){
  W <- Wald.stats[d,]$W
  return(list(W = W, reject = W < z, pvalue = pnorm(W)))
}))
```

```
##     W         reject pvalue
## 201 -2.226353 TRUE   0.01299528
## 220 -6.31003  TRUE   1.394903e-10
## 243 -6.392438 TRUE   8.163046e-11
## 260 -1.224745 FALSE  0.1103357
```

We can reject the null hypothesis that mice treated with Streptomycin are less
or equal probable to survive than mice treated with the saline control for radiation doses equal to $201, 220$ and 243, thus we can say that at an significance level of 0.05, Streptomycin is effective in contrasting the effects of radiations for those doses. While for radiation dose of 260 we do not have enough evidence to reject the null hypothesis and thus we can not say that Streptomycin is effective.

Q1.4

We perform here logistic regression to predict the probability of death as a function of the neutron dose for mice treated with Streptomycin only.

The first model is a simple logistic regression,

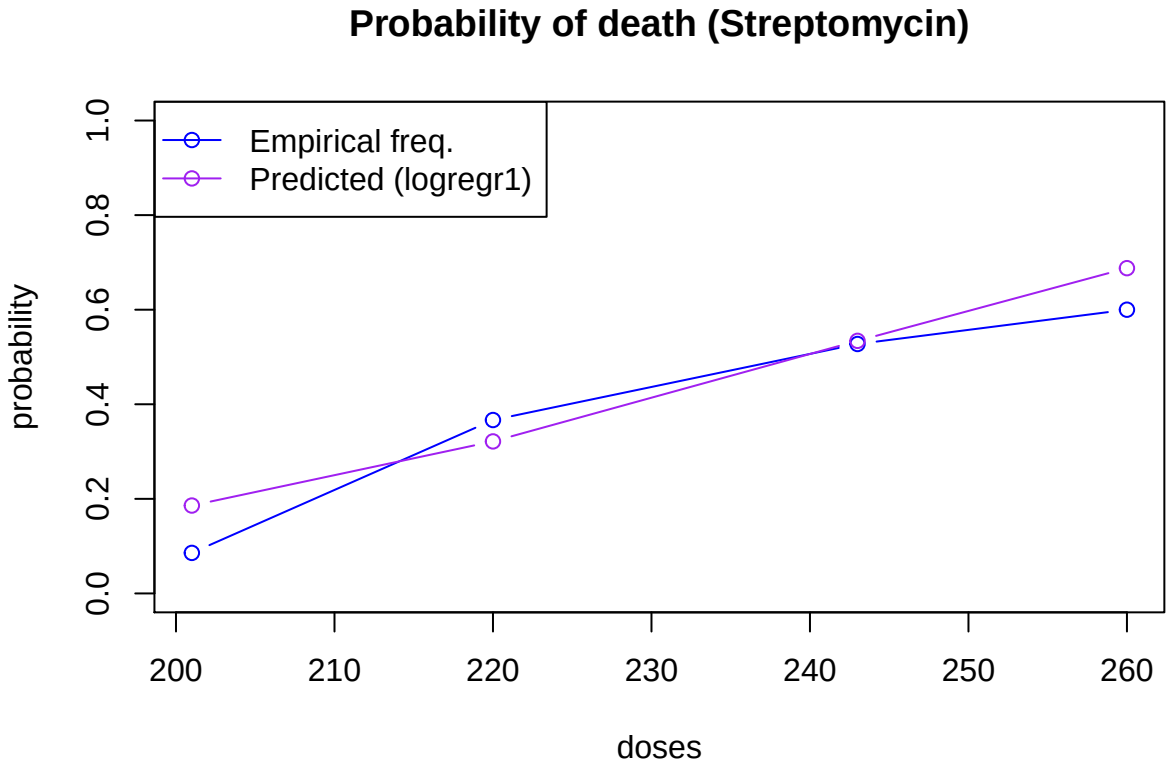
```
logregr1 <- glm(dead ~ dose, family = binomial,
                data = radiation[radiation$treatment == 1,])
summary(logregr1)
```

```
##
## Call:
## glm(formula = dead ~ dose, family = binomial, data = radiation[radiation$treatment ==
##     1, ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5259  -0.8807  -0.6413   1.1200   1.8346
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.200832   1.959687  -4.695 2.67e-06 ***
## dose         0.038426   0.008517   4.512 6.43e-06 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 350.99  on 260  degrees of freedom
## Residual deviance: 328.67  on 259  degrees of freedom
## AIC: 332.67
##
## Number of Fisher Scoring iterations: 4
```

We plot now the estimated probabilities, remember that to extract the probabilities we need to use the inverse of the link function or set `type = "response"` in the `predict` function.

```
doses <- as.numeric(doses) ##we need numeric values now
predicted_probs <- predict(logregr1, type = "response", newdata = data.frame(dose = doses))

### now we just repeta the plot of Q1.1
plot(doses, probs_death[, 2], type = "b", col = "blue", ylim = c(0,1),
     ylab = "probability", main = "Probability of death (Streptomycin)")
lines(doses, predicted_probs, type = "b", col = "purple")
legend("topleft", legend = c("Empirical freq.", "Predicted (logregr1)"),
       col = c("blue", "purple"), lty = 1, pch = 1)
```



We can see that the predicted probabilities are very close to the empirical frequencies.

We fit now two logistic regression models with polynomial terms.

```r
logregr2 <- glm(dead ~ dose + I(dose^2), family = binomial,
                data = radiation[radiation$treatment == 1,])
coefficients(logregr2)
```

```
##   (Intercept)          dose     I(dose^2)
## -6.175839e+01  4.966012e-01 -9.941736e-04
```

```r
logregr3 <- glm(dead ~ dose + I(dose^2) + I(dose^3),
                family = binomial, data = radiation[radiation$treatment == 1,])
coefficients(logregr2)
```

```
##   (Intercept)          dose     I(dose^2)
## -6.175839e+01  4.966012e-01 -9.941736e-04
```

Models election using AIC,

```r
AIC(logregr1, logregr2, logregr3)
```

```
##          df      AIC
## logregr1  2 332.6737
## logregr2  3 331.0922
## logregr3  4 332.2617
```

Using AIC we would select the model with the quadratic term `logregr2`.

We perform now the LRT between `logregr1` and `logregr2`

```r
anova(logregr1, logregr2, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: dead ~ dose
## Model 2: dead ~ dose + I(dose^2)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       259     328.67
## 2       258     325.09  1   3.5815  0.05843 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can not reject the null hypothesis that the simpler model `logregr1` is enough to model the data.

Problem 2

###Q2.1

We fit a linear regression model using all the available predictors.

```r
fitAll <- lm(fat ~ ., data = bodyfat)
summary(fitAll)
```

```
##
## Call:
## lm(formula = fat ~ ., data = bodyfat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1596  -2.8548  -0.0893   3.2070  10.0482
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.492266  22.243941  -0.921  0.35786
## age           0.062126   0.032416   1.916  0.05650 .
## weight       -0.206398   0.136774  -1.509  0.13262
## height       -1.754027   7.043080  -0.249  0.80354
## neck         -0.463918   0.236396  -1.962  0.05088 .
## chest        -0.019125   0.103370  -0.185  0.85338
## abdomen       0.959094   0.090451  10.604  < 2e-16 ***
## hip          -0.205217   0.146878  -1.397  0.16366
## thigh         0.240235   0.146783   1.637  0.10303
## knee          0.006447   0.248244   0.026  0.97930
## ankle         0.177386   0.222858   0.796  0.42685
## biceps        0.185049   0.172728   1.071  0.28511
## forearm       0.451047   0.199624   2.259  0.02476 *
## wrist        -1.618680   0.536172  -3.019  0.00281 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.314 on 237 degrees of freedom
## Multiple R-squared:  0.7463, Adjusted R-squared:  0.7324
## F-statistic: 53.64 on 13 and 237 DF,  p-value: < 2.2e-16
```

`abdomen`, `forearm` and `wrist` are probably the most relevant predictors, since the p-values are very small (<0.05).

