Statistics with R

Lecture notes

Niels Richard Hansen
November 16, 2012

Contents

Preface

These lecture notes were collected over the years 2007-2011 for a course on statistics for Master students in Bioinformatics or eScience at the University of Copenhagen. The present introduction to statistics is a spin-off from a set of more theoretical notes on probability theory and statistics. For the interested, they can found here:

`http://www.math.ku.dk/~richard/courses/StatScience2011/notes.pdf`

The present set of notes is designed directly for seven weeks of teaching, and is no more than a commented version of slides and R-programs used in the course. The notes are supposed to be supplemented with Peter Dalgaards book, *Introductory Statistics with R* (ISwR), second edition. That book covers in detail how to use R for doing standard statistical computations.

Each chapter in the notes corresponds to a week and each section corresponds to a lecture begining with a list of keywords and a reference to corresponding material in ISwR.

Chapter 1

First Week

1.1 R and neuron interspike times

Keywords: Densities, Neuron interspike times, The exponential distribution, The R language.

ISwR: 1-39

1.1.1 The R language is

- a giant calculator,
- a programming language for statistics,
- a huge library of packages for statistical computations and a lot more,
- a scripting and plotting system,
- expressive and fast to develop in for small to medium sized projects.

The R language was developed specifically for statistical data analysis and is today used by millions. It is just another interpreted language. An obvious alternative is Python, which is the preferred interpreted language by many computer scientists.

R has become the *de facto* standard for statistical computing in academia and many new methods in statistics are available for the user as R packages. There are widely accepted standards for data structure and method interfaces, which makes it easier to expand ones toolbox and include new methods in a computational pipeline.

You will need

- the R distribution itself, `http://www.r-project.org/`,
- preferably an integrated development environment (IDE), and RStudio is recommended, `http://www.rstudio.org/`,
- and if you want to use Sweave or Knitr you will need a working LaTeX installation.

On Windows and Mac the R distribution comes with a "GUI", which does do the job as an IDE for some purposes. On GNU/Linux you just get the command prompt and need at least an editor. A number of editors support the R language.

This author has used Emacs with ESS (Emacs Speaks Statistics) for the daily work for a number of years, but it is mostly a matter of taste and habit. The recent RStudio IDE looks as a promising alternative.

The fundamental concepts that you need to learn are:

- Data structures such as *vectors*, *lists* and *data frames*.
- Functions and methods in three categories:
  - mathematical functions to transform data and express mathematical relations,
  - statistical functions to *fit* statistical models to data,
  - and graphical functions for visualization.
- Programming constructs.

1.1.2 R interlude: Data structures

A fundamental data structure is a vector. In fact, there is not a more fundamental data structure. If you want just a single number stored in R it is, in fact, a vector of length 1. This is good and bad.

```
x <- 1.0        ## Being explicit, we want a double, not an integer
typeof(x)

## [1] "double"

class(x)

## [1] "numeric"

is.vector(x)

## [1] TRUE

length(x)

## [1] 1


y <- 2L         ## Being explicit, we want an integer, not an double
typeof(y)

## [1] "integer"
```

```
class(y)

## [1] "integer"

is.vector(y)

## [1] TRUE
```

One of the advantages of R for rapid programming is that R is not a strongly typed language, and you can do loads of things without having to worry much about types, and there is a range of automatic type casting going on behind the scene. You can, for instance, easily concatenate the two vectors above.

```
z <- c(x, y)
typeof(z)

## [1] "double"

class(z)

## [1] "numeric"

z[1]                ## The first entry, equals x

## [1] 1

z[2]                ## The second entry, equals y but now of class numeric.

## [1] 2

identical(z[1], x)

## [1] TRUE

identical(z[2], y)  ## This is FALSE, because of the difference in class.

## [1] FALSE

length(z)

## [1] 2
```

All mathematical functions and arithmetic operators are automatically vectorized, which means that they are automatically applied entry by entry when called with vectors of length > 1 as arguments.

```
z + z
```

```
## [1] 2 4
```

```
exp(z)
```

```
## [1] 2.718 7.389
```

```
cos(z)
```

```
## [1]  0.5403 -0.4161
```

There are many built-in functions for easy computations of simple statistics.

```
mean(z)
```

```
## [1] 1.5
```

```
var(z)
```

```
## [1] 0.5
```

For storing data you will quickly need a data frame, which can be best thought of as a rectangular data structure where we have one observation or one case per row and each column represents the different measurements made per case (the variables).

```
x <- data.frame(height = c(1.80, 1.62, 1.96),
                weight = c(89, 57, 82),
                gender = c("Male", "Female", "Male")
                )
```

There is a range of methods available for accessing different parts of the data frame:

```
x
```

```
##   height weight gender
## 1   1.80     89   Male
## 2   1.62     57 Female
## 3   1.96     82   Male
```

```
dim(x)        ## Number of rows and number of columns
```

```
## [1] 3 3
```

```
x[1, ]        ## First row, a data frame of dimensions (1, 3)
```

```
##   height weight gender
## 1    1.8     89   Male
```

```
x[, 1]        ## First column, a numeric of length 3

## [1] 1.80 1.62 1.96

x[, "height"] ## As above, but column selected by name

## [1] 1.80 1.62 1.96

x$height      ## As above, but column selected using the '$' operator

## [1] 1.80 1.62 1.96

x[1:2, ]      ## First two rows, a data frame of dimensions (2, 3)

##   height weight gender
## 1   1.80     89   Male
## 2   1.62     57 Female

x[, 1:2]      ## First two columns, a data frame of dimensions (2, 3)

##   height weight
## 1   1.80     89
## 2   1.62     57
## 3   1.96     82

x[, c("height", "gender")]  ## As above, but columns selected by name

##   height gender
## 1   1.80   Male
## 2   1.62 Female
## 3   1.96   Male
```

A very useful technique for dealing with data is to be able to filter the data set and extract a subset of the data set fulfilling certain criteria.

```
subset(x, height > 1.75)

##   height weight gender
## 1   1.80     89   Male
## 3   1.96     82   Male

subset(x, gender == "Female")

##   height weight gender
## 2   1.62     57 Female

subset(x, height < 1.83 & gender == "Male")
```

```
##   height weight gender
## 1    1.8     89   Male
```

What formally happens to the second argument above, the ”filter”, is a little tricky, but the unquoted name tags `height` and `gender` refer to the columns with those names in the data frame `x`. These columns are extracted from `x` and the logical expressions are evaluated row-by-row. The result is a data frame containing the rows where the evaluation of the logical expression is TRUE. An equivalent result can be obtained ”by hand” as follows.

```
rowFilter <- x$height < 1.83 & x$gender == "Male"
length(rowFilter)

## [1] 3

head(rowFilter)

## [1]  TRUE FALSE FALSE

class(rowFilter)

## [1] "logical"

x[rowFilter, ]

##   height weight gender
## 1    1.8     89   Male

x[x$height < 1.83 & x$gender == "Male", ]  ## A "one-liner" version

##   height weight gender
## 1    1.8     89   Male
```

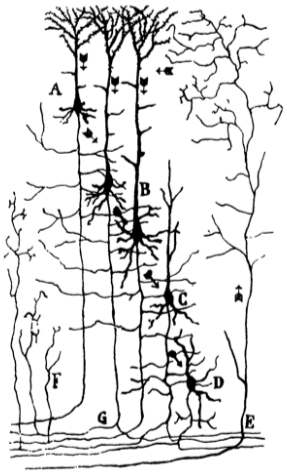
The only benefit of `subset` is that you don't have to write `x$` twice. With a longer, more meaningful, name of the data frame this is actually a benefit, and it gives a more readable filter.

The next data structure to consider is that of lists. A list is a general data container with entries that can be anything including other lists.

```
x <- list(height = list(Carl = c(1.69, 1.75, 1.80),
                        Dorthe = c(1.56, 1.62),
                        Jens = 1.96),
          weight = list(Carl = c(67, 75, 89),
                        Dorthe = c(52, 57),
                        Bent = c(74, 76),
                        Jens = 82)
          )

x[1]            ## First entry in the list, a list of lists
```

```
## $height
## $height$Carl
## [1] 1.69 1.75 1.80
##
## $height$Dorthe
## [1] 1.56 1.62
##
## $height$Jens
## [1] 1.96

x["height"]     ## As above, but entry selected by name

## $height
## $height$Carl
## [1] 1.69 1.75 1.80
##
## $height$Dorthe
## [1] 1.56 1.62
##
## $height$Jens
## [1] 1.96

x[[1]]          ## First entry in the list, a list

## $Carl
## [1] 1.69 1.75 1.80
##
## $Dorthe
## [1] 1.56 1.62
##
## $Jens
## [1] 1.96

x[["height"]]   ## As above, but entry selected by name

## $Carl
## [1] 1.69 1.75 1.80
##
## $Dorthe
## [1] 1.56 1.62
##
## $Jens
## [1] 1.96
```

1.1.3 Neuron interspike times



We measure the times between *spikes* of a neuron in a steady state situation.

The figure is from Cajal, S. R. (1894), *Les nouvelles idées sur la structure du systéme nerveux chez l'homme et chez les vertébrés*. It illustrates five neurons in a network from the Cerebral Cortex.

Neuron cells in the brain are very well studied and it is known that neurons transmit electrochemical signals. Measurements of a cells membrane potential show how the membrane potential can activate voltage-gated ion channels in the cell membrane and trigger an electrical signal known as a spike.

At the most basic level it is of interest to understand the interspike times, that is, the times between spikes, for a single neuron in a steady state situation. The interspike times behave in an intrinsically stochastic manner meaning that if we want to describe the typical interspike times we have to rely on a probabilistic description.

A more ambitious goal is to relate interspike times to external events such as visual stimuli and another goal is to relate the interspike times of several neurons.

Neuron interspike times

What scientific objectives can the study of neuron interspike times have?

- The single observation is hard to predict and is a consequence of the combined behavior of a complicated system – yet the combined *distribution* of interspike times encode information about the system.
- Discover characteristics of the *collection* of measurements.
- Discover *differences* in characteristics that reflect underlying differences in the biophysics and biochemistry.

Neuron interspike times

We attempt to model the interspike times using the exponential distribution, which is the probability distribution on $[0, \infty)$ with *density*

$$f_\lambda(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

where $\lambda > 0$ is an *unknown parameter.*

The interpretation is that the *probability* of observing an interspike time in the interval $[a, b]$ for $0 \leq a < b$ is

$$\begin{aligned} P([a,b]) &= \int_a^b \lambda e^{-\lambda x} \mathrm{d}x \\ &= \underline{\qquad\qquad\qquad} \end{aligned}$$

There is one technical detail. The probability of the entire positive halfline $[0, \infty)$ should equal 1. This is, indeed, the case,

$$\int_0^\infty \lambda e^{-\lambda x} \mathrm{d}x = -e^{-\lambda x}\Big|_0^\infty = 1.$$

For the last equality we use the convention $e^{-\infty} = 0$ together with the fact that $e^0 = 1$.

Mean

The *mean* of the theoretical exponential distribution with parameter $\lambda > 0$ is

$$\mu = \int_0^\infty x\lambda e^{-\lambda x} \mathrm{d}x = \frac{1}{\lambda}$$

The *empirical mean* is simply the average of the observations in a data set $x_1, \ldots, x_n$

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i.$$

Equating the theoretical mean equal to the empirical mean gives us an *estimate* of λ,

$$\hat{\lambda} = \frac{1}{\hat{\mu}}.$$

The derivation of the formula above for the mean is by partial integration

$$\begin{aligned} \mu &= \int_0^\infty x\lambda e^{-\lambda x} \mathrm{d}x \\ &= xe^{-\lambda x}\Big|_0^\infty + \int_0^\infty e^{-\lambda x} \mathrm{d}x \\ &= -\frac{1}{\lambda} e^{-\lambda x}\Big|_0^\infty = \frac{1}{\lambda}. \end{aligned}$$

The idea of equating an empirical, observable quantity equal to a theoretical quantity and then solve for unknowns is as old as quantitative sciences, and yet the idea is one of the fundamental ideas in statistics. There are, however, many equations and many observables (or quantities that are directly computable from observables), so which to choose? One of the noble objectives of statistics is to make the ad hoc procedure less ad hoc and more principled, and, what is more important, to give us the tools to understand the merits of the procedure – surely, the value of the parameter we have computed is an approximation, and *estimate*, but how good an approximation is it? What would, for instance, happen, if we were to repeat the experiment?

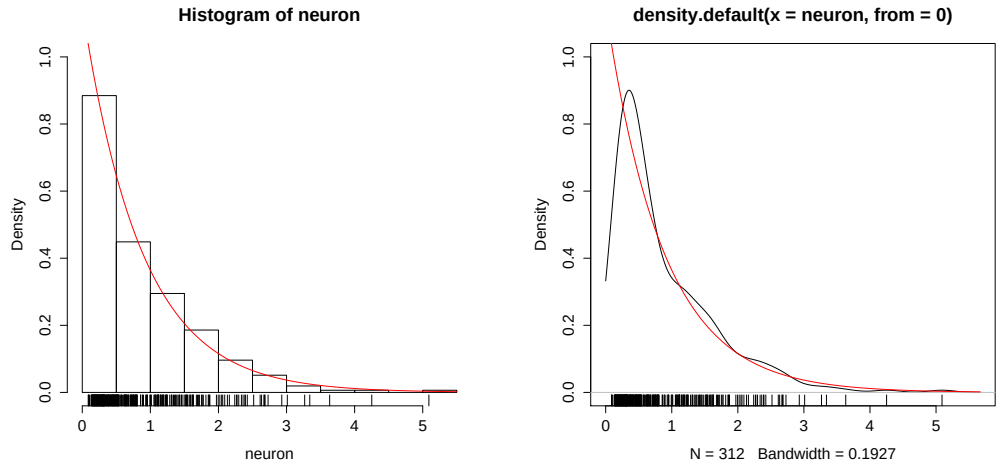


Figure 1.1: Histogram (left) and kernel density estimate (right) for a data set with 312 neuron interspike times. The actual data points are included as small marks at the x-axis. The estimated density for the exponential distribution is superimposed (red). The estimate of λ is $\hat{\lambda} = 1.147$

Model control

The *histogram* or *kernel density estimate* is a direct estimate, $\hat{f}$, of the *density*, f, of the distribution of interspike times without a specific (parametric) model assumption.

To check if the exponential distribution is a good model we can compare the density of the estimated exponential distribution

$$\hat{\lambda} e^{-\hat{\lambda} x}$$

to the histogram or kernel density estimate $\hat{f}$.

We will see other methods later in the course that are more powerful for the comparison of a theoretical distribution and a data set.

1.1.4 R interlude: First view on neuron data

Reading data in as a vector.

```
neuron <- scan("http://www.math.ku.dk/~richard/courses/StatScience2011/neuronspikes.txt")
```

Printing the entire data vector is rarely useful, but it may be useful to print the head or tail of the vector or compute some summary information.

```
head(neuron)

## [1] 0.08850 0.08985 0.09330 0.10650 0.13005 0.13245

tail(neuron)

## [1] 3.011 3.266 3.341 3.633 4.251 5.090

summary(neuron)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.088   0.314   0.585   0.872   1.220   5.090

length(neuron)

## [1] 312
```

Some technical information on the R object `neuron` that holds the data can also be useful.

```
typeof(neuron)

## [1] "double"

class(neuron)

## [1] "numeric"
```

The next thing to do is to try to visualize the distribution of the interspike times of neurons. A classical plot is the histogram. Here with the actual observations added on the x-axis as a rug-plot.

```
hist(neuron, prob = TRUE)  ## 'prob = TRUE'  makes the histogram comparable
rug(neuron)                ## to a density.
```

An alternative to a histogram is a kernel density estimate. Here we explicitly specify the left-most end point as 0 because interspike times cannot become negative. See Figure 1.1.

```
neuronDens <- density(neuron, from = 0)
plot(neuronDens, ylim = c(0, 1))
rug(neuron)
```

The function `plot` above is a so-called generic function. What it does depends on the class of the object it takes as first argument.

```
class(neuronDens)

## [1] "density"
```

The object is of class `density`, which means that an appropriate method for plotting objects of this class is called when you call `plot` above. In a technical jargon this is S3 object orientation in R. It is not something that is important for learning R in the first place, but it does explain that `plot` seems to magically adapt to plotting many different "things" correctly.

We compute the ad hoc estimator of λ and add the resulting estimated density for the exponential distribution to the plot of the kernel density estimate. See Figure 1.1.

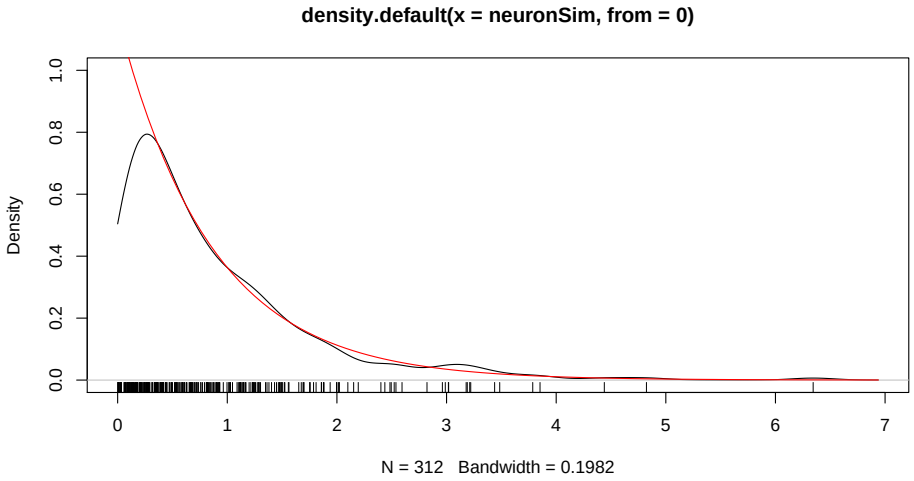
```
lambdaHat <- 1 / mean(neuron)
plot(neuronDens, ylim = c(0, 1))
rug(neuron)
curve(lambdaHat * exp(- lambdaHat * x), add = TRUE, col = "red")
```

We can also turn back to the histogram and add the estimated density for the exponential distribution to the histogram. See Figure 1.1.

```
hist(neuron, prob = TRUE, ylim = c(0, 1))
rug(neuron)
curve(lambdaHat * exp(- lambdaHat * x), add = TRUE, col = "red")
```

The histogram shows no obvious problem with the model, but using the kernel density estimate it seems that something is going on close to 0. To get an idea about whether this is due to problems with the fit or a peculiarity of the `density` estimate because we truncate at 0 we make a simulation.

```
neuronSim <- rexp(n = length(neuron),
                  rate = lambdaHat)

plot(density(neuronSim, from = 0), ylim = c(0, 1))
rug(neuronSim)
lambdaHat2 <- 1 / mean(neuronSim)
curve(lambdaHat2 * exp(- lambdaHat2 * x), add = TRUE, col = "red")
```



It does seem from this figure that even with simulated exponentially distributed data the kernel density estimate will decrease close to 0 as opposed to the actual density.

1.2 Continuous distributions

Keywords: Continuous probability distributions, density, descriptive methods, distribution function, frequencies, normal distribution, quantiles.

ISwR: 55-75

Exponential neuron interspike time model

With the exponential distribution as a model of the interspike times for neurons

- the parameter λ is called the *intensity* or *rate* parameter. A *large* λ corresponds to a *small* theoretical mean of the interspike times,
- and since

$$P([0, x]) \quad = \quad 1 - e^{-\lambda x}$$

is *increasing* as a function of λ for fixed x the *larger* λ is the *larger* is the probability of observing an interspike time smaller than a given x.

Thus λ controls the frequency of neuron spikes in this model.

Exponential neuron interspike time model

If two different neurons share the exponential distribution as a model but with different values of λ, the λ parameters carry insights on the differences – the neurons are not just different, but the difference is succinctly expressed in terms of the difference of the λ's.

If we estimate λ for the two neurons we can try to answer questions such as

- Which neuron is the slowest to fire?
- Are the neurons actually different, and if so how does that manifest itself in terms of the model?

The distribution function

Recall that for the exponential distribution on $[0, \infty)$ with density $\lambda e^{-\lambda x}$ the probability of $[0, x]$ is, as a function of x,

$$F(x) = \int_0^x \lambda e^{-\lambda x} \mathrm{d}x = 1 - e^{-\lambda x}.$$

We call F the *distribution function*. It is

- a function from $[0, \infty)$ into $[0, 1]$,
- monotonely increasing,
- tends to 1 when x tends to $+\infty$ (becomes large).

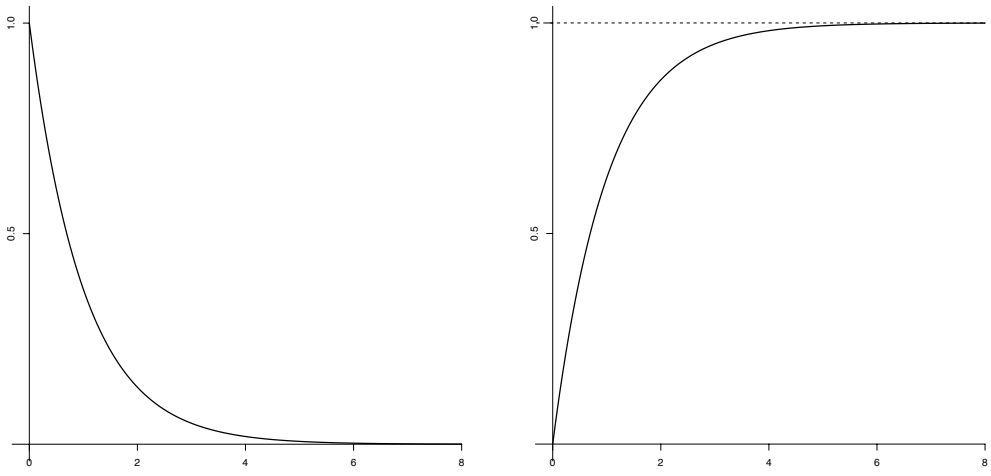


Figure 1.2: The density (left) and the distribution function (right) for the exponential distribution with intensity parameter $\lambda = 1$.

Empirical distribution function

For our neuron data set $x_1, \ldots, x_n$ we can order the observations in increasing order

$$x_{(1)} \leq \ldots \leq x_{(n)}$$

and define the *empirical distribution function* as

$$\hat{F}(x) = \frac{i}{n}$$

if $x_{(i)} \leq x < x_{(i+1)}$. Thus $\hat{F}(x)$ is the relative frequency of observations in the data set smaller than or equal to x. The *frequency interpretation* of probabilities says that the relative frequency is approximately equal to the theoretical probability if n is large.

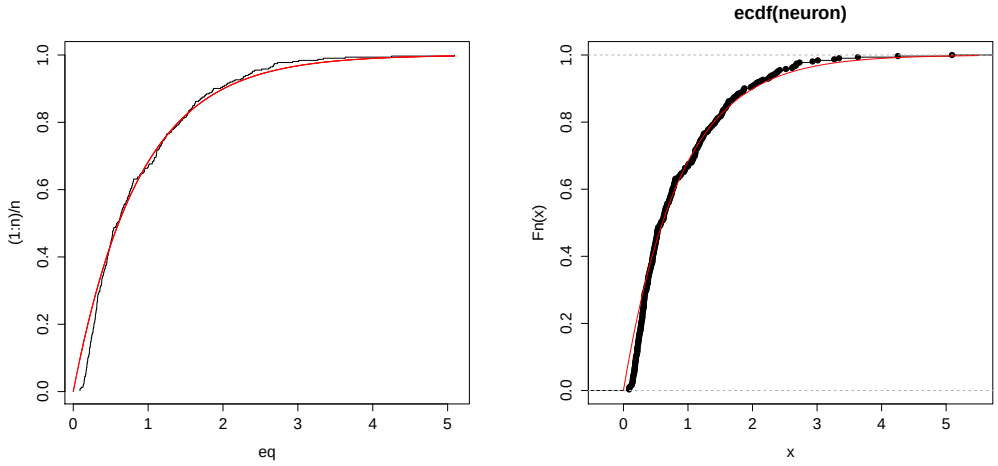


Figure 1.3: The empirical distribution function (black) for the neuron data plotted in R using `plot` with `type = "s"` (left) and using the `ecdf` function (right). The estimated distribution function for the exponential distribution is superimposed (red).

The probability model is an *idealized object*, whose real world manifestations are relative frequencies. We only "see" the distribution when we have an entire data set – a single observation will reveal almost nothing about the distribution. With a large data set we see the distribution more clearly (the approximation becomes better) than with a small data set.

The frequency interpretation is one interpretation of probabilities among several. The most prominent alternative is the subjective Bayesian. We have more to say on this later in the course. What is important to know is that, disregarding the interpretation, the mathematical theory of probability is completely well founded and well understood. How probability models should be treated when they meet data, that is, how we should do statistics, is a little more tricky. There are two major schools, the frequentistic and the Bayesian. We will focus on methods based on the frequency interpretation in this course.

Note that an alternative way to write the empirical distribution function without reference to the ordered data is as follows:

$$\hat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} 1(x_i \leq x).$$

Here $1(x_i \leq x)$ is an *indicator*, which is 1 if the condition in the parentheses holds and 0 otherwise. Thus the sum counts how many observations are smaller than x.

Quantiles

The distribution function for the exponential distribution gives the probability of intervals $[0, x]$ for any x. Sometimes we need to answer the opposite question:

For which x is the probability of getting an observation smaller than x equal to 5%?

If $q \in [0, 1]$ is a probability (5%, say) we need to solve the equation

$$1 - e^{-\lambda x} = q$$

in terms of x. The solution is

$$x_q = -\frac{1}{\lambda} \log(1 - q).$$

We call x_q the q-quantile for the exponential distribution with parameter $\lambda > 0$.

If $q = 1$ we allow, in principle, for the quantile $+\infty$, but generally we have no interests in quantiles for the extreme cases $q = 0$ and $q = 1$.

The quantile function

We call the function

$$F^{-1}(q) = -\frac{1}{\lambda} \log(1 - q)$$

the *quantile function* for the exponential distribution. It is the inverse of the distribution function.

Empirical Quantiles

If we order the observations in our neuron interspike time data set in increasing order

$$x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(n)}$$

with $n = 312$ there is a fraction of i/n observations smaller than or equal to $x_{(i)}$.

We will regard $x_{(i)}$ as an empirical approximation of the $\frac{i-0.5}{n}$-quantile for $i = 1, \ldots, n$.

QQ-plot

A *QQ-plot* of the data against the theoretical exponential distribution is a plot of

$$\left(F^{-1}\left(\frac{i - 0.5}{n} \right), x_{(i)} \right),$$

and if the data are exponentially distributed with distribution function F, these points should be close to a straight line. Since

$$F^{-1}(q) = -\frac{1}{\lambda} \log(1 - q)$$

changing λ only affects the slope of the line, and we typically use the plot with $\lambda = 1$.

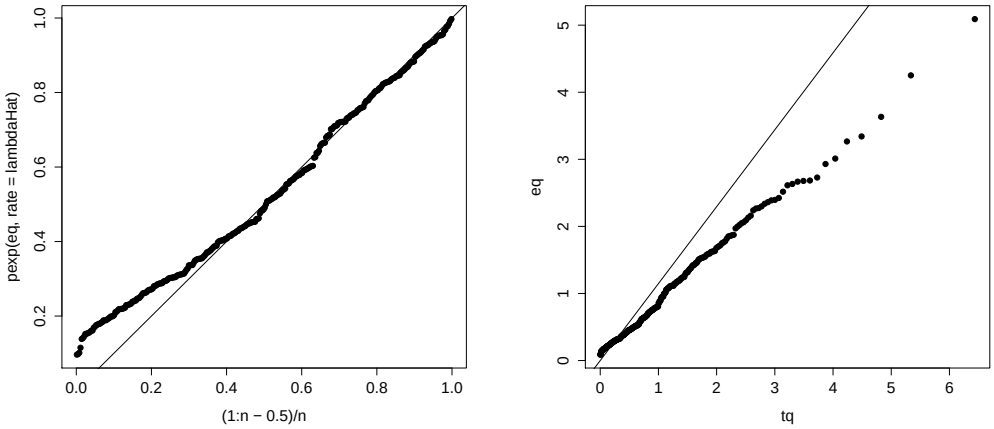


Figure 1.4: Neuron data model control. The PP-plot (left) and the QQ-plot (right) both show problems with the model. On the PP-plot we see this as deviations from the straight line in the bottom left. The QQ-plot is in itself OK in the sense that the points fall reasonably well on a straight line, but for the exponential model the line added with slope $\hat{\lambda}$ and intercept 0 does not match the points. We will follow up on these issues later in the course. For now, we just note that the exponential distribution may not be a perfect fit to the data.

QQ-plots, or Quantile-Quantile plots, are very useful as a tool for visually inspecting if a distributional assumption, such as the exponential, holds. We usually don't plug in the estimated λ but simply do the plot taking $\lambda = 1$. We then focus on whether the points fall approximately on a straight line or not. This is called model control and is important for justifying distributional assumptions. As we will see later, the parameter λ is an example of a scale parameter. One of the useful properties of QQ-plots, as we observed explicitly for the exponential distribution, is that if the distributional model is correct up to a scale- and location-transformation then the points in the QQ-plot should fall approximately on a straight line.

1.2.1 R interlude: Neuron model control.

The empirical distribution function. See Figure 1.3.

```
neuron <- scan("http://www.math.ku.dk/~richard/courses/StatScience2011/neuronspikes.txt")
n <- length(neuron)
lambdaHat <- 1/mean(neuron)
eq <- sort(neuron)              ## Ordered observations (empirical quantiles)
plot(eq, (1:n)/n, type = "s")  ## Plots a step function
curve(1 - exp(- lambdaHat*x), from = 0, add = TRUE, col = "red")
```

An alternative is to use the `ecdf` function. The resulting plot emphasizes the step nature of the function, and the `ecdf` object itself is a function that can be used for other purposes. See Figure 1.3.

```
neuronEcdf <- ecdf(neuron)
plot(neuronEcdf)
curve(1 - exp(- lambdaHat*x), from = 0,
      add = TRUE, col = "red")
plot(neuronEcdf, seq(1, 1.5, 0.01)) ## Zooming in
```

QQ- and PP-plots. See Figure 1.4

```
tq <- qexp((1:n - 0.5)/n)   ## Theoretical quantiles
plot(tq, eq, pch = 19)
abline(0, lambdaHat)

plot((1:n - 0.5)/n, pexp(eq, rate = lambdaHat),
     pch = 19, ylim = c(0,1))
abline(0, 1)
```

1.2.2 Transformations and simulations

Transformations and the uniform distribution

If $q \in [0,1]$ and we observe x from the exponential distribution with parameter $\lambda > 0$, what is the probability that $F(x)$ is smaller than q?

Since $F(x) \leq q$ if and only if $x \leq F^{-1}(q)$, the probability equals

$$P([0, F^{-1}(q)]) = F(F^{-1}(q)) = q.$$

We have derived that by *transforming* an exponential distributed observation using the distribution function we get an observation in $[0,1]$ with distribution function

$$G(q) = q.$$

This is the *uniform distribution* on $[0,1]$.

We can observe that for $q \in [0,1]$

$$G(q) = \int_0^q 1 \, \mathrm{d}x$$

and the function that is constantly 1 on the interval $[0,1]$ (and 0 elsewhere) is the *density* for the uniform distribution on $[0,1]$.

The uniform distribution, quantiles and transformations

The process can be inverted. If u is an observation from the uniform distribution then $F^{-1}(u) \leq x$ if and only if $u \leq F(x)$, thus the probability that $F^{-1}(u) \leq x$ equals the probability that $u \leq F(x)$, which for the uniform distribution is

$$P([0, F(x)]) = F(x).$$

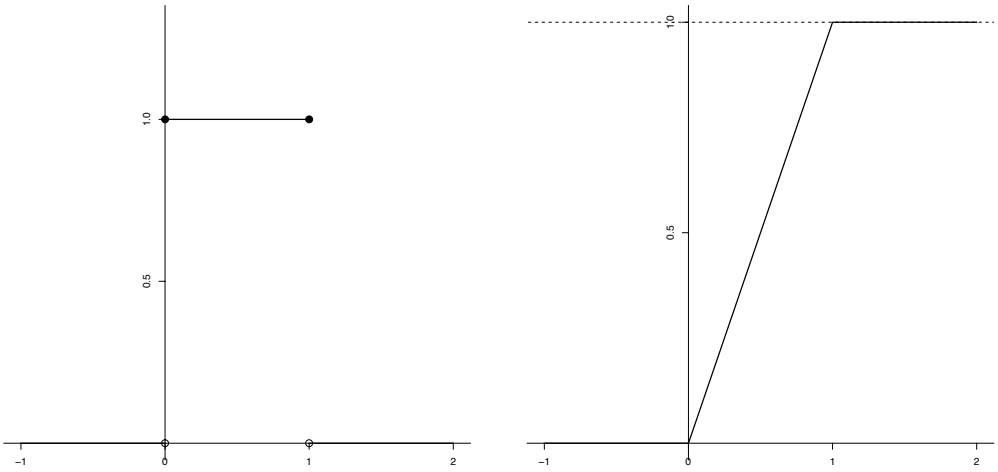


Figure 1.5: The density (left) and the distribution function (right) for the uniform distribution on the interval $[0, 1]$.

Thus the *transformed* observation

$$-\frac{1}{\lambda}\log(1-u)$$

has an exponential distribution with parameter $\lambda > 0$.

We rarely observe a uniformly distributed variable in the real world that we want to transform to an exponentially distributed variable, but the idea is central to computer simulations.

Simulations

- Computer simulations of random quantities has become indispensable as a supplement to theoretical investigations and practical applications.
- We can easily investigate a large number of scenarios and a large number of replications.
- The computer becomes an *in silico* laboratory where we can experiment.
- We can investigate the behavior of methods for statistical inference.
- But how can the deterministic computer generate the outcome from a probability distribution?

Generic simulation

Two step procedure behind random simulation:

- The computer *emulates* the generation of variables from the uniform distribution on the unit interval $[0, 1]$.
- The emulated uniformly distributed variables are by *transformation* turned into variables with the desired distribution.

The real problem

What we need in practice is thus the construction of a transformation that can transform the uniform distribution on $[0, 1]$ to the desired probability distribution.

In this course we cover two cases

- A general method for discrete distributions (next week)
- A general method for probability distributions on $\mathbb{R}$ given in terms of the distribution function.

But what about ...

... the simulation of the *uniformly distributed* variables?

That's a completely different story. Read D. E. Knuth, ACP, Chapter 3 or trust that R behaves well and that `runif` works correctly.

We rely on a sufficiently good *pseudo random number generator* with the property that as long as we cannot *statistically* detect differences from what the generator produces and "true" uniformly distributed variables, then we live happily in ignorance.

Exercise: Simulations

Write a function, `myRexp`, that takes two arguments such that

```
myRexp(10, 1)
```

generates 10 variables with an exponential distribution with parameter $\lambda = 1$.

How do you make the second argument equal to 1 by default such that

```
myRexp(10)
```

produces the same result?

1.3 Exercises

Exercise 1.1 We saw in the lecture that the exponential distribution has density $\lambda e^{-\lambda x}$. The Γ-function (gamma function) is defined by the integral

$$\Gamma(\lambda) = \int_0^\infty x^{\lambda-1} e^{-x} \mathrm{d}x$$

for $\lambda > 0$. The Γ-distribution with shape parameter $\lambda > 0$ has density

$$f(x) = \frac{1}{\Gamma(\lambda)} x^{\lambda-1} e^{-x}$$

for $x > 0$. The density and distribution function are available in R as `dgamma` and `pgamma`, respectively. Construct a figure showing the densities for the Γ-distribution for $\lambda = 1, 2, 5, 10$. Construct another figure showing the distribution functions for $\lambda = 1, 2, 5, 10$

The Γ-distribution can be given a *scale parameter* $\beta > 0$ in which case the density becomes

$$f(x) = \frac{1}{\beta^\lambda \Gamma(\lambda)} x^{\lambda-1} e^{-x/\beta}$$

for $x > 0$. If $\lambda = f/2$ and $\beta = 2$ the corresponding Γ-distribution is known as a χ^2-distribution with f degrees of freedom. In R, `dchisq` and `pchisq` are the density and distribution function, respectively, for this distribution.

Exercise 1.2 If $P([0, x])$ denotes the probability for the exponential distribution with parameter $\lambda > 0$ of getting an observation smaller than x find the solution to the equation

$$P([0, x]) = 0.5. \tag{1.1}$$

The solution is called the median. Explain what the R command `qexp(0.5, 10)` computes. Is it possible to solve the equation (1.1) if you replace the exponential distribution with the Γ-distribution? How can you solve the equation numerically using `qgamma`? Plot the median of the Γ-distribution against λ for $\lambda = 1, 2, \ldots, 10$.

Chapter 2

Second Week

2.1 Continuous distributions

Keywords: densities, distribution functions, empirical quantiles, generalized inverse, Gumbel distribution, local alignment, theoretical quantiles.

ISwR: 55-65, 145-153.

2.1.1 Local alignment statistics

Local alignment - a computer science problem?

What does the two words `ABBA` and `BACHELOR` have in common?

What about `BACHELOR` and `COUNCILLOR`, or `COUNCILLOR` and `COUNSELOR`?

Can we find the "optimal" way to match the words?

And does this optimal alignment make any sense – besides being a random matching of letters?

Local alignments

Assume that $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ are in total $n + m$ random letters from the 20 letter amino acid alphabet

$$E = \{\mathtt{A}, \mathtt{R}, \mathtt{N}, \mathtt{D}, \mathtt{C}, \mathtt{E}, \mathtt{Q}, \mathtt{G}, \mathtt{H}, \mathtt{I}, \mathtt{L}, \mathtt{K}, \mathtt{M}, \mathtt{F}, \mathtt{P}, \mathtt{S}, \mathtt{T}, \mathtt{W}, \mathtt{Y}, \mathtt{V}\}.$$

We want to find *optimal local alignments* and in particular we are interested in the *score* for optimal local alignments. This is a function

$$h : E^{n+m} \to \mathbb{R}.$$

Denote by

$$s_{n,m} = h(x_1, \ldots, x_n, y_1, \ldots, y_m)$$

the real valued score.

A local alignment of an x- and a y-subsequence is a letter by letter match of the subsequences, some letters may have no match, and matched letters are given a score, positive or negative, while unmatched letters in either subsequence are given a penalty. The optimal local alignment can be efficiently computed using a dynamic programming algorithm.

There is an implementation in the `Biostrings` package for R (`Biostrings` is a Bioconductor package). The function `pairwiseAlignment` implements, among other things, local alignment.

Local alignment scores

A large value of $s_{n,m}$ for two sequences means that the sequences have subsequences that are *similar* in the letter composition.

Similarity in letter composition is taken as evidence of functional similarity and/or homology in the sense of molecular evolution.

How large is large?

This is a question of central importance. The quantification of what large means. Functionally unrelated proteins will have an optimal local alignment score, and if we sample functionally and evolutionary unrelated amino acid segments at random from the pool of proteins, their optimal local alignments will have a distribution.

Local alignment score distributions

What is the distribution of $s_{n,m}$?

We can in principle compute its discrete distribution from the distribution of the x- and y-variables – futile and not possible in practice.

It is possible to rely on simulations, but it may be quite time-consuming and not a practical solution for usage with current database sizes.

Develop a good theoretical approximation.

Normal approximation

The most widely used continuous distribution is the *normal distribution* with density

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.1}$$

where $\mu \in \mathbb{R}$ and $\sigma > 0$ are the *mean* and *standard deviation*, respectively.

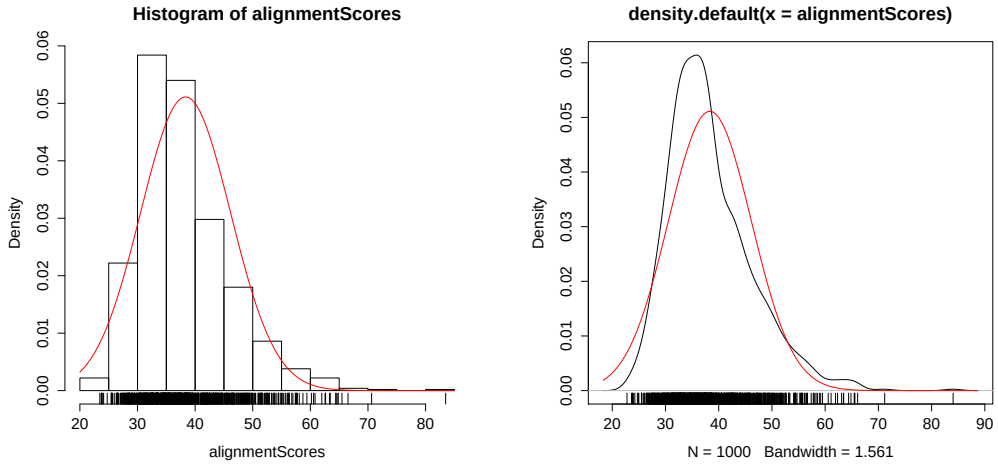


Figure 2.1: The histogram (left) and kernel density estimate (right) for 1000 simulated local alignment scores of two random length 100 amino acid sequences. The local alignment used the BLOSSUM50 score matrix, gap open penalty -12 and gap extension penalty -2. The density for the fitted normal approximation is superimposed (red). The actual local alignment scores are integer valued and have been "jittered" in the rug plot for visual reasons.

The standard estimators of μ and σ are the empirical versions of the mean and standard deviation

$$\begin{aligned} \hat{\mu} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \hat{\sigma} &= \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})^2}. \end{aligned}$$

We will later verify that with f as in (2.1) the theoretical mean is

$$\mu = \int_{-\infty}^{\infty} x f(x) \mathrm{d}x$$

and the theoretical variance is

$$\sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 f(x) \mathrm{d}x.$$

Thus the estimators above are simply empirical versions of these integrals based on the observed data rather than the theoretical density f. For the estimator of σ it is, arguably, not obvious why we divide by $n-1$ instead of n. There are reasons for this that have to do with the fact the μ is estimated, and in this case it is generally regarded as the appropriate way to estimate σ. For large n it does not matter, but for $n = 5$, say, it matters if we divide by $n = 5$ or $n-1 = 4$. Dividing by $n-1$ results in a larger estimate of σ, thus the uncertainty is estimated to be larger when dividing by $n-1$ as opposed to n.

Normal distribution function and quantiles

The distribution function for the normal distribution is

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} \mathrm{d}y$$

and the quantile function is the inverse, Φ^{-1}.

For the alignment score data the estimates of μ and σ are

$$\hat{\mu} = 38.3 \quad \text{and} \quad \hat{\sigma} = 7.80.$$

The 95% and 99% quantiles, say, for the normal distribution with mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ are

$$\Phi^{-1}(0.95) = 51.2 = \hat{\mu} + \hat{\sigma} \times 1.644.$$
$$\Phi^{-1}(0.99) = 56.5 = \hat{\mu} + \hat{\sigma} \times 2.33.$$

The empirical 95% and 99% quantiles are $x_{(950)} = 53$ and $x_{(990)} = 63$, which indicates that the normal distribution is not a good fit of the right tail of the local alignment scores distribution.

If the distribution function for the normal distribution with mean μ and standard deviation σ is denoted $\Phi_{\mu,\sigma}$ there are general relations

$$\Phi_{\mu,\sigma}(x) = \Phi_{0,1}\left(\frac{x-\mu}{\sigma}\right)$$

and

$$\Phi^{-1}_{\mu,\sigma}(q) = \mu + \sigma\Phi^{-1}_{0,1}(q).$$

This means that we only need to know how to compute the distribution function and the quantile function for the *standard normal distribution* with $\mu = 0$ and $\sigma = 1$. There is, however, no simpler closed form expression for $\Phi_{0,1}$ than the integral representation in terms of the density as above. Nor is there a simple closed form representation of $\Phi^{-1}_{0,1}$. This does not mean that we cannot compute very accurate numerical approximations such as those implemented in R as the `pnorm` and `qnorm` functions.

ISwR uses the notation N_q for the q-quantile in the standard normal distribution, thus $N_{0.95} = 1.644$, $N_{0.975} = 1.96$ and $N_{0.99} = 2.33$.

The quantiles are useful for setting a threshold to determine how large a score needs to be to be large. Using the normal approximation and choosing the threshold 51.2 there is only a 5% chance of getting a local alignment score for random amino acid sequences above this threshold value. Setting the threshold a little higher at 56.5 there is only a 1% chance of getting a score above this threshold.

One should appreciate the subtleties in the argument above. Having observed a score of 58, say, and this being larger than the 99% quantile (in the normal approximation), what is the conclusion? There are two possibilities, either the score is a relatively extreme observation *compared to the distribution of observations under the random model used* or the score reflects a deviation from the random model. We may argue that the latter is a more plausible explanation, and that the score thus represents evidence against the random model. This is,

however, not purely a statement based on the improbability of observing a score larger than 56.5. The interval $[38.241, 38.437]$ has probability 1% under the normal approximation to the score distribution, yet a score of 38.3 could never with any reason be regarded as evidence against the random model even though it falls nicely into the interval of probability 1%.

We should also note that the raw quantiles from the data are more pessimistic, and in particular the empirical 99% quantile is as high as 63. Thus we should definitely investigate if the normal approximation is an adequate approximation.

