Week 8 Exercises

R-team

31 October, 2019

Exercises part 1

1.1 Throwing a ball

During the lecture we talked about the statistics of throwing a ball. The question was: given that I am asked to throw the ball 50 meters, what is the average distance I will actually throw the ball?

We discussed three parts to this 'equation': the angle at which I throw the ball, the power with which I throw the ball, and my precision in my estimate of how far I need to throw a ball. As discussed, probably there are many more factors that play into this, and also surely we can break down a concept such as 'precision' into many more factors that each play a role (e.g. my 'talent', my concentration, whether I'm wearing glasses or not). Instead of breaking these down, and refining our 'model' further, we'll just 'take our losses' and use randomness to model these unknown, underlying, factors. We do this by formulating assumptions, i.e. models defined as probability distributions.

Suppose the outcome ($D :=$ distance) of a throw is the result of the following function $D(x, y, z)$, defined as

$$D(x, y, z) = x \cdot (y - (z - \frac{1}{4})^2),$$

where x denotes the power, y denotes the precision, and z denotes the angle a. The probability density function of a throw would then by defined by $f_D(x, y, z)$, i.e. the join probability density function of x, $y\,z$.

In this assignment we assume that these three variables are independent of each other and have the following (marginal) continuous probability density functions:

- $f_{\text{power}}(x) = \frac{1}{\sqrt{(0.5pi)}} e^{-\frac{(x-5)^2}{0.5}}$
- $f_{\text{precision}}(y) = \frac{1}{11-9}$, for $y \in [9, 11]$ and 0 otherwise
- $f_{\text{angle}}(z) = \frac{\frac{z}{0.5\pi}^{10-1}(1-\frac{z}{0.5\pi})^{10-1}}{B(10,10)}$, where $B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$, and $\Gamma(n) = (n-1)!$, for $z \in [0, \frac{1}{2}\pi]$, and 0 otherwise.

Thus,

$$f_D(x, y, z) := f_{\text{power}}(x) \cdot f_{\text{precision}}(y) \cdot f_{\text{angle}}(z).$$

a

We talked about two ways to find the average distance that the ball is thrown: an analytical one, and the monte carlo one. Write down what steps you would need to take to find the analytical solution.

Answer: We need to find $E[D]$ (over all three random variables). This means:

$$\int_x \int_y \int_z D(x, y, z) f_D(x, y, z)\, dx\, dy\, dz$$

Since the variables are independent we can write:

$$\int_x \int_y \int_z D(x, y, z) f_{\text{power}}(x) f_{\text{precision}}(y) f_{\text{angle}}(z)\, dx\, dy\, dz$$

Then see whether you can solve this multiple integral analytically or whether you'll have to use numerical approximations (not a part of this course!).

b

Take a careful look at the density functions provided in the introduction. They look an aweful lot like well known probability density functions. Look online, or in your favourite textbook and see which probablity functions (and particular parameterization) resembles the density functions provided. **Hint: these distributions are part of the `base` set of distribution functions `R` provides you.**

Answer:

The first is a normal density with mean 50, and standard deviation $\frac{1}{2}$.

The second a uniform distribution with values between 9 and 11.

The third a beta distribution with parameters 10 and 10, but 'stretched' so that it's values fall in the interval $[0, \frac{1}{2}\pi]$, instead of in $[0, 1]$.

c

Now use a Monte Carlo study to provide an answer to the original question: given the model provided in the introduction, what would be a good estimate of the average throwing distance?

Act as if you throw the ball 10 times. Make sure you set a seed, but make sure you set a different seed compared to the student sitting next to you. Compare your answers, and check whether you used similar code? Would you claim one of the two averages is correct?

Hint: notice that for the beta distribution the domain is 'stretched' so that it's values fall in the interval $[0, \frac{1}{2}\pi]$*, instead of in* $[0, 1]$*. To stretch your values coming out of the* `rbeta()` *multiply the values by* $\pi/2$*.*

Answer:

```
set.seed(438734587)

ComputeD <- function(x, y, z){
  x * (y - (z - 1/4)^2)
}

N <- 10
power_mean <- 5
power_sd <- 0.5
angle_a <- 10
angle_b <- 10
precision_left <- 9
precision_right <- 11

power_samples <- rnorm(N, power_mean, power_sd)
angle_samples <- rbeta(N, angle_a, angle_b)*(pi/2)
precision_samples <- runif(N, precision_left, precision_right)

distance <- ComputeD(power_samples, precision_samples, angle_samples)
```

```r
mean(distance)
```

```
## [1] 46.93817
```

```r
# mimicing second student:
set.seed(12314454)

power_samples <- rnorm(N, power_mean, power_sd)
angle_samples <- rbeta(N, angle_a, angle_b)*(pi/2)
precision_samples <- runif(N, precision_left, precision_right)

distance_colleague <- ComputeD(power_samples, precision_samples, angle_samples)

mean(distance_colleague)
```

```
## [1] 49.40683
```

There's a difference, which is to be expected, given that we're sampling (probably) different values! Neither is correct: since the distribution function of the mean of samples of throwing distance is continuous the probability of getting the mean exactly correct is negligible.

d

Use `replicate` to repeat the experiment in **c** many times (e.g. a 1000 times). What is the variation of the average distance obtained in each experiment?