For the variable `knee` the reported p-value of the t-test is 0.97930, we can thus not reject the null hypothesis that the coefficient for `knee` is 0.

Q2.2

We want to use now stepwise selection (forward and backward).

We are thus fitting two models,

Forward model:

```
fit0 <- lm(fat ~ 1, data = bodyfat)
fitForw <- step(object = fit0, scope = formula(fitAll), direction = "forward",
                trace = 0, k = log(nrow(bodyfat)))
summary(fitForw)
```

```
##
## Call:
## lm(formula = fat ~ abdomen + weight + wrist + forearm, data = bodyfat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.6012  -3.2504  -0.0821   3.1295   9.0853
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -35.60134    7.28236  -4.889 1.83e-06 ***
## abdomen       0.99417    0.05609  17.726  < 2e-16 ***
## weight       -0.30228    0.05465  -5.531 8.14e-08 ***
## wrist        -1.44497    0.44669  -3.235  0.00138 **
```

```
## forearm       0.47419    0.18166   2.610  0.00960 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.343 on 246 degrees of freedom
## Multiple R-squared:  0.7332, Adjusted R-squared:  0.7289
## F-statistic:   169 on 4 and 246 DF,  p-value: < 2.2e-16
```

Backward model:

```
fitBack <- step(fitAll, direction = "backward", trace = 0, k = log(nrow(bodyfat)))
summary(fitBack)
```

```
##
## Call:
## lm(formula = fat ~ weight + abdomen + forearm + wrist, data = bodyfat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.6012  -3.2504  -0.0821   3.1295   9.0853
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -35.60134    7.28236  -4.889 1.83e-06 ***
## weight       -0.30228    0.05465  -5.531 8.14e-08 ***
## abdomen       0.99417    0.05609  17.726  < 2e-16 ***
## forearm       0.47419    0.18166   2.610  0.00960 **
## wrist        -1.44497    0.44669  -3.235  0.00138 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.343 on 246 degrees of freedom
## Multiple R-squared:  0.7332, Adjusted R-squared:  0.7289
## F-statistic:   169 on 4 and 246 DF,  p-value: < 2.2e-16
```

The two models are exactly the same since the selected covariates are: `weight`, `abdomen`, `forearm` and `wrist` in both cases (we can thus just consider one of the two models).

This question could also be interpreted as performing stepwise selection using both directions, I accepted both answers as correct.

Q2.3

First of all we compute the BMI, we also add it as a new variable in the dataset.

```
bodyfat$bmi <- bodyfat$weight / (bodyfat$height^2)
```

Then we fit the linear model using `bmi` and `age`:

```
fitBMI <- lm(fat ~ bmi + age, data = bodyfat)
summary(fitBMI)
```

```
##
## Call:
## lm(formula = fat ~ bmi + age, data = bodyfat)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -21.6180  -4.0314  -0.1648   3.8417  12.6138
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -27.74978    2.61796 -10.600  < 2e-16 ***
## bmi           1.59681    0.09555  16.711  < 2e-16 ***
## age           0.14056    0.02768   5.078 7.51e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.49 on 248 degrees of freedom
## Multiple R-squared:  0.5702, Adjusted R-squared:  0.5667
## F-statistic: 164.5 on 2 and 248 DF,  p-value: < 2.2e-16
```

Q2.4

We compute here 95% CI using bootstrap and the percentile method.

```
M <- 10000
n <- nrow(bodyfat)
coeffs <- replicate(M, {
  fittemp <- lm(formula(fitBMI), bodyfat[sample(1:n, replace = TRUE),])
  return(coefficients(fittemp))
})
alpha <- 0.05
t(apply(coeffs, MARGIN = 1, quantile, probs = c(alpha/2, 1 - alpha/2)))
```

```
##                      2.5%       97.5%
## (Intercept) -36.01124441 -20.3130563
## bmi           1.29897693   1.9329215
## age           0.09033151   0.1902559
```

Using the built-in function `confint`:

```
confint(fitBMI)
```

```
##                    2.5 %      97.5 %
## (Intercept) -32.90604710 -22.5935166
## bmi           1.40860663   1.7850115
## age           0.08604041   0.1950836
```

Q2.5

We compare here the models above using BIC and errors estimated with cross validation.

BIC:

```
BIC(fitForw, fitBMI)
```

```
##         df      BIC
## fitForw  6 1477.576
## fitBMI   4 1586.255
```

Using BIC the model obtained with stepwise selection is selected.

Cross validation:

```r
squaredres_cv <- sapply(1:nrow(bodyfat), function(i){
  tmp_step <- lm(formula(fitForw), data = bodyfat[-i,])
  tmp_bmi <- lm(formula(fitBMI), data = bodyfat[-i,])
  prd_step <- predict(tmp_step, newdata = bodyfat[i,])
  prd_bmi <- predict(tmp_bmi, newdata = bodyfat[i,])
  return(c(step = (prd_step - bodyfat$fat[i])^2, bmi =  (prd_bmi - bodyfat$fat[i])^2))
} )
rowSums(squaredres_cv)
```

```
##   step.1    bmi.1
## 4886.385 7839.209
```

The stepwise selected model has a lower residual sum of squares estimated with leave-one-out.