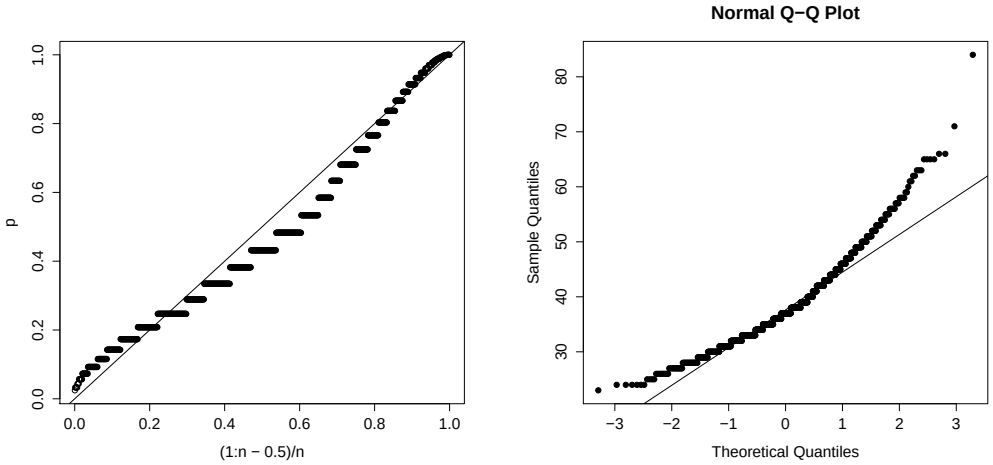


Figure 2.2: The PP-plot (left) and QQ-plot (right) of the data against the theoretical values. For the QQ-plot a line has been added to ease the interpretation; do the data points fall on a straight line?

The density estimates as well as the PP- and QQ-plots reveal that the normal distribution is not a perfect fit. The QQ-plot is particularly useful in revealing the main deviation – the sample quantiles are right-skewed. In other words, the tail probabilities of the normal distribution fall of too fast to the right and too slow to the left when compared to the data. The skewness is observable in the histogram and for the kernel density estimate as well.

Local alignment scores and the Gumbel distribution

Under certain conditions on the scoring mechanism and the letter distribution a valid approximation, for n and m large, of the probability that $s_{n,m} \leq x$ is

$$F(x) = \exp\left(-Knme^{-\lambda x}\right)$$

for parameters $\lambda, K > 0$.

With

$$\mu = \frac{\log(Knm)}{\lambda} \quad \text{and} \quad \sigma = \frac{1}{\lambda}$$

this distribution function can be written

$$F(x) = \exp\left(-e^{-\frac{x-\mu}{\sigma}}\right).$$

This is the distribution function for the *Gumbel distribution* with location parameter μ and scale parameter σ.

It is important to know that the location and scale parameters in the Gumbel distribution are *not* the theoretical mean and standard deviation for the Gumbel distribution. These are hard to determine, but they will be studied in an exercise.

Exercise: Maximum of random variables

Use

```
tmp <- replicate(100, max(rexp(10, 1)))
```

to generate 100 replications of the maximum of 10 independent exponential random variables.

Plot the distribution function for the Gumbel distribution with location parameter $\log(10)$ and compare it with the empirical distribution function for the simulated variables.

What if we take the max of 100 exponential random variables?

Quantiles for the Gumbel distribution

If $q \in (0, 1)$ we solve the equation

$$F(x) = \exp\left(-e^{-\frac{x-\mu}{\sigma}}\right) = q.$$

The solution is

$$x_q = \mu - \sigma \log(-\log(q)).$$

For the purpose of QQ-plots we can disregard the scale-location parameters μ and σ, that is, take $\mu = 0$ and $\sigma = 1$, and the QQ-plot will show points on a straight line if the Gumbel distribution fits the data.

As above, we generally stay away from the two extreme quantiles corresponding to $q = 1$ and $q = 0$. Though they could be taken as $\pm\infty$ in the Gumbel case, we prefer to avoid these infinite quantities.

2.1.2 R interlude: Local alignment statistics

We rely on the `Biostrings` package from Bioconductor, which has an implementation of the local alignment algorithm needed. The amino acid alphabet is also available, and we can use the `sample` function to generate random amino acid sequences.

```
require(Biostrings)
sample(AA_ALPHABET[1:20], 10, replace = TRUE)
```

```
##  [1] "D" "A" "T" "T" "W" "Y" "C" "V" "V" "A"
```

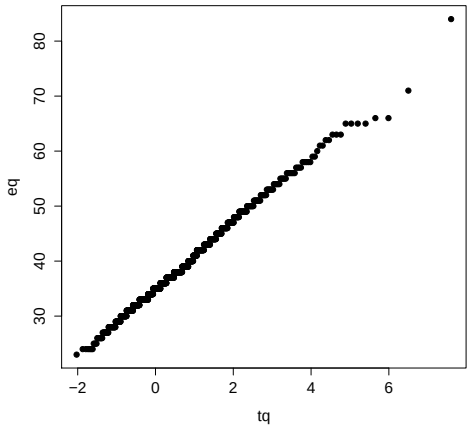


Figure 2.3: The QQ-plot of the local alignment scores against the Gumbel distribution. This is a very nice QQ-plot showing a good fit of the Gumbel distribution to the local alignment scores.

We use only the 20 amino acids (not the last three special letters in the `AA_ALPHABET` vector). We generate 10 random letters and the last argument to `sample` makes sure that the random sampling is done with replacement. To get a string, we need the `paste` function, which is used in the function `las` defined below.

For the local alignment we also need a matrix of scores. BLOSUM50 is one of the standards.

```
file <- "ftp://ftp.ncbi.nih.gov/blast/matrices/BLOSUM50"
BLOSUM50 <- as.matrix(read.table(file, check.names = FALSE))
```

Function for simulation of amino acid sequences and computation of local alignment score.

```
las <- function(n, m) {
  ## Simulating two random amino acid sequences
  x <- paste(sample(AA_ALPHABET[1:20], n, replace = TRUE), collapse = "")
  y <- paste(sample(AA_ALPHABET[1:20], m, replace = TRUE), collapse = "")
  ## Computing the optimal local alignment score of the two sequences
  ## and returning only the optimal score
  s <- pairwiseAlignment(pattern = x, subject = y, type = "local",
                         substitutionMatrix = BLOSUM50,
                         gapOpen = -12, gapExtension = -2,
                         scoreOnly = TRUE)
  return(s)
}
```

Simulation of 1000 random local alignment scores from two sequences of length 100 and computations of empirical mean and standard deviation.

```
alignmentScores <- replicate(1000, las(100, 100))
muHat <- mean(alignmentScores)
sigmaHat <- sd(alignmentScores)
```

Histograms and kernel density estimates compared to the estimated normal distribution.

```
hist(alignmentScores, prob = TRUE)
rug(jitter(alignmentScores, 3))
curve(dnorm(x, muHat, sigmaHat), add = TRUE, col = "red")

plot(density(alignmentScores))
rug(jitter(alignmentScores, 3))
curve(dnorm(x, muHat, sigmaHat), add = TRUE, col = "red")
```

The scores are integer scores. When plotting, the `jitter` function is useful in the rug plot for visual distinction of the observations.

```
qqnorm(alignmentScores, pch = 19)
qqline(alignmentScores)

n <- length(alignmentScores)
eq <- sort(alignmentScores)
p <- pnorm(eq, muHat, sigmaHat)
plot((1:n - 0.5)/n, p)
abline(0, 1)
```

Using the Gumbel distribution. Please note that the parameters estimated above as `muHat` and `sigmaHat` are *not* valid estimates of the corresponding location and scale parameters in the Gumbel distribution. We will return to this in an exercise next week, but for now we have not introduced any method for estimating these parameters. We can, however, make a QQ-plot without estimation of these parameters.

```
tq <- -log(-log((1:n - 0.5)/n))
plot(tq, eq, pch = 19)
```

2.1.3 Continuous distributions: A summary of theory

Probability distributions, also known as probability measures, are assignments of probabilities to events. If we think of the example with the neuron interspike data we could be interested in the probability of observing an interspike time smaller than x for a given x. We called this probability the distribution function as a function of x. The *event* is that the interspike time falls in the interval $[0, x]$. Since the probability of observing a neuron interspike time that is negative is 0 (it is impossible), this is technically also the probability of the event $(-\infty, x]$.

An event is for real valued observations a subset $A \subseteq \mathbb{R}$. We use the notation $P(A)$ to denote the probability of the event A for a given probability distribution P.

Probability distributions for a continuous observation, like the interspike time, are characterized by the distribution function. Knowing the distribution function, that is, the probabilities

of the special events $(-\infty, x]$, is enough to know the distribution. We will also consider densities below as a different way to specify a continuous probability distribution. There is a close relation between densities and distribution function. Actual computations of probabilities typical involve the distribution function directly and only the density indirectly, whereas computations with the density play an absolutely fundamental role in statistics through what is known as the likelihood function. This will be elaborated on later in the course.

Distribution functions

If P is a probability distribution on $\mathbb{R}$ the *distribution function* is defined as

$$F(x) = P((-\infty, x])$$

for $x \in \mathbb{R}$.

How does such a function look? What are the general characteristics of a distribution function?

Characterization

A distribution function $F : \mathbb{R} \to [0, 1]$ satisfies the following properties

(i) F is increasing.

(ii) $F(x) \to 0$ for $x \to -\infty$, $F(x) \to 1$ for $x \to \infty$.

(iii) F is right continuous.

Important characterization: Any function $F : \mathbb{R} \to [0, 1]$ satisfying the properties (i)-(iii) above is the distribution function for a unique probability distribution.

Examples

The *Gumbel distribution* has distribution function defined by

$$F(x) = \exp(-e^{-x})$$

The *exponential distribution* has distribution function

$$F(x) = 1 - e^{-\lambda x}.$$

Densities

If $f : \mathbb{R} \to [0, \infty)$ satisfies that

$$\int_{-\infty}^{\infty} f(x)\mathrm{d}x = 1$$

we call f a (probability) density function. The corresponding distribution function is

$$F(x) = \int_{-\infty}^{x} f(x)\mathrm{d}x.$$

The corresponding probability distribution can be expressed as

$$P(A) = \int_{A} f(x)\mathrm{d}x.$$

If a distribution function F is differentiable then there is a density;

$$f(x) = F'(x).$$

Example: The Gumbel distribution has distribution function $F(x) = \exp(-e^{-x})$. The derivative is

$$f(x) = \exp(-x)\exp(-e^{-x}) = \exp(-x - e^{-x}),$$

which is thus the density for the Gumbel distribution.

Exercise: Distribution functions

Plot the graph (use `plot` or `curve`) for the function

```
F <- function(x) 1 - x^(- 0.3) * exp(- 0.4 * (x - 1))
```

for $x \in [1, \infty)$. Argue that it is a distribution function.

Define

```
f <- function(x) x^3 * exp(- x) / 6
```

for $x \in [0, \infty)$ and use `integrate(f, 0, Inf)` to verify that f is a density. How can you use `integrate` to create the corresponding distribution function?

The normal distribution

It holds that

$$\int_{-\infty}^{\infty} e^{-x^2/2}\mathrm{d}x = \sqrt{2\pi}$$

Hence $f(x) = \frac{1}{\sqrt{2\pi}}\exp(-x^2/2)$ is a density – the probability distribution is the *standard normal distribution*.

The distribution function is

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-y^2/2}\mathrm{d}y.$$

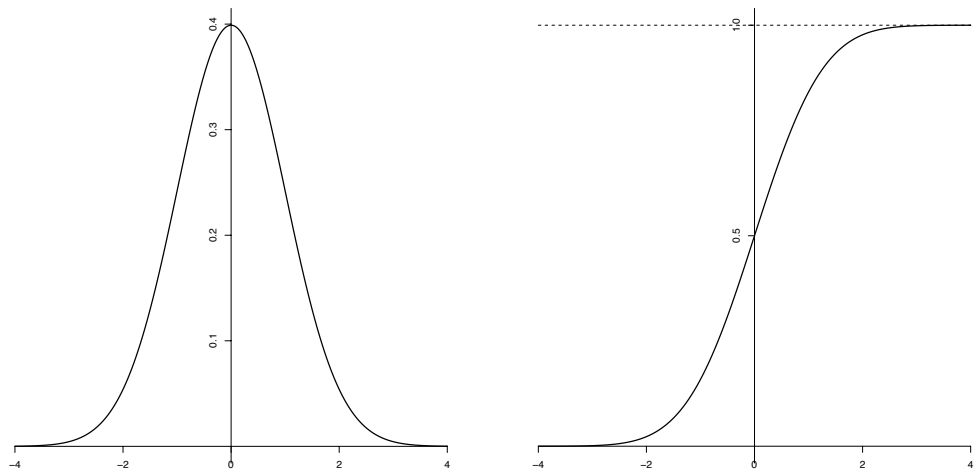


Figure 2.4: The density (left) and the distribution function (right) for the normal distribution.

Density interpretation

A density f has the interpretation that for small $h > 0$

$$f(x) \simeq \frac{1}{2h} P([x-h, x+h]).$$

The *frequency interpretation* says that $P([x-h, x+h])$ is approximately the relative frequency of observations in $[x-h, x+h]$, hence

$$f(x) \simeq \frac{1}{2h} \frac{1}{n} \sum_{i=1}^{n} 1(|x - x_i| \leq h).$$

The function

$$\hat{f}(x) = \frac{1}{2h} \frac{1}{n} \sum_{i=1}^{n} 1(|x - x_i| \leq h)$$

is one example of a *kernel density estimator* using the *rectangular kernel* – an alternative to the histogram.

Mean and Variance

The mean and variance for a probability distribution on $\mathbb{R}$ with density f are defined as

$$\mu = \int_{-\infty}^{\infty} x f(x) \mathrm{d}x \qquad \sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 f(x) \mathrm{d}x$$

provided the integrals are meaningful.

- The exponential distribution, $f(x) = \lambda \exp(-\lambda x)$ for $x \geq 0$, has

$$\mu = \frac{1}{\lambda} \qquad \sigma^2 = \frac{1}{\lambda^2}$$

- The normal distribution, $f(x) = (2\pi)^{-1/2} \exp(-x^2/2)$, has

$$\mu = 0 \qquad \sigma^2 = 1.$$

Scale-location parameters

If F is a distribution function for a probability distribution then

$$G(x) = F\left(\frac{x-\mu}{\sigma}\right)$$

for $\mu \in \mathbb{R}$ and $\sigma > 0$ is a distribution function.

G is called a *scale-location* transformation of F with scale parameter $\sigma > 0$ and location parameter μ.

If F has density f, the density for G is $g(x) = G'(x) = \frac{1}{\sigma} f\left(\frac{x-\mu}{\sigma}\right)$.

If F has mean μ_0 and variance σ_0^2 then G has mean

$$\mu + \sigma\mu_0$$

and variance

$$\sigma_0^2\sigma^2.$$

Empirical quantiles

Let $x_1, \ldots, x_n \in \mathbb{R}$ be n real observations from an experiment. We order the observations

$$x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(n)}.$$

If $q = i/n$ for $i = 1, \ldots, n-1$, then $x \in \mathbb{R}$ is called a *q-quantile* if $x_{(i)} \leq x \leq x_{(i+1)}$.

If $(i-1)/n < q < i/n$ for $i = 1, \ldots, n$ the q-quantile is $x_{(i)}$.

Quantiles are monotone in the sense that if x is a q-quantile and y is a p-quantile with $q < p$ then $x < y$.

The definition above is the usual definition of what empirical quantiles are. They are generally uniquely defined unless $q = i/n$ in which case anything in the interval $[x_{(i)}, x_{(i+1)}]$ is a valid q-quantile. When it comes to actually computing quantile estimates, or sample quantiles, a somewhat more liberal approach is often taken.

The `quantile` function in R computes a selection of sample quantiles, and by default the minimum $x_{(1)}$, the maximum $x_{(n)}$ and the *quartiles*, which are the 0.25-, 0.50- and 0.75-quantiles. However, there are 9 (nine) different types of sample quantiles that can be computed as specified by the `type` argument, and only some are actually empirical quantiles according to the definition above.

Theoretical quantiles

Definition 1. If $F : \mathbb{R} \to [0, 1]$ is a distribution function then $Q : (0, 1) \to \mathbb{R}$ is a *quantile function* for the distribution if

$$F(Q(y) - \epsilon) \leq y \leq F(Q(y)) \tag{2.2}$$

for all $y \in (0, 1)$ and all $\epsilon > 0$.

The general definition of a theoretical quantile function captures the idea that that Q should be a kind of inverse to the distribution function. If F has an inverse then this inverse is a quantile function, and it is the only quantile function there is.

It can also be shown that if Q is a quantile function for F then

$$\mu + \sigma Q$$

is a quantile function for G with G the scale-location transformation of F. The implication for QQ-plots is that as long at the distribution is correct up to a scale-location transformation then the QQ-plot will show points on approximately a straight line. A scale-location transformation will just change the slope and the intercept of the line.

Generalized inverse

Definition 2. Let $F : \mathbb{R} \to [0, 1]$ be a distribution function. A function

$$F^{\leftarrow} : (0, 1) \to \mathbb{R}$$

that satisfies

$$y \leq F(x) \;\Leftrightarrow\; F^{\leftarrow}(y) \leq x \tag{2.3}$$

for all $x \in \mathbb{R}$ and $y \in (0, 1)$ is called a generalized inverse of F.

General simulation

We will simulate from a distribution on $\mathbb{R}$ having distribution function F. First find the generalized inverse, $F^{\leftarrow} : (0, 1) \to \mathbb{R}$, of F.

Then we let $u_1, \ldots, u_n$ be generated by a pseudo random number generator that generates uniformly distributed variables and we define

$$x_i = F^{\leftarrow}(u_i)$$

for $i = 1, \ldots, n$. Then $x_1, \ldots, x_n$ are variables following the distribution with distribution function F.

To see that the statement is correct, we essentially use the same argument as previously used for the specific example with the exponential distribution. By the definition of the generalized inverse we have that $x_i = F^{\leftarrow}(u_i) \leq x$ if and only if $u_i \leq F(x)$, thus the probability that $x_i \leq x$ equals the probability that $u_i \leq F(x)$ and since the u_i is uniformly distributed the probability of this event is $F(x)$. This shows that the distribution of the variables x_i has distribution function F.

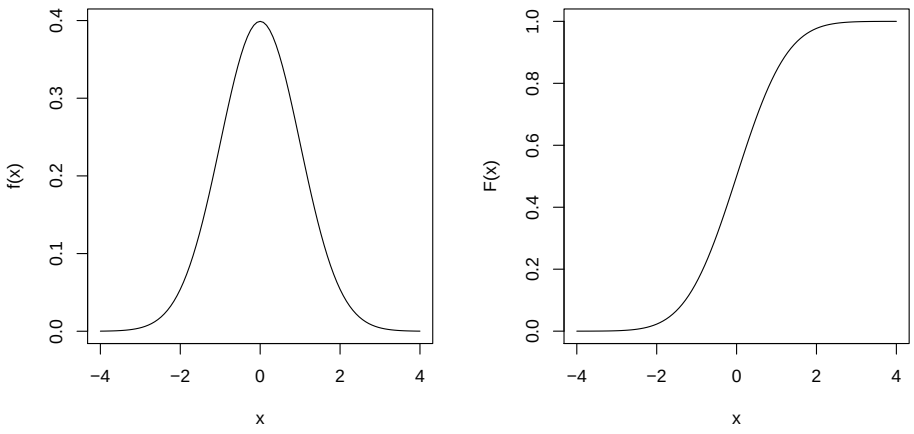
Generalized inverse

- If F has a true inverse (F is strictly increasing and continuous) then $F^{\leftarrow}$ equals the inverse, F^{-1}, of F.
- All distribution functions have a generalized inverse – we find it by "solving" the inequality $F(x) \geq y$. It is unique.
- $F^{\leftarrow}$ is a quantile function. It is the quantile function for the empirical distribution function you get from the `quantile`-function in R with `type = 1`.

2.1.4 R interlude: The normal distribution

The density for the normal distribution.

```
f <- function(x) exp(-x^2/2)/sqrt(2*pi)
integrate(f, -Inf, Inf)


## 1 with absolute error < 9.4e-05


F <- function(x) integrate(f, -Inf, x)$value
F(0) ## OK, but 'F' is not vectorized, does e.g. not work with 'curve'


## [1] 0.5


F <- Vectorize(F)
par(mfcol = c(1, 2))
curve(f, -4, 4)
curve(F, -4, 4)
```



Of course, all these computations are builtin for the normal distribution via the functions `dnorm` and `pnorm`.

Computation of mean and variance using the `integrate` function.

```
mu <- integrate(function(x) x*f(x), -Inf, Inf)$value
sigmasq <- integrate(function(x) (x-mu)^2*f(x), -Inf, Inf)$value
mu

## [1] 0

sigmasq

## [1] 1
```

2.2 Tables, tests and discrete distributions

Keywords: χ^2-test statistic, binomial distribution, dice games, discrete probability distributions, geometric distribution, patterns, tables, tandem repeats, random variables.

ISwR: Pages 145-153.

2.2.1 Tabular data, hypotheses and the χ^2-test

Genetic fingerprints

Short tandem repeats (STR) are used as *genetic fingerprints*.

TPOX has repeat pattern `AATG` and is located in intron 10 of the human thyroid peroxidase gene.

Humans are diploid organisms, data on STR repeats consist of pairs of repeat counts.

We are interested in how frequent a given repeat count occurs in a population, and we wish to investigate if this depends on the population.

In forensic science one of the interesting problems from the point of view of molecular biology and statistics is the ability to identify the person who committed a crime based on DNA-samples found at the crime scene. One approach known as the short tandem repeat (STR) analysis is to consider certain specific tandem repeat positions in the genome and count how many times the pattern has been repeated. The technique is based on tandem repeats with non-varying flanking regions to identify the repeat but with the number of pattern repeats varying from person to person. These counts of repeat repetitions are useful as "genetic fingerprints" because the repeats are not expected to have any function and the mutation of repeat counts is therefore neutral and not under selective pressure. Moreover, the mutations occur (and have occurred) frequently enough so that there is a sufficient variation in a population for discriminative purposes. It would be of limited use if half the population, say, have a repeat count of 10 with the other half having a repeat count of 11.

Without going into too many technical details, the procedure for a DNA-sample from a crime scene is quite simple. First one amplifies the tandem repeat(s) using PCR with primers that match the flanking regions, and second, one extracts the sequence for each of the repeats of interest and simply count the number of repeats. Examples of STRs used include TH01, which has the pattern `AATG` and occurs in intron 1 of the human tyrosine hydroxylase gene, and TPOX, which has the same repeat pattern but is located in intron 10 of the human thyroid peroxidase gene.

One tandem repeat is not enough to uniquely characterize an individual, so several tandem repeats are used. A major question remains. Once we have counted the number of repeats for k, say, different repeat patterns we have a vector $(n_1, \ldots, n_k)$ of repeat counts. The Federal Bureau of Investigation (FBI) uses for instance a standard set of 13 specific STR regions. If a suspect happens to have an identical vector of repeat counts – the same genetic fingerprint – we need to ask ourselves what the chance is that a "random" individual from the population has precisely this genetic fingerprint. This raises a number of statistical questions. First, what kind of random procedure is the most relevant – a suspect is hardly selected completely at random – and second, what population is going to be the reference population? And even if we can come up with a bulletproof solution to these questions, it is a huge task and certainly not a practical solution to go out and count the occurrences of the fingerprint $(n_1, \ldots, n_k)$ in the entire population. So we have to rely on smaller samples from the population to *estimate* the probability. This will necessarily involve model assumptions – assumptions on the probabilistic models that we will use.

One of the fundamental model assumptions is the classical *Hardy-Weinberg equilibrium* assumption, which is an assumption about *independence* of the repeat counts at the two different chromosomes in an individual.

TPOX repeat count

TPOX	African American	Caucasian	Hispanic	Total
8	192 (0.42)	323 (0.54)	132 (0.48)	647 (0.49)
9	92 (0.20)	72 (0.12)	29 (0.10)	193 (0.14)
10	46 (0.10)	34 (0.06)	9 (0.03)	89 (0.07)
11	113 (0.25)	147 (0.24)	77 (0.28)	337 (0.25)
12	11 (0.03)	25 (0.04)	30 (0.11)	66 (0.05)
Total	454 (1.00)	601 (1.00)	277 (1.00)	1332 (1.00)

The table shows the counts for the TPOX repeats in samples from three different populations. We have disregarded counts of 5, 6, 7 and 13, though they occur in the original data set. The primary reason is that the occurrence of these extreme repeat counts is rare – except 6 and to some extend 7 in the African American population. We want to focus the analysis on potential differences between the populations in the frequently occurring repeat counts.

Some questions

- A forensic question: Do we need different reference distributions to compute the probability of a genetic fingerprint for different populations?

- A scientific question: Has the repeat counts drifted apart since the branching of the three populations?

In both cases we ask if there is a

detectable difference in the repeat count distributions between the populations?

The formal hypothesis

To investigate the question we formulate the hypothesis

H: The distribution of the repeat counts *does not* depend on the population.

Under H we can estimate the probabilities of the repeat counts as

$$\hat{p}_0 = \frac{1}{1332}(647, 193, 89, 337, 66) = (0.49, 0.14, 0.07, 0.25, 0.05)$$

The formal hypothesis can then be investigated by investigating if $\hat{p}_0$ adequately represents the distributions of observed counts for the three populations.

The meaning of the formal hypothesis is that, if it is true, then if we sample random individuals from either population the distribution of the repeat counts would be approximately the same disregarding which population we sample from. Thus, there is a common vector of point probabilities, p_0, which we estimate from the data set as above.

Compare the setup with that of local alignment. We want to quantify if the actually observed data contain evidence against the hypothesis H, just as we sought a quantification of the evidence in the optimal local alignment of two amino acid sequences of whether it was anything but a random match.

The test statistic

One way to test the hypothesis is to compare the observed counts in the table with the expected counts under the hypothesis H.

The Pearson χ^2-*test statistic* is

$$X = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}} = 63.9348.$$

Large values of X are critical in the sense of being stronger evidence against the hypothesis than smaller values. But is 63.9348 large?

The definition of the χ^2-test statistic above is vague and general. For an $r \times c$-table, as in our example, then with n_{ij} the counts in cell (i, j) of the table and

$$\begin{aligned} n_{i\cdot} &= \sum_j n_{ij} \\ n_{\cdot j} &= \sum_i n_{ij} \end{aligned}$$

the row and column sums (the marginals) of the table we see that the estimated probabilities under the hypothesis H are

$$\hat{p}_{0i} = \frac{n_{i\cdot}}{n_{\cdot\cdot}}$$

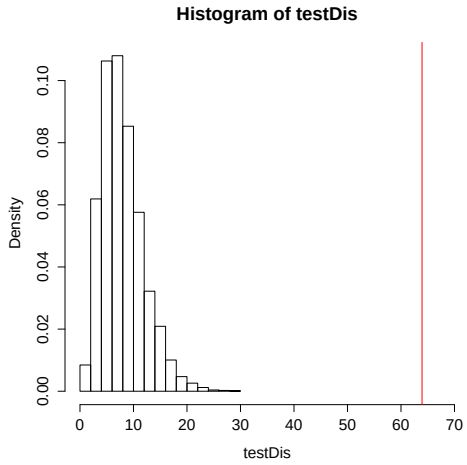
with $n_{\cdot\cdot} = \sum_i n_{i\cdot} = \sum_j n_{\cdot j} = 1332$ the total number of observations in the table.

The expected value for cell (i, j) in the cell is

$$\hat{e}_{ij} = \hat{p}_{0i} n_{\cdot j} = \frac{n_{i\cdot} n_{\cdot j}}{n_{\cdot\cdot}}.$$

How large is large

Imagine that H is true and that we generate a large number of tables. What is the probability of getting the table observed or a more extreme table?



There are two questions that need to be answered.

- What exactly do we mean by "generate"?
- What does "more extreme" mean?

To *generate* a table is usually interpreted as meaning to simulate a table using the same sampling scheme as used for the original table. Thus, as in our case we think of the total number of observations in each column as fixed, and we can simulate the distribution of counts according to the estimated probabilities $\hat{p}_0$ in the three columns. Another sampling scheme is to take only the total counts as fixed.

Alternatively, we could think of the total column counts as well as the total row counts (the marginals) as fixed and try to generate new tables (randomly) with these fixed marginals. It is an interesting fact that this distribution does not depend on any unknowns (no estimated quantities need to be plugged into the simulation as opposed to above), but it is also a fact that it is much more difficult to generate such new tables randomly. The Fisher's exact test as implemented in R as `fisher.test` is based on this conditional distribution.

There are also several options when it comes to expressing "extremeness". Given our introduction of the χ^2-statistic it is natural to define "more extreme" as meaning "with a larger value of X", and that is indeed a common choice. Note the relation to the local alignment problem. If we look as the simulated distribution of the χ^2-statistic, our observed value is quite extreme. In fact, it is so extreme that we have not seen a single simulated table in our simulation of 10.000 tables with that large a value of X. The probability of getting X as large as or larger than the observed X is thus very small, but, as with local alignment, this is not the only reason to conclude that the observed value of 63.93 of X is strong evidence against the hypothesis H. What we also need to have in mind is that there are alternatives to H, for instance, that each population has a distribution of repeat counts different from the other populations, which can explain the large value of X.

An alternative to extremeness based on the χ^2-statistic is to define all tables with a probability smaller than the observed table as more extreme and then compute the total probability of getting any of these tables. This is the approach taken in `fisher.test`.

The theoretical approximation

For a contingency table with r rows and c columns the χ^2-statistic follows, approximately, a χ^2-distribution with $(r-1) \times (c-1)$ degrees of freedom (df) under the hypothesis H.

In our example, $r = 5$ and $c = 3$ hence

$$\text{df} = 4 \times 2 = 8.$$

If F is the distribution function for the χ^2-distribution with 8 degrees of freedom the *p-value* is

$$1 - F(63.9348) = 7.835 \times 10^{-11}.$$

The p-value is defined as the probability of observing a table under the hypothesis with a test statistic as large as or larger than the actually observed test statistic. A small p-value is equivalent to a large value of the test statistic, and thus the smaller the p-value is, the stronger is the evidence against the hypothesis. The mixed blessing of p-values is that they give us an absolute scale for measuring deviations from the hypothesis, and two common (but arbitrary) *de facto* standard "thresholds" for p-values are 0.05 and 0.01. A p-value in the range from 0.01 to 0.05 is typically interpreted as moderate evidence against the hypothesis, and we often use the phrase that "the test is statistically significant at a 5% level". A p-value smaller than 0.01 is typically interpreted as strong evidence against the hypothesis. Even though this absolute scale is convenient for automatic procedures, it is a mixed blessing because p-values are so easily misunderstood, misused, and misinterpreted. Most importantly, a p-value **is not** a probability statement about the truth-value of the hypothesis. It is a probability statement about the data under the hypothesis.

The classical formal statistical tests, like Pearson's χ^2-test and the two-sample t-test that we will encounter later in the course, belong in the standard curriculum of statistics. We will, however, try to view statistics more from the model building perspective than as a theory for drawing rigid conclusions based on formal hypothesis tests. That is, we will attempt to

focus on the inductive process in science of going from data to model and how to deal with and report the uncertainties involved in this business.

We can, on the other hand, not simply dismiss hypothesis tests. If I report the table above on the repeat counts for this particular sample of individuals it is clear that there are differences between the populations. The table is, however, of no interest if it does not generalize to other samples and, indeed, to the entire populations. The sample is special and another sample will be different. The main question is whether the observed differences are predominantly due to differences at the population level or predominantly due to the randomness of the sample. If we could compute the table for the entire populations there would, undoubtedly, be differences between the populations, but this is of little interest and no practical value. What is interesting is whether we can detect and with reason estimate the differences at the population level from the sample. A formal hypothesis and corresponding test should be viewed as a surrogate for this more important question. If we cannot detect any differences above what is expected by randomness of the sample, the estimated differences are likely to represent very little but the randomness. Moreover, if the randomness dominates over the actual differences between the populations we are generally going to be better off by estimating the simpler model as represented by the hypothesis.

The p-value 7.835×10^{-11} is ridiculously small and tells us that the observed differences between the populations in the sample cannot be ascribed to the randomness of the sample.

To compute the p-value using the theoretical χ^2-distribution above we introduced the slightly mysterious quantity called the "degrees of freedom". At this point you are just given the formula above for the computation of the degrees of freedom for the table. What is worth noticing is that while the degrees of freedom depends on the size of the table, it is the only variable quantity that enters in the χ^2-distribution. In particular, the χ^2-distribution used to evaluate the extremeness of the χ^2-test statistic does not depend on the unknown probabilities under the hypothesis.

The χ^2-distribution is, however, an approximation, and it may not be accurate for tables with few counts in some cells.

Rules of thumb and alternatives

A common rule of thumb says that the χ^2-distribution is a good approximation if the expected cell count is ≥ 5 for all cells.

This is a conservative rule.

Alternatives include Fisher's exact test, and a number of corrections to improve on the approximation for small expected cell counts.

Though Fisher's test in a technical sense is exact and often recommended for tables with small cell counts, it has its own problems.

Don't get too excited about minor differences between different procedures and do use simulation experiments extensively whenever possible.

The discussion about the approximation of the distribution of the χ^2-test statistic is a delicate one. From a certain point of view it is important, and of interest to find alternatives or corrections that yield better approximations. On the other hand, though these improvements are, from a technical point of view, improvements, no serious conclusion about the subject matter should be based on a minor technical adjustment as long as the overall approach is sound.

A more specific question

We can be more specific and ask if the distribution of repeat count 8 is the same across the three populations.

H: The distribution of repeat counts 8 *does not* depend on the population.

This can be investigated just as previously by collapsing the other cells in the table to a single cell to obtain a 2×3 table. The degrees of freedom will be 2.

A different question

You might also know a vector of proportions for repeat count 8 in the three populations and want to compare this vector with the observed proportions.

H: The distribution of repeat counts 8 in the three populations is given by the probabilities $p = (0.41, 0.54, 0.48)$.

The χ^2-test statistic is defined as usual, but in this case the relevant degrees of freedom is 3 and not 2.

2.2.2 R interlude: NIST analysis

Loading the data set and forming a table.

```
download.file("http://www.math.ku.dk/~richard/courses/StatScience2011/NIST.RData",
              "NIST.RData")
load("NIST.RData")
TPOXall <- table(c(NIST[, "TPOX.1"], NIST[, "TPOX.2"]),
                 rep(NIST[, "population"], 2))
```

Subsetting the table, computation of margins and estimation of probabilities (computation of relative frequencies).

```
TPOX <- TPOXall[4:8, ]
addmargins(TPOX)


##
```

```
##       African American Caucasian Hispanic  Sum
##   8               192       323      132  647
##   9                92        72       29  193
##   10               46        34        9   89
##   11              113       147       77  337
##   12               11        25       30   66
##   Sum             454       601      277 1332

pHat <- prop.table(TPOX, margin = 2)
```

Estimation of marginal probabilities (relative frequencies of the row marginal here). Computation of the expected values and the χ^2 statistic.

```
pOHat <- prop.table(margin.table(TPOX, 1))
expect <- outer(pOHat, margin.table(TPOX, 2), "*")
Xsq <- sum((TPOX - expect)^2/expect)
```

The χ^2 statistic can also be computed using the `chisq.test` function. We return below to the p-value and degrees of freedom (df) that the function returns.

```
chisq.test(TPOX)


##
##  Pearson's Chi-squared test
##
## data:  TPOX
## X-squared = 63.93, df = 8, p-value = 7.835e-11
```

We can also compare the populations in a pairwise manner.

```
chisq.test(TPOX[, -3])   ## Comparing African Americans and Caucasians


##
##  Pearson's Chi-squared test
##
## data:  TPOX[, -3]
## X-squared = 27.5, df = 4, p-value = 1.572e-05

chisq.test(TPOX[, -2])   ## Comparing African Americans and Hispanics


##
##  Pearson's Chi-squared test
##
## data:  TPOX[, -2]
## X-squared = 44.16, df = 4, p-value = 5.94e-09

chisq.test(TPOX[, -1])   ## Comparing Caucasians and Hispanics
```

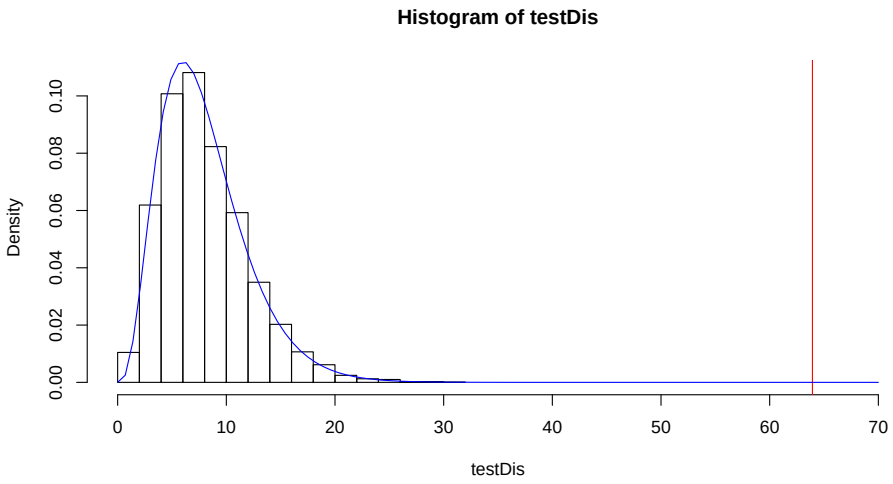
```
##
##  Pearson's Chi-squared test
##
## data:  TPOX[, -1]
## X-squared = 18.28, df = 4, p-value = 0.00109
```

To investigate if the computed χ^2 statistic is large, we implement the generation of a new table using the `sample` function. The generation is a simulation under the hypothesis that the distribution of the repeat counts is the same for all three populations.

```
simTab <- function(p0, n) {
  simData <- sample(as.integer(names(p0)), sum(n),
                    replace = TRUE, prob = p0)
  simData <- data.frame(simData, rep(names(n), times = n))
  table(simData[, 1], simData[, 2])
}
```

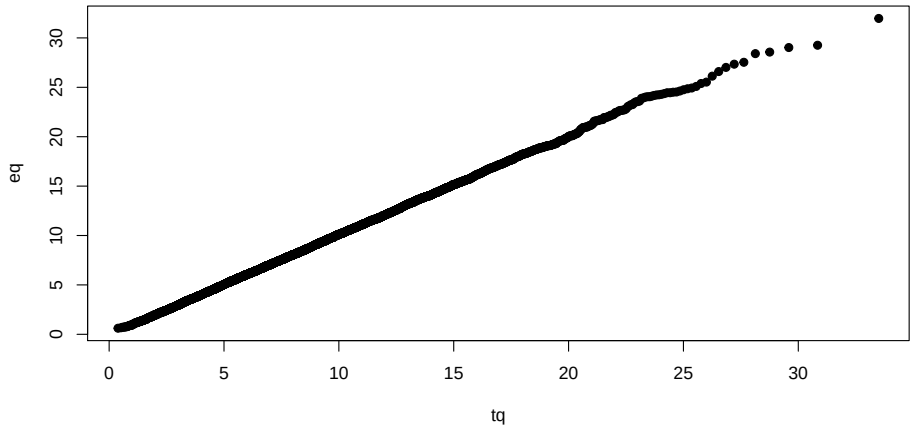
Then we replicate the generation of a table $B = 10,000$ times, compute the χ^2 test statistic, make a histogram and compare the observed χ^2 statistic for the original table with the distribution of χ^2 statistics for the generated tables.

```
colMarg <- colSums(TPOX)
B <- 10000
testDis <- replicate(B, chisq.test(simTab(p0Hat, colMarg))$statistic)
hist(testDis, prob = TRUE, xlim = c(0,70))
abline(v = chisq.test(TPOX)$statistic, col = "red")
curve(dchisq(x, df = 8), add = TRUE, col = "blue")
```



The observed value is very extreme in the distribution. Theory tells that the distribution can be approximated by a χ^2 distribution with 8 degrees of freedom. The density/histogram comparison is pretty good. Lets also take a look at the QQ-plot for a different comparison of the distributions.

```
eq <- sort(testDis)
tq <- qchisq((1:B - 0.5)/B, df = 8)
plot(tq, eq, pch = 19)
```



Indeed, the suggested χ^2 distribution is a very good fit to the distribution of the simulated test statistics. We use the theoretical approximation to compute the p-value as the probability of observing a table with a χ^2 test statistic larger than 63.9348.

```
pchisq(63.9348, df = 8, lower.tail = FALSE)

## [1] 7.835e-11
```

The use of `lower.tail = FALSE` means that we compute the probability of an observation greater than 63.9348. This is formally equivalent to

```
1 - pchisq(63.9348, df = 8)

## [1] 7.835e-11
```

but the latter may turn out numerically different for small p-values. In particular, the latter may be numerically 0 while the former is a very small (and more accurate) non-zero number.

A more specific question. Does the distribution of repeat count 8 depend on the population? The computation of the χ^2 test statistic and corresponding p-value can be achieved using the `prop.test` function, but it can also be achieved by `chisq.test` by collapsing rows in the original table.

```
prop.test(TPOX[1, ], colSums(TPOX))

##
##  3-sample test for equality of proportions without continuity
##  correction
```

```
##
## data:  TPOX[1, ] out of colSums(TPOX)
## X-squared = 13.7, df = 2, p-value = 0.00106
## alternative hypothesis: two.sided
## sample estimates:
## prop 1 prop 2 prop 3
## 0.4229 0.5374 0.4765
```

```
chisq.test(rbind(TPOX[1, ], colSums(TPOX[-1, ])))
```

```
##
##  Pearson's Chi-squared test
##
## data:  rbind(TPOX[1, ], colSums(TPOX[-1, ]))
## X-squared = 13.7, df = 2, p-value = 0.00106
```

A slightly different question is whether the observed proportions in the three populations of repeat count 8 equals a known vector of proportions? The relevant function to use is then `prop.test`.

```
prop.test(TPOX[1,], colSums(TPOX), p = c(0.41, 0.54, 0.48))
```

```
##
##  3-sample test for given proportions without continuity correction
##
## data:  TPOX[1, ] out of colSums(TPOX), null probabilities c(0.41, 0.54, 0.48)
## X-squared = 0.3419, df = 3, p-value = 0.952
## alternative hypothesis: two.sided
## null values:
## prop 1 prop 2 prop 3
##   0.41   0.54   0.48
## sample estimates:
## prop 1 prop 2 prop 3
## 0.4229 0.5374 0.4765
```

2.2.3 Sequence Patterns

Biological sequence patterns

The TPOX short tandem repeat has repeat pattern `AATG`.

The *start codon* for protein coding genes is `ATG`.

The genome encodes biology as *patterns* or *motifs*. We search the genome for biologically important patterns.

This is the *text mining* part of bioinformatics.

Text mining

How often and where does a *pattern* or *motif* occur in a text? By complete chance? Due to a "rule" that we want to understand and/or model??

THis is a core problem in biological sequence analysis.

Of broader relevance: Email spam detection, data mining of web pages or scientific papers.

A dice game

Throw the die until one of the patterns 1 4 3 or 2 1 4 occurs.

I win if 1 4 3 occurs. This is a winner:

$$4\ 6\ 2\ 3\ 5\ 1\ 3\ 2\ 4\ 5\ 1\ 4\ 3$$

Is this a fair game? How much should I be willing to bet if you bet 1 kroner on your pattern to be the winning pattern?

Fairness and odds

Lets play the game n times, let p denote the probability that I win the game, and let ξ denote my bet.

With ϵ_n the *relative frequency* that I win, my *average* gain in the n games is

$$\epsilon_n - \xi(1 - \epsilon_n) \simeq p - \xi(1 - p),$$

the approximation following from the *frequency interpretation*.

The game is *fair* if the average gain is 0, that is, if

$$\xi = \frac{p}{1 - p}.$$

The quantity ξ is called *the odds* of the event that I win.

The average gain is the gain *per game*. The total gain in n fair games may (and will) deviate substantially from 0. This is perceived as "luck" or "being in luck".

The odds, ξ, of an event and the probability, p, of the same event are thus linked by the formula above. Specifying one is equivalent to specifying the other. The computation of p can be done by simulation, see Figure 2.5.

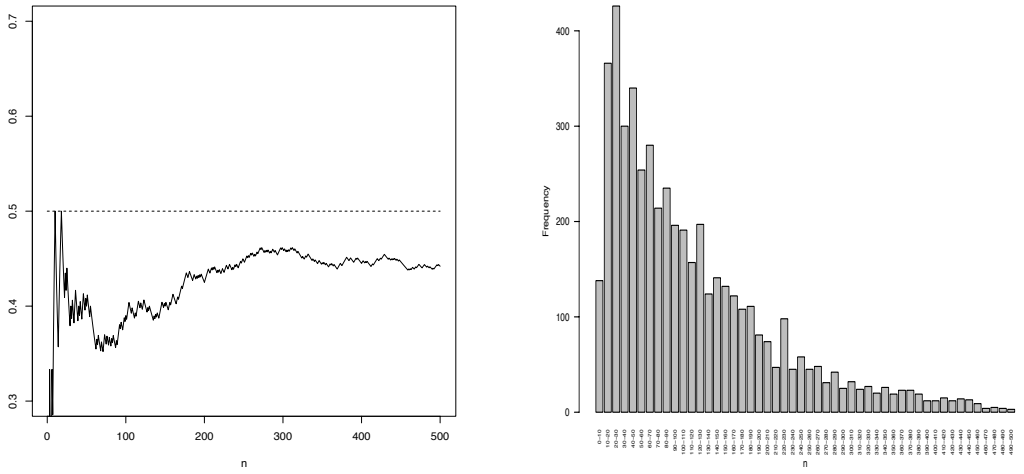


Figure 2.5: Relative frequency of occurrences of pattern 1 4 3 before 2 1 4 (left) and distribution of number of throws before first occurrence of one of the patterns (right).

The probability of the start codon

Lets try to compute the probability of the start codon `ATG`.

We need

- a probability model for the single letters,
- a model of how the letters are related,
- and some notation to support the computations.

Random variables

It is useful to introduce the concept of *random variables* as representations of unobserved variables.

Three unobserved DNA letters are denoted XYZ, and we want to compute

$$\mathbb{P}(XYZ = \texttt{ATG}) = \mathbb{P}(X = \texttt{A}, Y = \texttt{T}, Z = \texttt{G})$$

We assume that the random variables X, Y, and Z are *independent*, which means that

$$\mathbb{P}(X = \texttt{A}, Y = \texttt{T}, Z = \texttt{G}) = \mathbb{P}(X = \texttt{A})\mathbb{P}(Y = \texttt{T})\mathbb{P}(Z = \texttt{G}).$$

We assume that X, Y and Z *have the same distribution*, that is

$$\mathbb{P}(X = w) = \mathbb{P}(Y = w) = \mathbb{P}(Z = w)$$

for all letters w in the DNA alphabet.