Answer:

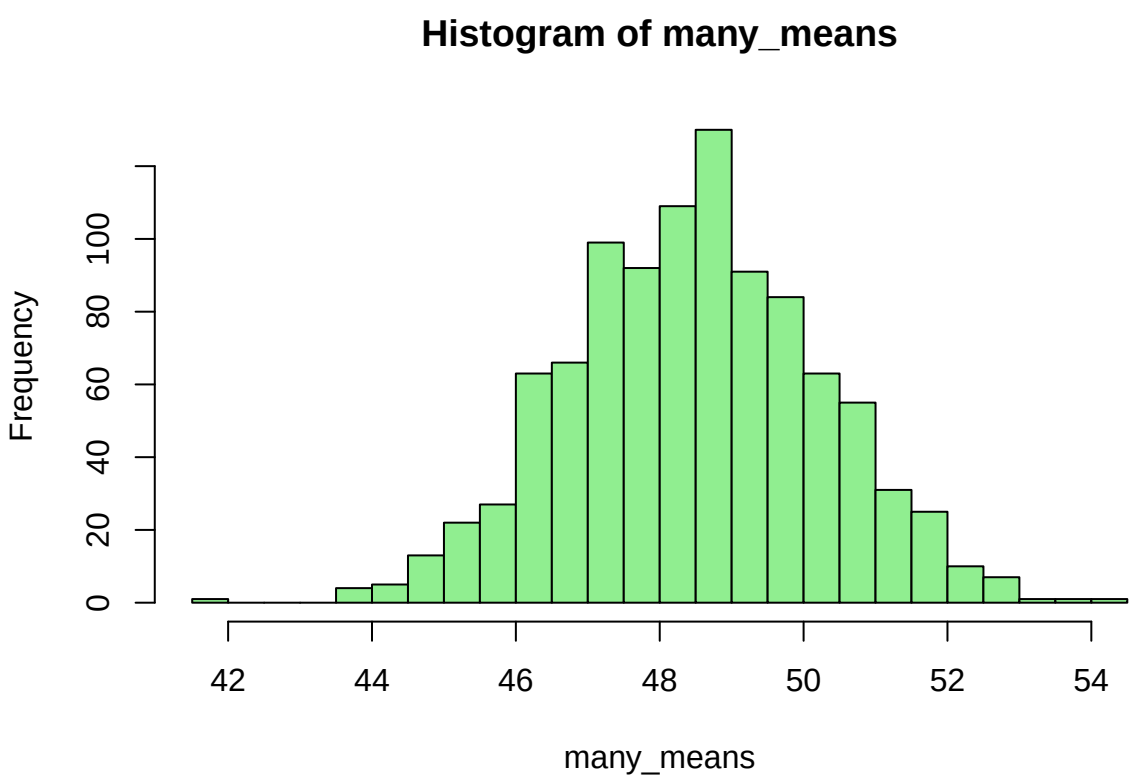
```r
N <- 10
many_means <- replicate(1000, expr = {
  power_samples <- rnorm(N, power_mean, power_sd)
  angle_samples <- rbeta(N, angle_a, angle_b)*(pi/2)
  precision_samples <- runif(N, precision_left, precision_right)

  distance_colleague <- ComputeD(power_samples, precision_samples, angle_samples)

  return(mean(distance_colleague))
})

# there's quite some difference in what the average throwing distance is.
mean(many_means)
```

```
## [1] 48.46936
```

```r
var(many_means)
```

```
## [1] 3.076221
```

```
hist(many_means, breaks="FD", col='lightgreen')
```



Note that this is like comparing the results of a 1000 students performing the experiment in **c**!

e

Suppose you would throw the ball a 1000 times, instead of just 10 times. Do you think your estimate of the average will improve? Also compare your result of a 1000 throws with those of a fellow student. First think about it together, then write some code to check your answer.

Answer:

The estimator is the same. However there variance is a lot smaller. Therefore the difference between your answers should be less, as the sample size gives more reliable results.

```
N <- 1000

many_means <- replicate(1000, expr = {
  power_samples <- rnorm(N, power_mean, power_sd)
  angle_samples <- rbeta(N, angle_a, angle_b)*(pi/2)
  precision_samples <- runif(N, precision_left, precision_right)

  distance_colleague <- ComputeD(power_samples, precision_samples, angle_samples)

  return(mean(distance_colleague))
})

mean(many_means)
```

```
## [1] 48.41666
```

```r
var(many_means)
```

```
## [1] 0.03393585
```

The variance drastically reduces (by a factor of (1000/10)).

f

Use summary statistics you can obtain from the results in **e** to produce a 95% confidence interval around the average throwing distance you've found. How would you judge my throwing technique?

Use central limit theorem, i.e. the average throwing distance follows the process of a normal distribution.

Answer:

```r
interval <- mean(many_means) - c(qnorm(0.975), qnorm(0.0275)) * sd(many_means)
interval
```

```
## [1] 48.05560 48.77015
```

Pretty bad, the throw is *biased*! On average, it is a few meters short from what it should be!

Exercises part 2

2.1 Statistical distributions in R (recap, i.e. skip if you are confident with these)

Go to the following website about Statistical distributions in R. Read the text of chapter 4 thoroughly, and perform the examples of 4.1 through 4.4 in R.

2.2 A mini Monte Carlo operation about the Standard Error of the Mean

As you know from your basic statistics course, the standard error of the sample mean is given by

$$\sigma_{\overline{X}} = \sigma/\sqrt{N},$$

where σ denotes the standard deviation in the population, $\overline{X}$ denotes the sample mean and N denotes the sample size. In this assignment you will provide some evidence for this formula using `R` programming.

a

Take `set.seed(1105)` and generate $B = 1000$ samples of each $N_1 = 10$ objects (= participants) from a χ^2 distribution with 10 degrees of freedom.

Repeat this (with $B = 1000$) for $N_2 = 100$ and $N_3 = 1000$.

Show in `R` with computations on the generated samples that the formula of the standard error holds for a growing number of objects N.

Answer: We generate the data with the following code:

```r
set.seed(1105)
B <- 1000
N1 <- matrix(rchisq(B * 10, df = 10), nrow = 10)
N2 <- matrix(rchisq(B * 100, df = 10), nrow = 100)
N3 <- matrix(rchisq(B * 1000, df = 10), nrow = 1000)
mc_data <- list(N1 = N1, N2 = N2, N3 = N3)
```

Now that we have the `mc_data` object, we can directly calculate the standard error of the sample mean for the sample sizes $N_1 = 10$, $N_2 = 100$ and $N_3 = 1000$:

```r
est_se <- lapply(mc_data, function(X){
  sd(colMeans(X))
})
```

The variance of a χ^2-distribution is twice the mean (degrees of freedom = `df`) of the distribution. Hence, the true standard error of the sampling mean four our specif sample sizes are

```r
exp_df <- 10; N_e1 <- 10; N_e2 <- 100; N_e3 <- 1000
true_se <- round(c( N1 = sqrt(2*exp_df/N_e1),
                    N2 = sqrt(2*exp_df/N_e2),
                    N3 = sqrt(2*exp_df/N_e3)
                  ), 3)
paste0("N", 1:3, " = ", true_se)
```

```
## [1] "N1 = 1.414" "N2 = 0.447" "N3 = 0.141"
```