Problem 3

Q3.1

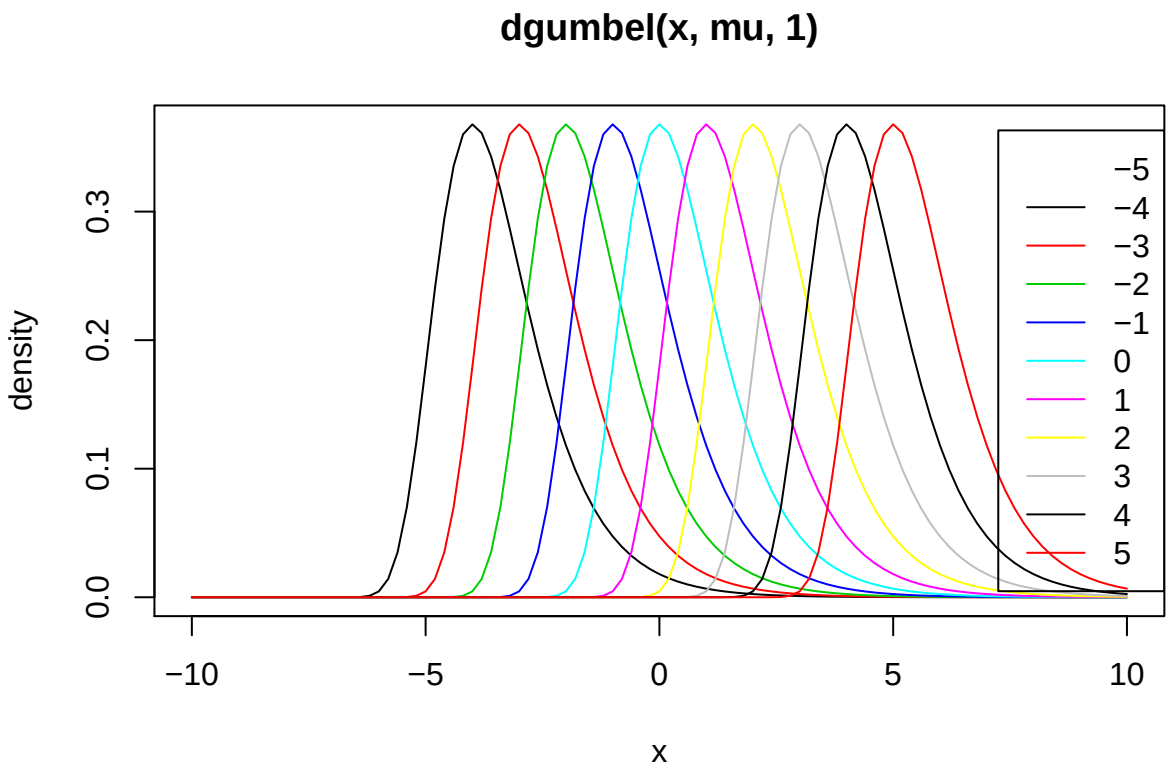
The implementation of the functions are as follow, we also show here how to use `roxygen2` style of comments for functions. We do not implement check on the parameters but that can be done easily.

```r
#' Density of Gumbel distribution
#'
#' @param x  vector of quantiles
#' @param mu location parameter
#' @param b  scale parameter
#' @param log logical; if \code{TRUE} the values of the log-density are returned
#'
#' @return the vector of the values of the density is returned
dgumbel <- function(x, mu = 0, b = 1, log = FALSE){ # PDF
  temp <- (mu - x) / b - exp((mu - x)/b)  - log(b)
  if (log){
    return(temp)
  }else{
  return(exp(temp))
  }
}

#' Cumulative distribution function of the Gumbel distribution
#'
#' @param q  vector of quantiles
#' @param mu location parameter
#' @param b  scale parameter
#' @param log.p logical; if \code{TRUE} the values of the log probabilities are returned
#'
#' @return the vector of the values of the CDF is returned
pgumbel <- function(q, mu = 0, b = 1, lower.tail = TRUE, log.p = FALSE){
  temp <- - exp(-((q - mu)/b))
  if (log.p){
    if (lower.tail){
     return(temp)
    }else{
     return( log(1 - exp(temp) ) )
    }
```

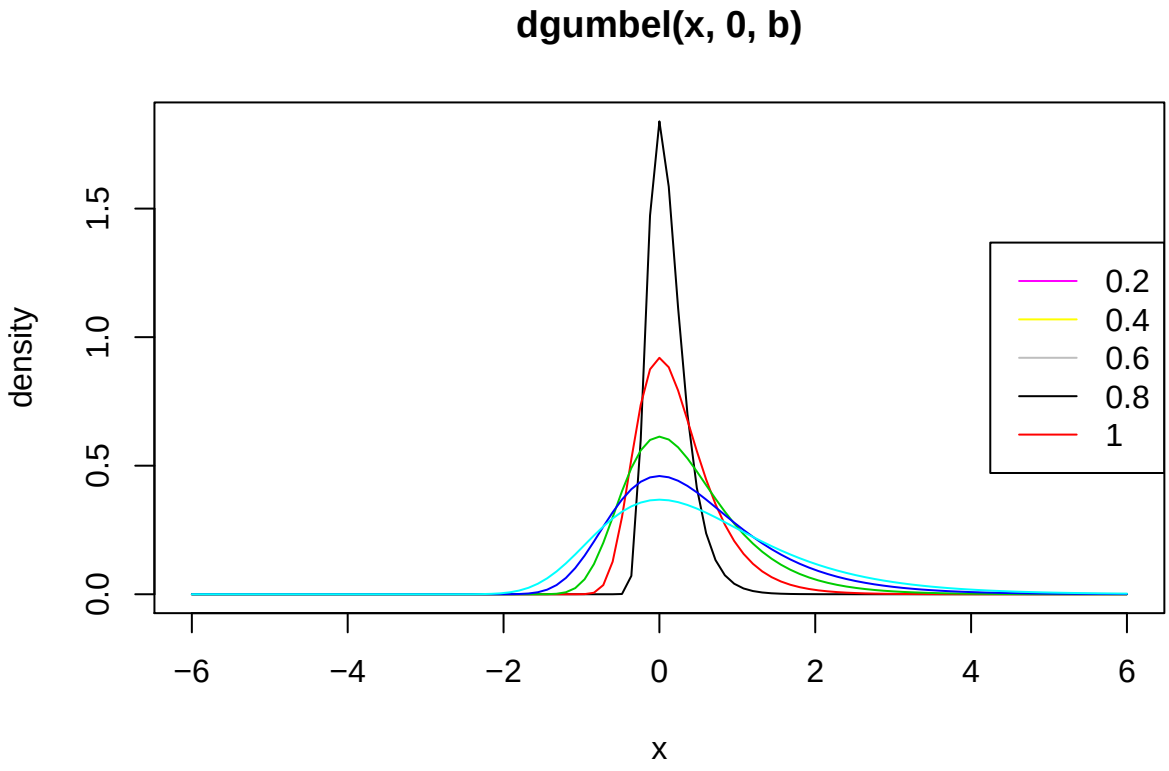
```
  }else{
    if (lower.tail){
      return(exp(temp))
    }else{
      return(1 - exp(temp))
    }
  }
}

#' Cumulative distribution function of the Gumbel distribution
#'
#' @param p  vector of probabilities
#' @param mu location parameter
#' @param b  scale parameter
#' @param log.p logical; if \code{TRUE} the values of the log probabilities are returned
#'
#' @return the vector of corresponding quantiles is returned
qgumbel <- function(p, mu = 0, b = 1, lower.tail = TRUE, log.p = FALSE){
  if (lower.tail){
    if (!log.p) p <- log(p)
  }else{
    if (!log.p) p <- log(1 - p)
    else p <- 1 - p
  }
  mu-b*log(-p)
}

#' Random generation following Gumbel distribution
#'
#' @param n  sample size
#' @param mu location parameter
#' @param b  scale parameter
#'
#' @return the sample of size \code{n} is returned
rgumbel <- function(n = 1, mu = 0, b = 1){
  qgumbel(runif(n), mu = mu, b = b)
}
```

We want now to test our implementation, we start by plotting the first three functions.