The model presented above is the *loaded die* model. It is like generating DNA sequences by throwing a loaded die. It is most likely not a very accurate model, but it is a starting point and the point of departure for learning about more complicated models.

The fundamental random mechanism that works at the molecular level is random mutation. This is the driving dynamic force of evolution. Due to selection, biological sequences are certainly not just randomly generated sequences – some mutations are favored over others. Biological sequences are, on the other hand, not designed with a unique perfect fit in mind either. There is variation, and we need distributions to describe collections of sequences. We can attack the modeling of this variation at many levels. A birds eye perspective using simple models may provide some qualitative knowledge and superficial understanding of these distributions, while more detailed models may be used to more accurately deal with specific applications.

Amino acid distributions

Proteins are amino acid sequences and a simple model as a point of departure is again the loaded die model. The choice of point probabilities on the individual amino acids may depend on the reference population of proteins to be considered.

On the *sample space* of amino acids

$$\{\mathtt{A}, \mathtt{R}, \mathtt{N}, \mathtt{D}, \mathtt{C}, \mathtt{E}, \mathtt{Q}, \mathtt{G}, \mathtt{H}, \mathtt{I}, \mathtt{L}, \mathtt{K}, \mathtt{M}, \mathtt{F}, \mathtt{P}, \mathtt{S}, \mathtt{T}, \mathtt{W}, \mathtt{Y}, \mathtt{V}\}$$

we can take the *uniform distribution* (unloaded die) with probability 1/20 for all amino acids.

We may encounter the *Robinson-Robinson point probabilities* from the relative frequencies of the occurrences of amino acids in a selection of real proteins. They read

Amino acid	Probability	Amino acid	Probability	Amino acid	Probability
A	0.079	G	0.074	P	0.052
R	0.051	H	0.022	S	0.071
N	0.045	I	0.051	T	0.058
D	0.054	L	0.091	W	0.013
C	0.019	K	0.057	Y	0.032
E	0.063	M	0.022	V	0.064
Q	0.043	F	0.039		

The probability of the start codon

If the point probabilities for the DNA alphabet are

A	C	G	T
0.21	0.29	0.29	0.21

the probability of ATG under the loaded die model is

$$\mathbb{P}(XYZ = \mathtt{ATG}) = 0.21 \times 0.21 \times 0.29 = 0.013.$$

The probability of *not* observing a start codon is

$$\mathbb{P}(XYZ \neq \mathtt{ATG}) = 1 - \mathbb{P}(XYZ = \mathtt{ATG}) = 1 - 0.013 = 0.987.$$

Exercise: Simulate occurrences of start codons

Use the R function `sample` to generate random DNA sequences of length 99 with the point probabilities as given above.

Generate 10,000 sequences of length 99 and compute the relative frequency of sequences with

- a start codon at any position
- a start codon in the reading frame beginning with the first letter.

2.2.4 R interlude: A dice game

A dice game function that takes three arguments. The number of games to play (`N`) and the two patterns. Default patterns are the patterns from above. It returns an array containing two rows. The first row is an indicator (0-1 variable), which is 1 if the first pattern showed up first. The second row contains the number of dice throws before any of the patterns showed up.

```
diceGame <- function(N, pattern1 = c(1, 4, 3), pattern2 = c(2, 1, 4)) {
  len <- numeric(N)    ## A numeric vector of length N - holds the length of the game
  result <- numeric(N) ## A numeric vector of length N - holds the winner of the game
  for(n in 1:N) {
      test1 <- 1    ## Control variable, how many outcomes match pattern1
      test2 <- 1    ## Control variable, how many outcomes match pattern2
      l <- 1
      while((test1 < 4) && (test2 < 4)) {
          temp <- sample(1:6, 1)  ## Random, uniform sampling of a
                                  ## single number between 1 and 6
          if(temp == pattern1[test1]) test1 <- test1 + 1
          else test1 <- 1
          if(temp == pattern2[test2]) test2 <- test2 + 1
          else test2 <- 1
          l <- l + 1
        }
      len[n] <- l
      result[n] <- (test1 == 4);
    }
  return(cbind(result, len))
}
```

500 replications of the dice game with results placed in the variable `tmp`. The relative frequency (`relFreq`), the cumulative relative frequency (`cumRelFreq`) of the number of games won by the first pattern, and a nice plot, see Figure 2.5.

```
tmp <- diceGame(500)
relFreq <- sum(tmp[, 1]) / 500
print(relFreq)
```

```
## [1] 0.466


cumRelFreq <- cumsum(tmp[, 1]) / (1:500)
plot(cumRelFreq, type = "l", ylim = c(0.3, 0.7))
abline(0.5, 0, lty = 2) ## Adds horizontal line (line with slope 0 and intercept 0.5)
```

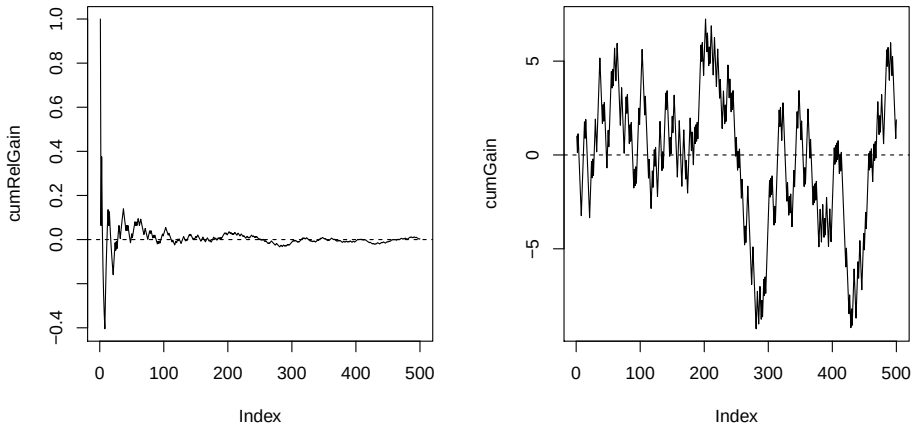
Odds computations

In a fair game the cumulative relative gain will stabilize around 0 but the cumulative absolute gain will fluctuate around zero with larger and larger fluctuations as N increases.

```
## The (estimated) odds of pattern1 being the winning pattern
xi <- relFreq/(1 - relFreq)
xi


## [1] 0.8727
```

New game: Bets are fair according to the estimated odds.

```
tmp <- diceGame(500)
cumRelFreq <- cumsum(tmp[, 1]) / (1:500)

cumRelGain <- (1 + xi) * cumRelFreq - xi         ## The cumulative, relative gain
cumGain <- seq(along = cumRelGain) * cumRelGain  ## The cumulative absolute gain

par(mfcol = c(1, 2))      ## Sets up a plot with multiple subplots with two plots
                          ## vertically and one horizontally.
plot(cumRelGain, type = "l")
abline(0, 0, lty = 2)     ## Adds horizontal line (line with slope and intercept 0)
plot(cumGain, type = "l")
abline(0, 0, lty = 2)
```



2.3 Exercises

Exercise 2.1 Find the quantile function for the standard Gumbel distribution with distribution function

$$F(x) = \exp(-e^{-x}).$$

Implement a function in R for simulation from the standard Gumbel distribution. Modify the implementation to simulate from the Gumbel distribution with location parameter μ and scale parameter $\sigma > 0$.

Exercise 2.2 Compute numerically the mean μ_0 and the variance σ_0^2 for the standard Gumbel distribution using the `integrate` function in R.

Exercise 2.3 Simulate 10,000 standard Gumbel distributed variables using Exercise 2.1 above. Compare the empirical mean and empirical variance for the data with the theoretical mean and theoretical variance computed above.

Exercise 2.4 If $x_1, \ldots, x_n$ is a data set with n observations use $\bar{x}$ to denote the average (empirical mean) and s^2 to denote the empirical variance (as computed by `var` in R). For a general scale-location transformation of a distribution F with mean μ_0 and variance σ_0^2 solve the equations

$$\mu + \sigma\mu_0 = \bar{x}$$
$$\sigma_0^2\sigma^2 = s^2$$

in terms of μ and σ^2. How can this be used to estimate the location and scale parameter?

Exercise 2.5 Use the `las` R function from Section 2.1.2 to generate 1000 local alignment scores of random amino acid sequences. Use Exercise 2.4 to estimate the scale-location parameters for a Gumbel distribution.

Exercise 2.6 Argue that the function

$$F(x) = 1 - \exp(-x^\beta), \quad x \geq 0$$

for $\beta > 0$ is a distribution function. It is called the *Weibull distribution* with parameter β. Find the density on the interval $[0, \infty)$ for the Weibull distribution.

Exercise 2.7 Download the `BUtraffic` data set using `read.table` from the url

`http://www.math.ku.dk/~richard/courses/StatScience2011/BUtraffic.txt`

The data set is a data frame with three columns, `time`, `duration` and `size`, which give the duration in seconds and size in bytes of internet downloads at given times at UC Berkeley. Compute the minimum, the maximum, and the quartiles for the duration and size of the downloads. Plot duration against size and log-duration against log-size.

Exercise 2.8 Investigate the distribution of the size of the downloads. Does a normal distribution fit? Does a normal distribution fit log size? Estimate the mean and variance for the model that fits best and compute the theoretical 95% and 99% quantile for the fitted distribution of download sizes. Compare with the empirical quantiles.

Exercise 2.9 Investigate the distribution of the duration of the downloads. Does a normal distribution fit? Does a normal distribution fit `log(duration)`?

Exercise 2.10 Recall the Weibull distribution with distribution function

$$F(x) = 1 - \exp(-x^\beta), \quad x \geq 0$$

for $\beta > 0$. Compute the quantile function.

Exercise 2.11 Restrict attention to the durations larger than one second. Use the QQ-plot to investigate if the Weibull distribution fits the distribution of log durations for the β parameter in the range from 1 to 2.

Exercise 2.12 How can you estimate the unknown β parameter for the Weibull distribution in the previous exercise? Try computing the theoretical mean in terms of β and equate it equal to the sample mean.

Exercise 2.13 Argue that the function

$$F(x) = 1 - x_0^{\beta} x^{-\beta}, \quad x \geq x_0 > 0$$

for $\beta > 0$ is a distribution function on $[x_0, \infty)$. It is called the *Pareto distribution* on the interval $[x_0, \infty)$. Compute the quantile function and the density for the Pareto distribution.

Chapter 3

Third Week

3.1 Discrete distributions

Hardy-Weinberg equilibrium

All diploid organisms like humans carry two copies of the chromosomes. For a gene occurring in two combinations as *allele* A or a there are three possible *genotypes*: AA, Aa, aa.

We sample a random individual from a population and observe Y – the genotype – taking values in the sample space $\{AA, Aa, aa\}$.

With X_f and X_m denoting unobservable father and mother alleles for the random individual the variable Y is a function of these. Under a random mating assumption – that is, *independence of* X_m *and* X_f – and an equal distribution assumption in the male and female populations:

$$\mathbb{P}(Y = AA) = p^2, \quad \mathbb{P}(Y = Aa) = 2p(1-p), \quad \mathbb{P}(Y = aa) = (1-p)^2$$

with $p = \mathbb{P}(X_m = A) = \mathbb{P}(X_f = A)$.

The probability of a least one start codon

We continue the exploration of independence using the die model of biological sequences but now for longer sequences of letters. Here of length $3n$ to give a total of n codons. That is, we allow only for a single reading frame, we count only the disjoint codons in this reading frame and not the overlapping codons corresponding to different reading frames.

If we have n codons ($3n$ DNA letters)

what is the probability of at least one start codon?

What is the probability of *not* observing a start codon?

The codons are *independent*

$$P(\text{no start codon}) = \mathbb{P}(XYZ \neq \texttt{ATG})^n = 0.987^n.$$

and

$$P(\text{at least one start codon}) = 1 - 0.987^n.$$

The general rule is that if A denotes an event with probability $P(A)$ then A^c denotes the complementary event and

$$P(A^c) = 1 - P(A).$$

If A_i denotes the event that codon i *is not* `ATG` then the intersection (joint occurrence) $A_1 \cap \ldots \cap A_n$ is the event that *no* codons equal `ATG`. It is the general definition that independence of events $A_1, \ldots, A_n$ means that

$$P(A_1, \ldots, A_n) = P(A_1) \times \ldots \times P(A_n),$$

that is, the probability of the joint occurrence of independent events is the product of their individual probabilities.

The number of codons before the first start codon

If L denotes the number of codons before the first start codon we have found that

$$\begin{aligned} \mathbb{P}(L = n) &= P(\text{no start codon in } n \text{ codons}, \text{start codon at codon } n+1) \\ &= 0.987^n \times 0.013. \end{aligned}$$

This is the *geometric distribution* with success probability $p = 0.013$ on the non-negative integers $\mathbb{N}_0$.

It has the general point probabilities

$$\mathbb{P}(L = n) = (1-p)^n p.$$

It actually follows from the derivation above that

$$\sum_{n=0}^{\infty} (1-p)^n p = 1$$

because this is the sum of the probabilities that $L = n$ for all $n \geq 0$. However, it is also a consequence of the general result on the geometric series

$$\sum_{n=0}^{\infty} s^n = \frac{1}{1-s}$$

for $|s| < 1$. Plugging $s = 1 - p$ into this formula yields that

$$\sum_{n=0}^{\infty} (1-p)^n = \frac{1}{1-(1-p)} = \frac{1}{p},$$

and by multiplication of p on both sides we see that the infinite sum above is 1.

The number of start codons

What is the probability of observing k start codons among the first n codons?

Any configuration of k start codons and $n-k$ non-start codons are equally probable with probability

$$0.013^k \times 0.987^{n-k}.$$

With S the number of start codons

$$\mathbb{P}(S = k) = \binom{n}{k} \times 0.013^k \times 0.987^{n-k}.$$

This is the *binomial distribution* with success probability $p = 0.013$. It has general point probabilities

$$\mathbb{P}(S = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

for $k \in \{0, \ldots, n\}$.

The combinatorial constant

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

is pronounced n-choose-k. It is the number of ways to choose k objects from n objects disregarding the order.

More complicated pattern problems

Start codons do not respect the reading frame I have chosen. More complicated motifs involve wild cards and self-overlap. The loaded die model is not accurate. The mathematical analysis of complicated patterns can be arbitrarily difficult and there are many unsolved theoretical problems from a probabilistic point of view. From a practical point of view, good approximations can often be found, as for the local alignment scores considered earlier, using combinations of theoretical knowledge, computational experiments and statistical analysis.

3.1.1 R interlude: The geometric distribution

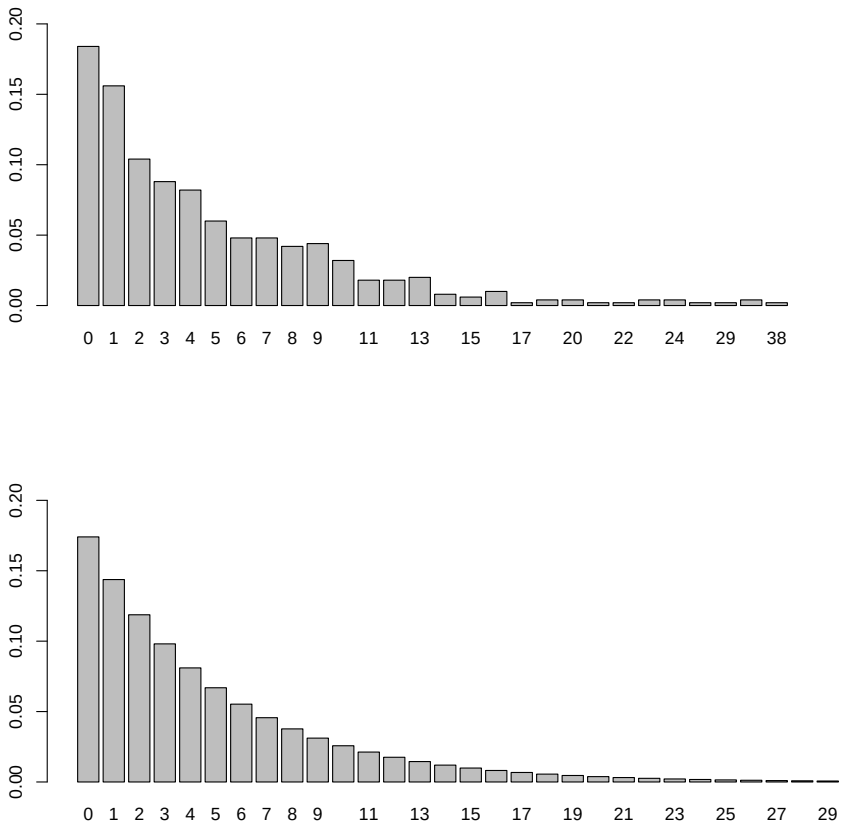
Still studying the dice game from previously we will investigate the distribution of the number of throws before one pattern occurs.

```
diceGame2 <- function(N, pattern = c(1, 4, 3)) {
  len <- numeric(N)    ## Number of throws before pattern start.
  patternLength <- length(pattern)
  stopLength <- patternLength + 1
  for(n in 1:N) {
      test <- 1             ## Control variable, how many outcomes match pattern
      l <- - patternLength  ## To get number of throws before the pattern start.
      while(test < stopLength) {
          temp <- sample(1:6, 1)  ## Random, uniform sampling of a
                                  ## single number between 1 and 6
```

```
          if(temp == pattern[test]) test <- test + 1
          else test <- 1
          l <- l + 1
        }
      len[n] <- l
    }
  return(len)
}
```

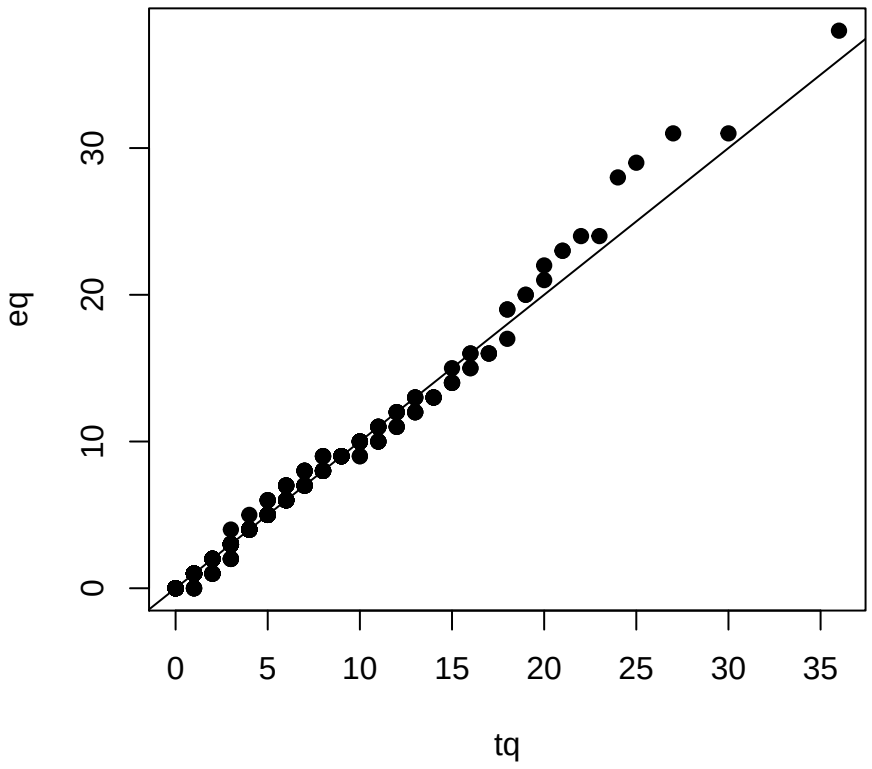
We illustrate the use with the simplest pattern and compare with the relevant geometric distribution. The mean in the geometric distribution with success probability p is $\mu = (1-p)/p$ corresponding to $p = 1/(1+\mu)$.

```
tmp <- diceGame2(500, pattern = 1)
lTab <- table(tmp)  ## Tabulation

barplot(lTab)

mu <- mean(tmp)            ## The empirical mean
p <- 1 / (1 + mu)
par(mfcol = c(2, 1))
barplot(lTab[1:30] / 500, ylim = c(0, 0.20))
barplot(dgeom(0:29, p), names.arg = 0:29, ylim = c(0, 0.20))
```



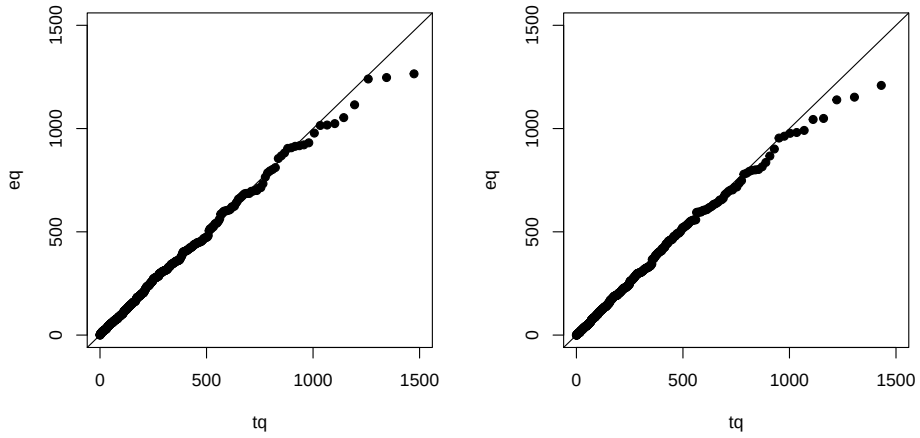
We can also use a QQ-plot.

```
eq <- sort(tmp)
tq <- qgeom((1:500 - 0.5) / 500, prob = p)
plot(tq, eq, pch = 19)
abline(0, 1)
```



For more complicated patterns we do not find a geometric distribution exactly because of "overlaps", but it is not a bad approximation for small and simple patterns, in particularly not for non-self-overlapping patterns.

```
tmp <- diceGame2(500)
p <- 1 / (1 + mean(tmp))
eq <- sort(tmp)
tq <- qgeom((1:500 - 0.5) / 500, prob = p)
par(mfcol = c(1, 2))
plot(tq, eq, pch = 19, xlim = c(0, 1500), ylim = c(0, 1500))
abline(0, 1)

tmp <- diceGame2(500, pattern = c(1, 1, 1))
p <- 1 / (1 + mean(tmp))
eq <- sort(tmp)
tq <- qgeom((1:500 - 0.5) / 500, prob = p)
plot(tq, eq, pch = 19, xlim = c(0, 1500), ylim = c(0, 1500))
abline(0, 1)
```



3.1.2 R digression: Compiling

R is an interpreted language and the execution of calls to `diceGame` and `diceGame2` is slow. The kind of simple but repetitive and iterative computations they represent are generally difficult to implement in R in a way that is competitive with compiled code in terms of execution time. However, there is a possibility to compile the R code using a form of byte code compiler, which can often improve on the execution times at no cost.

```
require(compiler)
diceGameCmp <- cmpfun(diceGame)
diceGameCmp2 <- cmpfun(diceGame2)
```

After the compilation the two byte code compiled functions behave and look just as the original functions. But they execute faster.

```
system.time(diceGame(1000))

##    user  system elapsed
##   1.973   0.005   2.011

system.time(diceGameCmp(1000))

##    user  system elapsed
##   1.173   0.002   1.180

system.time(diceGame2(1000))

##    user  system elapsed
##   2.985   0.008   3.051

system.time(diceGameCmp2(1000))
```

```
##    user  system elapsed
##   1.898   0.007   1.928
```

The improvements are not dramatic but they are noticable, and compiling is worth trying in all cases where execution time becomes a problem. The improvements typically range from no improvements to a factor 2-4 and in some cases perhaps a factor 5-6. All functions in the base packages that ship with R are byte code compiled.

3.1.3 The Poisson Distribution

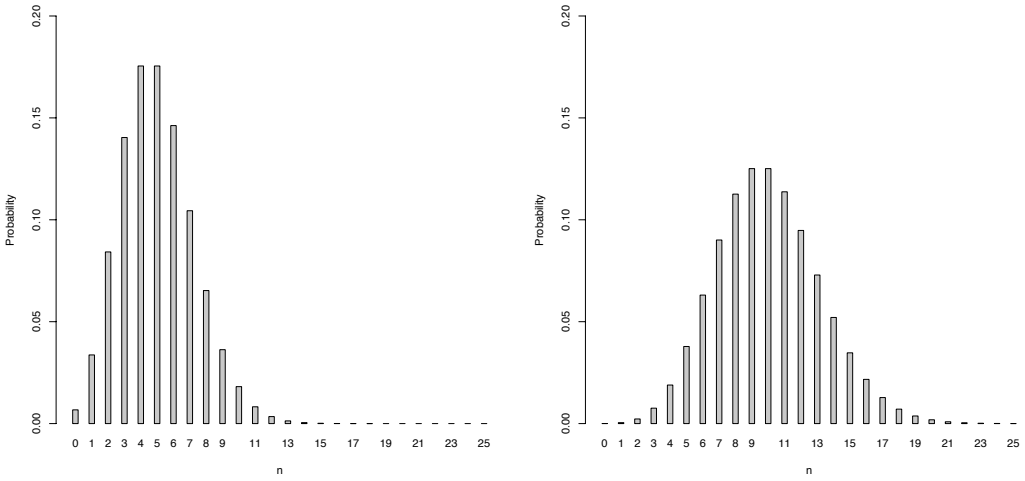


Figure 3.1: The point probabilities for the Poisson distribution with $\lambda = 5$ (left) and $\lambda = 10$ (right).

Recall that the exponential function has the infinite Taylor expansion

$$e^{\lambda} = \sum_{n=0}^{\infty} \frac{\lambda^n}{n!}.$$

The Poisson distribution

On the sample space $\mathbb{N}_0$ of non-negative integers the *Poisson distribution* with parameter $\lambda > 0$ is given by the point probabilities

$$p(n) = e^{-\lambda} \frac{\lambda^n}{n!}$$

for $n \in \mathbb{N}_0$.

The expected value is

$$\sum_{n=0}^{\infty} np(n) = \sum_{n=0}^{\infty} ne^{-\lambda} \frac{\lambda^n}{n!} = \lambda$$

The way to see this is as follows.

$$\begin{aligned}\sum_{n=0}^{\infty} n e^{-\lambda}\frac{\lambda^n}{n!} &= e^{-\lambda}\sum_{n=1}^{\infty}\frac{\lambda^n}{(n-1)!} \\ &= \lambda e^{-\lambda}\sum_{n=1}^{\infty}\frac{\lambda(n-1)}{(n-1)!} \\ &= \lambda e^{-\lambda}\underbrace{\sum_{n=0}^{\infty}\frac{\lambda^n}{n!}}_{e^{\lambda}} = \lambda\end{aligned}$$

where we have used the Taylor series for the exponential function.

The Poisson distribution is not derived from any simple context, as opposed to the geometric distribution and the binomial distribution. It is, however, a universal model for counting phenomena. It holds, in particular, that if n is large and p is small then the Poisson distribution with parameter $\lambda = np$ is a good approximation to the binomial distribution on $\{0, \ldots, n\}$ with success probability p. The fact is that the Poisson distribution is quite widely applicable way beyond being just an approximation of the binomial distribution.

Exercise: Distribution of patterns

Use `sample` as previously to generate random DNA sequences. This time of length 1,000.

Find the distribution of the number of occurrences of the pattern `AATG` using the R function `gregexpr`.

Compare the distribution with the Poisson distribution.

3.1.4 R interlude: The Poisson distribution

Point probabilities for the Poisson distribution with parameter $\lambda = 3$.

```
poispoint <- dpois(0:20, lambda = 3)
poispoint
```

```
##  [1] 4.979e-02 1.494e-01 2.240e-01 2.240e-01 1.680e-01 1.008e-01 5.041e-02
##  [8] 2.160e-02 8.102e-03 2.701e-03 8.102e-04 2.210e-04 5.524e-05 1.275e-05
## [15] 2.732e-06 5.463e-07 1.024e-07 1.808e-08 3.013e-09 4.757e-10 7.135e-11
```

A barplot using various plotting parameter tricks to set a wide linewidth (`lwd`), to set the ends of the lines to be "butted" (`lend = 2`) and not rounded (default) and to get a nice gray color instead of a massive black. See `help(par)` for information in plotting parameters

```
plot(0:20, poispoint, type = "h",
     lwd = 10, lend = 1, col = gray(0.7))
```

Or use the barplot function. See Figure 3.1

```
barplot(poispoint, names.arg = 0:20,
        main = "Poisson point probabilities with lambda=3",
        xlab = "n", ylab = "point probabilities")
```

3.2 Means and differences: The normal model

Keywords: confidence intervals, gene expression, standard error of the mean, t-tests, Wilcoxon test.

ISwR: 93-107

Throughout this lecture we will consider data from a microarray experiment. It is the so-called ALL data set (Chiaretti et. al., Blood, vol. 103, No. 7, 2004). It consists of samples from patients suffering from Acute Lymphoblastic Leukemia. We will consider only those patients with B-cell ALL, and we will group the patients according to presence or absence of the BCR/ABL fusion gene.

Histograms

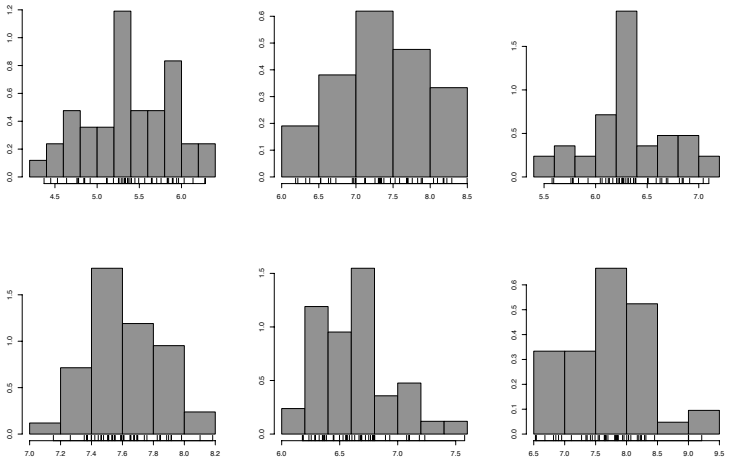


Figure 3.2: Examples of histograms for the log (base 2) expression levels for a random subset of six genes from the ALL data set (non BCR/ABL fusion gene).

Scientific questions

Is there a difference between the two groups in the gene expression example w.r.t. the expression measured for 1635_at?

Are there any genes for which the expression level differ between the two groups? In the confirmatory case, which?

3.2.1 The t-tests and friends

The two group normal model

For the gene expression example with 2 groups we consider the model of the log-expression $x_{i,j}$ for $i = 1$ and $j = 1, \ldots, 37$, $i = 2$ and $j = 1, \ldots, 42$ that $x_{i,j}$ follows a normal distribution $N(\mu_i, \sigma_i^2)$.

That is, $x_{i,j}$ is the j'th log-expression measurement in group i and the within-group distribution is a normal distribution.

The estimators of μ_1 and μ_2 are

$$\hat{\mu}_1 = \bar{x}_1 = \frac{1}{n_1} \sum_{j=1}^{n_1} x_{1,j} \quad \hat{\mu}_2 = \bar{x}_2 = \frac{1}{n_2} \sum_{j=1}^{n_2} x_{2,j}$$

with $n_1 = 37$ and $n_2 = 42$ is our example.

Standard error of the mean

The distribution of the average $\bar{x}_1$ is, under the model assumptions above,

$$\bar{x}_1 \sim N(\mu_1, \frac{\sigma_1^2}{n_1})$$

The standard deviation of the average is the *standard error of the mean* (SEM),

$$\text{SEM} = \frac{\sigma_1}{\sqrt{n_1}}.$$

We usually estimate σ_1 as

$$\hat{\sigma}_1 = \sqrt{\frac{1}{n_1 - 1} \sum_{j=1}^{n_1} (x_{1,j} - \bar{x}_1)^2}$$

and thus SEM as $\hat{\sigma}_1 / \sqrt{n_1}$.

Confidence interval

If we know σ^2, and thus SEM, then

$$\frac{\bar{x}_1 - \mu_1}{\text{SEM}} \sim N(0, 1)$$

and there is approximately 95% probability that the interval

$$[\bar{x}_1 - 1.96 \times \text{SEM}, \bar{x}_1 + 1.96 \times \text{SEM}]$$

contains the mean μ_1.

We don't know SEM but estimate it from data, in which case

$$t = \frac{\bar{x}_1 - \mu_1}{\text{SEM}}$$

has a t-distribution with $n_1 - 1$ degrees of freedom.

With $t_{0.975}$ the 0.975-quantile for this t-distribution the interval

$$[\bar{x}_1 - t_{0.975} \times \text{SEM}, \bar{x}_1 + t_{0.975} \times \text{SEM}]$$

is a 95% *confidence interval* for μ_1.

Note, the probability statement about the confidence interval is a probability statement about the random interval, not about the parameter μ_1. If we repeat the experiment the 95% confidence interval will contain the mean in 95% of the cases and miss the mean in 5% of the cases.

The practical difference between using 1.96 and $t_{0.975}$ in the construction of 95% confidence intervals is little unless n_1 is very small.

In the remaining part of this section we till consider a single gene only out of the total of 12625 measured.

Estimates

BCR/ABL present	BCR/ABL not present
$\bar{x}_1 = \frac{1}{37}\sum_{j=1}^{37} x_{1,j} = 8.54$	$\bar{x}_2 = \frac{1}{42}\sum_{j=1}^{42} x_{2,j} = 7.33$
$\hat{\sigma}_1^2 = \frac{1}{36}\sum_{i=1}^{37}(x_{1,j} - \bar{x}_1)^2 = 0.677$	$\hat{\sigma}_2^2 = \frac{1}{41}\sum_{i=1}^{42}(x_{2,j} - \bar{x}_2)^2 = 0.414$

Confidence intervals for the means are

$$8.54 \pm 2.028 \times \frac{\sqrt{0.677}}{\sqrt{37}} = [8.27, 8.81]$$

and

$$7.33 \pm 2.020 \times \frac{\sqrt{0.414}}{\sqrt{42}} = [7.13, 7.53]$$

Here we used the 0.975-quantiles for the t-distribution with 36 and 41 degrees of freedom, which are 2.028 and 2.020, respectively.

Difference of means

Under the stated model, the distribution of the difference of the means is

$$\bar{x}_1 - \bar{x}_2 \quad \sim \quad N\left(\mu_1 - \mu_2, \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right).$$

The difference $\mu_1 - \mu_2$ is the *parameter of interest* and $\bar{x}_1 - \bar{x}_2$ is the estimator of this parameter.

The *standard error of the difference of means* (SEDM) is

$$\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}.$$

It quantifies the order of magnitude of the random error in the estimate $\bar{x}_1 - \bar{x}_2$ of $\mu_1 - \mu_2$.

A hypothesis

We ask if $\mu_1 = \mu_2$ – this is the scientific hypothesis about equal mean log-expression level in the two groups.

We use the *test statistic* $\bar{x}_1 - \bar{x}_2$ to quantify the observed difference.

We use the *distribution*

$$N\left(0, \frac{\sigma_1^2}{n} + \frac{\sigma_2^2}{m}\right)$$

of the test statistics *under the hypothesis* to decide if the observed quantification is unreasonably large if the hypothesis is true. We *reject* the hypothesis if this is the case.

Statistical Tests - formalization

We consider a parameter of interest θ and two *nested* models, i.e. parameter sets

$$\Theta_0 \subseteq \Theta.$$

We ask for a procedure – a *statistical test* – such that we, for a given observation, can decide whether the parameter θ is in Θ_0 or whether it is in $\Theta \backslash \Theta_0$.

Formalized: We call the assumption that $\theta \in \Theta_0$ the *hypothesis* and write

$$H : \theta \in \Theta_0.$$

Sometimes, the hypothesis is referred to as the null-hypothesis and $\Theta \backslash \Theta_0$ as the *alternative* hypothesis.

In the previous example, the parameter sets are

$$\Theta = \{(\mu_1, \mu_2) \mid \mu_1, \mu_2 \in \mathbb{R}\}$$

and $\Theta_0 = \{(\mu_1, \mu_2) \in \Theta \mid \mu_1 = \mu_2\}$.

Statistical Tests

A test consists of a division of the sample space into disjoint events A and R with A called the *acceptance region* and R the *rejection region*.

We *reject* the hypothesis if our observation fall in R and we *accept* the hypothesis if the observation falls in A. The *level* of the test is

$$\alpha = \max_{\theta \in \Theta_0} P_\theta(R),$$

which is the largest probability of (wrongly) rejecting the test under the hypothesis. The *power* for $\theta \in \Theta$ of the test is

$$\beta(\theta) = P_\theta(R).$$

The power for $\theta \in \Theta \backslash \Theta_0$ is the probability of correctly rejecting the hypothesis under the specific alternative θ.

A good test has small level α and large power $\beta(\theta)$ for all $\theta \in \Theta \backslash \Theta_0$. However, these two interests pull in different directions, and if we enlarge the acceptance set, say, the level as well as the power goes down.

Returning to our gene expression example, we assume for the moment that the variances are known to both be equal to 1, and we will construct rejection regions as subsets of the 79-dimensional sample space with a large numerical difference between the empirical means in the two groups. It plays a central role that the distribution of the difference is known to be a $N(0, 1/37 + 1/42)$-distribution under the hypothesis of equal means.

Difference in group means

If we know both variances $\sigma_1^2 = \sigma_2^2 = 1$ and have $n_1 = 37$ and $n_2 = 42$ the 0.975 quantile for the $N(0, 1/37 + 1/42) = N(0, 0.051)$ distribution is 0.442.

The normal distribution is symmetric – the probability of getting an observed difference ≥ 0.442 or ≤ -0.442 becomes 0.05.

A level 5% test when assuming $\sigma_1^2 = \sigma_2^2 = 1$ is given by the rejection region

$$R = \{x \in \mathbb{R}^{79} \mid |\bar{x}_1 - \bar{x}_2| \geq 0.442\}.$$

We have *estimated* variances $\hat{\sigma}_1^2 = 0.677$ and $\hat{\sigma}_2^2 = 0.414$, which gives an *approximate* level 5% test with rejection region

$$R = \{x \in \mathbb{R}^{79} \mid |\bar{x}_1 - \bar{x}_2| \geq 0.329\}.$$

The estimated difference for the data set is $|8.54 - 7.33| = 1.21$, and the conclusion is that the hypothesis is rejected. The approximation above ignores the fact that the variances are estimated. For small samples the approximation is not that good.

Two sample t-test with equal variances

If $\sigma_1 = \sigma_2 = \sigma$ we estimate σ^2 by the *pooled variance estimator* defined as

$$\hat{\sigma}^2 = \frac{1}{n_1 + n_2 - 2}\left(\sum_{j=1}^{n}(x_{1,j} - \bar{x}_1)^2 + \sum_{j=1}^{m}(x_{2,j} - \bar{x}_2)^2\right).$$

With this setup we introduce the *t-test statistic*

$$t = \frac{\sqrt{\frac{n_1+n_2}{n_1 n_2}}(\bar{x}_1 - \bar{x}_2)}{\hat{\sigma}}.$$

Under the hypothesis that $\mu_1 = \mu_2$ the distribution of t is a *t-distribution with $n_1 + n_2 - 2$ degrees of freedom*. The test is carried out by rejecting the hypothesis for large values of $|t|$.

Two sample t-test without equal variances

If we don't assume equal variances then

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{\hat{\sigma}^2}{n} + \frac{\hat{\sigma}^2}{m}}}$$

where

$$\hat{\sigma}_1^2 = \frac{1}{n_1 - 1}\sum_{j=1}^{n_1}(x_{1,j} - \bar{x}_1)^2 \quad \hat{\sigma}_2^2 = \frac{1}{n_2 - 1}\sum_{j=1}^{n_2}(X_{2,j} - \bar{x}_2)^2$$

Under the hypothesis that $\mu_1 = \mu_2$ the distribution of t is *approximated* by a t-distribution. We reject at level α if

$$|\bar{x}_1 - \bar{x}_2| \geq \text{SEDM} \times t_{1-\alpha/2}$$

where SEDM is the estimated *standard error of the difference of the means* and $t_{1-\alpha/2}$ is the $1 - \alpha/2$ quantile for the appropriate t-distribution.

Note that when assuming equal variances there is a simple formula for the computation of the degrees of freedom and the results are exact under the stated model assumptions. When we don't assume equal variances the t-distribution is an approximation and the relevant degrees of freedom is given by a complicated formula. The latter is referred to as the Welch t-test.

Confidence interval

An alternative to reporting a conclusion from a test as "reject" or "accept" is to report a confidence interval.

A 95% confidence interval for the difference $\mu_1 - \mu_2$ is found as

$$(\bar{x}_1 - \bar{x}_2) \pm \text{SEDM} \times t_{0.975}.$$

The test is rejected at level 5% if and only if the 95% confidence interval does not contain 0. The confidence interval contain more information on the actual difference.

Note that the biggest problem with a difference in variance is not a formal problem with the computation of the test statistic or its distribution. The biggest problem lies in the interpretation. Differences in mean expression level and the variance of the gene at the same time gives a less clear cut interpretation about up- or down-regulation.

Wilcoxon test

The Wilcoxon signed rank test is an alternative to the one-sample t-test for testing if a sample has a specific mean.

The two-sample Wilcoxon test is an alternative to the two-sample t-test for testing if the mean in two groups are equal.

Both test make minimal distributional assumptions and are thus robust to deviations from the normality assumption.

Neither test produce confidence intervals.

These tests are sometimes referred to as *non-parametric* or *distribution free*. Their main justification is that they can be used without worrying too much about distributional assumptions – and perhaps even without giving the distribution much thought. Though the intention is fine, their use may so easily reduce modeling of data to an exercise in formalities and p-value computations.

The two-sample t-test is reasonably robust to deviations from distributional assumptions, and an analysis with a reasonable (graphical) justification of the normality assumption, a t-test and a confidence interval for the difference in means is a much stronger analysis than a two-sample Wilcoxon p-value. However, the t-test (and the estimates of mean and variance) may be sensitive to outliers, and outliers would typically have to be removed from the analysis after identification. For automatic procedures that are supposed to be reasonably robust to outliers the Wilcoxon test may be a suitable alternative to the t-test.

3.2.2 Multiple testing

Multiple testing

name	t-test statistics	p-value
1636_g_at	-9.26	$3.76e-14$
39730_at	-8.69	$4.79e-13$
1635_at	-7.28	$2.45e-10$
1674_at	-6.90	$1.28e-09$
40504_at	-6.57	$5.27e-09$
37015_at	-6.19	$2.74e-08$
40202_at	-6.18	$2.79e-08$
32434_at	-5.78	$1.54e-07$
37027_at	-5.65	$2.60e-07$
39837_s_at	-5.50	$4.74e-07$

The top 10 t-test statistics for all 12625 tests ordered according to p-value.

There are a total of 12625 genes represented on the array. We can do a t-test for each gene where we test if there is a difference in the mean value between those with the BCR/ABL fusion gene and those without. The tests are carried out under the assumption of equal variances, which means that a single t-distribution is used for all p-value computations.

We find our earlier considered gene, 1635_at, as number three from the top on this list. Our earlier conclusions may be problematic if the gene was considered because of its position as number three on this list.

Problems with multiple tests

If gene 1635_at was selected due to its position in the list there are two problems we need to address.

- The difference in gene expression may simply be a random artifact and not a real biological difference – among many genes with no differences, some will show large random differences in a sample.
- Even if there is a difference, the estimated difference may likely be biased upwards.

The problem with multiple statistical testing is that even though the individual tests are all sound the *selection* of tests based on p-value, say, introduces a selection bias in our conclusions. If we make a statistical test at a 5%-level there is 5% chance the we by mistake reject the hypothesis even though it is true. This is not a completely negligible probability but 5% has caught on in the literature as a suitable level. The problem is that if we carry out 100 tests at a 5%-level then we expect that 1 out of 20 tests, that is, 5 in total, reject the hypothesis even if it is true in all the 100 situations. What is perhaps even worse is that the probability of rejecting at least one of the hypothesis is in many cases rather large. If all the tests are *independent*, the number of tests we reject follows a binomial distribution with parameters $n = 100$ and $p = 0.05$, in which case the probability of rejecting at least one hypothesis if they are all true is $1 - (1 - 0.05)^{100} = 99.4\%$.

A binomial computation

If we make n independent tests at level α the number of tests we reject follows a binomial distribution with parameters (n, α).

The probability of rejecting at least one test assuming that the hypothesis of no mean difference is true for all cases is

$$1 - (1 - \alpha)^n.$$

This is called the *family wise error rate*, and if we want to control this at level 0.05, say, we solve for α and get

$$\alpha = 1 - (0.95)^{1/n}.$$

With $n = 12625$ we get

$$\alpha = 1 - (0.95)^{1/12625} = 4.06e - 06.$$

The procedure above for correcting the level of the individual tests to achieve control over the probability of wrongly rejecting a single test is known as the Sidak correction. The correction is valid if the tests are independent. This is sometimes a questionable assumption. A widely used correction, presented here to achieve a 5% family wise error rate, is the Bonferroni correction, which is simply

$$\alpha = 0.05/n = 0.05/12625 = 3.96e - 06.$$

The Bonferroni correction is always more conservative than the Sidak correction (it yields a smaller α), but in this case the difference is quite small. In either case, the resulting procedure can be very conservative.

If we carry out 100 two-sample t-tests on different data sets and find that at a 5% level we rejected in 4 cases the hypothesis that the means are equal, does this support a conclusion that the means are actually different in those 4 cases? No, it does not. If we reject in 30 out of the 100 cases we are on the other hand likely to believe that for a fair part of the 30 cases the means are actually different. The binomial probability of getting more than 10 rejections is 1.1% and getting more than 20 rejections has probability 2.0×10^{-8}. But for how many and for which of the 30 cases can we conclude that there is a difference? A natural thing is to order (the absolute value of) the test statistics

$$|t_{(1)}| \leq \ldots \leq |t_{(100)}|$$

and then take them from the top and down.

