```
Error in options(digits):  object 'digits' not found
```

Chapter 4

Statistics Simulation Based

Contents

4 Statistics Simulation Based **2**
4.1 Probability and Simulation 4
4.1.1 Introduction 4
4.1.2 Simulation as a general computational tool 5
4.1.3 Propagation of error 7
4.2 The parametric bootstrap 12
4.2.1 Introduction 12
4.2.2 One-sample confidence interval for μ 13
4.2.3 One-sample confidence interval for any feature assuming any distribution 15
4.2.4 Two-sample confidence intervals assuming any distributions 19
4.3 The non-parametric bootstrap 24
4.3.1 Introduction 24
4.3.2 One-sample confidence interval for μ 24
4.3.3 One-sample confidence interval for any feature 26
4.3.4 Two-sample confidence intervals 27
4.4 OPTIONAL: Bootstrapping – a further perspective 30
4.4.1 Non-parametric bootstrapping with the `boot`-package . . 31
4.5 Exercises 40

Glossaries **45**

Acronyms **46**

4.1 Probability and Simulation

4.1.1 Introduction

One of the really big gains for statistics and modeling of random phenomena, provided by computer technology during the last decades, is the ability to simulate random systems on the computer, as we have already seen much in use in Chapter 2. This provides possibilities to obtain results that otherwise from a mathematical analytical point of view would be impossible to calculate. And, even in cases where the highly educated mathematician/physicist might be able to find solutions, simulation is a general and simple calculation tool allowing solving complex problems without a need for deep theoretical insight.

An important reason for including this subject in an introductory statistics course, apart from using it as a pedagogical tool to aide the understanding of random phenomena, is the fact that the methods we are usually introducing in basic statistics are relying on one of two conditions:

1. The original data population density is a normal distribution
2. Or: The sample size n is large enough to make this assumption irrelevant for what we do

And in real settings it may be challenging to know for sure whether any of these two are really satisfied, so to what extend can we trust the statistical conclusions that we make using our basic tools, as e.g. the one- and two-sample statistical methods presented in Chapter 3. And how should we do the basic statistical analysis if we even become convinced that none of these two conditions are fulfilled? Statistical data analysis based on simulation tools is a valuable tool to complete the tool box of introductory statistics.

However, it will become clear that the simulation tools presented here will make us rapidly able to perform statistical analysis that goes way beyond what historically has been introduced in basic statistics classes or textbooks. Unfortunately, the complexity of real life engineering applications and data analysis challenges can easily go beyond the settings that we have time to cover within an introductory exposition. With the general simulation tool in our tool box, we have a multitool that can be used for (and adapted to) basically almost any level of complexity that we will meet in our future engineering activity.

The classical statistical practice would be to try to ensure that the data we're analyzing behaves like a normal distribution: symmetric and bell-shaped histogram. In Chapter 3 we also learned that we can make a normal q-q plot to

verify this assumption in practice, and possibly transform the data to get them closer to being normal. The problem with small samples is that it even with these diagnostic tools can be difficult to know whether the underlying distribution really is ”normal” or not.

And in some cases the assumption of normality after all simply may be obviuosly wrong. For example, when the response scale we work with is far from being quantitative and continuous - it could be a scale like ”small”, ”medium” and ”large” - coded as 1, 2 and 3. We need tools that can do statistical analysis for us WITHOUT the assumption that the normal distribution is the right model for the data we observe and work with.

Traditionally, the missing link would be covered by the so-called non-parametric tests. In short this is a collection of methods that make use of data at a more coarse level, typically by focusing on the rank of the observations instead of the actual values of the observations. So in a paired t-test setup, for example, one would just count how many times the observations in one sample is bigger than in the other – instead of calculating the differences. In that way you can make statistical tests without using the assumption of an underlying normal distribution. There are a large number of such non-parametric tests for different setups. Historically, before the computer age, it was the only way to really handle such situations in practice. These tests are all characterized by the fact that they are given by relatively simple computational formulas which in earlier times easily could be handled.

The simulation based methods that we now present instead have a couple of crucial advantages to the traditional non-parametric methods:

- Confidence intervals are much easier to achieve
- They are much easier to apply in more complex situations

4.1.2 Simulation as a general computational tool

Basically, the strength of the simulation tool is that one can compute arbitrary functions of random variables and their outcomes. In other words one can find probabilities of complicated outcomes. As such, simulation is really not a statistical tool, but rather a probability calculus tool. However, since statistics essentially is about analysing and learning from real data in the light of certain probabilities, the simulation tool indeed becomes of statistical importance, which we will exemplify very specifically below. Before starting with exemplifying the power of simulation as a general computational tool, we refer to the introduction to simulation in Chapter 2 – in particular read first Section 2.6, Example

2.15 and thereafter Section 2.6.

|||| Example 4.1 Rectangular plates

A company produces rectangular plates. The length of plates (in meters), X is assumed to follow a normal distribution $N(2, 0.01^2)$ and the width of the plates (in meters), Y are assumed to follow a normal distribution $N(3, 0.02^2)$. We're hence dealing with plates of size 2×3 meters, but with errors in both length and width. Assume that these errors are completely independent. We are interested in the area of the plates which of course is given by $A = XY$. This is a nonlinear function of X and Y, and actually it means that we, with the theoretical tools we presented so far in the material, cannot figure out what the mean area really is, and not at all what the standard deviation would be in the areas from plate to plate, and we would definitely not know how to calculate the probabilities of various possible outcomes. For example, how often such plates have an area that differ by more than 0.1 m^2 from the targeted 6 m^2? One statement summarizing all our lack of knowledge at this point: we do not know the probability distribution of the random variable A and we do not know how to find it! With simulation, it is straightforward: one can find all relevant information about A by just simulating the X and Y a high number of times, and from this compute A just as many times, and then observe what happens to the values of A. The first step is then given by:

```
## Number of simulations
k <- 10000
## Simulate X, Y and then A
X <- rnorm(k, 2, 0.01)
Y <- rnorm(k, 3, 0.02)
A <- X*Y
```

The R object `A` now contains 10.000 observations of A. The expected value and the standard deviation for A are simply found by calculating the average and standard deviation for the simulated A-values:

```
mean(A)

[1] 5.999511


sd(A)

[1] 0.04957494
```

and the desired probability, $P(|A - 6| > 0.1) = 1 - P(5.9 \leq A \leq 6.1)$ is found by counting how often the incident actually occurs among the k outcomes of A:

```
mean(abs(A-6)>0.1)

[1] 0.0439
```

The code `abs(A-6)>0.1` creates a vector with values `TRUE` or `FALSE` depending on whether the absolute value of $A - 6$ is greater than 0.1 or not. When you add (sum) these the `TRUE` is automatically translated into 1 and `FALSE` automatically set to 0, by which the desired count is available, and divided by the total number of simulations k by `mean()`.

Note, that if you do this yourself without using the same seed value you will not get exactly the same result. It is clear that this simulation uncertainty is something we must deal with in practice. The size of this will depend on the situation and on the number of simulations k. We can always get a first idea of it in a specific situation simply by repeating the calculation a few times and note how it varies. Indeed, one could then formalize such an investigation and repeat the simulation many times, to get an evaluation of the simulation uncertainty. We will not pursue this further here. When the target of the computation is in fact a probability, as in the latter example here, you can alternatively use standard binomial statistics, which is covered in Chapter 2 and Chapter 7. For example, with $k = 100000$ the uncertainty for a calculated proportion of around 0.044 is given by: $\sqrt{\frac{0.044(1-0.044)}{100000}} = 0.00065$. Or for example, with $k = 10000000$ the uncertainty is 0.000065. The result using such a k was 0.0455 and because we're a bit unlucky with the rounding position we can in practice say that the exact result rounded to 3 decimal places are either 0.045 or 0.046. In this way, a calculation which is actually based on simulation is turned into an exact one in the sense that rounded to 2 decimal places, the result is simply 0.05.