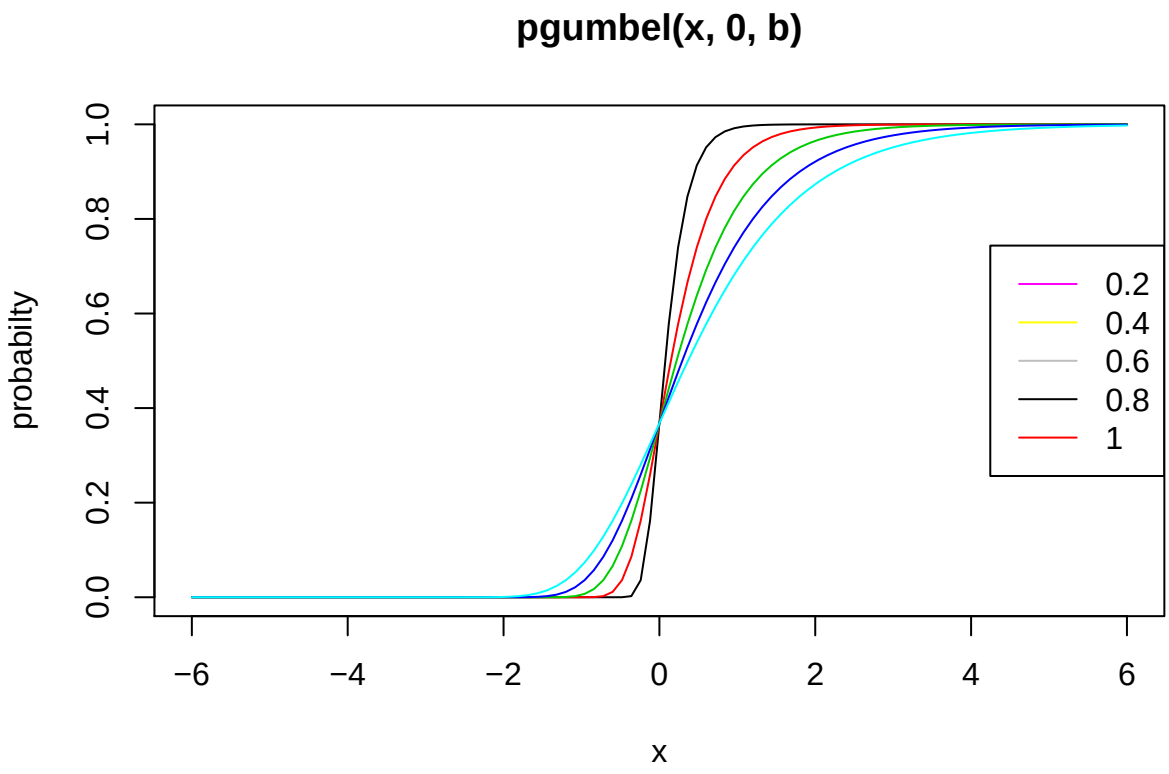
```
first <- FALSE
for (mu in -5:5){
  curve(dgumbel(x, mu = mu, b =1), from = -10, to = 10, col = (mu + 5), add = first,
        main = "dgumbel(x, mu, 1)", ylab = "density")
  first <- TRUE
}
legend("right", legend = -5:5, col = -5:5 + 5, lty = 1)
```



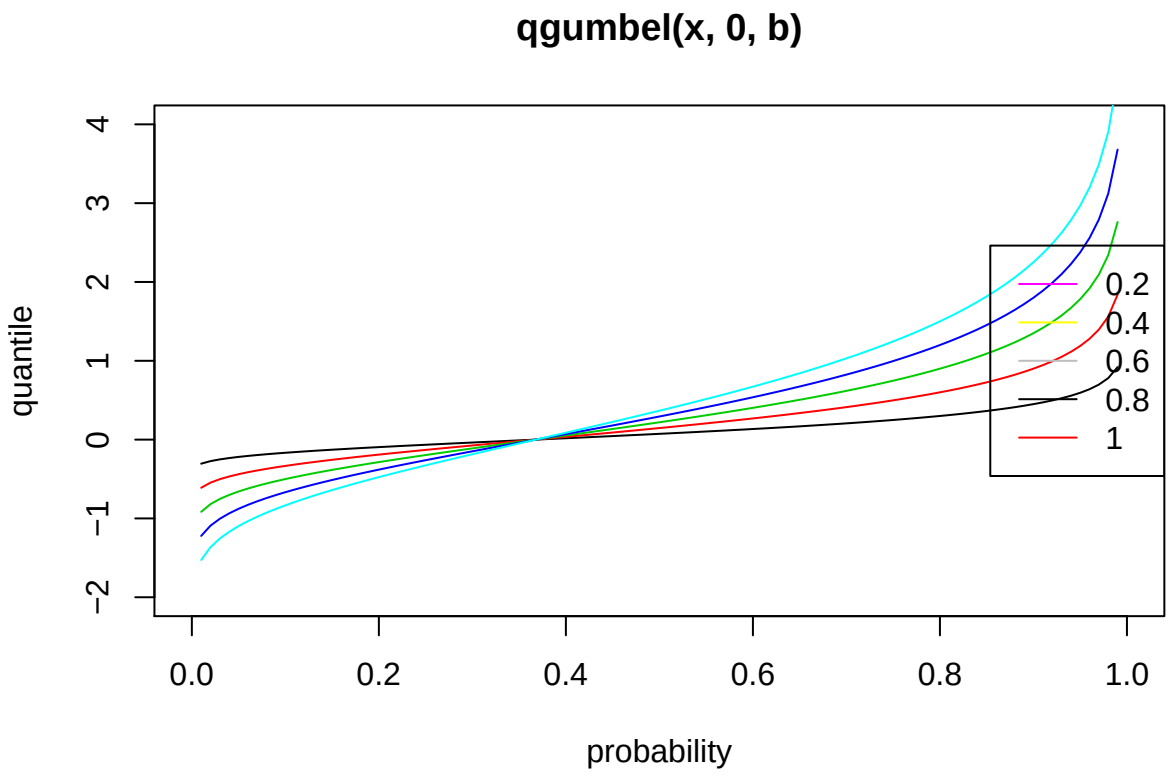

```
first <- FALSE
for (b in 1:5){
  curve(dgumbel(x, mu = 0, b = b / 5), from = -6, to = 6, col = (b), add = first,
        main = "dgumbel(x, 0, b)", ylab = "density")
  first <- TRUE
}
legend("right", legend = (1:5 ) / 5, col = 1:5 + 5, lty = 1)
```




```
first <- FALSE
for (b in 1:5){
  curve(pgumbel(x, mu = 0, b = b / 5), from = -6, to = 6, col = (b), add = first,
        main = "pgumbel(x, 0, b)", ylab = "probabilty")
  first <- TRUE
}
legend("right", legend = (1:5 ) / 5, col = 1:5 + 5, lty = 1)
```

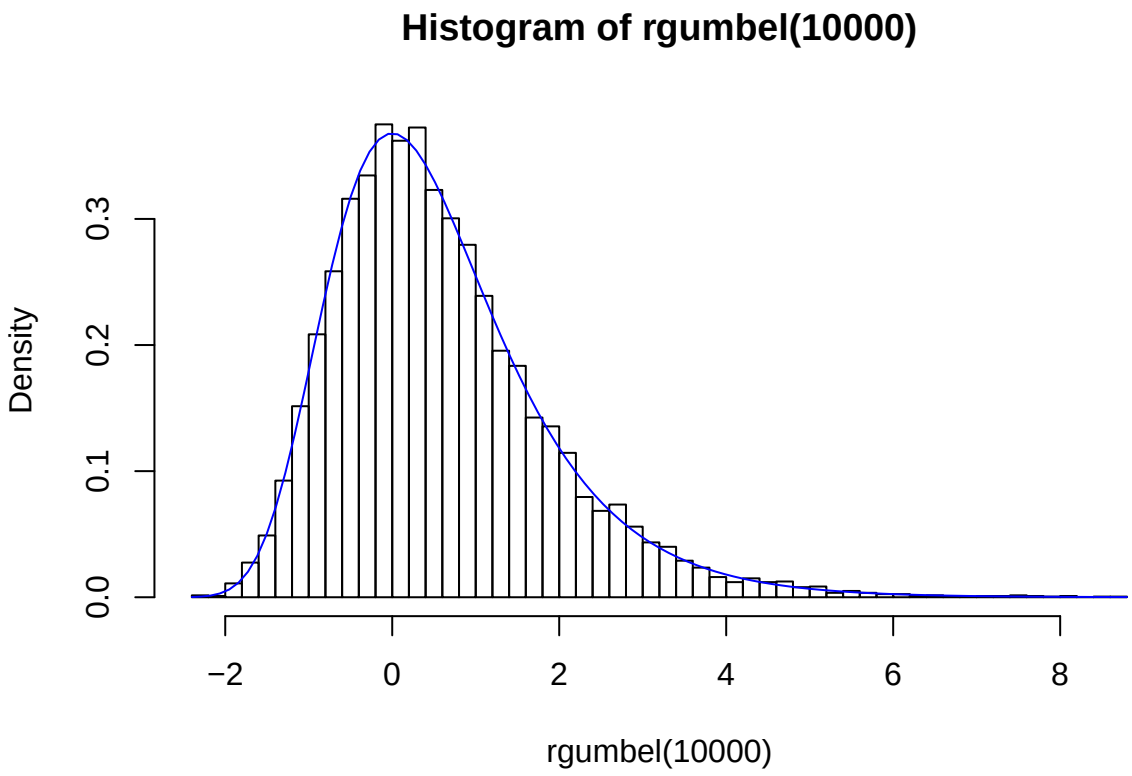



```
first <- FALSE
for (b in 1:5){
  curve(qgumbel(x, mu = 0, b = b / 5), from = 0, to = 1, col = (b), add = first,
        main = "qgumbel(x, 0, b)", ylab = "quantile", xlab = "probability", ylim = c(-2, 4))
  first <- TRUE
}
legend("right", legend = (1:5 ) / 5, col = 1:5 + 5, lty = 1)
```