Instead of a formal procedure, consider the QQ-plot of the computed t-test statistics against the t-distribution with 77 degrees of freedom or the histogram of the p-values.

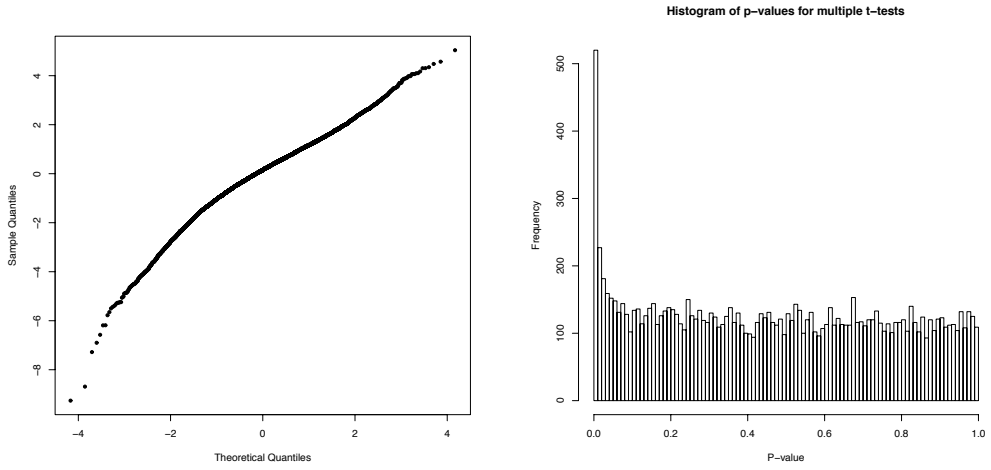


Figure 3.3: QQ-plot of the 12,625 t-test statistics for the ALL dataset (left) and histogram of the corresponding p-values.

The QQ-plot bends in a way that indicates that there are too many large and small values in the sample. This is confirmed by the histogram of p-values, which shows that there are in the order of several hundred p-values too many in the range from 0 to 0.01.

The conclusion in the example above is that there are a number of the cases where we should reject the hypothesis – even in the light of the fact that we do 12625 tests. The real question is that if we continued the list above, when should we stop? What should the threshold for the t-test statistic be in the light of the multiple tests carried out?

Current research suggests that for large, multiple testing problems focus should change from the family wise error rate to other quantities such as the *false discovery rate*, which is the expected relative number of falsely rejected hypotheses out of the total number of rejected hypotheses. The book *Multiple testing procedures with applications to genomics* by Dudoit and van der Laan (Springer, 2008) treats this and a number of other issues in relation to multiple testing problems.

3.2.3 R interlude: ALL microarray analysis

The data used are in the Bioconductor package `ALL` that can be installed from the Bioconductor repository. We extract the subset with B-cell ALL.

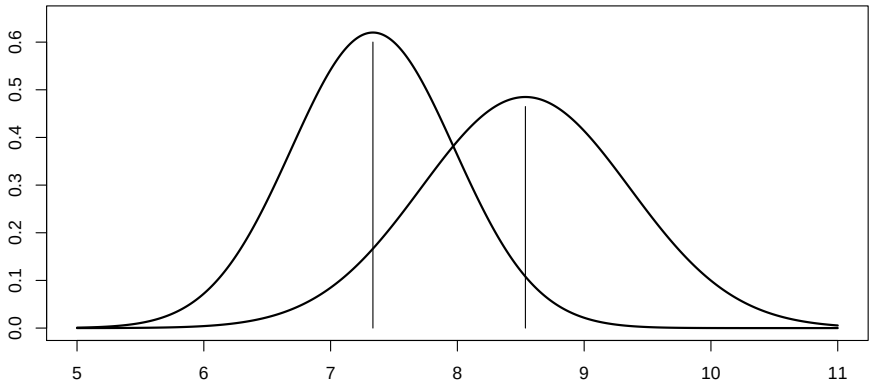
```
require(ALL)
data(ALL)
subset <-  grep("B", as.character(pData(ALL)$BT))
exNEG <- exprs(ALL)[, intersect(which(pData(ALL)$mol.b == "NEG"), subset)]
exBCR <- exprs(ALL)[, intersect(which(pData(ALL)$mol.b == "BCR/ABL"), subset)]
```

Investigation of one particular gene:

```
mu1 <- mean(exBCR["1635_at", ])
mu2 <- mean(exNEG["1635_at", ])
sd1 <- sqrt(var(exBCR["1635_at", ]))
sd2 <- sqrt(var(exNEG["1635_at", ]))
```

A normal model.

```
x <- seq(5, 11, 0.001)
y1 <- dnorm(x, mu1, sd1)
y2 <- dnorm(x, mu2, sd2)
plot(x, y1, ann = FALSE, ylim = c(0,0.65), lwd = 2, type = "l")
lines(x, y2, ann = FALSE, lwd = 2)
lines(c(mu1, mu1), c(0, dnorm(mu1, mu1, sd1) - 0.02))
lines(c(mu2, mu2), c(0, dnorm(mu2, mu2, sd2) - 0.02))
```



One-sample t-tests for each group. Formally tests the hypothesis that the mean is equal to 0 and computes 95% confidence intervals

```
t.test(exBCR["1635_at", ])


##
##  One Sample t-test
##
## data:  exBCR["1635_at", ]
## t = 63.11, df = 36, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  8.262 8.811
## sample estimates:
## mean of x
##     8.536

t.test(exNEG["1635_at", ])


##
##  One Sample t-test
##
## data:  exNEG["1635_at", ]
## t = 73.88, df = 41, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  7.133 7.534
## sample estimates:
## mean of x
##     7.334
```

Formal t-tests without and with the assumption of equal variances.

```
t.test(exBCR["1635_at", ], exNEG["1635_at", ])

##
##  Welch Two Sample t-test
##
## data:  exBCR["1635_at", ] and exNEG["1635_at", ]
## t = 7.168, df = 67.92, p-value = 7.103e-10
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.8679 1.5375
## sample estimates:
## mean of x mean of y
##     8.536     7.334

t.test(exBCR["1635_at", ], exNEG["1635_at", ],
       var.equal = TRUE)

##
##  Two Sample t-test
##
## data:  exBCR["1635_at", ] and exNEG["1635_at", ]
## t = 7.28, df = 77, p-value = 2.446e-10
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.8737 1.5317
## sample estimates:
## mean of x mean of y
##     8.536     7.334
```

By hand computation of the 95% confidence interval for the difference of the means with the equal variance assumption.

```
SEDM <- sqrt((37 + 42) / (37 * 42)) * sqrt((36 * sd1^2 + 41 * sd2^2)/(36 + 41))
mu1 - mu2 + qt(0.975, 77) * c(-1, 1) * SEDM

## [1] 0.8737 1.5317
```

Computations of all t-tests for the 12625 genes. The results are stored in a list.

```
tTestList <- list()
for(i in 1:dim(exBCR)[1])
  tTestList[[i]] <- t.test(exBCR[i, ], exNEG[i, ], var.equal = TRUE)
```

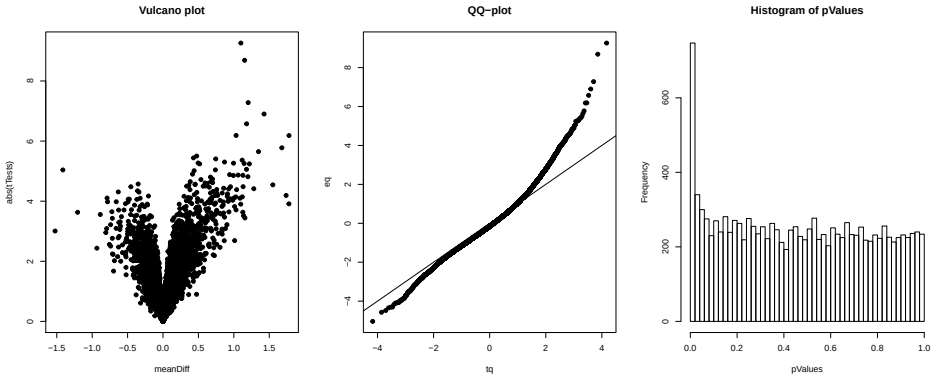
Extraction of mean differences, test-statistics and p-values from the list of t-test objects.

```
meanDiff <- sapply(tTestList, function(t) t$estimate[1] - t$estimate[2])
tTests <- sapply(tTestList, function(t) t$statistic)
pValues <- sapply(tTestList, function(t) t$p.value)
```

A vulcano plot may be a useful visualization of the result. You can see the relation between large estimated differences and large absolute values of the test statistic.

QQ-plots and histograms of p-values are useful to get an idea about how many genes actually show a difference in expression value between the two cases. The QQ-plot is against the theoretical t-distribution with 77 degrees of freedom.

```
eq <- sort(tTests)
tq <- qt((1:length(tTests) - 0.5)/length(tTests), 77)
par(mfcol = c(1, 3))
plot(meanDiff, abs(tTests), pch = 19, main = "Vulcano plot")
plot(tq, eq, pch = 19, main = "QQ-plot")
abline(0, 1)  ## There are no parameters involved here.
hist(pValues, breaks = 40)
```



Top 10 genes ordered according to absolute value of t-test statistic.

```
top10 <- order(abs(tTests), decreasing = TRUE)[1:10]
top10table <- data.frame(geneid = rownames(exBCR)[top10],
                         meanDiff = meanDiff[top10],
                         tTest = tTests[top10],
                         pValue = pValues[top10]
                         )
top10table
```

```
##        geneid meanDiff tTest    pValue
## 1   1636_g_at   1.1000 9.261 3.762e-14
## 2    39730_at   1.1525 8.688 4.792e-13
## 3     1635_at   1.2027 7.280 2.446e-10
## 4     1674_at   1.4272 6.901 1.281e-09
## 5    40504_at   1.1810 6.574 5.265e-09
## 6    37015_at   1.0327 6.188 2.740e-08
## 7    40202_at   1.7794 6.184 2.785e-08
## 8    32434_at   1.6786 5.776 1.536e-07
## 9    37027_at   1.3487 5.648 2.601e-07
## 10 39837_s_at   0.4757 5.502 4.741e-07
```

Note that if we use the Welch t-test instead of the equal variance t-test as above, the degrees of freedom will change from gene to gene. We could extract this number from the t-test objects, but in principle the computed test statistics should be compared to different t-distributions and the QQ-plot is no longer sensible. However, the t-distributions will resemble

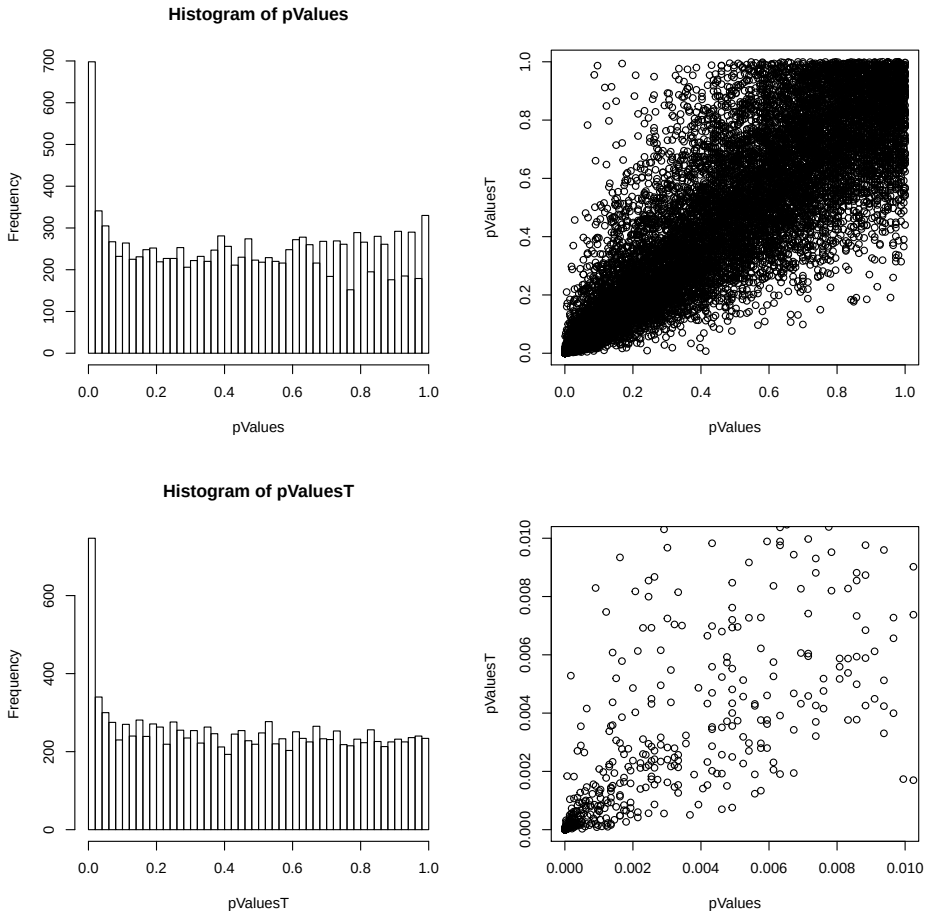
normal distributions with mean 0 and variance 1, and we could use this distribution as a surrogate. First we do this for the t-tests with the equal variance assumption.

```
qqnorm(tTests, pch = 19)
abline(0, 1) ## Preferable over 'qqline' here.
```

Try to repeat the computations above, but without the equal variance assumption. The difference is small. The list of top 10 genes is almost unchanged in this case.

We can also compare with the non-parametric Wilcoxon test. The `wilcox.test` function will warn against non being able to compute exact p-values with ties.

```
tTestList <- list()
wTestList <- list()
for(i in 1:dim(exBCR)[1]) {
  wTestList[[i]] <- wilcox.test(exBCR[i, ], exNEG[i, ])
  tTestList[[i]] <- t.test(exBCR[i, ], exNEG[i, ], var.equal = TRUE)
}

pValues <- sapply(wTestList, function(w) w$p.value)
pValuesT <- sapply(tTestList, function(t) t$p.value)
par(mfcol = c(2, 2))
hist(pValues, breaks = 40)
hist(pValuesT, breaks = 40)
plot(pValues, pValuesT)
plot(pValues, pValuesT,
     xlim = c(0, 0.01), ylim = c(0, 0.01)
     )
```



The top 10 genes according to the Wilcoxon test.

```
top10 <- order(abs(pValues))[1:10]
top10table <- data.frame(geneid = rownames(exBCR)[top10],
                         pValue = pValues[top10],
                         pValueT = pValuesT[top10]
                         )
top10table
```

```
##        geneid    pValue   pValueT
## 1   1636_g_at 8.306e-13 3.762e-14
## 2    39730_at 2.158e-12 4.792e-13
## 3    40504_at 1.830e-09 5.265e-09
## 4     1635_at 2.310e-09 2.446e-10
## 5     1674_at 3.390e-09 1.281e-09
## 6    37015_at 4.802e-08 2.740e-08
## 7  35162_s_at 8.739e-08 2.993e-06
## 8    40202_at 1.291e-07 2.785e-08
## 9    32434_at 3.749e-07 1.536e-07
## 10 39837_s_at 4.780e-07 4.741e-07
```

3.3 Exercises

For the following two exercises we consider the `BUtraffic` data set

`http://www.math.ku.dk/~richard/courses/StatScience2011/BUtraffic.txt`

introduced in Exercise 2.7.

Exercise 3.1 Compute a 95% confidence interval for the mean of the log size of the downloads.

Exercise 3.2 Break the data into two groups according to whether the time of observation is smaller or larger than 972500. Use the t-test to compare the means of the log size for the two groups. Does it make a difference in the comparison above whether we assume equal variances in the two groups or not.

Chapter 4

Fourth Week

4.1 Molecular evolution

Keywords: conditional distribution, likelihood function, maximum-likelihood estimator, molecular evolution.

Principled modeling

Up to this point we have considered:

- Ad hoc methods for the estimation of parameters – equating empirical and theoretical quantities.
- Ad hoc test statistics and confidence interval constructions for specific cases.

We have introduced workhorse methods like:

- Estimation of mean, variance or location, scale parameters.
- Estimating proportions (probabilities).
- Comparing proportions (tables, χ^2-tests).
- Comparing means (t-test).

Is there a unified, principled approach behind those methods?

The importance of a principled approach is not so much to "be able to follow the right path for the true believers", as it is to provide a reasonably general, powerful tool that can be adapted for new problems. Up to this point in the course you have learned a limited number of special case tools, which can be used to solve special problems.

As a case for the introduction of principled approach to statistical analysis we introduce a few models of molecular evolution.

The simplest models we are going to consider regard each nucleic acid in the DNA-sequence as evolving independently according to a dynamic model that depends upon the time between observations.

Sequences evolve according to a complicated interaction of random mutations and selection, where the random mutations can be single nucleotide substitutions, deletions or insertions, or higher order events like inversions or crossovers. We will only consider the substitution process. Thus we consider two DNA sequences that are evolutionary related via a number of nucleotide substitutions. We will regard each nucleotide position as unrelated to each other, meaning that the substitution processes at each position are independent. We are interested in a model of these substitution processes. We are especially interested in how the evolutionary distance – measured, for instance, in calendar time – enters into the models.

The models will from a biological point of view be very simplistic and far from realistic as models of real sequence evolution processes. However, they form the starting point for more serious models, if one, for instance, wants to enter the area of phylogenetics, and they are well suited to illustrate and train the fundamental concepts of a statistical models and the methods of statistical inference.

Obtaining data is also a little tricky, since we can rarely go out and read of evolutionary related sequences where we know the relation "letter by letter" – such relations are on the contrary established computationally using alignment programs. However, in some special cases, one can actually observe real evolution as a number of substitutions in the genome. This is for instance the case for rapidly evolving RNA-viruses.

Such a data set was obtained for the H strain of the Hepatitis C virus (HCV) (Ogata et al., Proc. Natl. Acad. Sci., 1991 (88), 3392-3396). A patient, called patient H, was infected by HCV in 1977 and remained infected at least until 1990 – for a period of 13 years. In 1990 a research group sequenced three segments of the HCV genome obtained from plasma collected in 1977 as well as in 1990. The three segments, denoted segment A, B and C, were all directly alignable without the need to introduce insertions or deletions. The lengths of the three segments are 2610 (A), 1284 (B) and 1029 (C) respectively.

Molecular evolution

Position	42	275	348	447	556	557	594	652	735	888	891	973	979	1008	1011	1020	1050	1059	1083	1149	1191	1195	1224	1266
H77	G	C	C	A	G	C	C	C	T	C	T	G	G	C	G	C	T	T	C	T	T	T	T	A
H90	A	T	T	G	A	T	T	T	C	T	C	A	A	T	A	T	A	C	T	C	C	A	C	G

Table 4.1: The segment position and nucleotides for 24 mutations on segment A of the Hepatitis C virus.

		H90			
		A	C	G	T
H77	A		1	11	1
	C	4		1	20
	G	13	3		1
	T	3	19	1	

segment A

		H90			
		A	C	G	T
H77	A		0	5	0
	C	1		0	8
	G	1	1		1
	T	2	6	0	

segment B

		H90			
		A	C	G	T
H77	A		1	2	0
	C	1		2	5
	G	4	0		0
	T	1	3	1	

segment C

Table 4.2: Tabulation of all mutations in the three segments A, B and C of the hepatitis C virus genome from the 1977 H strain to the 1990 H strain.

In Table 4.1 we see the position for the first 24 mutations as read from the 5'-end of segment

A out of the total of 78 mutations on segment A. In Table 4.2 we have tabulated all the mutations in the three segments.

Example

Consider the sample space

$$\{A, C, G, T\} \times \{A, C, G, T\},$$

and let X and Y denote random variables representing two *evolutionary related* nucleic acids in a DNA sequence.

Let the joint distribution of X and Y have point probabilities

	A	C	G	T
A	0.1272	0.0063	0.0464	0.0051
C	0.0196	0.2008	0.0082	0.0726
G	0.0556	0.0145	0.2151	0.0071
T	0.0146	0.0685	0.0069	0.1315

Independence and point probabilities

Definition 3. The discrete random variables X and Y are independent if

$$\mathbb{P}(X = x, Y = y) = \mathbb{P}(X = x)\mathbb{P}(Y = y).$$

In words: The random variables are independent if and only if the point probabilities for their joint distribution factorize as a product of the point probabilities for their marginal distributions.

Example

		Y				
		A	C	G	T	
X	A	0.1272	0.0063	0.0464	0.0051	0.1850
	C	0.0196	0.2008	0.0082	0.0726	0.3012
	G	0.0556	0.0145	0.2151	0.0071	0.2923
	T	0.0146	0.0685	0.0069	0.1315	0.2215
		0.2170	0.2901	0.2766	0.2163	

Same example as above but with the point probabilities for the marginal distributions. Note that X and Y are not independent! For instance

$$\begin{aligned} 0.1272 &= \mathbb{P}((X, Y) = (A, A)) \\ &\neq \mathbb{P}(X = A) \times \mathbb{P}(Y = A) = 0.1850 \times 0.2170 = 0.0401 \end{aligned}$$

Example

		Y				
		A	C	G	T	
X	A	0.0401	0.0537	0.0512	0.0400	0.1850
	C	0.0654	0.0874	0.0833	0.0651	0.3012
	G	0.0634	0.0848	0.0809	0.0632	0.2923
	T	0.0481	0.0643	0.0613	0.0479	0.2215
		0.217	0.2901	0.2766	0.2163	

Same marginals as above but X and Y are independent in this example.

Conditional distributions

Definition 4. The conditional distribution of Y given that $X \in A$ is defined as

$$\mathbb{P}(Y \in B | X \in A) = \frac{\mathbb{P}(Y \in B, X \in A)}{\mathbb{P}(X \in A)}$$

provided that $\mathbb{P}(X \in A) > 0$.

If X and Y are discrete we can condition on events $X = x$ and get conditional distributions in terms of point probabilities

$$p(y|x) = \mathbb{P}(Y = y | X = x) = \frac{\mathbb{P}(Y = y, X = x)}{\mathbb{P}(X = x)} = \frac{p(x, y)}{\sum_y p(x, y)}$$

where $p(x, y)$ are the joint point probabilities.

Example

Using

$$\mathbb{P}(Y = y | X = x) = \frac{\mathbb{P}(X = x, Y = y)}{\mathbb{P}(X = x)} = \frac{p(x, y)}{\sum_{y \in E} p(x, y)}.$$

we have to divide by precisely the row sums to get the matrix of conditional distributions:

		Y			
		A	C	G	T
X	A	0.6874	0.0343	0.2507	0.0276
	C	0.0649	0.6667	0.0273	0.2411
	G	0.1904	0.0495	0.7359	0.0242
	T	0.0658	0.3093	0.0311	0.5938

The row sums above equal 1 and this is an example of a matrix of *transition probabilities*.

The Jukes-Cantor model

With $t \geq 0$ denoting time from the observation of the (discrete) variable X to the observation of the variable Y we specify the conditional distribution, P^t, of $Y = y$ given $X = x$ by

$$\begin{aligned} P^t(x,x) &= 0.25 + 0.75 \times \exp(-4\alpha t) \\ P^t(x,y) &= 0.25 - 0.25 \times \exp(-4\alpha t), \quad \text{if } x \neq y, \end{aligned}$$

with $\alpha \geq 0$ a parameter.

In molecular evolution with sample space $\{A, C, G, T\}$ this model is known as the Jukes-Cantor model.

4.1.1 The Jukes-Cantor model

We want to consider pairs of nucleotides (X_i, Y_i) from the sample space $\{A, C, G, T\} \times \{A, C, G, T\}$ that are evolutionary related. We assume that the pairs are identically distributed, that is

$$\begin{aligned} \mathbb{P}(X_i = x, Y_i = y) &= p(x)P^t(x,y) \\ &= \begin{cases} p(x)(0.25 + 0.75 \times \exp(-4\alpha t)) & \text{if } x = y \\ p(x)(0.25 - 0.25 \times \exp(-4\alpha t)) & \text{if } x \neq y \end{cases}. \end{aligned}$$

The unknown parameters are $\alpha > 0$ and the four-dimensional probability vector p. Perhaps t is also an unknown parameter.

We assume that $(X_1, Y_1), \ldots, (X_n, Y_n)$ are *independent* with

$$\mathbb{P}_{t,p,\alpha}((X_i, Y_i) = (x, y)) = P_{t,p,\alpha}(x, y) = p(x)P_\alpha^t(x, y).$$

where p is a vector of point probabilities on $\{A, C, G, T\}$ and $P_\alpha^t(x, y)$ is the conditional probability that x mutates into y in time t.

The probability of observing the data $z = ((x_1, y_1), \ldots, (x_n, y_n))$ is

$$\begin{aligned} &\mathbb{P}((X_1, Y_1) = (x_1, y_1), \ldots, (X_n, Y_n) = (x_n, y_n)) \\ &= \prod_{i=1}^{n} P_{t,p,\alpha}(x_i, y_i) \\ &= \prod_{i=1}^{n} p(x_i)P_\alpha^t(x_i, y_i). \end{aligned}$$

The argument behind the product formula above is that the pairs of nucleotides are assumed independent. Thus the joint probability factorizes as a product of the marginal probabilities. Moreover, the pairs of nucleotides are assumed identically distributed, which means that each factor is from the same distribution.

The likelihood function

Inverting the point of view on the probability computed above as a function of the *unknown parameter* we get the *likelihood function*

$$\mathscr{L}_z(t,p,\alpha) = \prod_{i=1}^{n} P_{t,p,\alpha}(x_i,y_i) = \prod_{i=1}^{n} p(x_i)P_\alpha^t(x_i,y_i).$$

Definition 5. The maximum-likelihood estimator (MLE) of the unknown parameter(s) is the maximizer of the likelihood function.

We may encounter situations in practice, where there is no MLE.

For practical as well as theoretical reasons we often compute the log-likelihood or the minus-log-likelihood. Finding the MLE is equivalent to finding the maximum of the log-likelihood function or the minimum of the minus-log-likelihood.

The minus-log-likelihood function is

$$l_z(t,p,\alpha) = -\log \mathscr{L}_z(t,p,\alpha) = -\sum_{i=1}^{n} \log p(x_i) - \sum_{i=1}^{n} \log P_\alpha^t(x_i,y_i).$$

Observe that the first term depends upon p only and the second term on (t,α) only. *To find the MLE we can minimize each term separately over the separate parameters.*

Introducing $n(x,y)$ as the number of observed mutations of x to y, we can rewrite

$$\tilde{l}_z(\alpha) = -\sum_{i=1}^{n} \log P_\alpha^t(x_i,y_i) = -\sum_{x,y} n(x,y) \log P_\alpha^t(x,y).$$

To estimate α (and t?) using MLE we need to minimize this function as a function of α (and t).

The last rewriting of the likelihood function is a typical tabulation for discrete observations. We see that the likelihood function only depends on the data through the table of observed mutations.

Finding the MLE

We can use calculus to find the MLE. It holds that $\tilde{\alpha}$ is a *local* minimizer for $\tilde{l}_z(\alpha)$ if

$$\begin{aligned} \frac{\mathrm{d}\tilde{l}_z}{\mathrm{d}\alpha}(\tilde{\alpha}) &= 0 \quad \text{and} \\ \frac{\mathrm{d}^2\tilde{l}_z}{\mathrm{d}\theta^2}(\tilde{\alpha}) &> 0. \end{aligned}$$

The *global* minimizer of l_z is found among the local minimizers unless l_z attains the global minimum at the boundary of the parameter space or "misbehaves" when the boundary is approached.

For the parameter α in the open interval $(0, \infty)$ it holds that if there is a *unique* stationary point $\tilde{\alpha} \in (0, \infty)$, which is also a minimum, then $\tilde{\alpha}$ is the global minimizer.

For the Jukes-Cantor model we have that

$$\begin{aligned} P_\alpha^t(x,x) &= 0.25 + 0.75 \times \exp(-4\alpha t) \\ P_\alpha^t(x,y) &= 0.25 - 0.25 \times \exp(-4\alpha t), \quad \text{if } x \neq y. \end{aligned}$$

We consider the case with fixed and known $t > 0$ and $\alpha \geq 0$ the unknown parameter. Introducing

$$n_1 = \sum_x n(x,x) \quad \text{and} \quad n_2 = \sum_{x \neq y} n(x,y)$$

we find that

$$\begin{aligned} \tilde{l}_z(\alpha) &= -n_1 \log(0.25 + 0.75 \times \exp(-4\alpha t)) \\ &\quad -n_2 \log(0.25 - 0.25 \times \exp(-4\alpha t)). \end{aligned}$$

The derivative is

$$\frac{\mathrm{d}\tilde{l}_z}{\mathrm{d}\alpha}(\alpha) = 4t \exp(-4\alpha t) \left(\frac{3n_1}{1 + 3\exp(-4\alpha t)} - \frac{n_2}{1 - \exp(-4\alpha t)} \right)$$

and the *likelihood equation* $\frac{\mathrm{d}\tilde{l}_z}{\mathrm{d}\alpha}(\alpha) = 0$ is therefore equivalent to the equation

$$3n_1(1 - \exp(-4\alpha t)) = n_2(1 + 3\exp(-4\alpha t)).$$

This equation has a (unique) solution if and only if $3n_1 > n_2$ in which case

$$\hat{\alpha} = \frac{1}{4t} \log \frac{3(n_1 + n_2)}{3n_1 - n_2} = \frac{1}{4t} \log \frac{3n}{3n_1 - n_2}$$

is the maximum likelihood estimator.

There is some intuition behind the formula. The conditional probability $P^t(x,x)$ is the probability that a nucleotide does not change over the time period considered. For the Hepatitis C virus data set, Table 4.2 shows that out of the 2610 nucleotides in segment A there are 78 that have mutated over the period of 13 years leaving 2532 unchanged. With reference to the frequency interpretation we can estimate α in the Jukes-Cantor model by equating the formula for $P^t(x,x)$ equal to 2532/2610. In general terms this gives that we should solve for α in the equation

$$0.25 + 0.75 \times \exp(-4\alpha t) = \frac{n_1}{n_1 + n_2},$$

which is solved by the formula above whenever $3n_1 > n_2$. For the particular example this gives

$$\hat{\alpha} = -\frac{\log((\frac{2532}{2610} - \frac{1}{4})\frac{4}{3})}{4 \times 13} = 7.8 \times 10^{-4}.$$

However, the important point is that deriving an estimator based on the likelihood function is an general and principled method. That the result is interpretable and intuitive gives credit to the method.

Working with the minus-log-likelihood, and in particular differentiation in the α-parametrization, is principled but hideous. It is much easier to make a reparametrization by

$$\gamma = \gamma(\alpha) = 0.25 - 0.25 \exp(-4\alpha t)$$

such that

$$\alpha = \alpha(\gamma) = \frac{1}{4t} \log(1 - 4\gamma)$$

for $\gamma \in (0, 0.25)$. In the γ-parameter the minus-log-likelihood becomes

$$l_z(\gamma) = -n_1 \log(1 - 3\gamma) - n_2 \log \gamma$$

for $\gamma \in (0, 0.25)$. The differentiation is easier yielding the derivative

$$l_z'(\gamma) = \frac{3n_1}{1 - 3\gamma} - \frac{n_2}{\gamma},$$

and solving the likelihood equation is always possible for $\gamma \in (0, 1/3)$ and gives

$$\hat{\gamma} = \frac{n_2}{3n}.$$

The solution is in $(0, 0.25)$ if and only if $3n_1 > n_2$ in which case we get

$$\hat{\alpha} = \alpha(\hat{\gamma}) = \frac{1}{4t} \log\left(1 - 4\frac{n_2}{3n}\right) = \frac{1}{4t} \log \frac{3n}{3n_1 - n_2}.$$

Hepatitis C example

	Segment			
	A	B	C	A+B+C
n_1	2532	1259	1009	4800
n_2	78	25	20	123
$\hat{\alpha}$	7.8×10^{-4}	5.0×10^{-4}	5.0×10^{-4}	6.5×10^{-4}

Estimated mutation rates for the three segments separately and combined.

4.1.2 R interlude: Hepatitis C virus evolution

Load the data.

```
HepCevol <- read.table("http://www.math.ku.dk/~richard/courses/StatScience2011/HepCevol.txt")
```

The following implementation is a little fancier than needed, but can perhaps illustrate some quite nifty things in R.

First we write a function that, given an α parameter and time value, returns the matrix of transition probabilities. The default is `time = 1` to make the function work even if we don't specify time.

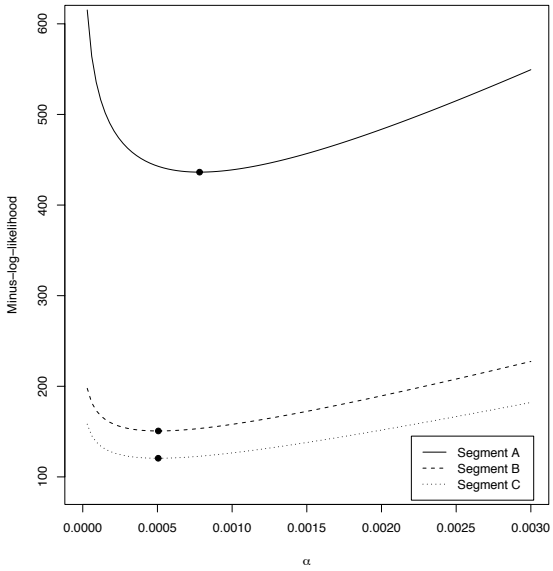


Figure 4.1: The minus-log-likelihood functions and corresponding MLE for the Hepatitis C example.

```
tpJC <- function(alpha, time = 1) {
  z <- -4 * alpha * time
  xx <- 0.25 + 0.75 * exp(z)
  xy <- 0.25 - 0.25 * exp(z)
  tp <- matrix(xy, ncol = 4, nrow = 4)
  diag(tp) <- xx
  return(tp)
}
```

Then we implement the computation of the minus-log-likelihood function from a tabulation of the number of transitions as presented in the lecture. Note that this is a one-line implementation of a generic likelihood computation, where the transition probabilities can be given as any function of any (univariate or multivariate) parameter theta. The default is that `tp` are computed using `tpJC` as implemented above.

```
mll <- function(theta, x , tp = tpJC, ...)
  - sum(log(tp(theta, ...)) * x)
```

We implement the computation of the MLE for the JC-model based on a tabulation of the transitions.

```
alphahat <- function(x, t){
  n1 <- sum(diag(x))
  n2 <- sum(x) - n1
  if(3 * n1 > n2) {
```

```
    alpha <- log(3 * (n1 + n2)/(3 * n1 - n2)) / (4 * t)
  } else {
    alpha <- Inf
  }
  return(alpha)
}
```

Then we tabulate our data for the three segments and compute the MLEs.

```
xSegA <- table(HepCevol[HepCevol$segment == "A", c(4, 3)])
diag(xSegA) <- c(470, 761, 746, 555)

xSegB <- table(HepCevol[HepCevol$segment == "B", c(4, 3)])
diag(xSegB) <- c(252, 389, 347, 271)

xSegC <- table(HepCevol[HepCevol$segment == "C", c(4, 3)])
diag(xSegC) <- c(230, 299, 282, 198)

alphaHat <- c(alphahat(xSegA, 13),
              alphahat(xSegB, 13),
              alphahat(xSegC, 13))
```

And finally we make a plot, see Figure 4.1. The explicit vectorization in `theta` is needed for the plotting using `curve`.

```
mll <- Vectorize(mll, "theta")
curve(mll(x, xSegA, time = 13), 0, 0.003, ylim = c(90, 600),
      ylab = "Minus-log-likelihood", xlab = expression(alpha)
)
curve(mll(x, xSegB, time = 13), 0, 0.003, col = "red", add = TRUE)
curve(mll(x, xSegC, time = 13), 0, 0.003, col = "blue", add = TRUE)
legend(0.0015, 400,
       legend = c("Segment A", "Segment B", "Segment C"),
       col = c("black", "red", "blue"),
       lty = c(1, 1, 1)
)

points(alphaHat,
       c(mll(alphaHat[1], xSegA, time = 13),
         mll(alphaHat[2], xSegB, time = 13),
         mll(alphaHat[3], xSegC, time = 13)),
       pch = 19)
```

4.2 Likelihood

Keywords: Analytic optimization, exponential distribution, Gumbel distribution, log-likelihood function, Newton-Raphson algorithm, normal distribution, profile log-likelihood function.

4.2.1 The exponential distribution

Neuronal interspike times

We measure the times between *spikes* of a neuron in a steady state situation.

We attempt to model the interspike times using the exponential distribution, which is the probability distribution on $[0, \infty)$ with density

$$f_\lambda(x) = \lambda \exp(-\lambda x), \quad x \geq 0$$

where $\lambda > 0$ is an unknown parameter.

Exponential distribution

The sample space for a single observation is $[0, \infty)$, the parameter space is $(0, \infty)$.

We want a probability distribution P_λ of n independent and identically (i.i.d.) exponentially distributed random variables $X_1, \ldots, X_n$ with intensity parameter λ.

The distribution of X_i has density

$$f_\lambda(x) = \lambda \exp(-\lambda x)$$

for $x \geq 0$. The probability distribution P_λ has density

$$\begin{aligned} f_\lambda(x_1, \ldots, x_n) &= \lambda \exp(-\lambda x_1) \cdot \ldots \cdot \lambda \exp(-\lambda x_n) \\ &= \lambda^n \exp(-\lambda(x_1 + \ldots + x_n)). \end{aligned}$$

With $(0, \infty)$ the parameter space, the family $(P_\lambda)_{\lambda \in (0,\infty)}$ of probability distributions is a statistical model.

The concept of a multivariate density as the product of univariate densities may be a little difficult to grasp. It means that the joint distribution of $X_1, \ldots, X_n$ can be expressed using multivariate integrals

$$\mathbb{P}(X_1 \in A_1, \ldots, X_n \in A_n) = \int_{A_1} \ldots \int_{A_n} f_\lambda(x_1, \ldots, x_n) \mathrm{d}x_n \ldots \mathrm{d}x_1.$$

It is often impossible to compute these integrals, but it is nevertheless a way to specify the joint probability. The most important part to take note of is that the joint density is given as a product of the marginal densities if the variables are independent. Thus the density is in itself computable.

Exponential distribution

Consider n i.i.d. exponentially distributed random variables with parameter λ whose joint distribution P_λ has density

$$f_\lambda(x_1, \ldots, x_n) = \lambda^n \exp\left(-\lambda \sum_{i=1}^n x_i\right) = \mathscr{L}_x(\lambda)$$

for $x_1, \ldots, x_n \geq 0$ and $\lambda > 0$.

The *likelihood function* is $\mathscr{L}_x(\lambda)$ and the minus-log-likelihood function is

$$l_x(\lambda) = -\log \mathscr{L}_x(\lambda) = \lambda \sum_{i=1}^{n} x_i - n \log \lambda.$$

Exercise: Find the MLE (the minimizer of $\ell_x(\lambda)$.

Exponential distribution – location-scale model

Introducing a location parameter in the expontial distribution and making n i.i.d. observations the likelihood becomes

$$\mathscr{L}_x(\mu, \lambda) = \begin{cases} \lambda^n e^{-\lambda \sum_{i=1}^{n}(x_i - \mu)} & \mu \leq \min\{x_1, \ldots, x_n\} \\ 0 & \text{otherwise} \end{cases}$$

Fixing $\mu \in [0, \min\{x_1, \ldots, x_n\}]$ the (unique) maximizer in λ is

$$\hat{\lambda}(\mu) = \frac{n}{\sum_{i=1}^{n}(x_i - \mu)}$$

The profile likelihood function is

$$\mathscr{L}_x(\mu, \hat{\lambda}(\mu)) = \hat{\lambda}(\mu)^n e^{-n} = \left(\frac{n}{\sum_{i=1}^{n}(x_i - \mu)} \right)^n e^{-n}$$

which is *monotonely increasing in* μ. Thus, $\hat{\mu} = x_{(1)} = \min\{x_1, \ldots, x_n\}$.

One has to be a little careful with the formalities. If $n = 1$ (there is only one observation), then there is no maximizer in λ for $\mu = x_1$, nor is there a maximizer if $n > 1$ and all the observations are equal. However, whenever you have at least two different observations there is a maximizer in λ for $\mu = x_{(1)}$ and this is the MLE of λ and the MLE of μ is $x_{(1)}$ (the smallest observation). This is a quite problematic estimator in several respects, or one should perhaps say that the resulting model is quite problematic. Can you think of why?

Finding the MLE in general

For continuous multivariate parameters we can in principle use calculus and results on multivariate optimization.

In general, for univariate and in particular for multivariate parameters, we cannot solve the *likelihood equation* let alone justify that a solution is a global minimum of l_x.

Numerical methods are needed. All methods start with an *initial guess* and iteratively improve on that guess. The *Newton-Raphson* algorithm attempts to solve the likelihood equation. `optimize` and `optim` in `R` can do numerical optimization in the univariate and the multivariate case, respectively.

In general, there is *no way* we can tell for sure if an algorithm has converged to the global minimum.

4.2.2 The normal distribution

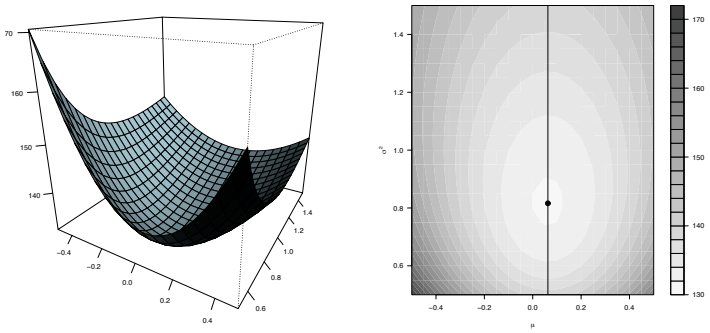
Let $X_1, \ldots, X_n$ be n i.i.d. random variables with the $N(\mu, \sigma^2)$ distribution. The statistical model is given by the sample space $\mathbb{R}^n$, the parameter space $\mathbb{R} \times (0, \infty)$, the parameter $\theta = (\mu, \sigma^2)$ and P_θ the probability distribution given by a density. The minus-log-likelihood function when observing $x = (x_1, \ldots, x_n)$ is

$$l_x(\mu, \sigma^2) = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2 + \frac{n}{2} \log \sigma^2 + n \log \sqrt{2\pi}.$$

The minimization is a two-step procedure. Fix σ^2 and minimize over μ, then minimize over σ^2. The result is

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2.$$

Normal minus-log-likelihood - simulated



The minus-log-likelihood function (left) and a contour plot (right) for the scale-location parameters (μ, σ^2) based on $n = 100$ simulations of i.i.d. $N(0, 1)$-distributed variables. The MLE is $\hat{\mu} = 0.063$ and $\tilde{\sigma}^2 = 0.816$. The profile minus-log-likelihood function of σ^2 is given by evaluating the minus-log-likelihood function along the straight line, as shown on the contour plot, given by $\mu = \hat{\mu}$.

Theoretical optimization

Fixing σ^2 we differentiate w.r.t. μ

$$\frac{\mathrm{d}}{\mathrm{d}\mu} l_x(\mu, \sigma^2) = -\frac{2}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu) = \frac{1}{\sigma^2} (n\mu - \sum_{i=1}^{n} x_i),$$

and if we equate this equal to 0 there is a unique solution $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i$. A second differentiation shows that this is a local minimum and since it is the unique solution, it is a global minimum in μ for fixed σ^2.

$$\begin{aligned} \min_{\mu, \sigma^2} l_x(\mu, \sigma^2) &= \min_{\sigma^2} \min_{\mu} l_x(\mu, \sigma^2) = \min_{\sigma^2} l_x(\hat{\mu}, \sigma^2) \\ &= \min_{\sigma^2} \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \hat{\mu})^2 + \frac{n}{2} \log \sigma^2 + n \log \sqrt{2\pi}. \end{aligned}$$

Differentiation w.r.t. σ^2 of this *profile minus-log-likelihood function* yields that there is a unique minimizer being

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

4.2.3 The Gumbel distribution

Scale-location in the Gumbel distribution

Sample space $\mathbb{R}^n$, parameter space $\mathbb{R} \times (0, \infty)$, and parameters $\theta = (\mu, \sigma)$. The probability distribution P_θ has density

$$f_{\mu,\sigma}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \frac{1}{\sigma} \exp\left(-\frac{x_i - \mu}{\sigma} - \exp\left(-\frac{x_i - \mu}{\sigma}\right)\right).$$

The minus-log-likelihood function for observing $x = (x_1, \ldots, x_n)$ is

$$l_x(\mu, \sigma) = n \log \sigma + \sum_{i=1}^{n} \frac{x_i - \mu}{\sigma} + \sum_{i=1}^{n} \exp\left(-\frac{x_i - \mu}{\sigma}\right). \tag{4.1}$$

This is the *scale-location* model for the Gumbel distribution.

