Bio4158 manual

Julien Martin

2024-01-22

Table of contents

Welcome **1**
General points to keep in mind . . . 1
What is R and why use it in this course? . . . 2
Software installation . . . 2
General laboratory instructions . . . 3
Notes about the manual . . . 3

1. Introduction to R **5**
1.1. Packages and data needed for the lab . . . 5
1.2. Importing and exporting data . . . 5
1.3. Preliminary examination of data . . . 8
1.4. Creating data subsets . . . 16
1.5. Data transformation . . . 18
1.6. Exercice . . . 18

2. Power Analysis with R and G*Power **21**
2.1. The theory . . . 21
2.2. What is G*Power? . . . 22
2.3. How to use G*Power . . . 23
2.4. Power analysis for a t-test on two independent means . . . 24
2.5. Important points to remember . . . 35

3. Correlation and simple linear regression **37**
3.1. R packages and data . . . 37
3.2. Scatter plots . . . 38
3.3. Data transformations and the product-moment correlation . . . 40
3.4. Testing the significance of correlations and Bonferroni probabilities . . . 43
3.5. Non-parametric correlations: Spearman's rank and Kendall's τ . . . 44
3.6. Simple linear regression . . . 45
3.7. Data transformations in regression . . . 52
3.8. Dealing with outliers . . . 55
3.9. Quantifying effect size in regression and power analysis . . . 58
3.10. Bootstrapping the simple linear regression . . . 63

4. Two - sample comparisons **67**
4.1. R packages and data . . . 67
4.2. Visual examination of sample data . . . 67
4.3. Comparing means of two independent samples: parametric and non-parametric comparisons 70
4.4. Bootstrap and permutation tests to compare 2 means . . . 74
4.5. Comparing the means of paired samples . . . 77
4.6. Bibliography . . . 82

5. One-way ANOVA **83**
5.1. R packages and data . . . 83
5.2. One-way ANOVA with multiple comparisons . . . 83
5.3. Data transformations and non-parametric ANOVA . . . 95
5.4. Dealing with outliers . . . 97
5.5. Permutation test . . . 98

6. Multiway ANOVA: factorial and nested designs **101**
6.1. R packages and data needed . . . 101
6.2. Two-way factorial design with replication . . . 102
6.3. 2-way factorial ANOVA without replication . . . 111
6.4. Nested designs . . . 114
6.5. Two-way non-parametric ANOVA . . . 118
6.6. Multiple comparisons . . . 120
6.7. Test de permutation pour l'ANOVA à deux facteurs de classification . . . 127
6.8. Bootstrap for two-way ANOVA . . . 129

7. Multiple regression **131**
7.1. R packages and data . . . 131
7.2. Points to keep in mind . . . 131
7.3. First look at the data . . . 132
7.4. Multiple regression models from scratch . . . 133
7.5. Stepwise multiple regression procedures . . . 138
7.6. Detecting multicollinearity . . . 141
7.7. Polynomial regression . . . 142
7.8. Checking assumptions of a multiple regression model . . . 147
7.9. Visualizing effect size . . . 150
7.10. Testing for interactions . . . 152
7.11. Dredging and the information theoretical approach . . . 156
7.12. Bootstrapping multiple regression . . . 159
7.13. Permutation test . . . 163

8. ANCOVA and general linear model **165**
8.1. R packages and data . . . 165
8.2. Linear models . . . 165
8.3. ANCOVA . . . 166
8.4. Homogeneity of slopes . . . 166
8.5. The ANCOVA model . . . 174
8.6. Comparing model fits . . . 180
8.7. Bootstrap . . . 181
8.8. Permutation test . . . 182

References **185**

Appendices **191**

A. Software Tools **191**
A.1. R and R packages . . . 191
A.2. Pandoc . . . 192
A.3. LaTeX . . . 192

Welcome

The laboratory exercises outlined in the following pages are designed to allow you to develop some expertise in using statistical software (R) to analyze data. R is powerful statistical software but, like all software, it has its limitations. In particular, it is dumb: it cannot think for you, it cannot tell you whether the analysis you are attempting to do is appropriate or even makes any sense, and it cannot interpret your results. The manual was written with the help of Rmarkfdown Xie (2015).

General points to keep in mind

- Before attempting any statistical procedure, you must familiarize yourself with what the procedure is actually doing. This does not mean you actually have to know the underlying mathematics (although this certainly helps!), but you should at least understand the principles involved in the analysis. Therefore, before doing a laboratory exercise, read the appropriate section(s) in the lecture notes. Otherwise, the output from your analyses - even if done correctly - will seem like drivel.

- The laboratories are designed to complement the lectures, and vice versa. Owing to scheduling constraints, it may not be possible to synchronize the two perfectly. But feel free to bring questions about the laboratories to class, or questions about the lectures to the labs.

- Work on the laboratories at your own speed: some can be donemuch more quickly than others, and one laboratory need not correspond to one laboratory session. In fact, for some laboratorieswe have allotted two laboratory sessions. Although you will notbe "graded" on the laboratories per se, be aware that completingthe labs is essential. If you do not complete the labs, it is veryunlikely that you will be able to complete the assignments and thefinal exam/term paper. So take these laboratories seriously!

- The objective of the first lab is to allow you to acquire or reviewthe minimum knowledge required to complete the following laboratory exercises with R. There are always several methods toaccomplish something in R, but you will only find simple ways inthis manual. Those amongst you that want to go further will easilyfind many examples of more detailed and sophisticated methods.In particular, I point you to the following resources:

  - R for beginners http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf
  - An introduction to R http://cran.r-project.org/doc/manuals/R-intro.html
  - If you prefer paper books, the CRAN web site has a commented list at : http://www.r-project.org/doc/bib/R-books.html
  - Excellent list of R books https://www.bigbookofr.com/
  - R reference card by Tom Short http://cran.r-project.org/doc/contrib/Short-refcard.pdf

What is R and why use it in this course?

R is multiplatform free software forming a system for statistical computation and graphics. R is also a programming language specially designed for statistical data analysis. It is a dialect of the S language. S- Plus is another dialect of the S language, very similar to R, incorporated into a commercial package. S-Plus has a built-in graphical design intreface that some find convivial.

R has 2 major advantages for this course. Initially, you will find that it also has one inconvenience. However, this "inconvenience" will rapidly force you to acquire very good working habits. So, I see it as a third advantage.

The first advantage is that you can install it freely on you personal computer(s). This is important because it is by doing analyses that you will learn and eventually master biostatistics. This implies that you have easy and unlimited access to a statistical software package. The second advantage is that R can do everything in statistics. R was conceived to be extensible and has become the preferred tool for statisticians around the world. The question is not "Can R do this?" but rather "How can I do this in R?". And search engine are your friends.

No other software package offers you these two advantages.

The inconvenience of R is that one has to type commands (or copy and paste code) rather than use a menu and select options. If you do not know what command to use, nothing will happen. It is therefore not that easy when you start. However, it is possible to rapidly learn to make basic operations (open a data file, plot data, and run a simple analysis). And once you understand the operating principle, you can easily find examples on the Web for more complex analyses and graphs for which you can adapt the code.

This is exactly what you will do in the first lab to familiarize yourself with R.

Why is this inconvenience really an advantage in my mind? Because this way of doing things is more efficient and will save you time on the long run. I guarantee it. Believe me, you will never do an analysis only once. As you'll proceed through analyses, you will find data entry errors, discover that the analysis must be run separately for subgroups, find extra data, have to rerun the analysis on transformed data, or you will make some analytical error along the way. If you use a graphical interface with menus, redoing an analysis implies that you reclick here, enter values there, select some options, etc. Each of these steps is a potential source of error. If, instead, you use lines of codes, you only have to fix the code and submit to repeat instantaneously the entire analysis. And you can perfectly document what you did, leaving an audit trail for the future. This is how pros work and can document the quality of the results of their analyses.

Software installation

R

To install R on a computer, go to http://cran.r-project.org/. You will find compiled versions (binairies) for your preferred operating system (Windows, MacOS, Linux).

Note : R has already been installed on the lab computers (the version may be slightly different, but this should not matter).

Rstudio or VS code

RStudio and VS code are integrated development environment software or IDE. RStudio was develop specifically to work with R. VScode is more generela but work extremely well with R. Both are available on Windows, OS X and Linux

- RStudio: https://www.rstudio.com/products/rstudio/download/
- VScode: https://code.visualstudio.com/download

R libraries

R is essentially unlimited in terms of functions that can be used, because is relies on functions packages that can be added as extra components to use in R.

- Rmarkdown
- tinytex

Those 2 packages should be installed automatically with RStudio but I recommend to install them manually in case they are not. To do so, just copy-paste the text below in R terminal.

```
install.packages(c("rmarkdown", "tinytex"))
```

General laboratory instructions

- Bring a USB key or equivalent so you can save your work. Alternatively, email your results to yourself.
- Read the lab exercise before coming to the lab. Read the R code and come with questions about the code.
- During pre-labs, listen to the special instructions
- Do the laboratory exercises at your own rhythm, in teams. Then, I recommend that you start (complete?) the lab assignment so that you can benefit from the presence of the TA or prof.
- During your analyses, copy and paste results in a separate document, for example in your preferred word processing program. Annotate abundantly
- Each time you shut down R, save the history of your commands (ex: labo1.1 rHistory, labo1.2.rHistory, etc). You will be able to redo the lab rapidly, get code fragments, or more easily identify errors.
- Create your own "library" of code fragments (snippets). Annotate it abundantly. You will thank yourself later.

Notes about the manual

You will find explanations on the theory, R code and functions, IDE best practice and exercises with R.

The manual tries to highlight some part of the text using the following boxes and icons.

Exercises

Solutions

Warning

warnings

Important

important points

Note

notes

Resources

The manual is based on the previous lab manual *Findlay, Morin and Rundle, BIO4158 Laboratory manual for BIO4158.*

License

The document is available follwoing the license License Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International.

Figure 1.: License Creative Commons

1. Introduction to R

After completing this laboratory exercise, you should be able to:

- Open R data files
- Import rectangular data sets
- Export R data to text files
- Verify that data were imported correctly
- Examine the distribution of a variable
- Examine visually and test for normality of a variable
- Calculate descriptive statistics for a variable
- Transform data

1.1. Packages and data needed for the lab

This labs needs the following:

- R packages:
  - ggplot2
- data files
  - ErablesGatineau.csv
  - sturgeon.csv

1.2. Importing and exporting data

There are multiple format to save data. The 2 most used formats with R are `.csv` and `.Rdata`.

- `.csv` files are used to store data in a simple format and are editable using any text editor (e.g. Word, Writer, atom, …) and spreadsheets (e.g. MS Excel, LO Calc). They can be read using the function `read.csv()` and created in R with `write.csv()`.
- `.Rdata` files are used to store not only data but any R object, however, those files can only be used in R. They are created using the `save()` function and read using the `load()` function.

Data for exercises and labs are provides in `.csv`.

1.2.1. Working directory

> ⚠ Warning
>
> Potentially the most frequent error when starting with R is link to loading data or reading data from an external file in R.

A typical error message is:

```
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'ou_est_mon_fichier.csv': No such file or directory
```

This type of error simply means that R cannot find the file you specified. By default, when R starts, a folder is defined as the base folder for R. This is the working directory. R by default will save any files in this folder and will start looking for files in this folder. So you need to specify to R where to look for files and where to save your files. This can be done in 3 different ways:

1. `file.choose()`. (not recommended, because not reproducible). This function will open a dialog box allowing you to click on the file you want. This is not recommended and can be long because you will have to do it absolutely every time you use R.
2. specify the complete path in the function. For example `read.csv("/home/julien/Documents/cours/BIO4558/`
   This is longer to type the first time and a bit tricky to get the correct path but after you can run the line of code and it works every time without trying to remember were you saved that damned file. However, this is specific to your own computer and would not work elsewhere.
3. specify a working directory with `setwd()`. This simplify tells R where to look for files and where to save files. (This is automatically done when using .Rmd files). Just set the working directory to where you want and after that all path will be relative to this working directory. The big advantage is that if you keep a similar folder structure for you R project it will be compatible and reproducible across all computer and OS

To know which folder is the workind directory simply type `getwd()`

> ℹ Note
>
> When opening Rstudio by double-clicking on a file, it will automatically set the working irectory to the folder where this file is located. This can be super handy.

> ❗ Important
>
> For all labs, I strongly recommend you to make a folder where you will save all your R scripts and data and use it as your working directory in R. For better organisation I suggest to save your data in a subfolder named `data All R code for data loading in the manual is based on that structure. This is why dat loading or saving code look like`data/my_file.xxx‘. If you follow it also all code for data loading can be simply copy-pasted and should work.

1.2.2. Opening a `.Rdata` file

You can double-click on the file and R/Rstudio should open. Alternatively, you can use `load()` function and specify the names (and path) of the file. For example to load the data `ErablesGatineau.Rdata` in R which is located in the folder `data` in the working directory you can use:

```
load("data/ErablesGatineau.Rdata")
```

1.2.3. Open a `.csv` file

To import data saved in a `.csv` file, you need to use the `read.csv()` function. For example, to create a R object named `erables` which contain the data from the file `ErablesGatineau.csv`, you need to use:

```
erables <- read.csv("data/ErablesGatineau.csv")
```

> **Warning**
>
> Beware of the coma. If you are working in adifferent language (other than english), be careful because the decimal symbol might ot be the same. By default R use the point for the decimal sign. If the dat use the coma for the decimal then R would not be able to read the file correctly. In this case you can use `read.csv2()` or `read.data()` which should solve the problem.

To verify that the data were read and loaded properly, you can list all objects in memory with the `ls()` function, or get a more detailed description with `ls.str()`:

> **Note**
>
> I do not recommend to use `ls.str()` since it can produce really long R ouputs when you have multiple R object loaded. I suggest instead to use the combination of `ls()` to get the list of all R objects and then `str()` only for the objects you want to look at.

```
ls()
```

```
[1] "erables"
```

```
str(erables)
```

```
'data.frame':   100 obs. of  3 variables:
 $ station: chr  "A" "A" "A" "A" ...
 $ diam   : num  22.4 36.1 44.4 24.6 17.7 ...
 $ biom   : num  732 1171 673 1552 504 ...
```

R confirms that the object `erables`. `erables` is a data.frame that contains 100 observations (lines) of 3 variables (columns) : `station` , a variable of type Factor with 2 levels, and `diam` and `biom` that are 2 numeric variables.

1.2.4. Entering data in R

R is not the ideal environment to input data. It is possible, but the syntax is heavy and makes most people upset. Use your preferred worksheet program instead. It will be more efficient and less frustrating.

1.2.5. Cleaning up / correcting data

Another operation that can be frustrating in R. Our advice: unless you want to keep track of all corrections made (so that you can go back to the original data), do not change data in R. Return to the original data file (in a worksheet or database), correct the data there, and then reimport into R. It is simple to resubmit the few lines of code to reimport data. Doing things this way will leave you with a single version of your data file that has all corrections, and the code that allows you to repeat the analysis exactly.

1.2.6. Exporting data from R

You have 2 options: export data in `.csv` or in `.Rdata`

To export in `.Rdata` use the function `save()` to export in `.csv` use `write.csv()`

For example, to save teh object `mydata` in a file `wonderful_data.csv`that will be saved in your working directory you can type:

```
write.csv(mydata, file = "wonderful_data.csv", row.names = FALSE)
```

1.3. Preliminary examination of data

The first step of data analysis is to examine the data at hand. This examination will tell you if the data were correctly imported, whether the numbers are credible, whether all data came in, etc. This initial data examination often will allow you to detect unlikely observations, possibly due to errors at the data entry stage. Finally, the initial plotting of the data will allow you to visualize the major trends that will be confirmed later by your statistical analysis.

The file `sturgeon.csv` contains data on sturgeons from the Saskatchewan River. These data were collected to examine how sturgeon size varies among sexes (`sex`), sites (`location`), and years(`year`).

- Load the data from `sturgeon.csv` in a R object named `sturgeon`.
- use the function `str()` to check that the data was loaded and read correctly.

```
sturgeon <- read.csv("data/sturgeon.csv")
str(sturgeon)
```

```
'data.frame':   186 obs. of  9 variables:
 $ fklngth : num  37 50.2 28.9 50.2 45.6 ...
 $ totlngth: num  40.7 54.1 31.3 53.1 49.5 ...
 $ drlngth : num  23.6 31.5 17.3 32.3 32.1 ...
 $ rdwght  : num  15.95 NA 6.49 NA 29.92 ...
 $ age     : int  11 24 7 23 20 23 20 7 23 19 ...
 $ girth   : num  40.5 53.5 31 52.5 50 54.2 48 28.5 44 39 ...
 $ sex     : chr  "MALE" "FEMALE" "MALE" "FEMALE" ...
 $ location: chr  "THE_PAS" "THE_PAS" "THE_PAS" "THE_PAS" ...
 $ year    : int  1978 1978 1978 1978 1978 1978 1978 1978 1978 1978 ...
```

1.3.1. Summary statistics

To get summary statistics on the contents of the data frame sturgeon, type the command:

```
summary(sturgeon)
```

```
    fklngth         totlngth        drlngth          rdwght     
 Min.   :24.96   Min.   :28.15   Min.   :14.33   Min.   : 4.73  
 1st Qu.:41.00   1st Qu.:43.66   1st Qu.:25.00   1st Qu.:18.09  
 Median :44.06   Median :47.32   Median :27.00   Median :23.10  
 Mean   :44.15   Mean   :47.45   Mean   :27.29   Mean   :24.87  
 3rd Qu.:48.00   3rd Qu.:51.97   3rd Qu.:29.72   3rd Qu.:30.27  
 Max.   :66.85   Max.   :72.05   Max.   :41.93   Max.   :93.72  
                 NA's   :85      NA's   :13      NA's   :4      
      age            girth           sex              location        
 Min.   : 7.00   Min.   :11.50   Length:186         Length:186        
 1st Qu.:17.00   1st Qu.:40.00   Class :character   Class :character  
 Median :20.00   Median :44.00   Mode  :character   Mode  :character  
 Mean   :20.24   Mean   :44.33                                        
 3rd Qu.:23.50   3rd Qu.:48.80                                        
 Max.   :55.00   Max.   :73.70                                        
 NA's   :11      NA's   :85                                           
      year     
 Min.   :1978  
 1st Qu.:1979  
 Median :1979  
 Mean   :1979  
 3rd Qu.:1980  
 Max.   :1980  
```

For each variable, R lists:

- the minimum
- the maximum
- the median that is the 50^{th} percentile, here the 93^{rd} value of the 186 observations ordered in ascending order

- values at the first (25%) and third quartile (75%)
- the number of missing values in the column.

