Week 4

gherardo varando

A trigonometric density

Ex 1

Source the file with the functions needed and load the data.

```
source("f_sink.R")

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

Recall from week 3 how to find MLE:

```
mll <- function(k, data){
  sum( -log(dsin(x = data, k = k )))
}
k_est <- optimize(mll, interval = c(0,100), data = angles)$minimum
```

1.1

To build a confidence interval we need first to obtain an estimate of the standard error of $\hat{k}$. We use bootstrap (it will take a while).

```
M <- 1000
v_k_bt <- replicate(M, {
  S_bt <- sample(x = angles, size = length(angles), replace = T)
  optimize(mll, interval = c(0,20), data = S_bt)$minimum
})
```

Now we can estimate the standard error.

```
se_est <- sd(v_k_bt)
se_est
```

```
## [1] 0.5439325
```

and we can compute a $0.99 = 1 - (0.01)$ confidence interval:

```
alpha <- 0.01
z <- qnorm(1 - alpha / 2)
a <- k_est - se_est * z
b <- k_est + se_est * z
c(a, b)
```

```
## [1]  9.999316 12.801471
```

We can also obtain the percentile confidence interval,

```r
quantile(v_k_bt, c(alpha/2, 1- alpha/2))
```

```
##     0.5%    99.5%
## 10.19346 12.89437
```

We can also use parametric bootstrap (but it is very slow in this case, we just use 10 bootstrap replicate).

```r
M <- 10 ##we should increase this
v_k_bt <- replicate(M, {
  S_bt <- rsin(length(angles), k = k_est) ## we just change this line
  optimize(mll, interval = c(0,20), data = S_bt)$minimum
})
se_est_p <- sd(v_k_bt)
a <- k_est - se_est_p * z
b <- k_est + se_est_p * z
c(a, b)
```

```
## [1] 10.31386 12.48692
```

1.2

We can perform a one-sided Wald test, using the Wald statistic,

$$W = \frac{\hat{k} - k_0}{\hat{se}(\hat{k})}$$

Which is extreme for $H_0 : k \leq 10$ if $W >> 0$ (is large).

In particular an asymptotically test at level α is given by: reject H_0 if $W > z_\alpha$.

```r
k0 <- 10
alpha <- 0.05
z <- qnorm(1 - alpha)
### using the standard error estimated with bootstrap
### in the previous point
w <- (k_est - k0)/ (se_est)
w > z
```

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

So we reject the null hypothesis $k \leq 10$ at a confidence level of $\alpha = 0.05$.

The approximate p-value is,

```r
pval <- 1 - pnorm(w)
pval
```

```
## [1] 0.00501821
```

A case study on neuronal data

We load the data

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

Ex 2

We recall from exercises week 3 the MLE for the exponential

