Statistical methods

Laboratory 2

N. Torelli, G. Di Credico, V. Gioia

vincenzo.gioia@units.it

Office hour: Friday, 17.00 - 18.30

23/10/2023

Contents

Let $\mathcal{F} = \{p(y;\theta), \theta \in \Theta\}$ a parametric statistical model, with Θ the parameter space and $p(y;\theta)$ a collection of density (or probability) functions. Assume that $\mathcal{F}$ is correctly specified, thus the true parameter value $\theta^0 \in \Theta$. Some problems of the statistical inference on the parameter θ rely on the following questions:

(1) What we can say, having observed y and assumed $\mathcal{F}$, [redacted] where θ^0 is allocated in Θ? This is an estimation problem: if we want obtain from y a specific value of θ^0 we are performing point estimation (previous lab), while if we want target a subset of Θ where is likely to be included θ^0, then we are involved in a interval estimation procedure.

(2) Are the data reasonably agreeing with the hypothesis that θ^0 belong to a subset Θ^0 of Θ? In such a case, we are involved in a hypothesis testing problem.

Note: the true parameter value θ^0 is unknown, and carrying out inference on θ (estimation, hypothesis testing) should be intended on θ^0

Inference: particulars to general, draws conclusions based on evidence

Deduction: general to particular, draws conclusions based on premises or established facts.

Interval estimation

The point estimate is in most cases a rude estimate. Rather, we may construct an interval for our parameter.

Recall that the confidence interval $\hat{\Theta}(y)$ (or region in the case of dimension of the parameter greater than 1) for θ with confidence level $1-\alpha$ is a subset of the parameter space Θ, defined as function of the data y and such that

$$Pr_\theta(\theta \in \hat{\Theta}(Y)) = 1 - \alpha$$

confidence level : $1-\alpha$

significance level : α

Probability that the true value of θ lie in $\hat{\Theta}(y)$ is $1-\alpha$

Note:

- The l.h.s. of the previous formula is called (null) coverage probability;
- A confidence interval can be obtained from the acceptance regions of a tests with level α for $H_0 : \theta = \theta_0$ (we will see later the connection between hypothesis testing and confidence intervals).

Interval estimation is based on **pivotal quantities** (functions of the data and the parameter whose distribution is known).

Interval estimation for μ under the Gaussian case, when σ^2 is known

Let $x = (x_1, \ldots, x_n)$ be the realizations of the r.v. $X = (X_1, \ldots, X_n)$ where the X_i, $i = 1, \ldots, n$, are independent and identically distributed according to $\mathcal{N}(\mu, \sigma^2)$. To obtain a confidence interval for the mean μ when σ^2 is known, we leverage the pivotal quantity:

$$T(\mu) = \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} \sim \mathcal{N}(0,1), \quad \forall \mu \in \mathbb{R}, \sigma^2 > 0.$$

$$-z_{1-\frac{\alpha}{2}} \leq \frac{\bar{x}-\mu}{\sigma/\sqrt{n}} \leq z_{1-\frac{\alpha}{2}}$$

$$-z_{1-\frac{\alpha}{2}} \leq \frac{\mu-\bar{x}}{\sigma/\sqrt{n}} \leq z_{1-\frac{\alpha}{2}}$$

$$\bar{x} - \frac{\sigma}{\sqrt{n}} z_{1-\frac{\alpha}{2}} \leq \mu \leq \bar{x} + z_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}$$

Then, it follows that for $0 < \alpha < 1$

$$\Pr(-z_{1-\alpha/2} \leq T(\mu) \leq z_{1-\alpha/2}) = 1 - \alpha \quad (*)$$

Our goal here is to compute $z_{1-\frac{\alpha}{2}}$ for which (*) holds.

and a confidence interval of level $1-\alpha$ for μ is given by

$$\mathrm{CI}_\mu^{(1-\alpha)} = \left(\bar{x} - z_{1-\alpha/2}\frac{\sigma}{\sqrt{n}}, \bar{x} + z_{1-\alpha/2}\frac{\sigma}{\sqrt{n}} \right)$$

where $-z_{1-\alpha/2} = z_{\alpha/2}$ and $z_{1-\alpha/2}$ represent the quantiles of order $\alpha/2$ and $1-\alpha/2$ of a standard normal distribution, respectively.

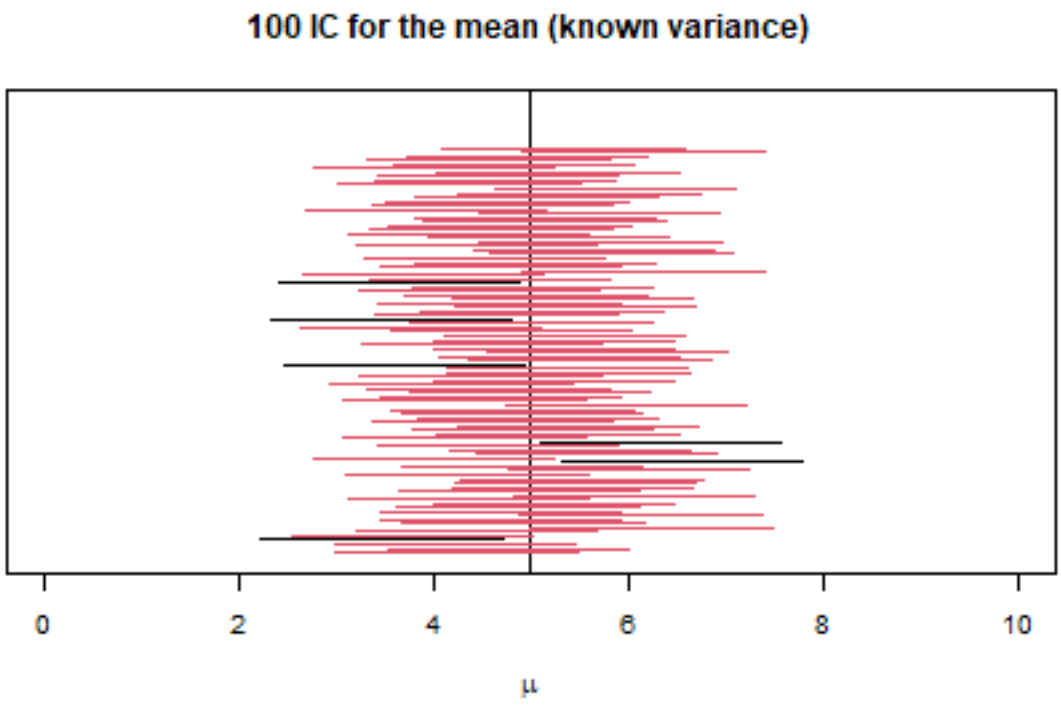
Also for interval estimation, Monte Carlo simulation is well suited to explore the procedure:

```
B <- 1000      # Number of replications
n <- 10        # sample size
mu <- 5        # True population mean
sigma <- 2     # True population standard deviation

alpha <- 0.05 # Confidence level: 1- alpha
```

```r
# CI is matrix where we save the confidence intervals for each replication:
# -) first column: lower bound
# -) second column: upper bound
CI <- matrix(0, B, 2)

# l is a vector whose elements assume TRUE (1) or FALSE(0) depending on
# whether the true parameter value lies within the interval
l <- rep(0, B)

set.seed(1234)
q0975 <- qnorm(1 - alpha/2) # quantile of order 1-alpha/2 of N(0,1)
for(i in 1 : B) {
 x <- rnorm(n, mu, sigma)
 CI[i,] <- mean(x) + c(-q0975, q0975) * sigma/sqrt(n)
 #Equivalently
 #CI[i, 1] <- mean(x) - qnorm(1-alpha/2) * sigma/sqrt(n)
 #CI[i, 2] <- mean(x) + qnorm(1-alpha/2) * sigma/sqrt(n)
 l[i] <- (mu > CI[i,1] & mu < CI[i,2])
}

#Empirical coverage probability: to do

#Plot the first 100 c.i.:
# black: intervals not including mu
# red: intervals including  mu
plot(1, xlim = c(0, 10), ylim = c(0, 11), type = "n",
     xlab = expression(mu), ylab = "", yaxt  = "n",
     main = paste("100 IC for the mean (known variance)"), cex.main = 1.2)
abline(v = mu)

d <- 0
for(i in 1 : 100){
  d <- d + 0.1
  lines(seq(CI[i, 1], CI[i, 2], length = 100), rep(d, 100), col = (l[i] + 1))
}
```

sum (l)

B




```
# number of intervals (out the 100) including the true parameter value
sum(l[1 : 100])
```

```
## [1] 94
```

Extra: R code on the interval estimation for μ under the Gaussian case, when σ^2 is unknown

Let $x = (x_1, \ldots, x_n)$ be the realization of the r.v. $X = (X_1, \ldots, X_n)$ where the X_i, $i = 1, \ldots, n$, are independent and identically distributed according to $\mathcal{N}(\mu, \sigma^2)$. To obtain a confidence interval for the mean μ when σ^2 is unknown, and it is estimated by the sample variance s^2, we leverage the pivotal quantity:

$$T(\mu) = \frac{\bar{X} - \mu}{s/\sqrt{n}} \sim t_{n-1}, \quad \forall \mu \in \mathbb{R}, \sigma^2 > 0$$

Then, it follows that for $0 < \alpha < 1$

$$\Pr(\mathrm{t}_{\mathrm{n}-1;\alpha/2} \leq \mathrm{T}(\mu) \leq \mathrm{t}_{\mathrm{n}-1;1-\alpha/2}) = 1 - \alpha$$

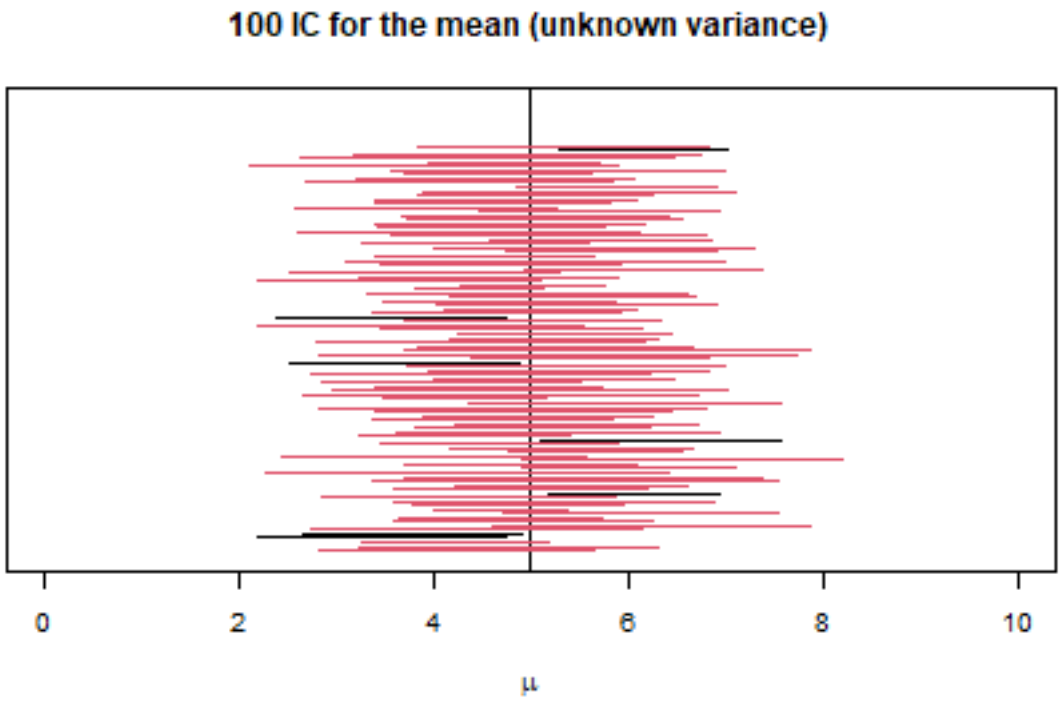
and a confidence interval of level $1 - \alpha$ for μ is given by

$$\mathrm{CI}_{\mu}^{1-\alpha} = \left(\bar{x} - t_{n-1;1-\alpha/2} \frac{s}{\sqrt{n}}, \bar{x} + t_{n-1;1-\alpha/2} \frac{s}{\sqrt{n}} \right)$$

where $-t_{n-1;1-\alpha/2} = t_{n-1;\alpha/2}$ and $t_{n-1;1-\alpha/2}$ represent the quantiles of order $\alpha/2$ and $1 - \alpha/2$ of a t-student distribution with $n - 1$ degrees of freedom, respectively.

```r
# consider the same setting as above:
# mu = 5, sigma=2, n=10, B=1000, alpha=0.05
CI <- matrix(0, B, 2)
l <- rep(0, B)

set.seed(1234)
q0975 <- qt(1 - alpha/2, n - 1) # quantile of order 1-alpha/2  of t(n-1)

# -) Generate samples from the N(mu, sigma^2)
# -) Obtain the confidence intarval for each replication
# -) Use a flag to detect if the confidence interval
#    includes the true parameter value
for (i in 1 : B){
 x <- rnorm(n, mu, sigma)
 CI[i, ] <- mean(x) + c(-q0975, q0975) * sd(x)/sqrt(n)
 l[i] <- (mu > CI[i,1] & mu < CI[i,2])
}

#Empirical coverage probability: to do
```

```r
# Plot of the (first 100) CIs
plot(1, xlim = c(0,10), ylim = c(0,11), type = "n",
     xlab = expression(mu), ylab = "", yaxt  = "n",
     main = paste("100 IC for the mean (unknown variance)"), cex.main = 1.2)
abline(v = mu)

d <- 0
for (i in 1 : 100){
  d <- d + 0.1
  lines(seq(CI[i, 1], CI[i, 2], length = 100), rep(d, 100), col = (l[i] + 1))
}
```




```r
# number of intervals (out the 100) including the true parameter value
sum(l[1 : 100])
```

```
## [1] 93
```

Extra: R code integrating the paired t-test example

The dataset **pair65** in the **DAAG** package contains the data on eighteen elastic bands, that are divided into nine pairs, with bands of similar stretchiness placed in the same pair. One member of each pair was placed in hot water for four minutes, while the others was left at ambient temperature. After ten minutes, stretch lengths (in mm) for the heated and unheated bands is recorded, and so the differences between the pairs can be obtained:

```
library(DAAG)
pair_data_frame <- cbind(pair65, pair65[,1] - pair65[,2])
pair_data_frame <- cbind(c(1:9), pair_data_frame)
dimnames(pair_data_frame) <- list(1 : 9, c("pair","heated",
                                           "ambient", "difference"))
pair_data_frame
```

```
##   pair heated ambient difference
## 1    1    244     225         19
## 2    2    255     247          8
## 3    3    253     249          4
## 4    4    254     253          1
## 5    5    251     245          6
## 6    6    269     259         10
## 7    7    248     242          6
## 8    8    252     255         -3
## 9    9    292     286          6
```

We have two paired samples, heated and ambient, of size n_1 and n_2, with $n_1 = n_2 = n = 9$. We assume that they are realizations of normally distributed random variablees. We are interested in obtaining a confidence interval for the mean difference, that is for $\mu = \mu_H - \mu_A$.