Note that several variables have missing values (NA). Only the variables `fklngth` (fork length), `sex` , `location` , and `year` have 186 observations.

> ⚠ Warning
>
> **Beware of missing values**
> Several R functions are sensitive to missing values and you will frequently have to do your analyses on data subsets without missing data, or by using optional parameters in various commands. We will get back to this, but you should always pay attention and take note of missing data when you do analyses.

1.3.2. Histogram, empirical probability density, boxplot, and visual assessment of normality

Let's look more closely at the distribution of `fklngth`. The command `hist()` will create a histogram. For the histogram of `fklngth` in the `sturgeon` data frame, type the command:

```
hist(sturgeon$fklngth)
```

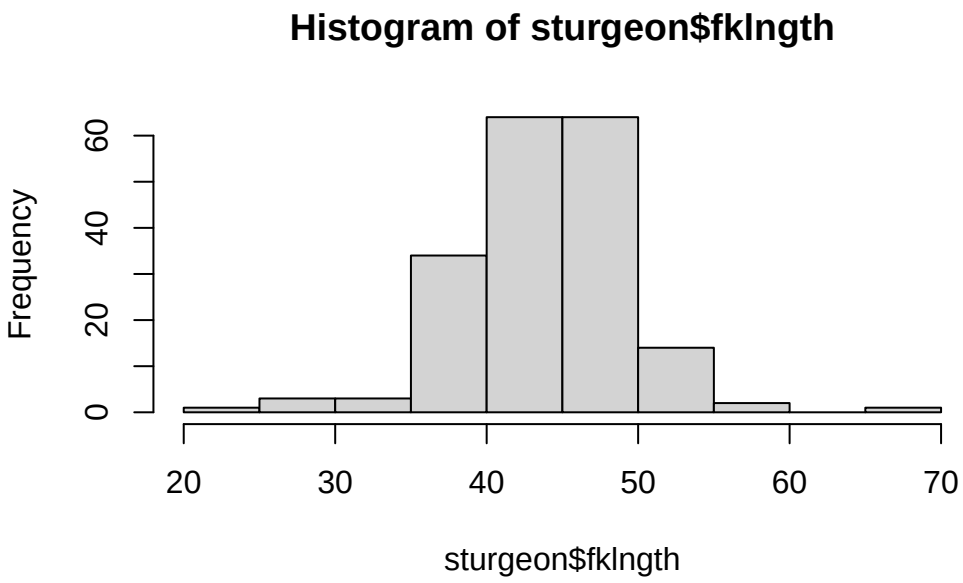


Figure 1.1.: Histogram of fluke length of sturgeons

The data appear to be approximately normal. This is good to know.

> **Note**
>
> Note that this syntax is a bit heavy as you need to prefix variable names by the data frame name `sturgeon$`. You can lighten the syntax by making the variables directly accessible by commands by typing the command `attach()`. However, I **strongly recommend not to use** it because it can lead to many problems hard to detect compare to the little benfit is provides

This histogram (Fig. @ref(fig:hist-stur)) is a very classical representation of the distribution. Histograms are not perfect however because their shape partly depends on the number of bins used, more so for small samples. One can do better, especially if you want to visually compare the observed distribution to a normal distribution. But you need to come up with a bit of extra R code based on the `ggplot2` .

```
## load ggplot2 if needed
library(ggplot2)

## use "sturgeon" dataframe to make plot called mygraph
# and define x axis as representing fklngth
mygraph <- ggplot(data = sturgeon, aes(x = fklngth))

## add data to the mygraph ggplot
mygraph <- mygraph +
  ## add semitransparent histogram
  geom_histogram(aes(y = ..density..),
    bins = 30, color = "black", alpha = 0.3
  ) +
  ##  add density smooth
  geom_density() +
  ## add observations positions or rug bars
  geom_rug() +
  ## add Gaussian curve adjusted to the data with mean and sd from fklngth
  stat_function(
    fun = dnorm,
    args = list(
      mean = mean(sturgeon$fklngth),
      sd = sd(sturgeon$fklngth)
    ),
    color = "red"
  )

## display graph
mygraph
```

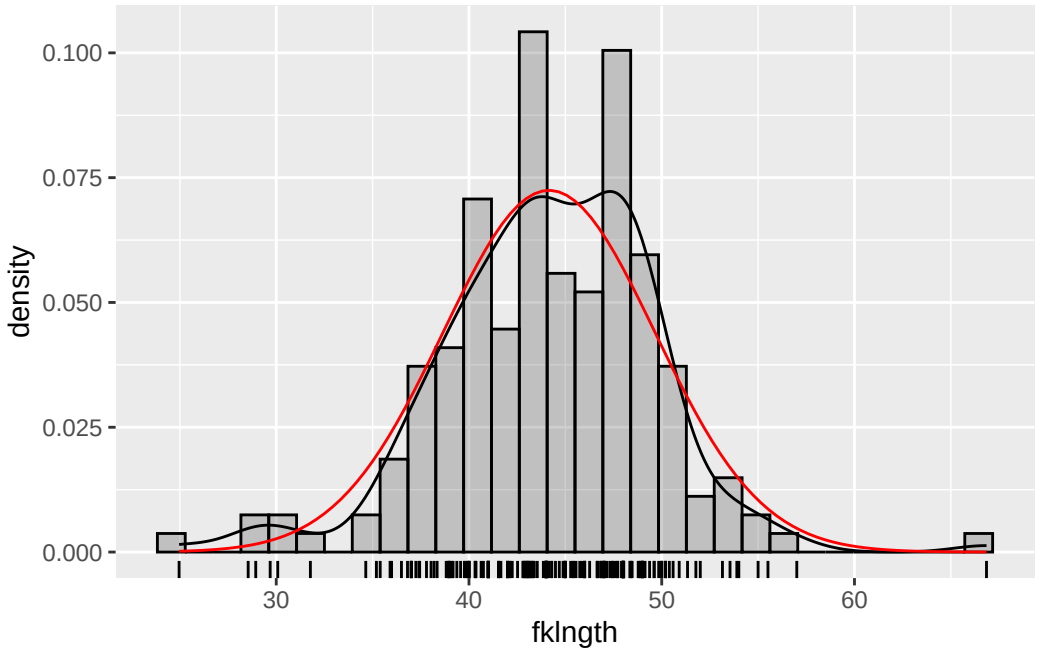
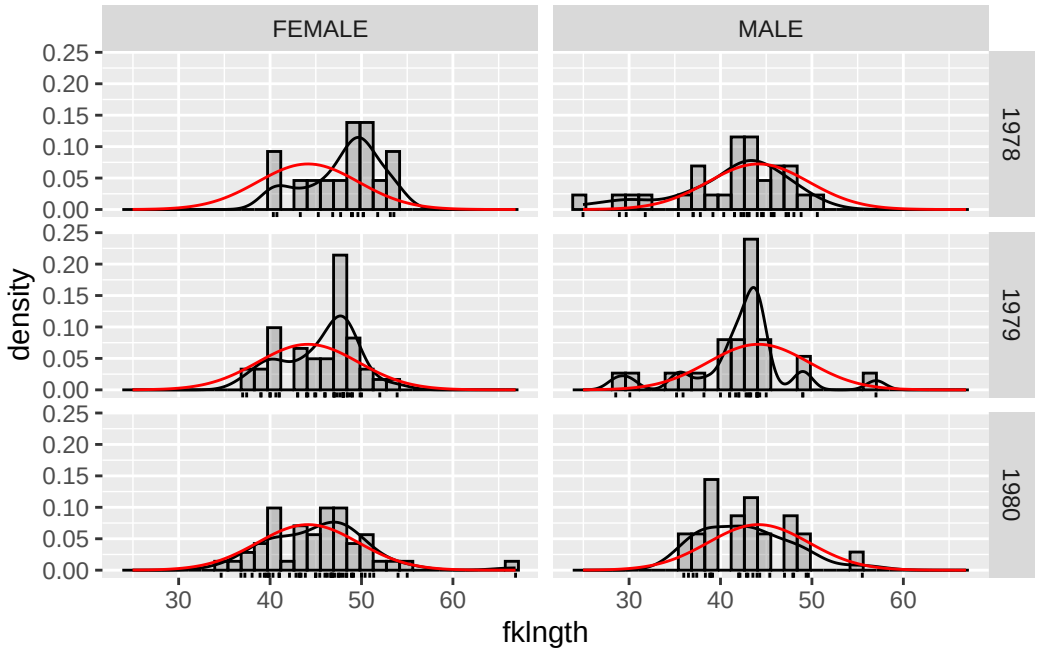


Figure 1.2.: Distribution of fluke length in sturgeon plotted with ggplot

Each observation is represented by a short vertical bar below the x- axis (rug). The red line is the normal distribution with the same mean and standard deviation as the data. The other line is the empirical distribution, smoothed from the observations.

The ggplot object you just created (`mygraph`) can be further manipulated. For example, you can plot the distribution of `fklngth` per `sex` and `year` groups simply by adding a `facet_grid()` statement:

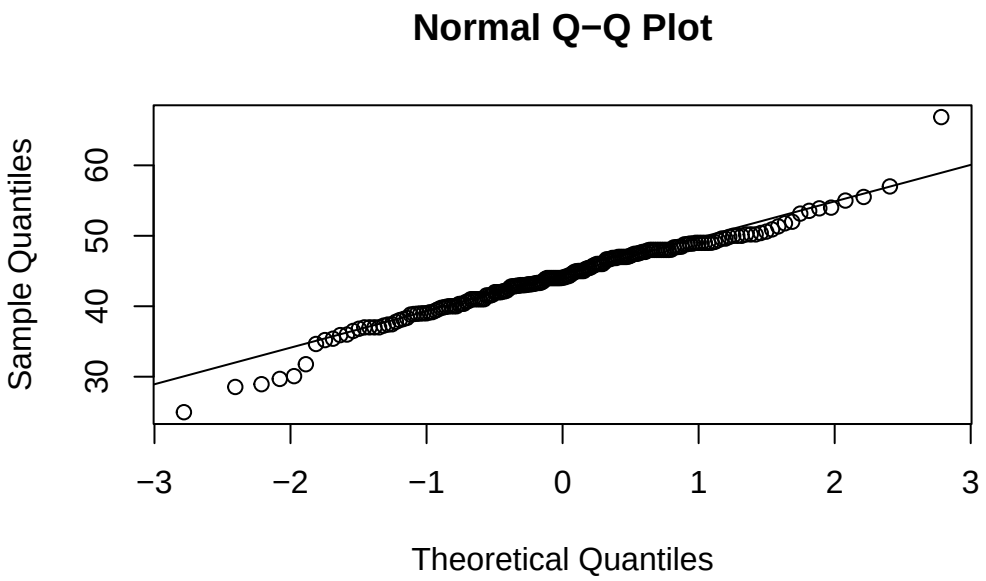
```
mygraph + facet_grid(year ~ sex)
```



Each panel contains the data distribution for one sex that year, and the recurring red curve is the normal distribution for the entire data set. It can serve as a reference to help visually evaluate differences among panels.

Another way to visually assess normality of data is the QQ plot that is obtained by the pair of commands `qqnorm()` and `qqline()`.

```
qqnorm(sturgeon$fklngth)
qqline(sturgeon$fklngth)
```



Perfectly normal data would follow the straight diagonal line. Here there are deviations in the tails of the distribution and a bit to the right of the center. Compare this representation to the two preceding graphs. You will probably agree that it is easier to visualize how data deviate from normality by looking at a histogram of an empirical probability density than by looking at the QQ plots. However, QQ plots are often automatically produced by various statistical routines and you should be able to interpret them. In addition, one can easily run a formal test of normality in R with the command `shapiro.test()` that computes a statistic (`W`) that measures how tightly data fall around the straight diagonal line of the QQ plot. If data fall perfectly on the line, then `W = 1`. If `W` is much less than 1, then data are not normal.

For the `fklngth` data:

```
shapiro.test(sturgeon$fklngth)
```

```
    Shapiro-Wilk normality test

data:  sturgeon$fklngth
W = 0.97225, p-value = 0.0009285
```

W is close to 1, but far enough to indicate a statistically significant deviation from normality.

Visual examination of very large data sets is often made difficult by the superposition of data points. Boxplots are an interesting alternative. The command `boxplot(fklngth~sex, notch=TRUE)` produces a boxplot of `fklngth` for each `sex` , and adds whiskers.

```
boxplot(fklngth ~ sex, data = sturgeon, notch = TRUE)
```

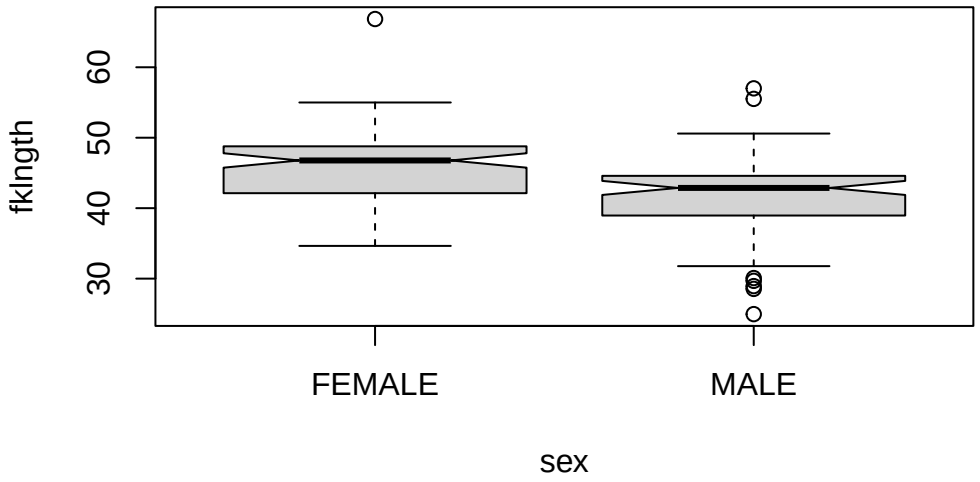


Figure 1.3.: Boxplot of fluke length in strugeon by sex

The slightly thicker line inside the box of figure @ref(fig:boxplot-stur) indicates the median. The width of the notch is proportional to the uncertainty around the median estimate. One can visually assess the approximate statistical significance of differences among medians by looking at the overlap of the notches (here there is no overlap and one could tentatively conclude that the median female size is larger than the median male size). Boxes extend from the first to third quartile (the 25^{th} to 75^{th} percentile if you prefer). Bars (whiskers) extend above and below the boxes from the minimum to the maximum observed value or, if there are extreme values, from the smallest to the largest observed value within 1.5x the interquartile range from the median. Observations exceeding the limits of the whiskers (hence further away from the median than 1.5x the interquartile range, the range between the 25^{th} and 75^{th} percentile) are plotted as circles. These are outliers, possibly aberrant data.

1.3.3. Scatterplots

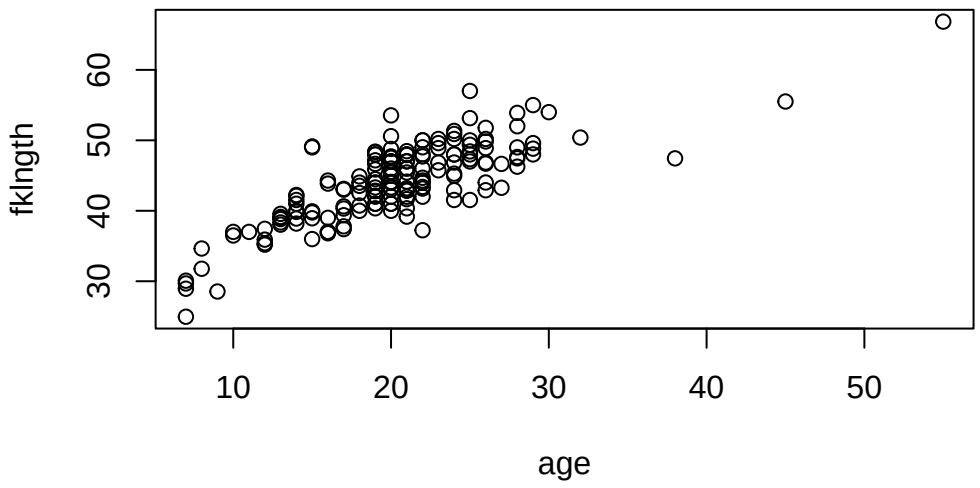
In addition to histograms and other univariate plots, it is often informative to examine scatter plots. The command `plot(y~x)` produces a scatter plot of y on the vertical axis (the ordinate) vs x on the horizontal axis (abscissa).