```
rate_est <- 1 / mean(isi)
```

And for the gamma distribution (computed numerically),

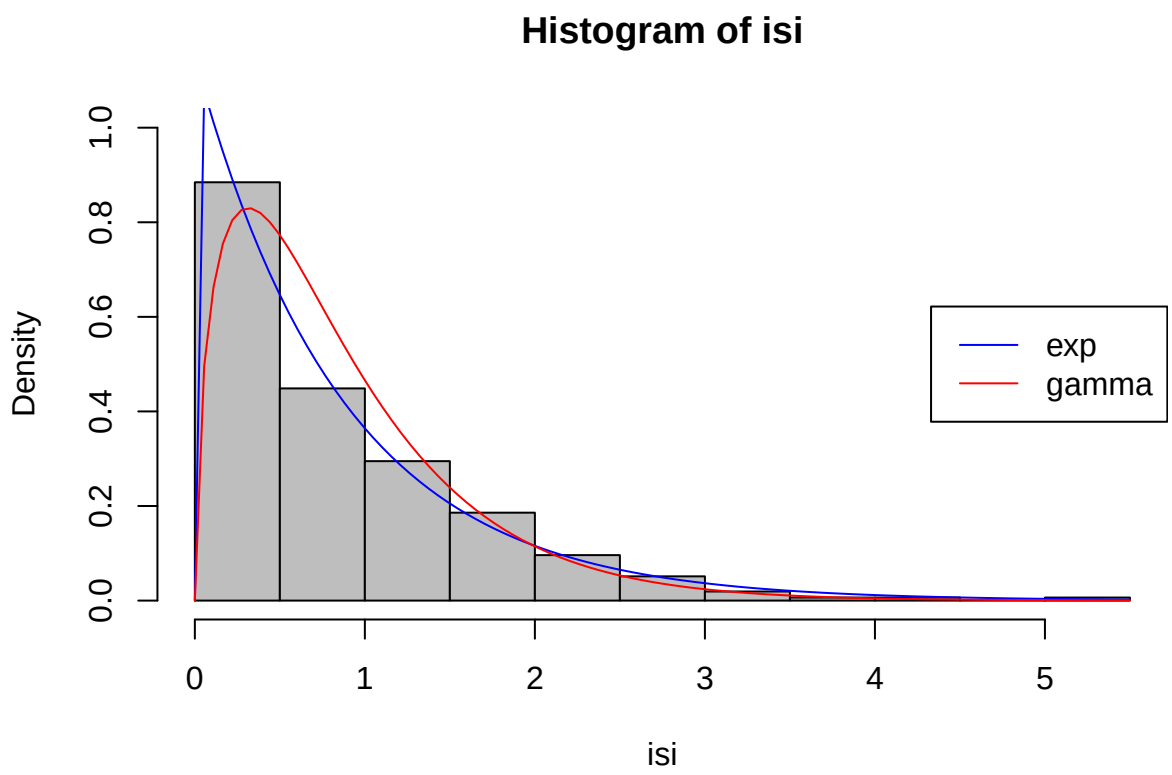
```
mllg <- function(pars, data){
  -sum(dgamma(data, shape = pars[1], rate = pars[2], log = TRUE))
}

### we use the method of moments to initialize the optimization
shape_mmest <- mean(isi)^2 / var(isi)  ##we just use the var function
rate_mmest <-  mean(isi) / var(isi)

res_optim <- optim(par = c(shape_mmest,rate_mmest),
                   fn = mllg, data = isi)
shape_g_est <- res_optim$par[1]
rate_g_est <-  res_optim$par[2]
```

The two densities are the following