4.1.3 Propagation of error

Within chemistry and physics one may speak of measurement errors and how measurement errors propagate/accumulate if we have more measurements and/or use these measurements in subsequent formulas/calculations. First of all: The basic way to "measure an error", that is, to quantify a measurement error is by means of a standard deviation. As we know, the standard deviation expresses the average deviation from the mean. It is clear it may happen that a measuring instrument also on average measures wrongly (off the target). This is called "bias", but in the basic setting here, we assume that the instrument has no bias.

Hence, reformulated, an error propagation problem is a question about how the standard deviation of some function of the measurements depends on the

standard deviations for the individual measurement: let $X_1, \ldots, X_n$ be n measurements with standard deviations (average measurement errors) $\sigma_1, \ldots, \sigma_n$. As usual in this material, we assume that these measurement errors are independent of each other. There are extensions of the formulas that can handle dependences, but we omit those here. We must then in a general formulation be able to find

$$\sigma^2_{f(X_1,\ldots,X_n)} = \mathrm{V}(f(X_1, \ldots, X_n)). \tag{4-1}$$

|||| Remark 4.2 For the thoughtful reader: Measurement errors, errors and variances

Although we motivate this entire treatment by the *measurement error* terminology, often used in chemistry and physics, actually everything is valid for *any kind of* errors, be it ”time-to-time” production errors, or ”substance-to-substance” or ”tube-to-tube” errors. What the relevant kind of errors/variabilities are depends on the situation and may very well be mixed together in applications. But, the point is that as long as we have a relevant error variance, we can work with the concepts and tools here. It does not have to have a ”pure measurement error” interpretation.

Actually, we have already in this course seen the linear error propagation rule, in Theorem in 2.56, which then can be restated here as

$$\text{If } f(X_1, \ldots, X_n) = \sum_{i=1}^{n} a_i X_i, \quad \text{then} \quad \sigma^2_{f(X_1,\ldots,X_n)} = \sum_{i=1}^{n} a_i^2 \sigma_i^2.$$

There is a more general non-linear extension of this, albeit theoretically only an approximate result, which involves the partial derivative of the function f with respect to the n variables:

|||| Method 4.3 The non-linear approximative error propagation rule

If $X_1, \ldots, X_n$ are independent random variables with variances $\sigma_1^2, \ldots, \sigma_n^2$ and f is a (potentially non-linear) function of n variables, then the variance of the f-transformed variables can be approximated linearly by

$$\sigma^2_{f(X_1,\ldots,X_n)} = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \right)^2 \sigma_i^2, \tag{4-2}$$

where $\frac{\partial f}{\partial x_i}$ is the partial derivative of f with respect to the i'th variable

In practice one would have to insert the actual measurement values $x_1, \ldots, x_n$ of $X_1, \ldots, X_n$ in the partial derivatives to apply the formula in practice, see the example below. This is a pretty powerful tool for the general finding of (approximate) uncertainties for complicated functions of many measurements or for that matter: complex combinations of various statistical quantities. When the formula is used for the latter, it is also in some contexts called the "delta rule" (which is mathematically speaking a so-called first-order (linear) Taylor approximations to the nonlinear function f). We bring it forward here, because as an alternative to this approximate formula one could use simulation in the following way:

|||| Method 4.4 Non-linear error propagation by simulation

Assume we have actual measurements $x_1, \ldots, x_n$ with known/assumed error variances $\sigma_1^2, \ldots, \sigma_n^2$:

1. Simulate k outcomes of all n measurements from assumed error distributions, e.g. $N(x_i, \sigma_i^2)$: $X_i^{(j)}, j = 1 \ldots, k$.
2. Calculate the standard deviation directly as the observed standard deviation of the k simulated values of f:

$$s^{\text{sim}}_{f(X_1,\ldots,X_n)} = \sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(f_j - \bar{f})^2}, \tag{4-3}$$

where

$$f_j = f(X_1^{(j)}, \ldots, X_n^{(j)}). \tag{4-4}$$

|||| Example 4.5

Let us continue the example with $A = XY$ and X and Y defined as in the example above. First of all note, that we already above used the simulation based error propagation method, when we found the standard deviation to be 0.04957 based on the simulation. To exemplify the approximate error propagation rule, we must find the derivatives of the function $f(x, y) = xy$ with respect to both x and y

$$\frac{\partial f}{\partial x} = y \quad \frac{\partial f}{\partial y} = x.$$

Assume, that we now have two specific measurements of X and Y, for example $X = 2.00$ m and $y = 3.00$ m the error propagation law would provide the following

approximate calculation of the "uncertainty error variance of the area result" $2.00\,\mathrm{m} \cdot 3.00\,\mathrm{m} = 6.00\,\mathrm{m}^2$, namely

$$\sigma_A^2 = y^2 \cdot 0.01^2 + x^2 \cdot 0.02^2 = 3.00^2 \cdot 0.01^2 + 2.00^2 \cdot 0.02^2 = 0.0025.$$

So, with the error propagation law we are managing a part of the challenge without simulating. Actually, we are pretty close to be able to find the correct theortical variance of $A = XY$ using tools provided in this course. By the definition and the following fundamental relationship

$$\mathrm{V}(X) = \mathrm{E}(X - \mathrm{E}(X))^2 = \mathrm{E}(X^2) - \mathrm{E}(X)^2. \tag{4-5}$$

So, one can actually deduce the variance of A theoretically, it is only necessary to know in addition that for independent random variables: $\mathrm{E}(XY) = \mathrm{E}(X)\,\mathrm{E}(Y)$ (which by the way then also tells us that $\mathrm{E}(A) = \mathrm{E}(X)\,\mathrm{E}(Y) = 6$)

$$\begin{aligned}
\mathrm{V}(XY) &= \mathrm{E}\left[(XY)^2\right] - \mathrm{E}(XY)^2 \\
&= \mathrm{E}(X^2)\,\mathrm{E}(Y^2) - \mathrm{E}(X)^2\,\mathrm{E}(Y)^2 \\
&= \left[\mathrm{V}(X) + \mathrm{E}(X)^2\right]\left[\mathrm{V}(Y) + \mathrm{E}(Y)^2\right] - \mathrm{E}(X)^2\,\mathrm{E}(Y)^2 \\
&= \mathrm{V}(X)\,\mathrm{V}(Y) + \mathrm{V}(X)\,\mathrm{E}(Y)^2 + \mathrm{V}(Y)\,\mathrm{E}(X)^2 \\
&= 0.01^2 \cdot 0.02^2 + 0.01^2 \cdot 3^2 + 0.02^2 \cdot 2^2 \\
&= 0.00000004 + 0.0009 + 0.0016 \\
&= 0.00250004.
\end{aligned}$$

Note, how the approximate error propagation rule actually corresponds to the two latter terms in the correct variance, while the first term – the product of the two variances is ignored. Fortunately, this term is the smallest of the three in this case. It does not always have to be like that. If you want to learn how to make a theoretical derivation of the density function for $A = XY$ then take a course in probability calculation.

Note, how we in the example actually found the "average error", that is, the error standard deviation by three different approaches:

1. The simulation based approach
2. The analytical, but approximate, error propagation method
3. A theoretical derivation

The simulation approach has a number of crucial advantages:

1. It offers a simple way to compute many other quantities than just the standard deviation (the theoretical derivations of such other quantities could be much more complicated than what was shown for the variance here)

2. It offers a simple way to use any other distribution than the normal – if we believe such better reflect reality

3. It does not rely on any linear approximations of the true non-linear relations

4.2 The parametric bootstrap

4.2.1 Introduction

Generally, a confidence interval for an unknown parameter μ is a way to express uncertainty using the sampling distribution of $\hat{\mu} = \bar{x}$. Hence, we use a distribution that expresses how our calculated value would vary from sample to sample. And the sampling distribution is a theoretical consequence of the original population distribution. As indicated, we have so far no method to do this if we only have a small sample size ($n < 30$), and the data cannot be assumed to follow a normal distribution. In principle there are two approaches for solving this problem:

1. Find/identify/assume a different and more suitable distribution for the population (”the system”)
2. Do not assume any distribution whatsoever

The simulation method called bootstrapping, which in practice is to simulate many samples, exists in two versions that can handle either of these two challenges:

1. Parametric bootstrap: simulate multiple samples from the assumed distribution.
2. Non-parametric bootstrap: simulate multiple samples directly from the data.