The pivotal quantity is then:

$$T(\mu) = \frac{\overline{D} - \mu}{s/\sqrt{n}} \sim t_8$$

where $\overline{D}$ denotes the sample mean difference. In our data, the observed sample mean difference is

$$\overline{d} = \overline{x}_H - \overline{x}_A = 6.33,$$

with $s = 6.10$, and $s/\sqrt{n} = 2.03$.

Once recognized the distribution of the pivotal quantity, we are able to compute the confidence intervals at 95% or 99% level. So, denoting with $1 - \alpha$ the confidence level we obtain

$$\text{CI}_{\mu}^{1-\alpha} = \left(\bar{d} - t_{n-1;1-\alpha/2} \frac{s}{\sqrt{n}}, \bar{d} + t_{n-1;1-\alpha/2} \frac{s}{\sqrt{n}} \right)$$

```r
n <- nrow(pair65) #sample size
# Quantiles of order  1-alpha/2 for a t-student(8)
alpha <- 0.05; q_0975 <- qt(1 - alpha/2, df = n - 1)
alpha <- 0.01; q_0995 <- qt(1 - alpha/2, df = n - 1)

d <- mean(pair_data_frame[,4]) #mean of the difference
d
```

```
## [1] 6.333333
```

```r
s <- sd(pair_data_frame[,4]) #sd of the difference
s
```

```
## [1] 6.103278
```

```r
# 95% confidence interval
CI_95 <- d + c(-q_0975, q_0975) * s/sqrt(n); CI_95
```

```
## [1]  1.641939 11.024728
```

```r
# 99% confidence interval
CI_99 <- d + c(-q_0995, q_0995) * s/sqrt(n); CI_99
```

```
## [1] -0.4929537 13.1596203
```

Note: There is a direct correspondence between the confidence intervals and the acceptance region of the test with level α

$$\begin{cases} H_0 : \mu_H - \mu_A = 0 \\ H_1 : \mu_H - \mu_A \neq 0 \end{cases}$$

that is the values of μ accepted by the test are those included in the interval $\bar{d} \pm t_{n-1;1-\alpha/2} \frac{s}{\sqrt{n}}$, which is the confidence interval for μ with confidence level $1 - \alpha$.

```r
t.test(pair_data_frame$heated, pair_data_frame$ambient, paired = TRUE)
```

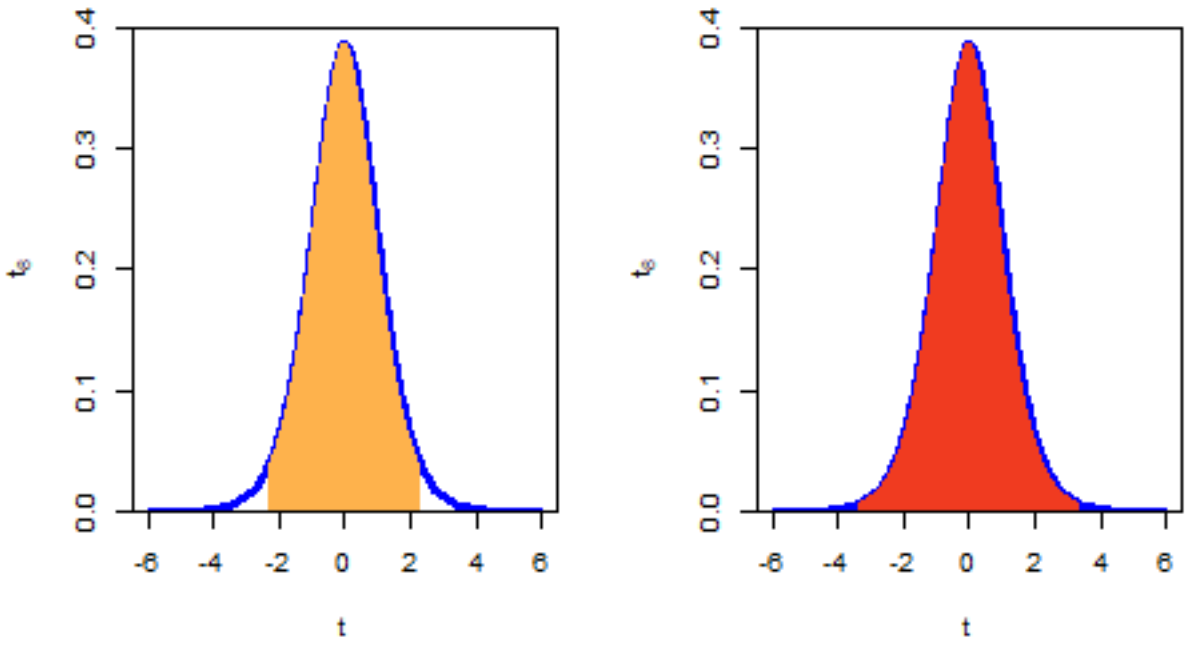
```
##
##  Paired t-test
##
## data:  pair_data_frame$heated and pair_data_frame$ambient
## t = 3.1131, df = 8, p-value = 0.01438
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##   1.641939 11.024728
## sample estimates:
## mean difference
##        6.333333
```

During the theory lecture, you have seen this result via the one-sample t-test

```r
t.test(pair_data_frame$difference)
```

The following plots report the probability density functions of the pivotal quantity with endpoints that encloses 95% of the probability ($\alpha = 0.05$) and 99% of the probability ($\alpha = 0.01$), respectively.

```
library(RColorBrewer)
plotclr <- brewer.pal(6, "YlOrRd")

par(mfrow = c(1, 2))
curve(dt(x, 8), xlim = c(-6, 6), ylim = c(0, 0.4),
      main = "", col = "blue", lwd = 3, xlab = "t",
      ylab = expression(t[8]),  yaxs = "i")
cord.x <- c(-q_0975, seq(-q_0975, q_0975, 0.01), q_0975)
cord.y <- c(0, dt(seq(-q_0975, q_0975, 0.01), 8), 0)
polygon(cord.x, cord.y, col = plotclr[3], border = NA )

curve(dt(x, 8), xlim = c(-6, 6), ylim = c(0, 0.4),
      main = "", col = "blue", lwd = 3, xlab = "t" ,
      ylab = expression(t[8]),  yaxs = "i")
cord.x2 <- c(-q_0995, seq(-q_0995, q_0995, 0.01), q_0995)
cord.y2 <- c(0, dt(seq(-q_0995, q_0995, 0.01), 8), 0)
polygon(cord.x2, cord.y2, col = plotclr[5], border = NA)
```



Basic concepts of hypothesis testing

The null hypothesis for the parameter θ is usually expressed as

$$H_0 : \theta = \theta_0$$

Complementary to the choice of H_0, we have to specify the alternative hypothesis H_1, specifying the values of the parameter which becomes reasonable when H_0 does not hold. Usually H_1 may be:

- $H_1 : \theta \neq \theta_0$ (**two-sided alternative**)
- $H_1 : \theta > \theta_0$ or $H_1 : \theta < \theta_0$(**one-sided alternative**)

Application: test for the mean difference

Consider the number of followers for the 15 most followed accounts on Instagram, expressed in millions, with each total rounded to the nearest million followers, as of March 31, 2018 (Source: Wikipedia). Among them there are

- 6 musicians
- 9 non musicians, referred as others

```
Mus <- c( "Selena Gomez",  "Ariana Grande", "Beyonce" ,
          "Taylor Swift",  "Justin Bieber", "Nicki Minaj")
Oth <- c("Cristiano Ronaldo", "Kim Kardashian",  "Kylie Jenner",
         "Dwayne Johnson", "Neymar", "Lionel Messi",
         "Kendall Jenner", "Kourtney Kardashian", "Kevin Hart")

Foll_M <- c(135, 118, 113, 107,  98,  86)
Foll_O <- c(123, 110, 106, 103,  91,  89,  89,  62,  58)
```

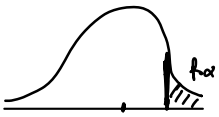
We could set up a test with the following aim: do the musicians, on average, have the same number of followers than the non-musicians? Or, do the musicians have more followers?

We suppose that $X_i \sim \mathcal{N}(\mu_m, \sigma_m^2)$, $i = 1, \dots, n_1$ and $Y_i \sim \mathcal{N}(\mu_o, \sigma_o^2)$, $i = 1, \dots, n_2$ are **independent normal samples**, where $n_1 = 6$ and $n_2 = 9$ denote the number of musicians and others, respectively. We aim to compare their means, μ_m and μ_o through the following **one-sided two-sample test** (consider $\alpha = 0.05$)

what is an F-test?

$$\begin{cases} H_0 : \mu_m - \mu_o = 0 \quad (\text{equivalently} \quad \mu_m - \mu_o \leq 0) \\ H_1 : \mu_m - \mu_o > 0 \end{cases}$$

$p = 1 - P(Z \leq z_\alpha)$



Let $\overline{X}$ and $\overline{Y}$ be the sample mean for the musicians and the others, respectively. After a preliminary F-test about the variances between the two groups (see below), we may assume that $\sigma_m^2 = \sigma_o^2$, and then the test statistic, under H_0 has the form

$$T = \frac{\overline{X} - \overline{Y}}{s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \underset{H_0}{\sim} t_{n_1+n_2-2}$$

F-test: statistical hypothesis test used to compare the variances of 2 or more groups.

$H_0: \sigma_1^2 = \sigma_2^2$
$H_1: \sigma_1^2 \neq \sigma_2^2$

var.test(,) → p = 0.6214

⇒ H_0 is not rejected

with $n_1 + n_2 - 2 = 13$. In the previous formula the pooled standard deviation (see Maindonald and Braun: pages 67 and 104) is:

$$s = \sqrt{\frac{(n_1 - 1)s_M^2 + (n_2 - 1)s_O^2}{n_1 + n_2 - 2}}$$

The results obtained by using the t.test() fucntion

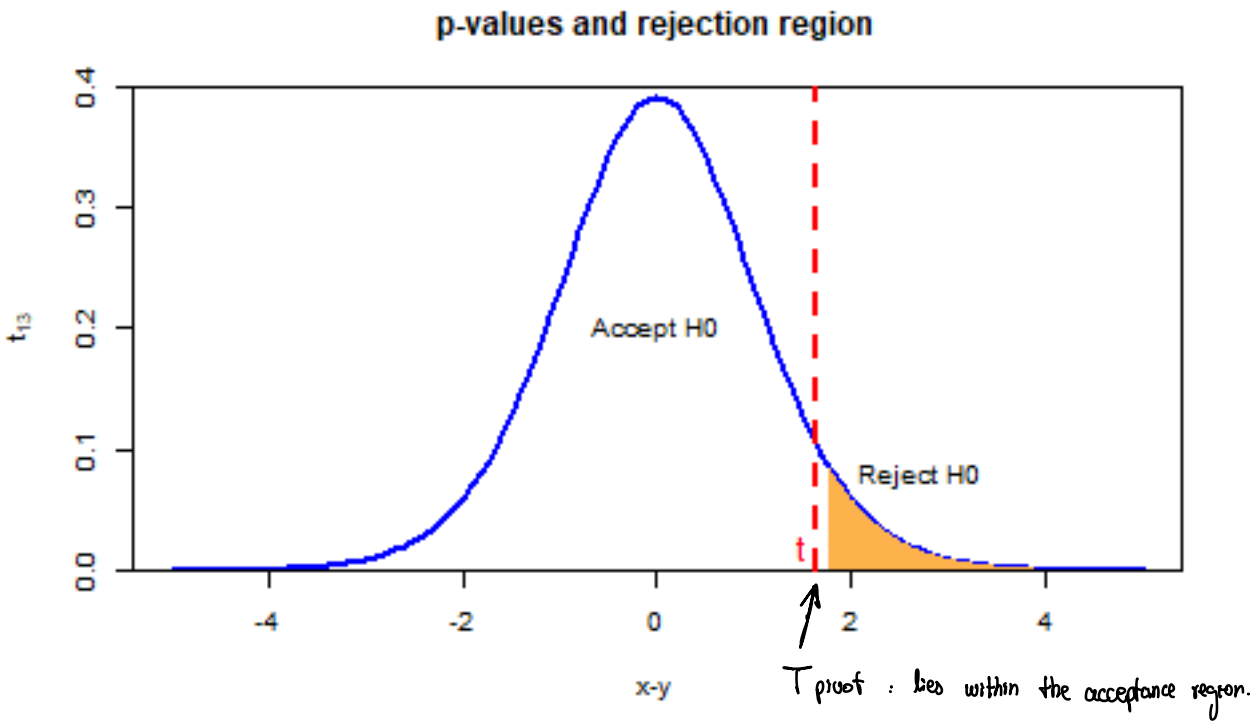
```
t <- t.test(Foll_M, Foll_O, mu = 0, alternative = "greater", var.equal = TRUE)
t
```

```
##
##  Two Sample t-test
##
## data:  Foll_M and Foll_O
## t = 1.6442, df = 13, p-value = 0.06204
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -1.322964       Inf
## sample estimates:
## mean of x mean of y
## 109.50000  92.33333
```

Thus, the p-value is 0.06204, slightly greater than α, when $\alpha = 0.05$. There is no enough/strong evidence to reject H_0.