We can also generate a sample and plot the histogram with the true density.

```
hist(rgumbel(10000), breaks = "FD", probability = TRUE)
curve(dgumbel(x), add = TRUE, col = "blue")
```



From the plots we can see that the functions behaves correctly, but we want to perform now some quantitative checks.

First of all we check that `dgumbel` is positive. We check it for x ranging from -1000 to 1000 and μ ranging from -100 to 100 and β from 1 to 10.

```
## if this code return TRUE it is ok
all(sapply(-100:100, function(mu){
  sapply(1:10, function(b){
    all(dgumbel(-1000:1000, mu = mu, b = b) >= 0)
  })
}))
```

```
## [1] TRUE
```

Then we check that `dgumbel` integrate to 1 (with some precision).

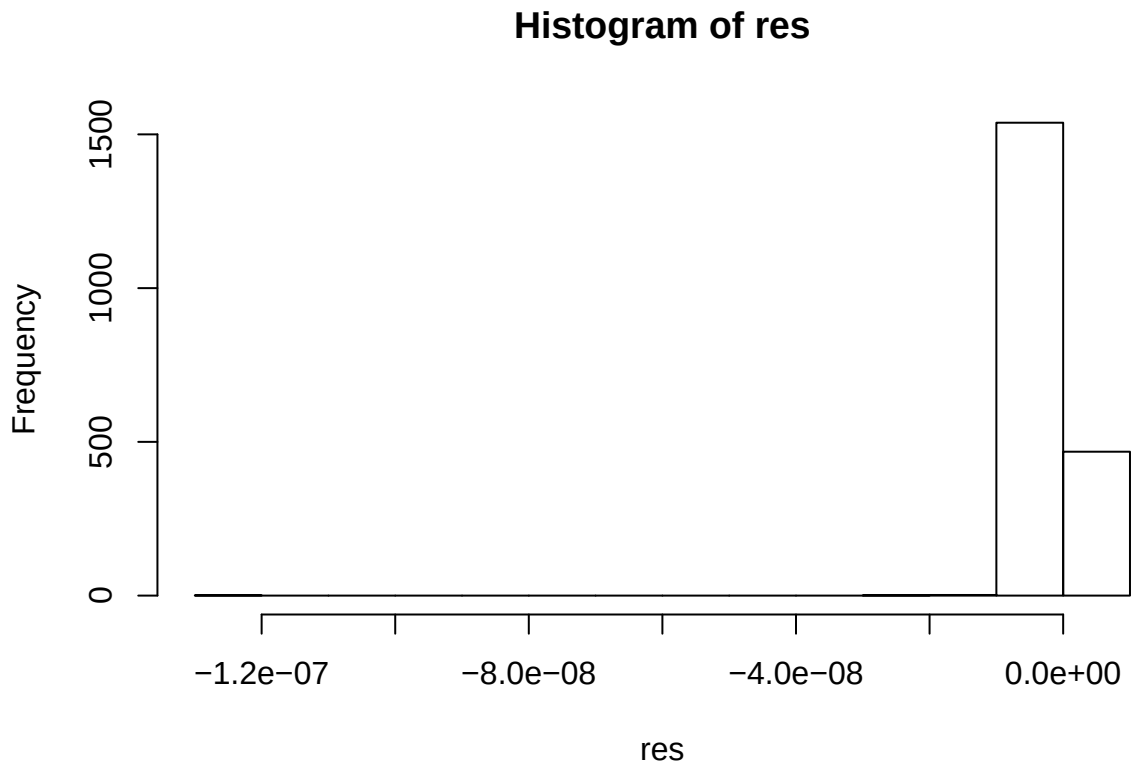
```
all(sapply(-10:10, function(mu){
  sapply(1:10, function(b){
    abs(integrate(dgumbel, -Inf, Inf, mu = mu, b = b )$value - 1) < 1e-7
  })
}) )
```

```
## [1] TRUE
```

Then we can check that `pgumbel` is the equal to the integral of `dgumbel`.

```
res <- (sapply(-100:100, function(mu){
  sapply(1:10, function(b){
    interval <- sort(rnorm(2, mu, 10 * b))
```

```
    p <- pgumbel(interval, mu = mu, b = b)
    return(integrate(dgumbel, interval[1], interval[2], mu = mu, b = b)$value - (p[2] - p[1]))
  })
}))
hist(res)
```



We can check that the `log` parameter in `dgumbel` works as required,

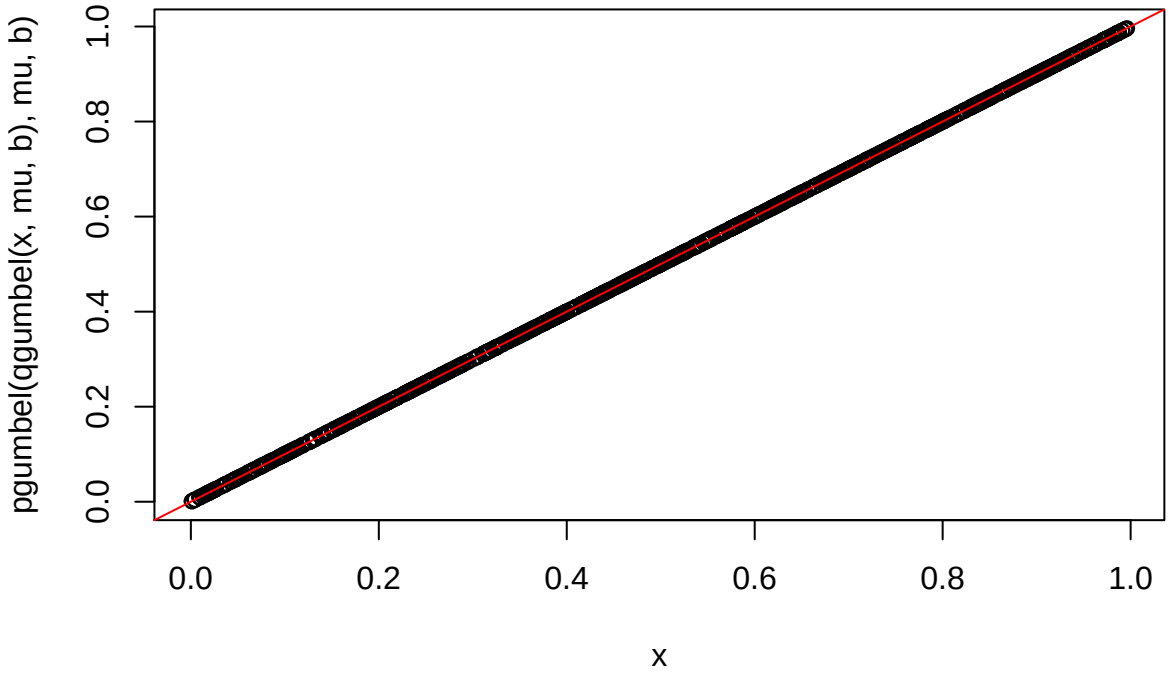
```
all(sapply(-5:5, function(mu){
  sapply(1:5, function(b){
    x <- dnorm(100, mean = mu, sd = 5 * b)
    all(abs( dgumbel(x, mu = mu, b = b, log = TRUE) - log(dgumbel(x, mu = mu, b = b, log = FALSE))) < 1e
  })
}) )
```

```
## [1] TRUE
```