The estimated standard errors were:

```r
est_se <- round(unlist(est_se), 3)
paste0("N", 1:3, " = ", est_se)
```

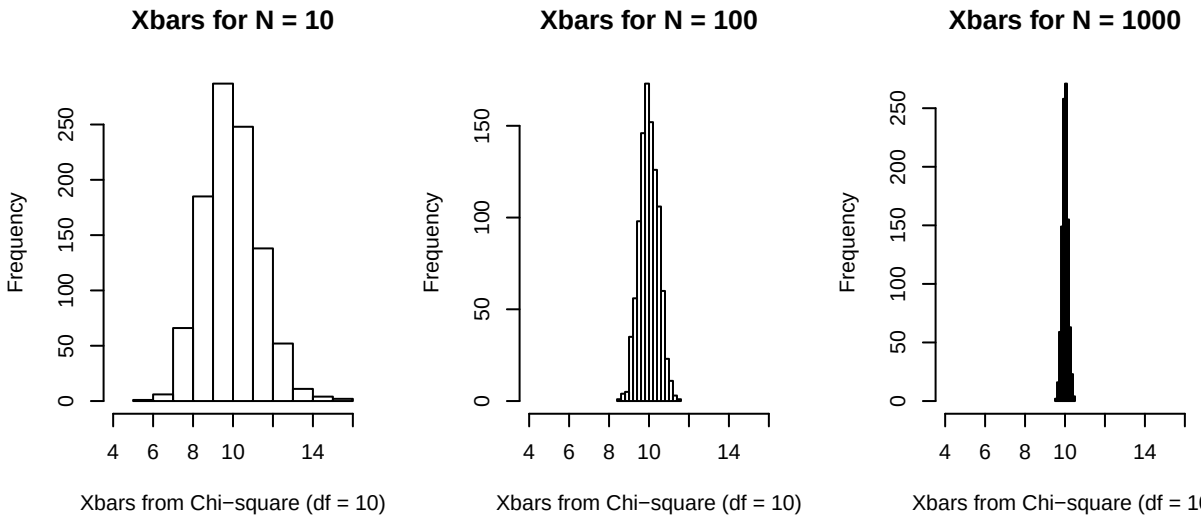
```
## [1] "N1 = 1.384" "N2 = 0.462" "N3 = 0.145"
```

Hence, we can verify that the estimate of our standard error of the mean is really close to the true value.

b

There is a clear relationship between sample size N and $\sigma_{\overline{X}}$. Explain this relationship by visualizing the sampling distributions of the mean for the samples with the three different sample sizes generated for question 1 (you may show three separate plots).

Answer: See below the histograms, of the sampling distributions of the mean, for the three different sample sizes



From these histograms, we may conclude that the smaller the size of the samples, the wider the sampling distribution of the sample mean is around the population mean. In other words, the sample means of larger samples are closer to the population mean.

c

What happens with the standard error of the sample mean for samples of sizes $N_1 = 10$, $N_1 = 100, N_3 = 1000$ if we add a non-centrality parameter $\lambda = 5$ to the $\chi^2(10)$-distribution from which we generate the data?

What are the theoretical standard errors of the mean?

What would be your estimates of the standard error of the mean for each sample size using R code? Use, again, $B = 1000$ Monte Carlo samples for each sample size.

Answer: The standard error of the mean would be

$$\sqrt{(2 * df + 4 * \lambda)/N}.$$

For our samples the theoretical standard errors are:

```
lmbd <- 5;
true_se <- round(c( N1 = sqrt((2 * exp_df + 4 * lmbd)/N_e1),
                    N2 = sqrt((2 * exp_df + 4 * lmbd)/N_e2),
                    N3 = sqrt((2 * exp_df + 4 * lmbd)/N_e3)
                  ), 3
                 )
paste0("N", 1:3, " = ", true_se)
```

```
## [1] "N1 = 2"     "N2 = 0.632" "N3 = 0.2"
```

Our estimates:

```
N_10nc <- matrix(rchisq(B*N_e1, df = 10, ncp = lmbd), nrow = N_e1)
N_100nc <- matrix(rchisq(B*N_e2, df = 10, ncp = lmbd), nrow = N_e2)
N_1000nc <- matrix(rchisq(B*N_e3, df = 10, ncp = lmbd), nrow = N_e3)
mc_ncdata <- list(N1 = N_10nc, N2 = N_100nc, N3 = N_1000nc)

est_ncse <- do.call(c, lapply(mc_ncdata, function(X){
```

```
  sd(colMeans(X))
}))
```

Exercises part 3

3.1 Biased exercise

In this exercise we'll look at something you know quite well: if a distribution is symmetric, the median and the mean are the same. If however a distribution is skewed, the mean and median are different.

a

Sample $N = 1000$ values from a standard normal distribution. Calculate the median and use this as an estimate for the mean.

Answer:

```
N <- 1000
my_sample <- rnorm(N)
median(my_sample)
```

```
## [1] -0.0437734
```

b

Repeat **a** $B = 1000$ times and determine the bias of the estimate of the mean, via the median.

Answer:

```
B <- 1000

set.seed(435344)
my_sim_result <- replicate(B, {
  median(rnorm(N))
})

mean(my_sim_result)
```

```
## [1] -0.001082661
```

```
t.test(my_sim_result)
```

```
##
##  One Sample t-test
##
## data:  my_sim_result
## t = -0.88955, df = 999, p-value = 0.3739
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
```

```
##  -0.003471006  0.001305683
## sample estimates:
##    mean of x
## -0.001082661
```

The average of the medians is not too far off the true mean, which is 0. If N is smaller, e.g. 100 you may see a different result!

c

Repeat **a** and **b** but instead of sampling from a normal distribution, sample from an exponential distribution with mean 1. Is the median biased as estimate of the mean?