Actually, the parametric bootstrap handles in addition the situation where data could perhaps be normally distributed, but where the calculation of interest is quite different than the average, for example, the coefficient of variation (standard deviation divided by average) or the median. This would be an example of a nonlinear function of data – thus not having a normal distribution nor a t-distribution as a sampling distribution. So, the parametric bootstrap is basically just an example of the use of simulation as a general calculation tool, as introduced above. Both methods are hence very general and can be used in virtually all contexts.

In this material we have met a few of such alternative continuous distributions, e.g. the log-normal, uniform and exponential distributions. But if we think about it, we have not (yet) been taught how to do any statistics (confidence intervals and/or hypothesis testing) within an assumption of any of these. The

parametric bootstrap is a way to do this without relying on theoretical derivations of everything. As for the theoretical variance deduction above, there are indeed methods for doing such general theoretical derivations, which would make us able to do statistics based on any kind of assumed distribution. The most welknown, and in many ways also optimal, overall approach for this is called *maximum likelihood* theory. The general theory and approach of maximum likelihood is not covered in this course, however it is good to know that, in fact, all the methods we present are indeed also maximum likelihood methods assuming normal distributions for the population(s).

4.2.2 One-sample confidence interval for μ

Example 4.6 Confidence interval for the exponential rate or mean

Assume that we observed the following 10 call waiting times (in seconds) in a call center

$$32.6,\ 1.6,\ 42.1,\ 29.2,\ 53.4,\ 79.3,\ 2.3,\ 4.7,\ 13.6,\ 2.0.$$

If we model the waiting times using the exponential distribution, we can estimate the mean as

$$\hat{\mu} = \bar{x} = 26.08,$$

and hence the rate parameter $\lambda = 1/\beta$ in the exponential distribution as (cf. 2.48)

$$\hat{\lambda} = 1/26.08 = 0.03834356.$$

However, what if we want a 95% confidence interval for either $\mu = \beta$ or λ? We have not been tought the methods, that is, given any formulas for finding this. The following few lines of R-code, a version of the simulation based error propagation approach from above, will do the job for us:

```
## Read the data
x <- c(32.6, 1.6, 42.1, 29.2, 53.4, 79.3, 2.3 , 4.7, 13.6, 2.0)
n <- length(x)
## Set the number of simulations
k <- 100000
## 1. Simulate 10 exponentials with the right mean k times
set.seed(9876)
simsamples <- replicate(k, rexp(10,1/26.08))
## 2. Compute the mean of the 10 simulated observations k times
simmeans <- apply(simsamples, 2, mean)
## 3. Find the two relevant quantiles of the k simulated means
quantile(simmeans, c(0.025, 0.975))
```

```
    2.5%    97.5%
12.58739 44.62749
```

Explanation: `replicate` is a function that repeats the call to `rexp(10,1/26.08)`, in this case 100000 times and the results are collected in a 10×100.000 matrix. Then in a single call the 100.000 averages are calculated and subsequently the relevant quantiles found.

So the 95%-confidence interval for the mean μ is (in seconds)

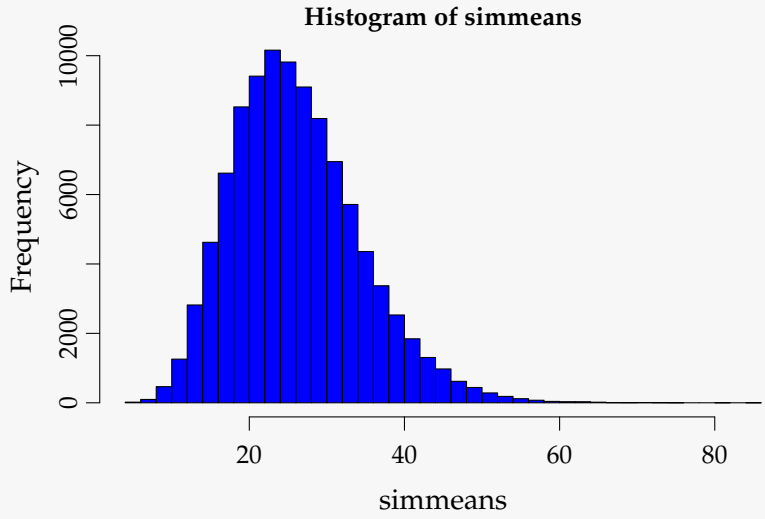
$$[12.6,\ 44.6].$$

And for the rate $\lambda = 1/\mu$ it can be found by a direct transformation (remember that the quantiles are 'invariant' to monotonic transformations, c.f. Chapter 3)

$$[1/44.6,\ 1/12.6] \Leftrightarrow [0.022,\ 0.0794].$$

The simulated sampling distribution of means that we use for our statistical analysis can be seen with the histogram:

```
hist(simmeans, col="blue", nclass=30, cex.main=0.8)
```



We see clearly that the sampling distribution in this case is not a normal nor a t-distribution: it has a clear right skewed shape. So $n = 10$ is not quite large enough for this exponential distribution to make the Central Limit Theorem take over.

The general method which we have used in the example above is given below as Method 4.7.

4.2.3 One-sample confidence interval for any feature assuming any distribution

We saw in the example above that we could easily find a confidence interval for the rate $\lambda = 1/\mu$ assuming an exponential distribution. This was so, since the rate was a simple (monotonic) transformation of the mean, and the quantiles of simulated rates would then be the same simple transformation of the quantiles of the simulated means. However, what if we are interested in something not expressed as a simple function of the mean, for instance the median, the coefficent of variation, the quartiles, Q_1 or Q_3, the IQR=$Q_3 - Q_1$ or any other quantile? Well, a very small adaptation of the method above would make that possible for us. To express that we now cover any kind of statistic one could think of, we use the general notation, the greek letter θ, for a general feature of the distribution. For instance, θ could be the true median of the population distribution, and then $\hat{\theta}$ is the sample median computed from the sample taken.

|||| Method 4.7 Confidence interval for any feature θ by parametric bootstrap

Assume we have actual observations $x_1, \ldots, x_n$ and assume that they stem from some probability distribution with density (pdf) f:

1. Simulate k samples of n observations from the assumed distribution f where the mean is set to $\bar{x}$ [a]
2. Calculate the statistic $\hat{\theta}$ in each of the k samples $\hat{\theta}_1^*, \ldots, \hat{\theta}_k^*$
3. Find the $100(\alpha/2)\%$ and $100(1-\alpha/2)\%$ quantiles for these, $q^*_{100(\alpha/2)\%}$ and $q^*_{100(1-\alpha/2)\%}$ as the $100(1-\alpha)\%$ confidence interval: $\left[q^*_{100(\alpha/2)\%}, q^*_{100(1-\alpha/2)\%}\right]$

[a] *(Footnote: And otherwise chosen to match the data as good as possible: some distributions have more than just a single mean related parameter, e.g. the normal or the log-normal. For these one should use a distribution with a variance that matches the sample variance of the data. Even more generally the approach would be to match the chosen distribution to the data by the so-called maximum likelihood approach)*

Please note again, that you can simply substitute the θ with whatever statistics that you are working with. This then also shows that the method box includes the often occuring situation, where a confidence interval for the mean μ is the aim.

|||| Example 4.8 Confidence interval for the median assuming an exponential distribution