We could also check the parameter `log.p` in `pgumbel` and `qgumbel`.

We check now that `qgumbel` is the inverse of `pgumbel`, we will do it for just one choice of the parameters.

```
mu <- -5
b <- 4
x <- runif(1000) ##some random probabilities
plot(x, pgumbel(qgumbel(x, mu, b), mu, b))
abline(0, 1,  col = "red") ##the points should be in this line
```




```
all(pgumbel(qgumbel(x, mu, b), mu, b) - x < 1e-12)
```

```
## [1] TRUE
```

```
##the inverse
x <- rnorm(1000, mean = mu, sd = 10)
all(qgumbel(pgumbel(x, mu, b), mu, b) - x < 1e-12)
```

```
## [1] TRUE
```

We have thus checked that the functions work as expected and we can use them.

Q3.2

Doing some easy algebraic operations we can see that the method of moments estimators for μ and β are:

$$\hat{mu} = \overline{X} - \hat{\beta}\gamma$$

$$\hat{\beta} = s\frac{\sqrt{6}}{\pi}$$

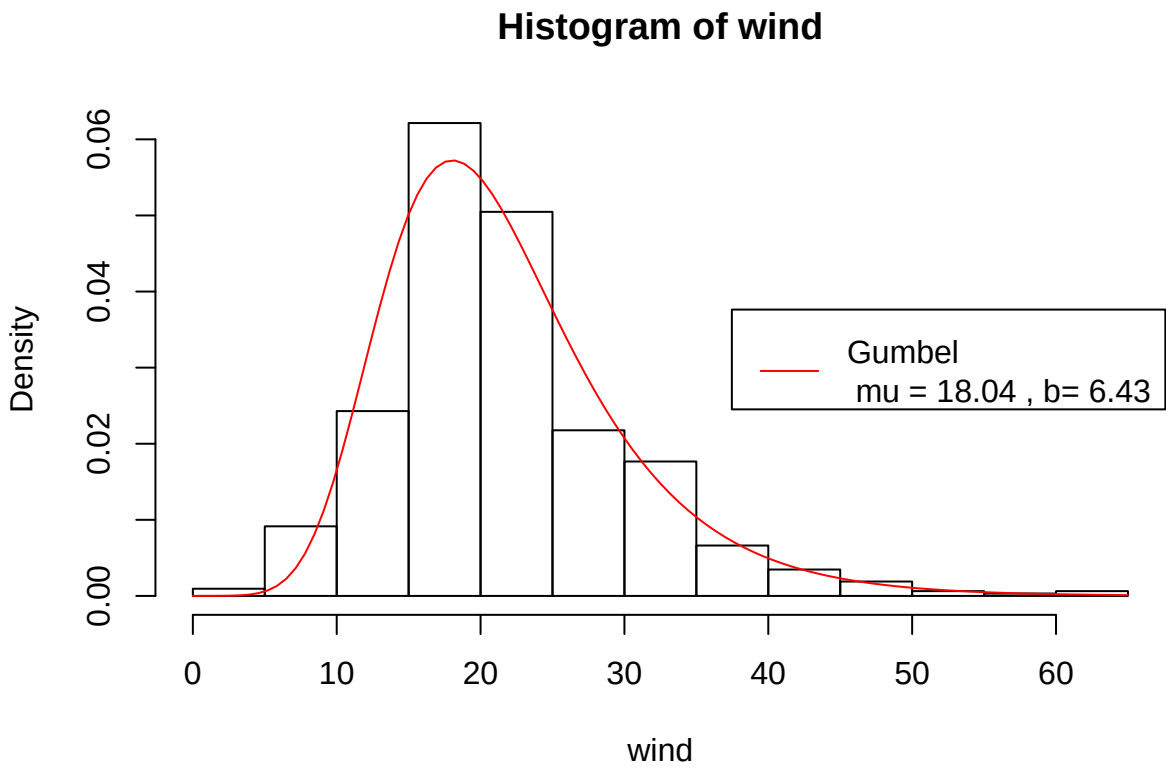
We thus estimate the parameters over the `wind` dataset:

```
b_m <- sqrt(var(wind) * 6) / pi
mu_m <- mean(wind) - (-digamma(1) ) * b_m
c(mu = mu_m, b = b_m)
```

```
##        mu         b
## 18.043939  6.426868
```

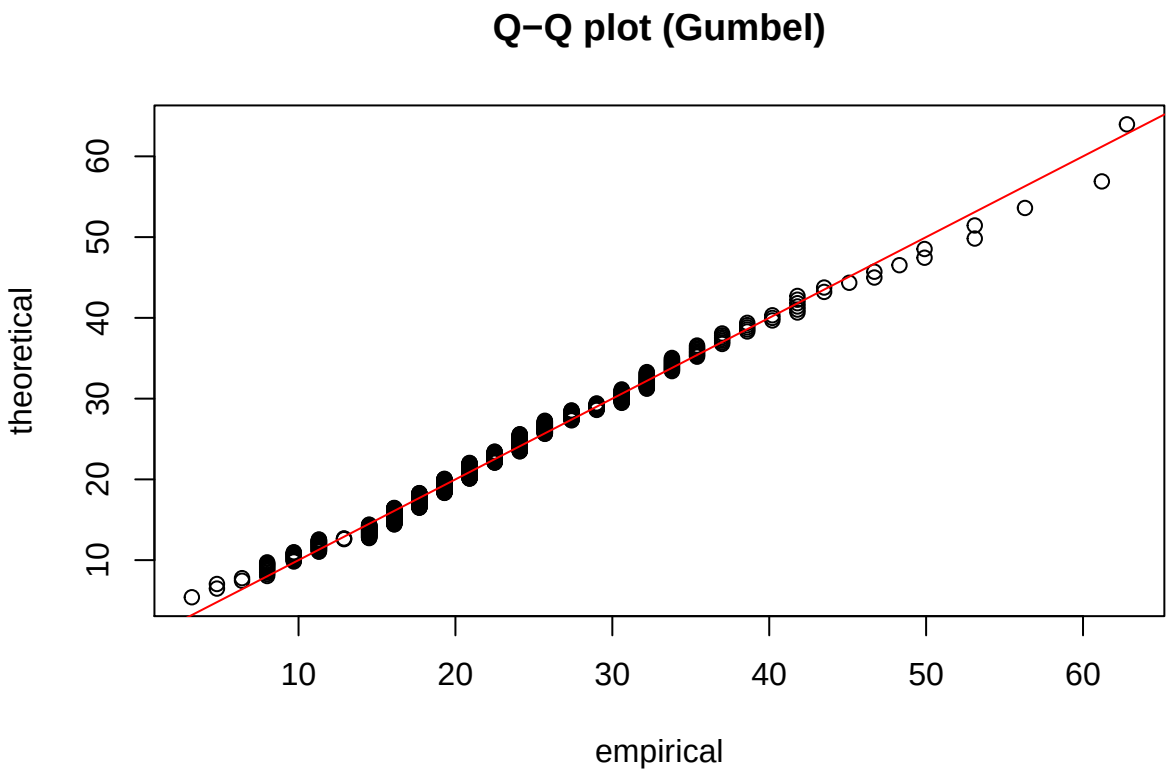
Plotting:

```
hist(wind, probability = TRUE)
curve(dgumbel(x, mu = mu_m, b = b_m), add = TRUE, col = "red")
legend("right",
       legend = paste("Gumbel \n mu =",
                      round(mu_m, digits = 2),
                      ", b=", round(b_m, digits = 2)),
       col = "red", lty = 1)
```