```
hist(isi, freq = FALSE, col = "gray", ylim = c(0,1))
curve(dexp(x, rate = rate_est), col = "blue", add = TRUE)
curve(dgamma(x, shape = shape_g_est, rate = rate_g_est), col = "red", add = TRUE)
legend("right", c("exp", "gamma"), col = c("blue", "red"), lty = 1)
```



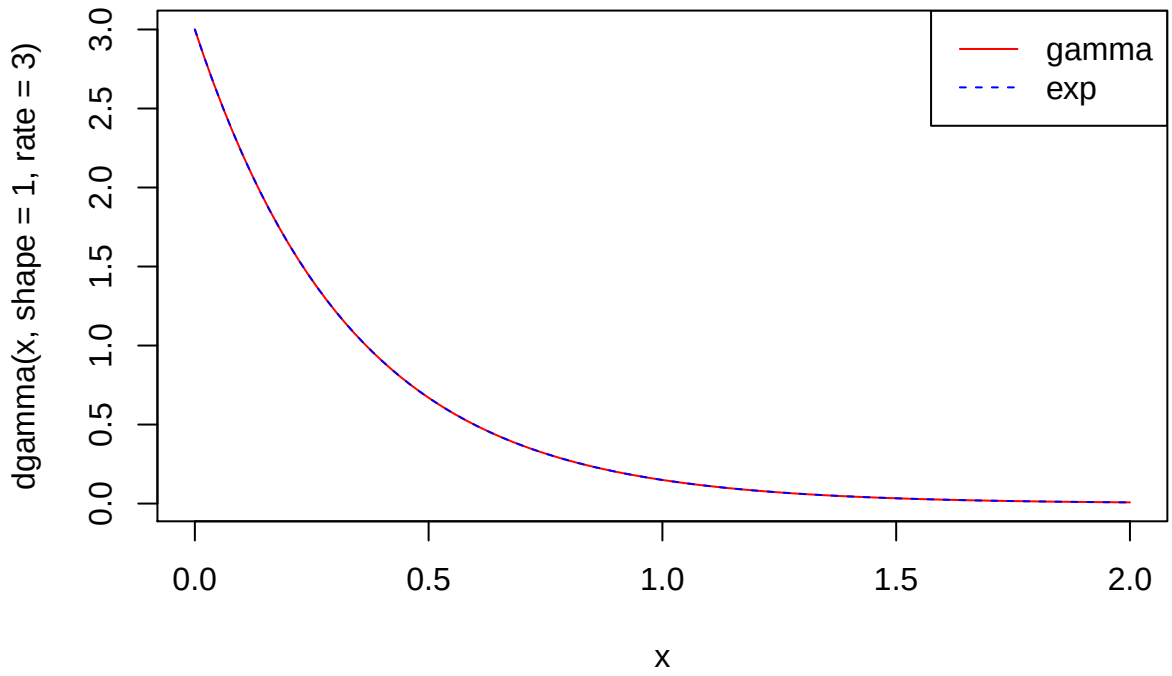
2.1

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 $\lambda = \beta$).

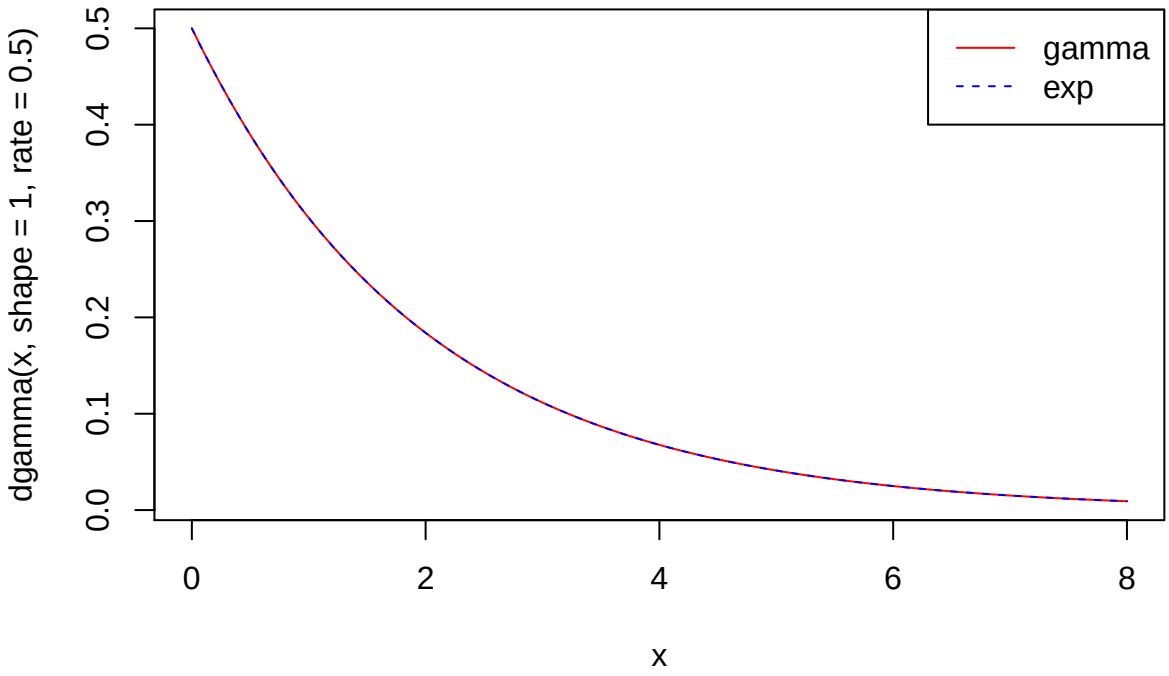
We check this fact graphically,

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



and for another value of the rate,