It's your turn Carry out the results of the test by hand

```r
curve(dt(x, 13), xlim = c(-5, 5), ylim = c(0, 0.4),
      main = "p-values and rejection region", col = "blue",
      lwd = 2, xlab = "x-y",  ylab = expression(t[13]),  yaxs="i")
cord.x <- c(qt(0.95, 13),seq(qt(0.95, 13), 5, 0.01), 5)
cord.y <- c(0, dt(seq(qt(0.95, 13), 5, 0.01), 13), 0)
polygon(cord.x, cord.y, col = plotclr[3], border = NA )


abline(v = t$statistic, lty = 2, lwd = 2, col = "red")
text(0, 0.2, paste("Accept", expression(H0)))
text(2.7, 0.08, paste("Reject", expression(H0)))
text(as.double(t$statistic) - 0.15, 0.02, "t", col = "red", cex = 1.2)
```



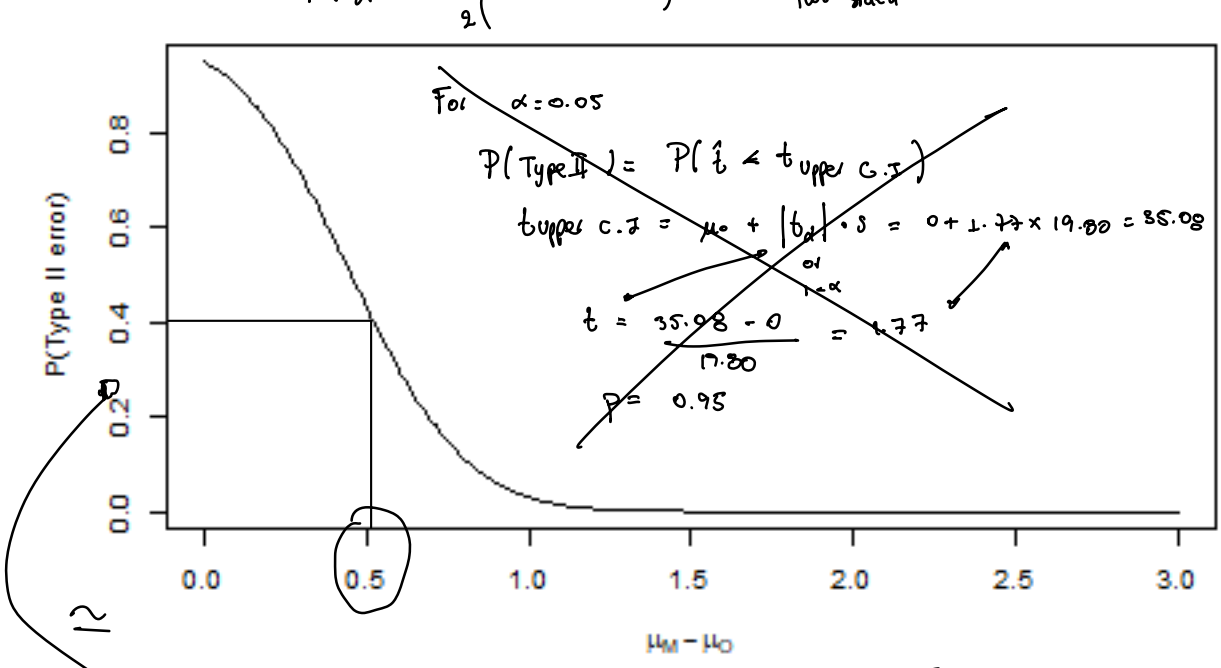
Extra exercise

Consider to include in our sample other 6 musicians and 9 musicians. However both the means and variances remain the same. What's now the conclusion? (I would suggest reading https://www.tandfonline.com/doi/full/10.1080/00031305.2016.1154108, there is the pdf in the Lab2 folder)

Type II error

Further, the following plot represent the probability of type II error (1 - power of the test), that is the probability of not rejecting H_0 when H_0 is false, by varying the difference $\mu_M - \mu_0$

```
# Mean difference grid
dgrid <- seq(0, 3, by=0.01)
# Significance level
alpha <- 0.05
# Critical t-value
t_crit <- qt(1 - alpha, n1 + n2 - 2)
# Non-centrality parameter
ncp <- dgrid * sqrt(n1 + n2) / sqrt(1 + dgrid^2 / (n1 + n2 - 2))
# Plot of the P(Type II error)
plot(dgrid, pt(t_crit, 13, ncp), type = "l",
     xlab = expression(mu[M] - mu[0]), ylab = "P(Type II error)" )
```



Test of the equality of the variances

Let us suppose to have two normally distributed populations. The test of the equality of the variances is useful for example when we want verify the assumption of equal variances before carrying out the test for the mean when the variances are unknown, as in the previous example on Instagram followers.

For two independent samples with sample sizes n_1 and n_2, respectively, from $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ we want test:

$$\begin{cases} \mathrm{H}_0 : \sigma_1^2 = \sigma_2^2 \\ \mathrm{H}_1 : \sigma_1^2 \neq \sigma_2^2 \end{cases}$$

Under H_0 (equal variances) the test statistic

$$S_1^2 \sim \chi^2_{n_1-1} \qquad S_2^2 \sim \chi^2_{n_2-1} \qquad \text{not known}$$

$$T = \frac{S_1^2}{S_2^2} \sim F_{n_1-1, n_2-1}$$

has an F-distribution with numerator degrees of freedom $n_1 - 1$ and denominator degrees of freedom $n_2 - 1$ (denoted with F_{n_1-1,n_2-1}). S_1^2 and S_2^2 are the unbiased sample variance estimators.

Then, by computing s_1^2 and s_2^2 and so $t_0 = s_1^2/s_2^2$, we reject H_0 if $t_0 < f_{n_1-1,n_2-1,\frac{\alpha}{2}}$ or if $t_0 > f_{n_1-1,n_2-1,1-\frac{\alpha}{2}}$.

$$p - value = 2\min(P(T < t_0), P(T > t_0))$$

```r
# Test for equality of variance using the var.test function
var.test(Foll_M, Foll_O, alternative = "two.sided")
```

```
##
##  F test to compare two variances
##
## data:  Foll_M and Foll_O
## F = 0.62046, num df = 5, denom df = 8, p-value = 0.6214
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.1287983 4.1925348
## sample estimates:
## ratio of variances
##           0.620457
```

```r
# Test for equality of variance by hand
rationvar <- var(Foll_M)/var(Foll_O) # Test statistic

pv_bi <- 2 * min(pf(rationvar, n1 - 1, n2 - 1, lower = FALSE),
                 1 - pf(rationvar, n1 - 1, n2 - 1, lower = FALSE))
pv_bi
```

```
## [1] 0.6214435
```

Pearson's chi-squared test

This is a class of test applied to sets of categorical data to evaluate whether any observed difference between the sets arose by chance. It is suitable for unpaired data from large samples. Pearson's chi-squared test is used to assess three types of comparison:

- **Goodness of fit**: establishes whether an observed frequency differs from a theoretical distribution;
- **Homogeneity**: test if two or more sub-groups of a population share the same distribution of a single categorical variable;
- **Independence**: determines whether two categorical variables are associated

In all the cases the test statistic is, under H_0, distributed according to the Chi-square distribution with said degrees of freedom.

Pearson's chi-squared test: Test for independence

Question: is there a relationship (association) between two categorical variables?

We want carry out the hypothesis test

$$\begin{cases} H_0 : \text{there is no relationship between the categorical variables} \\ H_1 : \text{there is relationship between the categorical variables} \end{cases}$$

H_1 is not one-sided or two-sided: sometimes it is referred to as 'many-sided' since it allows any kind of difference. However, H_1 says that there is a relationship but does not specify any particular kind of relationship.

To test H_0, we compare the observed counts with the expected counts, that is the counts that we would expect if H_0 were true. If the observed counts are far from the ~~observed~~ expected counts, there is evidence against H_0.

Then, the data are organised in a two-way table of (observed) counts or contingency table. Thus, let X and Y be the two categorical variables, with s and t categories, respectively.

Y/X	x_1	$\cdots$	x_j	$\cdots$	x_t	Total
y_1	n_{11}	$\cdots$	n_{1j}	$\cdots$	n_{1t}	$n_{1\cdot}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
y_i	n_{i1}	$\cdots$	n_{ij}	$\cdots$	n_{it}	$n_{i\cdot}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
y_s	n_{s1}	$\cdots$	n_{sj}	$\cdots$	n_{st}	$n_{s\cdot}$
Total	$n_{\cdot 1}$	$\cdots$	$n_{\cdot j}$	$\cdots$	$n_{\cdot t}$	$n_{\cdot\cdot}$

- n_{ij}= number of couples (y_i, x_j), $i = 1, \ldots, s$, $j = 1, \ldots, t$ (absolute frequencies)
- $n_{i\cdot} = \sum_{j=1}^{t} n_{ij}, i = 1, \ldots, s$ (marginal frequencies of y_i)
- $n_{\cdot j} = \sum_{i=1}^{s} n_{ij}, k = 1, \ldots, t$ (marginal frequencies of x_j)
- $n_{\cdot\cdot} = n = \sum_{i=1}^{s} \sum_{j=1}^{t} n_{ij}$ (total sample size)

The test statistic that allows the comparison between observed and expected counts is the Pearson's chi-squared statistic, which is a measure of how far the observed counts in the two-way table are from the expected counts. It is always positive and it is zero only when the observed counts are exactly equal to the expected counts.

Then

- Large value of the statistic are evidence against H_0
- Small values of the statistic do not provide evidence against H_0

Note that even though H_1 is many-sided, the Pearson's chi-squared test is one-sided because any violation of H_0 tends to produce a large value in the statistics.

Then, under H_0 leveraging the independence

$$P(X = x_i, Y = y_j) = P(X = x_i) \times P(Y = y_j),$$

for any $i = 1, \dots, s$ and $j = 1, \dots, t$, we can obtain the table of expected frequencies, denoted as n^*_{ij}, under the H_0

Y/X	x_1	$\cdots$	x_j	$\cdots$	x_t
y_1	n^*_{11}	$\cdots$	n^*_{1j}	$\cdots$	n^*_{1t}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
y_i	n^*_{i1}	$\cdots$	n^*_{ij}	$\cdots$	n^*_{it}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
y_s	n^*_{s1}	$\cdots$	n^*_{sj}	$\cdots$	n^*_{st}

where

$$n^*_{ij} = \frac{n_{i\cdot} n_{\cdot j}}{n}$$

Then, the X^2 Pearson statistic is

Pearson statistic $\overset{\cdot}{\sim}_{H_0} \chi^2_{(s-1)(t-1)}$

$$X^2 = \sum \frac{(N_{ij} - n^*_{ij})^2}{n^*_{ij}} \overset{\cdot}{\sim}_{H_0} \chi^2_{(s-1)(t-1)}$$

Then, using the observed frequencies, n_{ij}, and the expected frequencies, n^*_{ij}, we can obtain the observed test statistics

$$t_0 = \sum \frac{(n_{ij} - n^*_{ij})^2}{n^*_{ij}}$$

By fixing the significance level α, we reject H_0 if $t_0 > \chi^2_{(s-1)(t-1);1-\alpha}$ or equivalently if

$$t_0 \in \mathcal{R}_\alpha = (\chi^2_{(H-1)(K-1);1-\alpha}, +\infty)$$

Because Pearson chi square test is one sided

Example: Happiness by family income

The following data are collected for investigating the question: "What contributes to your overall happiness?". The data from the General Social Survey can be used to investigate which variables are associated with the happiness. Here, we consider the 2012 survey data and we analyse the relationship between happiness and family income for 1733 respondents.

```
obs_GSS <- matrix(c(29,178, 135, 83, 494, 277, 104, 314, 119), 3, 3, T)
colnames(obs_GSS) <- c("Not Too Happy", "Pretty Happy", "Very Happy")
rownames(obs_GSS) <- c("Above Average", "Average", "Below Average")
obs_GSS
```

```
##               Not Too Happy Pretty Happy Very Happy
## Above Average            29          178        135
## Average                  83          494        277
## Below Average           104          314        119
```

Here, as usually we will see two ways for perforning the hypothesys testing of independence between happiness and family income

- Manually: at first we need obtaining the table of expected frequencies, then we can compute the Chi-square test statistic and obtain the p-value

```
m_happy <- apply(obs_GSS, 1, sum)
m_income <- apply(obs_GSS, 2, sum)
n <- sum(obs_GSS)
exp_GSS <- outer(m_happy, m_income)/n
exp_GSS
```

```
##               Not Too Happy Pretty Happy Very Happy
## Above Average      42.62666     194.5828   104.7905
## Average           106.44201     485.8881   261.6699
## Below Average      66.93133     305.5291   164.5395
```

```
Xi_GSS <- sum((obs_GSS - exp_GSS)^2/exp_GSS)
Xi_GSS
```

```
## [1] 54.04308
```

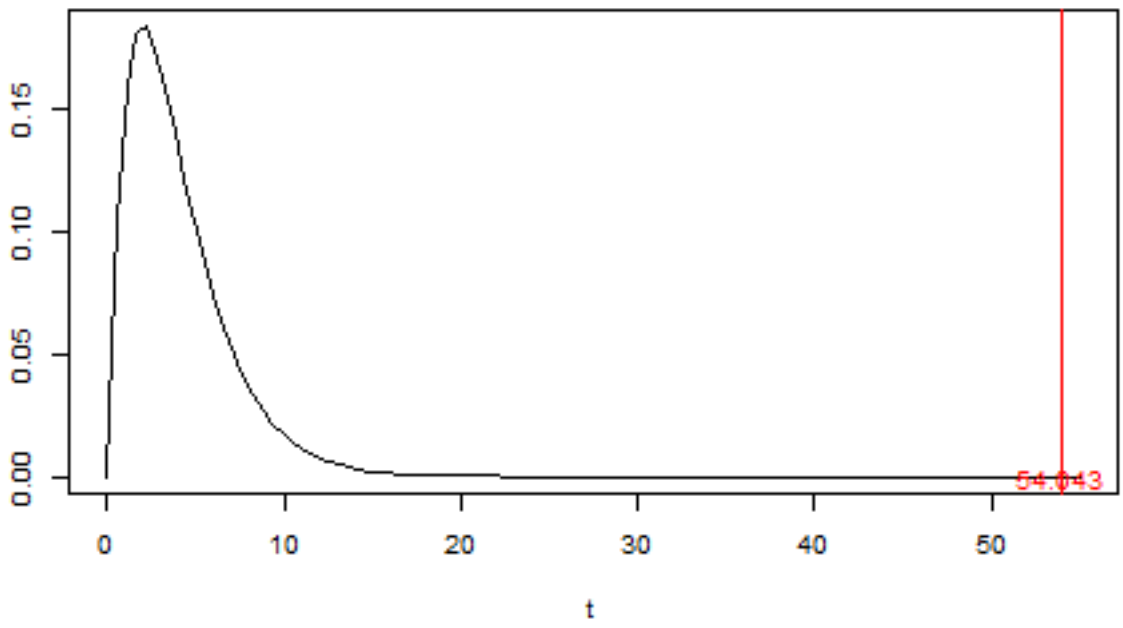
just 4 df

```
pchisq(Xi_GSS, 4, lower = FALSE)
```

```
## [1] 5.154502e-11
```

→ Reject H_0

```
curve(dchisq(x, 4), from = 0, to  = 55, xlab = "t", ylab = "")
abline(v = Xi_GSS, col="red")
text(Xi_GSS, 0, label = round(Xi_GSS,3), col="red")
```



- By using the **chisq.test**() function

```
chisq <- chisq.test(obs_GSS)
chisq
```

```
##
##  Pearson's Chi-squared test
##
## data:  obs_GSS
## X-squared = 54.043, df = 4, p-value = 5.155e-11
```

```
chisq$expected
```

```
##                Not Too Happy Pretty Happy Very Happy
## Above Average      42.62666     194.5828   104.7905
## Average           106.44201     485.8881   261.6699
## Below Average      66.93133     305.5291   164.5395
```

Extra: R code integrating Pearson's chi-squared test for the darts challenge example