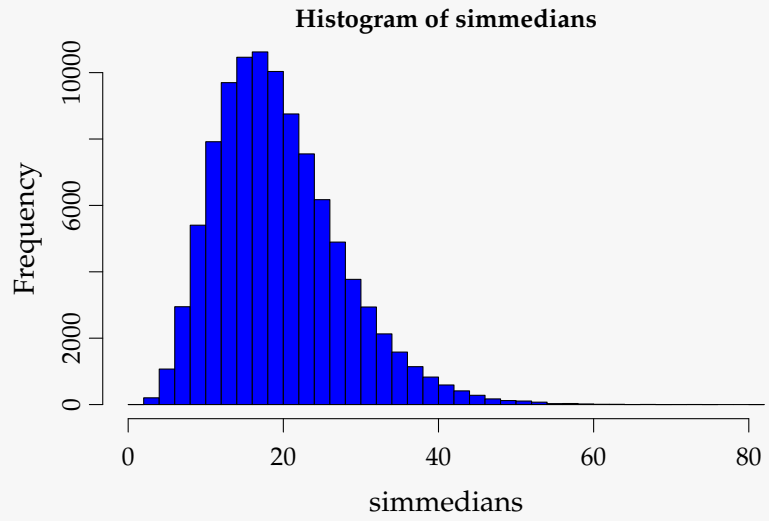
Let us look at the exponential data from the previous section and find the confidence interval for the median:

```
## Load the data
x <- c(32.6, 1.6, 42.1, 29.2, 53.4, 79.3, 2.3 , 4.7, 13.6, 2.0)
n <- length(x)
## Set the number of simulations
k <- 100000
## 1. Simulate k samples of n=10 exponentials with the right mean
simsamples <- replicate(k, rexp(n,1/26.08))
## 2. Compute the median of the n=10 simulated observations k times:
simmedians <- apply(simsamples, 2, median)
## 3. Find the two relevant quantiles of the k simulated medians:
quantile(simmedians, c(0.025, 0.975))
```

```
     2.5%     97.5%
 7.038026 38.465226
```

The simulated sampling distribution of medians that we use for our statistical analysis can be studied by the histogram:

```
hist(simmedians, col="blue", nclass=30, cex.main=0.8)
```



We see again clearly that the sampling distribution in this case is not a normal nor a t-distribution: it has a clear right skewed shape.

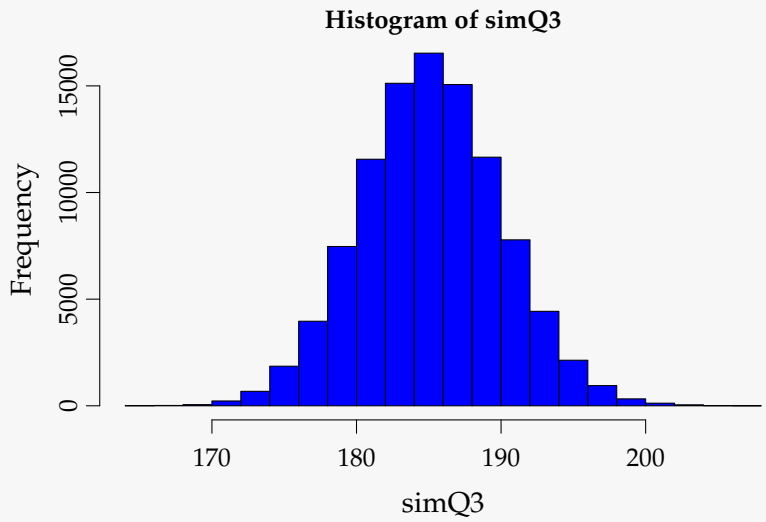
|||| Example 4.9 Confidence interval for Q_3 assuming a normal distribution

Let us look at the heights data from the previous chapters and find the 99% confidence interval for the upper quartile: (Please note that you will find NO theory nor analytically expressed method boxes in the material to solve this challenge). There is one litte extra challenge for us here, since with as well the mean as the median there were directly applicable R-functions to do these sample computations for us, namely the R-functions `mean` and `median`. The upper quartile Q_3 does not as such have its own R-function but it comes as part of the result of e.g. the `summary` function or the `quantile` function. However, in one little line of R-code, we could make such a `Q3`-function ourselves, e.g. by:
`Q3 <- function(x){ quantile(x, 0.75)}`
And now it goes exactly as before:

```
## load in the data
x <- c(168, 161, 167, 179, 184, 166, 198, 187, 191, 179)
n <- length(x)
## Set the number of simulations:
k <- 100000
## 1. Simulate k samples of n=10 normals with the right mean and variance:
set.seed(9876)
simsamples <- replicate(k, rnorm(n, mean(x), sd(x)))
## 2. Compute the Q3 of the n=10 simulated observations k times:
simQ3 <- apply(simsamples, 2, Q3)
## 3. Find the two relevant quantiles of the k simulated medians:
quantile(simQ3, c(0.005, 0.995))

   0.5%   99.5%
172.818 198.003
```

The simulated sampling distribution of upper quartiles that we use for our statistical analysis can be studied by the histogram:

```
hist(simQ3, col="blue", cex.main=0.8)
```



In this case the Q_3 of $n = 10$ samples of a normal distribution appear to be rather symmetric and nicely distributed, so maybe one could in fact use the normal distribution, also as an approximate sampling distribution in this case.

4.2.4 Two-sample confidence intervals assuming any distributions

In this section we extend what we learned in the two previous sections to the case where the focus is a comparison between two (independent) samples. We present a method box which is the natural extensions of the method box from above, comparing any kind of feature (hence including the mean comparison):

|||| Method 4.10 Two-sample confidence interval for any feature comparison $\theta_1 - \theta_2$ by parametric bootstrap

Assume we have actual observations $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ and assume that they stem from some probability distributions with density f_1 and f_2:

1. Simulate k sets of 2 samples of n_1 and n_2 observations from the assumed distributions setting the means to $\hat{\mu}_1 = \bar{x}$ and $\hat{\mu}_2 = \bar{y}$, respectively [a]
2. Calculate the difference between the features in each of the k samples $\hat{\theta}^*_{x1} - \hat{\theta}^*_{y1}, \ldots, \hat{\theta}^*_{xk} - \hat{\theta}^*_{yk}$
3. Find the $100(\alpha/2)\%$ and $100(1-\alpha/2)\%$ quantiles for these, $q^*_{100(\alpha/2)\%}$ and $q^*_{100(1-\alpha/2)\%}$ as the $100(1-\alpha)\%$ confidence interval $\left[q^*_{100(\alpha/2)\%}, q^*_{100(1-\alpha/2)\%}\right]$

[a] *(Footnote: And otherwise chosen to match the data as good as possible: some distributions have more than just a single mean related parameter, e.g. the normal or the log-normal. For these one should use a distribution with a variance that matches the sample variance of the data. Even more generally the approach would be to match the chosen distribution to the data by the so-called maximum likelihood approach)*

|||| Example 4.11 CI for the difference of two means from exponential distributed data

Let us look at the exponential data from the previous section and compare that with a second sample of $n = 12$ observations from another day at the call center

9.6, 22.2, 52.5, 12.6, 33.0, 15.2, 76.6, 36.3, 110.2, 18.0, 62.4, 10.3.