```
curve(dgamma(x , shape = 1, rate = 0.5), col = "red", from = 0, to = 8)
curve(dexp(x, rate = 0.5), col = "blue", lty = 8, add = TRUE)
legend("topright", legend = c("gamma", "exp"), col = c("red", "blue"),
       lty = c(1,2))
```



2.2

We perform now the likelihood ratio test.

We denote with $\mathcal{L}(\alpha, \beta)$ the likelihood of the gamma model. If $\mathcal{L}_1 = \max_{\alpha=1,\beta>0} \mathcal{L}(\alpha, \beta)$ and $\mathcal{L}_2 = \max_{\alpha,\beta>0} \mathcal{L}(\alpha, \beta)$, we have that

$$\mathcal{L}_1 = \mathcal{L}(1, 1/\overline{X})$$

where $\frac{1}{\overline{X}}$ is the MLE of the rate parameter for the exponential distribution. And

$$\mathcal{L}_2 = \mathcal{L}(\hat{\alpha}, \hat{\beta})$$

where $\hat{\alpha}, \hat{\beta}$ are the MLE estimators of the shape and rate parameter for the gamma model.

To perform the likelihood ratio test we need to compute the statistic:

$$\lambda(X_1, \ldots, X_n) = -2 \log \left(\frac{\mathcal{L}_1}{\mathcal{L}_2} \right)$$

That is asymptotically distributed as a $\chi^2_{d-d_0}$, a chi-squared distribution with $d - d_0$ degree of freedom. In our simple case we have that $d = 2$ (the number of parameters of the gamma model) and $d_0 = 1$ (the number of parameters of the
exponential model).

BE CAREFUL HERE λ IS NOT THE PARAMETER OF THE EXPONENTIAL BUT THE STATISTIC OF THE LIKELIHOOD RATIO TEST

We can rewrite the λ statistic as,

$$\lambda(X_1, \ldots, X_n) = -2(\log(\mathcal{L}_1) - \log(\mathcal{L}_2)) = 2(\ell_2 - \ell_1)$$

Where ℓ_1 is the maximum value of the log-likelihood for the exponential and ℓ_2 is the maximum value of the log-likelihood for the gamma model.

To compute ℓ_1 in R we just compute the log-likelihood for the MLE parameter,

```
l1 <- sum(dexp(isi, rate = rate_est, log = TRUE))
```

To compute ell_2 we can do the same with the log-likelihood of the gamma,

```
l2 <- sum(dgamma(isi, shape = shape_g_est, rate = rate_g_est,
                 log = TRUE))
l2
```

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

Or we can obtain it from the object, result of the optimization function (the `value` field), be careful that the optimization is on the minus log-likelihood (we need then to change sign),

```
l2 <- - res_optim$value
l2
```

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

As you can see the two numbers are the same (otherwise we now that we made some mistake in the code).

We can now obtain the value of the λ statistic, we check that it is a positive number (as the theory says)

```
lambda_chisq <- 2*(l2 - l1)
lambda_chisq
```

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

To compute the p-value of the likelihood ratio test we need to compute $1 - F_{\chi^2_{d-d_0}}(\lambda(X_1, \ldots, X_n))$ (in our case $d - d_0 = 2 - 1 = 1$)

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

```
## [1] 8.865963e-09
```

In this case we reject the null hypothesis $H_0 : \alpha = 1$ at a level as small as the p-value.

For example we reject the null hypothesis at a level 0.001.

We can thus state that it is preferable to use the gamma model to describe the data.

Ex 3

We consider now a total of four candidates models for the ISI data: exponential, gamma, inverse Gaussian and log-normal.

3.1

We find the MLE for each one of them (just a recap of previous week). We store each model in a list containing the MLE parameters and the log-likelihood.

The exponential model:

```
exponential <- list(
  density = function(x, par){ dexp(x, rate = par)},
  par = 1 / mean(isi),
  ll = sum(dexp(isi, rate = 1 / mean(isi), log = TRUE)),
  data = isi,
  col = "blue"
)
```

The gamma model (we use the values computed in the previous exercise):

```
gamma_model <- list(
  density = function(x, par){ dgamma(x, shape = par[1], rate = par[2])},
  par = c(shape_g_est, rate_g_est),
  ll = sum(dgamma(isi, shape = shape_g_est, rate = rate_g_est,
                  log = TRUE)),
  data = isi,
  col = "red"
)
```

Inverse Gaussian (from week 3),