Exercise

Create a scatterplot of fklngth vs age using the plot() command.

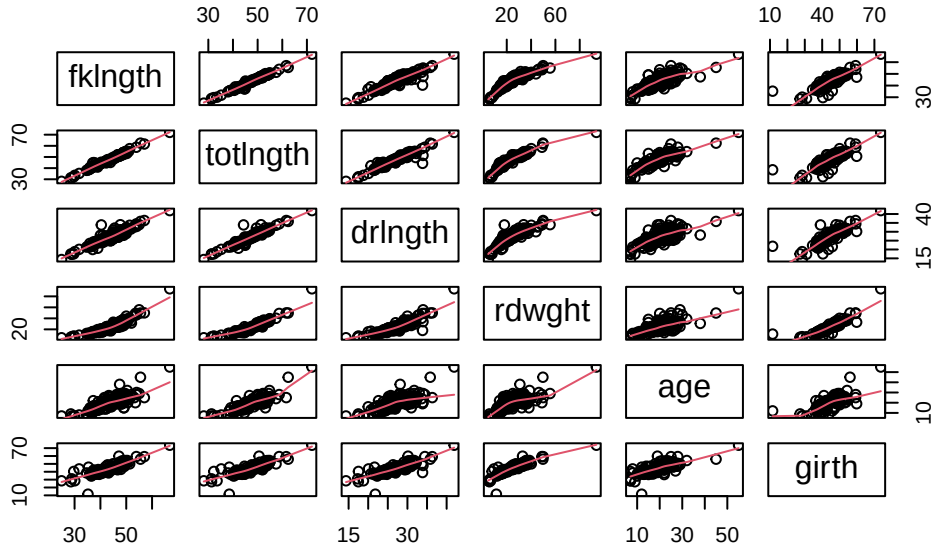
You should obtain:

```
plot(fklngth ~ age, data = sturgeon)
```



R has a function to create all pairwise scatterplots rapidly called `pairs()` . One of `pairs()` options is the addition of a lowess trace on each plot to that is a smoothed trend in the data. To get the plot matrix with the lowess smooth for all variables in the sturgeon data frame, execute the command `pairs(sturgeon, panel=panel.smooth)`. Howeber given the large number of variable in `sturgeon` we can limit the plot to the first 6 columns in the data.

```
pairs(sturgeon[, 1:6], panel = panel.smooth)
```



1.4. Creating data subsets

You will frequently want to do analyses on some subset of your data. The command `subset()` is what you need to isolate cases meeting some criteria. For example, to create a subset of the sturgeon data frame that contains only females caught in 1978, you could write:

```
sturgeon_female_1978 <- subset(sturgeon, sex == "FEMALE" & year == "1978")
sturgeon_female_1978
```

```
     fklngth totlngth  drlngth rdwght age girth    sex   location year
2   50.19685 54.13386 31.49606     NA  24  53.5 FEMALE    THE_PAS 1978
4   50.19685 53.14961 32.28346     NA  23  52.5 FEMALE    THE_PAS 1978
6   49.60630 53.93701 31.10236  35.86  23  54.2 FEMALE    THE_PAS 1978
7   47.71654 51.37795 33.97638  33.88  20  48.0 FEMALE    THE_PAS 1978
15  48.89764 53.93701 29.92126  35.86  23  52.5 FEMALE    THE_PAS 1978
105 46.85039       NA 28.34646  23.90  24    NA FEMALE CUMBERLAND 1978
106 40.74803       NA 24.80315  17.50  18    NA FEMALE CUMBERLAND 1978
107 40.35433       NA 25.59055  20.90  21    NA FEMALE CUMBERLAND 1978
109 43.30709       NA 27.95276  24.10  19    NA FEMALE CUMBERLAND 1978
113 53.54331       NA 33.85827  48.90  20    NA FEMALE CUMBERLAND 1978
114 51.77165       NA 31.49606  35.30  26    NA FEMALE CUMBERLAND 1978
116 45.27559       NA 26.57480  23.70  24    NA FEMALE CUMBERLAND 1978
118 53.14961       NA 32.67717  45.30  25    NA FEMALE CUMBERLAND 1978
119 50.19685       NA 32.08661  33.90  26    NA FEMALE CUMBERLAND 1978
123 49.01575       NA 29.13386  37.50  22    NA FEMALE CUMBERLAND 1978
```

Warning

When using criteria to select cases, be careful of the `==` syntax to mean equal to. In this context, if you use a single `=`, you will not get what you want. The following table lists the most common criteria to create expressions and their R syntax.

Operator	Explanation	Operator	Explanation
==	Equal to	!=	Not equal to
>	Larger than	<	Lower than
>=	Larger than or equal to	<=	Lower than or equal to
&	And (vectorized)	|	Or (vectorized)
&&	And (control)	||	Or (control)
!	Not		

Exercise

Using the commands `subset()` and `hist()` , create a histogram for females caught in 1979 and 1980 (hint: `sex=="FEMALE" & (year =="1979" | year=="1980")`)

Solution

```
sub_female_7980 <- subset(sturgeon, sex == "FEMALE" & (year == "1979" | year == "1980"))
hist(sub_female_7980$fklngth)
```

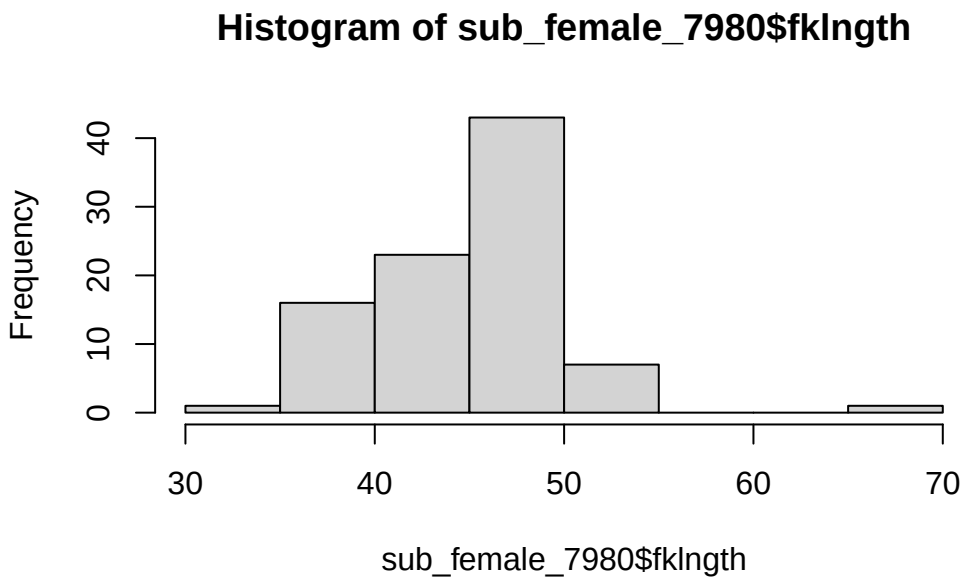


Figure 1.4.: Distibution of fluke length of female sturgeons in 1979 and 1980

1.5. Data transformation

You will frequently transform raw data to better satisfy assumptions of statistical tests. R will allow you to do that easily. The most used functions are probably:

- `log()`
- `sqrt()`
- `ifelse()`

You can use these functions directly within commands, create vector variables, or add columns in data frames. To do a plot of the decimal log of fklngth vs age, you can simply use the `log10()` function within the plot command:

```
plot(log10(fklngth)~age, data = sturgeon)
```

To create a vector variable, an orphan variable if you wish, one that is not part of a data frame, called `lfklngth` and corresponding too the decimal log of `fklngth`, simply enter:

```
logfklngth <- log10(sturgeon$fklngth)
```

If you want this new variable to be added to a data frame, then you must prefix the variable name by the data frame name and the `$` symbol. For example to add the variable `lfkl` containing the decimal log of `fklngth` to the `sturgeon` data frame, enter:

```
sturgeon$lfkl <- log10(sturgeon$fklngth)
```

`lfkl` will be added to the data frame `sturgeon` for the R session. Do not forget to save the modified data frame if you want to keep the modified version. Or better, save you Rscript and do not forget to run the line of code again next time you need it.

For conditional transformations, you can use the function `ifelse()`. For example, to create a new variable called dummy with a value of 1 for males and 0 for females, you can use:

```
sturgeon$dummy <- ifelse(sturgeon$sex == "MALE", 1, 0)
```

1.6. Exercice

The file `salmonella.csv` contains numerical values for the variable called ratio for two environments (`milieu`: `IN VITRO` or `IN VIVO`) and for 3 strains (`souche`). Examine the ratio variable and make a graph to visually assess normality for the wild (SAUVAGE) strain.

Solution

```r
## load the data
salmonella <- read.csv("data/salmonella.csv")

## create the base for the graph defining data and x
mygraph <- ggplot(subset(salmonella, souche == "SAUVAGE"), aes(x = ratio))
## add graph components
mygraph <- mygraph +
  # density smooth
  geom_density() +
  # obersations positions
  geom_rug() +
  # histogram
  geom_histogram(aes(y = ..density..),
    bins = 30,
    color = "black",
    alpha = 0.3
  ) +
  # ajusted Gaussian distribution
  stat_function(
    fun = dnorm,
    args = list(
      mean = mean(subset(salmonella, souche == "SAUVAGE")$ratio),
      sd = sd(subset(salmonella, souche == "SAUVAGE")$ratio)
    ),
    color = "red"
  )
## plot the graph
mygraph
```

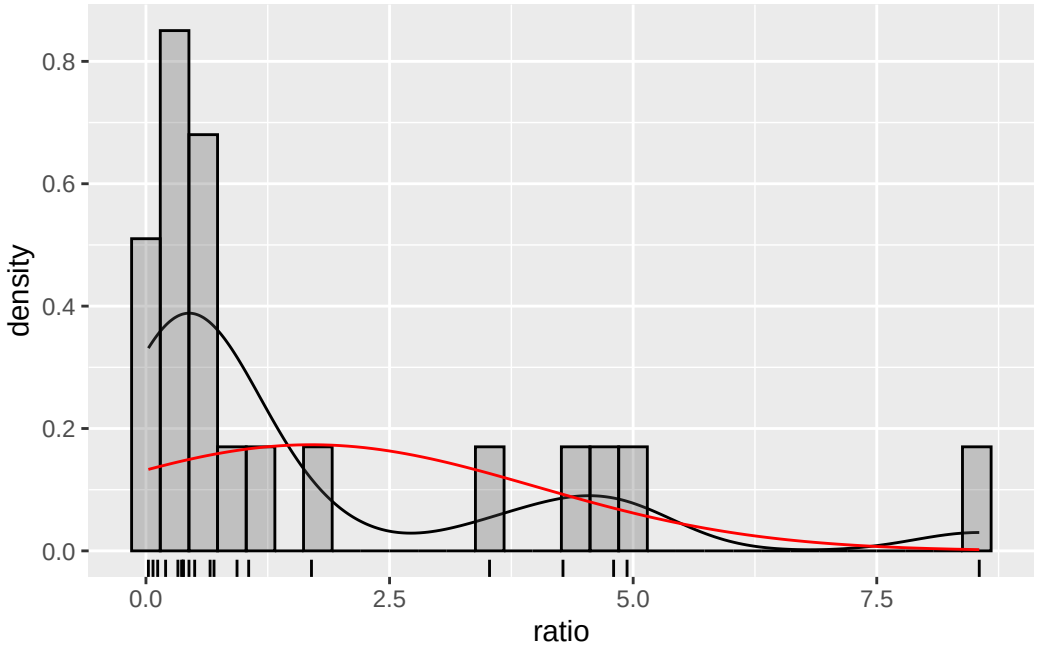


Figure 1.5.: Distibution of infection ratios by the wild (SAUVAGE) strain of salmonella

2. Power Analysis with R and G*Power

After completing this laboratory, you should :

- be able to compute the power of a t-test with G*Power and R
- be able to calculate the required sample size to achieve a desired power level with a t-test
- be able to calculate the detectable effect size by a t-test given the sample size, the power and α
- understand how power changes when sample size increases, the effect size changes, or when α decreases
- understand how power is affected when you change from a two-tailed to a one-tailed test.

2.1. The theory

2.1.1. What is power?

Power is the probability of rejecting the null hypothesis when it is false

2.1.2. Why do a power analysis?

Assess the strength of evidence

Power analysis, performed after accepting a null hypothesis, can help assess the probability of rejecting the null if it were false, and if the magnitude of the effect was equal to that observed (or to any other given magnitude). This type of *a posteriori* analysis is very common.

Design better experiments

Power analysis, performed prior to conducting an experiment (but most often after a preliminary experiment), can be used to determine the number of observations required to detect an effect of a given magnitude with some probability (the power). This type of *a priori* experiment should be more common.

Estimate minimum detectable effect

Sampling effort is often predetermined (when you are handed data of an experiment already completed), or extremely constrained (when logistics dictates what can be done). Whether it is *a priori* or *a posteriori*, power analysis can help you estimate, for a fixed sample size and a given power, what is the minimum effect size that can be detected.

2.1.3. Factors affecting power

For a given statistical test, there are 3 factors that affect power.

Decision criteria

Power is related to α, the probability level at which one rejects the null hypothesis. If this decision criteria is made very strict (i.e. if critical α is set to a very low value, like 0.1% or $p = 0.001$), then power will be lower than if the critical α was less strict.

Sample size

The larger the sample size, the larger the power. As sample size increases, one's ability to detect small effect sizes as being statistically significant gets better.

Effect size

The larger the effect size, the larger the power. For a given sample size, the ability to detect an effect as being significant is higher for large effects than for small ones. Effect size measures how false the null hypothesis is.

2.2. What is G*Power?

G*Power is free software developed by quantitative psychologists from the University of Dusseldorf in Germany. It is available in MacOS and Windows versions. It can be run under Linux using Wine or a virtual machine.

G*Power will allow you to do power analyses for the majority of statistical tests we will cover during the term without making lengthy calculations and looking up long tables and figures of power curves. It is a really useful tool that you need to master.

It is possible to perform all analysis made by G*Power in R, but it requires a bit more code, and a better understanding of the process since everything should be coded by hand. In simple cases, R code is also provided.

> Exercise
>
> Download the software **here** and install it on your computer and your workstation (if it is not there already).

2.3. How to use G*Power

2.3.1. General Principle

Using G*Power generally involves 3 steps:

1. Choosing the appropriate test
2. Choosing one of the 5 types of available power analyses
3. Enter parameter values and press the **Calculate** button

2.3.2. Types of power analyses

First, α is define as the probability level at which one rejects the null hypothesis, and β is $1 - power$.

A priori

Computes the sample size required given β, α, and the effect size. This type of analysis is useful when planning experiments.

Compromise

Computes α and β for a given α/β ratio, sample size, and effect size. Less commonly used (I have never used it myself) although it can be useful when the α/β ratio has meaning, for example when the cost of type I and type II errors can be quantified.

Criterion

Computes α for a given β, sample size, and effect size. In practice, I see little interest in this. Let me know if you see something I don't!

Post-hoc

Computes the power for a given α, effect size, and sample size. Used frequently to help in the interpretation of a test that is not statistically significant, but only if an effect size that is biologically significant is used (and not the observed effect size). Not relevant when the test is significant.

Sensitivity

Computes the detectable effect size for a given β ,α, and sample size. Very useful at the planning stage of an experiment.

2.3.3. How to calculate effect size

G*Power can perform power analyses for several statistical tests. The metric for effect size depends on the test. Note that other software packages often use different effect size metrics and that it is important to use the correct one for each package. G*Power has an effect size calculator for many tests that only requires you to enter the relevant values. The following table lists the effect size metrics used by G*Power for the various tests.

Test	Taille d'effet	Formule
t-test on means	d	$d = \frac{\lvert\mu_1 - \mu_2\rvert}{\sqrt{({s_1}^2 + {s_2}^2)/2}}$
t-test on correlations	r	
other t-tests	d	$d = \frac{\mu}{\sigma}$
F-test (ANOVA)	f	$f = \frac{\sqrt{\frac{\sum_{i=1}^{k}(\mu_i - \mu)^2}{k}}}{\sigma}$
other F-tests	f^2	$f^2 = \frac{{R_p}^2}{1 - {R_p}^2}$ R_p is the squared partial correlation coefficient
Chi-square test	w	$w = \sqrt{\sum_{i=1}^{m} \frac{(p_{0i} - p_{1i})^2}{p_{0i}}}$ p_{0i} and p_{1i} are theporportion in category i predicted by the null, $_0$, and alternative, $_1$, hypothesis

2.4. Power analysis for a t-test on two independent means

> **!** Important
>
> All the power analysis presented in this chapter can be done using 2 functions in R.
>
> - `pwr.t.test()` when the two samples have a similar size
> - `pwr.t2n.test()` when samples have different numbers of observations

The objective of this lab is to learn to use G*Power and understand how the 4 parameters of power analyses (α, β, sample size and effect size) are related to each other. For this, you will only use the standard t-test to compare two independent means. This is the test most used by biologists, you have all used it, and it will serve admirably for this lab. What you will learn today will be applicable to all other power analyses.