Answer:

```
set.seed(435344)
my_sim_result <- replicate(B, {
  median(rexp(N))
})

mean(my_sim_result)
```

```
## [1] 0.694768
```

```
t.test(my_sim_result, mu=1)
```

```
##
##  One Sample t-test
##
## data:  my_sim_result
## t = -298.14, df = 999, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 1
## 95 percent confidence interval:
##  0.692759 0.696777
## sample estimates:
## mean of x
##  0.694768
```

This result is much different from the result in **b**. In this case we would advice against using the median as an estimate of the mean (especially since we have such a good alternative available to us (which is just directly calculating the mean)).

3.2 Simulation with the use of set operations

Read the extended example 8.6.3 of the Matloff book "A combinatorial simulation".

a

First try to imagine what the output is of `sim()` and `choosecom()`.

b

Simulate a solution to the problem by using these functions (given below). Choose `nreps` = 100.

```
sim <- function(nreps) {
   commdata <- list()  # will store all our info about the 3 committees
   commdata$countabsamecomm <- 0
   for (rep in 1:nreps) {
      commdata$whosleft <- 1:20  # who's left to choose from
      commdata$numabchosen <- 0  # number among A, B chosen so far
      # choose committee 1, and check for A,B serving together
      commdata <- choosecomm(commdata,5)
      # if A or B already chosen, no need to look at the other comms.
      if (commdata$numabchosen > 0) next
      # choose committee 2 and check
      commdata <- choosecomm(commdata,4)
      if (commdata$numabchosen > 0) next
      # choose committee 3 and check
      commdata <- choosecomm(commdata,3)
   }
   print(commdata$countabsamecomm/nreps)
}

choosecomm <- function(comdat,comsize) {
   # choose committee
   committee <- sample(comdat$whosleft,comsize)
   # count how many of A and B were chosen
   comdat$numabchosen <- length(intersect(1:2,committee))
   if (comdat$numabchosen == 2)
      comdat$countabsamecomm <- comdat$countabsamecomm + 1
   # delete chosen committee from the set of people we now have to choose from
   comdat$whosleft <- setdiff(comdat$whosleft,committee)
   return(comdat)
}
```

Answer:

```
sim(100)
```

```
## [1] 0.08
```

Thus the probability that A and B are in the same committee is 12 percent.

c

Adapt the code in `sim()` in such a way that we can solve the following problem: Two committees of sizes 5 and 10 are chosen from 20 people. What is the probability that persons A and B are chosen for the same committee?

```
SimAdapt <- function(nreps) {
   commdata <- list()  # will store all our info about the 3 committees
   commdata$countabsamecomm <- 0
```

```
   for (rep in 1:nreps) {
      commdata$whosleft <- 1:20  # who's left to choose from
      commdata$numabchosen <- 0  # number among A, B chosen so far
      # choose committee 1, and check for A,B serving together
      commdata <- choosecomm(commdata, 5)
      # if A or B already chosen, no need to look at the other comms.
      if (commdata$numabchosen > 0) next
      # choose committee 2 and check
      commdata <- choosecomm(commdata, 10)
      if (commdata$numabchosen > 0) next
       }
   print(commdata$countabsamecomm/nreps)
}
SimAdapt(100)
```

```
## [1] 0.37
```

Thus, in this new situation with two committees of sizes 5 and 10 chosen from 20 people, the probability that A and B are chosen for the same committee is ca. 34 percent.

Exercises part 4

Simulation error

In this exercise we'll 'verify' our math in one of the slides about simulation error. In that particular slide we posit that:

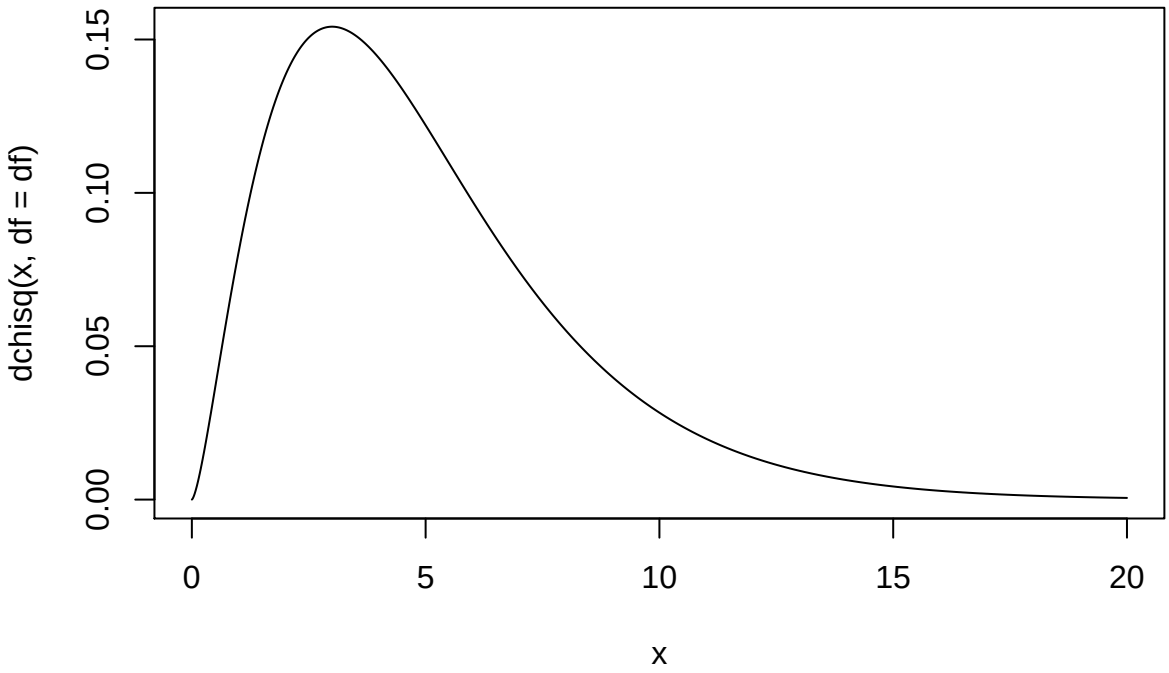
- if $\mathrm{E}[Z]$ is estimated by $\bar{Z}_B$ for i.i.d. $Z^1, \ldots, Z^B$, then $\bar{Z}_B - \mathrm{E}[Z] \sim\approx N(0, varZ/B)$.
- the simulation error can be made arbitrarily small, by making B larger.