Goodness of fit: Darts challenge against one friend

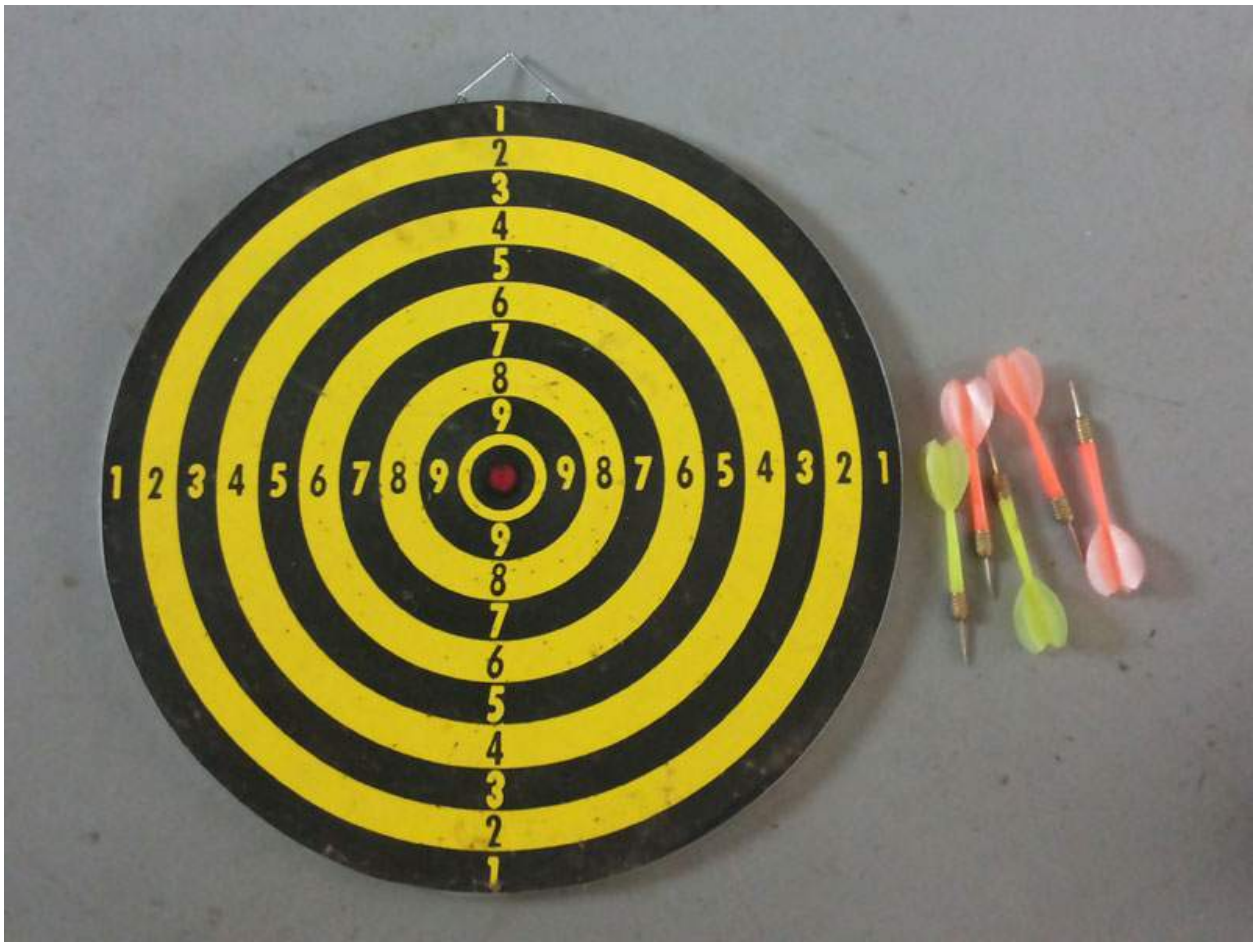
Suppose that n observations $x_1, \ldots, x_n$ are divided among K cells. The following test statistic is then defined:

$$X^2 = \sum_{k=1}^{K} \frac{(O_k - E_k)^2}{E_k} \underset{H_0}{\dot{\sim}} \chi^2_{K-1},$$

where

- O_k are the observed frequencies for cell k
- E_k are the expected frequencies for cell k, under H_0

But what is H_0 here? For illustration purposes, suppose you are playing darts agains another friend.



You suspect that your friend is not a great darts player, and that his shots along the game will hit the lowest points with great probability and the highest point with low probability. Translated in probability terms, you divide the darts target into $K = 4$ zones and you assign the following hitting probabilities:

- Zone 1 (from 1 to 3 points): $p_1 = 7/16$;
- Zone 2 (from 4 to 6 points): $p_2 = 5/16$;
- Zone 3 (from 7 to 9 points): $p_3 = 3/16$;
- Zone 4 (the highest points in the middle of the target, let's say ≥ 10 points): $p_4 = 1/16$.

Your null hypothesis is that, due to a moderate control on his darts skills, he has decreasing probabilities to hit the best zones:

$$\mathrm{H}_0 : p_1 = 7/16, \quad p_2 = 5/16, \quad p_3 = 3/16, \quad p_4 = 1/16.$$

Any significative deviation from the above probability distribution, would cause the rejection of the null hypothesis. For checking your assumption, you count the first $n = 50$ attempts $x_1, \ldots, x_n$ of your opponent, and you will code $x_i = k$, if the i-th shot hits the k-th zone.

```
# Initial settings:
n <- 50                          # number of attempts
K <- 4                           # number of zones
p <- c( 7/16, 5/16, 3/16, 1/16)  # vector of probabilities
```

Let us generate the 50 values and suppose that we are sampling from the distribution specified under H_0. Thus, the observed (absolute) frequencies are

```
set.seed(1234)
x <- sample(1 : K, n, replace = TRUE, prob = p)
obs <- table(x)
obs
```

```
## x
##  1  2  3  4
## 23 17  9  1
```

- Performing the test by hand: we need computing the expected frequencies under H_0, the observed value for the test statistic and computing p-value

```
exp <- n * p
exp
```

```
## [1] 21.875 15.625  9.375  3.125
```

```
X2 <- sum((obs - exp)^(2)/exp)
X2
```

```
## [1] 1.638857
```

```
pchisq(X2, df = K - 1, lower.tail = FALSE)
```

```
## [1] 0.6506117
```

- Performing the test by using the **chisq.test**() function: we just need passing in argument the observed frequencies and the vector of hitting probabilities you want investigate

```
chisq.test(obs, p = p)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  obs
## X-squared = 1.6389, df = 3, p-value = 0.6506
```

Then, your assumption may be accepted, your friend is hitting the target according to your hypothesized probabilities. But he wants to play again, and doing another challenge. Before starting, he requires to drink an energetic drink. You are ready to count his next 50 attempts. Does the drink improve the performance?

Thus, let us suppose to generate according to a different vector of probabilities (improving the performance of your friend):

```
p2 <- c(5/16, 6/16, 3/16, 2/16)
x_drink <- sample(1 : K, n, replace = TRUE, prob = p2)

new_obs_ad <- table(x_drink)
new_obs_ad
```

```
## x_drink
##  1  2  3  4
## 12 26  7  5
```

Then, we perform the hypothesis testing above using the new observations

```
chisq.test(new_obs_ad, p = p)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  new_obs_ad
## X-squared = 13.074, df = 3, p-value = 0.00448
```

It seems that the new shots do not follow your hypothesized distribution. While the test could not say if there is an improvement or decrease in the performance, apparently your friend improved his performance and the energetic drink was strongly required.

Homogeneity: Darts challenge with more friends

Suppose now that other five friends join you and the other guy, for a total amount of $M = 6$ friends. Do all of your friends share the same probabilities, with the above probabilities to hit the four zones? Now the test statistic is:

$$X^2 = \sum_{k=1}^{K} \sum_{m=1}^{M} \frac{(O_{k,m} - E_{k,m})^2}{E_{k,m}} \underset{H_0}{\overset{\cdot}{\sim}} \chi^2_{(K-1)(M-1)}$$

For each of them, we count the first 50 shots

```
# Initial settings:
n <- 50                             # number of attempts
K <- 4                              # number of zones
M <- 6                              # number of friends
p <- c(7/16, 5/16, 3/16, 1/16)      # vector of probabilities

x <- matrix(0, M, n)
set.seed(123)
for(m in 1 : M)   x[m, ] <- sample(1 : K, n, replace = TRUE, prob = p)

obs <- apply(x, 1, table)
obs
```

```
##   [,1] [,2] [,3] [,4] [,5] [,6]
## 1   21   23   22   21   23   25
## 2   15   14   16   21   17   11
## 3    9   12    7    7    6   11
## 4    5    1    5    1    4    3
```

```
chisq.test(obs, p = p)
```

```
##
##  Pearson's Chi-squared test
##
## data:  obs
## X-squared = 12.743, df = 15, p-value = 0.6221
```

Yes, the test is suggesting that all of your friends homogeneously hit the darts target.

What happens if a great player decides to join you? We simulate the data and perform the test again. Let us suppose simulating according to a specified vector of probabilities (characterizing the hitting probabilities of a great player). So in this case, we do not have evidence that all the players are homogeneously hitting the darts target according to the hypothisized hitting probabilities.

```
x_great <- sample(1 : K, n, replace = TRUE, prob = c(1/16, 3/16, 5/16, 7/16))
table(x_great)
```

```
## x_great
##  1  2  3  4
##  3  8 17 22
```

```
obs <- cbind(apply(x, 1, table), table(x_great))
obs
```

```
##   [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## 1   21   23   22   21   23   25    3
## 2   15   14   16   21   17   11    8
## 3    9   12    7    7    6   11   17
## 4    5    1    5    1    4    3   22
```

```
chisq.test(obs, p = p)
```

```
##
##  Pearson's Chi-squared test
##
## data:  obs
## X-squared = 88.166, df = 18, p-value = 3.077e-11
```

Note: Investigate the reasons of the warnings that appear using the **chisq.test**() procedure

Likelihood inference

Let $\mathcal{F}$ a parametric statistical model for the data y, where $f(y;\theta)$ is the density or probability function and $\theta \in \Theta$ is a p-dimensional parameter. By considering $f(y;\theta)$ as function of θ with y fixed to the observed value, then the likelihood function of θ based on y, $\mathcal{L} : \Theta \to \mathbb{R}^+$, is

$$\mathcal{L}(\theta) = \mathcal{L}(\theta; y) = c(y) f_Y(y;\theta) \ ,$$

where $c(y) > 0$ is a proportionality constant that does not depend on θ.

On the basis of the data y, $\theta \in \Theta$ is more likely than $\theta' \in \Theta$ as an index of the data generating model if $\mathcal{L}(\theta) > \mathcal{L}(\theta')$

Note:

- $\mathcal{L}(\theta)$ is not a density function (on Θ)
- It is convenient to work with $\ell(\theta) = \log \mathcal{L}(\theta)$
- If we assume independent observations for $y = (y_1, \ldots, y_n)$, then $\mathcal{L}(\theta) \propto \prod_{i=1}^n f_{Y_i}(y_i;\theta)$ and, up to an additive constant, $\ell(\theta) = \sum_{i=1}^n \log f_{Y_i}(y_i;\theta)$
- The value $\hat{\theta}$ that maximizes $\mathcal{L}(\theta)$ (or equivalently $\ell(\theta)$) is the maximum likelihood estimate

Binomial model

Let $y = (y_1, \ldots, y_n)$ a sample of i.i.d. values from a Bernoulli distribution, $Y \sim \text{Be}(p)$. Then the likelihood function is

$$\mathcal{L}(p) = \prod_{i=1}^n p^{y_i}(1-p)^{1-y_i}$$

and the log-likelihood function takes the form

$$\ell(p) = \sum_{i=1}^n y_i \log(p) + (1-y_i)\log(1-p)$$

The maximum likelihood estimate, $\hat{p}$, is the sample proportion $\hat{p} = \frac{1}{n}\sum_{i=1}^n y_i$. To derive it we obtain the score function

$$\ell_\star(p) = U(p) = \frac{\partial \ell(p)}{\partial p} = \sum_{i=1}^n \frac{y_i}{p} - \frac{1-y_i}{1-p} = \frac{\sum_{i=1}^n y_i}{p} - \frac{n - \sum_{i=1}^n y_i}{1-p}$$

By equating at zero the score function, we obtain the maximum likelihood estimate

$$U(p) = 0 \implies \hat{p} = \frac{1}{n}\sum_{i=1}^n y_i$$

Example

Suppose generating a random sample y of size $n = 100$ from Bernoulli distribution with parameter $p = 0.6$. We want make inference on p. Thus,

```
set.seed(13)
n <- 100
p <- 0.6
y <- rbinom(n, 1, p)
```

It's your turn: Write a function taking in argument the parameter and the data and returns the log-likelihood

We can do it in two ways:

- Obtaining the log-likelihood function using the **dbinom**() function
- Writing the log-likelihood function manually

We can check if both the functions return the same value. Thus, for instance by fixing $p = 0.5$

```
llik_bin(0.5,y)
```

```
## [1] -69.31472
```

```
llik_bin2(0.5,y)
```

```
## [1] -69.31472
```

The maximum likelihood estimate is in closed form (later we will see how carrying out the ML estimation when the estimate can not be obtained analytically), so

```
MLEp  <- mean(y)
MLEp
```

```
## [1] 0.63
```

```
llik_bin(MLEp, y)
```

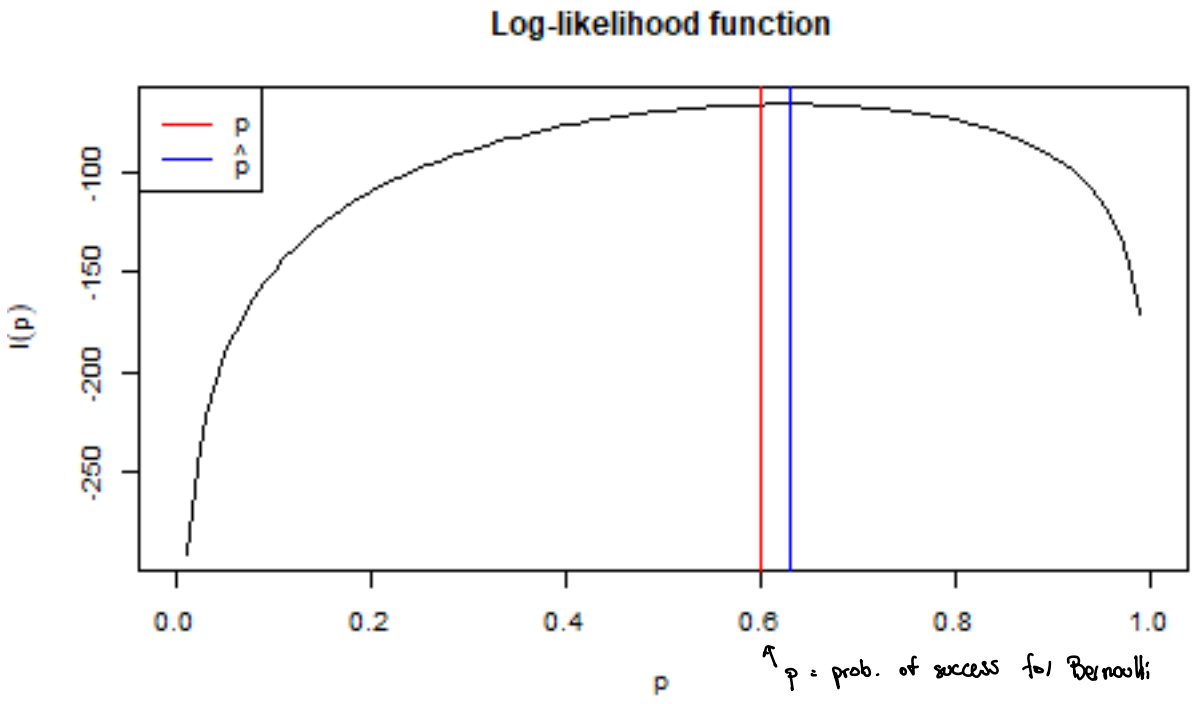
```
## [1] -65.89557
```

```
pgrid <- seq(0.01, 0.99, by = 0.01)

pgrid[which(llik_bin2(pgrid, y) == max(llik_bin2(pgrid, y)))]
```

```
## [1] 0.63
```