Jaynie Stephenson studied the productivity of streams in the Ottawa region. She has measured fish biomass in 36 streams, 18 on the Shield and 18 in the Ottawa Valley. She found that fish biomass was lower in streams from the valley (2.64 g/m^2 , standard deviation = 3.28) than from the Shield (3.31 g/m^2 , standard deviation = 2.79.).

When she tested the null hypothesis that fish biomass is the same in the two regions by a t-test, she obtained:

```
Pooled-Variance Two-Sample t-Test
t = -0.5746, df = 34, p-value = 0.5693
```

She therefore accepted the null hypothesis (since p is much larger than 0.05) and concluded that fish biomass is the same in the two regions.

2.4.1. Post-hoc analysis

Using the observed means and standard deviations, we can use G*Power to calculate the power of the two-tailed t-test for two independent means, using the observed effect size (the difference between the two means, weighted by the standard deviations) for $\alpha = 0.05$.

Start G*Power.

1. In ***Test family** , choose: t tests
2. For **Statistical test** , choose: Means: Difference between two independent means (two groups)
3. For **Type of power analysis** , choose: Post hoc: Compute achieved power - given α, sample size, and effect size
4. At **Input Parameters** ,

- in the box **Tail(s)** , chose: Two,
- check that α **err prob** is equal to 0.05
- Enter 18 for the **Sample size** of group 1 and of group 2
- then, to calculate effect size (d), click on **Determine =>**

5. In the window that opens,

- select **n1 = n2** , then
- enter the two means (**Mean group** 1 et 2)
- the two standard deviations(**SD group** 1 et 2)
- click on **Calculate** and **transfer to main window**

6. After you click on the **Calculate button** in the main window, you should get the following:

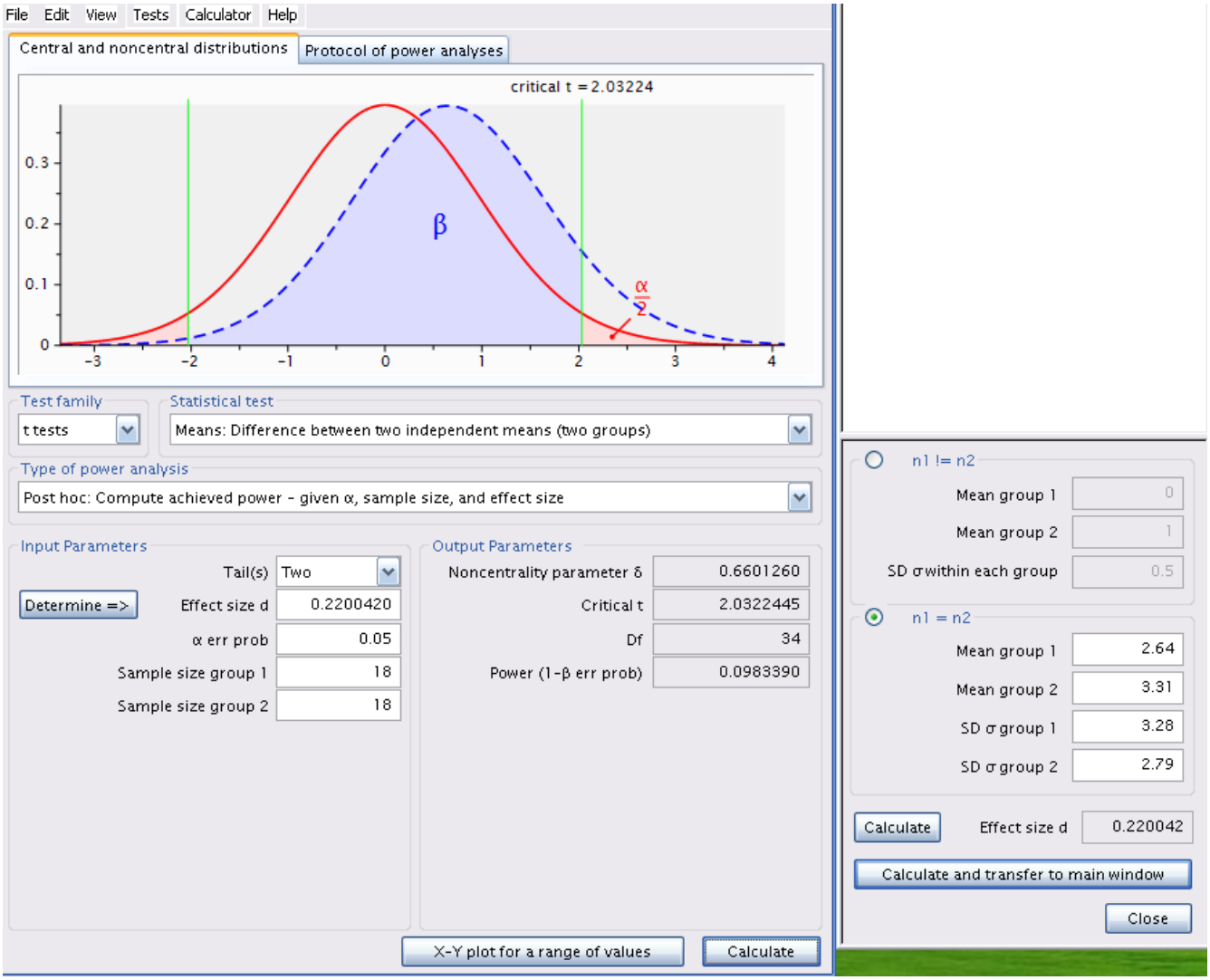


Figure 2.1.: Post-hoc analysis with estimated effect size

Similar analysis can be done in R. You first need to calculate the effect size `d` for a t-test comparing 2 means, and then use the `pwr.t.test()` function from the `pwr` . The easiest is to create anew function in R to estimate the effect size `d`since we are going to reuse it multiple times during the lab.

```r
# load package pwr
library(pwr)
# define d for a 2 sample t-test
d <- function(u1, u2, sd1, sd2) {
  abs(u1 - u2) / sqrt((sd1^2 + sd2^2) / 2)
}

# power analysis
pwr.t.test(
  n = 18,
  d = d(u1 = 2.64, sd1 = 3.28, u2 = 3.31, sd2 = 2.79),
```

```
  sig.level = 0.05,
  type = "two.sample")
```

```
     Two-sample t test power calculation

              n = 18
              d = 0.220042
      sig.level = 0.05
          power = 0.09833902
    alternative = two.sided

NOTE: n is number in *each* group
```

```
# plot similar to G*Power
x <- seq(-4, 4, length = 200)
plot(x, dnorm(x), type = "l", col = "red", lwd = 2)
qc <- qt(0.025, 16)
abline(v = qc, col = "green")
abline(v = -qc, col = "green")
lines(x, dnorm(x, mean = (3.31 - 2.64)), type = "l", col = "blue", lwd = 2)

# power corresponds to the shaded area
y <- dnorm(x, mean = (3.31 - 2.64))
polygon(
  c(x[x <= -qc], -qc), c(y[x <= -qc], 0),
  col = rgb(red = 0, green = 0.2, blue = 1, alpha = 0.5))
```

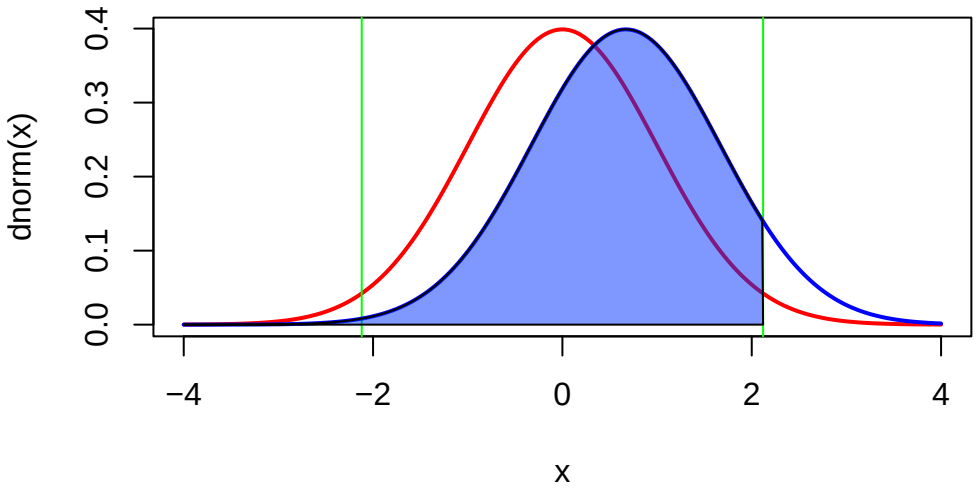


Figure 2.2.: Post-hoc analysis with estimated effect size in R

Let's examine the figure @ref(fig:gpower-1).

- The curve on the left, in red, corresponds to the expected distribution of the t-statistics when H_0 is true (*i.e.* when the two means are equal) given the sample size (18 per region) and the observed standard deviations.
- The vertical green lines correspond to the critical values of t for $\alpha = 0.05$ and a total sample size of 36 (2x18).
- The shaded pink regions correspond to the rejection zones for H_0. If Jaynie had obtained a *t-value* outside the interval delimited by the critical values ranging from -2.03224 to 2.03224, she would then have rejected H_0 , the null hypothesis of equal means. In fact, she obtained a t-value of -0.5746 and concluded that the biomass is equal in the two regions.
- The curve on the right, in blue, corresponds to the expected distribution of the t-statistics if H_1 is true (here H_1 is that there is a difference in biomass between the two regions equal to $3.33 - 2.64 = 0.69 g/m^2$, given the observed standard deviations). This distribution is what we should observe if H_1 was true and we repeated a large number of times the experiment using random samples of 18 streams in each of the two regions and calculated a t-statistic for each sample. On average, we would obtain a t-statistic of about 0.6.
- Note that there is considerable overlap of the two distributions and that a large fraction of the surface under the right curve is within the interval where H_0 is accepted between the two vertical green lines at -2.03224 and 2.03224. This proportion, shaded in blue under the distribution on the right is labeled β and corresponds to the risk of *type II error* (accept H_0 when H_1 is true).
- Power is simply $1 - \beta$, and is here 0.098339. Therefore, if the mean biomass differed by $0.69 g/m^2$ between the two regions, Jaynie had only 9.8% chance of being able to detect it as a statistically significant difference at =5% with a sample size of 18 streams in each region.

Let's recapitulate: The difference in biomass between regions is not statistically significant according to the t-test. It is because the difference is relatively small relative to the precision of the measurements. It is therefore not surprising that that power, i.e. the probability of detecting a statistically significant difference, is small. Therefore, this analysis is not very informative.

Indeed, a post hoc power analysis using the observed effect size is not useful. It is much more informative to conduct a post hoc power analysis for an effect size that is different from the observed effect size. But what effect size to use? It is the biology of the system under study that will guide you. For example, with respect to fish biomass in streams, one could argue that a two fold change in biomass (say from 2.64 to 5.28 g/m^2) has ecologically significant repercussions. We would therefore want to know if Jaynie had a good chance of detecting a difference as large as this before accepting her conclusion that the biomass is the same in the two regions. So, what were the odds that Jaynie could detect a difference of 2.64 g/m^2 between the two regions? G*Power can tell you if you cajole it the right way.

> Exercise
>
> Change the mean of group 2 to 5.28, recalculate effect size, and click on Calculate to obtain figure @ref(fig:gpower-2).

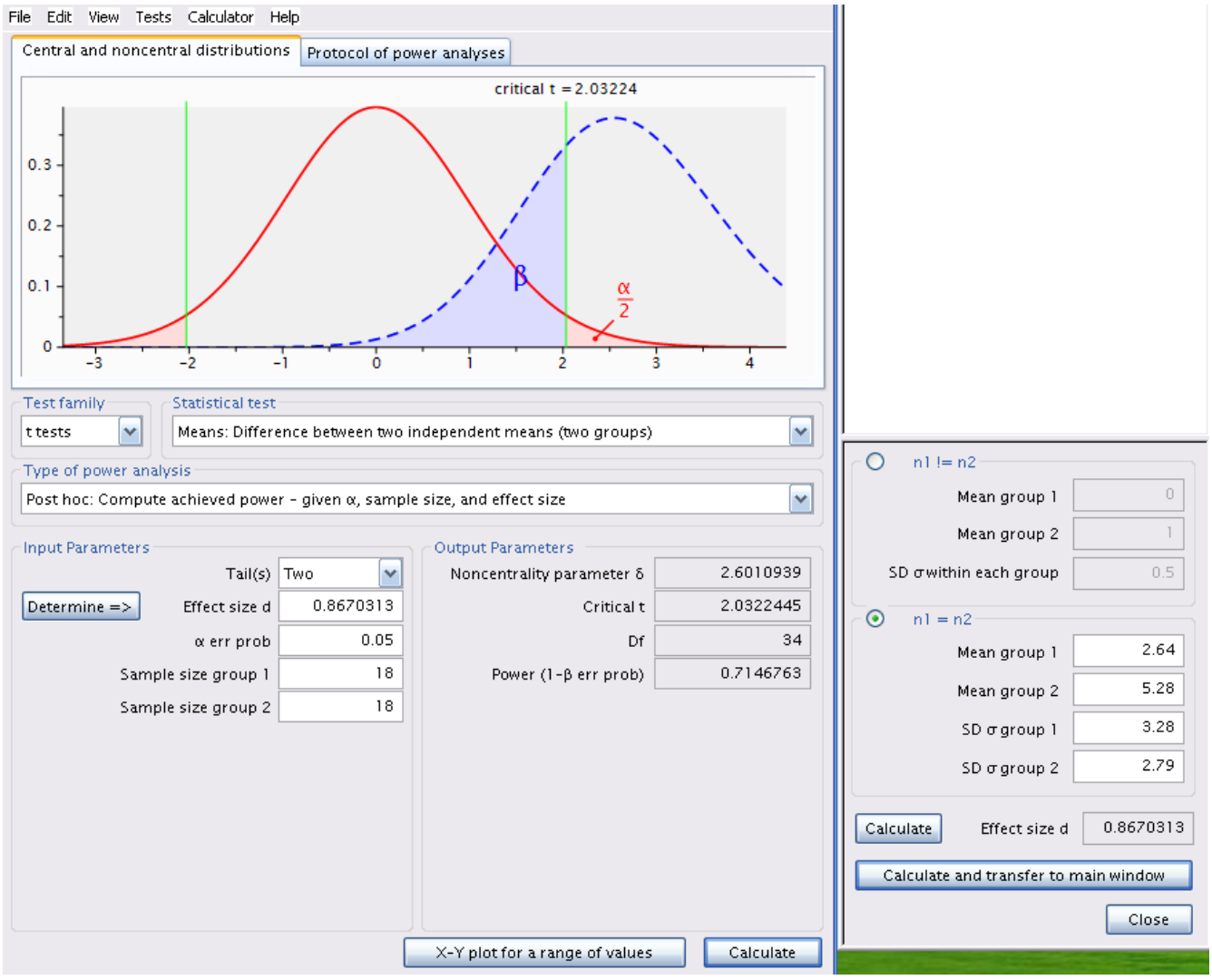


Figure 2.3.: Post-hoc analysis using an effect size different from the one estimated

Same analysis using R (without all the code for the interesting but not really useful plot)

```
pwr.t.test(
  n = 18,
  d = d(u1 = 2.64, sd1 = 3.28, u2 = 5.28, sd2 = 2.79),
  sig.level = 0.05,
  type = "two.sample")
```

```
     Two-sample t test power calculation

              n = 18
              d = 0.8670313
      sig.level = 0.05
          power = 0.7146763
    alternative = two.sided

NOTE: n is number in *each* group
```

The power is 0.71, therefore Jaynie had a reasonable chance (71%) of detecting a doubling of biomass with 18 streams in each region.

Note that this post hoc power analysis, done for an effect size considered biologically meaningful, is much more informative than the preceeding one done with the observed effect size (which is what too many students do because it is the default of so many power calculation programs). Jaynie did not detect a difference between the two regions. There are two possibilities: 1) there is really no difference between the regions, or 2) the precision of measurements is so low (because the sample size is small and/or there is large variability within a region) that it is very unlikely to be able to detect even large differences. The second power analysis can eliminate this second possibility because Jaynie had 71% chances of detecting a doubling of biomass.

2.4.2. A priori analysis

Suppose that a difference in biomass of $3.31 - 2.64 = 0.67 g/m^2$ can be ecologically significant. The next field season should be planned so that Jaynie would have a good chance of detecting such a difference in fish biomass between regions. How many streams should Jaynie sample in each region to have 80% of detecting such a difference (given the observed variability)?

> Exercise
>
> Change the type of power analysis in G*Power to **A priori: Compute sample size - given α , power, and effect size**. Ensure that the values for means and standard deviations are those obtained by Jaynie. Recalculate the effect size metric and enter 0.8 for power and you will obtain (@ref(fig:gpower-3).