```
mu_est <- mean(isi)
lambda_est <-1 /( mean(1/isi) - 1/mean(isi))
dinvgauss <- function(x, mu = 1, lambda = 1){
  sqrt(lambda/(2*pi*x^3))*exp( -(lambda* (x-mu)^2)/ (2*mu^2*x) )
}

inv_gaussian <- list(
  density = function(x, par){ dinvgauss(x, par[1], par[2])},
  par = c(mu_est, lambda_est),
  ll = sum(log(dinvgauss(isi, mu = mu_est, lambda = lambda_est))),
  data = isi,
  col = "green"
)
```

log-normal,

```
log_normal <- list(
  density = function(x, par){ dlnorm(x, par[1], par[2])},
  par = c(mean(log(isi)), sd(log(isi))),
  ll = sum(dlnorm(isi, meanlog = mean(log(isi)),
                  sdlog =  sd(log(isi)), log = TRUE)),
  data = isi,
  col = "orange"
)
```

We can store all the models in a list,

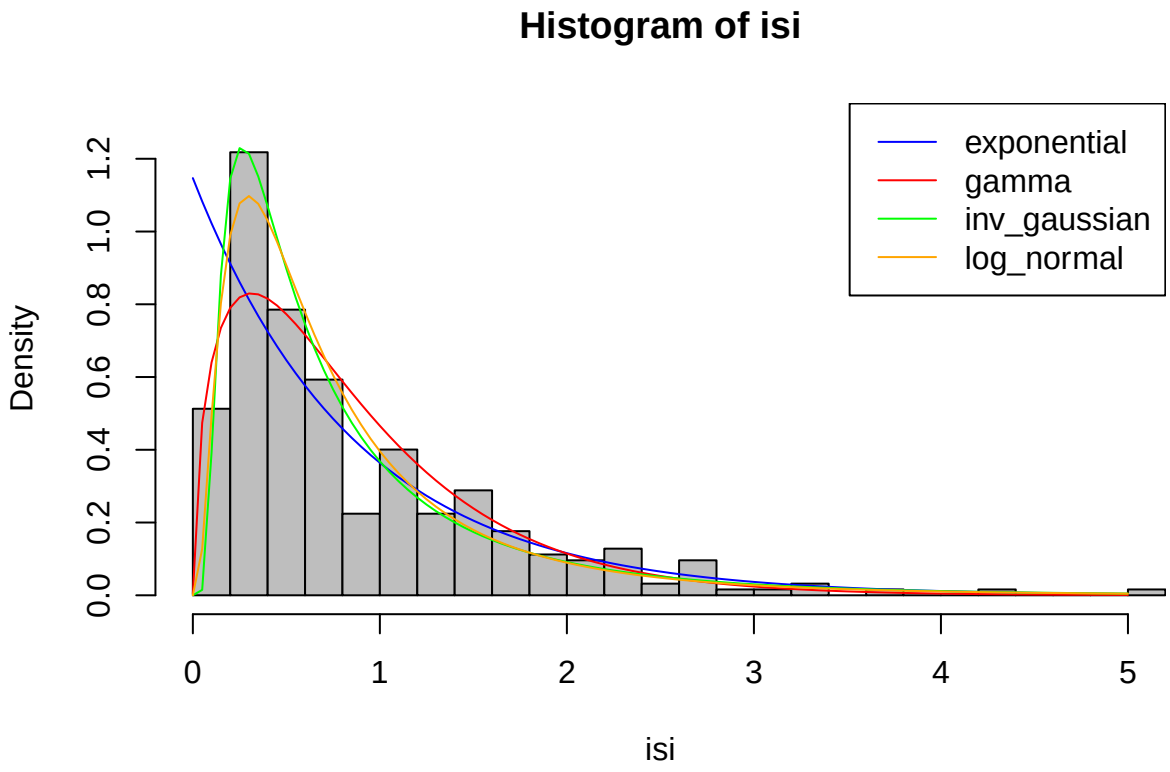
```
candidates <- list(
  exponential = exponential,
  gamma = gamma_model,
  inv_gaussian = inv_gaussian,
  log_normal = log_normal
)
```

We can also make a plot easily now,

```r
names <- names(candidates)
colors <- sapply(candidates, function(m){ m$col})
hist(isi, col = "gray", freq = FALSE, ylim = c(0,1.3), xlim = c(0.00001, 5), breaks = "FD")
sapply(candidates, function(model){
  curve(model$density(x, model$par), add = TRUE, col = model$col)
})
```

