Assignment4

Group 3

12/13/2019

Ex 1

```
angles <- read.table("angles.txt")
angles <- as.vector(angles$x)
```

K estimation by MLE numerical optimization from previous exercise:

```
dsink <- function(x, k = 1, lg = FALSE) {
  sinintegral <- integrate(function(x) sin(x) ^ k, lower = 0, upper = pi)$value
  if (lg == FALSE) {
    return(sin(x) ^ k / sinintegral)
  }
  else{
    return(log(sin(x) ^ k / sinintegral))
  }
}

minusll <- function(k, xvals) {
  return(-sum(dsink(xvals, k, lg = TRUE)))
}

k_est <- optimize(f = minusll, xvals = angles, interval = c(0, 100))$minimum
```

Ex 1.1 Build a 99% confidence interval for k based on the data in angles.txt, how you can estimate the standard error?

$$z_{\alpha/2} = F^{-1}(1 - (\alpha/2))$$

$$C_n = (a, b) = \left(\bar{X} - \frac{\sigma}{\sqrt{n}} z_{\alpha/2}, \bar{X} + \frac{\sigma}{\sqrt{n}} z_{\alpha/2}\right)$$

$$SEM = \frac{\sigma}{\sqrt{n}}$$

First we find $SEM(\hat{k})$ by bootstrap:

```
vect_k_est_bt <- replicate(1000, expr = {
  angles_bt <- sample(angles, size = length(angles), replace = TRUE)
  optimize(f = minusll, xvals = angles_bt, interval = c(0, 100))$minimum
})

se_est <- sd(vect_k_est_bt)
se_est
```

```
## [1] 0.5321813
```

Than we calculate the 99% confidence interval for the parameter k:

```
alpha <- 0.01
z <- qnorm(1 - alpha/2)
a <- k_est - se_est * z
b <- k_est + se_est * z
paste("99% Confidence Interval for k: ( a =", a, ", b= ", b, ")")
```

```
## [1] "99% Confidence Interval for k: ( a = 10.0295852115688 , b=  12.7712015980309 )"
```

We can also calculate the percentile confidence interval:

```
quantile(vect_k_est_bt, probs = c(alpha/2, 1 - alpha/2))
```

```
##     0.5%    99.5%
## 10.24543 12.85868
```

Ex 1.2 Test if k is larger than 10 at a confidence level alpha equeal to 0.05.

One side Wald test $H_0 = \theta \leq 10$

```
k0 <- 10
alpha <- 0.05
z <- qnorm(1 - alpha)
w <- (k_est - k0) / se_est      # w > Z  TRUE, so we reject Ho and we thinK theta > 10
w > z                           # (we can assume that because it is a 1 side test)
```

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

$$H_0 = \theta \leq 10$$
$$H_1 = \theta > 10$$

Since $w > Z$ we reject the null hyphotesis, so with 95% confidence we think that $k > 10$.

The approximate p-value is: (smaller it is, the stronger the evidence that the H0 is not true. WARNING: a large value is not a strong evidence in favor of H0 (ah no?!), so it is not the probability that H0 is true!)

```
pval <- 1 - pnorm(w)       #it is like this (1 - cdf) because it is one-sides;
pval                       # for 2 sided it is (2 * (cdf) - abs(w))
```