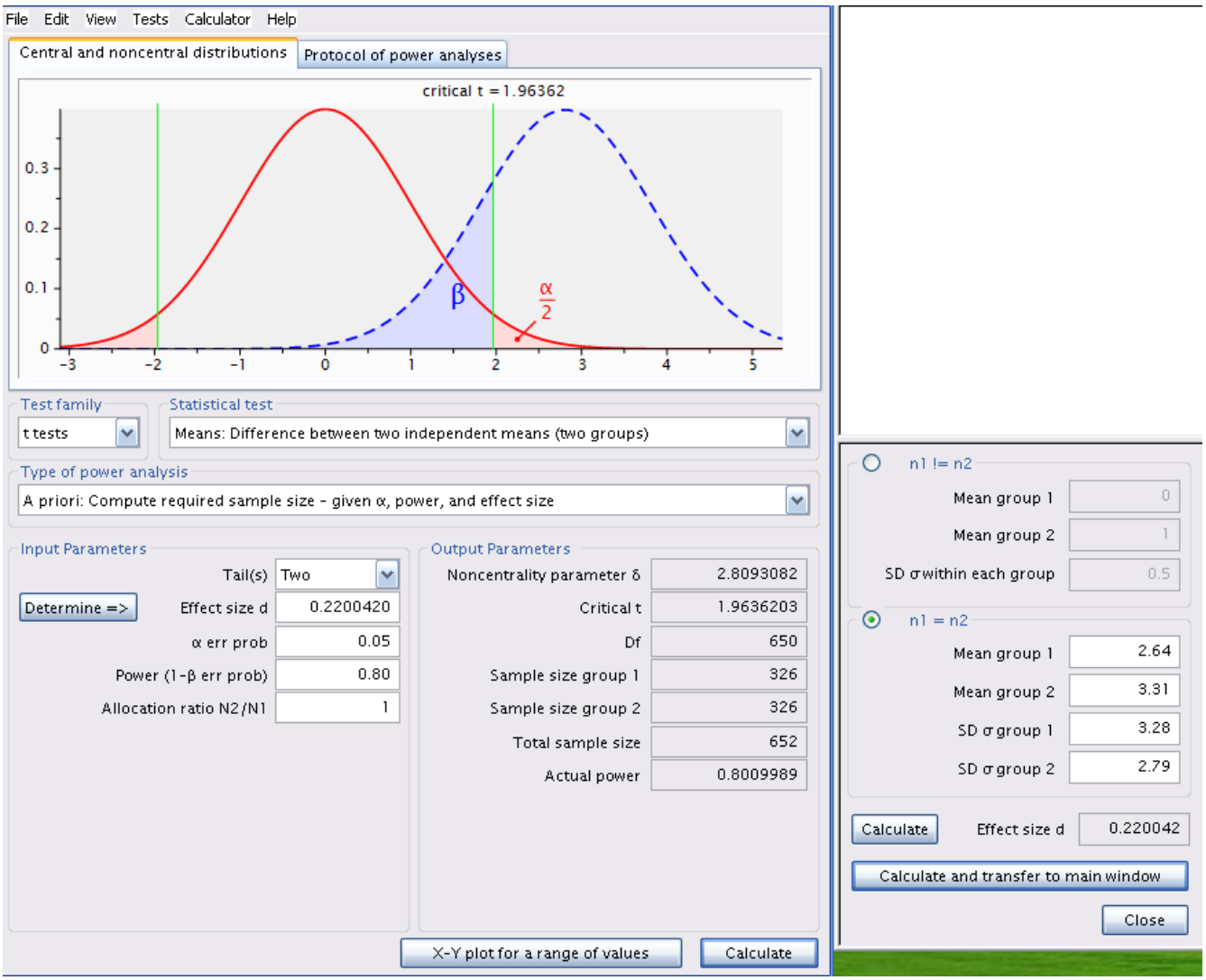


Figure 2.4.: A priori analysis

```
pwr.t.test(
  power = 0.8,
  d = d(u1 = 2.64, sd1 = 3.28, u2 = 3.31, sd2 = 2.79),
  sig.level = 0.05,
  type = "two.sample")
```

```
     Two-sample t test power calculation

              n = 325.1723
              d = 0.220042
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

```
NOTE: n is number in *each* group
```

Ouch! The required sample would be of 326 streams in each region! It would cost a fortune and require several field teams otherwise only a few dozen streams could be sampled over the summer and it would be very unlikely that such a small difference in biomass could be detected. Sampling fewer streams would probably be in vain and could be considered as a waste of effort and time: why do the work on several dozens of streams if the odds of success are that low?

If we recalculate for a power of 95%, we find that 538 streams would be required from each region. Increasing power means more work!

```
pwr.t.test(
  power = 0.95,
  d = d(u1 = 2.64, sd1 = 3.28, u2 = 3.31, sd2 = 2.79),
  sig.level = 0.05,
  type = "two.sample")
```

```
     Two-sample t test power calculation

              n = 537.7286
              d = 0.220042
      sig.level = 0.05
          power = 0.95
    alternative = two.sided

NOTE: n is number in *each* group
```

2.4.3. Sensitivity analysis - Calculate the detectable effect size

Given the observed variability, a sampling effort of 18 streams per region, and with $\alpha = 0.05$, what effect size could Jaynie detect with 80% probability $\beta = 0.2$)?

> Exercise
>
> Change analysis type in G*Power to **Sensitivity: Compute required effect size - given α , power, and sample size** and size is 18 in each region.

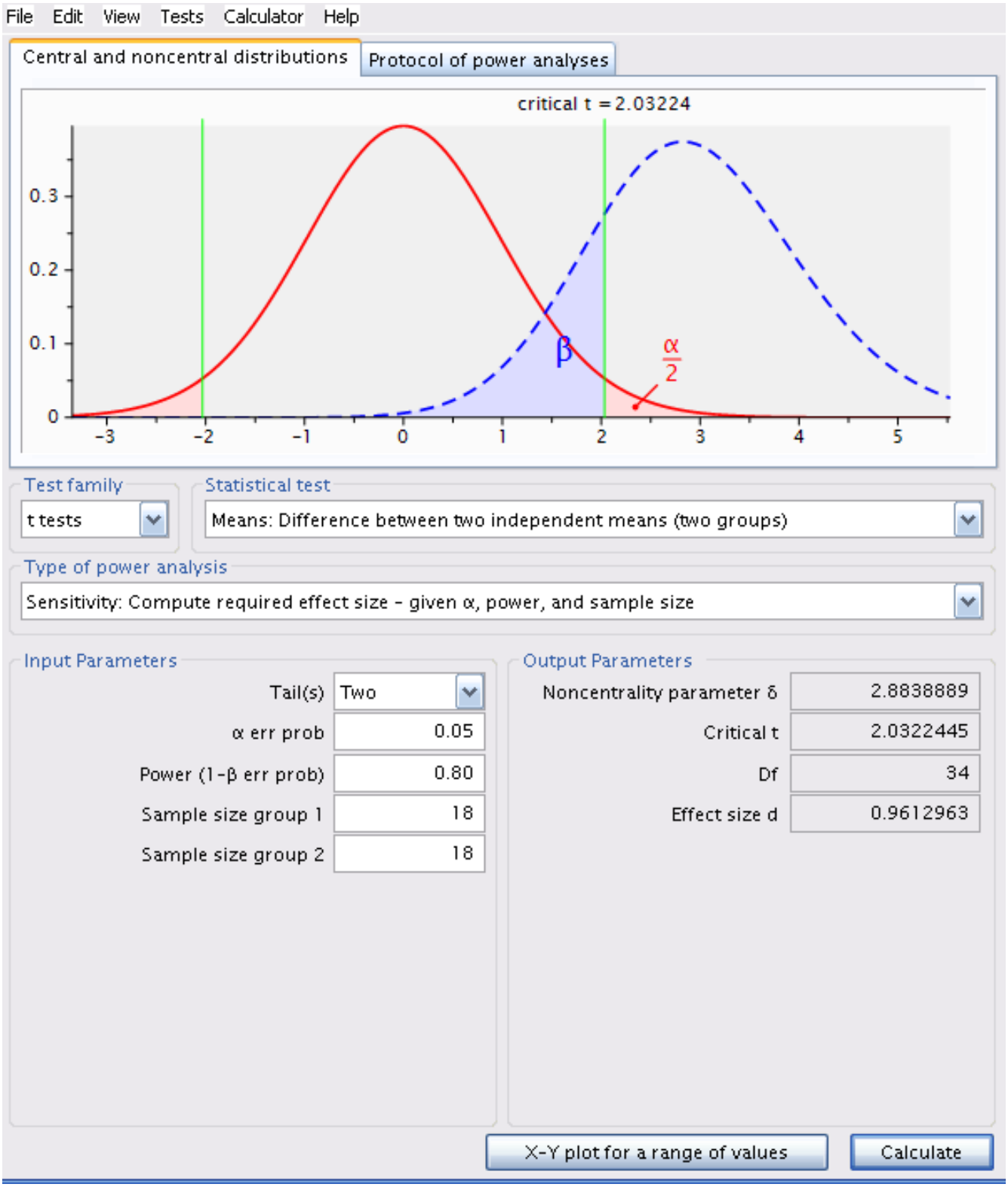


Figure 2.5.: Analyse de sensitivité

```
pwr.t.test(
  power = 0.8,
```

```
  n = 18,
  sig.level = 0.05,
  type = "two.sample")
```

```
     Two-sample t test power calculation

              n = 18
              d = 0.9612854
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

The detectable effect size for this sample size, $\alpha = 0.05$ and $\beta = 0.2$ (or power of 80%) is 0.961296.

> **Warning**
>
> Attention, this effect size is the metric `d` and is dependent on sampling variability.

Here, `d` is approximately equal to

$$d = \frac{|\bar{X}_1 \bar{X}_2|}{\sqrt{\frac{{s_1}^2 + {s_2}^2}{2}}}$$

To convert this d value without units into a value for the detectable difference in biomass between the two regions, you need to multiply `d` by the denominator of the equation.

$$|\bar{X}_1 - \bar{X}_2| = d * \sqrt{\frac{{s_1}^2 + {s_2}^2}{2}}$$

In R this can be done with the following code

```
pwr.t.test(
  power = 0.8,
  n = 18,
  sig.level = 0.05,
  type = "two.sample")$d * sqrt((3.28^2 + 2.79^2) / 2)
```

```
[1] 2.926992
```

Therefore, with 18 streams per region, $\alpha = 0.05$ and $\beta = 0.2$ (so power of 80%), Jaynie could detect a difference of 2.93 g/m^2 between regions, a bit more than a doubling of biomass.

2.5. Important points to remember

- Post hoc power analyses are relevant only when the null hypothesis is accepted because it is impossible to make a *type II error* when rejecting H_0 .
- With very large samples, power is very high and minute differences can be statistically detected, even if they are not biologically significant.
- When using a stricter significance criteria ($\alpha < 0.05$) power is reduced.
- Maximizing power implies more sampling effort, unless you use a more liberal statistical criteria ($\alpha > 0.05$)
- The choice of β is somewhat arbitrary. $\beta = 0.2$ (power of 80%) is considered relatively high by most.

3. Correlation and simple linear regression

After completing this laboratory exercise, you should be able to:

- Use R to produce a scatter plot of the relationship between two variables.
- Use R to carry out some simple data transformations.
- Use R to compute the Pearson product-moment correlation between two variables and assess its statistical significance.
- Use R to compute the correlation between pairs of ranked vari- ables using the Spearman rank correlation and Kendall's tau.
- Use R to assess the significance of pairwise comparisons from a generalized correlation matrix using Bonferroni-adjusted proba- bilities.
- Use R do a simple linear regression
- Use R to test the validity of the assumptions underlying simple lin- ear regression
- Use R to assess significance of a regression by the bootstrap method
- Quantify effect size in simple regression and perform a power analysis using G*Power

3.1. R packages and data

For this la b you need:

- R packages:
  - car
  - lmtest
  - boot
  - pwr
  - ggplot
  - performance
- data:
  - sturgeon.csv

You need to load the packages in R with `library()` and if need needed install them first with `install.packages()` For the data, load them using the `read.csv()` function.

```
library(car)
library(lmtest)
library(performance)
library(boot)
library(ggplot2)
library(pwr)
```

```
sturgeon <- read.csv("data/sturgeon.csv")
```

> **Note**
>
> Note that the command to read the data assumes that the data file is in a folder named `data` within the working directory. Adjust as needed.

3.2. Scatter plots

Correlation and regression analysis should always begin with an examination of the data: this is a critical first step in determining whether such analyses are even appropriate for your data. Suppose we are interested in the extent to which length of male sturgeon in the vicinity of The Pas and Cumberland House covaries with weight. To address this question, we look at the correlation between `fklngth` and `rdwght`. Recall that one of the assumptions in correlation analysis is that the relationship between the two variables is linear. To evaluate this assumption, a good first step is to produce a scatterplot.

- Load the data from `sturgeon.csv` in an obk=jcet named `sturgeon`. Make a scatter plot of `rdwght` vs `fklngth` fit with a locally weighted regression (Loess) smoother, and a linear regression line.

```
sturgeon <- read.csv("data/sturgeon.csv")
str(sturgeon)
```

```
'data.frame':   186 obs. of  9 variables:
 $ fklngth : num  37 50.2 28.9 50.2 45.6 ...
 $ totlngth: num  40.7 54.1 31.3 53.1 49.5 ...
 $ drlngth : num  23.6 31.5 17.3 32.3 32.1 ...
 $ rdwght  : num  15.95 NA 6.49 NA 29.92 ...
 $ age     : int  11 24 7 23 20 23 20 7 23 19 ...
 $ girth   : num  40.5 53.5 31 52.5 50 54.2 48 28.5 44 39 ...
 $ sex     : chr  "MALE" "FEMALE" "MALE" "FEMALE" ...
 $ location: chr  "THE_PAS" "THE_PAS" "THE_PAS" "THE_PAS" ...
 $ year    : int  1978 1978 1978 1978 1978 1978 1978 1978 1978 1978 ...
```

```
mygraph <- ggplot(
  data = sturgeon[!is.na(sturgeon$rdwght), ], # source of data
  aes(x = fklngth, y = rdwght)
)
# plot data points, regression, loess trace
mygraph <- mygraph +
  stat_smooth(method = lm, se = FALSE, color = "green") + # add linear regression, but no SE shad
  stat_smooth(color = "red", se = FALSE) + # add loess
```

```
  geom_point() # add data points

mygraph # display graph
```

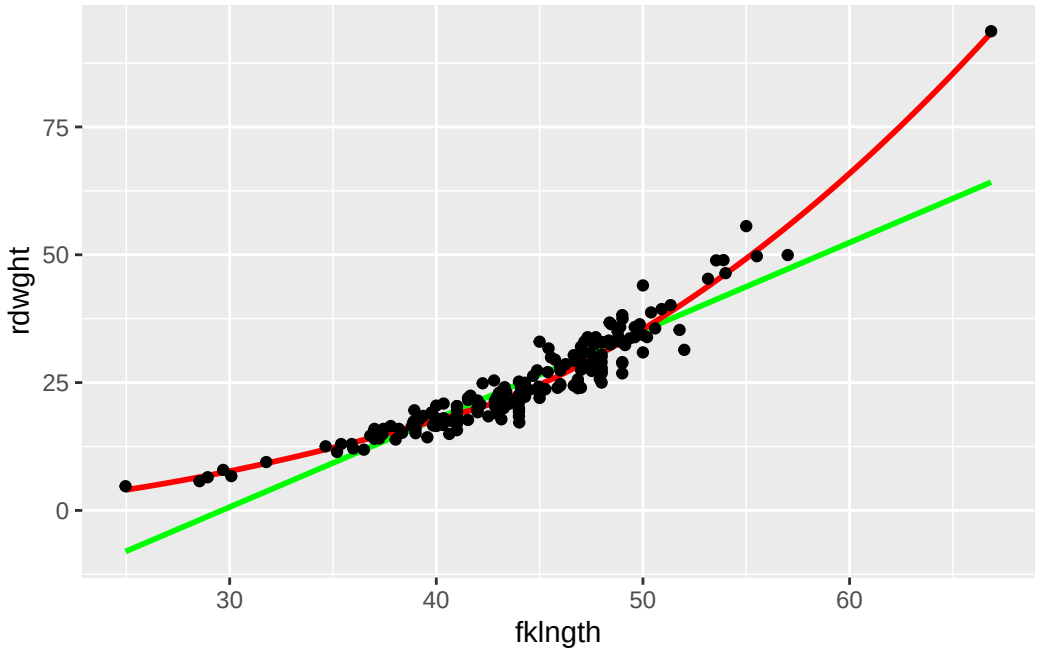


Figure 3.1.: Scatter plot of Weight as a function of length in sturgeons

- Does this curve suggest a good correlation between the two? Based on visual inspection, does the relationship between these two variables appear linear?

There is some evidence of nonlinearity, as the curve appears to have a positive second derivative (concave up). This notwithstanding, it does appear the two variables are highly correlated.