Reparametrized minus-log-likelihood

$$l_x(\mu, \sigma) = n \log \sigma + \sum_{i=1}^{n} \frac{x_i - \mu}{\sigma} + \sum_{i=1}^{n} \exp\left(-\frac{x_i - \mu}{\sigma}\right).$$

Useful trick; reparametrize

$$(\eta, \rho) = \left(\frac{\mu}{\sigma}, \frac{1}{\sigma}\right) \in \mathbb{R} \times (0, \infty).$$

$$\begin{aligned} l_x(\eta, \rho) &= \sum_{i=1}^{n}(\rho x_i - \eta) + \sum_{i=1}^{n} \exp\left(-\rho x_i + \eta\right) - n \log \rho \\ &= \rho \sum_{i=1}^{n} x_i + \exp(\eta) \sum_{i=1}^{n} \exp(-\rho x_i) - n\eta - n \log \rho \\ &= \rho n \overline{x} + \exp(\eta) \sum_{i=1}^{n} \exp(-\rho x_i) - n\eta - n \log \rho \end{aligned}$$

$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

Minimization over η

The *likelihood equation*

$$\frac{\mathrm{d}l_x}{\mathrm{d}\eta}(\eta, \rho) = \exp(\eta) \sum_{i=1}^{n} \exp(-\rho x_i) - n = 0$$

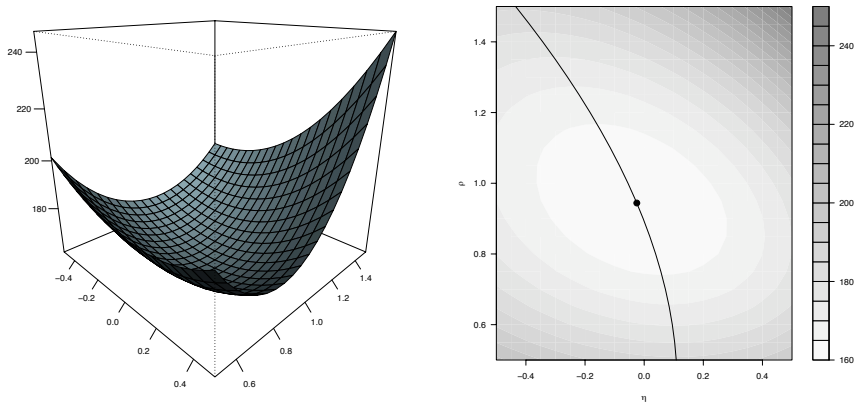


Figure 4.2: The minus-log-likelihood function (left) and a contour plot (right) for the (η, ρ) reparametrization of the scale-location model based on $n = 100$ simulations of i.i.d. Gumbel distributed variables. The MLE is $\hat{\eta} = 0.025$ and $\hat{\rho} = 0.944$. The profile minus-log-likelihood function of ρ is given by evaluating the minus-log-likelihood function along the curved line, as shown on the contour plot.

has solution

$$\hat{\eta}(\rho) = -\log\left(\frac{1}{n}\sum_{i=1}^{n}\exp(-\rho x_i)\right).$$

Second differentiation shows that this is a unique, global minimum for each fixed $\rho > 0$.

The profile minus-log-likelihood function

$$\begin{aligned} l_x(\mu,\sigma) &= n\log\sigma + \sum_{i=1}^{n}\frac{x_i-\mu}{\sigma} + \sum_{i=1}^{n}\exp\left(-\frac{x_i-\mu}{\sigma}\right) \\ \hat{\eta}(\rho) &= -\log\left(\frac{1}{n}\sum_{i=1}^{n}\exp(-\rho x_i)\right). \end{aligned}$$

$$l_x(\hat{\eta}(\rho),\rho) = \rho n\overline{x} + n + n\log\left(\frac{1}{n}\sum_{i=1}^{n}\exp(-\rho x_i)\right) - n\log\rho,$$

The (profile) likelihood equation becomes

$$eq : gumbellikeeq\frac{\sum_{i=1}^{n} x_i\exp(-\rho x_i)}{\sum_{i=1}^{n}\exp(-\rho x_i)} + \frac{1}{\rho} = \overline{x} \tag{4.2}$$

We note that for ρ approaching 0, the left hand side behaves as $\overline{x} + 1/\rho > \overline{x}$ and for ρ approaching ∞ the left hand side behaves as $\min\{x_1,\ldots,x_n\}$. If there are at least two different observations the latter is strictly smaller than $\overline{x}$ and since the left hand side is continuous in ρ there is in this case always a solution to the equation.

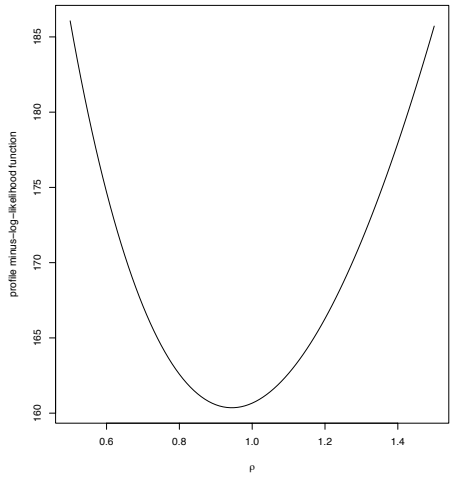


Figure 4.3: The profile minus-log-likelihood function for ρ with 100 simulated Gumbel variables.

A second differentiation of the profile minus-log-likelihood function, and some algebraic manipulations, give

$$\frac{\mathrm{d}^2 l_x}{\mathrm{d}\rho^2}(\hat{\eta}(\rho),\rho) = n\sum_{i=1}^{n}\left(x_i - \frac{\sum_{i=1}^{n} x_i \exp(-\rho x_i)}{\sum_{i=1}^{n}\exp(-\rho x_i)}\right)^2 \frac{\exp(-\rho x_i)}{\sum_{i=1}^{n}\exp(-\rho x_i)} + \frac{n}{\rho^2} > 0.$$

Since this shows that the derivative is strictly increasing there can be only one solution to the equation above.

The conclusion of our analysis is, that if $n \geq 2$ and at least two of the observations are different there is precisely one solution to the equation above, hence there is a unique global minimum for the profile minus-log-likelihood function, and consequently there is a unique global minimizer for the full minus-log-likelihood function.

From a practical point of view we need to solve (numerically) the equation (**??**) and then plug this solution into $\hat{\eta}(\rho)$. This gives the maximum likelihood estimate of (η, ρ).

Conclusions

The second derivative of $l_x(\hat{\eta}(\rho),\rho)$ is strictly positive, hence there can only be one solution to the (profile) likelihood equation in ρ – but if there are at least two different observations there will be a solution.

If there are at least two different observations there will be a unique maximum of the likelihood function, which can be found by solving the univariate (non-linear) equation in ρ numerically.

We still need to solve the non-linear profile likelihood equation, which does not have an explicit analytic expression. The Newton-Raphson algorithm is a possible choice.

The Newton-Raphson algorithm

Likelihood equation:

$$\frac{\mathrm{d}l_x}{\mathrm{d}\theta}(\theta) = 0.$$

First order Taylor expansion

$$\frac{\mathrm{d}l_x}{\mathrm{d}\theta}(\theta) \simeq \frac{\mathrm{d}l_x}{\mathrm{d}\theta}(\theta_0) + \frac{\mathrm{d}^2 l_x}{\mathrm{d}^2\theta}(\theta_0)(\theta - \theta_0).$$

Solve for $\theta_0 \in \Theta$ the *linear* equation

$$\frac{\mathrm{d}l_x}{\mathrm{d}\theta}(\theta_0) + \frac{\mathrm{d}^2 l_x}{\mathrm{d}^2\theta}(\theta_0)(\theta - \theta_0) = 0.$$

With initial guess θ_0 the iterative solutions

$$\theta_n = \theta_{n-1} - \left(\frac{\mathrm{d}^2 l_x}{\mathrm{d}^2\theta}(\theta_{n-1})\right)^{-1} \frac{\mathrm{d}l_x}{\mathrm{d}\theta}(\theta_{n-1})$$

give the Newton-Raphson algorithm.

Newton-Raphson for the profile likelihood equation

$$\rho_n = \rho_{n-1} - \frac{\mathrm{d}^2 l_x}{\mathrm{d}\rho^2}(\hat{\eta}(\rho_{n-1}), \rho_{n-1})^{-1} \frac{\mathrm{d}l_x}{\mathrm{d}\rho}(\hat{\eta}(\rho_{n-1}), \rho_{n-1})$$

for $n \geq 1$ and an initial guess ρ_0.

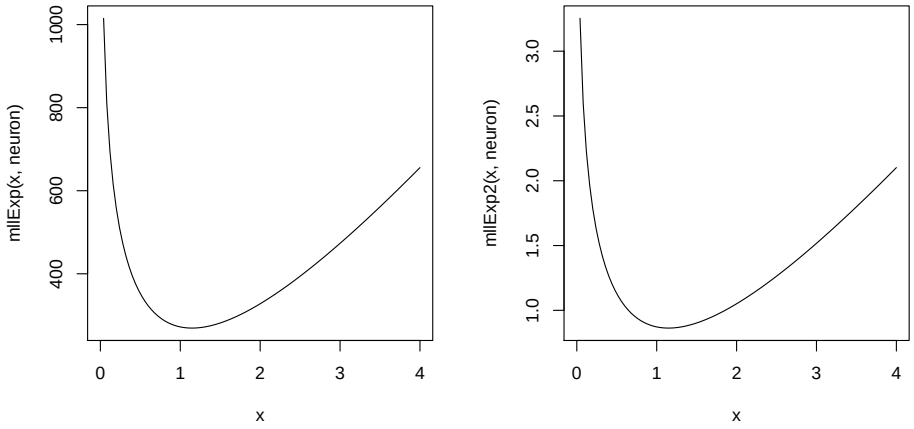
4.2.4 R interlude: Likelihood and optimization

Neuron exponential model minus-log-likelihood

```
neuron <- scan("http://www.math.ku.dk/~richard/courses/StatScience2011/neuronspikes.txt")
mllExp <- function(lambda, x)
  lambda * sum(x) - length(x) * log(lambda)
```

Alternatively, the minus-log-likelihood is sometimes normalized by the number of observations. It's the same function but a different scale on the y-axis.

```
mllExp2 <- function(lambda, x)
  lambda*mean(x) - log(lambda)
par(mfcol = c(1, 2))
curve(mllExp(x, neuron), 0, 4)
curve(mllExp2(x, neuron), 0, 4)
```



Gumbel numerical optimization

Download the simulated local alignment scores data set.

```
alignmentScores <- scan("http://www.math.ku.dk/~richard/courses/StatScience2011/alignmentScores.txt")
```

The minus-log-likelihood divided by n (`mean` replaces `sum`). This is sometimes preferable in practice. It makes the absolute size of the likelihood less dependent upon the number of observations.

```
mllGumbel <- function(mu, sigma, x)
  log(sigma) + (mean(x) - mu) / sigma + mean(exp(- (x - mu) / sigma))
```

Optimization (minimization) of the minus-log-likelihood function. Note that `optim` requires that the function to be optimized takes the parameters as a vector and that starting values are provided.

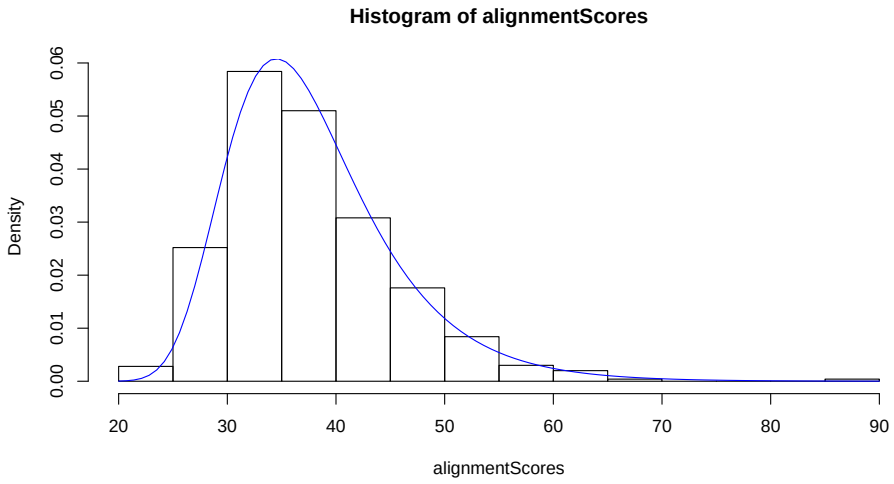
```
mllToBeOptimized <- function(par)
  mllGumbel(par[1], par[2], alignmentScores)
mle <- optim(c(40, 10), mllToBeOptimized)$par
```

The contraint $\sigma > 0$ is not strictly enforced by the optimization above. The minus-log-likelihood will, however, complain if the `optim` function attempts to evaluate the function in a non-positive σ. It is possible to enforce parameter constraints of box-type in `optim`. See the help page.

We compare the histogram to the estimated density.

```
dgumbel <- function(x, mu = 0, sigma = 1) {
  x <- (x - mu)/sigma
  exp(- x - exp(- x)) / sigma
}
```

```
hist(alignmentScores, probability = TRUE)
curve(dgumbel(x, mle[1], mle[2]), col = "blue", add = TRUE)
```



As an alternative to the generic `optim` we can implement the semianalytic solution that relies on a one-dimensional Newton-Raphson solver. This is done in the reparametrized model

$$(\eta, \rho) = (\mu/\sigma, 1/\sigma).$$

```
etaHat <- function(rho, x) - log(mean(exp(- rho * x)))
```

The Newton-Raphson update for minimizing the profile minus-log-likelihood in the ρ parameter.

```
rhoUpdate <- function(rho0, x) {
  tmp <- sum(x * exp(- rho0 * x)) / sum(exp(- rho0 * x))
  d2rho0 <- sum((x - tmp)^2 * exp(- rho0 * x)) / sum(exp(- rho0 * x)) + 1 / rho0^2
  d1rho0 <- mean(x) - tmp - 1 / rho0
  rho0 - d1rho0 / d2rho0
}
```

The function that actual computes the ρ estimate.

```
rhoHat <- function(x, rho0 = 1) {
  rho1 <- rhoUpdate(rho0, x)
  while(abs(rho1 - rho0) > 1e-10) {  ## a simple convergence criteria
    rho0 <- rho1
    rho1 <- rhoUpdate(rho0, x)
  }
  return(rho1)
}
```

Combined function for the complete computation of the MLE. The function uses the ad hoc estimate of ρ_0 from Exercise 2.5 for initialization. The raw Newton-Raphson can be quite

sensitive to the choice of initial value and a sensible initial value from a simple estimator is often preferable for numerical stability.

```
rhoetaHat <- function(x) {
  rho0 <- pi / (sd(x) * sqrt(6))
  rhoHat <- rhoHat(x, rho0 = rho0)
  etaHat <- etaHat(rhoHat, x)
 return(c(eta = etaHat, rho = rhoHat))
}
```

The resulting MLE for the data set is computed and compared with the one found using `optim`.

```
rhoetaHat(alignmentScores)

##    eta    rho
## 5.7041 0.1653

c(eta = mle[1] / mle[2], rho = 1 / mle[2])  ## From 'optim'

##    eta    rho
## 5.7048 0.1653
```

Is the MLE estimate better than that obtained from Exercise 2.5? It may be or it may not be for this particular data set. The crux of the matter is that as a general method the MLE will be among the most accurate estimators, and for this reason, and because it takes the arbitrariness out of choosing a method for estimation, it is the preferable and recommended standard estimator. There may be reasons to modify the estimator in cases where we have prior information or can make assumptions about the value of the parameters, but this is beyond the scope of this course.

4.3 Exercises

Exercise 4.1 The density for the *inverse Gaussian distribution* is

$$f_{\mu,\lambda}(x) = \left[\frac{\lambda}{2\pi x^3}\right]^{1/2} \exp\left(-\frac{\lambda(x-\mu)^2}{2\mu^2 x}\right)$$

for $x > 0$ with two parameters $\mu, \lambda > 0$. Given that we observe $x_1, \ldots, x_n$ as realizations of i.i.d. random variables with the inverse Gaussian distribution find the minus-log-likelihood function $l_x(\mu, \lambda)$ and implement a function in R that computes the minus-log-likelihood function.

Exercise 4.2 Use the `optim` function in R to find the maximum-likelihood estimate for this model of μ and λ for the neuron interspike data set from the lectures. Investigate the model fit.

Exericse 4.3 Find an analytic expression for the MLE. Compare the results for the neuron interspike data set with the numerical solution found above.

Hint: First, find the MLE of μ for fixed $\lambda > 0$, plug this into the minus-log-likelihood to get the profile minus-log-likelihood and find the MLE of λ.

Exercise 4.4 Color blindness is a so-called X-linked recessive trait. This means that the gene responsible for color blindness is located on the X chromosome and that color blindness is recessive. Females who have two X chromosomes are thus color blind if the allele resulting in color blindness (the CB-allele in the following) is present on both X chromosomes whereas males, who have only a single X chromosome, are color blind whenever the CB-allele is present on their X chromosome.

We denote the proportion of males in a population that have the CB-allele by $p \in (0, 1)$. This is the parameter we want to estimate.

We assume in the following that we have observations from m randomly selected males, $x_1, \ldots, x_m \in \{0, 1\}$, such that $x_i = 1$ if male number i is color blind (has the CB-allele). We assume that the observations are independent.

Argue that the probability of observing $x = (x_1, \ldots, x_m)$ equals

$$p^{m_b}(1-p)^{m_B}$$

where $m_b = \sum_{i=1}^{m} x_i$ is the number of males with the CB-allele and $m_B = m - m_b$ is the number of males without the CB-allele, find the likelihood function and the minus-log-likelihood function for p and show that the MLE equals

$$\hat{p} = \frac{m_b}{m}.$$

Exercise 4.5 Assume in addition that we have observations from f randomly selected females, $y_1, \ldots, y_f \in \{0, 1\}$, where $y_i = 1$ if female number i is color blind, that is, if she has two CB-alleles. We will assume that the allele distribution in the total population satisfies the *Hardy-Weinberg equilibrium*, which means that the proportion of females with 2 CB-alleles is p^2, the proportion with 1 is $2p(1-p)$ and the proportion with 0 is $(1-p)^2$. We assume that the observations are independent and also independent of the male observations above.

Argue that the probability that $y_i = 1$ equals p^2 and the probability that $y_i = 0$ equals $2p(1-p) + (1-p)^2 = (1-p)(1+p)$. Letting $y = (y_1, \ldots, y_f)$ argue that the probability of observing (x, y) equals

$$p^{m_b+2f_b}(1+p)^{f_B}(1-p)^{m_B+f_B}$$

where $f_b = \sum_{i=1}^{f} y_i$ is the number of females with two CB-alleles (that are color blind) and $f_B = f - f_b$ is the number of females with at most one CB-allele.

Exercise 4.6 In the setup above, having the combined observation (x, y) of males and females, find the likelihood function and the minus-log-likelihood function for p and show that the MLE equals

$$\hat{p} = \frac{-m_B + \sqrt{m_B^2 + 4n(m_b + 2f_b)}}{2n}$$

where $n = 2(f_b + f_B) + m_b + m_B = 2f + m$ is the total number of X chromosomes in the sample of males and females. In a study we find $f_b = 40$, $f_B = 9032$, $m_b = 725$ and $m_B = 8324$. Compute the MLE of p.

Exercise 4.7 Read the following data set into R

```
read.table("http://www.math.ku.dk/~richard/courses/StatScience2011/Ceriodaphnia.txt", header = TRUE)
```

It is a plain text file with two columns, containing the data from a small experiment where Ceriodaphnia where exposed to different doses of a compund that impairs reproduction. The dataset consists of the counts of organisms combined with the concentration in grams per liter of the compound. Plot the `organisms` and `log(organisms)` against the `concentration`.

In the following exercises we will build and analyse a model suitable for modeling the count of organisms as a function concentration.

Exercise 4.8 For this and following exercises we consider n independent random variables $X_1, \ldots, X_n$ and we assume that the distribution of X_i is the Poisson distribution with mean value parameter

$$\lambda_i = e^{\beta y_i + \alpha}$$

where $y_1, \ldots, y_n \in \mathbb{R}$ are known but $\alpha, \beta \in \mathbb{R}$ are unknown. We have the observation $x = (x_1, \ldots, x_n) \in \mathbb{N}_0^n$ and the objective is to estimate α, β.

Show that the minus-log-likelihood function is

$$l_x(\alpha, \beta) = \sum_{i=1}^{n} e^{\beta y_i + \alpha} - \beta x_i y_i - \alpha x_i + \log x_i!$$

Exercise 4.9 Fix β and show that for fixed β the minimum of the minus-log-likelihood function in α is

$$\hat{\alpha}(\beta) = \log \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} e^{\beta y_i}}.$$

Exercise 4.10 Show that the profile minus-log-likelihood function in β is

$$l_x(\hat{\alpha}(\beta), \beta) = \sum_{i=1}^{n} x_i - \beta x_i y_i - \log \left(\frac{\sum_{j=1}^{n} x_i}{\sum_{j=1}^{n} e^{\beta y_j}} \right) x_i + \log x_i!$$

and that the minimizer of the profile minus-log-likelihood solves the equation

$$\frac{\sum_{i=1}^{n} y_i e^{\beta y_i}}{\sum_{i=1}^{n} e^{\beta y_i}} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i}.$$

Exercise 4.11 Implement the Newton-Raphson algorithm for solving the equation above in β and implement a function for estimation of (α, β) for a given dataset $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$.

Exercise 4.12 Fit the Poisson model, that is, estimate the α and β parameters in the model above, to the Ceriodaphnia data.

Chapter 5

Fifth Week

5.1 Likelihood ratio tests and confidence intervals

Keywords: Confidence intervals, likelihood ratio tests, profile likelihood.

ISwR: 109-125

The general setup for this lecture is a statistical model consisting of a parametrized collection P_θ of probability distributions on some sample space and a parameter space Θ such that the parameters θ are in Θ.

5.1.1 The likelihood ratio test

X is a random variable representing our data set with distribution P_θ for some $\theta \in \Theta_0$, and we have a likelihood function $\mathcal{L}_x(\theta)$ for $\theta \in \Theta$ when observing $X = x$.

Definition 6. Observing $X = x$ then

$$Q(x) = \frac{\max_{\theta \in \Theta_0} \mathcal{L}_x(\theta)}{\max_{\theta \in \Theta} \mathcal{L}_x(\theta)}$$

is called the likelihood ratio test statistic.

Since $\Theta_0 \subseteq \Theta$, we have $Q(x) \in (0, 1]$. *Small* values of $Q(x)$ are critical.

The distribution of the likelihood ratio test statistic

Theorem 7. *If Θ is a d-dimensional parameter space and Θ_0 d_0-dimensional the distribution of*

$$-2 \log Q(X)$$

can be approximated by a χ^2-distribution with $d - d_0$ degrees of freedom. Large values of $-2 \log Q(x)$ are critical.

The theorem is not accurate and not true in general – it *is true* with a suitable set of mathematical regularity assumptions, and it is the *working horse* for practical statistical hypothesis testing.

5.1.2 Molecular evolution

To illustrate the use of the likelihood ratio test we introduce the Kimura model of molecular evolution. It is a 2-parameter model and we will compare it with the 1-parameter Jukes-Cantor model.

The Kimura model

$$\begin{aligned} P^t(x,x) &= 0.25 + 0.25\exp(-4\beta t) + 0.5\exp(-2(\alpha+\beta)t) \\ P^t(x,y) &= 0.25 + 0.25\exp(-4\beta t) - 0.5\exp(-2(\alpha+\beta)t), \quad \text{for a transition} \\ P^t(x,y) &= 0.25 - 0.25\exp(-4\beta t), \quad \text{for a transversion.} \end{aligned}$$

We regard t as fixed and $(\alpha, \beta) \in [0,\infty) \times [0,\infty)$ as the unknown parameters.

If we introduce the three quantities

$$\begin{aligned} n_1 &= n_z(A,A) + n_z(C,C) + n_z(G,G) + n_z(T,T) \\ n_2 &= n_z(A,G) + n_z(C,T) + n_z(G,A) + n_z(T,C) \\ n_3 &= n - n_1 - n_2 \end{aligned}$$

being the number of nucleotide pairs with no mutations, the number of transitions and the number of transversions, respectively, then the minus-log-likelihood function becomes

$$\begin{aligned} \tilde{l}_z(\alpha,\beta) = & -n_1 \log(0.25 + 0.25\exp(-4\beta t) + 0.5\exp(-2(\alpha+\beta)t)) \\ & -n_2 \log(0.25 + 0.25\exp(-4\beta t) - 0.5\exp(-2(\alpha+\beta)t)) \\ & -n_3 \log(0.25 - 0.25\exp(-4\beta t)). \end{aligned}$$

The Kimura model for the Hepatitis C example

	Segment		
	A	B	C
n_1	2532	1259	1009
n_2	63	20	14
n_3	15	5	6
$\hat{\alpha}$	1.9×10^{-3}	1.2×10^{-3}	1.2×10^{-3}
$\hat{\beta}$	2.2×10^{-4}	1.5×10^{-4}	2.3×10^{-4}

The estimates above are found using numerical minimization of the minus-log-likelihood function.

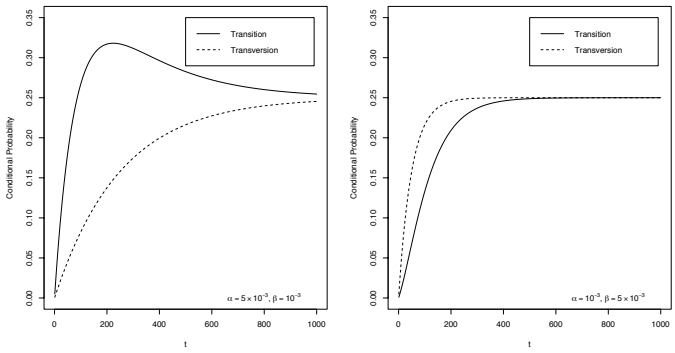


Figure 5.1: Two examples of transition and transversion probabilities for the Kimura model as a function of evolutionary distance in units of time.

Is the Jukes-Cantor model adequate?

In the Kimura model we have two parameters α and β that give the rates of *transitions* and *transversions*, respectively.

The hypothesis

$$H_0 : \alpha = \beta$$

is equivalent to the Jukes-Cantor model.

We test the hypothesis for Segment A and find

$$-2 \log Q(x) = 74.21,$$

which is to be compared to the χ^2-distribution with 1 degrees of freedom. The p-value is 7.0×10^{-18} – we *reject* the hypothesis, and the Jukes-Cantor model is *not* adequate.

Using R, the p-value is computed as follows.

```
alphaHat <- 1.9e-3
betaHat <- 2.2e-4
t <- 13

mllKim  <- -2532 * log(0.25 + 0.25 * exp(-4 * betaHat * t) +
                       0.5 * exp(-2 * (alphaHat + betaHat) * t)) -
  63 * log (0.25 + 0.25 * exp(-4 * betaHat * t) -
              0.5 * exp(- 2 * (alphaHat + betaHat) * t)) -
  15 * log(0.25 - 0.25 * exp(- 4 * betaHat * t))

alphaHat <- 7.8e-4

mllJC <- - 2532 * log(0.25 + 0.75 * exp(- 4 * alphaHat * t)) -
  78 * log(0.25 - 0.25 * exp(- 4 * alphaHat * t))

m2logQ <- 2 * (mllJC - mllKim)
m2logQ
```

```
## [1] 74.21

pchisq(m2logQ, df = 1, lower.tail = FALSE)

## [1] 7.007e-18
```

Exercise

Use the inverse Gaussian distribution as a model for the neuron interspike data and use the likelihood ratio test to test the hypothesis

$$H : \mu = 1$$

Hint: Use the formula derived from the exercise session to estimate λ under the hypothesis.

5.1.3 Tables

For an rc cross-classification table $n = (n_{ij})$ the likelihood is proportional to

$$\mathcal{L}_n(p) = \prod_{ij} p_{ij}^{n_{ij}}.$$

Here we assume a sampling scheme where the total

$$n_{..} = \sum_{i,j} n_{ij}$$

is fixed. It can be shown that the MLE is

$$\hat{p}_{ij} = \frac{n_{ij}}{n_{..}}.$$

Under the independence hypothesis

$$H : p_{ij} = p_i q_j$$

the MLE is

$$\hat{p}_i = \frac{n_{i.}}{n_{..}} \quad \hat{q}_j = \frac{n_{.j}}{n_{..}}.$$

Likelihood ratio and the χ^2-statistic

The likelihood ratio test statistic of H is

$$Q = \prod_{ij} \left(\frac{n_{i.} n_{.j}}{n_{ij} n_{..}} \right)^{n_{ij}}.$$

Thus

$$-2 \log Q = 2 \sum_{ij} n_{ij} \log \frac{n_{ij} n_{..}}{n_{i.} n_{.j}}.$$

Under the full model there are $rc-1$ free parameters and under the hypothesis H there are $r+c-2$ free parameters, thus $-2\log Q$ follows approximately a χ^2-distribution with

$$rc-1-(r+c-2)=(r-1)(c-1)$$

degrees of freedom.

Note that $-2\log Q$ can be written as

$$-2\log Q = 2\sum \text{observed}\log\left(\frac{\text{observed}}{\text{expected}}\right).$$

By a second order Taylor expansion of $2\log z$ around 1

$$2\log z \simeq 2(z-1)+(z-1)^2$$

and assuming that observed $\simeq$ expected we arrive at the approximation

$$\begin{aligned} 2\sum \text{observed}\log\left(\frac{\text{observed}}{\text{expected}}\right) &\simeq \sum \text{expected}\left(2\frac{\text{observed}-\text{expected}}{\text{expected}}+\frac{(\text{observed}-\text{expected})^2}{\text{expected}^2}\right) \\ &= \sum 2(\text{observed}-\text{expected})+X^2 \end{aligned}$$

with X^2 the χ^2-test statistic. It happens so (check) that for the tables the first sum is 0, hence the χ^2-test statistic can be seen as an approximation to $-2\log Q$.

5.1.4 Likelihood and confidence intervals

Profile likelihood

We are more interested in confidence intervals than formal hypothesis tests.

If τ is a parameter of interest and Θ_{τ_0} the subset of the full parameter space where $\tau=\tau_0$ we form the *profile likelihood*

$$\mathcal{L}_x(\tau)=\max_{\theta:\theta\in\Theta_\tau}\mathcal{L}_x(\theta)$$

and the likelihood ratio for testing the hypothesis

$$H:\tau=\tau_0 \quad (\theta\in\Theta_{\tau_0}),$$

$$Q_{\tau_0}(x)=\frac{\mathcal{L}_x(\tau_0)}{\max_\theta \mathcal{L}_x(\theta)}=\frac{\mathcal{L}_x(\tau_0)}{\mathcal{L}_x(\hat\theta)}$$

with $\hat\theta$ the MLE.

Likelihood intervals

Definition 8. A likelihood interval for the parameter of interest τ is

$$\begin{aligned} & \{\tau_0 \mid Q_{\tau_0}(x)\geq c\} \\ = \; & \{\tau_0 \mid -2\log Q_{\tau_0}(x)\leq -2\log(c)\} \\ = \; & \{\tau_0 \mid 2(\log\mathcal{L}_x(\hat\theta)-\log\mathcal{L}_x(\tau_0))\leq -2\log(c)\} \\ = \; & \{\tau_0 \mid 2(\log\mathcal{L}_x(\hat\tau)-\log\mathcal{L}_x(\tau_0))\leq -2\log(c)\} \\ = \; & \{\tau_0 \mid 2(l_x(\tau_0)-l_x(\hat\tau))\leq -2\log(c)\} \end{aligned}$$

If τ is one-dimensional and $-2\log(c) = q_\alpha$ is the $1-\alpha$ quantile for the χ^2-distribution with one degrees of freedom, the likelihood interval is *the set of* τ_0 *for which the hypothesis* $H : \tau = \tau_0$ *will not be rejected* at level α.

For any threshold c the likelihood interval is always interpretable as the values of the parameter τ that are best in agreement with the observed data. The question is how to choose c to reflect the uncertainty. The derivation shows that based on the approximating distribution of the $-2\log Q$ test statistic, the likelihood interval with threshold $-2\log(c) = q_\alpha$ will with probability $1-\alpha$ contain the true parameter value under the assumed model.

Quadratic approximations

If we have a *quadratic* approximation

$$2(l_x(\tau_0) - l_x(\hat{\tau})) \simeq \left(\frac{\tau_0 - \hat{\tau}}{\text{se}}\right)^2$$

we can solve for τ_0 and compute the likelihood interval explicitly.

$$\{\tau_0 \mid \left(\frac{\tau_0 - \hat{\tau}}{\text{se}}\right)^2 \geq q_\alpha\} \quad = \quad [\hat{\tau} - \text{se}\sqrt{q_\alpha}, \hat{\tau} - \text{se}\sqrt{q_\alpha}]$$

Note that for $\alpha = 0.05$ the 95% quantile for the χ^2-distribution with one degrees of freedom is $q_{0.05} = 3.841459 = z^2_{0.975}$ where $z_{0.975} = 1.959964$ is the 97.5% quantile for the standard normal distribution.

The likelihood interval is computable from the "raw" profile minus-log-likelihood function where we typically need to solve the equation

$$l_x(\tau_0) - l_x(\hat{\tau}) = -\log(c)$$

in terms of τ_0. There will typically be two solutions and the likelihood interval is the set of τ_0's between those two solutions. In the alternative, by a second order Taylor expansion around the MLE $\hat{\tau}$ (which is a solution to $l'_x(\hat{\tau}) = 0$)

$$\begin{aligned} 2(l_x(\tau_0) - l_x(\hat{\tau})) &\simeq 2(l_x(\hat{\tau}) + l'_x(\hat{\tau})(\tau_0 - \hat{\tau}) + \frac{1}{2}l''_x(\hat{\tau})(\tau_0 - \hat{\tau})^2 - l_x(\hat{\tau})) \\ &= l''_x(\hat{\tau})(\tau_0 - \hat{\tau})^2 \\ &= \left(\frac{\tau_0 - \hat{\tau}}{\sqrt{l''_x(\hat{\tau})^{-1}}}\right)^2 \end{aligned}$$

From this we get a quadratic approximation with

$$\text{se} = \sqrt{l''_x(\hat{\tau})^{-1}}.$$

There may be other ways to achieve a quadratic approximation, but the quantity se is generally regarded as an estimate of the *standard error*, that is, the standard deviation for the distribution of the estimator $\hat{\tau}$. The quantity $l''_x(\hat{\tau})$ is called the observed Fisher

information, and this estimate of the standard error is thus the square root of the inverse of the observed Fisher information.

If we have an estimator of the standard error for any given estimator of τ (not necessarily the MLE), we can proceed in a slightly different manner and get similar standard confidence intervals.

The standard confidence interval

We consider a real valued parameter of interest τ and corresponding estimator $\hat{\tau}$. The fundamental approximation under the hypothesis $H : \tau = \tau_0$ is

$$\hat{\tau} - \tau_0 \overset{\text{approx}}{\sim} N(0, \text{se}^2)$$

If we plug in an estimate $\hat{\text{se}}$ of se we *accept* the test at level α for

$$\tau_0 \in [\hat{\tau} - \hat{\text{se}} z_\alpha, \hat{\tau} + \hat{\text{se}} z_\alpha]$$

with z_α the $(1 - \alpha/2)$-quantile in the $N(0, 1)$ distribution.

Definition 9. The standard $(1 - \alpha)$-confidence interval for the parameter of interest τ is

$$[\hat{\tau} - \hat{\text{se}} z_\alpha, \hat{\tau} + \hat{\text{se}} z_\alpha].$$

How to estimate the standard error of an estimator?

We have a parameter of interest $\tau(\theta)$, an estimator $\hat{\tau}$ and an estimator $\hat{\theta}$ of θ.

- Somebody provided a formula $\text{se}(\theta)$ and we use the *plug-in* principle
$$\text{se}(\hat{\theta}).$$
- You used the MLE for θ and the *Fisher information* or the *observed Fisher information* to obtain estimates of the standard error.
- You simulate B new data sets using $P_{\hat{\theta}}$, reestimate τ and use the empirical standard deviation for the B estimates as an estimate of $\text{se}(\theta)$.

The last bullet point is *parametric bootstrapping* and is a simulation based implementation of bullet point 1.

5.1.5 R interlude: Confidence intervals

We consider the neuron interspike data and use the exponential distribution as a model.

We investigate the second order Taylor approximation of the minus-log-likelihood

$$l_x(\lambda) \simeq l_x(\hat{\lambda}) + \frac{1}{2} l_x''(\hat{\lambda})(\lambda - \hat{\lambda})^2$$

around the MLE (explain why there is no first order term?). The second derivative is

$$l_x''(\hat{\lambda}) = 1/\lambda^2.$$

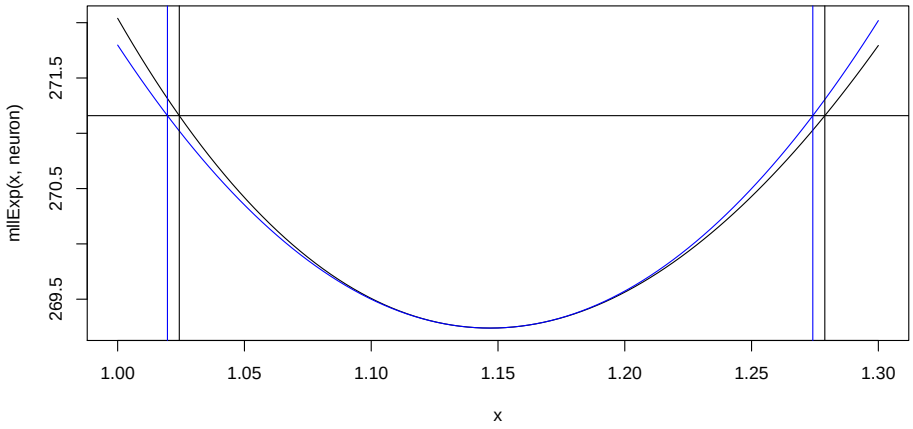
```
neuron <- scan("http://www.math.ku.dk/~richard/courses/StatScience2011/neuronspikes.txt")
mllExp <- function(lambda, x)
 lambda * sum(x) - length(x) * log(lambda)
mllExpApprox <- function(lambda, x) {
  lambdaHat <- 1 / mean(x)
  mllExp(lambdaHat, x) + length(x) * (lambda - lambdaHat)^2 / (2 * lambdaHat^2)
}
```

We want to compute likelihood intervals based on the likelihood function and on the approximation. The interval is directly computeable based on the quadratic approximation.

```
lambdaHat <- 1 / mean(neuron)
qalpha <- qchisq(0.95, 1)
likeIntApprox <- lambdaHat + c(-1, 1) * sqrt(qalpha) * lambdaHat / sqrt(length(neuron))
likeIntApprox  ## Standard confidence interval

## [1] 1.020 1.274
```

To find the likelihood interval for the actual likelihood is a little more complicated. We need to solve two equations.

```
mll0 <- function(lambda)
  mllExp(lambda, neuron) -  mllExp(lambdaHat, neuron) - qalpha / 2

lo <- uniroot(mll0, c(0.5, 1.1))$root
up <- uniroot(mll0, c(1.1, 1.8))$root
likeInt <- c(lo, up)
likeInt ## Likelihood interval

## [1] 1.024 1.279


curve(mllExp(x, neuron), 1, 1.3)
curve(mllExpApprox(x, neuron), add = TRUE, col = "blue")
abline(h = mllExp(lambdaHat, neuron) + qalpha / 2)
abline(v = likeIntApprox[1], col = "blue")
abline(v = likeIntApprox[2], col = "blue")
abline(v = likeInt[1])
abline(v = likeInt[2])
```



One can interpret the quantity `lambdaHat^2 / length(neuron)` as an approximation of the variance of the estimator of lambda. Let's estimate that variance using simulations.

```
lambdaHat <- 1 / mean(neuron)
n <- length(neuron)
lambdaDist <- replicate(1000, {x <- rexp(n, lambdaHat); 1 / mean(x)})
var(lambdaDist)
```

```
## [1] 0.004624
```

```
lambdaHat^2 / length(neuron)
```

```
## [1] 0.004216
```

R provides some support in the `stats4` package and with the `mle` function for likelihood computations, in particular, more automatic computation of maximum-likelihood estimation and (profile) likelihood intervals.

```
require(stats4)
neuronMle <- mle(function(lambda) mllExp(lambda, neuron), start = list(lambda = 1))
confint(neuronMle)
```

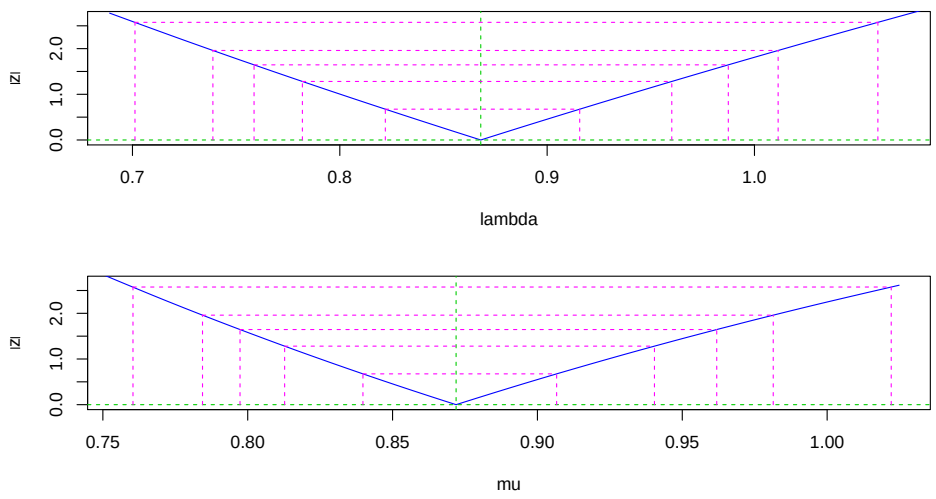
```
## Profiling...
```

```
##  2.5 % 97.5 %
##  1.024  1.279
```

It is more impressive if we work with multiple parameters. Lets take a look at the inverse Gaussian distribution from last lecture.

```
mllIg <- function(lambda, mu, x) {
  n <- length(x)
```

```
  - n * log(lambda)/2 + lambda * sum((x - mu)^2 / x) / (2 * mu^2)
}

neuronMle <- mle(function(lambda, mu) {
  ifelse(lambda > 0 & mu > 0, mllIg(lambda, mu, neuron), Inf)
},
                 start = list(lambda = 1, mu = 1))
coef(neuronMle)

## lambda     mu
## 0.8680 0.8719

confint(neuronMle)

## Profiling...

##         2.5 % 97.5 %
## lambda 0.7388 1.0114
## mu     0.7844 0.9814

par(mfcol = c(2, 1))
plot(profile(neuronMle))
```



Check `help(plot.profile, "MASS")` for information about the plot. Note the asymmetry in the intervals for the μ parameter.

5.2 Regression

ISwR Example

Relation between ventricular shortening velocity and blood glucose for type 1 diabetic pa-

tients.

Y_i the velocity – 24 observations, $i = 1, \ldots, 24$.

x_i the blood glucose $i = 1, \ldots, 24$.

Model:

$$Y_i = \alpha + \beta x_i + \sigma \epsilon_i$$

with ϵ_i i.i.d. $N(0,1)$ for $i = 1, \ldots, 24$.

Parameters $(\alpha, \beta, \sigma^2) \in \mathbb{R}^2 \times (0, \infty)$.

General regression model

A model of real valued variables $Y_1, \ldots, Y_n$

$$Y_i = g_\beta(x_i) + \sigma \epsilon_i$$

where $\epsilon_1, \ldots, \epsilon_n$ are i.i.d., $x_1, \ldots, x_n$ are the *regressors* and g_β transforms the regressors into $\mathbb{R}$.

Parameters $\beta \in \mathbb{R}^d$ and $\sigma^2 \in (0, \infty)$.

In this model the observables are assumed *independent* but not identically distributed.

The value of x_i dictates through g_β the mean value of Y_i.

5.2.1 The MLE, the normal distribution and least squares

The likelihood function

Due to independence the joint density for the distribution of $Y_1, \ldots, Y_n$ is

$$f_{\beta,\sigma}(y_1, \ldots, y_n) = \prod_{i=1}^{n} \frac{1}{\sigma} f\left(\frac{y_i - g_\beta(x_i)}{\sigma}\right). \tag{5.1}$$

with f the density for the distribution of the ϵ_i's.

The minus-log-likelihood function

$$l_y(\beta, \sigma) = n \log \sigma - \sum_{i=1}^{n} \log f\left(\frac{y_i - g_\beta(x_i)}{\sigma}\right). \tag{5.2}$$

Normal distribution

If $\epsilon_i \sim N(0,1)$ then

$$l_y(\beta,\sigma) = n\log\sigma + \frac{1}{2\sigma^2}\underbrace{\sum_{i=1}^{n}(y_i - g_\beta(x_i))^2}_{\mathrm{RSS}(\beta)} + n\log\sqrt{2\pi}.$$

The minimizer over β is the minimizer of the *residual sum of squares*

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{n}(y_i - g_\beta(x_i))^2, \tag{5.3}$$

and if there is a unique minimizer $\hat{\beta}$ then

$$\tilde{\sigma}^2 = \frac{1}{n}\mathrm{RSS}(\hat{\beta})$$

is the MLE of the variance.

Variance estimation

The MLE of σ^2,

$$\tilde{\sigma}^2 = \frac{1}{n}\mathrm{RSS}(\hat{\beta}),$$

generally underestimates σ^2 by a factor $\frac{n-d}{n}$.

Therefore the variance is always estimated by

$$\hat{\sigma}^2 = \frac{1}{n-d}\mathrm{RSS}(\hat{\beta})$$

where d is the dimension of the parameter space for β.

5.2.2 Linear regression

The linear regression model is obtained by

$$g_{\alpha,\beta}(x) = \alpha + \beta x$$

where $d = 2$. There is a unique and explicit minimizer of $\mathrm{RSS}(\alpha,\beta)$.

The estimators are computed in R using `lm` (for linear model).

The result is nicely formatted by the `summary` function of the resulting object.

For the interested it is possible to theoretically minimize RSS for linear regression. Here we derive the solution geometrically. We denote by $y \in \mathbb{R}^n$ our vector of observations, by $x \in \mathbb{R}^n$ the vector of regressors and by $\mathbf{1} \in \mathbb{R}^n$ a column vector of 1's. The quantity $\mathrm{RSS}(\beta)$

is the length of the vector $y - \beta_0 \mathbf{1} - \beta_1 x$ in $\mathbb{R}^n$. This length is minimized over β_0 and β_1 by the *orthogonal projection* of y onto the space in $\mathbb{R}^n$ spanned by x and $\mathbf{1}$.