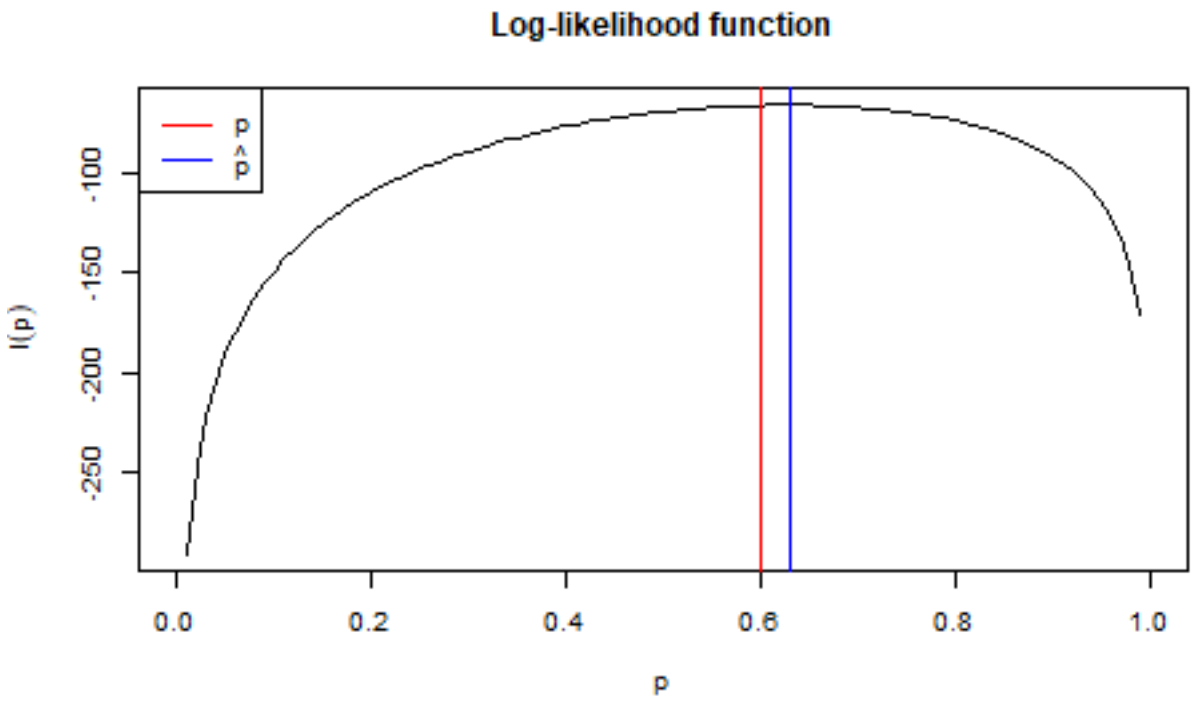
Then we can plot it, including two vertical lines denoting where the true parameter value and the maximum likelihood estimate are located

```
curve(llik_bin2(theta=x, data = y), 0, 1,
      xlab= expression(p), ylab = expression(l(p)),
      main = "Log-likelihood function")
abline(v = p, col = "red")
abline(v = MLEp, col = "blue")
legend("topleft", legend = c(expression(p),
                             expression(hat(p))),
       col = c("red", "blue"), lty = c(1, 1))
```



Note that we used the function built manually. This because that function is vectorized, while the first one should be vectorized before plotting it. Thus

```
llik_bin_v <- Vectorize(llik_bin, 'theta')

curve(llik_bin_v(theta=x, data = y), 0, 1,
      xlab= expression(p), ylab = expression(l(p)),
      main = "Log-likelihood function")
abline(v = p, col = "red")
abline(v = MLEp, col = "blue")
legend("topleft", legend = c(expression(p),
                             expression(hat(p))),
       col = c("red", "blue"), lty = c(1,1 ))
```



A note on the vectorisation

```
llik_bin(c(0.5, 0.99), y) # Wrong
```

```
## [1] -117.872
```

```
llik_bin2(c(0.5, 0.99), y) # Correct
```

```
## [1]  -69.31472 -171.02447
```

```
llik_bin_v(c(0.5, 0.99), y) # Correct
```

```
## [1]  -69.31472 -171.02447
```

We know that $\hat{\theta} \dot{\sim} \mathcal{N}(\theta, i^{-1}(\theta))$, where $i(\theta)$ is the expected information matrix, that is $i(\theta) = E(J(\theta; Y))$, with $J(\theta; Y)$ the observed information matrix, that is the negative hessian. In our one-dimensional parameter example

$$J(p) = -\frac{\partial^2 \ell(p)}{\partial p^2} = \left[\frac{\sum_{i=1}^n y_i}{p^2} + \frac{(n - \sum_{i=1}^n y_i)}{(1-p)^2}\right]$$

Thus, since $\sum_{i=1}^n Y_i \sim Bin(n, p)$ we have $\frac{n \cdot p}{p^2} + \frac{n(1-p)}{(1-p)^2} = n \frac{1}{p(1-p)}$

$$i(p) = E[J(p; Y)] = \left[\frac{\sum_{i=1}^n E[Y_i]}{p^2} + \frac{(n - \sum_{i=1}^n E[Y_i])}{(1-p)^2}\right] = \frac{n}{p(1-p)}$$

Then $\hat{p} \dot{\sim} \mathcal{N}(p, \frac{p(1-p)}{n})$, where we can replace $i^{-1}(p)$ with the estimate $i^{-1}(\hat{p}) = j^{-1}(\hat{p}) = \hat{p}(1-\hat{p})/n$

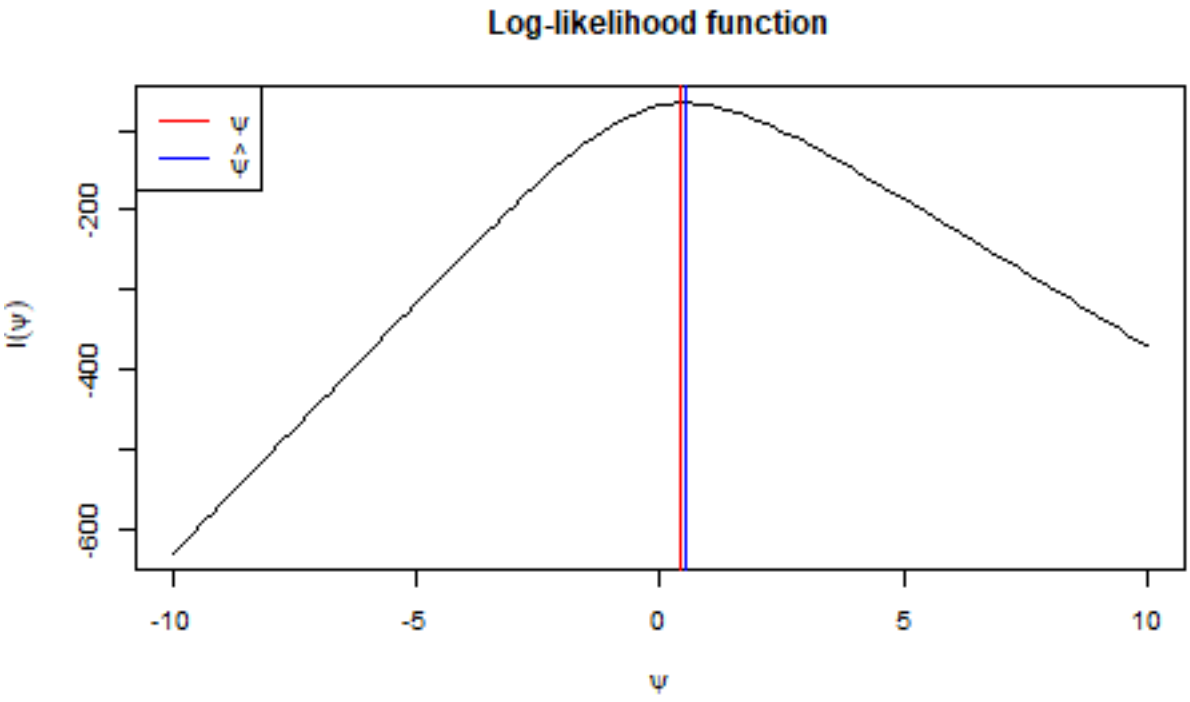
Why ?

Reparametrizations

Let us consider the logit function $\psi(p) = \text{logit}(p) = \log\left(\frac{p}{1-p}\right)$, which gives the log-odds. We can use it to reparametrise the model, so the parametric space is unbounded. At such point, the parameter is expressed in the logit scale, and we need to re-express them in the original scale, that is $p(\psi) = \exp(\psi)/(1 + \exp(\psi))$. We can leverage the property of invariance under reparametrisations of the maximum likelihood estimator and so the maximum likelihood estimate is simply $\hat{\psi} = \log\left(\frac{\hat{p}}{1-\hat{p}}\right)$. Then, we do not need to write down the log-likelihood under the reparametrisation obtain the MLE of ψ from it.

However, we can easily visualize the log-likelihood function under the reparametrisation as

```
psi <- function(theta){
  log(theta/(1-theta))
}

theta <- function(psi){
 exp(psi)/(1+exp(psi))
}

llik_bin_rep <- function(param, data) llik_bin2(theta(param), data)

curve(llik_bin_rep(param = x, data = y), -10, 10,
      xlab= expression(psi), ylab = expression(l(psi)),
      main = "Log-likelihood function")
abline(v = psi(p), col = "red")
abline(v = psi(MLEp), col = "blue")
legend("topleft", legend = c(expression(psi),
                             expression(hat(psi))),
       col = c("red", "blue"), lty = c(1, 1))
```



The asymptotic distribution of the ML estimator of ψ is given by

$$\hat{\psi} \sim \mathcal{N}(\psi, V(\hat{\psi}))$$

By using the delta method, we can easily obtain

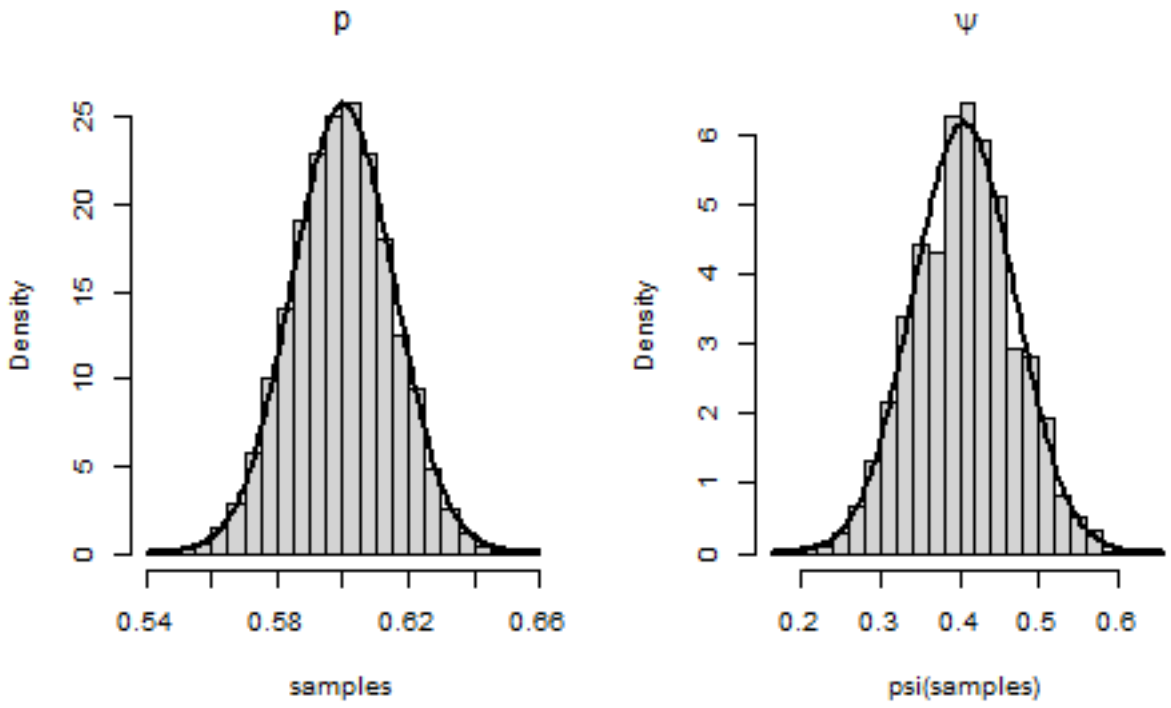
$$V(\hat{\psi}) = V(\hat{p})\left(\frac{d}{dp}\psi(p)\right)^2 = V(\hat{p})\left(\frac{d}{dp}\log\frac{p}{1-p}\right)^2 = \frac{p(1-p)}{n}\frac{1}{p^2(1-p)^2} = \frac{1}{np(1-p)}$$

and a consistent estimate of such variance is $\hat{V}(\hat{\psi}) = \frac{1}{n\hat{p}(1-\hat{p})}$

Then, we can use Monte Carlo simulation to assess the asymptotic normality of the MLE.

```
set.seed(1234)
R <- 5000
p <- 0.6
n <- 1000

samples <- rep(0, R)

for(i in 1:R){
  samples[i] <- mean(rbinom(n, 1, p))
}
```

```r
par(mfrow=c(1,2))
hist(samples, freq = F, nclass = 30, main = expression(p))
curve(dnorm(x, mean = p, sd = sqrt((p * (1 - p)/n))), → n Bernoulli
      lwd = 2, xlab = "", ylab = "", add = T)

hist(psi(samples), freq = F, nclass = 30, main = expression(psi))
curve(dnorm(x, mean = psi(p), sd = 1/sqrt((p * (1 - p) * n) )),
      lwd = 2, xlab = "", ylab = "", add = T)
```



Under transformations the distribution of a MLE parameter is invariant.