Let us quantify the difference between the two days and conclude whether the call rates and/or means are any different on the two days:

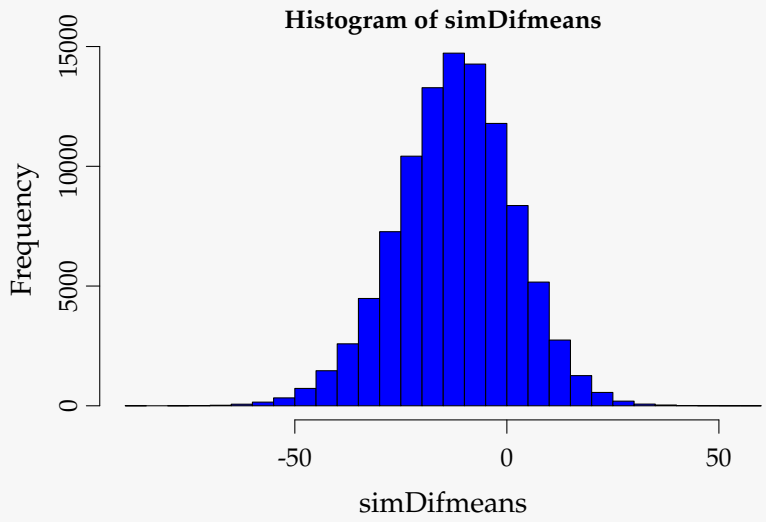
```
## Read the data
x <- c(32.6, 1.6, 42.1, 29.2, 53.4, 79.3, 2.3 , 4.7, 13.6, 2.0)
y <- c(9.6, 22.2, 52.5, 12.6, 33.0, 15.2, 76.6, 36.3, 110.2, 18.0,
       62.4, 10.3)
n1 <- length(x)
n2 <- length(y)
## Set the number of simulations
k <- 100000

## 1. Simulate k samples of each n1=10 and n2=12 exponentials
##    with the right means
simXsamples <- replicate(k, rexp(n1,1/mean(x)))
simYsamples <- replicate(k, rexp(n2,1/mean(y)))
## 2. Compute the difference between the simulated means k times
simDifmeans <- apply(simXsamples,2,mean) - apply(simYsamples,2,mean)
## 3. Find the two relevant quantiles of the k simulated differences
##    in sample means
quantile(simDifmeans, c(0.025, 0.975), cex.main=0.8)
```

```
     2.5%     97.5%
-40.73539  14.11699
```

Thus, although the mean waiting time was higher on the second day ($\bar{y} = 38.24$ s), the range of acceptable values (the confidence interval) for the difference in means is $[-40.7,\ 14.1]$ – a pretty large range and including 0, so we have no evidence of the claim that the two days had different mean waiting times (nor call rates then) based on the current data.

Let us, as in previous examples take a look at the distribution of the simulated samples. In a way, we do not really need this for doing the analysis, but just out of curiosity, and for the future it may give a idea of how far from normality the relevant sampling distribution really is:

```
hist(simDifmeans, col="blue", nclass=25, cex.main=0.8)
```



In this case the differences of means of exponential distributions appears to be rather symmetric and nicely distributed, so maybe one could in fact use the normal distribution, also as an approximate sampling distribution in this case.

|||| Example 4.12 Nutrition study: comparing medians assuming normal distributions

Let us compare the median energy levels from the two-sample nutrition data from Example 3.47. And let us do this still assuming the normal distribution as we also assumed in the previous example. First we read in the data:

```
## Read the data
xA <- c(7.53, 7.48, 8.08, 8.09, 10.15, 8.4, 10.88, 6.13, 7.9)
xB <- c(9.21, 11.51, 12.79, 11.85, 9.97, 8.79, 9.69, 9.68, 9.19)
nA <- length(xA)
nB <- length(xB)
```

Then we do the two-sample median comparison by the parametric, normal based, bootstrap:

```
## Set the number of simulations
k <- 100000
## 1. Simulate k samples of each nA=9 and nB=9 exponentials with the
## right means and standard deviations
simAsamples <- replicate(k, rnorm(nA, mean(xA), sd(xA)))
simBsamples <- replicate(k, rnorm(nB, mean(xB), sd(xB)))

## 2. Compute the difference between the simulated medians k times
simDifmedians <- apply(simAsamples, 2, median) - apply(simBsamples, 2, median)
## 3. Find the two relevant quantiles of the k simulated differences of means
quantile(simDifmedians, c(0.025, 0.975))

      2.5%      97.5%
-3.6013672 -0.3981232
```

Thus, we accept that the difference between the two medians is somewhere between 0.4 and 3.6, and confirming the group difference that we also found in the means, as the 0 is not included in the interval.

Note, how the only differences in the R code compared to the previous bootstrapping example: the calls to the `rexp`-function into calls to the `rnorm`-function and subsituting `mean` with `median`.

|||| Remark 4.13 Hypothesis testing by simulation based confidence intervals

We have also seen that even though the simulation method boxes given are providing confidence intervals: we can also use this for hypothesis testing, by using the basic relation between hypothesis testing and confidence intervals. A confidence interval includes the 'acceptable' values, and values outside the confidence interval are the 'rejectable' values.

4.3 The non-parametric bootstrap

4.3.1 Introduction

In the introduction to the parametric bootstrap section above it was discussed that another approach instead of finding the 'right' distribution to use is to not assume any distribution at all. This can be done, and a way to do this simulation based is called the *non-parametric bootstrap* and is presented in this section. The section is structured as the parametric bootstrap section above – including the similar subsections and similar method boxes. So there will be two method boxes in this section: one for the one-sample analysis and one for the two-sample analysis.

In fact, the non-parametric approach could be seen as the parametric approach but substituting the density/distribution used for the simulation by the observed distribution of the data, that is, the empirical cumulative distribution function (ecdf), cf. Chapter 1. In practice this is carried out by (re)-sampling the data we have again and again: To get the sampling distribution of the mean (or any other feature) based on the n observations that we have in our given sample, we simply again and again take new samples with n observations from the one we have. This is done "with replacement" such that the "new" samples, from now on called the *bootstrap samples* would contain some of the original observations in duplicates (or more) and others will not be there.

4.3.2 One-sample confidence interval for μ

We have the sample: $x_1, \ldots, x_n$.

The $100(1-\alpha)\%$ confidence interval for μ determined by the non-parametric bootstrap is first exemplified:

|||| Example 4.14 Women's cigarette consumption

In a study women's cigarette consumption before and after giving birth is explored. The following observations of the number of smoked cigarettes per day were observed:

before	after	before	after
8	5	13	15
24	11	15	19
7	0	11	12
20	15	22	0
6	0	15	6
20	20		

This is a typical paired t-test setup, as discussed in Section 3.2.3, which then was handled by finding the 11 differences and thus transforming it into a one-sample setup. First we read the observations into R and calculate the differences by:

```
## Read and calculate the differences for each woman before and after
x1 <-  c(8, 24, 7, 20, 6, 20, 13, 15, 11, 22, 15)
x2 <-  c(5, 11, 0, 15, 0, 20, 15, 19, 12, 0, 6)
dif <- x1-x2
dif

 [1]  3 13  7  5  6  0 -2 -4 -1 22  9
```

There is a random-sampling function in R (which again is based on a uniform random number generator): `sample`. Eg. you can get 5 repeated samples (with replacement - `replace=TRUE`) by:

```
t(replicate(5, sample(dif, replace=TRUE)))

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
[1,]   13    9   -2   -4    3   -2   -1   -1   22    22    -1
[2,]    3   22    6    3   22   -2    9    6    0     7     9
[3,]    6    6    6   13   22    7    0   -4    0    22     7
[4,]    0    9    9    3    6    9    7    9   13    -2    -1
[5,]   -1   22    6   -2    9   13   -1    6   22     0     9
```

Explanation: `replicate` is a function that repeats the call to `sample` - in this case 5 times. The function `t` simply transposes the matrix of numbers, making it 5×11

instead of 11×5 (only used for showing the figures row-wise in slightly fewer lines than otherwise necessary)

One can then run the following to get a 95% confidence interval for μ based on $k = 100000$:

```
## Number of simulated samples
k <- 100000
## Simulate
simsamples <- replicate(k, sample(dif, replace=TRUE))
## Calculate the mean of each simulated sample
simmeans <- apply(simsamples, 2, mean)
## Quantiles of the differences gives the CI
quantile(simmeans, c(0.025,0.975))

    2.5%    97.5%
1.363636 9.818182
```

Explanation: The `sample` function is called 100.000 times and the results collected in an 11×100.000 matrix. Then in a single call the 100.000 averages are calculated and subsequently the relevant quantiles found.