We will plot now a Q-Q plot.

```
p <- ppoints(wind)
th <- qgumbel(p, mu = mu_m, b = b_m)
plot(sort(wind), th, xlab = "empirical", ylab = "theoretical", main = "Q-Q plot (Gumbel)")
abline(0, 1, col = "red")
```



The Gumbel distribution seems to fit well the data.

Q3.3

We obtain now MLE estimators for the Gumbel distribution.

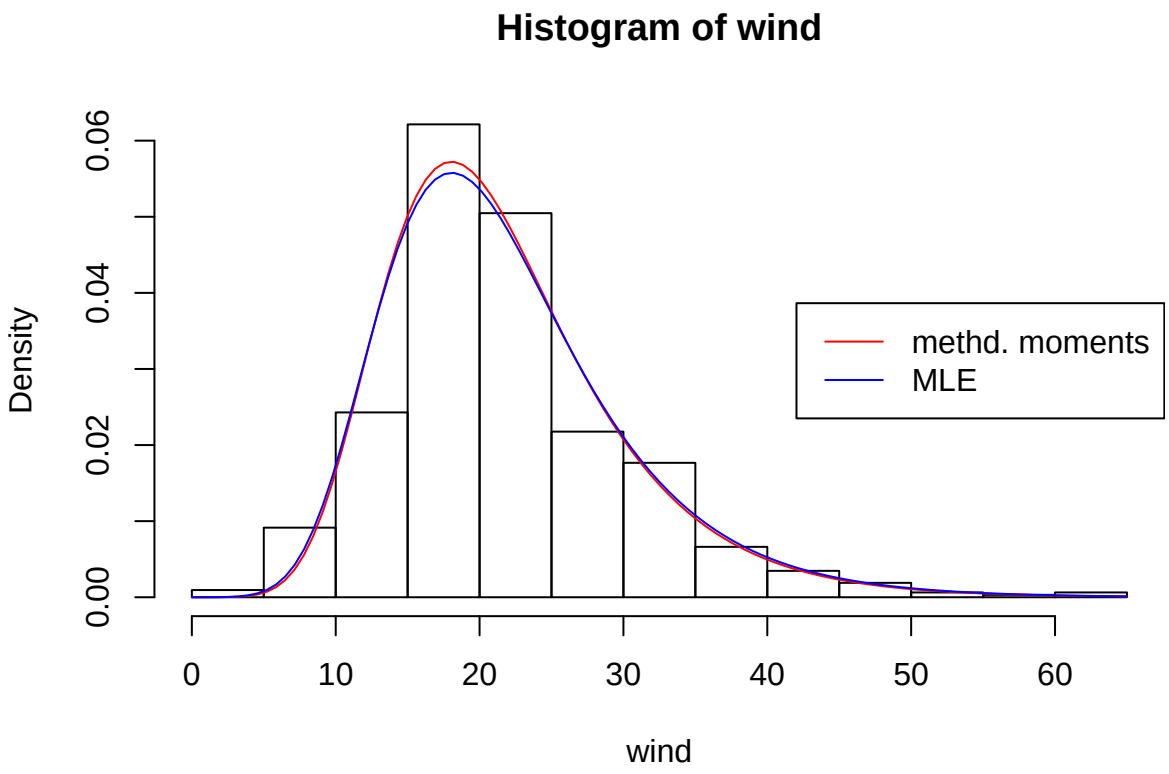
```
mll <- function(par, data){
  sum(-dgumbel(data, mu = par[1], b = par[2], log = TRUE))
}
optim_res <- optim(par = c(mu_m, b_m), fn = mll, data = wind)
par_mle <- optim_res$par
par_mle
```

```
## [1] 18.054794  6.594547
```

The obtained parameters are very similar to the method of moments (we also have a large sample so it is to be expected).

We plot the obtained Gumbel density.

```
hist(wind, probability = TRUE)
curve(dgumbel(x, mu = mu_m, b = b_m), add = TRUE, col = "red")
curve(dgumbel(x, mu = par_mle[1], b = par_mle[2]), add = TRUE, col = "blue")
legend("right",
       legend = c("methd. moments", "MLE"),
       col = c("red", "blue"), lty = 1)
```

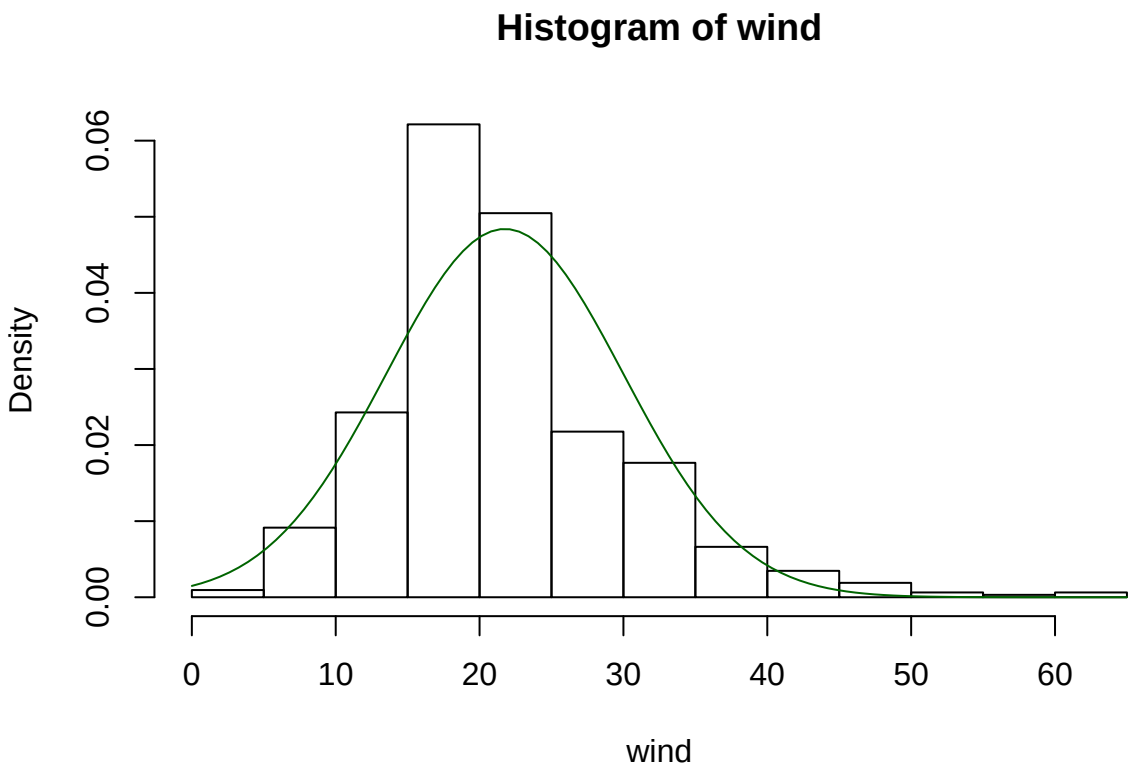


As we expected the two densities are very closed, but they are not the same.