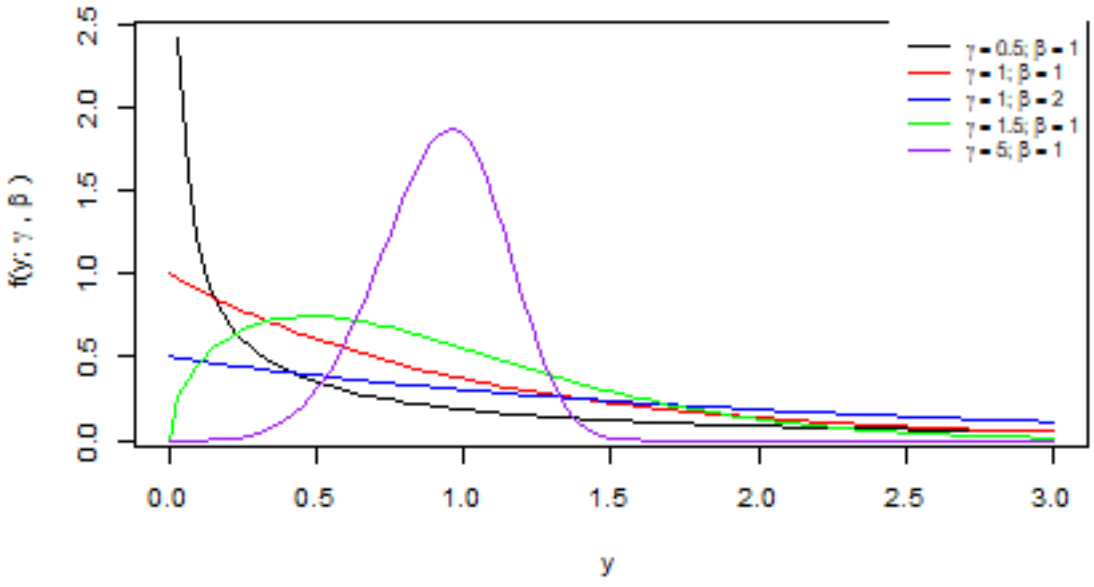
Weibull model

Let $y = (y_1, \ldots, y_n)$ a random sample from a Weibull distribution, $Y \sim \text{We}(\gamma, \beta)$, with parameter $\theta = (\gamma, \beta)$ and density function:

$$f(y; \gamma, \beta) = \frac{\gamma}{\beta}\left(\frac{y}{\beta}\right)^{\gamma-1} e^{-(y/\beta)^{\gamma}}, \quad y \geq 0, \quad \gamma, \beta > 0,$$

where γ is the *shape* parameter and β is the *scale* parameter. The following plot shows the probability density function, by considering 5 combinations of β and λ.

```
sh_val <- c(0.5, 1, 1, 1.5, 5)
sc_val <- c(1, 1, 2, 1, 1)
colours <- c("black", "red", "blue", "green", "purple")
curve(dweibull(x, shape = sh_val[1], scale = sc_val[1]), from = 0, to = 3,
      ylab = expression("f(y;" ~ gamma ~"," ~ beta ~")"), xlab = "y")

for(i in 2 : length(sh_val)){
  curve(dweibull(x, shape = sh_val[i], scale = sc_val[i]),
        add = TRUE, col = colours[i])
}
legend("topright", legend = c(expression(gamma == 0.5 * ";" ~ beta == 1),
                              expression(gamma == 1 * ";" ~ beta == 1),
                              expression(gamma == 1 * ";" ~ beta == 2),
                              expression(gamma == 1.5 * ";" ~ beta == 1),
                              expression(gamma == 5 * ";" ~ beta == 1)),
       col = colours, lty = rep(1, 5), cex = 0.8, box.lty = 0)
```



Then, the likelihood is defined as:

$$\mathcal{L}(\theta) = \mathcal{L}(\gamma, \beta; y) = \prod_{i=1}^{n} \mathcal{L}_i(\gamma, \beta) = \prod_{i=1}^{n} f(y_i; \gamma, \beta),$$

where $\mathcal{L}_i(\gamma, \beta) = f(y_i; \gamma, \beta)$ is the i-th likelihood contribution, and the log-likelihood is defined as:

$$\begin{aligned}\ell(\gamma, \beta; y) = \log\left(\prod_{i=1}^{n} \mathcal{L}_i(\gamma, \beta)\right) &= \sum_{i=1}^{n} \log f(y_i; \gamma, \beta) \\ &= c(y) + n\log(\gamma) - n\gamma\log(\beta) + \gamma\sum_{i=1}^{n}\log(y_i) - \sum_{i=1}^{n}(y_i/\beta)^{\gamma},\end{aligned}$$

where $c(y) = \sum_{i=1}^{n} \log(y_i)$. We may write the log-likelihood function in R

Note: in the following, we consider and implement the negative log-likehood function, since some numerical optimisers that we will see are only able to perform minimisation of functions.

Here, we implement a function taking in argument the 2-dimensional parameter and the data. Then, it computes and returns the value of the negative log-likehood function. Two ways:

- By using the dweibull() function

```
n_logLik_Weib <- function(param, data){
 -sum(dweibull(data, shape = param[1], scale = param[2], log = TRUE))
}
```

- Manually, that is leveraging the expression reported above (note that the last term of the summation, `sum(log(data))`, is the additive constant which could be removed without affecting the inferential results)

```
n_logLik_Weib2 <- function(param, data){
 n <- length(data)
 res <- n * log(param[1]) -
        n * param[1] * log(param[2]) +
        param[1] * sum(log(data)) -
        sum((data/param[2])^param[1]) -
        sum(log(data))
 return( -res )
}
```

Weibull model example: failure times of light bulbs

Let us suppose to observe $n = 15$ failure times (in days) of a sample of light bulbs: we assume $y_1, \ldots, y_{15} \overset{iid}{\sim} \text{We}(\gamma, \beta)$, with γ and β unknown. Here the observed failure times

```
y <- c(173.187, 139.334, 140.205, 139.261, 118.176, 138.105, 193.096,
       163.589, 136.288, 146.226, 134.261, 144.331, 160.262, 107.985, 159.651)
```

Note: to get this values of y, I fixed a seed and I generated the values from a Weibull distribution, by specifying a value for the parameters θ (the values are rounded to the third decimal place). We will see the latter after obtaining the MLE

By using the data, and considering a suitable value for the shape and the scale parameter, at first we can check whether the two negative log-likelihood functions built using the **dweibull**() function and manually return the same value. Thus

```
n_logLik_Weib(param = c(5, 125), y)
```

```
## [1] 79.35402
```

```
n_logLik_Weib2(param = c(5, 125), y)
```

```
## [1] 79.35402
```

We aim to find the maximum likelihood estimate of $\theta = (\gamma, \beta)$. Preliminary, we inspect the log-likelihood function through **contour**() function (or via the **image**() function).

Step to build the plot:

- Define a grid of values for the parameters (we will use the **expand.grid**() function)
- Obtain the log-likelihood values for each point of the grid (Remember to change the sign to the negative log-likelihood)
- Define the levels of the confidence regions
- Use the **contour**() function or the **image**() function to plot the relative log-likelihood, that is $\ell(\theta) - \ell(\hat{\theta})$, with $\theta = (\gamma, \beta)^\top$

Note: here we are plotting the confidence regions based on the LRT. Recall:

- (log) - likelihood ratio test (LRT) $W(\theta) = 2(\ell(\hat{\theta}) - \ell(\theta)) \dot{\sim} \chi_p$, where p is the dimension of θ; In such a case a confidence region with level $1 - \alpha$ is

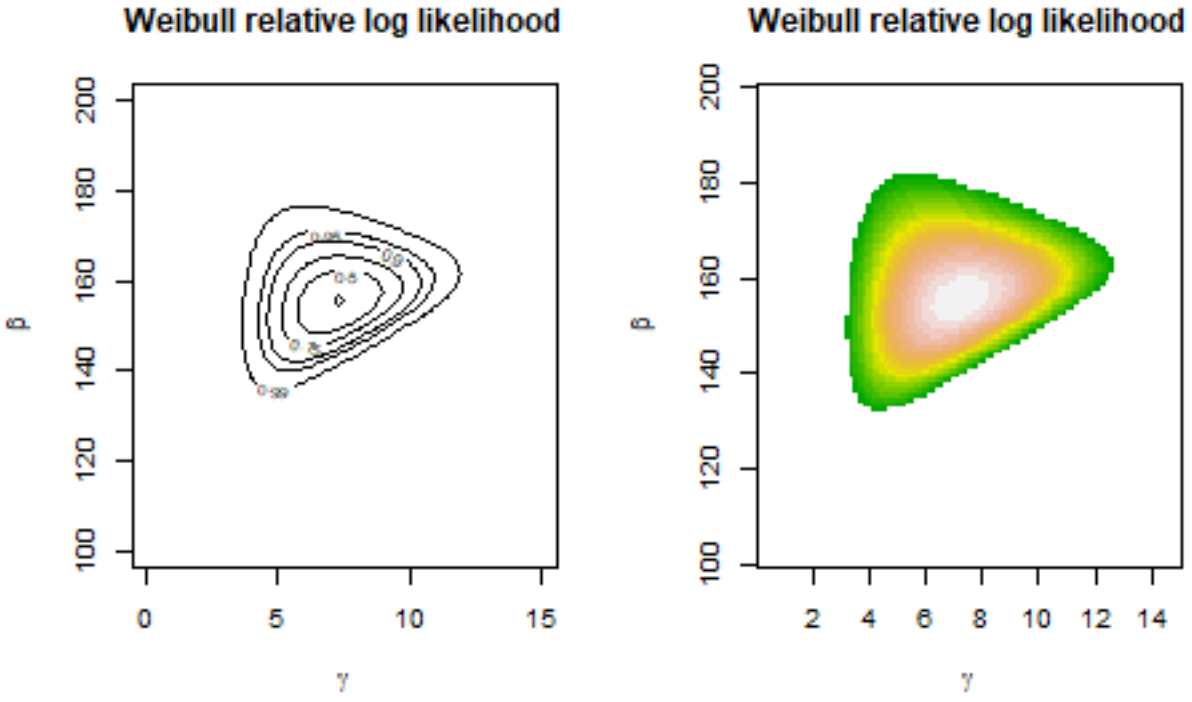
WHY ?

$$\{\theta : W(\theta) < \chi_{p;1-\alpha}\} = \{\theta : \ell(\theta) > \ell(\hat{\theta}) - \frac{1}{2}\chi_{p;1-\alpha}\}$$

  where $\chi_{p;1-\alpha}$ is the quantile of order p of a Chi-square distribution with p degrees of freedom.

```
# Define a parameter grid to plot the log-likelihood
gamma <- seq(0.1, 15, length = 100)
beta <- seq(100, 200, length = 100)
parvalues <- expand.grid(gamma, beta)
```

```r
# obtain the log-likelihood values  for each point of the grid
n_llikvalues <- apply(parvalues, 1, n_logLik_Weib, data = y)
llikvalues <- matrix(-n_llikvalues,
                     nrow = length(gamma), ncol = length(beta),
                     byrow = FALSE)

# Define the confidence levels
conf_levels <- c(0, 0.5, 0.75, 0.9, 0.95, 0.99)

par(mfrow = c(1, 2))
# contour plot
contour(gamma, beta, llikvalues - max(llikvalues),
        levels = -qchisq(conf_levels, 2)/2,
        xlab = expression(gamma), ylab = expression(beta),
        labels = as.character(conf_levels))
title("Weibull relative log likelihood")

# image plot
image(gamma, beta, llikvalues - max(llikvalues),
      zlim = c(-6, 0), col = terrain.colors(20),
      xlab = expression(gamma), ylab = expression(beta))
title("Weibull relative log likelihood")
```



Parameter estimates

We may compute the maximum likelihood estimate $\hat{\theta} = (\hat{\gamma}, \hat{\beta})$ by equating at zero the score, or in other words by solving the score equations:

$$\frac{\partial}{\partial\gamma}\ell(\gamma,\beta;y) = \frac{n}{\gamma} - n\log(\beta) + \sum_{i=1}^{n}\log(y_i) - \sum_{i=1}^{n}(y_i/\beta)^{\gamma}\log(y_i/\beta) = 0$$
$$\frac{\partial}{\partial\beta}\ell(\gamma,\beta;y) = -\frac{n}{\beta}\gamma + \frac{\gamma}{\beta^{\gamma+1}}\sum_{i=1}^{n}y_i^{\gamma} = 0$$

Solving the second equation we get the constrained estimate $\beta_\gamma = (\sum_{i=1}^{n} y_i^{\gamma}/n)^{1/\gamma}$. Substituting it in the first equation, we get

$$\frac{n}{\gamma} + \sum_{i=1}^{n}\log(y_i) - n\frac{\sum_i y_i^{\gamma}\log(y_i)}{\sum_i y_i^{\gamma}} = 0$$

The last equation needs to be solved numerically. To do that, at first we write the score function above, that is the partial derivative of $\ell(\theta)$ w.r.t. to γ (after substituting β_γ in the expression above).

```
logLik_score_g <- function(x, data){
 n <- length(data)
 res <- n/x + sum(log(data)) - n * (sum(data^x * log(data))/(sum(data^x)))
 return(res)
}
```

So the MLE of γ is obtained by solving the equation reported above; this can be done numerically by using the **uniroot**() function; obvioulsy given $\hat{\beta}$ we can obtain the MLE of γ

```
# MLE of gamma: solve the lok-likelihood equation via the uniroot function
gammahat <- uniroot(logLik_score_g, c(1e-5, 15), data = y)$root
gammahat
```

```
## [1] 7.289334
```

```
# MLE of beta, for a fixed value of gamma (the MLE of gamma)
betahat <- mean(y^gammahat)^(1/gammahat)
betahat
```

```
## [1] 155.3337
```

```
# Check if the score is (0,0) at the MLE
library(numDeriv)
grad(c(gammahat, betahat), func = n_logLik_Weib, data = y)
```

```
## [1]  1.563736e-05 -1.244338e-12
```

Of course, in addition to the point estimate we are interested in assessing the variability of our ML estimator. Let $\theta = (\gamma, \beta)$ be the 2-dimensional parameter vector. In this respect, we compute the **observed information** matrix

$$J(\theta; y) = -\frac{\partial^2 \ell(\theta; y)}{\partial \theta \partial \theta^T} \qquad [J(\theta)]_{rs} = -\frac{\partial^2 \ell(\theta; y)}{\partial \theta_r \partial \theta_s}, \quad r, s = 1, 2$$

$$\frac{\partial^2 \ell(\theta; y)}{\partial \gamma^2} = -\frac{n}{\gamma^2} - \sum_{i=1}^{n} \left(\frac{y_i}{\beta}\right)^{\gamma} \left\{ \log\left(\frac{y_i}{\beta}\right) \right\}^2$$

$$\frac{\partial^2 \ell(\theta; y)}{\partial \gamma \partial \beta} = -\frac{n}{\beta} + \sum_{i=1}^{n} \left(\frac{y_i^{\gamma}}{\beta^{\gamma+1}}\right) \left\{ \gamma \log\left(\frac{y_i}{\beta}\right) + 1 \right\}$$

$$\frac{\partial^2 \ell(\theta; y)}{\partial \beta^2} = \frac{n\gamma}{\beta^2} - \frac{\gamma(\gamma+1)}{\beta^{\gamma+2}} \sum_{i=1}^{n} y_i^{\gamma}$$