- Redo the scatterplot, but after logtransformation of both axes.

```
# apply log transformation on defined graph
mygraph + scale_x_log10() + scale_y_log10()
```

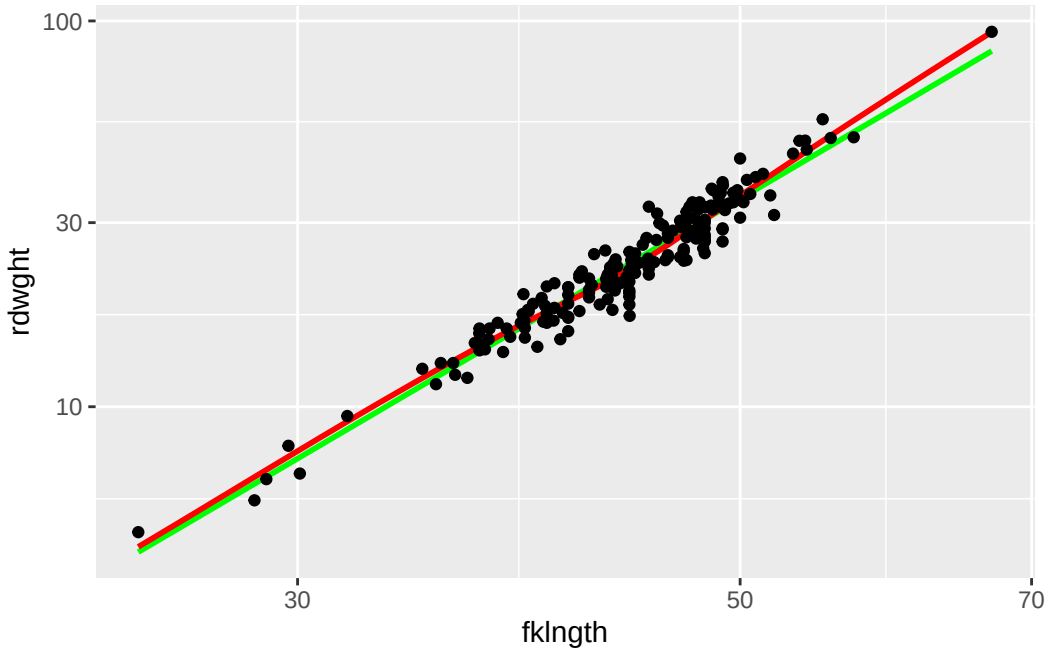


Figure 3.2.: Plot weight-length in sturgeon using a log scale

Compare the diagrams before and after the transformation (Figs @ref(fig:stur-2) and @ref(fig:stur-log)). Since the relationship is more linear after transformation, correlation analysis should be done on the transformed data

3.3. Data transformations and the product-moment correlation

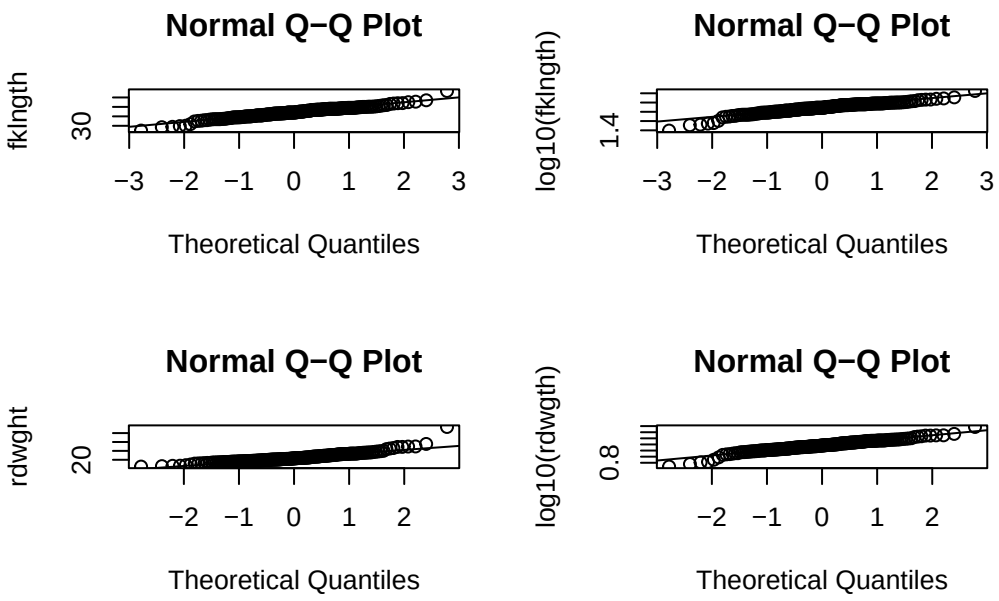
Recall that another assumption underlying significance testing of the product-moment correlation is that the distribution of the two variables in question is bivariate normal. We can test to see whether each of the two variables are normally distributed using the same procedures outlined in the exercise on two-sample comparisons. If the two variables are each normally distributed, then one is usually (relatively) safe in assuming the joint distribution is normal, although this needn't necessarily be true.

- Examine the distribution of the 4 variables (the two original variables and the log-transformed variables). What do you conclude from visual inspection of these plots?

The following graph contains the 4 QQ plots (`qqplot()`). It was produced by the code below that starts with the `par()` command to ensure that all 4 plots would appear together on the same page in 2 rows and 2 columns:

```
par(mfrow = c(2, 2)) # split graph in 4 (2 rows, 2 cols) filling by rows
qqnorm(sturgeon$fklngth, ylab = "fklngth")
qqline(sturgeon$fklngth)
qqnorm(log10(sturgeon$fklngth), ylab = "log10(fklngth)")
qqline(log10(sturgeon$fklngth))
qqnorm(sturgeon$rdwght, ylab = "rdwght")
qqline(sturgeon$rdwght)
```

```
qqnorm(log10(sturgeon$rdwght), ylab = "log10(rdwgth)")
qqline(log10(sturgeon$rdwght))
```



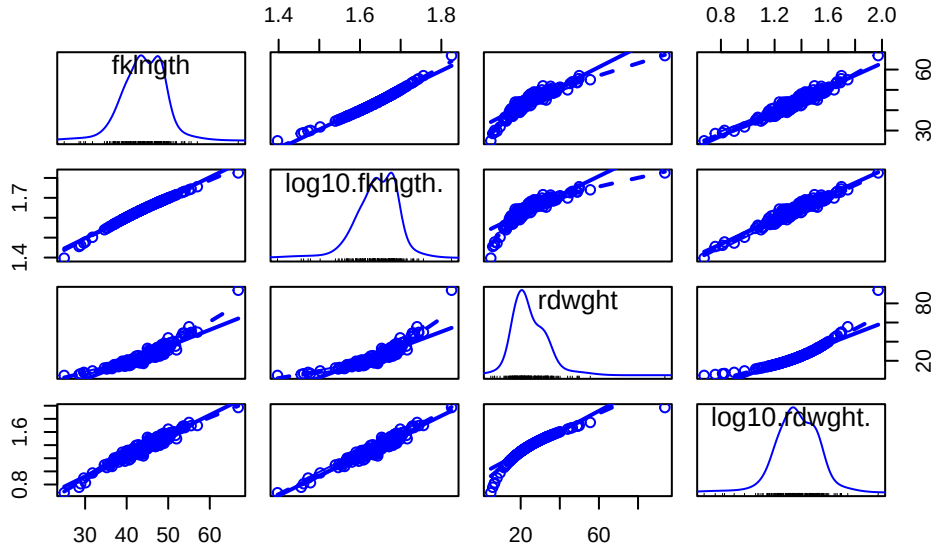

```
par(mfrow = c(1, 1)) # redefine plotting area to 1 plot
```

None of these distributions are perfectly normal, but deviations are mostly minor.

- To generate a scatterplot matrix of all pairs of variables, with linear regression and lowess traces, you can use `scatterplotMatrix` from `car` ‘r emo::ji(“package”).

```
scatterplotMatrix(
  ~ fklngth + log10(fklngth) + rdwght + log10(rdwght),
  data = sturgeon,
  smooth = TRUE, diagonal = "density"
)
```

```
Warning in applyDefaults(diagonal, defaults = list(method = "adaptiveDensity"),
: unnamed diag arguments, will be ignored
```



- Next, calculate the Pearson product-moment correlation between each pair (untransformed and log transformed) using the `cor()` command. However, to do this, it will be easier if you first add your transformed data as columns in the sturgeon data frame.

```
sturgeon$lfklngth <- with(sturgeon, log10(fklngth))
sturgeon$lrdwght <- log10(sturgeon$rdwght)
```

Then you can get the correlation matrix by:

```
cor(sturgeon[, c("fklngth", "lfklngth", "lrdwght", "rdwght")], use = "complete.obs")
```

Note the `use="complete.obs"` parameter. It tells R to keep only lines of the data frame where all variables were measured. If there are missing data, some lines will be removed, but correlations will be calculated for the same subset of cases for all pairs of variables. One could use, instead, `use="pairwise.complete.obs"` , to tell R to only eliminate observations when values are missing for this particular pair of variables. In this situation, if there are missing values in the data frame, the sample size for pairwise correlations will vary. In general, I recommend you use the option `use="complete.obs"`, unless you have so many missing values that it eliminates the majority of your data.

- Why is the correlation between the untransformed variables smaller than between the transformed variables?

```
cor(sturgeon[, c("fklngth", "lfklngth", "lrdwght", "rdwght")], use = "complete.obs")
```

```
           fklngth  lfklngth   lrdwght    rdwght
fklngth  1.0000000 0.9921435 0.9645108 0.9175435
lfklngth 0.9921435 1.0000000 0.9670139 0.8756203
lrdwght  0.9645108 0.9670139 1.0000000 0.9265513
rdwght   0.9175435 0.8756203 0.9265513 1.0000000
```

Several things should be noted here.

1. the correlation between fork length and round weight is high, regardless of which variables are used: so as might be expected, heavier fish are also longer, and vice versa
2. the correlation is greater for the transformed variables than the untransformed variables.

Why? Because the correlation coefficient is inversely proportional to the amount of scatter around a straight line. If the relationship is curvilinear (as it is for the untransformed data), the scatter around a straight line will be greater than if the relationship is linear. Hence, the correlation coefficient will be smaller.

3.4. Testing the significance of correlations and Bonferroni probabilities

It's possible to test the significance of individual correlations using the commands window. As an example, let's try testing the significance of the correlation between lfklngth and rdwght (the smallest correlation in the above table).

- In the R script window, enter the following to test the correlation between `lfkgnth` and `rdwght` :

```
cor.test(
  sturgeon$lfklngth, sturgeon$rdwght,
  alternative = "two.sided",
  method = "pearson"
)
```

```
    Pearson's product-moment correlation

data:  sturgeon$lfklngth and sturgeon$rdwght
t = 24.322, df = 180, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8367345 0.9057199
sample estimates:
      cor
0.8756203
```

We see here that the correlation is highly significant ($p < 2.2e-16$), which is no surprise given how high the correlation coefficient is (0.8756).

It's important to bear in mind that when you are estimating correlations, the probability of finding any one correlation that is *"significant"* by pure chance increases with the number of pairwise correlations

examined. Suppose, for example, that you have five variables; there are then a total of 10 possible pairwise correlations, and from this set, you would probably not be surprised to find at least one that is "significant" purely by chance. One way of avoiding the problem is to adjust individual α levels for pairwise correlations by dividing by the number of comparisons, k, such that: $\alpha' = \frac{\alpha}{k}$ (Bonferroni probabilities), i.e. if initially, $\alpha = 0.05$ and there are a total of 10 comparisons, then $\alpha' = 0.005$.

In the above example where we examined correlations between `fklngth` and `rdwght` and their `log`, it would be appropriate to adjust the α at which significance is tested by the total number of correlations in the matrix (in this case, 6, so $\alpha' = 0.0083$). Does your decision about the significance of the correlation between `lfklngth` and `rdwght` change?

3.5. Non-parametric correlations: Spearman's rank and Kendall's τ

The analysis done with the sturgeon data in the section above suggests that one of the assumptions of correlation, namely, bivariate normality, may not be valid for `fklngth` and `rdwght` nor for the log transforms of these variables. Finding an appropriate transformation is sometimes like looking for a needle in a haystack; indeed, it can be much worse simply because for some distributions, there is no transformation that will normalize the data. In such cases, the best option may be to go to a non-parametric analysis that does not assume bivariate normality or linearity. All such correlations are based on the ranks rather than the data themselves: two options available in R are Spearman and Kendall's τ (tau).

- Test the correlation between `fklngth` and `rdwght` using both the Spearman and Kendall's tau. The following commands will produce the correlations:

```
cor.test(
  sturgeon$lfklngth, sturgeon$rdwght,
  alternative = "two.sided",
  method = "spearman"
)
```

```
Warning in cor.test.default(sturgeon$lfklngth, sturgeon$rdwght, alternative =
"two.sided", : Cannot compute exact p-value with ties
```

```

    Spearman's rank correlation rho

data:  sturgeon$lfklngth and sturgeon$rdwght
S = 47971, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.9522546
```

```
cor.test(
  sturgeon$lfklngth, sturgeon$rdwght,
  alternative = "two.sided",
  method = "kendall"
)
```

```
    Kendall's rank correlation tau

data:  sturgeon$lfklngth and sturgeon$rdwght
z = 16.358, p-value < 2.2e-16
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.8208065
```

Contrast these results with those obtained using the Pearson product-moment correlation. Why the difference?

Test the non-parametric correlations on pairs of the transformed variables. You should immediately note that the non-parametric correlations are identical for untransformed and transformed variables. This is because we are using the ranks, rather than the raw data, and the rank ordering of the data does not change when a transformation is applied to the raw values.

Note that the correlations for Kendall's tau (0.820) are lower than for the Spearman rank (0.952) correlation. This is because Kendall's gives more weight to ranks that are far apart, whereas Spearman's weights each rank equally. Generally, Kendalls's is more appropriate when there is more uncertainty about the reliability of close ranks.

The sturgeons in this sample were collected using nets and baited hooks of a certain size. What impact do you think this method of collection had on the shapes of the distributions of `fklngth` and `rdwght` ? Under these circumstances, do you think correlation analysis is appropriate at all?

Note that correlation analysis assumes that each variable is *randomly sampled*. In the case of sturgeon, this is not the case: baited hooks and nets will only catch sturgeon above a certain minimum size. Note that in the sample, there are no small sturgeons, since the fishing gear targets only larger fish. Thus, we should be very wary of the correlation coefficients associated with our analysis, as the inclusion of smaller fish may well change our estimate of these correlations.

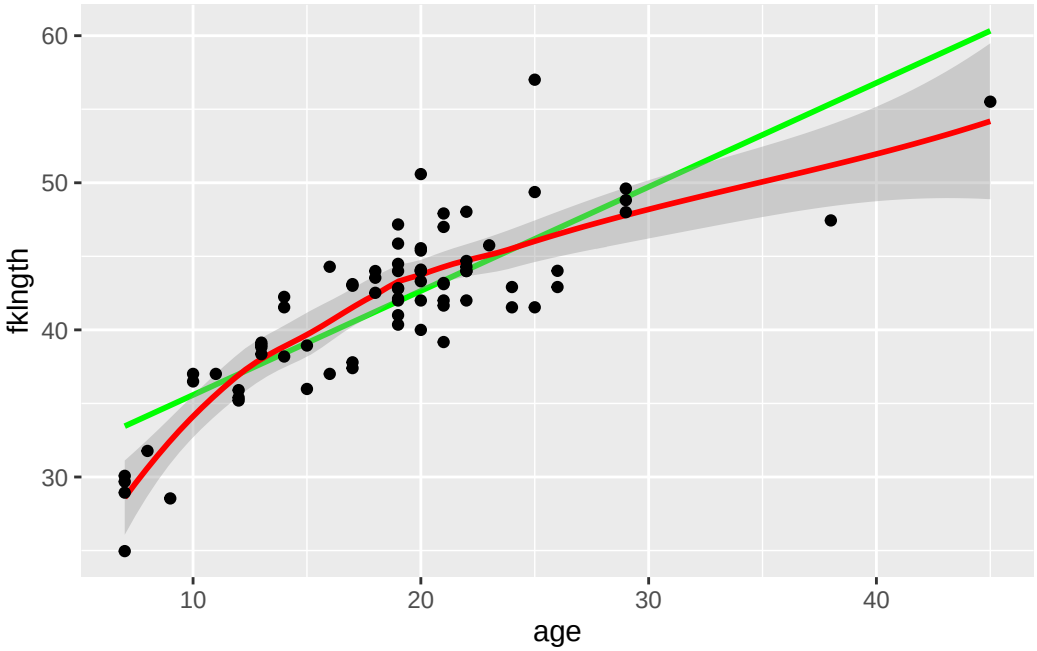
3.6. Simple linear regression

In correlation analysis we are interested in how pairs of variables covary: However, in regression analysis, we are attempting to estimate a model that predicts a variable (the dependent variable) from another variable (the independent variable).

As with any statistical analysis, the best way to begin is by looking at your data. If you are interested in the relationship between two variables, say, Y and X, produce a plot of Y versus X just to get a "feel" for the relationship.

- The data file `sturgeon.csv` contains data for sturgeons collected from 1978-1980 at Cumberland House, Saskatchewan and The Pas, Manitoba. Make a scatterplot of `fklngth` (the dependent variable) versus `age` (the independent variable) for males and add a linear regression and a loess smoother. What do you conclude from this plot?

```
sturgeon.male <- subset(sturgeon, subset = sex == "MALE")
mygraph <- ggplot(
  data = sturgeon.male, # source of data
  aes(x = age, y = fklngth)
) # aesthetics: y=fklngth, x=rdwght
# plot data points, regression, loess trace
mygraph <- mygraph +
  stat_smooth(method = lm, se = FALSE, color = "green") + # add linear regression, but no SE shad
  stat_smooth(color = "red") + # add loess
  geom_point() # add data points
mygraph # display graph
```



This suggests that the relationship between age and fork length is not linear.

Suppose that we want to know the growth rate of male sturgeon. One estimate (perhaps not a very good one) of the growth rate is given by the slope of the fork length - age regression.