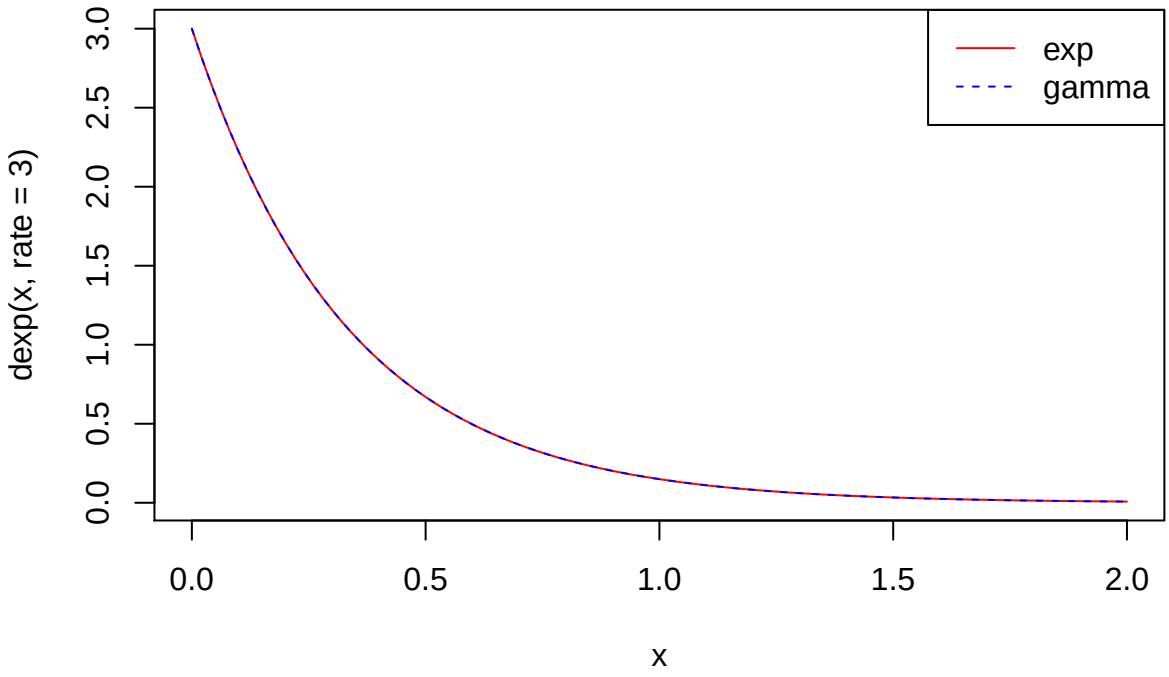
```
## [1] 0.004251423
```

Ex 2

```
spikes <- read.table("neuronspikes.txt")
spikes <- spikes$V1
```

2.1 The exponential distribution is a special case of the gamma distribution when the shape parameter is equal to 1. Check this fact graphically in R.

```
curve(dexp(x, rate = 3), col = "red", from = 0, to = 2)
curve(dgamma(x, rate = 3, shape = 1), col = "blue", lty = 2, add = TRUE)
legend("topright", legend = c("exp", "gamma"), col = c("red", "blue"), lty = c(1, 2))
```



2.2 Since the exponential model is nested in the gamma model we can perform the likelihood ratio test to select between the two models.

We want to use now the likelihood ratio test to check if the simpler exponential model is sufficient to model the data. We can apply the likelihood ratio test because the exponential model is nested (that is, it is a special case) of the gamma model, specifically when the shape parameter $\alpha = 1$ (and the rate parameter of the obtained exponential is exactly the gamma rate $\lambda = \beta$) (as we have seen in ex. 2.1).

LRT to test whether the parameter_rich model (gamma) is significantly better than the simpler model.

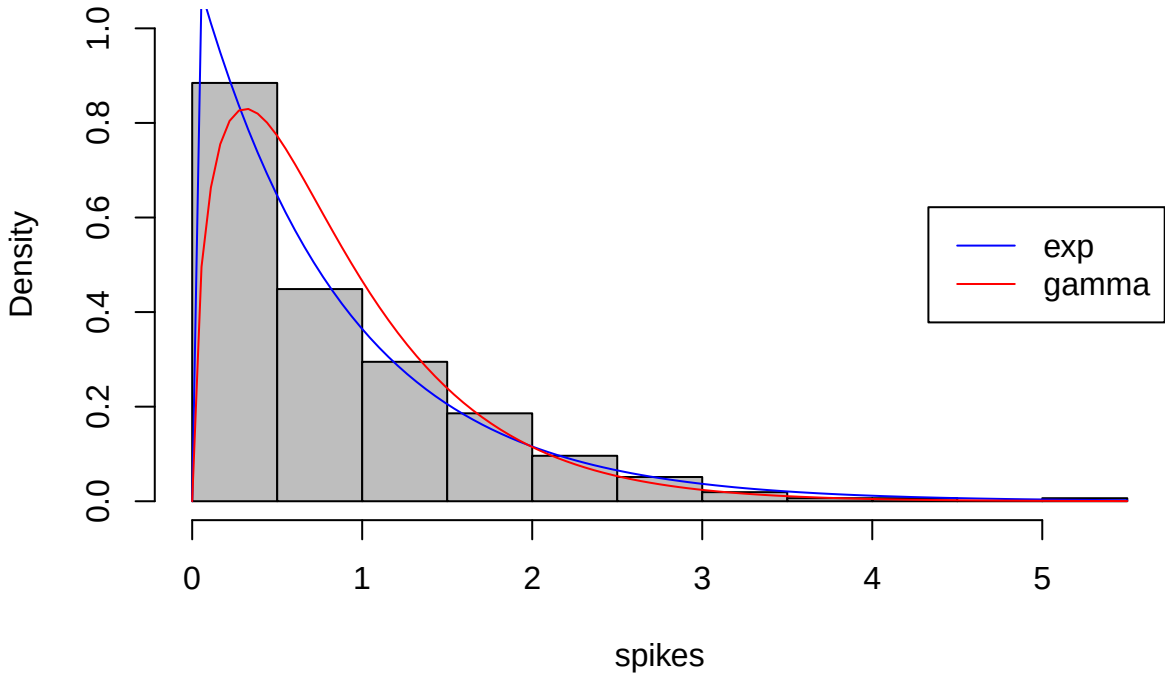
H_0 = the model M1 is sufficient to describe the data.

```
exp_mle_rate <- 1/mean(spikes)                          # find mle for exponential distr. par
exp_mle_value <- sum(dexp(spikes, rate = exp_mle_rate, log = TRUE))# calculate max L for exp

mll_gamma <- function(par, xvals){                              # find mle for gamma distr. par
  return(-sum(dgamma(xvals, shape = par[1], rate = par[2], log = TRUE)))
}
gamma_mle <- optim(f = mll_gamma, par = c(1, 1), xvals = spikes)
gamma_mle_value <- -gamma_mle$value                                  # calculate max L for gamma
gamma_mle_shape <- gamma_mle$par[1]
gamma_mle_rate <- gamma_mle$par[2]
```