One can find this projection if we know an *orthonormal basis*, and such a one is found by setting $a = \mathbf{1}/\sqrt{n}$ (so that a has length 1) and then replacing x by

$$b = \frac{x - (x^T a)a}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}}.$$

Both a and b have unit length and they are orthogonal as $a^T b = 0$. Note that $(xa^T)a = \bar{x}\mathbf{1}$ where $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$. We have to assume that at least two of the x_i's differ or the sum in the denominator above is 0. If all the x_i's are equal, the vector x and $\mathbf{1}$ are linearly dependent, they span a space of dimension one, and β_0 and β_1 are not both identifiable. Otherwise the vectors span a space of dimension two.

The projection of y onto the space spanned by a and b is

$$(y^T a)a + (y^T b)b = \underbrace{\left(\bar{y} - \bar{x}\frac{y^T x - n\bar{x}\bar{y}}{\sum_{i=1}^n (x_i - \bar{x})^2}\right)}_{\hat{\beta}_0}\mathbf{1} + \underbrace{\frac{y^T x - n\bar{x}\bar{y}}{\sum_{i=1}^n (x_i - \bar{x})^2}}_{\hat{\beta}_1} x.$$

Thus the theoretical solution to the minimization problem can be written

$$\hat{\beta}_1 = \frac{y^T x - n\bar{x}\bar{y}}{\sum_{i=1}^n (x_i - \bar{x})^2} \qquad \hat{\beta}_0 = \bar{y} - \bar{x}\hat{\beta}_1.$$

Residuals

The *fitted values* are defined as

$$\hat{y}_i = \hat{\alpha} + \hat{\beta} x_i$$

and the *residuals* as

$$e_i = y_i - \hat{y}_i = y_i - \hat{\alpha} - \hat{\beta} x_i.$$

The residual e_i is an approximation of the error variable $\sigma\epsilon_i$.

Leverage and standardized residuals

The error variables $\sigma\epsilon_i$ for $i = 1, \ldots, n$ are i.i.d. and have variance σ^2.

The residuals e_i for $i = 1, \ldots, n$ are *mildly* dependent and the variance of e_i is $\sigma^2(1 - h_{ii})^2$ where h_{ii} is known as the *leverage* of the i'th observation.

The *standardized* residuals are

$$r_i = \frac{e_i}{\hat{\sigma}\sqrt{1 - h_{ii}}}.$$

The formula for the leverage for linear regression is

$$h_{ii} = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{j=1}^n (x_j - \bar{x})^2}$$

and $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$.

Standard confidence intervals

Estimates of standard errors of α and β are computable and reported by R.

Standard 95% confidence intervals are

$$\hat{\alpha} \pm t_{n-2}\hat{\text{se}}(\alpha) \quad \text{and} \quad \hat{\beta} \pm t_{n-2}\hat{\text{se}}(\beta)$$

with t_{n-2} the 0.975-quantile for the t-distribution with $n-2$ degrees of freedom where n is the number of observations.

Model diagnostic

The model assumptions are investigated via the residuals or standardized residuals.

- Is the model, $\alpha + \beta x$, of the mean value adequate? Plot the residuals e_i against x_i or against the fitted values $\hat{y}_i$.
- Is the assumption of a constant variance σ^2 reasonable? Plot the standardized residuals r_i – or as done in R, $\sqrt{|r_i|}$ – against the fitted values $\hat{y}_i$.
- Is the error distribution normal? Do a QQ-plot of the residuals against the normal distribution.
- Are the outliers and/or influential observations. Look in the residual plots. Plot the standardized residuals against the leverage.

Pearson correlation

The empirical (Pearson) correlation of two variables is

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}.$$

If $\hat{\beta}$ is the regression coefficient (regressing y on x) then

$$r = \hat{\beta}\sqrt{\frac{\sum_{i=1}^n (y_i - \bar{y})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}}.$$

The last factor can be interpreted as a ratio of the empirical standard deviation of the y and the empirical standard deviation of the x variable.

Linear regression example

The Berkeley University internet traffic data set consists of 4161 observations of size and durations of internet downloads.

We suggest the following linear regression model

$$Y_i = \alpha + \beta x_i + \sigma\epsilon_i$$

where $Y_i = \log(\text{duration}_i)$ is the log-duration and $x_i = \log(\text{size}_i)$ is the log-size.

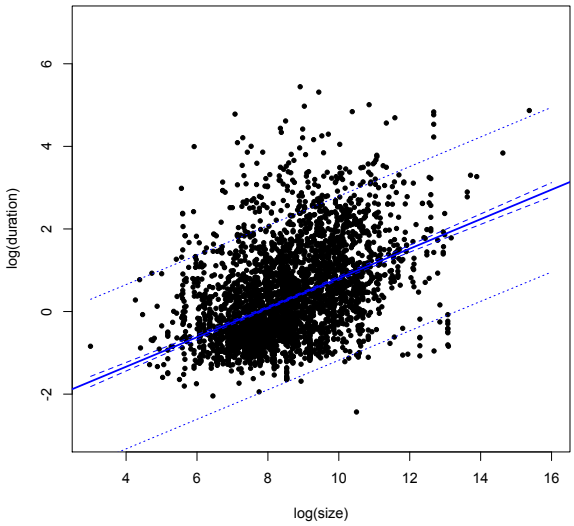


Figure 5.2: Internet download data including the regression line, a 95% confidence interval and and 95% prediction interval.

That is

$$\text{duration}_i = \text{size}_i^{\beta} e^{\alpha} e^{\sigma \epsilon_i}.$$

BU internet traffic data

The fitted model:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-2.7661	0.0965	-28.68	0.0000
log(size)	0.3573	0.0113	31.74	0.0000

The figure includes a 95% *confidence band* for the regression line as well as a 95% *prediction band*. The confidence band gives how accurate the mean of log-duration is estimated for a given value of log-size. The prediction band gives the interval into which new observations will fall with probability 95% taking into account the uncertainty in the estimation of the regression line.

We should note a skewness in the actual distribution of the data that is not taken into account.

The main conclusions from the diagnostic plots are that

- the linear model is a reasonable fit
- there is a tendency of an increasing variance with the fitted value, and thus with the size

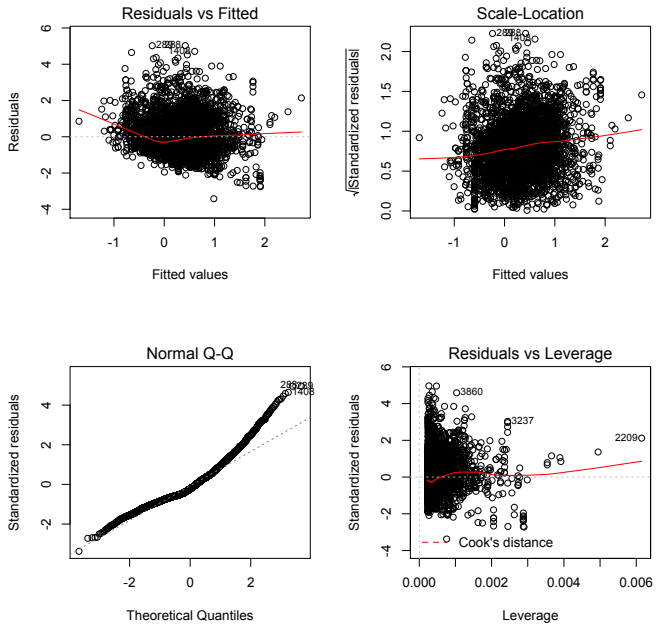


Figure 5.3: Plots to support model control for the Berkeley internet data.

- there is a clear deviation from the normal distribution for the residuals, and the deviation is in the direction of a right-skewed distribution
- there are no clear outliers or influential observations

Some conclusions for the BU data set

- The linear model of the mean of log-duration in terms of log-size is OK, and we can rely on the fitted model as reasonably accurate w.r.t. the mean.
- However, the model based on a homogeneous variance and normality underestimates the fraction of large durations (the residual distribution is not normal) across the whole range of size, and it also seems to underestimate the variance for large values of size.

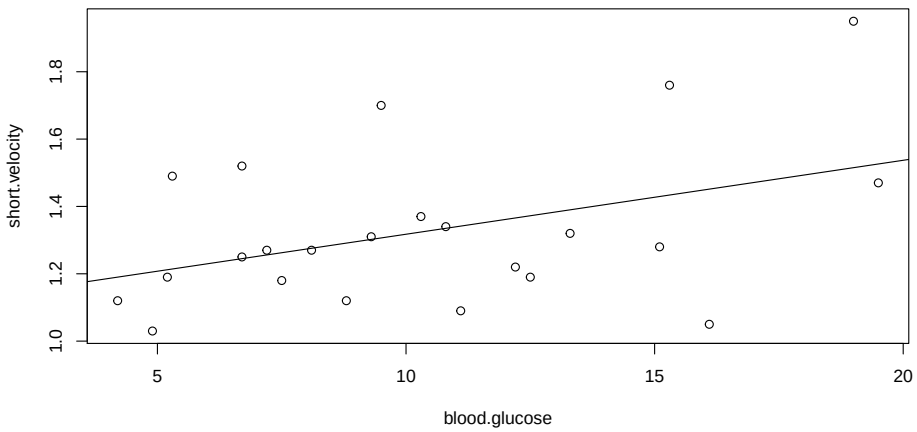
5.2.3 R interlude: Linear regression

We first consider the data set from `ISwR` (PD's book).

```
require(ISwR)
data(thuesen)
```

A linear model is fitted using `lm`.

```
thuesenLm <- lm(short.velocity ~ blood.glucose, data = thuesen)
summary(thuesenLm)

##
## Call:
## lm(formula = short.velocity ~ blood.glucose, data = thuesen)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -0.401 -0.148 -0.022  0.030  0.435
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.0978     0.1175    9.34  6.3e-09 ***
## blood.glucose   0.0220     0.0105    2.10    0.048 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.217 on 21 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared: 0.174,Adjusted R-squared: 0.134
## F-statistic: 4.41 on 1 and 21 DF,  p-value: 0.0479

plot(short.velocity ~ blood.glucose, data = thuesen)
abline(thuesenLm)
```



The information in the summary above is quite well explained in ISwR, Section 6.1. What we should note, in particular, is that there is an estimated standard error reported for each estimated parameter (and a formal test of whether the parameter is significantly different from 0). You can extract these standard errors as follows:

```
summary(thuesenLm)$coefficients[, 2]
```

```
##   (Intercept) blood.glucose
##       0.11748       0.01045
```

and it is possible to compute standard 95% confidence intervals using the t-distribution with 21 degrees of freedom to determine the relevant quantile.

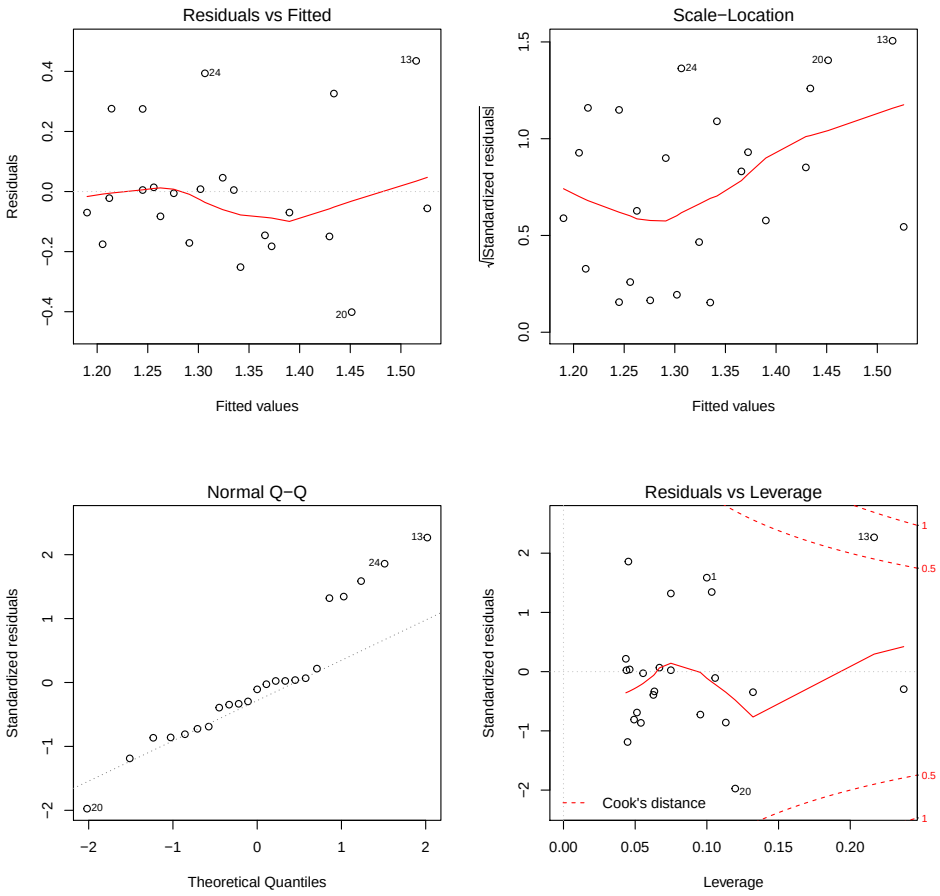
```
sumThLm <- summary(thuesenLm)
sumThLm$coefficients[2, 1] +
  c(-1, 1) * qt(0.975, 21) * sumThLm$coefficients[2, 2] ## for the regression coef.

## [1] 0.0002231 0.0437019
```

Of course, there is a convenience wrapper function for doing this.

```
confint(thuesenLm)

##                   2.5 % 97.5 %
## (Intercept)   0.8534994 1.3421
## blood.glucose 0.0002231 0.0437
```

Standard graphical model diagnostics are available using the `plot` function

```
par(mfcol = c(2,2))
plot(thuesenLm)
```



These are, residuals versus fitted values (is the linear model reasonable?). The square root of the standardized residuals (is a homogeneous variance assumption reasonable?). A QQ-plot against the normal distribution (is it reasonable to assume that the errors are normally distributed?). And finally a slightly more technical plot on whether there are any very influential observations (possible outliers).

```
cor(thuesen[, 1], thuesen[, 2], use = "complete.obs")^2

## [1] 0.1737
```

This is the squared correlation, which is reported by the summary function as the R^2 value. Note that we have to be explicit above on how to handle the missing observation.

Then we turn to the internet traffic data.

```
BUtraffic <- read.table("http://www.math.ku.dk/~richard/courses/StatScience2011/BUtraffic.txt")
## BUtraffic <- BUtraffic[sample(dim(BUtraffic)[1], 40), ]
```

It can be interesting to see the effect of the size of the data set on the uncertainty in the

obtained results by subsampling a smaller data set. Try uncommenting the code above to see the effect.

```
cor(BUtraffic[, c(2,3)])

##          duration   size
## duration   1.0000 0.2669
## size       0.2669 1.0000

BUtrafficLm <- lm(log(duration) ~ log(size), data = BUtraffic)
summary(BUtrafficLm)

##
## Call:
## lm(formula = log(duration) ~ log(size), data = BUtraffic)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.418 -0.673 -0.230  0.517  5.027
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.7661     0.0965   -28.7   <2e-16 ***
## log(size)     0.3573     0.0113    31.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.01 on 4159 degrees of freedom
## Multiple R-squared: 0.195,Adjusted R-squared: 0.195
## F-statistic: 1.01e+03 on 1 and 4159 DF,  p-value: <2e-16
```

Computations of confidence intervals for the estimated parameters and a confidence band.

```
confint(BUtrafficLm)

##               2.5 %  97.5 %
## (Intercept) -2.9552 -2.5770
## log(size)    0.3353  0.3794

newSizes <- data.frame(size = exp(3:16))
BUpred <- predict(BUtrafficLm, newdata = newSizes,
                  interval = "confidence")
```

The regression line is very well determined, see Figure 5.2. When it comes to predictions there is, however, still a considerable uncertainty.

```
BUpred2 <- predict(BUtrafficLm, newdata = newSizes,
                   interval = "prediction")
plot(log(duration) ~ log(size), data = BUtraffic, pch = 20,
     xlim = c(3, 16), ylim = c(-3, 7))
```

```
abline(BUtrafficLm, col = "blue", lwd = 2)
lines(log(newSizes$size), BUpred[, "lwr"], col = "blue", lty = 2)
lines(log(newSizes$size), BUpred[, "upr"], col = "blue", lty = 2)
lines(log(newSizes$size), BUpred2[, "lwr"], col = "blue", lty = 3)
lines(log(newSizes$size), BUpred2[, "upr"], col = "blue", lty = 3)
```

5.3 Exercises

In the following exercises we consider the Ceriodaphnia data from Exercises 4.7-4.12.

Exercise 5.1 Within the setup of Exercise 4.8 we will investigate the hypothesis

$$H : \beta = 0.$$

Use the result in Exercise 4.10 to compute an expression for $-2 \log Q(x)$.

Exercise 5.2 Test H for the Ceriodaphnia data.

Exercise 5.3 Simulate the distribution of $-2 \log Q(x)$ under H and compare this resulting distribution with the theoretical approximation.

Exercise 5.4 Fit a linear regression model of `log(organisms)` regressed on `concentration`. Investigate the model fit and compare the model with the one from Exercise 4.12.

Chapter 6

Sixth Week

6.1 Multiple linear regression

Keywords: design matrices, interactions, linear models, polynomial regression.

ISwR: 185-190, 195-206

We consider in some detail the cystic fibrosis example from ISwR, Chapter 11, and introduce various concepts in multiple linear regression. Most notably how the design matrix (or model matrix in R jargon) reflects a given choice of parametrization.

6.1.1 Cystic fibrosis

Cystic fibrosis data

	pemax	weight	height	sex
1	95	13.10	109	0
2	85	12.90	112	1
3	100	14.10	124	0
4	85	16.20	125	1
.	.	.	.	.
.	.	.	.	.
24	95	51.10	175	0
25	195	71.50	179	0

and we will consider the following regression model

$$Y_i = \alpha_{\text{sex}} + \beta_{\text{sex}} \times \text{weight}_i + \epsilon_i$$

where the response variable Y_i is `pemax`.

The design matrix

The parametrization $\alpha_{\text{male}}, \alpha_{\text{female}}, \beta_{\text{male}}, \beta_{\text{female}}$ corresponds to the *design matrix*

$$\begin{bmatrix} 1 & 0 & 13.1 & 0.0 \\ 0 & 1 & 0.0 & 12.9 \\ 1 & 0 & 14.1 & 0 \\ 0 & 1 & 0.0 & 16.2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 51.1 & 0.0 \\ 1 & 0 & 71.5 & 0.0 \end{bmatrix}$$

$$\begin{aligned} Y_1 &= \alpha_{\text{male}} + \beta_{\text{male}} \times 13.1 + \epsilon_1 \\ Y_2 &= \alpha_{\text{female}} + \beta_{\text{female}} \times 12.9 + \epsilon_2 \\ &\vdots \quad \vdots \quad \vdots \\ Y_{25} &= \alpha_{\text{male}} + \beta_{\text{male}} \times 71.5 + \epsilon_{25} \end{aligned}$$

Reparametrization

The typical model specification is in the parametrization

$$\beta_0 = \alpha_{\text{male}}, \quad \beta_1 = \alpha_{\text{female}} - \alpha_{\text{male}}, \quad \beta_2 = \beta_{\text{male}}, \quad \beta_3 = \beta_{\text{female}} - \beta_{\text{male}}.$$

With design matrix

$$\begin{bmatrix} 1 & 0 & 13.1 & 0.0 \\ 1 & 1 & 12.9 & 12.9 \\ 1 & 0 & 14.1 & 0.0 \\ 1 & 1 & 16.2 & 16.2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 51.1 & 0.0 \\ 1 & 0 & 71.5 & 0.0 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{e}_{\text{female}} & \mathbf{weight} & \mathbf{e}_{\text{female}} : \mathbf{weight} \end{bmatrix}$$

$$\begin{aligned} Y_2 &= \beta_0 + \beta_1 + \beta_2 \times 12.9 + \beta_3 \times 12.9 + \epsilon_2 \\ &= \alpha_{\text{female}} + \beta_{\text{female}} \times 12.9 + \epsilon_2 \end{aligned}$$

We have used the notation

$$\mathbf{1} = [1, \ldots, 1]^T, \quad \mathbf{e}_{\text{female}} = [0, \ldots, 0, 1, \ldots, 1]^T$$

and **weight** is the vector of weights. The notation $\mathbf{e}_{\text{female}}$: **weight** is used for the 'interaction term' in the R-terminology. It is effectively the coordinatewise multiplication of the vectors $\mathbf{e}_{\text{female}}$ and **weight**.

Hypotheses

In the second parametrization *differences* are expressible as parameters being non-zero;

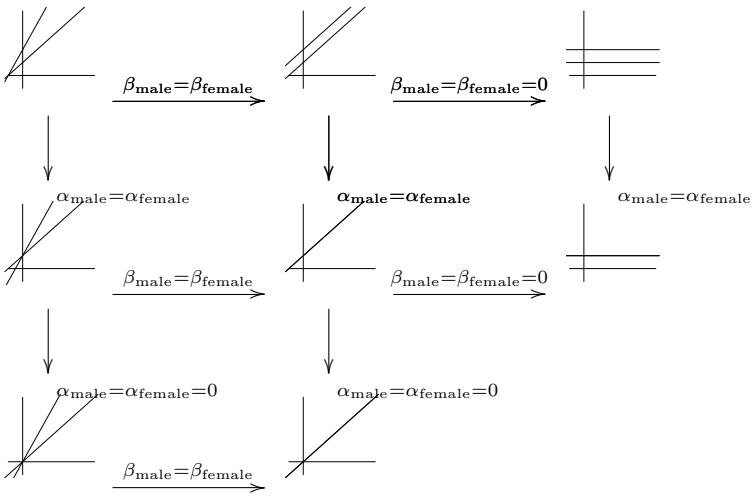
$$H : \beta_1 = 0 \quad \text{is} \quad H : \alpha_{\text{male}} = \alpha_{\text{female}}$$

or

$$H : \beta_3 = 0 \quad \text{is} \quad H : \beta_{\text{male}} = \beta_{\text{female}}$$

From the full model with 4 free parameters there are 9 submodels and many different nested model comparisons.

Model diagram



ANOVA, nested models and F-tests

Under the assumption that the errors ϵ_i are i.i.d. with the $N(0, \sigma^2)$-distribution it is possible to compute the likelihood ratio test statistic for the comparison of two nested models, count degrees of freedom and use the *approximate* χ^2-distribution of $-2 \log Q$.

It is, however, possible to obtain exact results, which are prefered and reported in terms of the *F-test*, which is equivalent to the likelihood ratio test.

The F-tests are effectively comparisons of variance estimates in nested models – known as *ANOVA*= ANalysis Of VAriance.

Two sequences of nested models

From the model diagram we can for instance pursue two directions of nested model testing.

$$\begin{aligned}\beta_3 &= 0 \quad (\beta_{\text{male}} = \beta_{\text{female}})\\ \beta_2 &= 0 \quad (\alpha_{\text{male}} = \alpha_{\text{female}})\\ \beta_1 &= 0 \quad (\beta_{\text{male}} = \beta_{\text{female}} = 0)\end{aligned}$$

Or

$$\begin{aligned}\beta_3 &= 0 \quad (\beta_{\text{male}} = \beta_{\text{female}})\\ \beta_1 &= 0 \quad (\beta_{\text{male}} = \beta_{\text{female}} = 0)\\ \beta_2 &= 0 \quad (\alpha_{\text{male}} = \alpha_{\text{female}})\end{aligned}$$

6.1.2 Polynomial regression

If we focus on regressing `pemax` on `height` we might be satisfied with the linear regression model, but we might also speculate if we can get a better model fit if we allow for a more flexible (non-linear) model of the relation between the two variables.

In the subsequent section we consider non-linear regression models, but an alternative is to write the relation as a polynomial in `height`, e.g.

$$\texttt{pemax} = \beta_0 + \beta_1 \times \texttt{height} + \beta_2 \times \texttt{height}^2 + \epsilon$$

This would be a multiple linear regression model in the variables `height` and $\texttt{height}^2$.

In general, with $f_1, \ldots, f_d$ known functions we can formulate the multiple linear regression model

$$y_i = \beta_0 + \beta_1 f_1(x_i) + \ldots + \beta_{d+1} f_d(x_i) + \epsilon_i.$$

This model corresponds to a design matrix with the i'th row being

$$[1 \; f_1(x_i) \; \ldots \; f_d(x_i)].$$

With $f_k(x) = x^k$ we get polynomial regression.

The fact that we can transform the x-variables with known functions, and in this way computed derived variables for the regression, makes multiple linear regression an extremely powerful and flexible tool. The flexibility does not come without any costs, though. There is a risk of overfitting the model to data if you search through a lot of possible transformations, which will make the model fit the observed data very well but it will fit future data poorly.

6.1.3 R interlude: More linear regression

We use the small example data set on cystic fibrosis from ISwR, see Chapter 11.

```
require(ISwR)
data(cystfibr)
?cystfibr
```

A so-called scatter plot matrix is produced when "plotting" a data frame.

```
plot(cystfibr)
```

As described in ISwR, page 186, this is actually a call of the `pairs` function and for help on the setting of plotting parameters see `help(pairs)`.

One should note the clear correlation between the age, height and weight variables. This is to be expected (the data is on children and young adults).

```
print(cor(cystfibr), digits = 2)
```

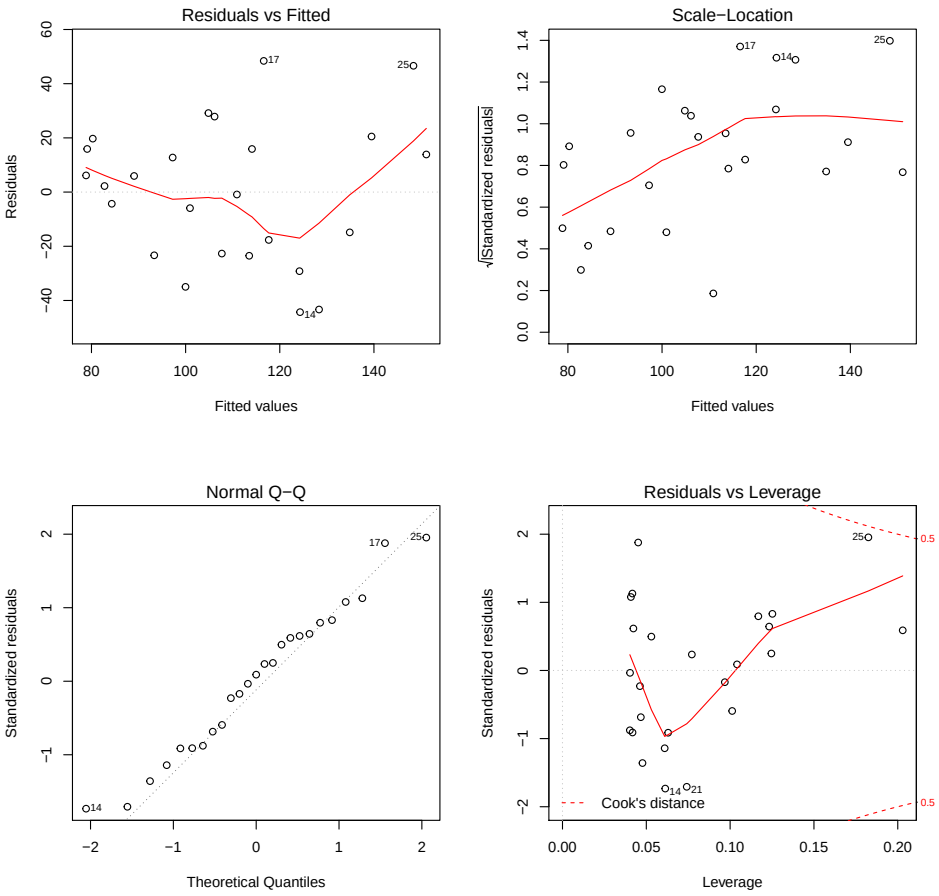
```
##          age    sex height weight   bmp  fev1    rv   frc    tlc pemax
## age     1.00 -0.167   0.93   0.91  0.38  0.29 -0.55 -0.64 -0.469  0.61
## sex    -0.17  1.000  -0.17  -0.19 -0.14 -0.53  0.27  0.18  0.024 -0.29
## height  0.93 -0.168   1.00   0.92  0.44  0.32 -0.57 -0.62 -0.457  0.60
## weight  0.91 -0.190   0.92   1.00  0.67  0.45 -0.62 -0.62 -0.418  0.64
## bmp     0.38 -0.138   0.44   0.67  1.00  0.55 -0.58 -0.43 -0.365  0.23
## fev1    0.29 -0.528   0.32   0.45  0.55  1.00 -0.67 -0.67 -0.443  0.45
## rv     -0.55  0.271  -0.57  -0.62 -0.58 -0.67  1.00  0.91  0.589 -0.32
```

```
## frc    -0.64  0.184  -0.62  -0.62 -0.43 -0.67  0.91  1.00  0.704 -0.42
## tlc    -0.47  0.024  -0.46  -0.42 -0.36 -0.44  0.59  0.70  1.000 -0.18
## pemax   0.61 -0.289   0.60   0.64  0.23  0.45 -0.32 -0.42 -0.182  1.00
```

The response variable is `pemax`, the maximum expiratory pressure (a measure of strength of respiratory muscles when exhaling). The variables `age`, `height` and `weight` have the largest (marginal) correlation with the response variable, but they are also highly correlated themselves.

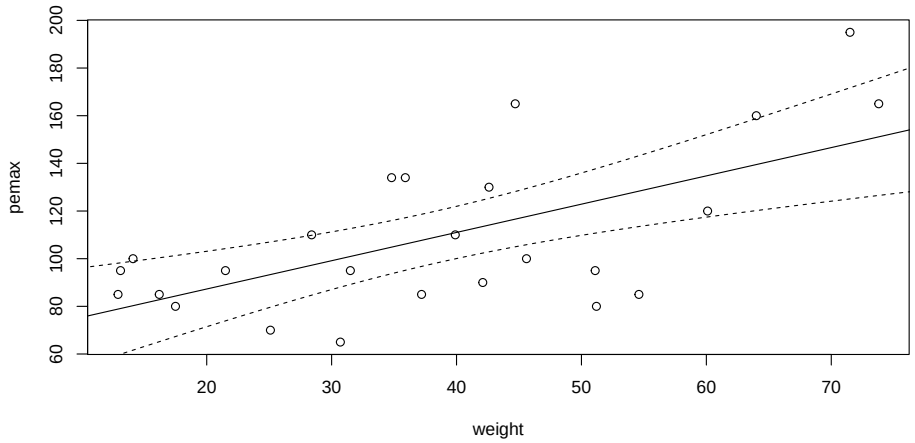
The first model we will investigate is a linear model of `pemax` regressed against `weight`.

```
cystLm1 <- lm(pemax ~ weight, data = cystfibr)
summary(cystLm1)


##
## Call:
## lm(formula = pemax ~ weight, data = cystfibr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -44.31 -22.69   2.23  15.91  48.41
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   63.546     12.702    5.00  4.6e-05 ***
## weight         1.187      0.301    3.94  0.00065 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.4 on 23 degrees of freedom
## Multiple R-squared: 0.404,Adjusted R-squared: 0.378
## F-statistic: 15.6 on 1 and 23 DF,  p-value: 0.000646



par(mfcol = c(2, 2))
plot(cystLm1)  ## Model control
```



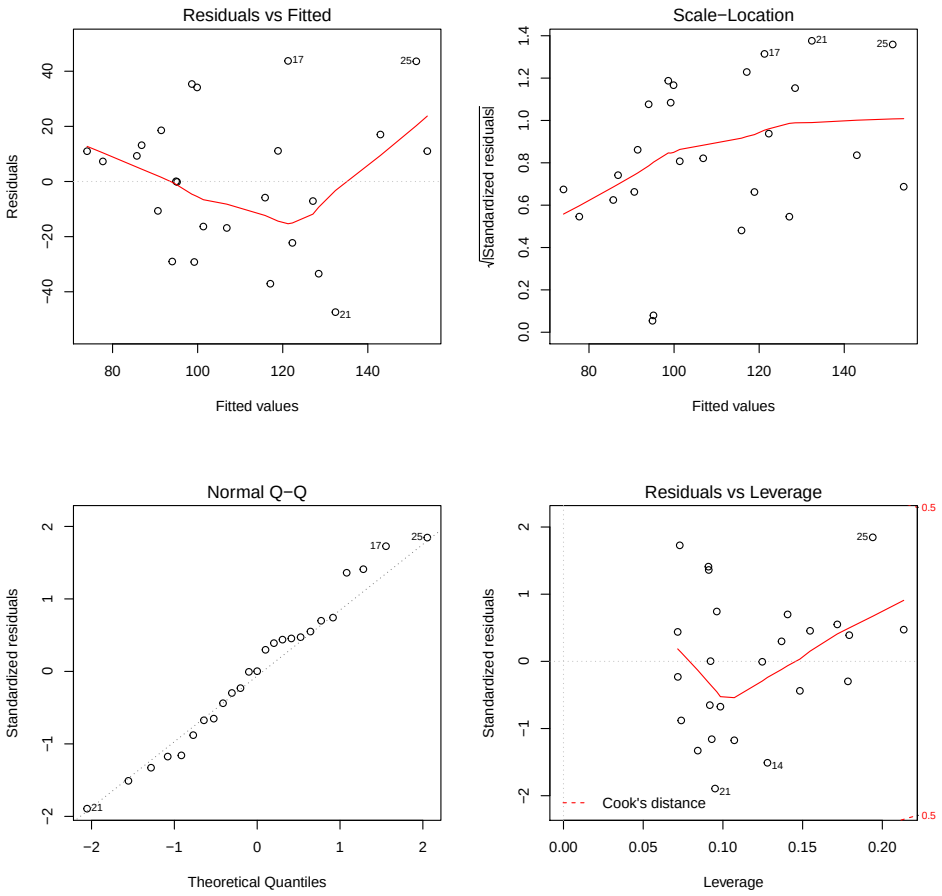
It is a small data set, but there is a tendency of an increased variance for large fitted values, thus variance heterogeneity.

```
newdata <- data.frame(weight = seq(10, 80, 1))
cystPred1 <- predict(cystLm1, newdata, interval = "conf")
plot(pemax ~ weight, data = cystfibr)
matlines(newdata$weight, cystPred1, lty = c(1, 2, 2), col = "black")
```



We are interested in investigating if there is a gender specific relation between the maximum expiratory pressure and the weight. First, we introduce a gender specific intercept only.

```
cystLm2 <- lm(pemax ~ weight + sex, data = cystfibr)
summary(cystLm2)


##
## Call:
## lm(formula = pemax ~ weight + sex, data = cystfibr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -47.39 -16.85   0.07  13.17  43.75
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   70.972     14.464    4.91  6.6e-05 ***
## weight         1.125      0.306    3.68   0.0013 **
## sex          -11.478     10.796   -1.06   0.2993
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.3 on 22 degrees of freedom
## Multiple R-squared: 0.433,Adjusted R-squared: 0.381
## F-statistic: 8.39 on 2 and 22 DF,  p-value: 0.00196



par(mfcol = c(2, 2))
plot(cystLm2)  ## Model control
```



It does not really improve on the model fit, and the conclusion from the summary table above is that the effect of including the `sex` variable is not statistically significant. That is, the coefficient estimated for the difference between the two genders (coded as 1 = female, meaning that the estimated parameter of −11.48 is the estimated difference of the intercept for females compared to males) is not significantly different from 0.

An interaction model is one where the slope depends on the gender as well, and can be fitted as follows.

```
cystLm3 <- lm(pemax ~ sex * weight, data = cystfibr)
summary(cystLm3)


##
## Call:
## lm(formula = pemax ~ sex * weight, data = cystfibr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -50.5  -14.6   -2.1   14.2   43.0
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   61.360     15.934    3.85  0.00093 ***
## sex           22.091     27.292    0.81  0.42736
## weight         1.357      0.347    3.91  0.00081 ***
## sex:weight    -0.924      0.692   -1.33  0.19619
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.8 on 21 degrees of freedom
## Multiple R-squared: 0.477,Adjusted R-squared: 0.402
## F-statistic: 6.39 on 3 and 21 DF,  p-value: 0.00303
```

The effect of `sex` is not significant. The conclusion is that for this very small data set, we cannot detect a difference in slope between the two genders either.

The design matrix corresponding to the interaction model above can be obtained as:

```
model.matrix(cystLm3)
```

```
##    (Intercept) sex weight sex:weight
## 1            1   0   13.1        0.0
## 2            1   1   12.9       12.9
## 3            1   0   14.1        0.0
## 4            1   1   16.2       16.2
## 5            1   0   21.5        0.0
## 6            1   0   17.5        0.0
## 7            1   1   30.7       30.7
## 8            1   1   28.4       28.4
## 9            1   0   25.1        0.0
## 10           1   1   31.5       31.5
## 11           1   0   39.9        0.0
## 12           1   1   42.1       42.1
## 13           1   0   45.6        0.0
## 14           1   1   51.2       51.2
## 15           1   1   35.9       35.9
## 16           1   1   34.8       34.8
## 17           1   0   44.7        0.0
## 18           1   1   60.1       60.1
## 19           1   0   42.6        0.0
## 20           1   1   37.2       37.2
## 21           1   0   54.6        0.0
## 22           1   0   64.0        0.0
## 23           1   0   73.8        0.0
## 24           1   0   51.1        0.0
## 25           1   0   71.5        0.0
## attr(,"assign")
## [1] 0 1 2 3
```

If we compute confidence intervals, say, for the estimated parameters, we will get the confidence intervals for the parametrization corresponding to the design matrix.

```
confint(cystLm3)
```

```
##                2.5 %  97.5 %
## (Intercept)  28.2247 94.4959
## sex         -34.6668 78.8479
## weight        0.6354  2.0791
## sex:weight   -2.3635  0.5154
```

This is not always desirable. One possibility is to force a particular parametrization. In the next analysis we explicitly cast the `sex` variable as a factor to get the desired parametrization.

```
cystLm3mod <- lm(pemax ~ sex + sex:weight - 1,
                 data = transform(cystfibr, sex = as.factor(sex)))
summary(cystLm3mod)
```

```
##
## Call:
## lm(formula = pemax ~ sex + sex:weight - 1, data = transform(cystfibr,
##     sex = as.factor(sex)))
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -50.5  -14.6   -2.1   14.2   43.0
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## sex0          61.360     15.934    3.85  0.00093 ***
## sex1          83.451     22.158    3.77  0.00113 **
## sex0:weight    1.357      0.347    3.91  0.00081 ***
## sex1:weight    0.433      0.599    0.72  0.47745
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.8 on 21 degrees of freedom
## Multiple R-squared: 0.957,Adjusted R-squared: 0.949
## F-statistic:  116 on 4 and 21 DF,  p-value: 5.25e-14
```

```
model.matrix(cystLm3mod)
```

```
##    sex0 sex1 sex0:weight sex1:weight
## 1     1    0        13.1         0.0
## 2     0    1         0.0        12.9
## 3     1    0        14.1         0.0
## 4     0    1         0.0        16.2
## 5     1    0        21.5         0.0
## 6     1    0        17.5         0.0
## 7     0    1         0.0        30.7
## 8     0    1         0.0        28.4
## 9     1    0        25.1         0.0
## 10    0    1         0.0        31.5
```

```
## 11    1    0        39.9         0.0
## 12    0    1         0.0        42.1
## 13    1    0        45.6         0.0
## 14    0    1         0.0        51.2
## 15    0    1         0.0        35.9
## 16    0    1         0.0        34.8
## 17    1    0        44.7         0.0
## 18    0    1         0.0        60.1
## 19    1    0        42.6         0.0
## 20    0    1         0.0        37.2
## 21    1    0        54.6         0.0
## 22    1    0        64.0         0.0
## 23    1    0        73.8         0.0
## 24    1    0        51.1         0.0
## 25    1    0        71.5         0.0
## attr(,"assign")
## [1] 1 1 2 2
## attr(,"contrasts")
## attr(,"contrasts")$sex
## [1] "contr.treatment"
```

```
confint(cystLm3mod)
```

```
##               2.5 %  97.5 %
## sex0        28.2247  94.496
## sex1        37.3701 129.531
## sex0:weight  0.6354   2.079
## sex1:weight -0.8122   1.679
```

It is possible to compare the interaction model with the model without any effect of `sex`. The test belongs to the family of tests of linear model restrictions within the framework of linear regression models with normally distributed errors. The analysis of such models and hypotheses is known as *analysis of variance* – or ANOVA – because it boils down to comparisons of residual variance estimates.

```
anova(cystLm1, cystLm3)
```

```
## Analysis of Variance Table
##
## Model 1: pemax ~ weight
## Model 2: pemax ~ sex * weight
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1     23 16005
## 2     21 14033  2      1973 1.48   0.25
```

This is technically a test of a hypothesis about two parameters being 0, that is a drop from a model with 4 parameters to a model with 2 parameters. The `anova` function computes a test-statistic called the F-test, which is equivalent to the likelihood ratio test statistic under the normal error model assumptions. However, in this particular case, there is no need for approximative results based on χ^2-distributions. Instead, the F-test has exactly

an F-distribution under the model assumptions. Above, the p-value is computed to 0.2513, thus the conclusion is still that we cannot detect any effect of including `sex` in the model.

The linear models are, in fact, quite versatile when combined with transformations of the variables. It can be something of an art to find the correct transformations, but they may sometimes be given by subject matter reasons (equations that describe relations between variables may be known in certain cases).

However, sometimes we need a non-linear relation, which is not obviously known, but perhaps approximated reasonably by a low-order polynomial.

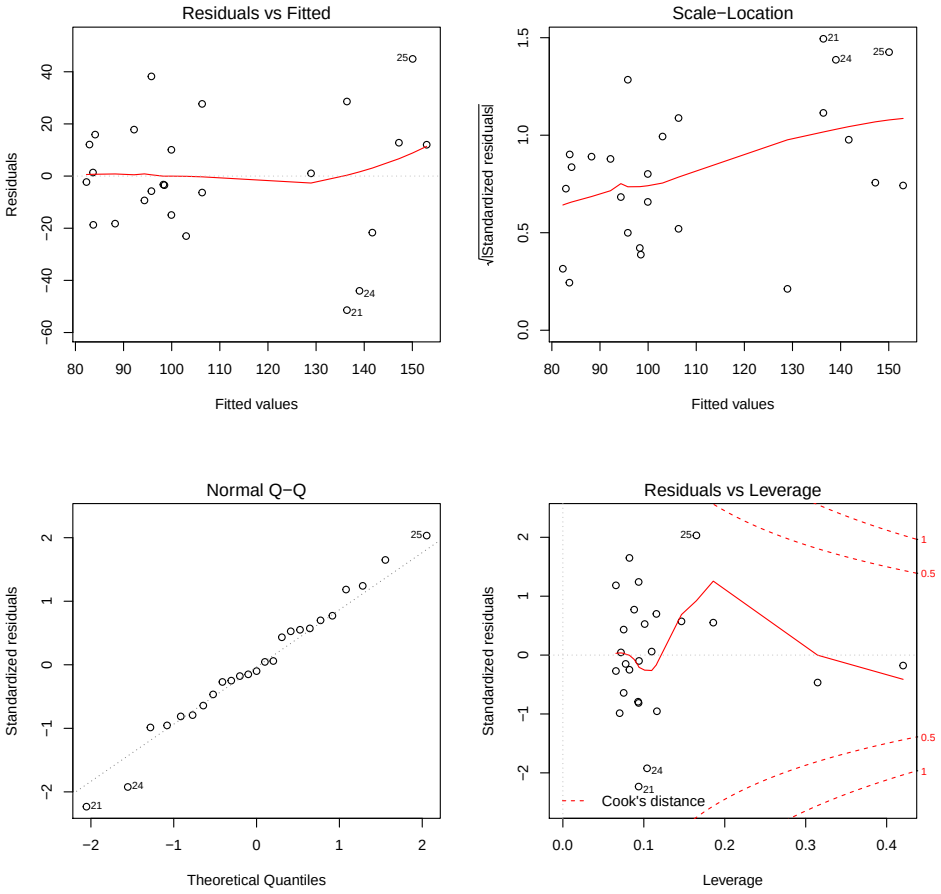
```
cystLm4 <- lm(pemax ~ height + I(height^2), data = cystfibr)
```

Note the use of the `I` wrapper of the square term. The technical reason for this is that the `^` operator has a special meaning in formulas, e.g.

```
(a + b + c)^2 = (a + b + c) * (a + b + c) = a + b + c + a:b + b:c + a:c
```

That is, it can be used to specify complicated models involving interactions among different terms. Therefore, there is a special way to specify that `^` has to be interpreted as "squaring" in the formula, and this is done with the wrapper function `I`.

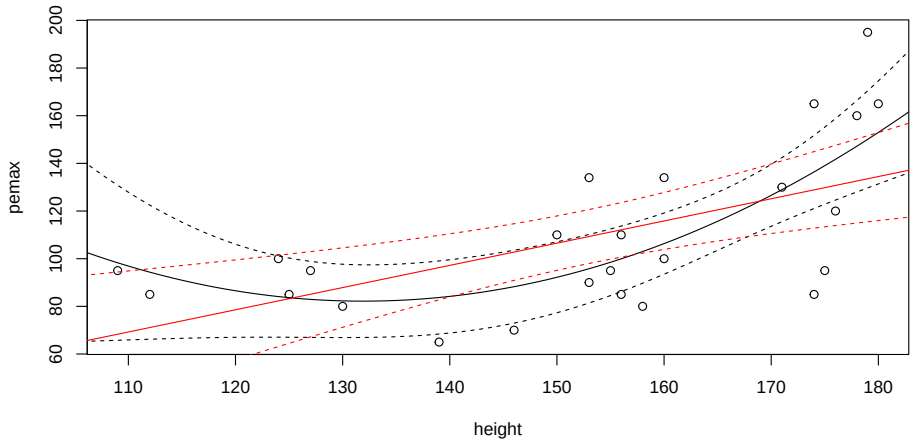
```
summary(cystLm4)

##
## Call:
## lm(formula = pemax ~ height + I(height^2), data = cystfibr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -51.41 -14.93  -2.29  12.79  44.93
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 615.3625   240.9558    2.55    0.018 *
## height       -8.0832     3.3205   -2.43    0.023 *
## I(height^2)   0.0306     0.0113    2.72    0.012 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.2 on 22 degrees of freedom
## Multiple R-squared: 0.52,Adjusted R-squared: 0.477
## F-statistic: 11.9 on 2 and 22 DF,  p-value: 0.000308

par(mfcol = c(2, 2))
plot(cystLm4)  ## Model control
```



The model seems to fit decently, and we can read of directly from the summary table that the coefficient for the quadratic term looks statistically significantly different from 0 with a p-value of 1.25%.