Note, that we use the similar three steps as above for the parametric bootstrap, with the only difference that the simulations are carried out by the resampling the given data rather than from some probability distribution.

4.3.3 One-sample confidence interval for any feature

What we have just done can be more generally expressed as follows:

|||| Method 4.15 Confidence interval for any feature θ by non-parametric bootstrap

Assume we have actual observations $x_1, \ldots, x_n$:

1. Simulate k samples of size n by randomly sampling among the available data (with replacement)
2. Calculate the statistic $\hat{\theta}$ in each of the k samples $\hat{\theta}_1^*, \ldots, \hat{\theta}_k^*$
3. Find the $100(\alpha/2)\%$ and $100(1-\alpha/2)\%$ quantiles for these, $q^*_{100(\alpha/2)\%}$ and $q^*_{100(1-\alpha/2)\%}$ as the $100(1-\alpha)\%$ confidence interval: $\left[q^*_{100(\alpha/2)\%}, q^*_{100(1-\alpha/2)\%}\right]$

|||| Example 4.16

Let us find the 95% confidence interval for the median cigarette consumption change in the example from above:

```
## The 95% CI for the median change
k <- 100000
simsamples <- replicate(k, sample(dif, replace = TRUE))
simmedians <- apply(simsamples, 2, median)
quantile(simmedians, c(0.025,0.975))
```

```
 2.5% 97.5%
   -1     9
```

4.3.4 Two-sample confidence intervals

We now have two random samples: $x_1, \ldots, x_{n_1}$ and $y_1, \ldots, y_{n_2}$. The $100(1-\alpha)\%$ confidence interval for $\theta_1 - \theta_2$ determined by the non-parametric bootstrap is defined as:

|||| Method 4.17 Two-sample confidence interval for $\theta_1 - \theta_2$ by non-parametric bootstrap

Assume we have actual observations $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$:

1. Simulate k sets of 2 samples of n_1 and n_2 observations from the respective groups (with replacement)
2. Calculate the difference between the features in each of the k samples $\hat{\theta}^*_{x1} - \hat{\theta}^*_{y1}, \ldots, \hat{\theta}^*_{xk} - \hat{\theta}^*_{yk}$
3. Find the $100(\alpha/2)\%$ and $100(1-\alpha/2)\%$ quantiles for these, $q^*_{100(\alpha/2)\%}$ and $q^*_{100(1-\alpha/2)\%}$ as the $100(1-\alpha)\%$ confidence interval: $\left[q^*_{100(\alpha/2)\%}, q^*_{100(1-\alpha/2)\%}\right]$

|||| Example 4.18 Teeth and bottle

In a study it was explored whether children who received milk from bottle as a child had worse or better teeth health conditions than those who had not received milk from the bottle. For 19 randomly selected children it was recorded when they had their first incident of caries:

bottle	age	bottle	age	bottle	Age
no	9	no	10	yes	16
yes	14	no	8	yes	14
yes	15	no	6	yes	9
no	10	yes	12	no	12
no	12	yes	13	yes	12
no	6	no	20		
yes	19	yes	13		

One can then run the following to obtain a 95 % confidence interval for $\mu_1 - \mu_2$ based on $k = 100000$:

```
## Reading in "no bottle" group
x <- c(9, 10, 12, 6, 10, 8, 6, 20, 12)
## Reading in "yes bottle" group
y <- c(14,15,19,12,13,13,16,14,9,12)

## Number of simulations
k <- 100000
## Simulate each sample k times
simxsamples <- replicate(k, sample(x, replace=TRUE))
simysamples <- replicate(k, sample(y, replace=TRUE))
## Calculate the sample mean differences
simmeandifs <- apply(simxsamples,2,mean) - apply(simysamples,2,mean)
## Quantiles of the differences gives the CI
quantile(simmeandifs, c(0.025,0.975))

      2.5%      97.5%
-6.2111111 -0.1222222
```

|||| Example 4.19

Let us make a 99% confidence interval for the difference of medians between the two groups in the tooth health example:

```
## CI for the median differences
simmediandifs <- apply(simxsamples,2,median)-apply(simysamples,2,median)
quantile(simmediandifs, c(0.005,0.995))

 0.5% 99.5%
   -8     0
```

4.4 OPTIONAL: Bootstrapping – a further perspective

This section is not part of the syllabus, but can be seen as providing some optional perspectives of this very versatile method. First of all there are actually other principles of constructing the bootstrap confidence intervals than the one presented here. What we have used is the so-called *percentile* method. Other methods to come from the non-parametric bootstrapped samples to a confidence interval are called the *bootstrap-t* or *studentized bootstrap* and *Bias Corrected Accelerated* confidence intervals. These are often considered superior to the straightforward *percentile* method presented above.

All of these bootstrap based confidence interval methods are also available in specific bootstrap functions and packages in R. There are two major bootstrap packages in R: the `boot` and the `bootstrap` packages. The former is the one recommended by the QuickR-website: (Adv. Stats, Bootstrapping) http://www.statmethods.net/advstats/bootstrapping.html.

The other package called `bootstrap` includes a function (also) called `bootstrap`. To e.g. use this one, first install this package, e.g. by:

```
## Install the bootstrap package
install.packages("bootstrap")
```

|||| Example 4.20 Teeth and bottle

Now the calculation can be performed in a single call:

```
## Calculate the 95% CI for the Teech and bottle example above
library(bootstrap)
quantile(bootstrap(dif,k,mean)$thetastar, c(0.025,0.975))

    2.5%    97.5%
1.363636 9.818182
```

These bootstrap packages are advantageous to use when looking for confidence intervals for more complicated functions of data.

4.4.1 Non-parametric bootstrapping with the `boot`-package

Now, we will show how to use the `boot`-package for more general non-parametric bootstrapping than described so far. Both the `bootstrap` and the `boot` packages require that the statistics that are bootstrapped in more general situations are on a specific functional form in R. The nice thing about the `bootstrap` package is that it may handle the simple cases without this little extra complication, whereas the `boot` package requires this for all applications, also the simple ones. However, then the `boot`-package has a number of other good features that makes it a good choice, and the point is, that for the applications coming now, the additional complexity would be needed also for the `bootstrap` package. Let us begin with a simple example that is already covered by our methods up to now:

|||| Example 4.21 Bootstrapping the mean μ by the `boot`-package

We will use again the women's cigarette consumption data:

```
## Read and calculate the differences for each woman before and after
x1 <- c(8,24,7,20,6,20,13,15,11,22,15)
x2 <- c(5,11,0,15,0,20,15,19,12,0,6)
dif <- x1-x2
```

Our aim is to find a 95%-confidence interval for the mean difference. The additional complexity mentioned is that we cannot just use the `mean` function as it comes, we need to re-define it on a specific function form, where the indices enters the function:

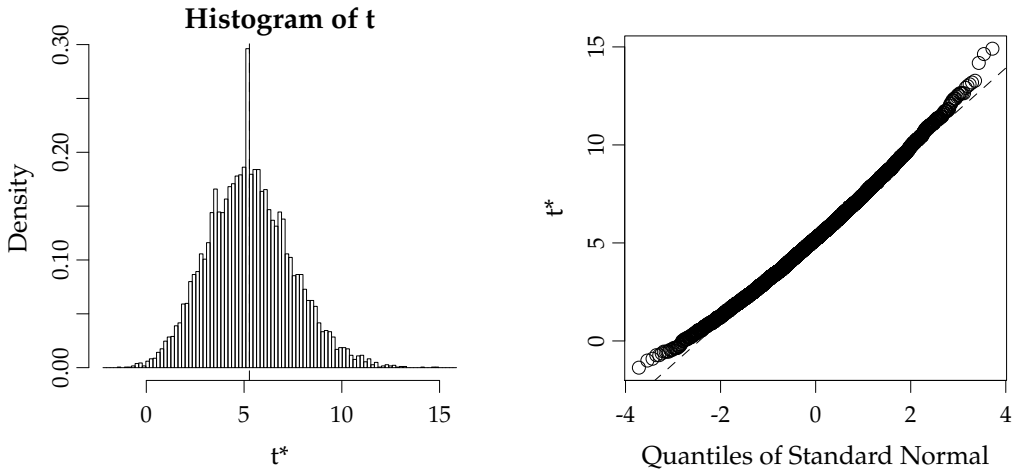
```
## Define function for calculating the mean of the d indexes
samplemean <- function(x, d){ mean(x[d]) }
```

This is a version of the mean function, where we explicitly have to specify which of the available observations should be used in the computation of the mean. Let us check the result of that:

```
## Call the new function
mean(dif)

[1] 5.272727


samplemean(dif,1:11)

[1] 5.272727


samplemean(dif,c(1,3))

[1] 5


dif

 [1]  3 13  7  5  6  0 -2 -4 -1 22  9


dif[c(1,3)]

[1] 3 7


mean(dif[c(1,3)])

[1] 5
```

We see that `samplemean(dif,c(1,3))` means that we compute the mean of observation numbers 1 and 3. Now we can use the boot package (to do what we so far already were able to using methods from previous sections) and firstly we look at the bootstrap distribution:

```
## Load the boot package
library(boot)

## Non-parametric bootstrap of the mean difference:
k <- 10000
meandifboot <- boot(dif, samplemean, k)
plot(meandifboot)
```



The actual confidence interval corresponding then to Method 4.15 can then be extracted using the dedicated R-function `boot.ci` as:

```
## Percentile bootstrap CI:
boot.ci(meandifboot, type="perc")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = meandifboot, type = "perc")

Intervals :
Level     Percentile
95%   ( 1.364,  9.818 )
Calculations and Intervals on Original Scale
```

One of the nice features of the `boot`-package is that we can now easily get instead the so-called Bias Corrected and Accelerated (bca) confidence interval simply by writing `type="bca"`:

```
## Bias Corrected Accelerated CI:
boot.ci(meandifboot, type="bca")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = meandifboot, type = "bca")

Intervals :
Level       BCa
95%   ( 1.636, 10.364 )
Calculations and Intervals on Original Scale
```

And now we can apply this bca-method to any case, e.g. bootstrapping the x1 median:

```
## Define a function for taking the median in the needed format
samplemedian <- function(x, d) {
  return(median(x[d]))
}
## Non-parametric bootstrap of the x1 median
b <- boot(x1,samplemedian,k)
boot.ci(b, type="bca")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = b, type = "bca")

Intervals :
Level       BCa
95%   ( 7, 20 )
Calculations and Intervals on Original Scale
```

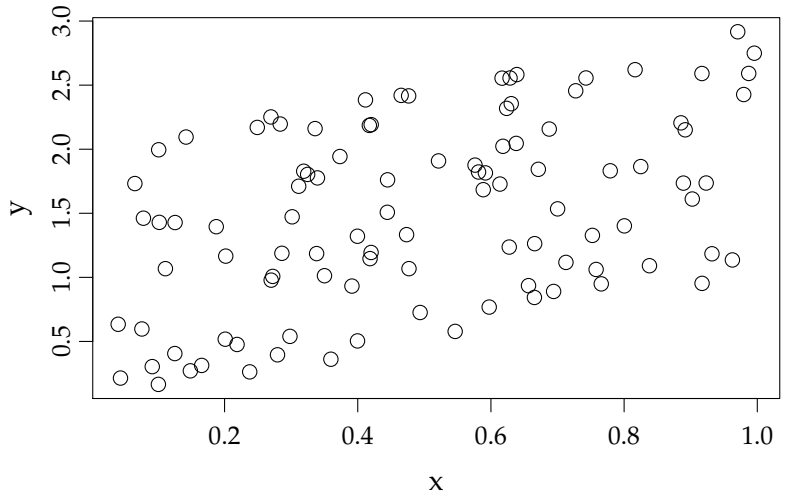
Or the coefficient of variation for the difference:

```
## Non-parametric bootstrap of the Dif coef. of var
samplecoefvar <- function(x, d) {
  return(sd(x[d])/mean(x[d]))
}
##
b <- boot(dif,samplecoefvar,k)
boot.ci(b, type="bca")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 9996 bootstrap replicates

CALL :
boot.ci(boot.out = b, type = "bca")

Intervals :
Level       BCa
95%   ( 0.811,  5.452 )
Calculations and Intervals on Original Scale
```

Now we will show how we can work directly on data frames, which in real applications always will be how we have data available. Also, we will show how we can bootstrap statistics that depend on more than a single input variable. The first example we will use is that of finding a confidence interval for a sample correlation coefficient, cf. Chapter 1 and Chapter 5. If you read through Section 5.6.1, you will see that no formula is given for this confidence interval. Actually, this is not so easily found, and only approximate explicit solutions can be found for this. We illustrate it on some data that we produce(simulate) ourselves and then store in a data frame:

|||| Example 4.22

```
## Example with data frame and two variables

## Making our own data for the example - into a data frame:
x <- runif(100)
y <- x + 2*runif(100)
D <- data.frame(x, y)
head(D)

          x        y
1 0.8917505 2.151290
2 0.1018970 1.429229
3 0.2695788 2.251663
4 0.6379553 2.045412
5 0.3358962 2.160790
6 0.6389259 2.582587


plot(D)
```




```
cor(D$x,D$y)

[1] 0.4500598
```

Then we make a version of the correlation function on the right form, where the entire data frame is taken as the input to the function (together with the index):

```
## The correlation function on the right form:
mycor  <- function(D, d) {
  E <- D[d,]
  return(cor(E$x, E$y))
}
```

```
## Check:
mycor(D, 1:100)

[1] 0.4500598


mycor(D, 1:15)

[1] 0.5629782
```

The `E` selects the chosen observations from the data frame `D`, and then in the computations we use the variables needed from the data frame (think about how this then can easily be extended to using any number of variables in the data frame). And we can now do the actual bootstrap:

```
## Doing the bootstrap on the data frame:
b <-  boot(D, mycor, 10000)
boot.ci(b, type="bca")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = b, type = "bca")

Intervals :
Level       BCa
95%   ( 0.2835,  0.5964 )
Calculations and Intervals on Original Scale
```

Our last example will show a way to bootstrap output from linear models based on bootstrapping the entire rows of the data frame. (Other bootstrap principles exist for linear models, e.g. bootstrapping only residuals).