We can plot the densities of the exp and gamma distributions together with the spikes:

```
hist(spikes, freq = FALSE, col = "gray", ylim = c(0,1))
curve(dexp(x, rate = exp_mle_rate), col = "blue", add = TRUE)
curve(dgamma(x, shape = gamma_mle_shape, rate = gamma_mle_rate), col = "red", add = TRUE)
legend("right", c("exp", "gamma"), col = c("blue", "red"), lty = 1)
```



Under the null hypothesis assumption: $\lambda(X_1, \ldots, X_n) = -2\log(q(X)) \approx \chi^2_{d-d_0}$.

We can now calculate λ: (we check that it is a positive number, as the theory says)

```
LR_lambda <- 2*(gamma_mle_value - exp_mle_value) # lambda = 2*(ll_large.m - ll_small.m)
LR_lambda # bigger is the LR_lambda and more convenient is to use the less restricted model
```

```
## [1] 33.07526
```

LR test statistic is 33.07 (distributed chi_squared), with 1 degree of freedom (degree of freedom correspond to difference btw paramenters of 'parent' distribution(gamma) and those of the nested distribution (exponential).

Now we compare λ with the upper quantile of the chi squared distribution with $d - d_0$ degrees of freedom:

```
alpha = 0.05
LR_lambda > qchisq(1-alpha, df = 1)    # not sure about this
```

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

Since $\lambda(X_1, \ldots, X_n) > \chi^2_{d-d_0;\alpha}$ I reject H_0. <- (NOT SURE HERE)

The p_value $= 1 - F^2_{\chi_{d-d_0}}(\lambda(X_1, \ldots, X_n))$ (in our case $d - d_0 = 2 - 1 = 1$)

```
pvalue <- 1 - pchisq(LR_lambda, df = 1)
pvalue
```

```
## [1] 8.86598e-09
```

P-value is really low (p-value < alpha?) so we reject the null hypothesis.

Thus we can state that it is preferable to use the gamma model to describe the data.

Ex 3.

Ex 3.1 Find the MLE for the exponential, gamma, inverse Gaussian and log_normal model.

Mll exponential (check Gherardo's code! he did it with formula, not numerical optimization) :

```
mll_exp <- function(par, xvals){
  return(-sum(dexp(xvals, rate = par, log = TRUE)))
}
exp_mle <- optimize(f = mll_exp, interval = c(0,100), xvals = spikes)
exp_mle_par <- exp_mle$minimum
exp_mle_value <- -exp_mle$objective              # convert from -sum(logll) to +sum(logll)
exp_mle_value
```

```
## [1] -269.2388
```

Mll gamma (check Gherardo's code from ex.2:

```
mll_gamma <- function(par, xvals){
  return(-sum(dgamma(xvals, shape = par[1], rate = par[2], log = TRUE)))
}
gamma_mle <- optim(f = mll_gamma, par = c(1, 1), xvals = spikes)
gamma_mle_par <- gamma_mle$par
gamma_mle_value <- -gamma_mle$value
gamma_mle_value
```

```
## [1] -252.7012
```

Mll inverse Gaussian:

```
dinvnorm <- function(x, mu, lambda, lg = FALSE){
  if(lg == TRUE){
    return(log(sqrt(lambda/(2*pi*(x^3)))*exp(-lambda*((x-mu)^2)/(2*(mu^2)*x))))
    }
  else{
    return(sqrt(lambda/(2*pi*(x^3)))*exp(-lambda*((x-mu)^2)/(2*(mu^2)*x)))
    }
}
mll_invnorm <- function(par, xvals){
  return(-sum(dinvnorm(xvals, mu = par[1], lambda = par[2], lg = TRUE)))
  }
invnorm_mle <- optim(par = c(1, 1), fn = mll_invnorm, xvals = spikes)
invnorm_mle_par <- invnorm_mle$par
invnorm_mle_value <- -invnorm_mle$value
invnorm_mle_value
```

```
## [1] -235.4785
```

Mll log normal:

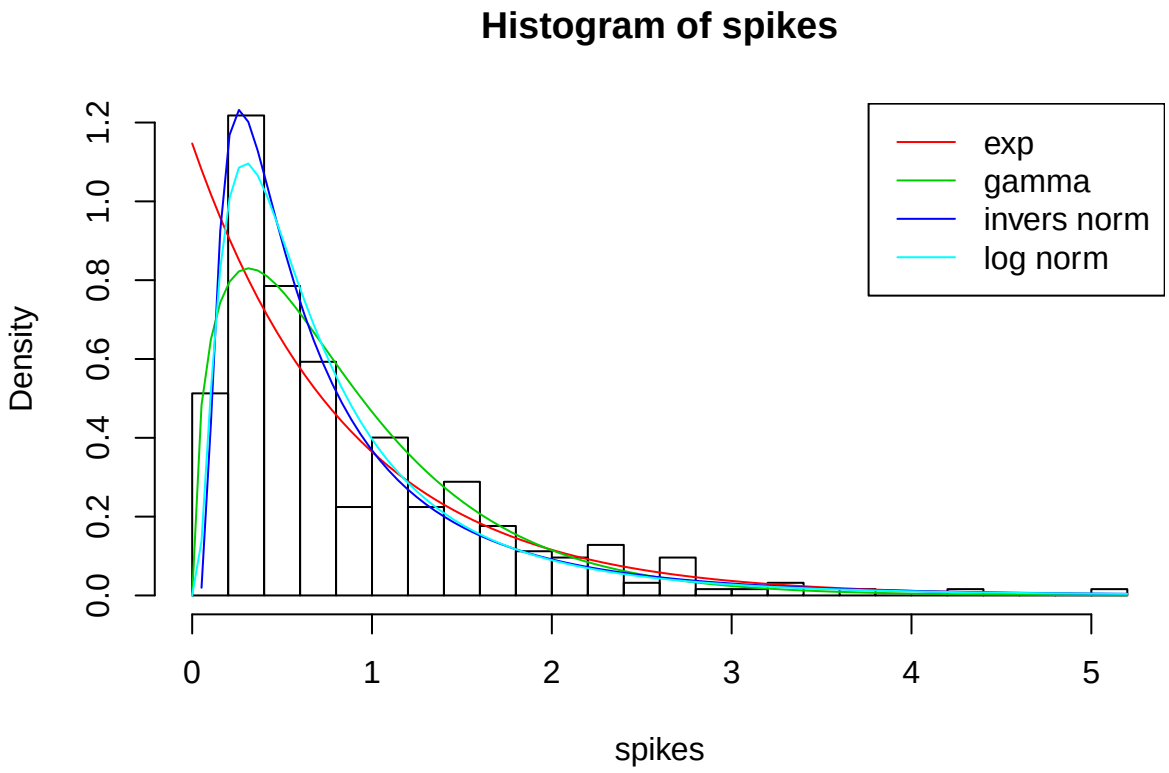
```
mll_lnorm <- function(par, xvals){
  return(-sum(dlnorm(xvals, meanlog = par[1], sdlog = par[2], log = TRUE)))
}
lnorm_ml <- optim(par = c(1, 1), fn = mll_lnorm, xvals = spikes)
lnorm_ml_par <- lnorm_ml$par
lnorm_ml_value <- -lnorm_ml$value
```

```
lnorm_ml_value
```

```
## [1] -240.3783
```

We now plot the densities of the models on top of the data.

```
hist(spikes, freq = FALSE, breaks = 30, ylim = c(0,1.2))    # frequencies are F
curve(dexp                                      # because for freq they intend counts,
      (x, rate = exp_mle_par), col = 2, add = TRUE)     # we need densities
curve(dgamma
      (x, shape = gamma_mle_par[1], rate = gamma_mle_par[2]), col = 3, add = TRUE)
curve(dinvnorm
      (x, mu = invnorm_mle$par[1], lambda = invnorm_mle$par[2]), col = 4, add = TRUE)
curve(dlnorm
      (x, meanlog = lnorm_ml_par[1], sdlog = lnorm_ml_par[2]), col = 5, add = TRUE)
legend("topright", legend = c("exp", "gamma", "invers norm", "log norm"),
       col = c(2,3,4,5), lty = 1)
```



Ex.3.2 Perform model selection using AIC and BIC.

```
k = 1
AIC_exp <- -2 * exp_mle_value + 2 * k
BIC_exp <- -2 * exp_mle_value + k * log(length(spikes))
AIC_exp
```

```
## [1] 540.4776
```

```
BIC_exp
```

```
## [1] 544.2206
```

```r
k = 2
AIC_gamma <- -2 * gamma_mle_value + 2 * k
BIC_gamma <- -2 * gamma_mle_value + k * log(length(spikes))
AIC_gamma
```

```
## [1] 509.4023
```

```r
BIC_gamma
```

```
## [1] 516.8883
```

```r
AIC_invnorm <- -2 * invnorm_mle_value + 2 * k
BIC_invnorm <- -2 * invnorm_mle_value + k * log(length(spikes))
AIC_invnorm
```

```
## [1] 474.957
```

```r
BIC_invnorm
```

```
## [1] 482.443
```

```r
AIC_lnorm <- -2 * lnorm_ml_value + 2 * k
BIC_lnorm <- -2 * lnorm_ml_value + k * log(length(spikes))
AIC_lnorm
```

```
## [1] 484.7566
```

```r
BIC_lnorm
```

```
## [1] 492.2426
```

We choose the model with the smallest AIC or BIC, the inverse normal distribution fit best our data.