First, let's run the regression with the lm() command, and save its results in an object called `RegModel.1`.

```
RegModel.1 <- lm(fklngth ~ age, data = sturgeon.male)
```

Nothing appears on the screen, but don't worry, it all got saved in memory. To see the statistical results, type:

```
summary(RegModel.1)
```

```
Call:
lm(formula = fklngth ~ age, data = sturgeon.male)

Residuals:
    Min      1Q  Median      3Q     Max
-8.4936 -2.2263  0.1849  1.7526 10.8234

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 28.50359    1.16873   24.39   <2e-16 ***
age          0.70724    0.05888   12.01   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.307 on 73 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.664, Adjusted R-squared:  0.6594
F-statistic: 144.3 on 1 and 73 DF,  p-value: < 2.2e-16
```

R output gives you:

1. `Call:` A friendly reminder of the model fitted and the data used.
2. `Residuals:` General statistics about the residuals around the fitted model.
3. `Coefficients:` Fitted model parameter estimates, standard errors, t values and associated probabilities.
4. `Residual standard error:` Square root of the residual variance.
5. `Multiple R-squared:` Coefficient of determination. It corresponds to the proportion of the total variance of the dependent variable that is accounted for by the regression (i.e. by the independent variable)
6. `Adjusted R-squared:` The adjusted R-squared accounts for the number of parameters in the model. If you want to compare the performance of several models with different numbers of parameters, this is the one to use
7. `F-statistic:` This is the test of the overall significance of the model. In the simple regression case, this is the same as the test of the slope of the regression.

The estimated regression equation is therefore:

$$Fklngth = 28.50359 + 0.70724 * age$$

Given the highly significant F-value of the model (or equivalently the highly significant t-value for the slope of the line), we reject the null hypothesis that there is no relationship between fork length and age.

3.6.1. Testing regression assumptions

Simple model I regression makes four assumptions:

1. the X variable is measured without error;
2. the relationship between Y and X is linear;
3. that for any value of X, the Y's are independently and normally distributed;
4. the variance of Y for fixed X is independent of X.

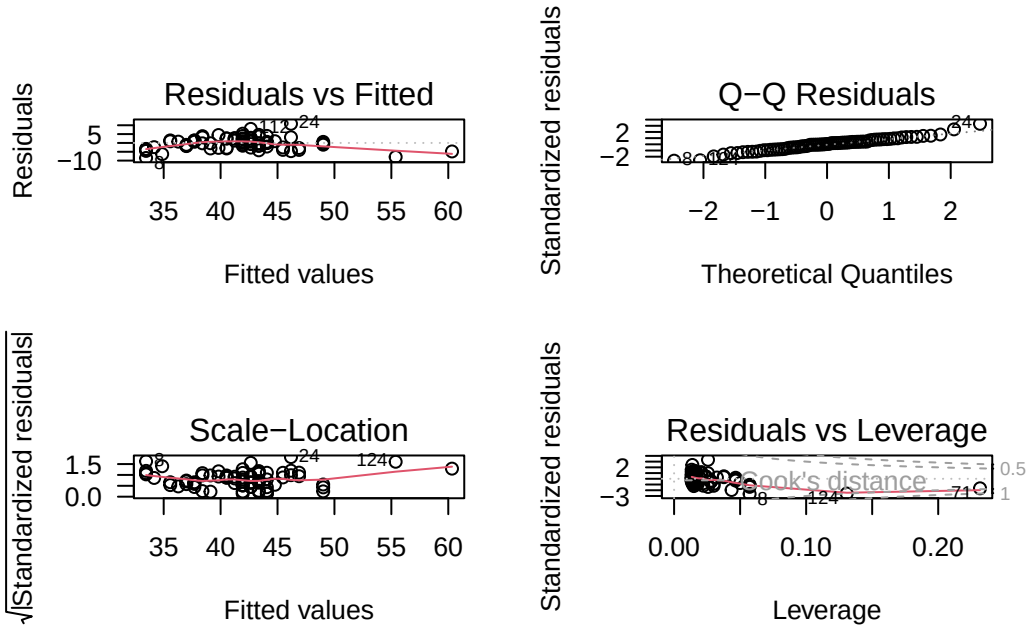
Having done the regression, we can now test the assumptions. For most biological data, the first assumption is almost never valid; usually there is error in both Y and X. This means that in general, slope estimates are biased, but predicted values are unbiased. However, so long as the error in X is small relative to the range of X in your data, the fact that X has an associated error is not likely to influence the outcome dramatically. On the other hand, if there is substantial error in X, regression results based on a model I regression may give poor estimates of the functional relationship between Y and X. In this case, more sophisticated regression procedures must be employed which are, unfortunately, beyond the scope of this course.

The other assumptions of a model I regression can, however, be tested, or at least evaluated visually. The `plot()` command can display diagnostics for linear models.

```
par(mfrow = c(2, 2), las = 1)
plot(RegModel.1)
```

The `par()` command is used here to tell R to display 2 rows and 2 columns of graphs per page (there are 4 diagnostic graphs for linear models generated automatically), and the last statement is to tell R to rotate the labels of the Y axis so that they are perpendicular to the Y axis. (Yes, I know, this is not at all obvious.)

You will get:

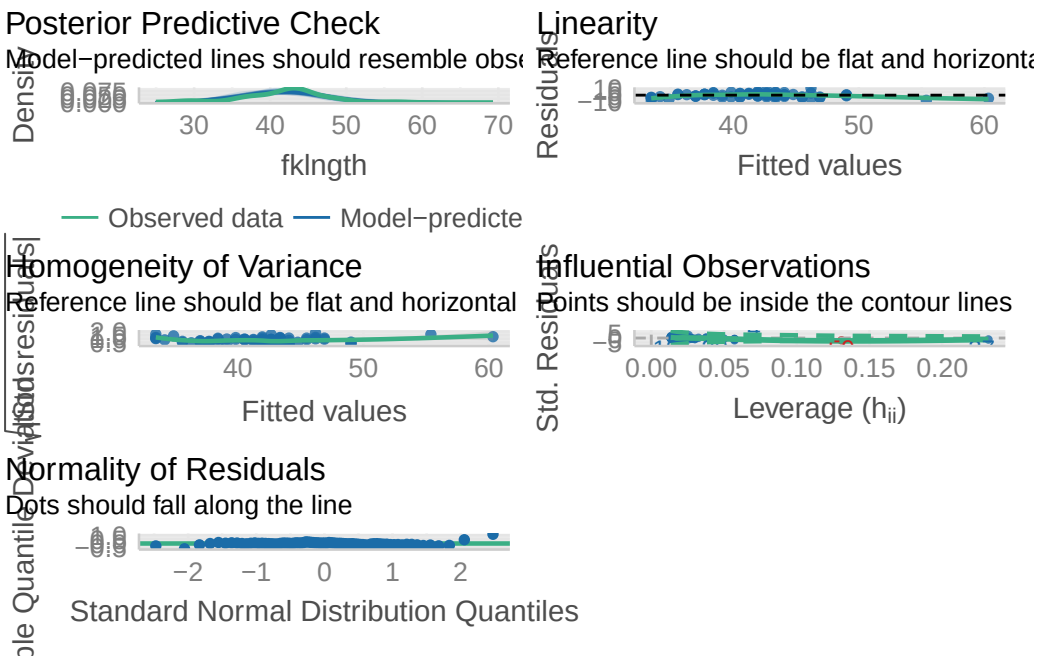


1. **Upper left** tell you about linearity, normality, and homoscedasticity of the residuals. It shows the deviations around the regression vs the predicted values. Remember that the scatterplot (fklngth vs age) suggested that the relationship between fork length and age is not linear. Very young and very old sturgeons tended to fall under the line, and fish of average age tended to be a bit above the line. This is exactly what the residual vs fitted plot shows. The red line is a lowess trace through these data. If the relationship was linear, it would be approximately flat and close to 0. The scatter of residuals tells you a bit about their normality and homoscedasticity, although this graph is not the best way to look at these properties. The next two are better.
2. **Upper right** is to assess the normality of the residuals. It is a QQ plot of the residuals . If the residuals were normally distributed, they would fall very close to the diagonal line. Here, we see it is mostly the case, except in the tails
3. **Bottom left** titled Scale-Location, helps with assessing homoscedasticity. It plots the square root of the absolute value of the standardized residual (residual divided by the standard error of the residuals, this scales the residuals so that their variance is 1) as a function of the fitted value. This graph can help you visualize whether the spread of the residuals is constant or not. If residuals are homoscedastic, then the average will not change with increasing fitted values. Here, there is slight variability, but it is not monotonous (i.e. it does not increase or decrease systematically) and there is no strong evidence against the assumption of homoscedasticity.
4. **Bottom right** plots the residuals as a function of leverage and can help detecting the presence of outliers or points that have a very strong influence on the regression results. The leverage of a point measures how far it is from the other points, but only with respect to the independent variable. In the case of simple linear regression, it is a function of the difference between the observation and the mean of the independent variable. You should look more closely at any observation with a leverage value that is greater than: $2(k+1)/n$, where $\$k$ is the number of independent variables (here 1), and n is the number of observations. In this case there is 1 independent variable, 75 observations, and points with a leverage higher than 0.053 may warrant particular scrutiny. The plot also gives you information about how the removal of a point from the data set would change the predictions. This is measured by the Cook's distance, illustrated by the red lines on the plot. A data point with a Cook distance larger than 1 has a large influence.

> ⚠ Warning
>
> Note that R automatically labels the most extreme cases on each of these 4 plots. It does not mean that these cases are outliers, or that you necessarily need be concerned with them. In any data set, there will always be a minimum and a maximum residual.

The R package `performance` offers a new and updated version of those graphs with colours and more plots to help visually assess the assumptions with the function `model_check()`

```
check_model(RegModel.1)
```



So, what is the verdict about the linear regression between `fklngth` and `age` ? It fails the linearity, possibly fails the normality, passes homoscedasticity, and this does not seem to be too strongly affected by some bizarre points.

3.6.2. Formal tests of regression assumptions

In my practice, I seldom use formal tests of regression assumptions and mostly rely on graphs of the residuals to guide my decisions. To my knowledge, this is what most biologists and data analysts do. However, in my early analyst life I was not always confident that I was interpreting these graphs correctly and wished that I had a formal test or a statistic quantifying the degree of deviation from the regression assumptions.

The `lmtest` R package, not part of the base R installation, but available from CRAN, contains a number of tests for linearity and homoscedasticity. And one can test for normality using the Shapiro-Wilk test seen previously.

First, you need to load (and maybe install) the `lmtest` package.

```
library(lmtest)
```

> Exercise
>
> Run the following commands

```
bptest(RegModel.1)
```

```
    studentized Breusch-Pagan test

data:  RegModel.1
BP = 1.1765, df = 1, p-value = 0.2781
```

The Breusch-Pagan test examines whether the variability of the residuals is constant with respect to increasing fitted values. A low p value is indicative of heteroscedasticity. Here, the p value is high, and supports my visual assessment that the homoscedasticity assumption is met by these data.

```
dwtest(RegModel.1)
```

```
    Durbin-Watson test

data:  RegModel.1
DW = 2.242, p-value = 0.8489
alternative hypothesis: true autocorrelation is greater than 0
```

The Durbin-Watson test can detect serial autocorrelation in the residuals. Under the assumption of no autocorrelation, the D statistic is 2. This test can detect violation of independence of observations (residuals), although it is not foolproof. Here there is no problem identified.

```
resettest(RegModel.1)
```

```
    RESET test

data:  RegModel.1
RESET = 14.544, df1 = 2, df2 = 71, p-value = 5.082e-06
```

The RESET test is a test of the assumption of linearity. If the linearity assumption is met, the RESET statistic will be close to 1. Here, the statistic is much larger (14.54), and very highly significant. This confirms our visual assessment that the relationship is not linear.

```
shapiro.test(residuals(RegModel.1))
```

```
    Shapiro-Wilk normality test

data:  residuals(RegModel.1)
W = 0.98037, p-value = 0.2961
```

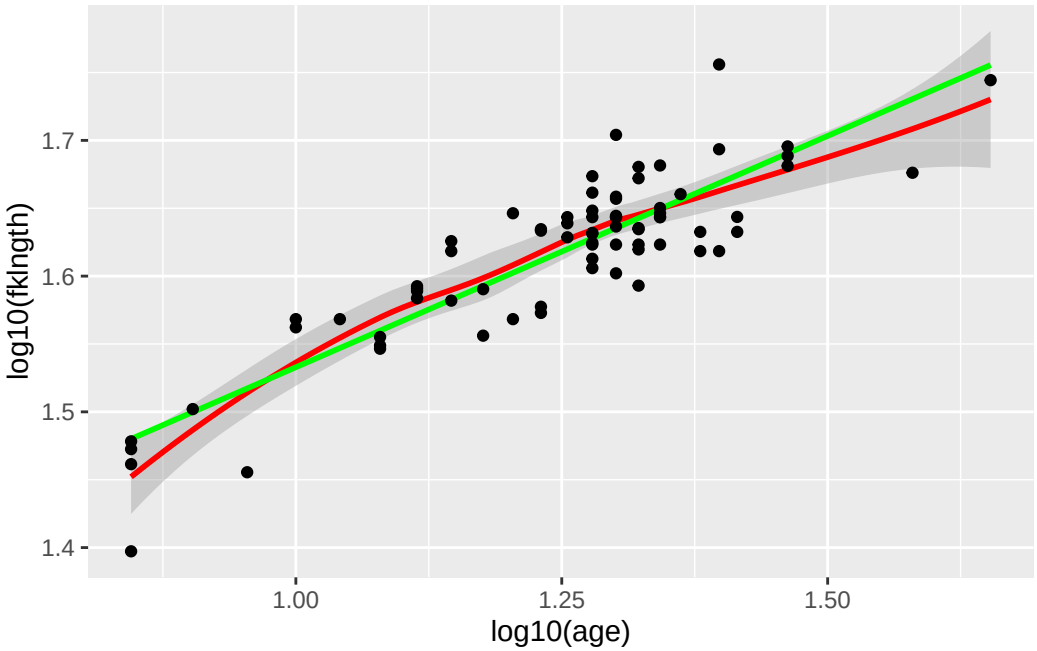
The Shapiro-Wilk normality test on the residual confirms that the deviation from normality of the residuals is not large.

3.7. Data transformations in regression

The analysis above revealed that the linearity assumption underlying regression analysis is not met by the `fklngth` - `age` data. If we want to use regression analysis, data transformations are required:

Let's plot the log-transformed data

```
par(mfrow = c(1, 1), las = 1)
ggplot(
  data = sturgeon.male,
  aes(x = log10(age), y = log10(fklngth))
) +
  geom_smooth(color = "red") +
  geom_smooth(method = "lm", se = FALSE, color = "green") +
  geom_point()
```



We can fit the linear regression model on the log-transformed variables.