```
##   exponential gamma       inv_gaussian log_normal
## x Numeric,101 Numeric,101 Numeric,101  Numeric,101
## y Numeric,101 Numeric,101 Numeric,101  Numeric,101
```

```r
legend("topright", legend = names, col = colors, lty = 1)
```



3.2

We can now perform model selection using AIC or BIC.

We write again the formulas for AIC and BIC,

$$AIC = -2\log(\mathcal{L}) + 2k$$
$$BIC = -2\log(\mathcal{L}) + k\log(n)$$

where k is the dimension of the parameter space and n is the sample size.

We can write two functions that given a model (stored in a list as the models above) compute AIC and BIC score.

```r
aic <- function(model){
  -2 * model$ll + 2 * length(model$par)
}

bic <- function(model){
  - 2 * model$ll + length(model$par) * log(length(model$data))
}

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

scores
```

```
##                   AIC      BIC
## exponential  540.4776 544.2206
## gamma        509.4023 516.8883
## inv_gaussian 474.9570 482.4430
## log_normal   484.7582 492.2442
```

The inverse Gaussian model is the model that obtain the smallest BIC and AIC scores.

```r
names(candidates[which.min(scores$AIC)])
```

```
## [1] "inv_gaussian"
```

```r
names(candidates[which.min(scores$BIC)])
```

```
## [1] "inv_gaussian"
```

Brain cell dataset

We load the data

```r
cells <- read.csv("cell_types.csv", na.strings = "")
rst <- cells$ef__peak_t_ramp
```

Ex 4

To fit the log-normal distribution we can transform the data with the logarithm and then compute the MLE for the Gaussian model.

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \log(X_i)$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} (\log(X_i) - \hat{\mu})^2$$

```r
log_rst <- log(rst) ## works also for NA, log(NA) = NA
n <- sum(!is.na(rst)) ## how many non NA observations
log_mean <- mean(log_rst, na.rm = TRUE)
log_sd <- sqrt( (n-1) * var(log_rst, na.rm = TRUE) / n )
```

4.1

We can obtain an estimator of the standard error for the log-normal distribution analytically, since

$$X \sim logNormal(\mu, \sigma^2) \Leftrightarrow \log(X) \sim N(\mu, \sigma^2)$$

$$sd(\hat{\mu}) = sd\left(\frac{1}{n}\sum_{i=1}^{n}\log(X_i)\right) = \sqrt{\frac{1}{n}\mathbb{V}(\log(X_i))} = \frac{\sigma}{\sqrt{n}}$$

Thereafter an estimate of the standard error of $\hat{\mu}$ is

$$\hat{se}(\hat{\mu}) = \frac{\hat{\sigma}}{\sqrt{n}}$$

```
 se_est <- log_sd / sqrt(n)
se_est
```

```
## [1] 0.01270309
```

Otherwise using bootstrap we obtain a similar result,

```
M <- 10000
v_mu_bt <- replicate(M, {
  mean(sample(na.omit(log_rst),size = n, replace = TRUE))
})
se_est_bt <- sd(v_mu_bt)
se_est_bt
```

```
## [1] 0.01295497
```

4.2

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

We can build an (asymptotically) confidence interval using the asymptotic normality of MLE estimators.

```
alpha <- 0.05
z <- qnorm(1 - alpha/2 ) ## alpha/2 upper quantile
a <- log_mean - se_est * z
b <- log_mean + se_est * z
c(a, b)
```

```
## [1] 1.643997 1.693793
```

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

Or we can build the percentile confidence interval using the sample of the estimator obtain from bootstrap.

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

```
##     2.5%    97.5%
## 1.643478 1.693675
```

As you can see the two confidence intervals are very similar.

4.3

To obtain a 95 % confidence interval just for the human cell we have to repeat the above computations just with human cells.