Taking the diagonal elements of its inverse, evaluated in $(\hat{\gamma}, \hat{\beta})$, we obtain an estimate for the variance of our estimators. Recall that $\hat{\theta} \dot{\sim} \mathcal{N}(\theta, i^{-1}(\theta))$, where if available we can replace the expected information matrix with $i(1hat\theta)$ or $j(\hat{\theta})$.

```
n <- length(y)
# observed information matrix evaluated at the MLE
jhat <- matrix(NA, 2, 2)
jhat[1,1] <- n/gammahat^2 + sum((y/betahat)^gammahat*(log(y/betahat)^2))
jhat[1,2] <- n/betahat -
  sum(y^gammahat/(betahat^(gammahat+1)) * (gammahat*log(y/betahat)+1))
jhat[2,1] <- jhat[1,2]
jhat[2,2] <- -n*gammahat/(betahat^2) +
  gammahat*(gammahat+1) * sum(y^gammahat)/(betahat^(gammahat+2))

jhat
```

```
##             [,1]        [,2]
## [1,]  0.60640858 -0.04735997
## [2,] -0.04735997  0.03303206
```

```
# Check by using the hessian function of the numDeriv package
hessian(c(gammahat, betahat), func = n_logLik_Weib, data = y)
```

```
##             [,1]        [,2]
## [1,]  0.60640858 -0.04735997
## [2,] -0.04735997  0.03303206
```

```
# Estimate of the std.err of the MLE estimators
mle.se <- sqrt(diag(solve(jhat)))
mle.se
```

```
## [1] 1.362714 5.838747
```

Confidence regions

We will see confidence regions for θ, with $dim(\theta) = p$, based on

- (log) - likelihood ratio test (LRT) $W(\theta) = 2(l(\hat{\theta}) - l(\theta)) \dot{\sim} \chi_p$, which is asymptotically distributed as a Chi-square distribution with p degrees of freedom; In such a case a confidence region with level $1 - \alpha$ is

$$\{\theta : W(\theta) < \chi_{p;1-\alpha}\} = \{\theta : l(\theta) > l(\hat{\theta}) - \frac{1}{2}\chi_{p;1-\alpha}\}$$

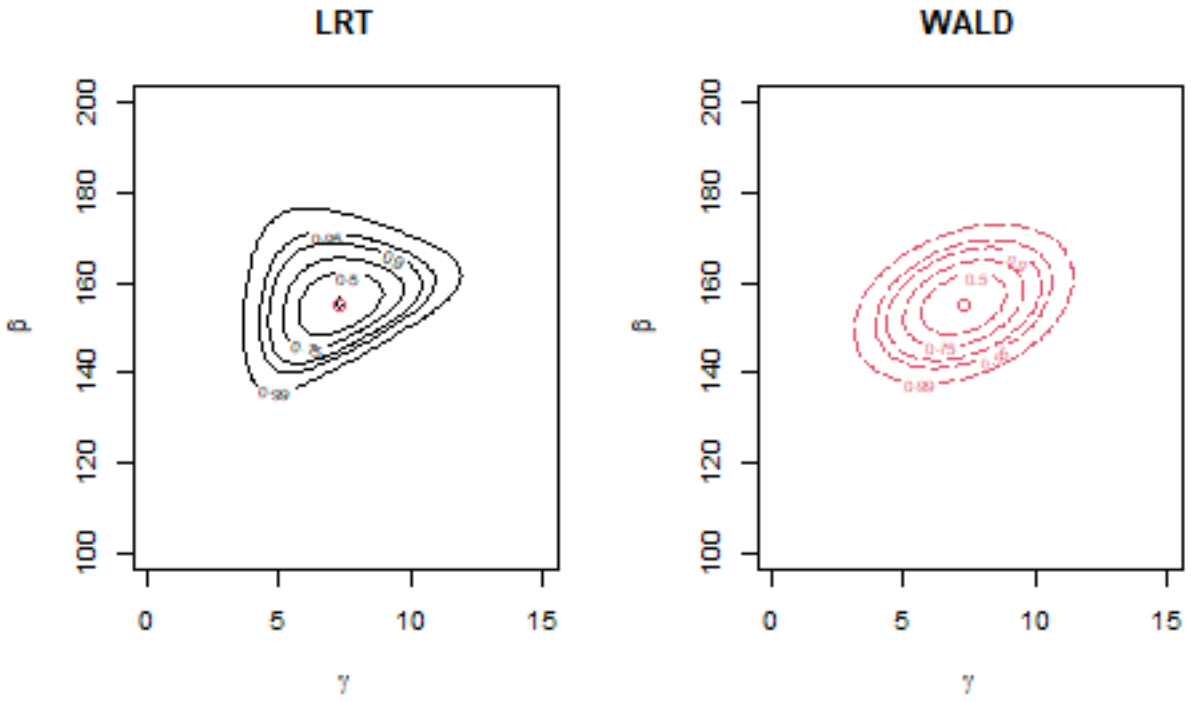
- Wald test is based on the quadratic approximation of the log-likelihood function:

$$W_e(\theta) \approx -\frac{1}{2}(\hat{\theta} - \theta)^\top j(\hat{\theta})(\hat{\theta} - \theta)$$

  , which is asymptotically distributed as $W(\theta)$, that is χ_p. In such a case a confidence region with level $1 - \alpha$ is

$$\{\theta : W_e(\theta) < \chi_{p;1-\alpha}\}$$

```
par(mfrow = c(1, 2))
contour(gamma, beta, llikvalues - max(llikvalues),
        ylab = expression(beta), xlab = expression(gamma),
        levels = -qchisq(conf_levels, 2)/2, main = 'LRT',
        labels = as.character(conf_levels))
points(gammahat, betahat, col = 2)

weib.y.mle <- c(gammahat, betahat)

wt <-  function(par, jhat){
  difftheta <-   as.matrix(weib.y.mle-par)
  return(-.5*t(difftheta) %*% jhat %*% difftheta)
}

waldvalues <- apply(parvalues, 1, wt, jhat = jhat)

waldvalues <- matrix(waldvalues, nrow = length(gamma),
                     ncol = length(beta), byrow = F)

contour(gamma, beta, waldvalues,
        ylab = expression(beta), xlab = expression(gamma),
        levels = -qchisq(conf_levels,2)/2,
        labels = as.character(conf_levels),
        col = 2, main = 'WALD', lty = 'longdash')
points(gammahat, betahat, col = 2)
```



Note: some differences between Wald-type confidence regions and the LRT ones

Wald-type confidence intervals (regions):

- Symmetric around $\hat{\theta}$ by construction
- They could be not included into the parametric space
- Lack of invariance w.r.t. reparametrisations

LRT-type (sometimes you can find deviance-type) confidence intervals (regions):

- Generally, they are not symmetric around $\hat{\theta}$
- They are included into the parametric space
- Invariance w.r.t. reparametrisations
- Usually, they show better empirical coverage than the Wald-type ones

Extra exercise 1

Compute the MLE and the observed information matrix for a gamma model with shape parameter α and scale parameter β. Recall that for $Y \sim \text{Ga}(\alpha, \beta)$ the probability density function is

$$f_Y(y; \alpha, \beta) = \frac{1}{\beta^\alpha \Gamma(\alpha)} y^{\alpha-1} e^{-y/\beta}, \quad y \geq 0, \quad \alpha, \beta > 0,$$

Numerical optimisation

So far, we used R simply as a pocket calculator, computing MLE and the variance of our estimators *analytically*, and then obtaining the numerical values just plugging into the formulas the inputs. However, remind the MLE for γ, where the equation:

$$\frac{n}{\gamma} + \sum_{i=1}^{n} \log(y_i) - n\frac{\sum_i y_i^\gamma \log(y_i)}{\sum_i y_i^\gamma} = 0$$

does not have an analytic solution. Many times we do not have a closed form for MLE estimates and we may need **numerical optimisation**. R provides various functions for performing numerical methods:

- nlm(): minimizes a function using a Newton-type algorithm. It needs a starting value and does not allow constraints on the parameters. It is usually fast and reliable. It returns the ML estimate $\hat{\theta}$ (estimate), the value of the likelihood $-l(\hat{\theta})$ (minimum) and the hessian (hessian), if hessian = TRUE.
- optim(): minimizes a function using Nelder-Mead, quasi-Newton and conjugate gradients algorithms. It includes an option for box-constrained optimization, and it requires a starting value. It returns the ML estimate $\hat{\theta}$ (par) and the value of the likelihood $-l(\hat{\theta})$ (value) and the hessian (hessian), if hessian = TRUE. You also can maximize the function by using fnscale = -1 in control argument.
- nlminb(): often is more stable, robust and reliable than optim (in particular with "nasty" functions). It performs only minimization and does not yield numerical derivatives as output. It returns the ML estimate $\hat{\theta}$ (par) and the value of the likelihood $-l(\hat{\theta})$ (objective).
- optimize(): by using a combination of golden section search and successive parabolic interpolation, searches in an interval for a minimum or a maximum (if maximum = TRUE) of a function. It returns the ML estimate $\hat{\theta}$ (minimum) and the value of the likelihood $l(\hat{\theta})$ (objective). Drawback: suited only for one-dimensional parameter.

There exists others functions (e.g. https://cran.r-project.org/web/packages/ucminf/index.html)

```
weib.nlm_start1 <- nlm(f = n_logLik_Weib, p = c(5, 160),
                       data = y, hessian = TRUE)
weib.nlm_start1
```

```
## $minimum
## [1] 67.48238
##
## $estimate
## [1]   7.289296 155.333560
##
## $gradient
## [1]  1.113194e-06 -1.282634e-07
##
## $hessian
##             [,1]        [,2]
## [1,]  0.60638834 -0.04728076
## [2,] -0.04728076  0.03299795
##
## $code
## [1] 1
##
## $iterations
## [1] 10
```

```
weib.nlm_start2 <- nlm(f = n_logLik_Weib, p = c(0.1, 0.1),
                       data = y, hessian = TRUE)
weib.nlm_start2
```

```
## $minimum
## [1] 67.48238
##
## $estimate
## [1]   7.289289 155.333552
##
## $gradient
## [1] -2.674787e-06 -5.516609e-08
##
## $hessian
##             [,1]        [,2]
## [1,]  0.60638881 -0.04728066
## [2,] -0.04728066  0.03299788
##
## $code
## [1] 1
##
## $iterations
## [1] 27
```

```r
weib.optim_start1 <- optim(par = c(5, 160), fn= n_logLik_Weib, hessian = TRUE,
  data = y, method = "L-BFGS-B", lower = rep(1e-7,2), upper = rep(Inf,2))
weib.optim_start1
```

```
## $par
## [1]   7.289303 155.333650
##
## $value
## [1] 67.48238
##
## $counts
## function gradient
##       11       11
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
##
## $hessian
##             [,1]        [,2]
## [1,]  0.60641101 -0.04735962
## [2,] -0.04735962  0.03303179
```

```r
weib.optim_start2 <- optim(par = c(0.1, 0.1), fn = n_logLik_Weib,
  data = y, method = "L-BFGS-B", lower = rep(1e-7, 2), upper = rep(Inf,2))
weib.optim_start2
```

```
## $par
## [1]   7.289305 155.333656
##
## $value
## [1] 67.48238
##
## $counts
## function gradient
##       35       35
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Reparametrisations

To avoid numerical problems, it is better to work with **reparametrizations**:

$$\psi = \psi(\theta) = (\psi_1 = \log(\gamma), \psi_2 = \log(\beta)).$$

Generally, for numerical purposes, it is convenient to reparameterize the model in such a way that the new parameter space is unbounded. In this case $\psi \in \mathbb{R}^2$. At such point, the parameter estimates will be expressed in the log-scale, and we need to re-express them in the original scale. Obviously, $\theta = \theta(\psi) = (e^{\psi_1}, e^{\psi_2})$.

```
# Reparameterization
theta <- function(omega) exp(omega)

# Negative log-likelihood
n_logLik_Weib_rep <- function(param, data) n_logLik_Weib(theta(param), data)

# Optimize the log-likelihood function by using nlm (
# also here the are some warnings but the algorithm works)
weib_nlm_start3_rep <- nlm(f = n_logLik_Weib_rep, p = c(0, 0), data = y)
weib_nlm_start3_rep
```

```
## $minimum
## [1] 67.48238
##
## $estimate
## [1] 1.986402 5.045572
##
## $gradient
## [1] -1.559587e-06  1.847624e-06
##
## $code
## [1] 1
##
## $iterations
## [1] 35
```

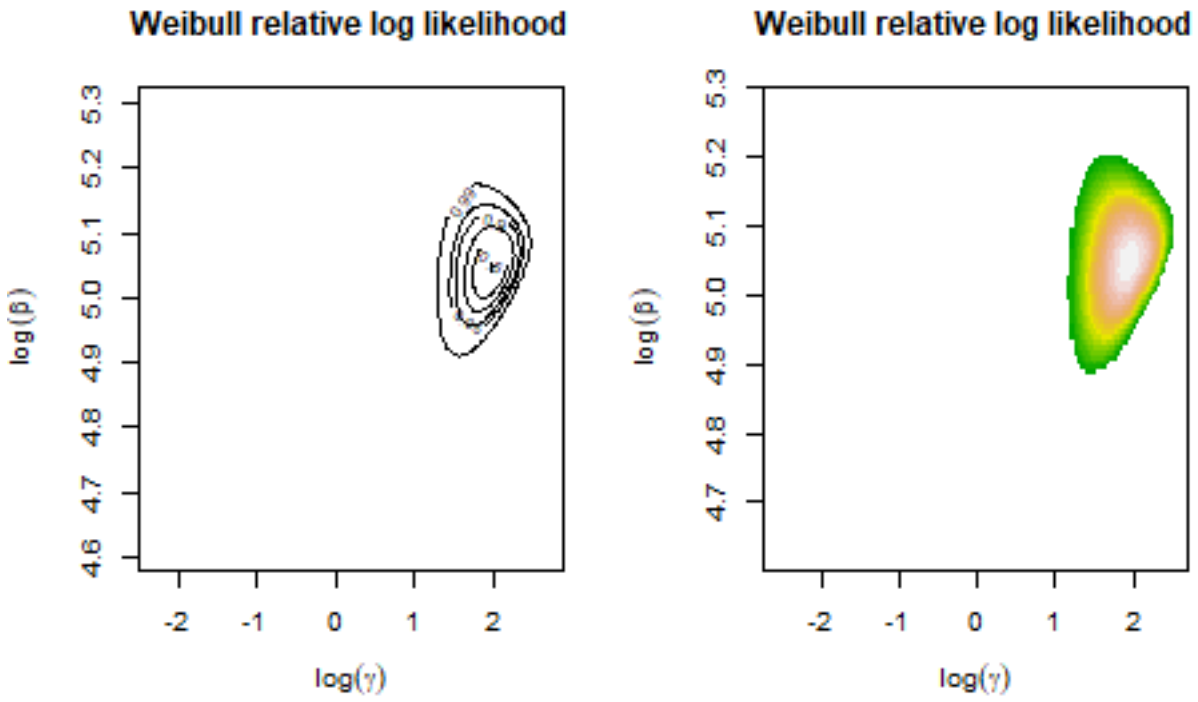
```
# Check
theta(weib_nlm_start3_rep$estimate)
```