Now we do the same using functions

First I need a list for each model with the loglikelihood value and parameters, then I need a list of all models

```r
exp <- list(
  value = exp_mle_value,
  par = exp_mle_par
)

gamma <- list(
  value = gamma_mle_value,
  par = gamma_mle_par
)

invnorm <- list(
  value = invnorm_mle_value,
  par = invnorm_mle_par
)
lognorm <- list(
  value = lnorm_ml_value,
  par = lnorm_ml_par
)

candidates <- list( exp = exp, gamma = gamma, invnorm = invnorm, lognorm = lognorm)
```

Now we can write the functions for AIC and BIC and apply them to the list of models (candidates)

```r
aic <- function(model){
  return(-2 * model$value + 2 * length(model$par))
}
bic <- function(model){
  return(-2* model$value + length(model$par) * log(length(spikes)))
}

scores <- data.frame(
  AIC = sapply(X = candidates, FUN = aic),
  BIC = sapply(candidates, bic)
)
scores
```

```
##              AIC      BIC
## exp     540.4776 544.2206
## gamma   509.4023 516.8883
## invnorm 474.9570 482.4430
## lognorm 484.7566 492.2426
```

Ex.4

Ex 4.1 Estimate the standard error of the MLE estimator of μ for the log_normal distribution applied to the ramp spike time data.

```r
ramp_spikes <- read.csv("cell_types.csv", na.strings = "")
ramp_spikes <- ramp_spikes$ef__peak_t_ramp
ramp_spikes <- ramp_spikes[!is.na(ramp_spikes)]
```

Mll log_normal distribution and MLE:

```r
mll_lnormal <- function(par, xvals){
  return(-sum(dlnorm(xvals, meanlog = par[1], sdlog = par[2], log = TRUE)))
}
mu_lnorm_est <-  optim(f = mll_lnormal, par = c(1, 1), xvals = ramp_spikes)$par[1]
mu_lnorm_est
```

```
## [1] 1.668835
```

Calculate $SE(\hat{\log \mu})$ (Gherardo obtains it also analytically, without Bootstrapping) :

```r
vect_lnorm_par_est_bt <- replicate(1000, expr = {
  ramp_spikes_bt <- sample(ramp_spikes, size = length(ramp_spikes), replace = TRUE)
  optim(par = c(1, 1), f = mll_lnormal, xvals = ramp_spikes_bt)$par
})

se_mu_lnorm_est <- sd(vect_lnorm_par_est_bt[1,])
se_mu_lnorm_est
```

```
## [1] 0.01266919
```

Ex 4.2 Obtain a 95% confidence interval for the parameter μ (try the different methods).

To obtain a 95 % confidence interval we can use two different methods.

```
alpha <- 0.05
```

1° method: we can build an (asymptotically) confidence interval using the asymptotic normality of MLE estimators.

```
z <- qnorm(1 - alpha/2)
a <- mu_lnorm_est - se_mu_lnorm_est * z
b <- mu_lnorm_est + se_mu_lnorm_est * z
paste("95% Confidence Interval for k: ( a =", a, ", b= ", b, ")")
```

```
## [1] "95% Confidence Interval for k: ( a = 1.6440041771239 , b=  1.69366647566306 )"
```

We can also use the bootstrap standard error in the above code.

Or

2° method (percentile confidence intervals): we can build the percentile confidence interval using the sample of the estimator obtained from bootstrap.

```
quantile(vect_lnorm_par_est_bt[1,], probs = c(alpha/2, 1 - alpha / 2))
```

```
##     2.5%    97.5%
## 1.644585 1.695100
```

Ex 4.3 Obtain again a 95% confidence interval for the parameter μ using only the human cells.

```
ramp_spikes <- read.csv("cell_types.csv")# It's faster to reload the data than change all the names
m <- ramp_spikes$donor__species == "Homo Sapiens"
ramp_spikes_human <- na.omit(ramp_spikes$ef__peak_t_ramp[m])
sum(is.na(ramp_spikes_human))
```

```
## [1] 0
```

Find MLE for human cells:

```
mu_lnorm_est_human <-  optim(f = mll_lnormal, par = c(1, 1), xvals = ramp_spikes_human)$par[1]
```

Find SEM:

```
vect_lnorm_par_est_bt_human <- replicate(1000, expr = {
  ramp_spikes_bt <- sample(ramp_spikes_human, size = length(ramp_spikes_human), replace = TRUE)
  optim(par = c(1, 1), f = mll_lnormal, xvals = ramp_spikes_bt)$par
})

se_mu_lnorm_est_human <- sd(vect_lnorm_par_est_bt_human[1,])
se_mu_lnorm_est_human
```

```
## [1] 0.03475123
```