```
h <- cells$donor__species == "Homo Sapiens"
m <- cells$donor__species == "Mus musculus"
```

Ex 5

Since as state before,

$$X \sim logNormal(\mu, \sigma^2) \Leftrightarrow \log(X) \sim N(\mu, \sigma^2)$$

If we model our observations as a log-normal distribution, the logarithm of the observations is modeled with a Gaussian distribution and we can use the t-test to compare human and mouse observations.

5.1

We extract the data for human and mouse, and we remove NA

```
log_rst_h <- na.omit(log_rst[h])
log_rst_m <- na.omit(log_rst[m])
```

We can now fit Gaussian distributions to this two groups of observations

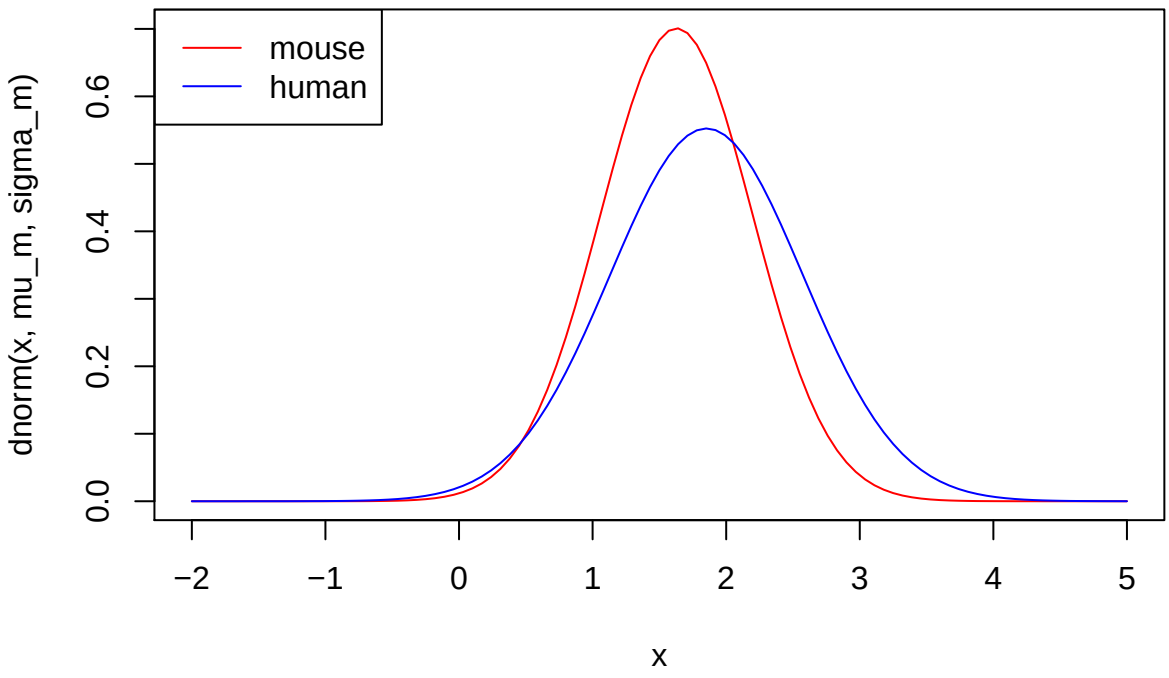
```
mu_h <- mean(log_rst_h)
sigma_h <- sd(log_rst_h)

mu_m <- mean(log_rst_m)
sigma_m <- sd(log_rst_m)
data.frame(mu = c(mu_h, mu_m), sigma = c(sigma_h, sigma_m), row.names = c("human", "mouse"))
```

```
##             mu     sigma
## human 1.852376 0.7220939
## mouse 1.628275 0.5691298
```

We can also plot the densities,

```
curve(dnorm(x , mu_m, sigma_m), from = -2, to = 5, col = "red")
curve(dnorm(x , mu_h, sigma_h), add = TRUE, col = "blue")
legend("topleft", legend = c("mouse", "human"), col = c("red", "blue"),
       lty = 1)
```



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

```
t.test(x = log_rst_m, y = log_rst_h)
```

```
##
##  Welch Two Sample t-test
##
## data:  log_rst_m and log_rst_h
## 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:
##  -0.2986374 -0.1495649
## sample estimates:
## mean of x mean of y
##  1.628275  1.852376
```

By default the test is performed without assuming equal variance in the two sample, which is correct in this case. We can also perform the classical t-test with equal variance.