```
## [1]   7.289262 155.333202
```

```
weib.nlm_start1$estimate
```

```
## [1]   7.289296 155.333560
```

```r
# Contour and image plot
gamma <- seq(0.1, 15, length = 100)
beta <- seq(100, 200, length = 100)
parvalues <- expand.grid(log(gamma), log(beta))

n_llikvalues <- apply(parvalues, 1, n_logLik_Weib_rep, data = y)
llikvalues <- matrix(-n_llikvalues,
                     nrow = length(gamma), ncol = length(beta),
                     byrow = FALSE)

conf_levels <- c(0, 0.5, 0.75, 0.9, 0.95, 0.99)

par(mfrow = c(1, 2))
contour(log(gamma), log(beta), llikvalues - max(llikvalues),
        levels = -qchisq(conf_levels, 2)/2, labels = as.character(conf_levels),
        xlab = expression(log(gamma)), ylab = expression(log(beta)))
title("Weibull relative log likelihood")

image(log(gamma), log(beta), llikvalues - max(llikvalues),
      zlim = c(-6, 0), col = terrain.colors(20),
      xlab = expression(log(gamma)), ylab = expression(log(beta)))
title("Weibull relative log likelihood")
```



The optim() function provides a lot of numerical methods, such us Nelder-Mead, quasi-Newton, conjugate-gradient methods and simulated annealings. As a drawback, the user has to set up very carefully the initial parameters and the adopted method, since the final solution may be quite sensitive to these choices... To compute the observed information, the function optimHess() computes numerical derivatives of generic functions, if hessian = TRUE was forgotten in optim().

```
# Working on the reparameterization
# Log-likelihood optimization by using optim:
# starting values = c(1,20), quasi-Newton Method
weib_optim_start3_rep_qn <- optim(par = c(1, 20), fn = n_logLik_Weib_rep,
                                  method = "BFGS", data=y)


# starting values = c(1,6), conjugate-gradient Method
weib_optim_start3_rep_CG <- optim(par = c(1, 6), fn = n_logLik_Weib_rep,
                                  method = "CG", data = y)

# check
theta(weib_optim_start3_rep_qn$par)
```

```
## [1]   7.289313 155.333473
```

```
theta(weib_optim_start3_rep_CG$par)
```

```
## [1]   7.289064 155.333536
```

```
weib.nlm_start1$estimate
```

```
## [1]   7.289296 155.333560
```

```
# Check the hessian
optimHess(theta(weib_optim_start3_rep_qn$par), n_logLik_Weib, data = y)
```

```
##             [,1]        [,2]
## [1,]  0.60641566 -0.04736195
## [2,] -0.04736195  0.03303230
```

```
jhat
```

```
##             [,1]        [,2]
## [1,]  0.60640858 -0.04735997
## [2,] -0.04735997  0.03303206
```

Extra: R code integrating the material on the profile likelihood

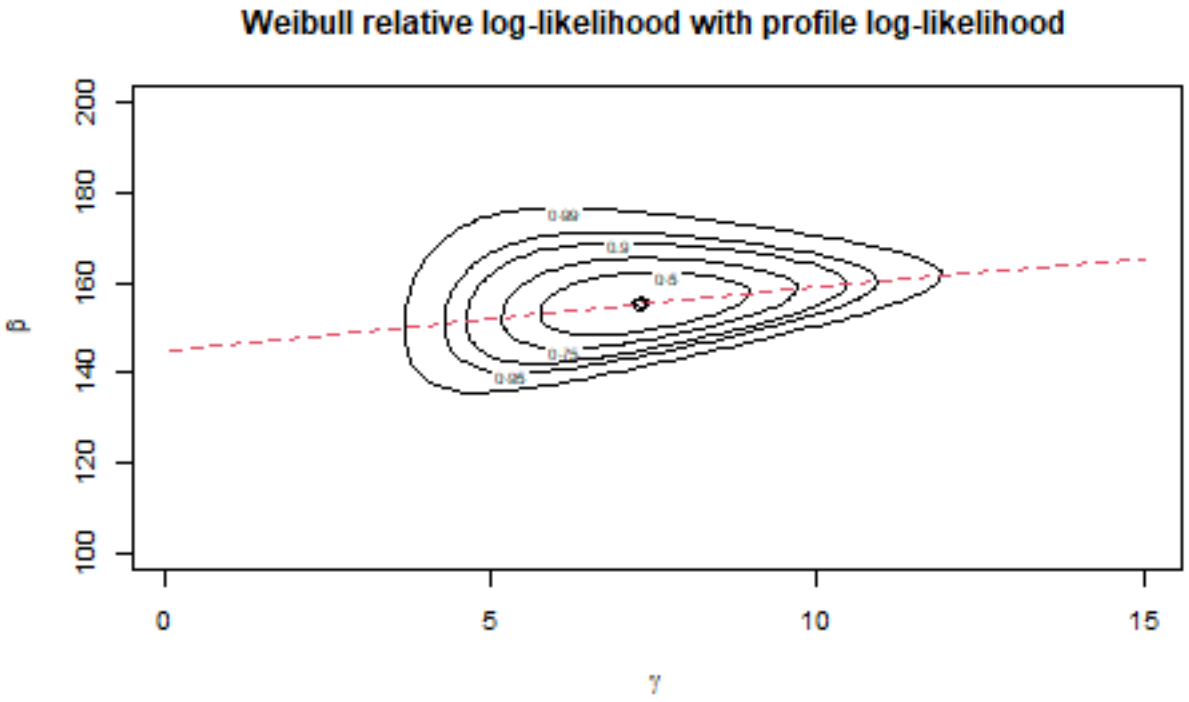
In practical situations, some components of the parameter vector θ are more important than others; essentially, in such situations it is of interest for us making inference only on those subgroups of parameters. In the Weibull case, we could treat γ as the *parameter of interest* and β as the *nuisance parameter*. We may then define the profile **log-likelihood**:

$$\ell_P(\gamma) = \max_{\beta} \ell(\gamma, \beta; y) = \ell(\gamma, \hat{\beta}_\gamma; y),$$

where $\hat{\beta}_\gamma$ is the *constrained* MLE for β (note that we obtained above), with γ fixed. Some issues deserve a quick consideration:

- the profile log-likelihood is simply the log-likelihood for the bi-dimensional parameter θ, with the nuisance component β replaced by $\hat{\beta}_\gamma = (\sum_{i=1}^n y_i^\gamma/n)^{1/\gamma}$.
- ℓ_P is not a *genuine* likelihood. However, it has some nice features which ease to work with it.

```
weib.y.mle<- optim(par = c(1,1), fn = n_logLik_Weib, hessian = TRUE,
                   method = "L-BFGS-B", data = y,
                   lower = rep(1e-7, 2), upper = rep(Inf, 2) )

# Visualisation of the profile log-likelihood
# on the contour plot of the log-likelihood
gamma <- seq(0.1, 15, length = 100)
beta <- seq(100, 200, length = 100)
parvalues <- expand.grid(gamma, beta)
llikvalues <- apply(parvalues, 1, n_logLik_Weib, data = y)
llikvalues <- matrix(-llikvalues, nrow = length(gamma),
                     ncol = length(beta), byrow = FALSE)

conf.levels <- c(0, 0.5, 0.75, 0.9, 0.95, 0.99)

par(mfrow=c(1,1))
contour(gamma, beta, llikvalues - max(llikvalues),
        levels = -qchisq(conf.levels, 2)/2,
        xlab = expression(gamma),
        ylab = expression(beta),
        labels = as.character(conf_levels))


beta.gamma <- sapply(gamma, function(x) mean(y^x)^(1/x))
# line of the constrained estimate
lines(gamma, beta.gamma, lty = "dashed", col = 2)
points(weib.y.mle$par[1], weib.y.mle$par[2])
title("Weibull relative log-likelihood with profile log-likelihood")
```



In some sense, we *reduced the dimension* of the problem, and we acknowledged that we may work with a one-dimensional likelihood evaluated in the constrained value $\hat{\beta}_\gamma$ for the nuisance component. Then, we may now compute some **deviance confidence intervals** with level $1-\alpha$ as:

$$\{\gamma : l_P(\gamma) \geq l_P(\hat{\gamma}) - \frac{1}{2}\chi^2_{1;1-\alpha}\},$$

where $\chi^2_{1;1-\alpha}$ is the $1-\alpha$-th quantile of a chi-squared distribution with 1 d.f, the asymptotic distribution for the **profile likelihood-ratio test statistic**:

$$W_P(\gamma) = 2\{l_P(\hat{\gamma}) - l_P(\gamma)\}.$$

```
# profile log-likelihood
n_logLik_Weib_profile <- function(gamma, data){
 beta.gamma <- mean(data^gamma)^(1/gamma)
 n_logLik_Weib(c(gamma, beta.gamma), data)
}
```

In this respect, we must vectorise the function for properly visualising it. Vectorised function allows to compute the function in a vector of points and get the corresponding output results.

Thus, we vectorise w.r.t. γ

```
# vectorize the function with respect to gamma
n_logLik_Weib_profile_v <- Vectorize(n_logLik_Weib_profile, 'gamma')

# Wrong because it returns the log-likelihood evaluated in c(5,6)
n_logLik_Weib_profile(c(5, 6, 7), data = y)
```

```
## [1] 156638836
```

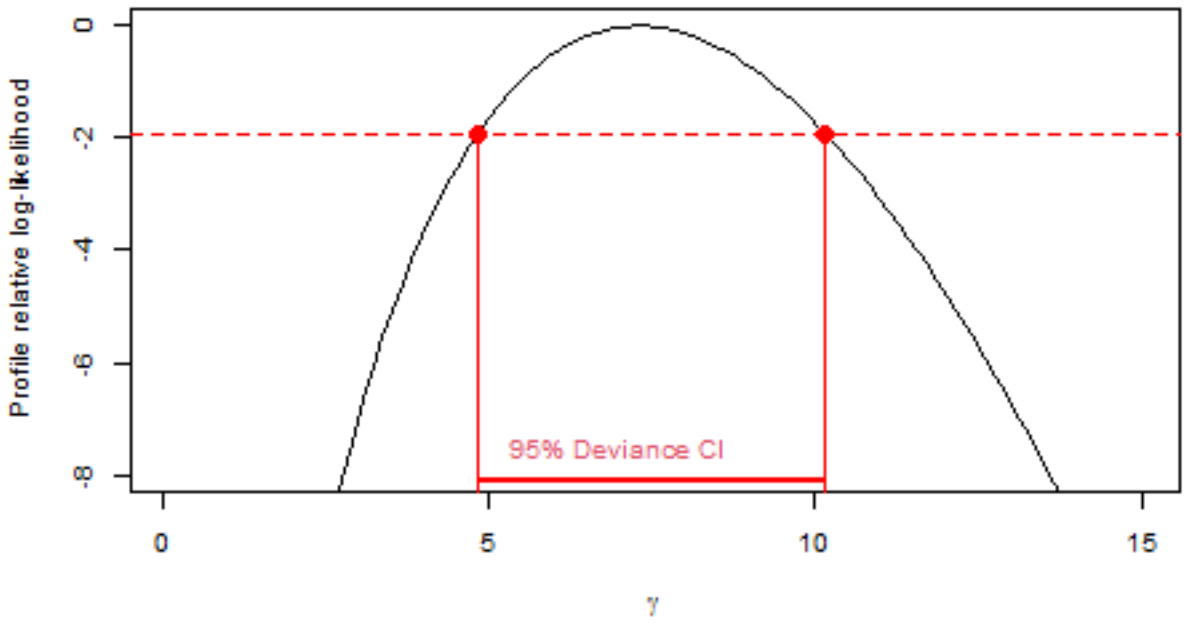
```
# Indeed,
n_logLik_Weib(c(5,6), data = y)
```

```
## [1] 156638836
```

```
# While what we want is
n_logLik_Weib_profile_v(c(5, 6, 7), data = y)
```

```
## [1] 69.12127 67.96645 67.50529
```

```
# Plot the relative profile log-likelihood
plot(function(x) -n_logLik_Weib_profile_v(x, data = y) + weib.y.mle$value,
     from = 0.1, to = 15, xlab = expression(gamma),
     ylab = "Profile relative log-likelihood", ylim =  c(-8,0))

conf.level <- 0.95 # set the confidence level

# Upper and the lower limits of the deviance confidence interval
abline(h = -qchisq(conf.level, 1)/2, lty = "dashed", col = "red")

# Find the numerical values by using the uniroot function
lrt.ci1 <- uniroot(function(x) -n_logLik_Weib_profile_v(x, data = y) +
                     weib.y.mle$value + qchisq(conf.level, 1)/2,
                   c(1e-7, weib.y.mle$par[1]))$root

lrt.ci <- c(lrt.ci1,
            uniroot(function(x) -n_logLik_Weib_profile_v(x, data = y) +
                                  weib.y.mle$value + qchisq(conf.level, 1)/2,
                                c(weib.y.mle$par[1], 15))$root)

# Plotting some quantities in the relative profile log-likelihood plot
segments(lrt.ci[1], -qchisq(conf.level, 1)/2,
         lrt.ci[1], -n_logLik_Weib_profile_v(lrt.ci[1], data = y), col = "red")
segments(lrt.ci[2], - qchisq(conf.level, 1)/2,
         lrt.ci[2], -n_logLik_Weib_profile_v(lrt.ci[2], data = y), col = "red")
points(lrt.ci[1], -qchisq(conf.level, 1)/2, pch = 16, col = "red", cex = 1.5)
points(lrt.ci[2], -qchisq(conf.level, 1)/2, pch = 16, col = "red", cex = 1.5)
segments(lrt.ci[1], -8.1, lrt.ci[2], -8.1, col = "red", lty = 1, lwd = 2)
text(7, -7.5, "95% Deviance CI", col = 2)
```



Extra Exercise: The **Wald confidence interval** with level $1-\alpha$ is defined as:

$$\hat{\gamma} \pm z_{1-\alpha/2} j_P(\hat{\gamma})^{-1/2}.$$

Compute the Wald confidence interval of level 0.95, plot the results, and evaluate via simulation the empirical coverage of the confidence interval.

Extra Exercise: Repeat the steps above —write the profile log-likelihood, plot it and find the deviance confidence intervals— considering this time γ as a nuisance parameter and β as the parameter of interest.