Find confidence interval:

```
alpha <- 0.05
z <- qnorm(1 - alpha/2)
```

```r
a <- mu_lnorm_est_human - se_mu_lnorm_est_human * z
b <- mu_lnorm_est_human + se_mu_lnorm_est_human * z
paste("95% Confidence Interval for k: ( a =", a, ", b= ", b, ")")
```

```
## [1] "95% Confidence Interval for k: ( a = 1.78418540726554 , b=  1.92040770635802 )"
```

2° method (percentile confidence intervals):

```r
quantile(vect_lnorm_par_est_bt_human[1,], probs = c(alpha/2, 1 - alpha / 2))
```

```
##     2.5%    97.5%
## 1.789414 1.918158
```

Ex 5.

Ex 5.1 Transform the ramp spike time using the logarithm as we did in Week 3 and then perform a two sample t_test between the human and mouse cells.

```r
m = ramp_spikes$donor__species == "Mus musculus"
ramp_spikes_mouse <- na.omit(ramp_spikes$ef__peak_t_ramp[m])
sum(is.na(ramp_spikes_mouse))
```

```
## [1] 0
```

```r
ramp_spikes_human_log <- log(ramp_spikes_human)
ramp_spikes_mouse_log <- log(ramp_spikes_mouse)

# We want now to perform a test to compare the two mean value of the logarithm of the ramp spike time.

t.test(ramp_spikes_human_log, ramp_spikes_mouse_log)
```

```
##
##  Welch Two Sample t-test
##
## data:  ramp_spikes_human_log and ramp_spikes_mouse_log
## t = 5.9063, df = 529.61, p-value = 6.26e-09
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1495649 0.2986374
## sample estimates:
## mean of x mean of y
##  1.852376  1.628275
```

$H_0 : \mu_h = \mu_m$

$H_1 : \mu_h \neq \mu_m$

p_value $< \alpha$, we reject H_0.

We think that the the 2 means are significantly different with a confidence of 95% (actually at a level 0.001 from Gherardo solution).

Basic t_test (var.equal = T) works only with samples that have homogeneous variance but by default the var.equal is set to FALSE and so we can use it with sample that have different variance. We can also perform the classical t-test with equal variance.

```r
t.test(ramp_spikes_human_log, ramp_spikes_mouse_log, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  ramp_spikes_human_log and ramp_spikes_mouse_log
## t = 6.8632, df = 2271, p-value = 8.658e-12
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1600687 0.2881335
## sample estimates:
## mean of x mean of y
##  1.852376  1.628275
```

In both case the p-values are very small and we thus reject the null hypothesis (e.g. at a level 0.001). We can thus say that there is a difference in the average value of the ramp spike time in human cells and mouse cells (in this particular study).

Da farsi spiegare il 5.2!!!!!

Ex 5.2 Perform directly a Wald test to check if μh = μm where μh and μm are the mean_log parameters of the log_normal distributions for the human and mouse cells.

$$w = \frac{|\bar{x}_h - \bar{x}_m|}{\sqrt{\frac{s_h^2}{n_h} + \frac{s_m^2}{n_m}}}$$

```r
alpha <- 0.05
Zquantile <- qnorm(1 - alpha/2)

w <- abs(mean(ramp_spikes_human_log) - mean(ramp_spikes_mouse_log)) / (
  sqrt(
    (var(ramp_spikes_human_log)^2 / length(ramp_spikes_human_log)) +
      (var(ramp_spikes_mouse_log)^2/length(ramp_spikes_mouse_log))
    )
  )
w <= Zquantile                # Ho: mean_human = mean_mouse, since Z (or W) > Z, I reject Ho.
```

```
## [1] FALSE
```

```r
p_value = 2 * pnorm(-abs(w))
p_value
```

```
## [1] 5.596015e-17
```

```r
p_value > alpha               # p-value < alpha, so I reject the Ho
```

```
## [1] FALSE
```

$H_0 : \bar{x}_h = \bar{x}_m$

Since $w > Z$ and also, p_value $\leq \alpha$, we reject H_0.