Show using code that this relation is approximately true for the following case: $N = 50$ and data $Z \sim \chi^2_5$. Thus Z^1 is the mean of a vector of N values that are distributed according to a chi-squared distribution with 5 degrees of freedom. Take $B = 50$ and $B = 250$ and compare the results.

Answer:

To clarify: we perform a monte carlo study, to look at, for example, the sampling distribution of the mean of a sample Z. Z^1 is a single sample, of which we can calculate the mean. This mean will have a sampling dsitribution: depending on the sample, the mean may be smaller, or larger, than the true mean, with varying size differences. The mean of this sampling distribution, can be considered our 'monte carlo' estimate of the mean of this sampling distribution. The standard error of simulation is the standard error of this monte carlo estimate. To find this empirically, simply repeat our monte carlo study and look at the standard deviation of the various results we got from these monte carlo replicates.

```
# some preliminary setting of constants and looking at distribution of values from chi-squared distribu
N <- 50
df <- 5
x <- seq(0, 20, by = 0.01)
plot(x, dchisq(x, df = df), type = 'l')
```




```
set.seed(5674)
B <- 50
# theoretically:
variance <- df*2
Z <- replicate(B, mean(rchisq(N, df = df)))
# error of sampling distribution of Z_bar:
sqrt(sum((Z-mean(Z))^2)/B)
```

```
## [1] 0.446217
```

```
# simulation error of mean
sqrt(1/B * sum((Z-mean(Z))^2)/B)
```

```
## [1] 0.06310461
```

```
# emperically, via monte carlo:
# standard deviation of the replicates of simulation of the mean of Z:
monte_carlo_replications <- 100
sd(replicate(monte_carlo_replications, {
  Z <- replicate(B, mean(rchisq(N, df = df)))
  mean(Z)
}))
```

```
## [1] 0.05929689
```

```
set.seed(5674)
B <- 250
# theoretically:
variance <- df*2
Z <- replicate(B, mean(rchisq(N, df=df)))
```

```r
# error of sampling distribution of Z_bar:
sqrt(sum((Z - mean(Z))^2)/B)
```

```
## [1] 0.4004295
```

```r
# simulation error of mean
sqrt(1/B * sum((Z - mean(Z))^2)/B)
```

```
## [1] 0.02532539
```

```r
# via monte carlo:
# standard deviation of the replicates of simulation of the mean of Z:
sd(replicate(monte_carlo_replications, {
  Z <- replicate(B, mean(rchisq(N, df = df)))
  mean(Z)
}))
```

```
## [1] 0.02936527
```

Simulation error is smaller for $B = 250$ of course. This size however also depends on N! A study with more replicates, will be more precise and so the average of those studies will also be more precise.

3.3 Power, an example

In this exercise we'll see how you could implement a monte carlo study to estimate the power. We'll work with a test we devise ourselves. Power is an important property of the test, so it is only natural we would want to investigate this for our newly devised test. Work through the pieces of code in each of the subquestions and try to understand what the code does.

a

This is test statistic for the test we've devised. The test is going to be two-sided.

```r
# our test statistic function
TestStat <- function(x) {
    n <- length(x)
    k <- trunc(0.3 * n)
    y <- sort(x)[(k + 1):(n - k)]
    statistic_T <- mean(y)/sd(y)
    return(statistic_T)
}
```

Answer: The statistic of test is basically a standardized mean, based on trimmed data!

b

In this piece of code, we're looking at some of the properties of our test under the null hypothesis.

```r
mean_h0 <- 0
N <- 20
B <- 10000

statistic_t <- numeric(B)

set.seed(37682362)
for (i in 1:B) {
  statistic_t[i] <- TestStat(rnorm(N, mean_h0))
}

alpha <- 0.05
critical_value <- quantile(statistic_t, 1-alpha/2)
```

Answer: More specifically, we're looking at the sampling distribution of our test statistic of the test, when $N = 20$. Then we use the sampling distribution to determine the critical value for a two-sided test: if we reject the null hypotheses for observed test statistics more extreme than this value, we're garanteed to *falsely* reject the null hypothesis in at most $\alpha = 0.05$ proportion of cases. *Disclaimer*: we've determined the critical value using a simulation using random sampling! `quantile` will have a sampling distribution *around* the true quantile value for our test. Furthermore we've based the result on a single replicate of a monte carlo study, so we're also subject to some simulation error!

c

Here we check the set-up we've created for the test in **a** and **b**.

```r
a <- numeric(B)

set.seed(14234712)
for (i in 1:B) {
    x <- rnorm(N, mean_h0)
    a[i] <- (abs(TestStat(x)) > critical_value)
}

mean(a)
```

```
## [1] 0.0542
```