Q3.4

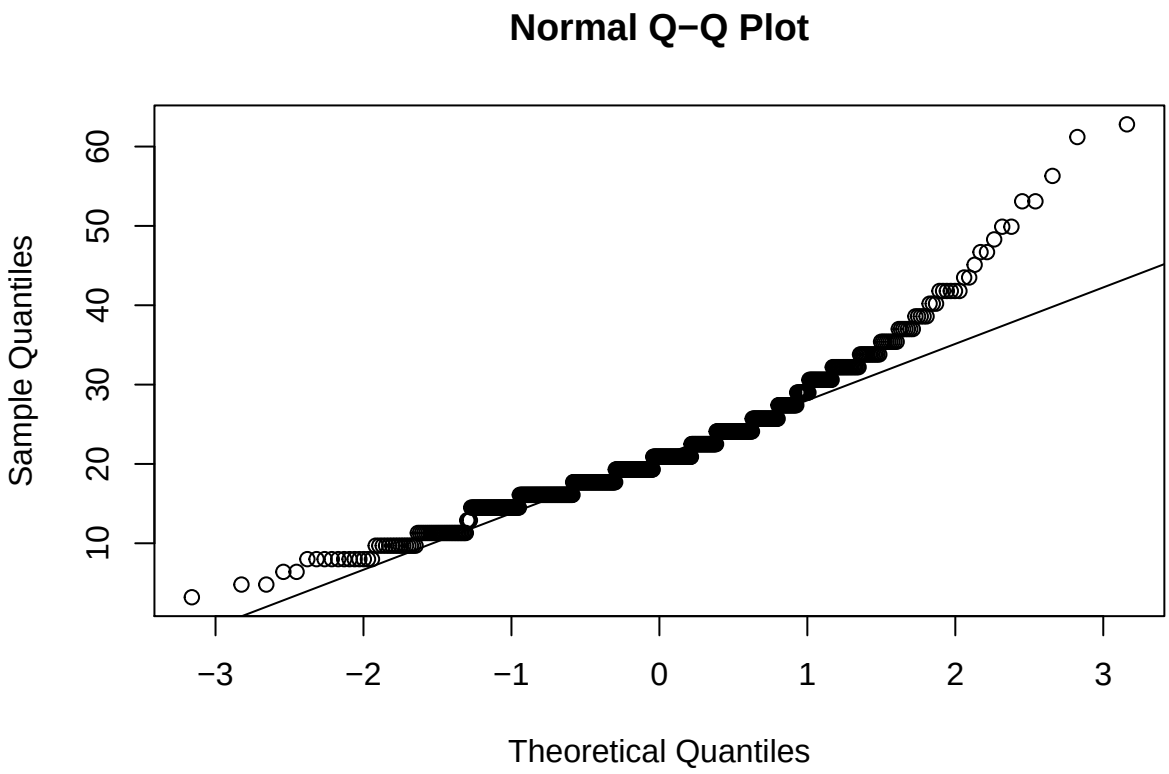
To fit a gaussian model we should remember that the MLE estimator of the mean and the sd parameter are well-known.

```
mean_w <- mean(wind)
sd_w <- sd(wind) ## we avoid the (n-1)/n correction since we have a large sample
hist(wind, probability = TRUE)
curve(dnorm(x, mean_w, sd_w), add = TRUE, col = "darkgreen")
```



It does not seem very bad, but..

```
qqnorm(wind)
qqline(wind)
```



The normal Q-Q plot shows that the model is worst than the Gumbel.

We can also use AIC and BIC (actually we could use directly the log-likelihood here since both models have the same number of parameters, 2), but to answer the question:

```
mll_gumbel <-  optim_res$value
mll_gauss <-  -sum(dnorm(wind, mean_w, sd_w, log = TRUE))
aic_gumbel <- 2*mll_gumbel + 2 * 2
aic_gauss <- 2*mll_gauss + 2 * 2
bic_gumbel <- 2*mll_gumbel + 2 * log(length(wind))
bic_gauss <- 2*mll_gauss + 2 * log(length(wind))
data.frame(row.names = c("gumbel", "gauss"),
           mll = c(mll_gumbel, mll_gauss),
           aic = c(aic_gumbel, aic_gauss),
           bic = c(bic_gumbel, bic_gauss))
```

```
##             mll      aic      bic
## gumbel 2185.525 4375.050 4383.955
## gauss  2236.427 4476.854 4485.758
```

The Gumbel model obtain the lowest score always.

Gumbel and Gaussian model are not nested, thus we can not use the likelihood ratio test.

Q3.5

We compute now confidence intervals for the parameters of the Gumbel distribution.

First we compute the standard errors:

```
M <- 1000
pars_bt <- replicate(M, {
  optim(par = c(mu_m, b_m), fn = mll, data = sample(wind, replace =TRUE))$par
})
SE <- apply(pars_bt, MARGIN = 1, sd)
SE
```

```
## [1] 0.2755853 0.2224195
```

Now we can obtain CI with normal quantiles:

```
alpha <- 0.05
z <- qnorm(alpha / 2, lower.tail = FALSE)
cbind("2.5%" = par_mle - SE * z, "97.5%" = par_mle + SE * z)
```

```
##           2.5%     97.5%
## [1,] 17.514657 18.594931
## [2,]  6.158613  7.030481
```

And percentile CI

```
t(apply(pars_bt, MARGIN = 1, quantile,  probs = c(alpha/2, 1- alpha/2)))
```

```
##           2.5%     97.5%
## [1,] 17.520110 18.613533
## [2,]  6.156683  6.999061
```

Q3.6

We perform here some Bayesian inference over the parameters of the Gumbel dsitribution.

To obtain the MAP estimators, we first of all define the (unnormalized) minus log-posterior:

```
mlpost <- function(par, data){
  mll(par, data) - dnorm(par[1], 0, sqrt(10), log = TRUE)  - dexp(par[2], rate = 1, log = TRUE)
}
```

Then we use `optim`:

```
par_map <- optim(c(mu_m, b_m), fn = mlpost, data = wind)$par
par_map
```

```
## [1] 17.902544  6.526501
```

They are very similar to the MLE parameters (we have a lot of data and the prior is not very strong).

For the posterior mean we can use the Monte-Carlo method

```
N <- 100000
mus <- rnorm(N, mean = 0, sd = sqrt(10))
bs <- rexp(N, rate = 1 )

lw <- sapply(1:N, function(i){
  sum(dgumbel(wind, mu = mus[i], b = bs[i], log = TRUE))
})

b <- max(lw)
w <- exp(lw - b) / sum(exp(lw - b))
```

```r
mu_pm <- sum(mus * w)
b_pm <- sum(bs * w)

c(mu_pm, b_pm)
```

```
## [1] 10.282771  6.159092
```