Repeat 5.2 from Gherardo solution

Various calculation to obtain the stadard error of the difference of the means

Since we assume that $\mathbb{V}(X_i) = \mathbb{V}(Y_j) = \sigma^2$, we can write

$$se(\delta) = \sigma\sqrt{\frac{m+n}{mn}}$$

and an estimator of $se(\delta)$ is

$$\hat{se}(\delta) = \hat{\sigma}\sqrt{\frac{m+n}{mn}}$$

where $\hat{\sigma}$ is the empirical standard deviation of the joined sample.

```
n <- length(ramp_spikes_mouse_log) ## mouse sample size
m <- length(ramp_spikes_human_log) ## human sample size
sigma_est <- sd(c(ramp_spikes_mouse_log, ramp_spikes_human_log))## empirical sd joined sample
se_delta_est <- sigma_est * sqrt((n + m)/ (n * m))
```

Calculate delta and the p-value

$$\text{p-value } = 1 - 2F_Z(|\delta|/\hat{se}(\delta)) = 2F_Z(-|\delta|/\hat{se}(\delta))$$

```
delta <- mean(ramp_spikes_mouse_log) - mean(ramp_spikes_human_log)
pvalue <- 1 - 2 * pnorm( abs(delta) / se_delta_est )
pvalue
```

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

```
pvalue <- 2 * pnorm( - abs(delta) / se_delta_est )  # why it's not the same? use this one
pvalue
```

```
## [1] 1.086408e-11
```

Also the Wald test obtain a very small p-value and thus a similar result to the t-test.

Ex 6.

Ex 6.1 Simulate two groups of i.i.d. data following two normal distributions.

```
m = 20
n = 40
mu1 = 2
mu2 = 2.5
sigma = 4
x = rnorm(m, mean = mu1, sd = sigma)
y = rnorm(n, mean = mu2, sd = sigma)
```

Ex 6.2 Compute the p_value of the two_sample t_test with equal variance.

T-Test

```
t <- t.test(x, y, var.equal = TRUE)
t$p.value
```

```
## [1] 0.2210306
```

if p_value $> \alpha$, (?? where is alpha here?? 0.05 by default?) we can't reject the $H_0 : m1 = m2$, but the result of the p-value in this case change from one sample generated to another.

Ex 6.3

Compute the Wald test for $H_0 : m1 = m2$

Wald-Test

```
alpha <- 0.05
delta <- mean(x) - mean(y)

Z <- qnorm(1 - alpha/2)

w <- abs(delta / (
  sqrt( (var(x)^2 / length(x)) + (var(y)^2/length(y)) ) )  # <- ERROR!!
)
w <= Z                              # the SE of delta found in this way it's not correct
```

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

```
p_value_Wtest = 2 * pnorm(-abs(w))
p_value_Wtest
```

```
## [1] 0.8115566
```

The Wald t-test use the $\delta = \bar{X} - \bar{Y}$ statistic. Where $se(\delta) = \sigma\sqrt{\frac{m+n}{mn}}$

Estimate SE of delta analytically (from the formula above obtained in 5.2)

```
delta <- mean(x) - mean(y)
se_delta <- sigma * sqrt( (m+n) / (m*n))
w <- list(
  w = delta / se_delta,
  p.value = 2 * pnorm(-abs(delta) / se_delta)
)
w$p.value
```

```
## [1] 0.2098371
```

Estimate SE of delta by bootstraping

```
vect_norm_meandiff_bt <- replicate(10000, expr = {
  x_bt <- sample(x, size = length(x), replace = TRUE)
  y_bt <- sample(y, size = length(y), replace = TRUE)
  mean(x_bt)-mean(y_bt)
```

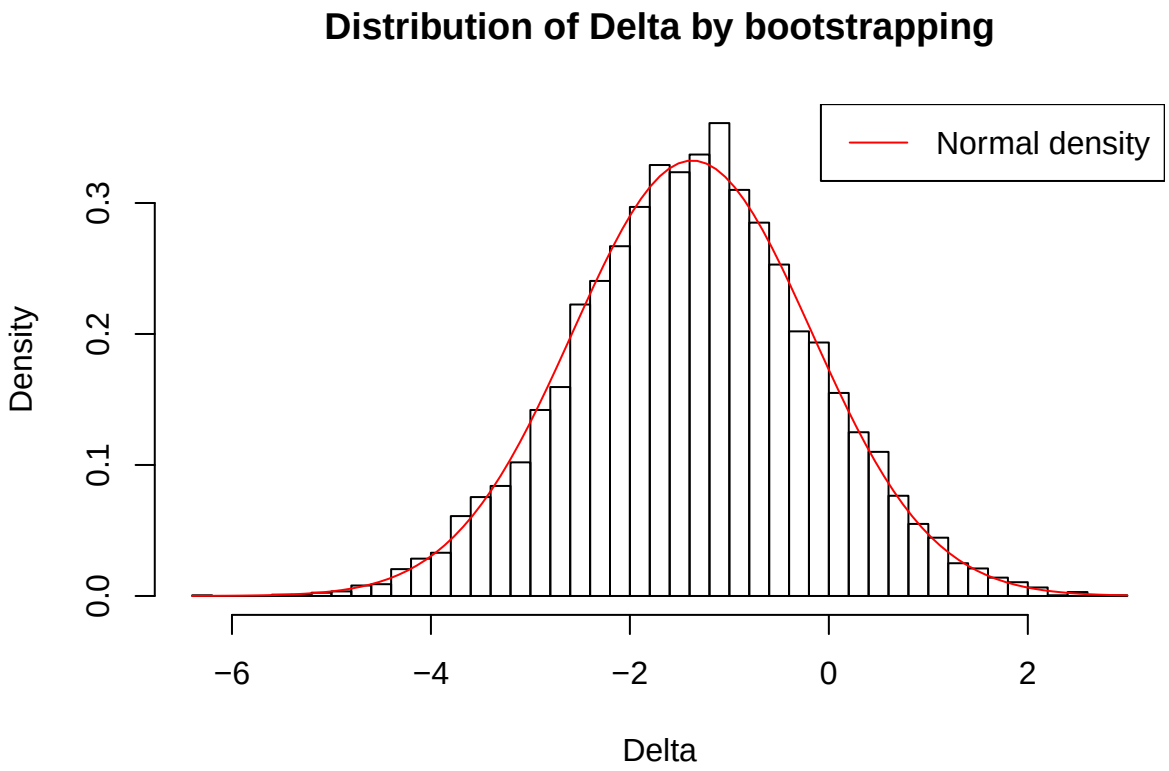
```
})
se_delta_bt <- sd(vect_norm_meandiff_bt)

p_value.bt <- 2 * pnorm( - abs(delta) / se_delta_bt)
p_value.bt
```

```
## [1] 0.2523527
```