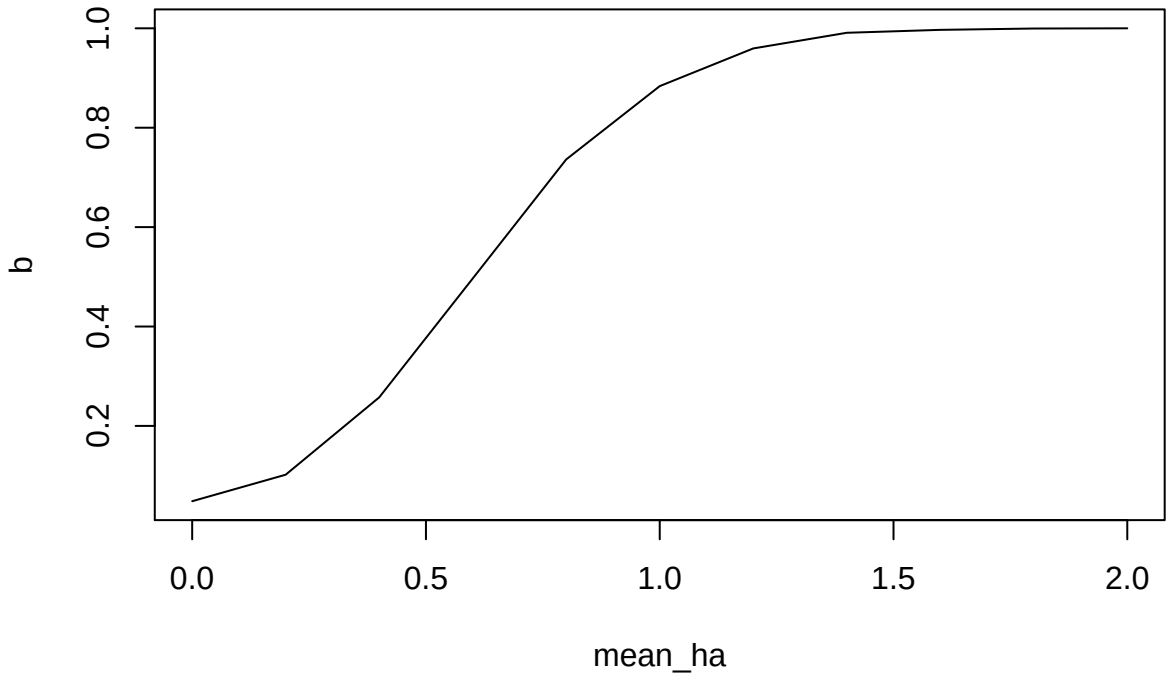
Answer: As mentioned the critical value is determined with some simulation error, furthermore the mean of the vector `a` is subject to some simulation error and sampling error, so it's not going to be *exactly* equal to α! But it is close.

d

Here we look at some properties of our test under an alternative hypothesis.

```r
mean_ha <- seq(0, 2, by = 0.2)

b <- numeric(length(mean_ha))

set.seed(1231273)
```

```r
for (j in 1:length(mean_ha)) {
  statistic_t <- numeric(B)
    for (i in 1:B) {
        x <- rnorm(N, mean_ha[j])
        statistic_t[i] <- TestStat(x)
    }
    b[j] <- mean(abs(statistic_t) > critical_value)
}

plot(mean_ha, b, type='l')
```



Answer: We're evaluating the proportion of times the null hypothesis is rejected when data is generated under some alternative hypothesis. These alternative hypotheses are represented by varying means in `mean_ha`. The test shows some favourable properties: under the null hypothesis our α is garanteed, and as the alternative hypothesis is more different from the null hypothesis, the power increases (it even goest to 1).

Note thus that: $\alpha \neq 1 - \beta$.

e

Repeat the above process but instead of calculating the standardized mean of trimmed data, calculate the standardized mean of the complete set of data. Repeat the process of determining α and β under the same set of alternative hypotheses. Which test is better?

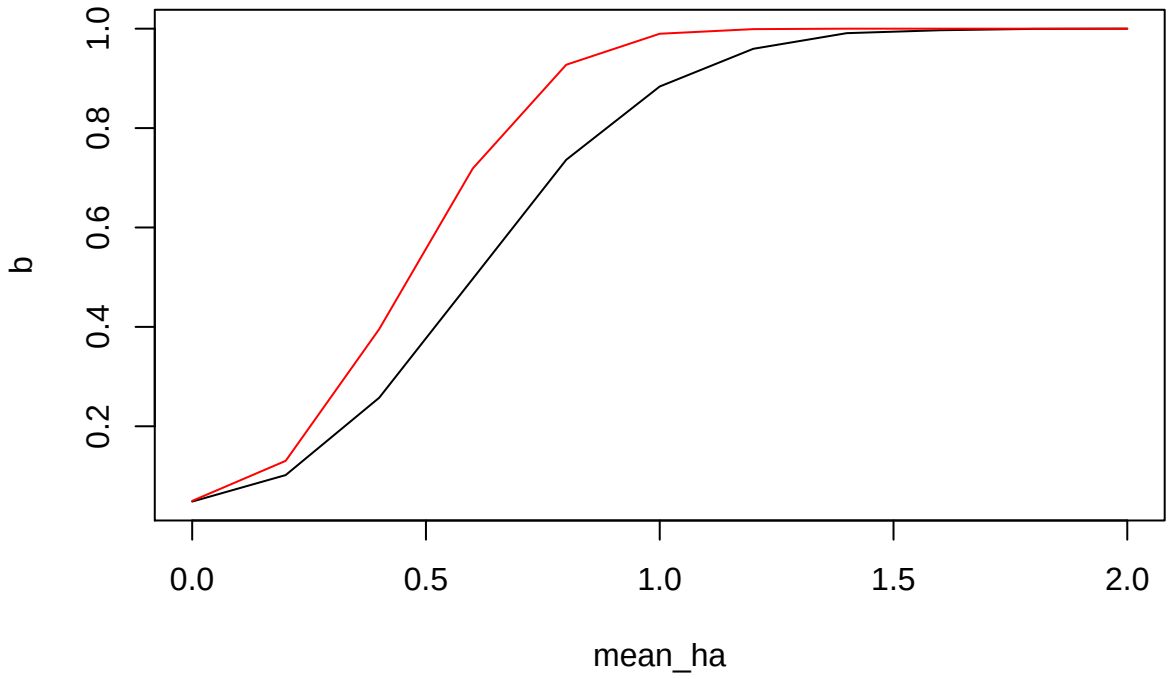
Answer:

```r
TestStat <- function(x) {
    statistic_T <- mean(x)/sd(x)
    return(statistic_T)
}
```

```r
# Already defined in the above code:
# mean_h0 <- 0
# N <- 20
# B <- 10000
# alpha <- 0.05
# mean_ha <- seq(0, 2, by = 0.2)

# determine critical value:
statistic_t <- numeric(B)
set.seed(37682362)
for (i in 1:B) {
  statistic_t[i] <- TestStat(rnorm(N, mean_h0))
}
critical_value <- quantile(statistic_t, 1-alpha/2)

#check alpha
a <- numeric(B)
set.seed(14234712)
for (i in 1:B) {
    x <- rnorm(N, mean_h0)
    a[i] <- (abs(TestStat(x)) > critical_value)
}
mean(a)
```

```
## [1] 0.0525
```

```r
b_untrimmed <- numeric(length(mean_ha))

set.seed(1231273)
for (j in 1:length(mean_ha)) {
  statistic_t <- numeric(B)
    for (i in 1:B) {
        x <- rnorm(N, mean_ha[j])
        statistic_t[i] <- TestStat(x)
    }
    b_untrimmed[j] <- mean(abs(statistic_t) > critical_value)
}

plot(mean_ha, b, type = 'l')
lines(mean_ha, b_untrimmed, col = 'red')
```



The power of the untrimmed test is higher! This would indicate that this test is better to detect a mean different from the null hypothesis.

f

Do you think the power curve for the test looked at in **a** through **e** would be different if the sample size was different? For example $N = 100$ instead of $N = 20$?