We can compare this model with a corresponding model without the quandratic term in terms of the fitted functional relation (predictions).

```
cystLm5 <- lm(pemax ~ height, data = cystfibr)
newdata <- data.frame(height = seq(100, 190, 1))
cystPred4 <- predict(cystLm4, newdata, interval = "conf")
cystPred5 <- predict(cystLm5, newdata, interval = "conf")
plot(pemax ~ height, data = cystfibr)
matlines(newdata$height, cystPred4, lty = c(1, 2, 2), col = "black")
matlines(newdata$height, cystPred5, lty = c(1, 2, 2), col = "red")
```



A model including all variables in the data set is also possible

```
cystLm6 <- lm(pemax ~ ., data = cystfibr)
summary(cystLm6)


##
## Call:
## lm(formula = pemax ~ ., data = cystfibr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -37.34 -11.53   1.08  13.39  33.41
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  176.058    225.891    0.78     0.45
## age           -2.542      4.802   -0.53     0.60
## sex           -3.737     15.460   -0.24     0.81
## height        -0.446      0.903   -0.49     0.63
## weight         2.993      2.008    1.49     0.16
## bmp           -1.745      1.155   -1.51     0.15
## fev1           1.081      1.081    1.00     0.33
## rv             0.197      0.196    1.00     0.33
## frc           -0.308      0.492   -0.63     0.54
## tlc            0.189      0.500    0.38     0.71
##
## Residual standard error: 25.5 on 15 degrees of freedom
## Multiple R-squared: 0.637,Adjusted R-squared: 0.42
## F-statistic: 2.93 on 9 and 15 DF,  p-value: 0.032
```

One should note that none of the variables have parameters significantly different from 0, but that this is a conclusion based on models where all other variables are included. A combined test of the model with all variables included against the model with only the intercept can be computed.

```
anova(lm(pemax ~ 1, data = cystfibr), cystLm6)

## Analysis of Variance Table
##
## Model 1: pemax ~ 1
## Model 2: pemax ~ age + sex + height + weight + bmp + fev1 + rv + frc +
##     tlc
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1     24 26833
## 2     15  9731  9     17101 2.93  0.032 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is marginally significant suggesting that not all variables can be dropped. Formal model search procedures may be able to produce reasonable models for prediction, though there are better methods than the step-wise procedure considered in ISwR (this is a whole new course in statistical learning or machine learning). The analysis of the data set suggests, in particular, a relation between `pemax` and "bodysize" as measured by one of the variables `age`, `weight` or `height`. Which one to choose and the precise formulation of the model are typically specific to the subject matter. General "rules" are hard to make. Formal statistical procedures can typically not distinguish between highly correlated regressors in a data set.

6.2 Non-linear regression

Keywords: curve fitting, ELISA, non-linear regression.

ISwR: 275-288

ELISA and DNase

ELISA is an assay where we measure the concentration of a protein/antibody indirectly by a measure of "optical density" (OD-value).

For all experiments we are interested in estimating a *standard curve* relating concentration and observed OD-value.

Using a known dilution series we can try linear regression of the *log-OD* on the *log-concentration*.

The model is mildly misspecified – consequence; the error variance is overestimated as it has to capture the model misspecification too.

In this lecture we will also consider estimation using the least squares method of the regression model of the form

$$Y_i = g_\beta(x_i) + \sigma\epsilon_i.$$

This is the maximum-likelihood method if the ϵ_i-terms are normally distributed. In the world of regression this is often referred to as *non-linear least squares*. We can in general not expect to find closed form solutions to these minimization problems and must rely on numerical

optimization. A word of warning is appropriate here. We cannot expect that the numerical optimization always goes as smoothly as desirable. To find the correct set of parameters that globally minimizes the residual sum of squares we may need to choose appropriate starting values for the algorithm to converge, and it may very easily be the case that there are multiple local minima, that we need to avoid.

For non-linear regression there are generalizations of many of the concepts from linear regression. The fitted values are defined as

$$\hat{x}_i = g_{\hat{\beta}}(y_i)$$

and the residuals are

$$e_i = x_i - \hat{x}_i = x_i - g_{\hat{\beta}}(y_i).$$

To check the model we make residual plots of the residuals against either the regressors y_i or the fitted values $\hat{x}_i$ and we look for systematic patterns that either indicate that the model of the mean via the function $g_{\hat{\beta}}$ is inadequate or that the constant variance assumption is problematic. Moreover, we can compare the empirical distribution of the residuals to the normal distribution via a QQ-plot. We don't have a simple leverage measure, nor do we have a formula for the variance of the residuals. So even though the residuals may very well have different variances it is not as easy to introduce standardized residuals that adjust for this. In the context of non-linear regression the term standardized residual often refers to $e_i/\hat{\sigma}$ where we simply divide by the estimated standard deviation.

The four parameter logistic model

A non-linear regression model with

$$g_\beta(y) = \frac{\beta_2 - \beta_1}{1 + \exp(\beta_4(y - \beta_3))} + \beta_1,$$

$\beta = (\beta_1, \beta_2, \beta_3, \beta_3) \in \mathbb{R}^4$ with $\beta_1 < \beta_2$ for identifiability.

Applied a lot to dosis-response modeling and assay standard curve esimation, parameters estimated in R using non-linear least-squares regression with `nls`.

Other models

The functional form of g_β is, in principle, only limited by our fantasy, but practice renders it difficult to estimate very complicated functions. Examples include

- the Gompertz (growth) model with

$$g_\beta(x) = \alpha \exp\left(-\beta e^{-\gamma x}\right)$$

- the Weibull (growth) model

$$g_\beta(x) = \alpha_0 - \alpha_1 \exp\left(-\beta x^{-\gamma}\right)$$

- the Michaelis-Menten rate equation

$$g_\beta(x) = \frac{\alpha x}{\beta + x}.$$

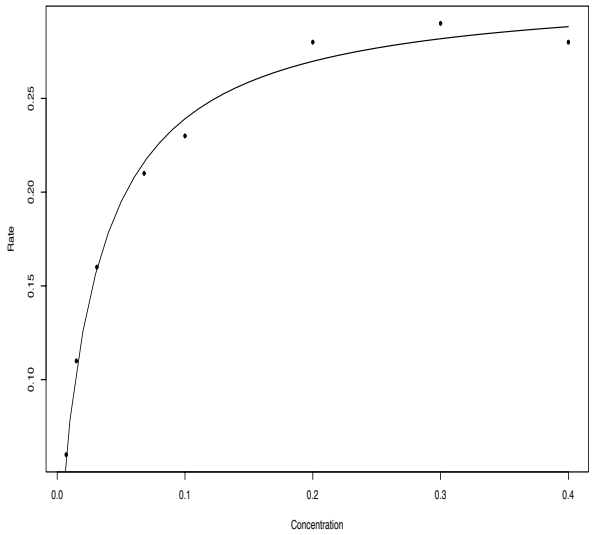


Figure 6.1: The data and estimated Michaelis-Menten rate curve for ethanol conversion.

6.2.1 Michaelis-Menten

Enzymes work as catalysts in the conversion of a substrate into a product. The enzyme *alcohol dehydrogenase* catalyzes the conversion of ethanol (the substrate) to acetaldehyde (the product). The data below are from Bendinskas et al. *Journal of Chemical Education, 82(7), 1068 (2005).* The *Michaelis-Menten rate equation* states that the rate, r, for the conversion is related to the concentration x of the substrate via

$$r = \frac{\beta_1 x}{\beta_2 + x}. \tag{6.1}$$

With $\beta = (\beta_1, \beta_2)$ the two unknown parameters and measurements $r_1, \ldots, r_n$ of the conversion rate for different substrate concentrations $x_1, \ldots, x_n$ we set up a non-linear regression model

$$R = g_\beta(x) + \sigma\epsilon$$

where $g_\beta(x) = \frac{\beta_1 x}{\beta_2 + x}$. We assume in this model that the measurement noise, $\sigma\epsilon$, of the conversion rate enters additively.

Substrate concentration	Conversion rate
0.007	0.06
0.015	0.11
0.031	0.16
0.068	0.21
0.100	0.23
0.200	0.28
0.300	0.29
0.400	0.28

Using the R function `nls` we can fit the model and get the following summary result:

```
Formula: rate ~ beta1 * conc/(beta2 + conc)

Parameters:
      Estimate Std. Error t value Pr(>|t|)
beta1 0.309408   0.006420   48.19 5.35e-09 ***
beta2 0.029391   0.002503   11.74 2.30e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.00807 on 6 degrees of freedom

Number of iterations to convergence: 8
Achieved convergence tolerance: 5.363e-06
```

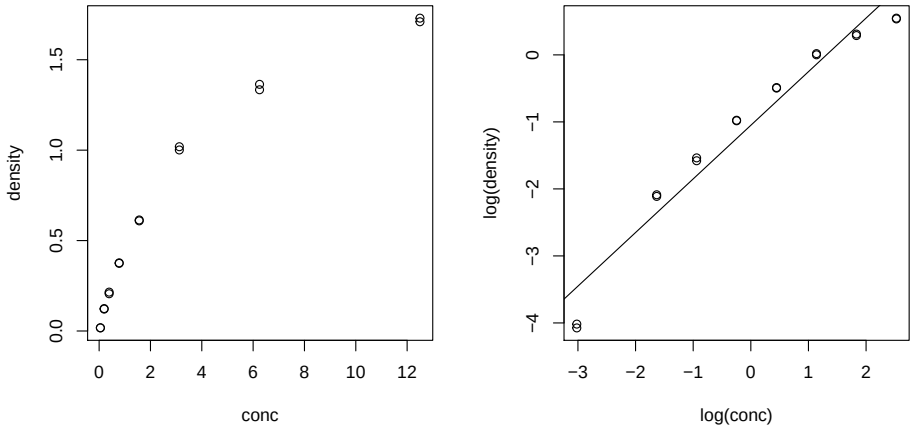
6.2.2 R interlude: Standard curves

```
require(nlme)  ## Just to get the dataset
data(DNase)
```

The data set `DNase` contains data from 11 runs. We select run 1 for this illustration. We do the linear regression estimation, compute the summary and add the estimated straight line to the plot.

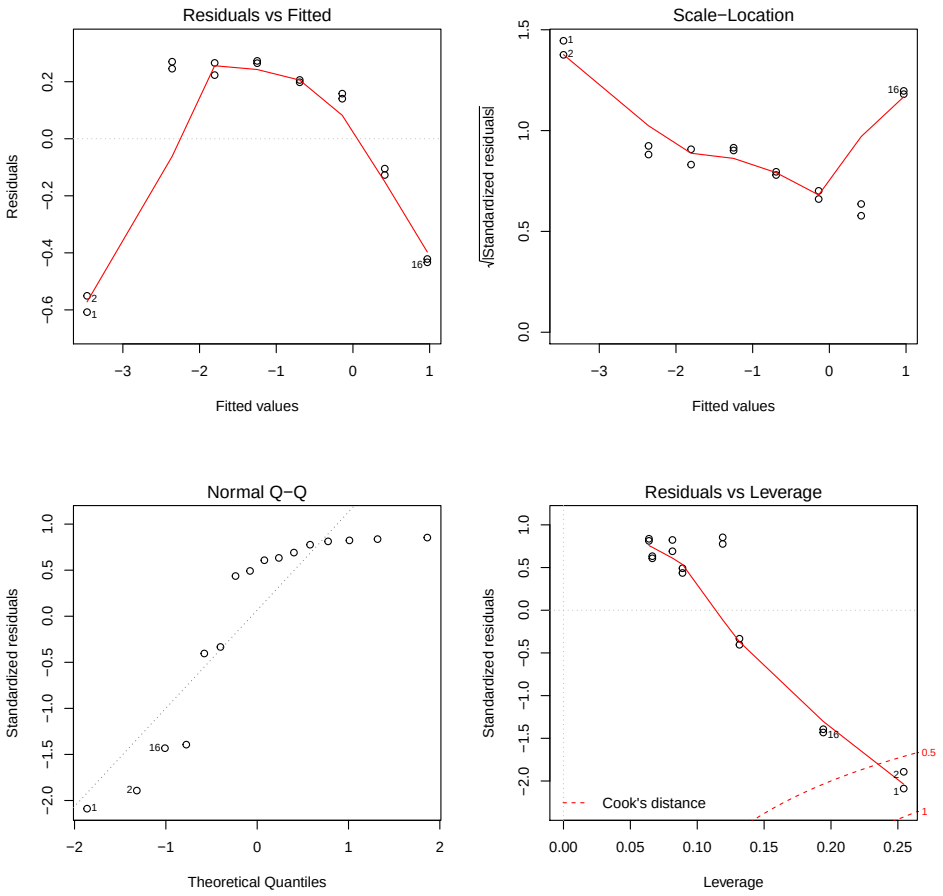
```
myDNase <- DNase[DNase$Run == 1, ]
par(mfcol = c(1, 2))
plot(density ~ conc, myDNase)
plot(log(density) ~ log(conc), myDNase)
DNaseLm <- lm(log(density) ~ log(conc), myDNase)
summary(DNaseLm)


##
## Call:
## lm(formula = log(density) ~ log(conc), data = myDNase)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -0.608 -0.201  0.178  0.250  0.273
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.0510     0.0843   -12.5  5.7e-09 ***
## log(conc)     0.8001     0.0487    16.4  1.5e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.337 on 14 degrees of freedom
## Multiple R-squared: 0.951,Adjusted R-squared: 0.947
## F-statistic:  270 on 1 and 14 DF,  p-value: 1.51e-10
```

```
abline(reg = DNaseLm)
```



The four standard diagnostic plots. The residual plot shows that the model does not quite fit the data.

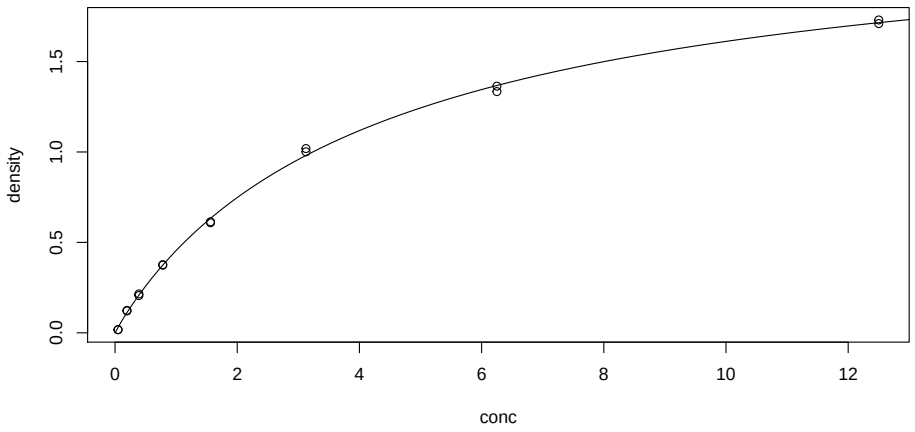
```
par(mfcol = c(2, 2))
plot(DNaseLm)
```



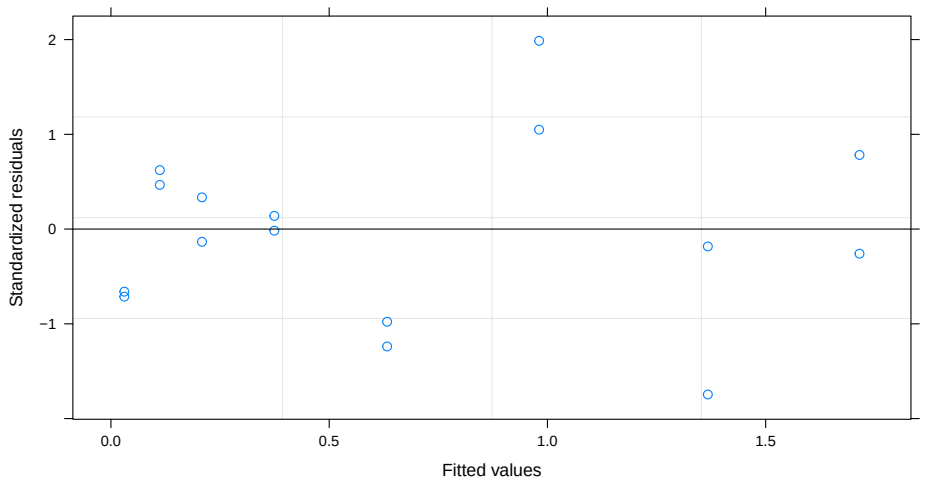
We turn to a non-linear regression model.

```
DNaseNls <- nls(density ~ beta2/(1 + exp(beta4*(log(conc) - beta3))),
                data = myDNase,
                start = list(beta2 = 2, beta3 = 1, beta4 = -1.5))
summary(DNaseNls)

##
## Formula: density ~ beta2/(1 + exp(beta4 * (log(conc) - beta3)))
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## beta2   2.3452     0.0782    30.0  2.2e-13 ***
## beta3   1.4831     0.0814    18.2  1.2e-10 ***
## beta4  -0.9602     0.0298   -32.3  8.5e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0192 on 13 degrees of freedom
##
## Number of iterations to convergence: 5
```

```
## Achieved convergence tolerance: 8.77e-07

plot(density ~ conc, myDNase)
newdata <- data.frame(conc = seq(0, 13, 0.1))
densPred <- predict(DNaseNls, newdata)
lines(newdata$conc, densPred)
```




```
## Residual plot
plot(DNaseNls)
```



It may not be easy to find good stating values. The error message produced below is intentional.

```
DNaseNls <- nls(density ~ beta2/(1 + exp(beta4*(log(conc) - beta3))),
                data = myDNase,
                start = list(beta2 = 0, beta3 = 0, beta4 = 0))

## Error: singular gradient matrix at initial parameter estimates
```

Starting values can be obtained by visual inspection of the graph, or ad hoc estimators. This can sometimes be formalized so that a sensible set of starting values are provided automatically. For the 3-parameters logistic model this is implemented in the `SSlogis`

```
DNaseNls2 <- nls(density ~ SSlogis(log(conc), beta, gamma, delta),
                 data = myDNase)

summary(DNaseNls2)

##
## Formula: density ~ SSlogis(log(conc), beta, gamma, delta)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## beta    2.3452     0.0782    30.0  2.2e-13 ***
## gamma   1.4831     0.0814    18.2  1.2e-10 ***
## delta   1.0415     0.0323    32.3  8.5e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0192 on 13 degrees of freedom
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 3.28e-06
```

Note that the parametrization is

$$f(x) = \beta/(1 + \exp((\gamma - x)/\delta))$$

when using the built in self starting model. Hence $\beta = \beta_2$, $\gamma = \beta_3$ and $\delta = -1/\beta_4$.

There is an extension, called the four-parameter logistic model. It's parametrized as

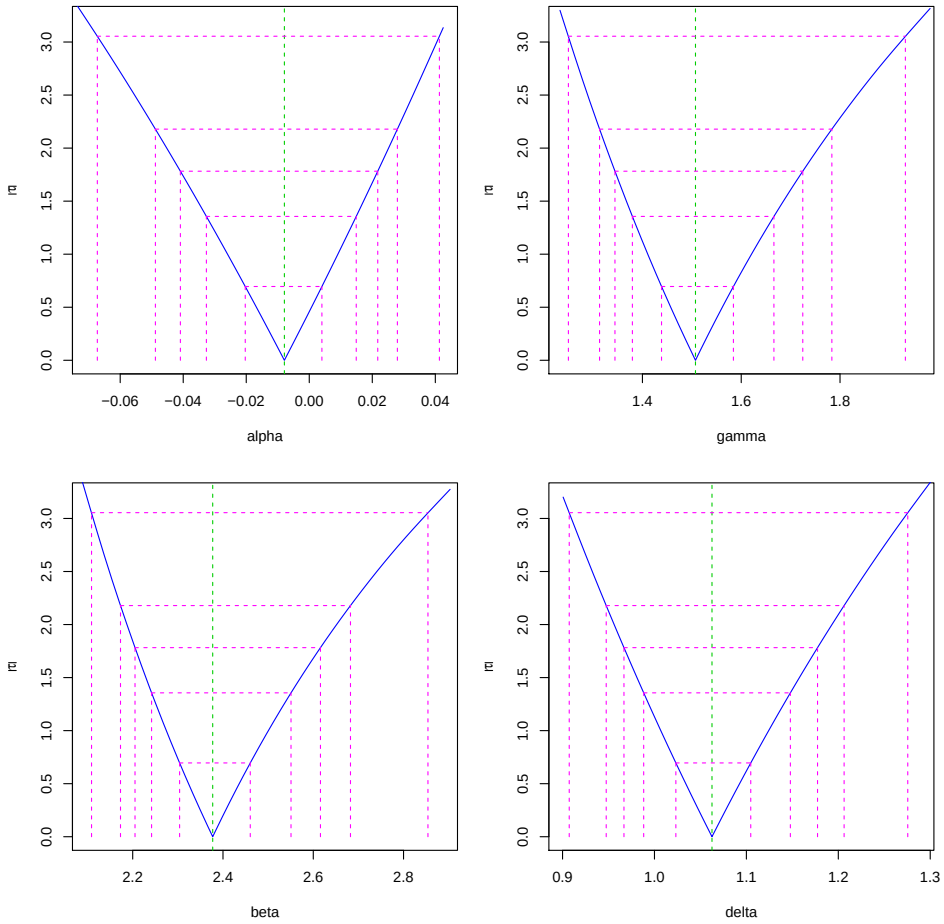
$$f(x) = \alpha + (\beta - \alpha)/(1 + \exp((\gamma - x)/\delta))$$

```
DNaseNls3 <- nls(density ~ SSfpl(log(conc), alpha, beta, gamma, delta),
                 data = myDNase)

summary(DNaseNls3)

##
## Formula: density ~ SSfpl(log(conc), alpha, beta, gamma, delta)
##
## Parameters:
```

```
##       Estimate Std. Error t value Pr(>|t|)
## alpha  -0.0079     0.0172   -0.46     0.65
## beta    2.3772     0.1095   21.71  5.4e-11 ***
## gamma   1.5074     0.1021   14.77  4.6e-09 ***
## delta   1.0626     0.0570   18.64  3.2e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0198 on 12 degrees of freedom
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 2.52e-07

confint(DNaseNls3)

## Waiting for profiling to be done...

##           2.5%   97.5%
## alpha -0.04883 0.02796
## beta   2.17311 2.68247
## gamma  1.31337 1.78354
## delta  0.94754 1.20632

confint.default(DNaseNls3)

##          2.5 %  97.5 %
## alpha -0.04161 0.02581
## beta   2.16259 2.59189
## gamma  1.30733 1.70748
## delta  0.95087 1.17429
```

The conclusion from this is that there is no reason to include the extra parameter. We investigate the profile "likelihood" (sum of squares). Note the clear curvature for some of the parameters.

```
par(mfcol = c(2, 2))
plot(profile(DNaseNls3))
```



Microarray data are similar to the data from the ELISA experiment. We have different concentrations and measure a corresponding light intensity. In the following example we consider a so-called spike-in experiment with known concentrations.

We need some bioconductor packages for the data and the extraction of data.

```
require("SpikeInSubset")
data("spikein95")

spikeIn95 <- as.vector(pm(spikein95, names(pData(spikein95))))
spikeIn95 <- cbind(intensity = spikeIn95,
             data.frame(conc = rep(as.numeric(t(as.matrix(pData(spikein95)))),
                                   each = 16)))
spikeIn95 <- spikeIn95[spikeIn95$conc != 0, ]
```

We analyse a four-parameter logistic model.

```
spikeIn95Nls <- nls(log2(intensity) ~ SSfpl(log2(conc), alpha, beta, gamma, delta),
                    data=spikeIn95
                    )
```

```
summary(spikeIn95Nls)


##
## Formula: log2(intensity) ~ SSfpl(log2(conc), alpha, beta, gamma, delta)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## alpha   7.7950     0.0606   128.7   <2e-16 ***
## beta   12.6808     0.1218   104.2   <2e-16 ***
## gamma   5.8037     0.0943    61.5   <2e-16 ***
## delta   1.0484     0.0880    11.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.29 on 1388 degrees of freedom
##
## Number of iterations to convergence: 10
## Achieved convergence tolerance: 5.2e-06


plot(log2(intensity) ~ jitter(log2(conc)),
     data = spikeIn95,
     pch = 20,
     ylab = "Log-intensity (base 2)",
     xlab="Log-concentration (base 2)"
     )
ord <- order(log2(spikeIn95$conc))
lines(spline(log2(spikeIn95$conc)[ord],
             predict(spikeIn95Nls)[ord]),
      col = "red", lwd = 3)
```

Some more model control, residual plots and QQ-plots.

```
par(mfcol=c(1, 2))
plot(jitter(fitted(spikeIn95Nls), amount = 0.1),
     residuals(spikeIn95Nls,type="pearson"), pch = 20)
abline(c(0, 0))
qqnorm(residuals(spikeIn95Nls), pch = 20)
qqline(residuals(spikeIn95Nls))
plot(density(residuals(spikeIn95Nls)))
hist(residuals(spikeIn95Nls), probability = TRUE, add = TRUE)
```

This shows that the model does not fit perfectly. In particular, the (technical) variance increases with concentration and the residuals seem to have a left skewed distribution.

6.3 Exercises

In an ELISA experiment we consider a standard dilution series and the corresponding measured OD-values:

	Standard dilution									
c	100	100	200	200	400	400	800	800	1600	1600
OD	1.04	1.11	0.710	0.720	0.350	0.380	0.190	0.260	0.090	0.110

In the following you are going to investigate polynomial regression for fitting the relationship between the dilution factor $1/c$ and the OD-values. Thus our model setup is that

$$\mathrm{OD}_i = \beta_0 + \beta_1 1/c_i + \beta_2 (1/c_i)^2 + \ldots + \beta_d (1/c_i)^d + \epsilon_i$$

where we can assume that $\epsilon_1, \ldots, \epsilon_{10}$ are i.i.d. and have distribution $N(0, \sigma^2)$ for some $\sigma^2 > 0$.

Exercise 6.1 Estimate the β-parameters in the three cases $d = 1, 2, 3$. Which of the models seems most appropriate?

We will be interested in the mean value of OD given that $c = 500$. Given estimates of our parameters $\beta_0, \ldots, \beta_d$ we can compute this mean value form the regression formulation above.

Exercise 6.2 Compute in the three cases $d = 1, 2, 3$ the estimate of mean OD when $c = 500$.

The mean value of OD when $c = 500$ is in the following our *parameter of interest*. We want to compute a confidence interval for this parameter.

Exercise 6.3 Take $d = 2$, compute the estimate of σ^2 and use parametric bootstrapping to compute a 95% confidence interval for the parameter of interest – the mean OD value given that $c = 500$.

Exercise 6.4 Take instead $d = 1$, compute the estimate of σ^2 and use parametric bootstrapping to compute a 95% confidence interval for the parameter of interest – the mean OD value given that $c = 500$. Compare with the interval obtained above when $d = 2$. Can you explain the difference?

Exercise 6.5 Show that the Michaelis-Menten rate equation (6.1) can be rephrased as

$$\frac{1}{r} = \frac{\beta_2}{\beta_1}\frac{1}{y} + \frac{1}{\beta_1}.$$

Argue that this formula suggests a linear regression model for the inverse of the rate regressed on the inverse of the substrate concentration. Estimate the parameters using this linear regression model and compare with the example above.

Chapter 7

Seventh Week

7.1 Logistic regression

Keywords: bootstrapping, dosis-response, flies, LD_{50}, logistic regression, odds

ISwR: 227-247

7.1.1 Flies

Dosis-response experiments

"Alle Dinge sind Gift und nichts ist ohne Gift; allein die Dosis macht, dass ein Ding kein Gift ist."

Theophrastus Phillipus Auroleus Bombastus von Hohenheim (1493-1541). All compounds are toxic in the right dose – the question is just what the dose is.

In an experiment, carried out by Jørgen Jespersen at Statens Skadedyrslaboratorium, 260 flies (*Musca domestica*) were exposed to the insecticide dimethoat in varying concentrations. The experiment was organized with 13 groups of 20 flies, and flies in the same group were given the same concentration.

Fly death

Concentration	log(concentration)	Deaths	Survivors
0.016	−4.135	0	20
0.0226	−3.790	0	20
0.032	−3.442	0	20
0.0453	−3.094	0	20
0.064	−2.749	5	15
0.0905	−2.402	4	16
0.128	−2.056	5	15
0.181	−1.709	15	5
0.256	−1.363	14	6
0.362	−1.016	18	2
0.515	−0.664	20	0
0.724	−0.323	20	0
1.024	0.024	20	0

The number of surviving flies depending on the concentration of the insecticide dimethoat.

This is a quite classical situation with a *dichotomous dose-response experiment*. The response is a 0-1 variable, like dead/alive or red/green, but the probability of the variable depends upon a continuous dose, for instance the concentration of a toxic compound, pesticide or insecticide.

Logistic regression

Fly-death is a 0-1-variable (a Bernoulli variable), whose distribution depends upon the log-concentration x of the insecticide. We introduce the point probability that a fly dies ($y_i = 1$) as

$$p(x) = \frac{\exp(\alpha + \beta x)}{1 + \exp(\alpha + \beta x)},$$

where $p(x) \in (0, 1)$ and $\alpha, \beta \in \mathbb{R}$. This is the *logistic regression model*.

The *log-odds*

$$\log \frac{p(x)}{1 - p(x)} = \alpha + \beta x$$

is linear in x.

What is the difference between a probability of $p = 0.5$ and $q = 0.1$ of an event (fly death, say)? How should we interpret such a difference?

What is the difference between between $p = 0.01$ and $q = 0.00112$?

The absolute difference is in the former case 0.4 and in the latter case 0.00888. The relative difference is in the former case 5 and in the latter case 8.93.

If I were to gamble on the event occurring (I bet on the fly dying!) and bet 1 kroner when $p = 0.5$ I expect to be paid 1 kroner in a fair bet if the fly dies. If somebody changes the game by changing the dose so that the probability goes down to 0.1, I expect that I should be paid 9 kroner instead if the fly dies (still betting 1 kroner that it dies). This is because the odds are now

$$\frac{q}{1 - q} = 1/9.$$

The difference in terms of the odds is an odds ratio of 9 – if q is the probability instead of p I should get 9 times the payback if the fly dies.

In the second case the difference in odds – the odds-ratio – is

$$\frac{0.01/0.99}{0.00112/0.99888} = 9.0.$$

Thus the same as in the first case. From a gambling point of view a change of probability from 0.5 to 0.1 or from 0.01 to 0.00112 has the same consequence that the payback should be increased by a factor 9.

The odds interpretation

The odds, $p/(1-p)$ is a quantification of the probability p of an event (fly death).

With q a different probability of the same event, the *odds-ratio* is

$$\frac{p/(1-p)}{q/(1-q)} = \frac{p(1-q)}{q(1-p)}$$

and quantifies, in terms of the odds, how much more probable the event is when p is the probability than when q is the probability.

The *relative risk* is the ratio

$$\frac{p}{q} \left(\simeq \frac{p(1-q)}{q(1-p)} \quad \text{if } p, q \ll 1 \right)$$

which approximately equals the odds-ratio if p and q are small.

Interpretation of β

The parameter β corresponds to a unit change of x,

$$\begin{aligned} \beta &= \beta(x+1) - \beta x \\ &= \alpha + \beta(x+1) - (\alpha + \beta x) \\ &= \log \frac{p(x+1)}{1-p(x+1)} - \log \frac{p(x)}{1-p(x)} \\ &= \log \frac{p(x+1)(1-p(x))}{p(x)(1-p(x+1))}. \end{aligned}$$

Thus β is the log-odds-ratio of a unit change in x and

$$\exp(\beta) = \frac{p(x+1)(1-p(x))}{p(x)(1-p(x+1))}$$

is the odds-ratio (often the parameter of interest) of a unit change in x.

The logistic regression model is particularly popular in so-called retrospective studies. A retrospective study corresponds to a study where we take a group of 100 dead flies and

measure the dose of dimethoat that they were exposed to and take another group of 100 alive flies and measure the dose of dimethoat that they were exposed to. With such a sampling scheme there is no hope of estimating the probability of dying for a given dose. However, the logistic regression model allows us to estimate the odds-ratio between two different doses even with a retrospective study.

Exercise

Write an R function that takes an x-vector and two parameters α and β and simulates the Bernoulli variables with probabilities given by the logistic regression model.

With $\alpha = 5.137$ and $\beta = 2.706$ and the x-vector the log-concentrations from the fly data set simulate a new data set.

Generalized linear models

A Bernoulli variable with probability p of success (being 1) has mean value

$$1 \times p + 0 \times (1 - p) = p.$$

The function

$$p \mapsto \log \frac{p}{1-p} = \text{logit}(p)$$

is called the *logistic function* (or logit for short).

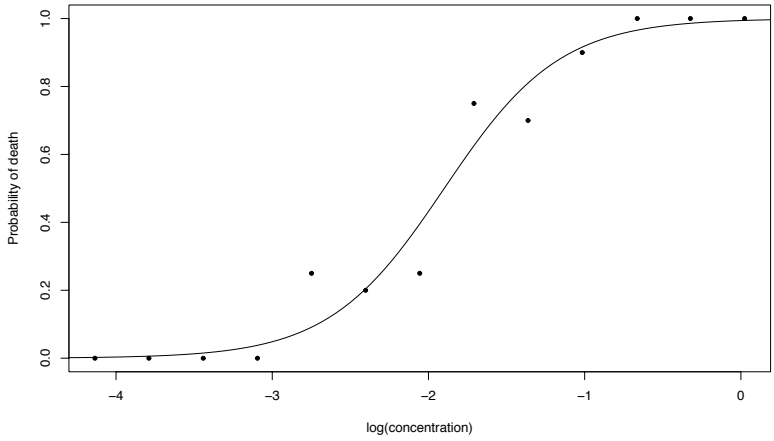
The regression model we consider is a *binomial* model with the logit of the mean being a linear function of the regressor

$$\text{logit}(p(y)) = \alpha + \beta y.$$

It is in the class of *generalized linear models* with logit *link function*, ML-estimation and other statistical analyses are done in R with the `glm` function with argument

```
family = binomial("logit")
```

Logistic regression curve - fly-death



Lethal dose 50

For the logistic regression model the log odds equals 0 precisely when $p(x) = 1/2$ and this happens when $x = -\alpha/\beta$.

The value of x where $p(x) = 1/2$ is called LD_{50}, which means the *Lethal Dose* for 50% of the subjects considered.

In terms of the parameters in our logistic regression model

$$\mathrm{LD}_{50} = -\frac{\alpha}{\beta}.$$

Often LD_{50} is the parameter of interest and it is used to summarize the toxicity level of the insecticide.

In other contexts LD_{50} is referred to as MD_{50} instead, which means Median Effective Dose. It has the same interpretations.

7.1.2 Bootstrapping

Confidence intervals and bootstrapping

One practical problem is how to compute a sensible confidence interval (or make a formal test) on derived parameters like $\mathrm{LD}_{50} = -\alpha/\beta$.

One solution is to use profiling methods, but it requires, in principle, a reparametrization in terms of the derived parameter.

Simulation based *bootstrapping* is a different idea that can be implemented with little analytic skills at the expense of computer time.

The basic idea in bootstrapping for constructing confidence intervals for a parameter of interest τ, as a derived parameter $\tau = \tau(\theta)$ of the full parameter θ, when we have an estimator $\hat{\tau}$ of τ is to find an approximation of the distribution of $\hat{\tau} - \tau(\theta_0)$. We don't know "the true" parameter θ_0, and the approximation is usually done by simulations that depend upon the observed data set. What we attempt is to approximate the distribution of

$$\hat{\tau} - \tau(\theta_0),$$

by the distribution of

$$\hat{\tau} - \hat{\tau}(x),$$

which depends on the data set x but not the unknown parameter. Since the distribution is allowed to depend upon the concrete observation x the construction of the resulting confidence set – that provides information about the uncertainty of the estimate $\hat{\tau}(x)$ – depends upon the data itself. Hence what we suggest is to pull information about the uncertainty of an estimate out from the very same data used to obtain the estimate, and for this reason the method is known as *bootstrapping*. Supposedly one of the stories in *The Surprising Adventures of Baron Munchausen* by Rudolf Erich Raspe (1736 - 1794) contains a passage where the Baron pulls himself out of a deep lake by pulling his own bootstraps. Such a story is, however, not to be found in the original writings by Raspe, but the stories of Baron Munchausen were borrowed and expanded by other writers, and one can find versions where the Baron indeed did something like that. To bootstrap is nowadays, with reference to the Baron Munchausen story, used to describe various seemingly paradoxical constructions or actions. To boot a computer is for instance an abbreviation of running a so-called bootstrap procedure that gets the computer up and running from scratch.

The computation of the distribution of $\hat{\tau} - \hat{\tau}(x)$ is usually done by letting the computer generate a large number of new data sets $x_1, \ldots, x_B$ and relevant derived quantities such as quantiles for the distribution of $\hat{\tau} - \hat{\tau}(x)$ or the standard deviation of $\hat{\tau}$ are estimated from the simulated data.

The bootstrap for confidence interval construction

With x the data set, $\hat{\theta} = \hat{\theta}(x)$ the estimated full parameter and $\hat{\tau} = \hat{\tau}(x)$ the estimated parameter of interest the *parametric bootstrap algorithm* for producing a level $(1-\alpha)$-confidence interval:

- Simulate B new independent, identically distributed data sets, $x_1, \ldots, x_B$, from the estimated model with parameter $\hat{\theta}$.
- Compute, for each dataset x_i, $i = 1, \ldots, B$, new estimates $\hat{\tau}(x_i)$ using the estimator $\hat{\tau}$.
- Compute $\hat{z}_\alpha$ and $\hat{w}_\alpha$ as the $\alpha/2$ and $1 - \alpha/2$ quantiles for the sample $\hat{\tau}(x_i) - \hat{\tau}(x)$, $i = 1, \ldots, B$.
- Define $I(x) = [\hat{\tau}(x) - \hat{w}_\alpha, \hat{\tau}(x) - \hat{z}_\alpha]$.

The bootstrap for confidence interval construction

A minor modification of the algorithm is given by replacing the last two bullet points by

- Simulate B new independent, identically distributed datasets, $x_1, \ldots, x_B$, from the estimated model with parameter $\hat{\theta}$.
- Compute, for each dataset x_i, $i = 1, \ldots, B$, new estimates $\hat{\tau}(x_i)$ using the estimator $\hat{\tau}$.
- Compute the empirical mean $\overline{\tau} = \frac{1}{B}\sum_{i=1}^{B}\hat{\tau}(x_i)$ and the empirical standard deviation

$$\hat{\text{se}} = \sqrt{\frac{1}{B-1}\sum_{i=1}^{B}(\hat{\tau}(x_i) - \overline{\tau})^2}$$

- Define $I(x) = [\hat{\tau}(x) - \hat{\text{se}}z_\alpha, \hat{\tau}(x) + \hat{\text{se}}z_\alpha]$ where z_α is the $1 - \alpha/2$ quantile for the $N(0,1)$-distribution.

7.1.3 Logistic regression likelihood

Likelihood A

Fly-death is a 0-1-variable (a Bernoulli variable), whose distribution depends upon the log-concentration, x, of the insecticide. Let $p(x)$ denote the probability of fly death as a function of dose.

The likelihood is

$$\mathscr{L}_{x,y} = \prod_{i=1}^{260} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

and the log-likelihood function becomes

$$l_{x,y} = \sum_{i=1}^{260} y_i \log\left(\frac{p(x_i)}{1-p(x_i)}\right) - \log(1 - p(x_i)),$$

Likelihood B

Fly-death is a binomial$(p(x), 20)$ variable, whose distribution depends upon the log-concentration, x, of the insecticide.

The likelihood is

$$\mathscr{L}_{x,y} = \prod_{i=1}^{13} \binom{20}{y_i} p(x_i)^{y_i}(1 - p(x_i))^{20-y_i}$$

and the log-likelihood function becomes

$$l_{x,y} = \sum_{i=1}^{13} y_i \log\left(\frac{p(x_i)}{1-p(x_i)}\right) - \log(1 - p(x_i)) + \log\binom{20}{y_i},$$

The minus-log-likelihood

The two minus-log-likelihoods (A and B) are identical up to a constant that does not depend upon the parameters when the dose within groups is constant. The general likelihood function is

$$\mathscr{L}_{x,y}(\alpha,\beta) = \prod_{i=1}^{n} p(x_i)^{y_i}(1-p(x_i))^{1-y_i} = \prod_{i=1}^{n} \frac{\exp(\alpha y_i + \beta x_i y_i)}{1+\exp(\alpha+\beta x_i)}$$

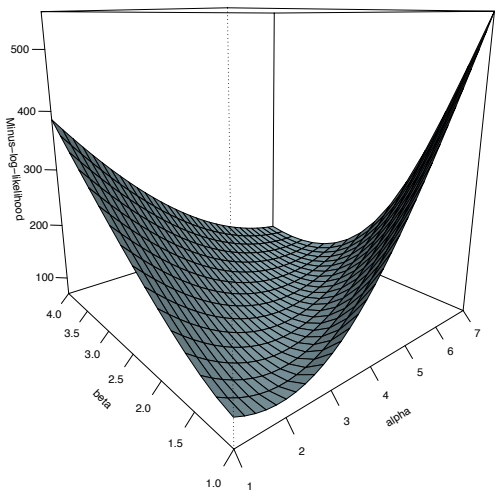
and the minus-log-likelihood function becomes

$$l_{x,y}(\alpha,\beta) \quad = \quad \sum_{i=1}^{n} \log(1+\exp(\alpha+\beta x_i)) - \alpha S - \beta SS,$$

where

$$S = \sum_{i=1}^{n} y_i \quad \text{and} \quad SS = \sum_{i=1}^{n} y_i x_i.$$

Minus-log-likelihood function



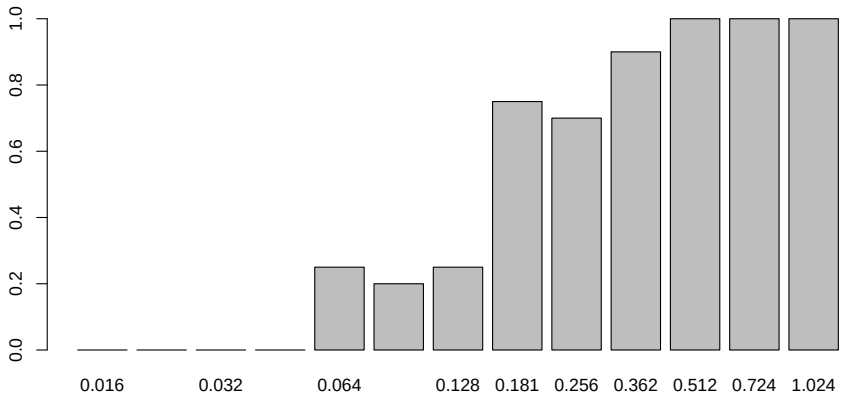
7.1.4 R interlude: Flies

Downloading the flies data (the data is in the data frame `fly.death`).

```
download.file("http://www.math.ku.dk/~richard/download/courses/binf_2008/flies.RData",
              "flies.RData")
load("flies.RData")
```

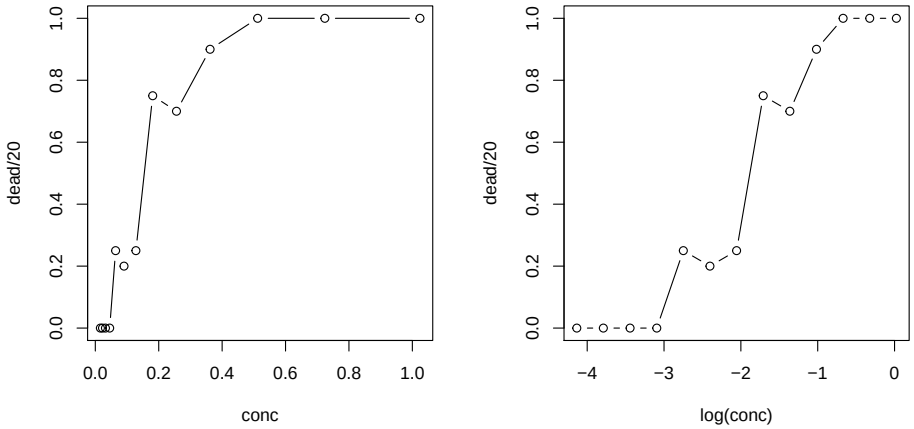
Tabulation and various plots.

```
flyDeathTable <- table(fly.death)

barplot(flyDeathTable[, 2] / 20)
```




```
flyDeath <- as.data.frame(flyDeathTable)
flyDeath <- data.frame(conc = as.numeric(as.character(flyDeath$conc[1:13])),
                       alive = flyDeath$Freq[1:13],
                       dead = flyDeath$Freq[14:26]
                       )

flyDeath$conc <- as.numeric(as.character(flyDeath$conc))
```

```
par(mfcol = c(1, 2))
plot(dead / 20 ~ conc, flyDeath, type = "b")
plot(dead / 20 ~ log(conc), flyDeath, type = "b")
```



It is often found that it is better to regress on log-concentration than concentration. Consequently the experiment is designed with equidistant log-concentrations.

MLE computation and further statistical analysis are based on the `glm` function.