Prove that the mean difference is Gaussian distributed

```
hist(vect_norm_meandiff_bt, freq = FALSE, breaks = 50,
     main = "Distribution of Delta by bootstrapping",xlab = "Delta")
curve(dnorm(x, mean = delta, sd = se_delta_bt), add = TRUE, col = "red")
legend("topright", legend = "Normal density", col = "red", lty = 1)
```



It is visually possible to observe that $\delta = \bar{X} - \bar{Y}$ is Gaussian distributed.

Ex 6.4

Perform the likelihood ratio test for $H_0 : m1 = m2$.

LRT-Test

```
mu_est <- mean(c(x,y))
l1 <- sum(dnorm(c(x,y), mean = mu_est, sd = sigma, log = TRUE)) # nested model
l2 <- sum(dnorm(x, mean = mean(x), sd = sigma, log = TRUE)) +   # larger model
      sum(dnorm(y, mean = mean(y), sd = sigma, log = TRUE))
```

```r
lambda <- 2*(l2 - l1)
LRT <- list(
  lambda = lambda,
  p.value = 1 - pchisq(lambda, df = 1)
)
LRT$p.value
```

```
## [1] 0.2098371
```

Ex 6.5 Compare the results obtained in the different tests, in particular report the different p_values.

```r
tests <- list(
  wald = w,
  t.test = t,
  LRT = LRT
)
sapply(tests, function(x) return(x$p.value))
```

```
##      wald    t.test       LRT
## 0.2098371 0.2210306 0.2098371
```

For all the test performed the p-value is always larger than α and so we don't reject the H_0.

Now we can try with more sample size.

```r
m = 2000
n = 4000
mu1 = 2
mu2= 2.5
sigma = 4
x = rnorm(m, mu1, sigma)
y = rnorm(n, mu2, sigma)
```

1) T-Test

```r
t <- t.test(x, y, var.queal = TRUE)
t$p.value
```

```
## [1] 3.545244e-10
```

2) Wald-Test

Estimate SE_delta by bootstrap

```r
delta_bt <- replicate(10000, expr = {
  x_bt <- sample(x, size = length(x), replace = TRUE)
  y_bt <- sample(y, size = length(y), replace = TRUE)
  mean(x_bt) - mean(y_bt)
```

```
})
se_delta_bt <- sd(delta_bt)
```

Calculate w and the p-value

```
delta = mean(x) - mean(y)
w <- list(
  w = delta / se_delta_bt,
  p.value = 2 * pnorm( -abs(delta) / se_delta_bt)
)
w$p.value
```

```
## [1] 3.268148e-10
```

3) LRT-Test

```
mu_est <- mean(c(x,y))
l1 <- sum(dnorm(c(x,y), mean = mu_est, sd = sigma, log = TRUE))
l2 <- sum(dnorm(x, mean = mu1, sd = sigma, log = TRUE)) +
      sum(dnorm(y, mean = mu2, sd = sigma, log = TRUE))
lambda <- 2 * (l2 - l1)
LRT <- list(
  lambda = lambda,
  p.value = 1 - pchisq(lambda, df = 1)
)
LRT$p.value
```

```
## [1] 2.029555e-09
```

Conclusions

```
tests <- list(
  t.test = t,
  wald = w,
  LRT = LRT
)
sapply(tests, function(x) return(x$p.value))
```

```
##       t.test         wald          LRT
## 3.545244e-10 3.268148e-10 2.029555e-09
```

With larger sample size the p-values from all tests are really low, thus we reject Ho. It become evident that $\mu_1 \neq \mu_2$