Answer: Yes, would be very different! Try this for yourself by simply changing the value of `N` and plotting the power curve again.

Self-Study (Difficult!!! Already at Difficult Questions of the Exam Level)

Power of a t-test

Time to do this on your own. Suppose we have a random sample A of N observations from a normal distribution with $\mu = 3$ and $\sigma = 1$. (Imagine, for example, that the population mean of 3 indicates that members of this population play *on average* three hours of computer games a day). In addition, we have a random sample B of N observations from a normal distribution with $\mu = 2.5$ and $\sigma = 1$. (Imagine, for example, that due to an intervention, members of population B play *on average* two and a half hours of computer games a day).

Let the null hypothesis be $\mu_A = \mu_B$, where μ_A denotes the mean value of population A (the control group), and μ_B denotes the mean value of population B (the intervention group). The alternative hypothesis is that $\mu_A \neq \mu_B$. By means of an independent Student's t-test, we are investigating whether the intervention indeed changed the amount of time spend on computer gaming (in other words whether the alternative hypothesis was true). Remember that we can make two types of error: Either we accept the alternative hypothesis, while the null-hypothesis is true (i.e., Type I error, denoted with α), or we accept the null hypothesis, while the alternative hypothesis is true (i.e., Type II error, denoted with β).

If $N_A = N_B = 64$, the power (i.e., $1 - \beta$) of the independent Student's t-test equals 0.80, given a two-sided $\alpha = 0.05$. Show empirically (by simulating many samples, and performing a t-test on each of these samples) that this is true. Use set.seed(4444).

Answer:

```
set.seed(4444)
k <- 10000
matA <- matrix(rnorm(64 * k, 2, 1), nrow = 64)
matB <- matrix(rnorm(64 * k, 1.5, 1), nrow = 64)
pval <- numeric(k)

for (i in 1:k){
  pval[i] <- t.test(matA[ ,i], matB[ ,i])$p.value
}

sum(pval < 0.05) / k
```

```
## [1] 0.8007
```

3.4 Sample size calculation

In many applied research fields, a sample size calculation is performed before the actual experiment (e.g., a randomized controlled trial) is conducted. Besides levels for power and α, a sample size calculation (for Student's t-test) needs a difference in the means between two samples (e.g., this difference can be based on previous studies). The mean difference is usually expressed as a standardized mean difference (i.e. an effect size d). In the example above, the effect size d was equal to 0.50, which is considered to be a medium effect size (Cohen, 1988).

Perform a simulation study to investigate empirically which minimum sample size is needed for a given combination of effect size, power level and alpha level. Use the following design factors in your simulation study: an effect size of 0.20, 0.50, and 0.80, and a power level of 0.80 and 0.90. Fix the two-sided α level at 0.05. Use a full factorial design, thus in this case 3 by 2 combinations are possible. For each of these

combinations, find **empirically** the minimum sample size needed. Do not use any existing formula for sample size calculation. Again, use `set.seed(4444)`.

Show in your answer: - the code you used for this simulation study, - a clear presentation of the results of the simulations, and - a conclusion.

Hint: start with writing code to sample the data you need to perform a test. Then write some code to perform the actual test. Then write code to repeat those two steps.

Answer: First we create a function to generate the data for two populations with varying standardized difference in means.

```
gendata <- function(es, nsamp, k){
  #function that generates data
  #
  #Args:
  #es = standardized difference in population means
  #nsamp = number of observations in our sample
  #k = number of samples
  #population A is our "control" group population
  meanA <- 2
  matA <- matrix(rnorm(nsamp * k, meanA, 1), nrow = nsamp)
  #Population B is our "intervention" group population
  matB <- matrix(rnorm(nsamp * k, meanA - es, 1), nrow = nsamp)
  #We combine the data in a list object with pairs of columns,
  #one column from matA and one from matB.
  listobj <- lapply(1:k, function(i){
                  cbind(matA[ ,i], matB[ ,i])
                  })
  return(listobj)
}
```

Now, we can generate the data with small, medium, and large effect size, using this function:

```
set.seed(4444)
es <- c(0.20, 0.50, 0.80)
nsamp <- 800
datasmall <- gendata(es[1], nsamp, 1000)
datamed <- gendata(es[2], nsamp, 1000)
datalarge <- gendata(es[3], nsamp, 1000)
```

Now that we have our list objects, we can make a function that we are going to apply to this list. The function needs to return the p-value of a two-sample t-test.

```
ttestp <- function(X, n) {
  # function that returns p-value of indep samples t-test
  #
  # Args:
  #   X: matrix with observations in the rows, and the
  #     two samples in the (two) columns.
  #   n: number of observations we select
  #
  tobj <- t.test(X[1:n, 1], X[1:n, 2])
  return(tobj$p.value)
}
```

Now we can start to determine the minimum sample size n to achieve a power of higher or equal to a critical value, say, 0.80. Let's write a function to do this:

```
samplesize <- function(nobs, listobj, powercrit = 0.80){
  #function to determin the minimumn sample size for a critical power   level
  #Args:
  #   nobs: a vector with possible values for n
  #   listobj: list with pairs of observations from 2 populations
  #   powercrit: the critical power level
  power <- 0
  i <- 1
  while (power < powercrit){
    i <- i + 1
    pval <- sapply(listobj, ttestp, n = nobs[i])
    #calculate the power:
    power <- sum(pval < 0.05)/length(pval)
    #follow the output:
    print(c(nobs[i], power))
  }
  return(nobs[i])
}
```

We apply the sample size function to our data. To speed up the process, we first use a vector with possible values of n (`nobs`) with large intervals in between, and then with small intervals.