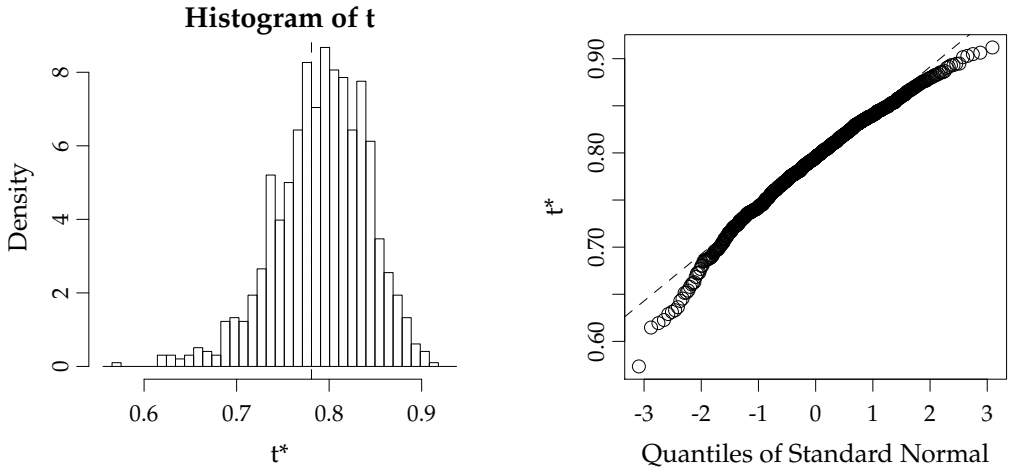
|||| Example 4.23

We will show how to find the confidence interval for the percentage of explained variation in a multiple linear regression (MLR - covered in Chapter 6). We use the data set `mtcars` from the `boot` package:

```
## Bootstrapping an R-Squared from an MLR using
## the data set mtcars from the boot package:
## (And showing how to send EXTRA stuff to your function)

# function to obtain R-Squared from the data
# AND working for ANY model fit you want!!
```

```
rsq <- function(formula, data, d) {
  fit <- lm(formula, data=data[d,])
  return(summary(fit)$r.square)
}
```

```
# bootstrapping with 1000 replications
b<- boot(mtcars, rsq, 1000, formula=mpg~wt+disp)
plot(b)
```



```
boot.ci(b, type="bca")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = b, type = "bca")

Intervals :
Level       BCa
95%   ( 0.6308,  0.8549 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
```

4.5 Exercises

|||| Exercise 4.1 Reliability: System lifetime (simulation as a computation tool)

A system consists of three components A, B and C serially connected, such that A is positioned before B, which is again positioned before C. The system will be functioning only so long as A, B and C are all functioning. The lifetime in months of the three components are assumed to follow exponential distributions with means: 2 months, 3 months and 5 months, respectively (hence there are three random variables, X_A, X_B and X_C with exponential distributions with $\lambda_A = 1/2$, $\lambda_B = 1/3$ and $\lambda_C = 1/5$ resp.) A little R-help: You will probably need (or at least it would help) to put three variables together to make e.g. a $k \times 3$-matrix - this can be done by the `cbind` function:

```
x <- cbind(xA,xB,xC)
```

And just as an example, remember from the examples in the chapter that the way to easily compute e.g. the mean of the three values for each of all the k rows of this matrix is:

```
simmeans <- apply(x, 1, mean)
```

a) Generate, by simulation, a large number (at least 1000 – go for 10000 or 100000 if your computer is up for it) of system lifetimes (hint: consider how the random variable Y = System lifetime is a function of the three X-variables: is it the sum, the mean, the median, the minimum, the maximum, the range or something even different?).

b) Estimate the mean system lifetime.

c) Estimate the standard deviation of system lifetimes.

d) Estimate the probability that the system fails within 1 month.

e) Estimate the median system lifetime

f) Estimate the 10th percentile of system lifetimes

g) What seems to be the distribution of system lifetimes? (histogram etc)

|||| Exercise 4.2 Basic bootstrap CI

(Can be handled without using R) The following measurements were given for the cylindrical compressive strength (in MPa) for 11 prestressed concrete beams:

$$38.43, 38.43, 38.39, 38.83, 38.45, 38.35, 38.43, 38.31, 38.32, 38.48, 38.50.$$

1000 bootstrap samples (each sample hence consisting of 11 measurements) were generated from these data, and the 1000 bootstrap means were arranged on order. Refer to the smallest as $\bar{x}^*_{(1)}$, the second smallest as $\bar{x}^*_{(2)}$ and so on, with the largest being $\bar{x}^*_{(1000)}$. Assume that

$$\begin{aligned}
\bar{x}^*_{(25)} &= 38.3818, \\
\bar{x}^*_{(26)} &= 38.3818, \\
\bar{x}^*_{(50)} &= 38.3909, \\
\bar{x}^*_{(51)} &= 38.3918, \\
\bar{x}^*_{(950)} &= 38.5218, \\
\bar{x}^*_{(951)} &= 38.5236, \\
\bar{x}^*_{(975)} &= 38.5382, \\
\bar{x}^*_{(976)} &= 38.5391.
\end{aligned}$$

a) Compute a 95% bootstrap confidence interval for the mean compressive strength.

b) Compute a 90% bootstrap confidence interval for the mean compressive strength.

|||| Exercise 4.3 Various bootstrap CIs

Consider the data from the exercise above. These data are entered into R as:

```
x <- c(38.43, 38.43, 38.39, 38.83, 38.45, 38.35,
       38.43, 38.31, 38.32, 38.48, 38.50)
```

Now generate $k = 1000$ bootstrap samples and compute the 1000 means (go higher if your computer is fine with it)

a) What are the 2.5%, and 97.5% quantiles (so what is the 95% confidence interval for μ without assuming any distribution)?

b) Find the 95% confidence interval for μ by the parametric bootstrap assuming the normal distribution for the observations. Compare with the classical analytic approach based on the t-distribution from Chapter 2.

c) Find the 95% confidence interval for μ by the parametric bootstrap assuming the log-normal distribution for the observations. (Help: To use the `rlnorm` function to simulate the log-normal distribution, we face the challenge that we need to specify the mean and standard deviation on the log-scale and not on the raw scale, so compute mean and standard deviation for log-transformed data for this R-function)

d) Find the 95% confidence interval for the lower quartile Q_1 by the parametric bootstrap assuming the normal distribution for the observations.

e) Find the 95% confidence interval for the lower quartile Q_1 by the non-parametric bootstrap (so without any distributional assumptions)

|||| Exercise 4.4 Two-sample TV data

A TV producer had 20 consumers evaluate the quality of two different TV flat screens - 10 consumers for each screen. A scale from 1 (worst) up to 5 (best) were used and the following results were obtained:

TV screen 1	TV screen 2
1	3
2	4
1	2
3	4
2	2
1	3
2	2
3	4
1	3
1	2

a) Compare the two means without assuming any distribution for the two samples (non-parametric bootstrap confidence interval and relevant hypothesis test interpretation).

b) Compare the two means assuming normal distributions for the two samples - without using simulations (or rather: assuming/hoping that the sample sizes are large enough to make the results approximately valid).

c) Compare the two means assuming normal distributions for the two samples - simulation based (parametric bootstrap confidence interval and relevant hypothesis test interpretation – in spite of the obviously wrong assumption).

|||| Exercise 4.5 Non-linear error propagation

The pressure P, and the volume V of one mole of an ideal gas are related by the equation $PV = 8.31T$, when P is measured in kilopascals, T is measured in kelvins, and V is measured in liters.

a) Assume that P is measured to be 240.48 kPa and V to be 9.987 L with known measurement errors (given as standard deviations): 0.03 kPa and 0.002 L. Estimate T and find the uncertainty in the estimate.

b) Assume that P is measured to be 240.48kPa and T to be 289.12K with known measurement errors (given as standard deviations): 0.03kPa and 0.02K. Estimate V and find the uncertainty in the estimate.

c) Assume that V is measured to be 9.987 L and T to be 289.12 K with known measurement errors (given as standard deviations): 0.002 L and 0.02 K. Estimate P and find the uncertainty in the estimate.

d) Try to answer one or more of these questions by simulation (assume that the errors are normally distributed).

Glossaries

Maximum likelihood [Estimator baseret på maximum likelihood metoden] 13

Non-parametric (test) [Ikke-parametriske (tests)] 24

Acronyms

ANOVA Analysis of Variance *Glossary:* Analysis of Variance

cdf cumulated distribution function *Glossary:* cumulated distribution function

CI confidence interval *Glossary:* confidence interval

CLT Central Limit Theorem *Glossary:* Central Limit Theorem

IQR Inter Quartile Range *Glossary:* Inter Quartile Range

LSD Least Significant Difference *Glossary:* Least Significant Difference

pdf probability density function *Glossary:* probability density function

```
Error in options(digits = digitsKeep):  object 'digitsKeep' not found
```