```
t.test(x = log_rst_m, y = log_rst_h, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  log_rst_m and log_rst_h
## 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:
```

```
##  -0.2881335 -0.1600687
## sample estimates:
## mean of x mean of y
##  1.628275  1.852376
```

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).

5.2

We want now to perform the Wald test.

Let now $X_i, i = 1, \dots, n$ be the observations of the ramp spike time for the human cells and $Y_j, j = 1, \dots, m$. And suppose they follow log-normal distributions with same log-variance σ^2 and log-means μ_1, μ_2. We define the statistic

$$\delta = \frac{1}{n}\sum_{i=1}^{n}\log(X_i) - \frac{1}{m}\sum_{j=1}^{m}\log(Y_j)$$

Under the null hypothesis $H_0 : \mu_1 = \mu_2$ we have that

$$\frac{\delta}{se(\delta)} \sim N(0,1)$$

We need to obtain the standard error of δ that is, its standard deviation.

$$\begin{aligned}
se(\delta) = \sqrt{\mathbb{V}(\delta)} &= \sqrt{\mathbb{V}\left(\frac{1}{n}\sum_{i=1}^{n}\log(X_i) - \frac{1}{m}\sum_{j=1}^{m}\log(Y_j)\right)} \\
&= \sqrt{\mathbb{V}\left(\frac{1}{nm}\left(m\sum_{i=1}^{n}\log(X_i) - n\sum_{j=1}^{m}\log(Y_j)\right)\right)} \\
&= \sqrt{\frac{1}{n^2m^2}\mathbb{V}\left(m\sum_{i=1}^{n}\log(X_i) - n\sum_{j=1}^{m}\log(Y_j)\right)} \\
&= \sqrt{\frac{1}{n^2m^2}\left(m^2\mathbb{V}(\sum_{i=1}^{n}\log(X_i)) + n^2\mathbb{V}(\sum_{j=1}^{m}\log(Y_j))\right)} \\
&= \sqrt{\frac{1}{n^2m^2}\left(m^2n\mathbb{V}(\log(X_i)) + n^2m\mathbb{V}(\log(Y_j))\right)}
\end{aligned}$$

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

The Wald test consider the statistic $\frac{\delta}{\hat{se}(\delta)}$, that is extreme if $\frac{|\delta|}{\hat{se}(\delta)} >> 0$. And since it is asymptotically normal, the approximate p-value is computed as,

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

```
delta <- mean(log_rst_m) - mean(log_rst_h)
pvalue <- 2 * pnorm( - abs(delta) / se_delta_est )
pvalue
```

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

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

Different tests

We have two sets of observations

$$X_1, \ldots, X_m \sim N(\mu_1, \sigma^2)$$
$$Y_1, \ldots, Y_n \sim N(\mu_2, \sigma^2)$$

Exercise 6

6.1

We simulate the two groups of observations.

```
m <- 20
n <- 40
mu1 <- 2
mu2 <- 2.5
sigma <- 4
x <- rnorm(m, mu1, sigma)
y <- rnorm(n, mu2, sigma)
```

6.2

The p-value of the two-sample t-test with equal variance is

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

```
## [1] 0.6334053
```

6.3

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

```
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] 0.6702937
```

6.4

The likelihood ratio test for $H_0 : \mu_1 = \mu_2$

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

```
## [1] 0.6702937
```

6.5

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

```
##      wald    t.test       lrt
## 0.6702937 0.6334053 0.6702937
```

For all the test we obtain that we cannot reject the null hypothesis. We can try with more sample size.

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

```
)
t <- t.test(x, y, var.equal = TRUE)
mu_est <- mean(c(x,y))
l1 <- sum(dnorm(c(x,y), mean = mu_est, sd = sigma, log = TRUE))
l2 <- sum(dnorm(x, mean = mean(x), sd = sigma, log = TRUE)) +
      sum(dnorm(y, mean = mean(y), sd = sigma, log = TRUE))
lambda <- 2*(l2 - l1)
lrt <- list(
  lambda = lambda,
  p.value = 1-pchisq(lambda, df = 1)
)
tests <- list(
  wald = w,
  t.test = t,
  lrt = lrt
)
sapply(tests, function(test) return(test$p.value))
```

```
##         wald       t.test          lrt
## 0.0002613742 0.0003264926 0.0002613742
```