```
flyGlm <- glm(dead ~ log(conc),
              family = binomial("logit"),
              data = fly.death,
              )
summary(flyGlm)


##
## Call:
## glm(formula = dead ~ log(conc), family = binomial("logit"), data = fly.death)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2260  -0.2778  -0.0686   0.2655   2.1888
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    5.140      0.674    7.62  2.5e-14 ***
## log(conc)      2.707      0.335    8.07  7.1e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 359.19  on 259  degrees of freedom
## Residual deviance: 138.69  on 258  degrees of freedom
## AIC: 142.7
##
## Number of Fisher Scoring iterations: 6

confint(flyGlm)  ## Profile based confidence intervals
```

```
## Waiting for profiling to be done...

##             2.5 % 97.5 %
## (Intercept) 3.943  6.607
## log(conc)   2.115  3.440
```

Standard confidence intervals based on the estimated standard error can be computed by calling `confint.default` explicitly.

```
confint.default(flyGlm)

##             2.5 % 97.5 %
## (Intercept) 3.818  6.462
## log(conc)   2.049  3.364
```

Formal likelihood ratio based test of $H : \beta = 0$.

```
anova(flyGlm, test = "Chisq")

## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: dead
##
## Terms added sequentially (first to last)
##
##
##           Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                        259        359
## log(conc)  1      220       258        139   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The model can also be fitted based on the tabular data. This gives an identical model fit, but different values for the Null and Residual deviances.

```
flyGlm2 <- glm(cbind(dead, alive) ~ log(conc),
               family = binomial("logit"),
               data = flyDeath,
               )
summary(flyGlm2)

##
## Call:
## glm(formula = cbind(dead, alive) ~ log(conc), family = binomial("logit"),
##     data = flyDeath)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
```

```
## -1.373  -0.781  -0.252   0.747   2.080
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    5.140      0.674    7.62  2.5e-14 ***
## log(conc)      2.707      0.335    8.07  7.1e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 234.255  on 12  degrees of freedom
## Residual deviance:  13.755  on 11  degrees of freedom
## AIC: 36.2
##
## Number of Fisher Scoring iterations: 5
```

Formal likelihood ratio based test of $H : \beta = 0$ gives the same result as above. The difference between the deviances is the same.

```
anova(flyGlm2, test = "Chisq")


## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: cbind(dead, alive)
##
## Terms added sequentially (first to last)
##
##
##           Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                         12      234.3
## log(conc)  1      220        11       13.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results computed above can also be computed using the generic tools from the `stats4` package.

Note that the defaults below makes it possible to just call `mle` without specifying initial values for the algorithm.

```
require(stats4)
x <- log(fly.death[, 1])
y <- fly.death[, 2]
S = sum(y)
SS = sum(y * x) ## y %*% x

mll <- function(alpha = 0, beta = 0)
  sum(log(1 + exp(alpha + beta * x))) - alpha * S - beta * SS
```

```
flyMle <- mle(mll)
summary(flyMle)

## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = mll)
##
## Coefficients:
##       Estimate Std. Error
## alpha    5.140     0.6744
## beta     2.707     0.3354
##
## -2 log L: 138.7

confint(flyMle)

## Profiling...

##       2.5 % 97.5 %
## alpha 3.943  6.607
## beta  2.115  3.440
```

One benefit of this procedure is that we have full control over the parametrization. If we wish to use the parametrization in terms of LD_{50} (γ below) and β (to obtain profile based confidence intervals for LD_{50}, say), we are free to do so.

```
mll2 <- function(gamma = 0, beta = 0) mll(- gamma * beta, beta)
flyMle2 <- mle(mll2)
summary(flyMle2)

## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = mll2)
##
## Coefficients:
##       Estimate Std. Error
## gamma   -1.899     0.0802
## beta     2.707     0.3354
##
## -2 log L: 138.7

confint(flyMle2)

## Profiling...

##        2.5 % 97.5 %
## gamma -2.057  -1.74
## beta   2.115   3.44
```

7.1.5 R interlude: Bootstrapping flies

```
fit <- function(myData) {
  theFit <- glm(dead ~ log(conc),
                family = binomial,
                data = myData)  ## Using glm to fit the logistic regression model
  return(coefficients(theFit))
}
```

Warning, the following is not recommended for production usage. The repeated calls of `glm` is way to slow, and can be replaced by calls of `glm.fit`, which is faster, but less user friendly for interactive usage. For the present purpose the following implementation works just fine.

```
boot <- function(myData, B = 999) {
  ## Initial fit
  fitted <- glm(dead ~ log(conc),
                family = binomial,
                data = myData)$fitted
  bootPar <- data.frame(alpha = numeric(B), beta = numeric(B))

  ## The actual bootstrapping
  for(i in 1:B){
    bootData <- data.frame(conc = myData$conc,
                           dead = rbinom(length(fitted), 1, fitted))
    bootPar[i, ] <- fit(bootData)
  }
  return(as.data.frame(bootPar))
}
```

We do 999 bootstrap replications.

```
parDist <- boot(fly.death)
```

We compute confidence intervals based on two different methods.

For the normal approximation the estimate of the standard error is taken from the bootstrap procedure.

```
flyFit <-  glm(dead ~ log(conc),
               family = binomial,
               data = fly.death)

alphaHat <- coefficients(flyFit)[1]
betaHat <-  coefficients(flyFit)[2]
alphaHat + 1.96 * sqrt(var(parDist$alpha)) * c(-1, 1)

## [1] 3.805 6.474

betaHat + 1.96 * sqrt(var(parDist$beta)) * c(-1, 1)

## [1] 2.032 3.381
```

```
## and LD50
-alphaHat/betaHat +  1.96 * sqrt(var(- parDist$alpha / parDist$beta)) * c(-1, 1)

## [1] -2.050 -1.749
```

We can also use bootstrap quantiles directly so that the confidence intervals do not rely on the normal approximation.

```
2 * alphaHat - quantile(parDist$alpha, c(0.975, 0.025))

## 97.5%  2.5%
## 3.506 6.191

2 * betaHat - quantile(parDist$beta, c(0.975, 0.025))

## 97.5%  2.5%
## 1.832 3.206

## and LD50
-2 * alphaHat / betaHat - quantile(-parDist$alpha / parDist$beta, c(0.975, 0.025))

##  97.5%   2.5%
## -2.042 -1.742
```

Compare the two previous confidence intervals with the confidence interval computed above for the γ parameter but based on the profile likelihood method.

7.2 Poisson regression

Keywords: log-linear model, MEF2, motif occurrences, Poisson regression

ISwR: 259-270

Motivating question – miRNA co-occurrences

In a thesis project at binf a central question was

Do putative target sites for miRNA in human 3'UTR's co-occur?

The scientific objective is to understand the function of miRNA and whether different miRNAs collaboratively target genes.

Computationally determined target sites have many false positives or a pour sensitivity.

Statistical challenge: Find a sound measure of co-occurrence that will detect pairs of miRNAs that have a tendency to co-occur.

Note that we do not try to pinpoint the target sites where the miRNAs may co-bind in the 3'UTRs. We only try to justify if certain miRNAs have putative target sites that co-occur beyond expectations.

Solution and problems

The easy solution is to tabulate and test for independence. That is, for each 3'UTR sequence register which target sites are present, cross-tabulate the result, and test for independence using a χ^2-test.

Problems:

- The 3'UTR's are of different lengths.
- The nucleotide composition in the 3'UTR's is heterogeneous.

Both of the problems above lead to different probabilities of a random co-occurrence of target sites in different 3'UTR's. It is a fundamental assumption for the χ^2-test for independence that the tabulated data are independent and identically distributed.

What we aim at is a more precise model of word and motif occurrences in random sequences that we can use as a reference model.

7.2.1 A log-linear counting model

A computation in a simple model

If $X_i, \ldots, X_{i+m-1}$ are random variables representing the DNA letters in a random sequence at the positions $i, \ldots, i+m-1$ are *independent*, and $w = w_1 \ldots w_m$ is a word, then

$$\begin{aligned} P(X_i X_{i+1} \ldots X_{i+m-1} = w) &= P(X_i = w_1) \times \ldots \times P(X_{i+m-1} = w_m) \\ &= p(w_1)p(w_2)\ldots p(w_m) \\ &= p(\mathrm{A})^{n_w(\mathrm{A})} p(\mathrm{C})^{n_w(\mathrm{C})} p(\mathrm{G})^{n_w(\mathrm{G})} p(\mathrm{T})^{n_w(\mathrm{T})} \end{aligned}$$

where $p(\mathrm{A}), p(\mathrm{C}), p(\mathrm{G})$ and $p(\mathrm{T})$ are the point probabilities for the distribution of the letters and $n_w(\mathrm{A}), n_w(\mathrm{C}), n_w(\mathrm{G})$ and $n_w(\mathrm{T})$ are the number of A's, C's, G's and T's in w.

An expectation

If N denotes the number of occurrences of the word in the entire sequence of length n then its expectation is

$$\mu = (n-m+1) p(\mathrm{A})^{n_w(\mathrm{A})} p(\mathrm{C})^{n_w(\mathrm{C})} p(\mathrm{G})^{n_w(\mathrm{G})} p(\mathrm{T})^{n_w(\mathrm{T})}.$$

with μ denoting the expectation, or mean, of N.

Thus

$$\begin{aligned}\log \mu &= \log(n-m+1) \\ &\quad + n_w(\text{A}) \log p(\text{A}) + n_w(\text{C}) \log p(\text{C}) \\ &\quad + n_w(\text{G}) \log p(\text{G}) + n_w(\text{T}) \log p(\text{T}).\end{aligned}$$

A log-linear model

The model is a *log-linear* model in the log-length and log-probabilities.

Or a generalized linear model with a *log-link function*.

$$\begin{aligned}\log \mu &= \alpha_{n,m} \\ &\quad + \beta_{\text{A}} \log p(\text{A}) + \beta_{\text{C}} \log p(\text{C}) \\ &\quad + \beta_{\text{G}} \log p(\text{G}) + \beta_{\text{T}} \log p(\text{T}).\end{aligned}$$

Note that the coefficients are known for a given word under this model.

Generalizations

The model above is for inspiration only because

- motifs are generally composite and not just single word motifs,
- the actual distribution, even for non-self-overlapping word, is relatively complicated though the mean is simple,
- but most importantly, DNA sequences do not consist of i.i.d. random letters.

The model is quantitatively wrong, but the log-linear structure of the mean in the log of the letter frequencies may be a resonable general model assumption.

The Poisson regression model

Recall that the Poisson distribution with parameter $\lambda > 0$ on the positive integers $\{0, 1, 2, \ldots\}$ has point probabilities

$$p(n) = \frac{\lambda^n}{n!} e^{-\lambda}$$

and mean λ. It also has variance λ.

A *log-linear* Poisson regression is a regression model where the observations are Poisson distributed and the mean is specified as

$$\log(\lambda) = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k$$

based on k regressors $x_1, \ldots, x_k$.

As usual in regression, the regressors may themself be transformations of observed variables, e.g. the logarithm of relative frequencies. Note that there is no error term in the model. The error is implicit in the assumption of a Poisson distribution, but it is not an additive error in the formula. In particular, one should note that there is no scale parameter and the variance equals the mean in the Poisson regression model.

7.2.2 MEF2 binding sites

The binding sites for the myocyte-specific enhancer factor 2 (MEF2) involved in the muscle-specific expression of a number of genes were obtained from TRANSFAC.

A motif is created for the binding site for MEF2 in skeletal muscles based on 104 binding sites.

MEF2 motif count matrix

	Position										
	1	2	3	4	5	6	7	8	9	10	11
A	3	3	88	60	95	100	97	0	104	9	35
C	89	8	0	0	0	0	0	0	0	0	49
G	3	0	0	0	1	0	1	0	0	94	11
T	9	93	16	44	8	4	6	104	0	1	9

From this the consensus pattern is seen to be CTAAAAATAGC.

Regarding each column as observations from independent random variables (whose distributions are position specific), we can estimate a position specific distribution of nucleotides at each of the 11 positions. Notationally we write $p_i(x)$ for the probability of nucleotide x at position i.

Weights

With reference nucleotide probabilities

$p(\mathrm{A})$	$p(\mathrm{C})$	$p(\mathrm{G})$	$p(\mathrm{T})$
0.255	0.238	0.245	0.262

,

which are obtained as the average nucleotide frequencies in the dataset, a weight matrix is computed with entries

$$s_i(x) = \log \frac{p_i(x)}{p(x)}$$

for $i = 1, 2, \ldots, 11$ and $x = \mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}$.

A pseudo-count of 1 was added to all positions in the count matrix before the probabilities $p_i(x)$ were estimated.

Position Specific Weight Matrix (PSWM)

	Position										
	1	2	3	4	5	6	7	8	9	10	11
A	-1.93	-1.93	1.17	0.8	1.25	1.3	1.27	-3.32	1.34	-1.01	0.27
C	1.25	-1.05	-3.25	-3.25	-3.25	-3.25	-3.25	-3.25	-3.25	-3.25	0.67
G	-1.89	-3.28	-3.28	-3.28	-2.58	-3.28	-2.58	-3.28	-3.28	1.28	-0.79
T	-1.04	1.2	-0.51	0.46	-1.15	-1.73	-1.4	1.31	-3.34	-2.65	-1.04

Data

From the human chromosome 1 (Ensembl 41) 400 gene transcripts were chosen at random, and we extracted 5000 nucleotides from the 5'-flanking region for each of the transcripts.

Using the PSWM for the MEF2-factor binding site, we search each of the 400 sequences for occurrences of a word with a score greater than a baseline threshold $t_0 = -2$.

For each sequence we also computed the relative nucleotide frequencies and di-nucleotide frequencies.

For each of the 400 flanking regions we will model the count of the number of motif with a score exceeding $t > t_0 = -2$ using a Poisson regression model with mean λ_t.

Each of the occurrences is recorded together with the actual score, the position and the word. This part of the dataset is organized in a table with four columns and each row corresponding to a single word occurrence with a score exceeding t_0. The nucleotide and di-nucleotide frequencies are collected in a table with 21 columns giving the sequence identifier (between 1 and 400) and the 20 relative frequencies.

The model

The Poisson regression model of the counts for threshold t is

$$\log(\lambda_t) = \beta_0 + \beta_{\mathrm{A}} \log(f_{\mathrm{A}}) + \beta_{\mathrm{C}} \log(f_{\mathrm{C}}) + \beta_{\mathrm{G}} \log(f_{\mathrm{G}}) + \beta_{\mathrm{T}} \log(f_{\mathrm{T}}). \tag{7.1}$$

where $(f_{\mathrm{A}}, f_{\mathrm{C}}, f_{\mathrm{G}}, f_{\mathrm{T}})$ denotes the vector of relative frequencies of nucleotides in the sequence considered.

Excess distributions

It might also be interesting to study the excess score above the chosen threshold. That is

$$s_i - t$$

for those motifs with a score $\geq t$.

After all, motifs with extreme values of the score corresponds to better matches of the motif.

miRNA model conclusions

The model worked fine as a reference model for the (random) occurrences of putative miRNA target sites in human 3'UTRs.

The model was able to detect putative miRNA target sites that showed considerable co-occurrence relative to the reference model.

Further analysis revealed that the co-occurring motifs were part of a longer repeat pattern (SINEs) and when removed no other clear signals were detected.

7.2.3 R interlude: MEF2 binding site occurrences

The data set consists of two parts. The `counts` data frame contains the "matches" of the MEF2 motif with a score above -2 in the 400 human flanking DNA sequences. The `nucl` data frame contains frequencies of nucleotides and dinucleotides for the 400 flanking DNA sequences.

```
counts <- read.table("http://www.math.ku.dk/~richard/courses/StatScience2011/pr2_counts.txt",
  header=TRUE)
nucl <- read.table("http://www.math.ku.dk/~richard/courses/StatScience2011/pr2_nucl_freq.txt",
                   header=TRUE)
head(counts)
```

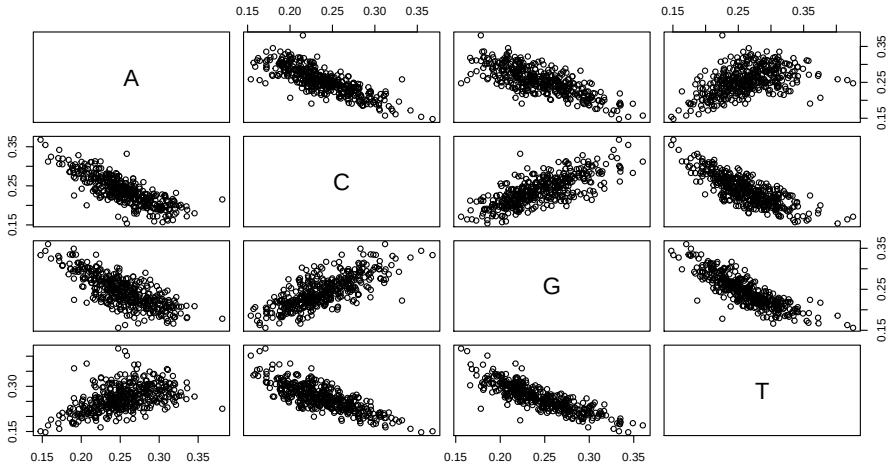
```
##   id   score        word position
## 1  1 -1.0621 AAATGTATAGA       16
## 2  1  7.9177 CTAAAAATACA      147
## 3  1 -1.0178 ACAAAAATTAG      155
## 4  1  7.0440 CAAAAATTAGC      156
## 5  1 -0.2032 CTCAAAAAAAT      298
## 6  1 -0.6523 TCAAAAAAATA      299
```

```
head(nucl)
```

```
##   id      A      C      G      T     AA     AC     AG     AT     CA     CC
## 1  1 0.2570 0.2662 0.2592 0.2176 0.0756 0.0526 0.0844 0.0444 0.0786 0.0892
## 2  2 0.2548 0.2272 0.2114 0.3066 0.0716 0.0468 0.0714 0.0650 0.0742 0.0716
## 3  3 0.2338 0.2458 0.2530 0.2674 0.0636 0.0464 0.0728 0.0510 0.0666 0.0760
## 4  4 0.3042 0.2108 0.1982 0.2868 0.1058 0.0552 0.0702 0.0730 0.0714 0.0526
## 5  5 0.2978 0.1896 0.2356 0.2770 0.0980 0.0456 0.0830 0.0710 0.0698 0.0474
## 6  6 0.2546 0.2466 0.2348 0.2640 0.0742 0.0540 0.0714 0.0550 0.0718 0.0742
##       CG     CT     GA     GC     GG     GT     TA     TC     TG     TT
## 1 0.0256 0.0728 0.0660 0.0688 0.0794 0.0448 0.0368 0.0554 0.0698 0.0556
## 2 0.0060 0.0754 0.0604 0.0452 0.0548 0.0508 0.0484 0.0636 0.0792 0.1154
## 3 0.0262 0.0770 0.0576 0.0660 0.0792 0.0502 0.0460 0.0574 0.0746 0.0892
## 4 0.0076 0.0790 0.0650 0.0366 0.0474 0.0492 0.0620 0.0664 0.0730 0.0854
## 5 0.0048 0.0676 0.0706 0.0444 0.0626 0.0580 0.0594 0.0522 0.0852 0.0802
## 6 0.0196 0.0810 0.0578 0.0574 0.0658 0.0536 0.0508 0.0610 0.0778 0.0744
```

The plot of nucleotide frequencies and di-nucleotides against each other. Note that, as should be expected, these variables are clearly dependent.

```
plot(nucl[, 2:5])
```




```
## plot(nucl[, 6:21], gap = 0, pch = 20, col = "#F1000020")
```

Setting the opacity or transparency of colors with a lot of overplotting gives more readable plots. See `help(rgb)`.

For later use we construct a function that tabulates and organizes the data in a data frame useful for later analyzes.

```
select.data <- function(threshold = 0) {
  tmp <- table(counts[counts$score > threshold, ]$id)
  temp <- as.numeric(dim(nucl)[1])
temp[as.numeric(names(tmp))] <- tmp
       return(cbind(data.frame(count = temp), nucl))
}
count.data <- select.data(0)
```

We construct various models by programatically building formulas.

```
form1 <- as.formula(paste("count ~",
                          paste("log(", names(count.data)[3:6], ")",
                                sep = "", collapse = "+")
                          )
                    )
form2 <- as.formula(paste("count ~",
                          paste("log(", names(count.data)[7:22], ")",
                                sep = "", collapse = "+")
                          )
                    )

form3 <- as.formula(paste("count ~",
                          paste("log(", names(count.data)[3:22], ")",
                                sep = "", collapse = "+")
                          )
                    )
```

```
par(mfcol = c(2, 2))
fit.obj1 <- glm(form1, family = poisson, data = count.data)
summary(fit.obj1)
```

```
##
## Call:
## glm(formula = form1, family = poisson, data = count.data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -4.503  -1.225  -0.152   0.982  23.250
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   16.685      0.991   16.84  < 2e-16 ***
## log(A)         4.506      0.192   23.47  < 2e-16 ***
## log(C)         1.158      0.177    6.55  5.9e-11 ***
## log(G)         1.479      0.182    8.12  4.6e-16 ***
## log(T)         2.267      0.203   11.15  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 5853.0  on 399  degrees of freedom
## Residual deviance: 1715.7  on 395  degrees of freedom
## AIC: 3890
##
## Number of Fisher Scoring iterations: 4
```

```
plot(fit.obj1)
```

```
fit.obj2 <- glm(form2, family = poisson, data = count.data)
plot(fit.obj2)
fit.obj3 <- glm(form3, family = poisson, data = count.data)
anova(fit.obj1, fit.obj3, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: count ~ log(A) + log(C) + log(G) + log(T)
## Model 2: count ~ log(A) + log(C) + log(G) + log(T) + log(AA) + log(AC) +
##     log(AG) + log(AT) + log(CA) + log(CC) + log(CG) + log(CT) +
##     log(GA) + log(GC) + log(GG) + log(GT) + log(TA) + log(TC) +
##     log(TG) + log(TT)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       395       1716
## 2       379        724 16      991   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can try a simpler model – trying to remove the highly correlated "symmetric" di-

nucleotide frequencies.

```
fit.obj5 <- update(fit.obj2, . ~ . - log(CA) - log(GA) - log(TA) -
                     log(GC) - log(TC) - log(TG))
anova(fit.obj5, fit.obj2, test = "Chisq")

## Analysis of Deviance Table
##
## Model 1: count ~ log(AA) + log(AC) + log(AG) + log(AT) + log(CC) + log(CG) +
##     log(CT) + log(GG) + log(GT) + log(TT)
## Model 2: count ~ log(AA) + log(AC) + log(AG) + log(AT) + log(CA) + log(CC) +
##     log(CG) + log(CT) + log(GA) + log(GC) + log(GG) + log(GT) +
##     log(TA) + log(TC) + log(TG) + log(TT)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       389       1021
## 2       383        768  6      252   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A highly significant difference. Thus removing all these term gives a significantly worse fit.

```
fit.obj3 <- glm(form1, family = quasipoisson, data = count.data)
fit.obj4 <- glm(form3, family = quasipoisson, data = count.data)
anova(fit.obj3, fit.obj4, test = "Chisq")   ## Same thing as above.

## Analysis of Deviance Table
##
## Model 1: count ~ log(A) + log(C) + log(G) + log(T)
## Model 2: count ~ log(A) + log(C) + log(G) + log(T) + log(AA) + log(AC) +
##     log(AG) + log(AT) + log(CA) + log(CC) + log(CG) + log(CT) +
##     log(GA) + log(GC) + log(GG) + log(GT) + log(TA) + log(TC) +
##     log(TG) + log(TT)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       395       1716
## 2       379        724 16      991   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The point in the quasi model is to use the Poisson model as surrogate model for fitting the mean value structure, but the estimate of the "errors" in the model allows for dispersion parameter, which is used in the construction of confidence intervals. Compare

```
confint(fit.obj1)

## Waiting for profiling to be done...

##               2.5 % 97.5 %
## (Intercept) 14.7572 18.641
## log(A)       4.1323  4.885
## log(C)       0.8129  1.506
## log(G)       1.1243  1.838
## log(T)       1.8714  2.668
```

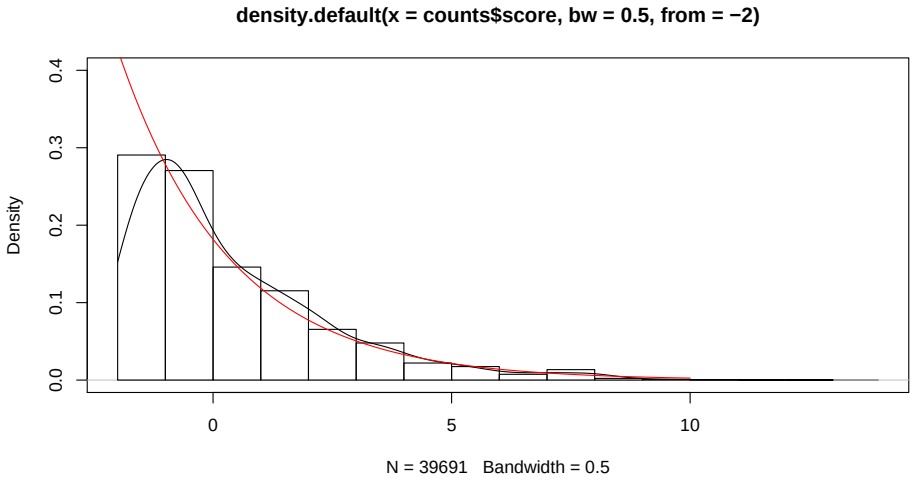
```
confint(fit.obj3)

## Waiting for profiling to be done...

##               2.5 % 97.5 %
## (Intercept) 12.2162 21.304
## log(A)       3.6408  5.402
## log(C)       0.3567  1.979
## log(G)       0.6550  2.326
## log(T)       1.3492  3.214


confint.default(fit.obj1)

##              2.5 % 97.5 %
## (Intercept) 14.743 18.627
## log(A)       4.129  4.882
## log(C)       0.811  1.504
## log(G)       1.122  1.836
## log(T)       1.869  2.665

confint.default(fit.obj3)

##               2.5 % 97.5 %
## (Intercept) 12.1414 21.229
## log(A)       3.6253  5.386
## log(C)       0.3465  1.969
## log(G)       0.6440  2.314
## log(T)       1.3349  3.199
```

Observation 237 seems to be an outlier and should be removed. Try the following data with the methods above.

```
count.data <- select.data(0)[-237, ]
count.data <- select.data(-1)[-237, ]
```

We will also investigate the overshoot distribution. We compare the overshoot distribution with the exponential distribution.

```
plot(density(counts$score, bw = 0.5, from = -2), ylim = c(0, 0.4))
hist(counts$score, prob = TRUE, add = TRUE)
muhat <- mean(counts$score+2)
curve(dexp(x+2, 1 / muhat), -2, 10, col = "red", add = TRUE)
```


density.default(x = counts$score, bw = 0.5, from = −2)
Density
0.0
0.1
0.2
0.3
0.4
0
5
10
N = 39691 Bandwidth = 0.5

Exam 2011/2012

Formalities

This assignment is the final, individual assignment in the course *Statistics BI/E*, 2011/2012. It is handed out Monday 16/1 2012 and the deadline for handing in the solution is Friday 20/1 2012 before 12.00 to Niels Richard Hansen, either in office 04.3.18 or alternatively it can be handed in at the secretariat, room 04.1.03, Building E, HCØ. The assignment must be handed in with an official front page that is made available from the course web page. You confirm with your signature on the front page that you have solved the assignment yourself without help from any others.

The assignment consists of 4 problems with a total of 13 questions.

Data

Two data sets will be used, and both can be downloaded from Absalon and are found together with this assignment. In addition, R commands are given in Problem 1 and Problem 2 below for reading the data into R directly from the internet.

The first data set, `qpcr`, is only used in the first problem. It is an typical (anonymized) raw data set from a quantitative PCR experiment.

The second data set, `sage`, is used in the remaining three problems. The data set is a subset of the data analyzed in the Science paper *Lin Zhang et al.* (1997), Gene Expression Profiles in Normal and Cancer Cells, Science, 276, 1268-72. It is one of the early reported studies where sequencing of short tags was used to quantify gene expression and compare the expression level for many genes between normal and cancer cells. The technique used is called serial analysis of gene expression or SAGE.

Problem 1 (20 %)

We consider the qpcr data set obtainable in R by

```
qpcr <- read.table("http://www.math.ku.dk/~richard/courses/StatScience2011/qpcr.txt")
```

with 3 columns named `intensity`, `cycle`, and `run`. In the experiment a target molecule is amplified in 98 samples (`run`) each time using 45 PCR cycles (`cycle`). The `intensity` is the measured fluorescence intensity, measured for each cycle and each run.

Question 1.1. *Make a scatter plot of the measured intensity against cycle. Restrict the data set to cycles 28 to 32 and make a scatter plot of log intensity against cycles.*

Question 1.2. *Restrict the data set to cycles 28 to 32. Fit a linear regression model of the intensity regressed on the cycle number and fit a linear regression model of log intensity regressed on cycle number. Investigate how well the two models fit the data. Which model is to be preferred?*

As the scatter plot from Question 1.1 shows, the relation between measured intensity and cycle number is highly non-linear. If Y_i denotes the intensity and x_i the cycle number we

will therefore consider the non-linear four parameter logistic regression model

$$Y_i = \beta_0 + \frac{\beta_3 - \beta_0}{1 + \exp\left(\frac{\beta_1 - x_i}{\beta_2}\right)} + \sigma\epsilon_i.$$

Question 1.3. *Fit the four parameter logistic regression model to the data. Can any of the parameters be taken equal to 0? Investigate how well the model fits the data.*

Problem 2 (20%)

For this and the subsequent Problems 3 and 4 we consider the `sage` data, which can be read into R by

```
sage <- read.table("http://www.math.ku.dk/~richard/courses/StatScience2011/sage.txt")
```

It is a data frame with 2456 rows and 4 columns. The columns correspond to 4 different samples, the two first, `NC1` and `NC2`, from normal cells and the two last, `Tu98` and `Tu102`, from tumor cells. The row names are *tags* – short DNA sequences – and for each tag the columns hold the number of times this tag was sequenced in each of the four samples. We let x_i^{NC1}, x_i^{NC2}, x_i^{Tu98} and x_i^{Tu102} denote the 4 counts for the i'th tag. Throughout the assignment it can be assumed that the counts are *independent* for all tags and all four samples as well.

Define the *log-fold-change* for the i'th tag as

$$y_i = \log\left(\frac{x_i^{\text{NC1}} + x_i^{\text{NC2}} + 1}{x_i^{\text{Tu98}} + x_i^{\text{Tu102}} + 1}\right)$$

where we add 1 to both denominator and numerator to avoid problems with dividing by 0 and taking the logarithm of 0. A simple model of y_i is a normal distribution, $N(\mu_i, \sigma^2)$, that is, for the i'th tag the log-fold-change follows a normal distribution with mean μ_i and a common variance σ^2 for all tags.

We investigate a simple minded hypothesis;

$$H : \mu_i = 0 \text{ for all } i.$$

Question 2.1. *Interpret the hypothesis H, estimate the parameter σ^2 under the hypothesis H and compute, under the hypothesis and based on the estimated σ^2, the probability of observing a log-fold-change larger than 2.3 in absolute value. Compare this probability with the data.*

Remark: *A log-fold-change (natural logarithm) larger than 2.3 in absolute value is the same as a fold-change larger than 10 or smaller than 1/10.*

Question 2.2. *Test the hypothesis H. Remember to check and comment on the model assumptions.*

The hypothesis above was called simple minded, because in reality our working hypothesis is that for some of the tags $\mu_i \neq 0$ while for other tags $\mu_i = 0$. The objective is to find those tags with a true difference between the normal and the tumor samples.

Problem 3 (40%)

The raw SAGE data are count data, that is, the values are non-negative integers. This fact was ignored in the initial approach in Problem 2 above where the log-fold-change could just as well have been based on continuous measurements of the abundances of the tags.

Here we will analyze a specific count model which has a certain relation between the mean and variance. It is the probability distribution on the non-negative integers with parameters $\nu \geq 0$ and $\rho > 0$ defined by the point probabilities

$$p(x) = G(x, \rho) \left(\frac{\rho}{\rho + \nu}\right)^{\rho} \left(\frac{\nu}{\rho + \nu}\right)^{x} \tag{3.2}$$

for $x = 0, 1, 2, \ldots$. The G function is discussed below. It can be shown that for this model the mean value is ν and the variance is $\nu + \nu^2/\rho$, thus the variance is a quadratic function of the mean. The ν parameter is, naturally, called the *mean value parameter*, and we call ρ the *inverse dispersion parameter*. You will not explicitly need these facts, but they are important for interpretations of the model. In particular, we will investigate hypotheses about the parameter ν, that is, about the mean value of the distribution.

You will not need to handle the G function analytically, but you will need to be able to compute the function $\log G$, which can be computed in R using

```
logG <- function(x, rho) {
  lgamma(x + rho) - lgamma(x + 1) - lgamma(rho)
}
```

You can also use the fact that $\log G(0, \rho) = 0$ for all $\rho > 0$ without further comments.

We assume that we have a data set consisting of n independent and identically distributed variables $x_1, \ldots, x_n$ with a distribution given by (3.2). Let

$$\bar{x} = \frac{1}{n} \sum_{j=1}^{n} x_j$$

denote the empirical mean. You should think of x as a vector of n sample counts for a given tag.

Question 3.1. *Find the minus-log-likelihood function, show that for fixed ρ it is minimized as a function of ν for*

$$\hat{\nu} = \bar{x}$$

and argue that this is the MLE of ν.

N.B. Pay attention to the case where $\bar{x} = 0$.

Question 3.2. *Show that the profile minus-log-likelihood function for ρ is*

$$l_x(\bar{x}, \rho) = \begin{cases} n(\rho + \bar{x}) \log(\rho + \bar{x}) - n\rho \log \rho - \sum_{j=1}^{n} \log G(x_j, \rho) - n\bar{x} \log \bar{x} & \text{if } \bar{x} > 0 \\ 0 & \text{if } \bar{x} = 0 \end{cases}$$

Plot the profile minus-log-likelihood for the tag `GTTGTGGTTA` *in the SAGE data, that is, where the count data x is the vector with the four counts* $104, 153, 51, 103$.

Question 3.3. *Find the MLE of ρ for the tag* `GTTGTGGTTA` *and compute a 95% confidence interval for ρ.*

The computations above are made under the implicit assumption that the same model (3.2) with parameters $\nu \geq 0$ and $\rho > 0$ applies to the count data for the two normal cells as well as the two tumor cells for the particular tag considered. Consider the extended model where ν^{NC} and ν^{Tu} are mean value parameters specific to the normal cells and the tumor cells, respectively, for the tag `GTTGTGGTTA`. We will assume that ρ is the same for normal cells and tumor cells. Thus we have an extended model with three parameters $\nu^{\mathrm{NC}} \geq 0$, $\nu^{\mathrm{Tu}} \geq 0$ and $\rho > 0$, and the hypothesis

$$H : \nu^{\mathrm{NC}} = \nu^{\mathrm{Tu}}$$

corresponds to the model used above with the same ν for the counts for normal cells and tumor cells.

Question 3.4. *With $x^{NC} = (104, 153)$ the counts for the normal cells and $x^{Tu} = (51, 103)$ the counts for the tumor cells, and $\hat{\nu}^{NC}$ and $\hat{\nu}^{Tu}$ the corresponding MLE of ν for the two groups, that is, the averages of the counts within each group, show that the profile minus-log-likelihood for ρ is*

$$l_{x^{NC}}(\hat{\nu}^{NC}, \rho) + l_{x^{Tu}}(\hat{\nu}^{Tu}, \rho).$$

Compute the MLE of ρ and test the hypothesis H above.

We expand the model from the specific tag considered to all tags. Thus for the i'th tag we have count data x_i^{NC} for the normal cells and x_i^{Tu} for the tumor cells and we have mean value parameters $\nu_i^{\mathrm{NC}} \geq 0$ and $\nu_i^{\mathrm{Tu}} \geq 0$. Moreover, we assume a common $\rho > 0$ for all tags and normal as well as tumor cells. We are interested in the hypotheses

$$H_i : \nu_i^{\mathrm{NC}} = \nu_i^{\mathrm{Tu}}$$

for all tags.

Question 3.5. *Argue that under the model above the profile minus-log-likelihood for ρ is*

$$\sum_{i=1}^{2456} l_{x_i^{NC}}(\hat{\nu}_i^{NC}, \rho) + l_{x_i^{Tu}}(\hat{\nu}_i^{Tu}, \rho).$$

Compute the MLE, $\hat{\rho}$, of ρ and compute a confidence interval for ρ. Compare with the result from Question 3.3.

Question 3.6. *Compute for all 2456 tags the $-2 \log Q$ and corresponding p-value for testing the hypotheses H_i. Report the tags with a p-value below 10^{-12} ordered according to p-value.*

Hint: In principle, you need to recompute the estimate of ρ and the entire minus-log-likelihood under each of the hypotheses. However, the changes in ρ will be small, and you can avoid excessive computations by fixing ρ as the MLE $\hat{\rho}$ obtained in Question 3.5. Then for the i'th hypothesis for the i'th tag

$$-2 \log Q \simeq 2(l_{x_i}(\bar{x}_i, \hat{\rho}) - l_{x_i^{\mathrm{NC}}}(\hat{\nu}_i^{\mathrm{NC}}, \hat{\rho}) - l_{x_i^{\mathrm{Tu}}}(\hat{\nu}_i^{\mathrm{Tu}}, \hat{\rho}))$$

and you can use this approximation without further comments.

Remark: *For the interested, the function $G(x, \rho)$ can be expressed as*

$$G(x, \rho) = \frac{(x - 1 + \rho)^{(x)}}{x!}$$

where $x! = x \times (x - 1) \ldots \times 2 \times 1$ is x factorial and

$$(x - 1 + \rho)^{(x)} = (x - 1 + \rho) \times (x - 2 + \rho) \ldots \times \rho$$

is the x'th decreasing factorial. These quantities can also be expressed in terms of the Γ function, and this was used in the R function `logG` *given above. The distribution is usually called the negative binomial distribution, but be warned, the parametrization in terms of the mean, ν, and the inverse dispersion parameter, ρ, is non-standard! If you look up the negative binomial distribution you will most likely find it in a different parametrization.*

Problem 4 (20 %)

We will investigate the variability in the SAGE data a little further. For this, it will be the working hypothesis that there are no differences between the normal and tumor cells for all tags. That is most likely not the case, but the method is reasonable if it is true for most tags. If ν_i and σ_i^2 denote the mean and variance for the i'th tag we will investigate if the following approximate relation

$$\sigma_i^2 \simeq \alpha \nu_i^\beta. \tag{4.3}$$

holds for parameters α and β independent of the tag.

Question 4.1. *Compute for each tag the empirical mean $\hat{\nu}_i$ and the empirical variance $\hat{\sigma}_i^2$. Plot the log empirical variance against the log empirical mean for all tags. Does the plot support the relation (4.3)?*

Question 4.2. *Fit a linear regression model of $\log(\hat{\sigma}_i^2)$ against $\log(\hat{\nu}_i)$. Interpret the fitted parameters in relation to the α and β parameters in (4.3). Interpret and test the hypothesis that the slope parameter in the linear regression model is equal to 1.*

Exam 2010/2011

For this exam you will need the populations data set, which is available here:

`http://www.math.ku.dk/~richard/courses/StatScience2011/populations.txt`

Final Assignment

Statistics BI/E, 2010/2011

Formalities

This assignment is the final, individual assignment in the course *Statistics BI/E*, 2010/2011. It is handed out Wednesday 19/1 2010 and the deadline for handing in the solution is Wednesday 26/1 2010 before 15.15 to Jessica Kasza, either in office 04.3.24 or alternatively it can be handed in at the secretariat, room 04.1.03, Building E, HCØ. The assignment must be handed in with an official front page that is made available from the course web page on Absalon. You confirm with your signature on the front page that you have solved the assignment yourself without help from any others.

Problem 1 (55%)

In this problem we consider the loblolly tree data, which you can load into your R workspace using the command `data(Loblolly)`. We first consider the first two columns of the data set, which consist of 84 measurements of heights in feet and ages in years of loblolly pine trees. We would like to model the growth of the trees: how does the height of the tree change as the tree ages?

Let X be a random variable representing the height of trees measured in feet, and y be tree ages in years. We assume

$$X_i = g_\beta(y_i) + \sigma\varepsilon_i, \;\; \varepsilon_i \sim N(0,1), \;\; i = 1, \ldots, 84.$$

Two commonly considered models for growth are

- the logistic model:
$$g^l_\beta(y_i) = \frac{\beta^l_0}{1 + \exp\left(\frac{\beta^l_1 - y_i}{\beta^l_2}\right)}; \tag{1.1}$$
- the asymptotic regression model, which is an alternative to the logistic regression model:
$$g^a_\beta(y_i) = \beta_0 + (\beta_1 - \beta_0)\exp\left(-\exp(\beta_2)y_i\right). \tag{1.2}$$

Question 1.1. Calculate the limit of each of $g^l_\beta(y_i)$ and $g^a_\beta(y_i)$ as $y_i \to \infty$, and use your result to interpret the parameters β^l_0 and β_0.

Question 1.2. Fit both the logistic and asymptotic regression models to the loblolly tree data, using non-linear least squares. Make a scatterplot of the data, and, using the `lines` command, add both of the fitted lines to the plot.

Hint: Typically when we use non-linear least squares to fit a regression model, you need to provide starting points for each of the parameters in the model. The `SSlogis` and `SSasymp` functions in R allow you to fit a logistic or asymptotic regression model without the need to specify starting points. You can use a command like

```
my.model <- nls(my.response ~ SSlogis(my.covariate, beta0, beta1, beta2))
```

Question 1.3. For each model, plot the residuals versus the fitted values, and a normal quantile-quantile plot of the residuals. Using these plots, assess the regression assumptions.

Question 1.4. Which of the two models do you think is a better fit for this data set? Why? Are all of the regression assumptions satisfied for this model?

The loblolly data set consists of growth data for 14 individual loblolly trees. The third column of the data set contains information on the seed number of each of these 14 trees. We now consider the data associated with the trees that were tallest and shortest when they were 25 years old. The shortest tree has seed number 329, and the tallest tree has seed number 305. We are interested in the difference between the asymptotic heights of these trees: the heights these trees will approach as they get very old.

Question 1.5. If β^s_0 is β_0 for the shortest tree and β^t_0 is β_0 for the tallest tree, where β_0 is as in Equation (1.2), estimate β^s_0 and β^t_0. Our parameter of interest is $\tau = \beta^t_0 - \beta^s_0$. Estimate τ.

Question 1.6. Using either parametric or non-parametric bootstrapping, obtain 1000 estimates of τ. Based on these 1000 values of $\hat{\tau}$, estimate a 95% confidence interval for τ. Test the null hypothesis $H_0 : \tau = 0$ at level $\alpha = 0.05$. What do you conclude about the difference between the asymptotic heights of these two trees?

Question 1.7. Suppose we want to perform hypothesis tests to compare the asymptotic heights of each pair of trees. If we do this, how many hypothesis tests will we conduct? If there is no difference in the asymptotic heights of the trees, and we test them at level $\alpha = 0.10$, what is the probability that we incorrectly reject at least one of our null hypotheses?

Problem 2 (45%)

Many quantities of interest are thought to be distributed according to power-law distributions. These distributions are useful for describing situations where large observations are thought to occur with non-negligible probability. For example, in some protein-protein

interaction networks, it is thought that the number of proteins that each protein is related to follows a power-law distribution: most proteins are related to few other proteins, but there are also so-called "hub" proteins, that are related to a comparitively large number of other proteins. Other quantities are thought to be distributed in this way include populations of cities: there are lots of small cities, but a few very large ones (New York, London, etc.), and intensities of earthquakes (as measured by the Richter scale): most quakes have very small intensities, but earthquakes with very large intensities do occur.

The continuous power law distribution function is given by

$$F(x) = 1 - x^{-\rho+1}, \;\; x > 1, \tag{2.3}$$

where $\rho > 1$ is a parameter, known as the scaling parameter.

Question 2.1. What three conditions must be satisfied for $F(x)$ to be a valid distribution function? Show that the corresponding density function is given by

$$f(x) = (\rho - 1)x^{-\rho}.$$

Question 2.2. Plot, on the same figure, $f(x)$ for $\rho = 1.5, 3, 5$ and 10, for $x \in (1, 10)$. Calculate $P(X > 7)$ for each of these values of ρ. What happens to this probability as ρ increases?

Suppose X has a power-law distribution with scale parameter ρ.

Question 2.3. Find $\mathbb{E}X$ and $\mathbb{V}X$. Find an expression for the q-quantile of X in terms of ρ, and use this expression to write down formulae for the median and interquartile range of X.

Question 2.4. Would you prefer to use the expectation and variance or the median and interquartile range to describe the distribution of X? Why?

Suppose we have n random variables $X_1, \ldots, X_n$ that are independent and identically distributed with density function $f(x_i) = (\rho - 1)x_i^{-\rho}$. Hence, the minus log-likelihood function is given by

$$\ell_x(\rho) = -n \log(\rho - 1) + \rho \sum_{i=1}^{n} \log(x_i)$$

Question 2.5. Show that the maximum likelihood estimator of ρ is given by

$$\hat{\rho} = 1 + \frac{n}{\sum_{i=1}^{n} \log(x_i)},$$

and that the inverse of the Fisher information is given by

$$i(\rho) = \frac{(\rho - 1)^2}{n}.$$

Question 2.6. Write down the approximate distribution of $\hat{\rho}$.

Question 2.7. Download the population data from Absalon: the file `population.txt` contains the populations of cities in the United States from the 2000 US census. For the purposes of this question, we assume that these populations are iid observations of a continuous random variable with a power-law distribution. Compute the maximum likelihood estimate of ρ given this data. Does the model fit the data? What range of values of ρ are plausible given this data set?