Results for effect size = 0.20 and power = 0.80:

```
ntot <- nrow(datasmall[[1]])
#Vector nobs with large intervals:
nobs <- seq(1, ntot, by = 50)
nminrough <- samplesize(nobs, datasmall, powercrit = 0.80)
```

```
## [1] 51.000  0.178
## [1] 101.000   0.287
## [1] 151.000   0.393
## [1] 201.000   0.501
## [1] 251.000   0.615
## [1] 301.000   0.693
## [1] 351.000   0.756
## [1] 401.000   0.802
```

```
#Vector nobs with small intervals:
nobs <- seq(nminrough - 20, nminrough, by = 1)
nminsmall80 <- samplesize(nobs, datasmall, powercrit = 0.80)
```

```
## [1] 382.000   0.787
## [1] 383.000   0.787
## [1] 384.000   0.786
## [1] 385.00   0.79
## [1] 386.000   0.788
## [1] 387.000   0.792
## [1] 388.000   0.792
## [1] 389.000   0.797
```

```
## [1] 390.000   0.796
## [1] 391.000   0.799
## [1] 392.000   0.794
## [1] 393.000   0.803
```

```
nminsmall80
```

```
## [1] 393
```

Thus the minimum sample size needed is 393.

Results for effect size = 0.20 and power = 0.90: For a critical power level of 0.90, the minimum sample size needed will be higher than the previous one. We apply the same stategy as before, but start with the previous result, to speed up the search process a bit.

```
#Vector nobs with large intervals:
nobs <- seq(nminsmall80, ntot, by = 50)
nminrough <- samplesize(nobs, datasmall, powercrit = 0.90)
```

```
## [1] 443.00   0.84
## [1] 493.000   0.872
## [1] 543.000   0.904
```

```
#Vector nobs with small intervals:
nobs <- seq(nminrough - 20, nminrough, by = 1)
nminsmall90 <- samplesize(nobs, datasmall, powercrit = 0.90)
```

```
## [1] 524.000   0.893
## [1] 525.000   0.894
## [1] 526.000   0.893
## [1] 527.000   0.897
## [1] 528.000   0.898
## [1] 529.000   0.895
## [1] 530.000   0.893
## [1] 531.000   0.894
## [1] 532.000   0.896
## [1] 533.000   0.896
## [1] 534.000   0.899
## [1] 535.000   0.899
## [1] 536.000   0.898
## [1] 537.000   0.899
## [1] 538.000   0.895
## [1] 539.000   0.896
## [1] 540.000   0.898
## [1] 541.000   0.899
## [1] 542.000   0.902
```

```
nminsmall90
```

```
## [1] 542
```

Thus the minimum sample size needed is 542.

Results for effect size = 0.50 and power = 0.80:

```
nminrough <- samplesize(seq(1, ntot, by = 50), datamed)
```

```
## [1] 51.0  0.7
## [1] 101.000   0.946
```

```
#Vector nobs with small intervals:
nobs <- seq(nminrough - 40, nminrough, by = 1)
nminmed80 <- samplesize(nobs, datamed)
```

```
## [1] 62.000  0.779
## [1] 63.000  0.787
## [1] 64.000  0.791
## [1] 65.000  0.784
## [1] 66.000  0.797
## [1] 67.000  0.804
```

```
nminmed80
```

```
## [1] 67
```

Results for effect size = 0.50 and power = 0.90:

```
nminrough <- samplesize(seq(nminmed80, ntot, by = 50),
                        datamed, powercrit = 0.90)
```

```
## [1] 117.000   0.963
```

```
#Vector nobs with small intervals:
nobs <- seq(nminrough - 40, nminrough, by = 1)
nminmed90 <- samplesize(nobs, datamed, powercrit = 0.90)
```

```
## [1] 78.000  0.874
## [1] 79.000  0.877
## [1] 80.000  0.874
## [1] 81.000  0.884
## [1] 82.000  0.887
## [1] 83.000  0.898
## [1] 84.000  0.901
```

```
nminmed90
```

```
## [1] 84
```

Results for effect size = 0.80 and power = 0.80:

```
nminrough <- samplesize(seq(0, ntot, by = 50),
                        datalarge, powercrit = 0.80)
```

```
## [1] 50.000  0.974
```

```
#Vector nobs with small intervals:
nobs <- seq(nminrough - 40, nminrough, by = 1)
nminlarge80 <- samplesize(nobs, datalarge, powercrit = 0.80)
```

```
## [1] 11.000  0.459
## [1] 12.000  0.494
## [1] 13.000  0.513
## [1] 14.00  0.54
## [1] 15.00  0.58
## [1] 16.000  0.605
## [1] 17.000  0.625
## [1] 18.000  0.651
## [1] 19.000  0.683
## [1] 20.000  0.712
## [1] 21.00  0.74
## [1] 22.000  0.747
## [1] 23.000  0.764
## [1] 24.000  0.782
## [1] 25.000  0.808
```

```
nminlarge80
```

```
## [1] 25
```

Results for effect size = 0.80 and power = 0.90:

```
#Vector nobs with small intervals:
nobs <- seq(nminlarge80, nminlarge80 + 50, by = 1)
nminlarge90 <- samplesize(nobs, datalarge, powercrit = 0.90)
```

```
## [1] 26.000  0.815
## [1] 27.000  0.835
## [1] 28.000  0.843
## [1] 29.000  0.856
## [1] 30.000  0.864
## [1] 31.00  0.87
## [1] 32.000  0.879
## [1] 33.000  0.889
## [1] 34.000  0.898
## [1] 35.000  0.908
```

```
nminlarge90
```

```
## [1] 35
```

Conclusion: the minimum sample size needed is highly influenced by the effect size. Higher effect sizes result in smaller required sample sizes. The minimum sample size needed is also influenced by the critical power level: to achieve a higher power level (i.e. lower risk of Type II error) a higher minimum sample size is needed.