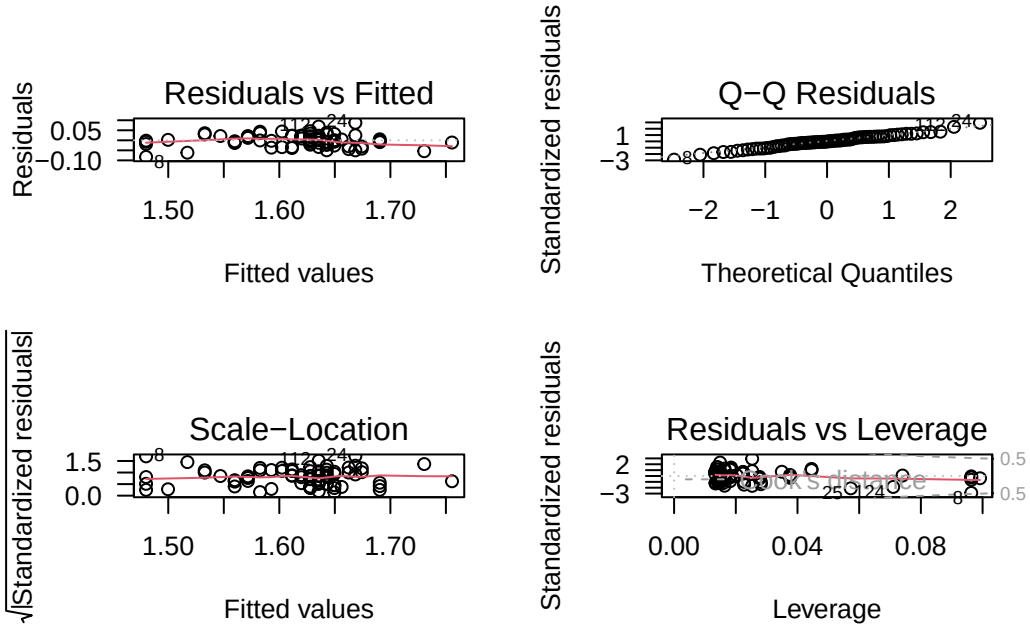
```
RegModel.2 <- lm(log10(fklngth) ~ log10(age), data = sturgeon.male)
summary(RegModel.2)
```

```
Call:
lm(formula = log10(fklngth) ~ log10(age), data = sturgeon.male)

Residuals:
      Min        1Q    Median        3Q       Max
```

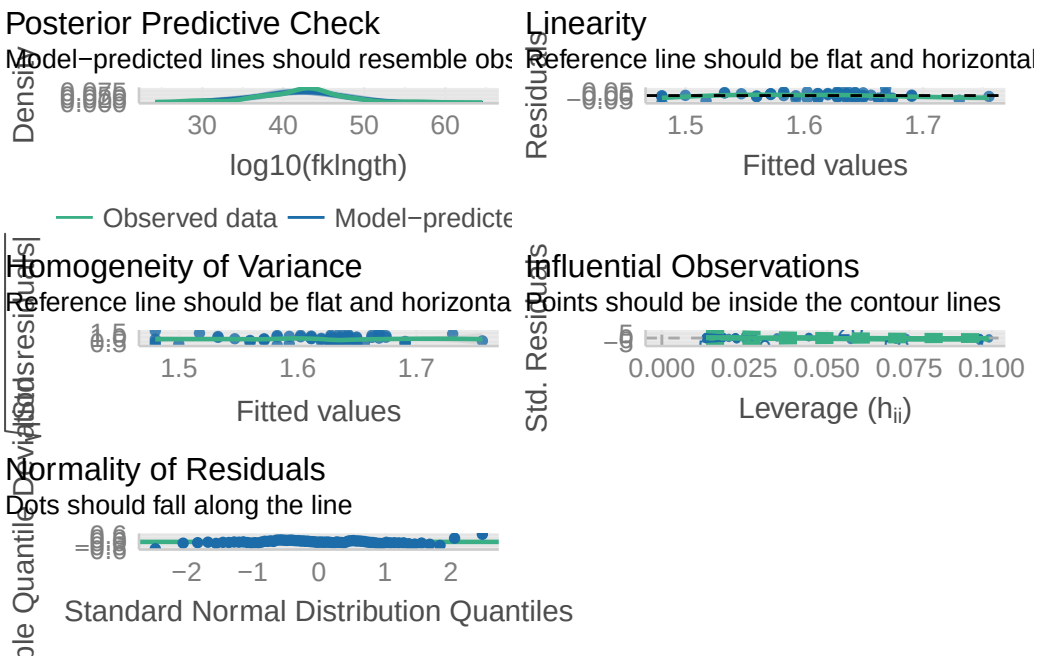
```
-0.082794 -0.016837 -0.000719  0.021102  0.087446

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.19199    0.02723   43.77   <2e-16 ***
log10(age)   0.34086    0.02168   15.72   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03015 on 73 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.772, Adjusted R-squared:  0.7688
F-statistic: 247.1 on 1 and 73 DF,  p-value: < 2.2e-16
```

Note that by using the log transformed data, the proportion of variation explained by the regression has increased by 10% (from 0.664 to 0.772), a substantial increase. So the relationship has become more linear. Good. Let's look at the residual diagnostic plots:

```
par(mfrow = c(2, 2), las = 1)
plot(RegModel.2)
```




```
check_model(RegModel.2)
```



So things appear a little better than before, although still not ideal. For example, the Residual vs fitted plot still suggests a potential nonlinearity. The QQ plot is nicer than before, indicating that residuals are more normally distributed after the log-log transformation. There is no indication of heteroscedasticity. And, although there are still a few points with somewhat high leverage, none have a Cook's distance above 0.5. It thus seems that transforming data improved things: more linear, more normal, less dependence on extreme data. Do the formal tests support this visual assessment?

```
bptest(RegModel.2)
```

```
    studentized Breusch-Pagan test

data:  RegModel.2
BP = 0.14282, df = 1, p-value = 0.7055
```

```
dwtest(RegModel.2)
```

```
    Durbin-Watson test

data:  RegModel.2
DW = 2.1777, p-value = 0.6134
alternative hypothesis: true autocorrelation is greater than 0
```

```
resettest(RegModel.2)
```

```
    RESET test

data:  RegModel.2
RESET = 4.4413, df1 = 2, df2 = 71, p-value = 0.01523
```

```
shapiro.test(residuals(RegModel.2))
```

```
    Shapiro-Wilk normality test

data:  residuals(RegModel.2)
W = 0.98998, p-value = 0.8246
```

Indeed, they do: residuals are still homoscedastic (Breusch-Pagan test), show no autocorrelation (Durbin-Watson test), are normal (Shapiro-Wilk test), and they are more linear (p value of the RESET test is now 0.015, instead of 0.000005). Linearity has improved, but is still violated somewhat.

3.8. Dealing with outliers

In this case, there are no real clear outliers. Yes, observations 8, 24, and 112 are labeled as the most extreme in the last set of residual diagnostic plots. But they are still within what I consider the "reasonable" range. But how does one define a limit to the reasonable? When is an extreme value a real outlier we have to deal with? Opinions vary about the issue, but I favor conservatism.

My rule is that, unless the value is clearly impossible or an error in data entry, I do not delete "outliers"; rather, I analyze all my data. Why? Because, I want my data to reflect natural or real variability. Indeed, variability is often what interests biologists the most.

Keeping extreme values is the fairest way to proceed, but it often creates other issues. These values will often be the main reason why the data fail to meet the assumptions of the statistical analysis. One solution is to run the analysis with and without the outliers, and compare the results. In many cases, the two analyses will be qualitatively similar: the same conclusions will be reached, and the effect size will not be very different. Sometimes, however, this comparison will reveal that the presence of the outliers changes the story. The logical conclusion then is that the results depend on the outliers and that the data at hand are not very conclusive. As an example, let's rerun the analysis after eliminating observations labeled 8, 24, and 112.

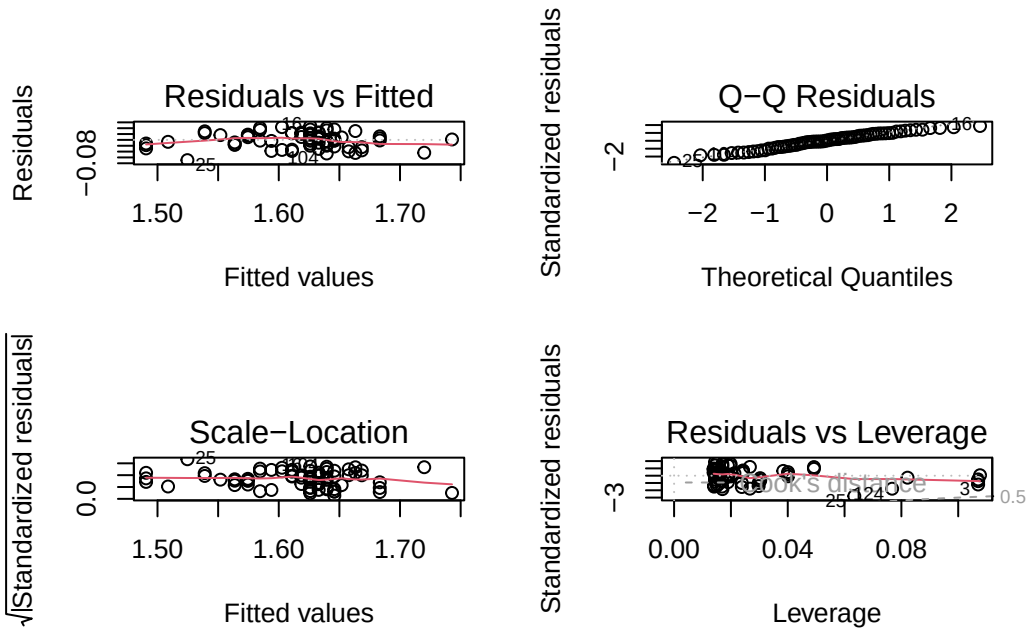
```
RegModel.3 <- lm(log10(fklngth) ~ log10(age), data = sturgeon.male, subset = !(rownames(sturgeon.
summary(RegModel.3)
```

```
Call:
lm(formula = log10(fklngth) ~ log10(age), data = sturgeon.male,
    subset = !(rownames(sturgeon.male) %in% c("8", "24", "112")))

Residuals:
      Min        1Q    Median        3Q       Max
-0.069163 -0.017390  0.000986  0.018590  0.047647

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.22676    0.02431   50.46   <2e-16 ***
log10(age)   0.31219    0.01932   16.16   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02554 on 70 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.7885,    Adjusted R-squared:  0.7855
F-statistic:   261 on 1 and 70 DF,  p-value: < 2.2e-16
```

The intercept, slope, and R squared are about the same, and the significance of the slope is still astronomical. Removing the "outliers" has little effect in this case.

As for the diagnostic residual plots and the formal tests of assumptions:

```
par(mfrow = c(2, 2))
plot(RegModel.3)
```



```
bptest(RegModel.3)
```

```
    studentized Breusch-Pagan test

data:  RegModel.3
BP = 0.3001, df = 1, p-value = 0.5838
```

```
dwtest(RegModel.3)
```

```
    Durbin-Watson test

data:  RegModel.3
DW = 2.0171, p-value = 0.5074
alternative hypothesis: true autocorrelation is greater than 0
```

```
resettest(RegModel.3)
```

```
    RESET test

data:  RegModel.3
RESET = 3.407, df1 = 2, df2 = 68, p-value = 0.0389
```

```
shapiro.test(residuals(RegModel.3))
```

```
    Shapiro-Wilk normality test

data:  residuals(RegModel.3)
W = 0.98318, p-value = 0.4502
```

No real difference either. Overall, this suggests that the most extreme values do not have undue influence on the results.

3.9. Quantifying effect size in regression and power analysis

Biological interpretation differs from statistical interpretation. Statistically, we conclude that size increase with age (i.e. the slope is positive and different from 0). But this conclusion alone does not tell if the difference between young and old fish is large. The slope and the scatterplot are more informative than the p-value here. The slope (in log-log space) is 0.34. This means that for each unit increase of X (`log10(age)`), there is an increase of 0.34 units of `log10(fklngth)`. In other words, when age is multiplied by 10, fork length is multiplied by about 2 (`10^0.34^`). Humm, length increases more slowly than age. This slope value (0.34) is an estimate of raw effect size. It measure how much length changes with age.

It would also be important to estimate the confidence interval around the estimate of the slope. This can beobtained using the `confint()` function.

```
confint(RegModel.2)
```

```
                2.5 %   97.5 %
(Intercept) 1.1377151 1.246270
log10(age)  0.2976433 0.384068
```

The 95% confidence interval for the slope is 0.29-0.38. It is quite narrow and include only values far from zero.

3.9.1. Power to detect a given slope

You can compute power with G*Power for some slope value that you deem of sufficient magnitude to warrant detection.

1. Go to **t Tests: Linear bivariate regression: One group, size of slope**.
2. Select **Post hoc: Compute achieved power- given α, sample size,and effect size**

For example, suppose that sturgeon biologists deem that a slope of 0.1 for the relationship between `log10(fklngth)` and `log10(age)` is meaningful and you wanted to estimate the power to detect such a slope with a sample of 20 sturgeons. Results from the log-log regression contain most of what you need:

```
summary(RegModel.2)
```

```
Call:
lm(formula = log10(fklngth) ~ log10(age), data = sturgeon.male)

Residuals:
      Min        1Q    Median        3Q       Max
-0.082794 -0.016837 -0.000719  0.021102  0.087446

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.19199    0.02723   43.77   <2e-16 ***
log10(age)   0.34086    0.02168   15.72   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03015 on 73 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.772, Adjusted R-squared:  0.7688
F-statistic: 247.1 on 1 and 73 DF,  p-value: < 2.2e-16
```

Note the Residual standard error value (0.03015). You will need this. The other thing you need is an estimate of the standard deviation of `log10(age)`. R can (of course) compute it. Be careful, the `sd()` function will return `NA` if there are missing values. You can get around this by adding `na.rm=TRUE` as an argument ot the `sd()` function.

```
sd(log10(sturgeon.male$age), na.rm = TRUE)
```

```
[1] 0.1616675
```

You can then enter these values (slope to be detected, sample size, alpha, standard deviation of the independent variable) to calculate another quantity that G*Power needs (standard deviation of y) using the Determine panel. Finally you can calculate Power. The filled panels should look like this

> **Note**
>
> Note: The SD of y can't just be taken from the data because if the slope chanages (e.g. H1) then this will change the SD of y. SD y needs to be estimated from the observed scatter around the line and the hypothesized slope).

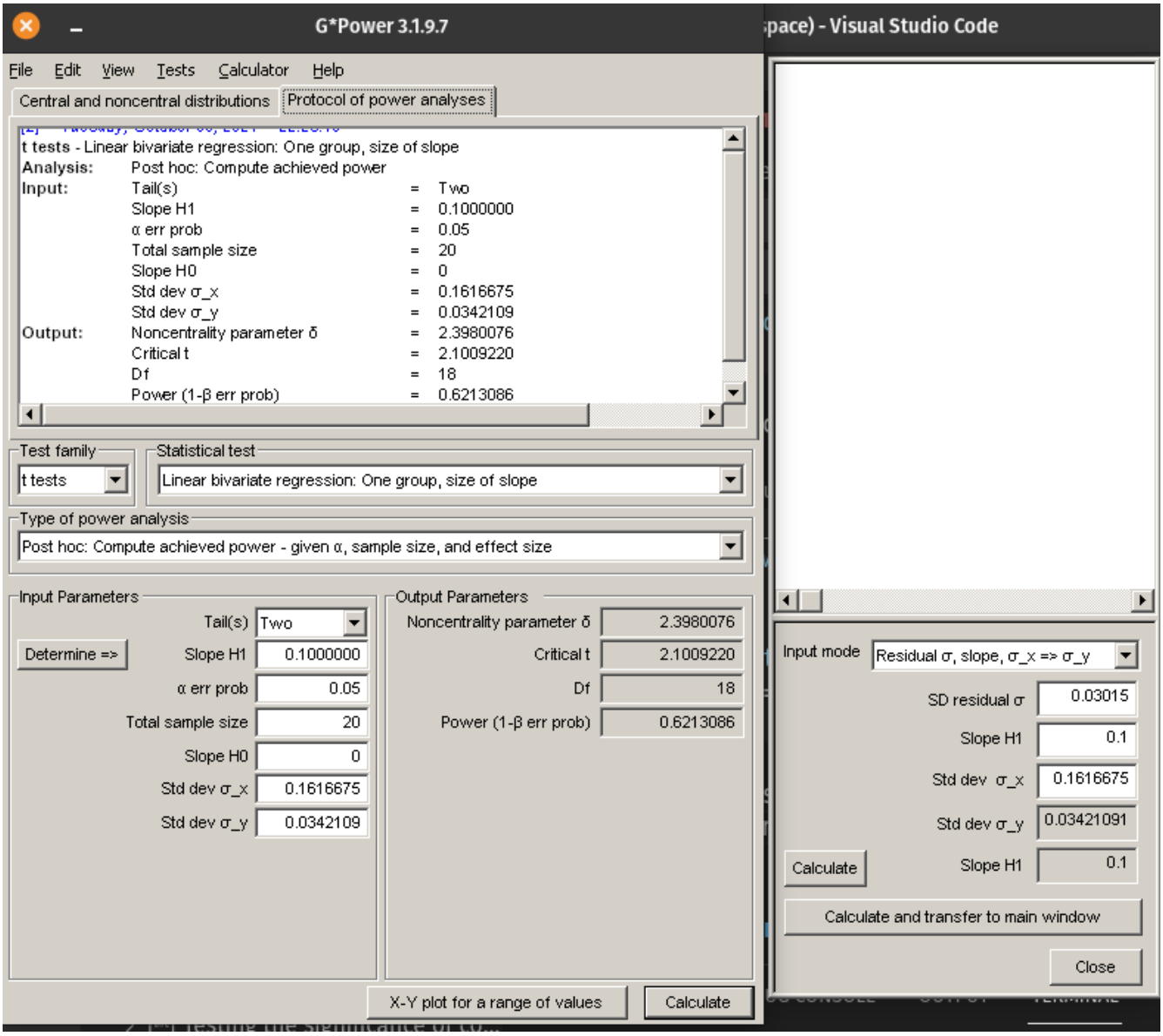


Figure 3.3.: Power analysis for age-length in sturgeon with N = 20 and slope = 0.1

Power to detect a significant slope, if the slope is 0.1, variability of data points around the regression is like in our sample, for a sample of 20 sturgeons, with $\alpha = 0.05$ is 0.621. Only about 2/3 of samples of that size would detect a significant effect of `age` on `fklngth`.

In R, you can do the analysis also but we will use another trick to work with the `pwr.t.test()` function. First we, need to estimate the effect size `d`. IN this case `d` is estimated as:

$$d = \frac{b}{s_b\sqrt{n-k-1}}$$

where b is the slope, s_b is the standard error on the slope, n is the number of observations and k is the number of independent variables (1 for simple liner regression).

SE of the slope is 0.02168. The model was fitted using 75 fishes (n=75). We can then estimate `d`.

$$d = \frac{b}{s_b\sqrt{n-k-1}} = \frac{0.1}{0.02168\sqrt{74-1-1}} = 0.54$$

We can simply use the `pwr.t.test()` function to estimate the power.

```
library(pwr)

# analyse de puissance
pwr.t.test(n = 20, d = 0.54, sig.level = 0.05, type = "one.sample")
```

```
     One-sample t test power calculation

              n = 20
              d = 0.54
      sig.level = 0.05
          power = 0.6299804
    alternative = two.sided
```

You can see that the results is really similar but not exactly the same than with G*power which is normal since we did not use the exact same formula to estimate power.

3.9.2. Sample size required to achieve desired power

To estimate the sample size required to achieve 99% power to detect a slope of 0.1 (in log-log space), with alpha=0.05, you simply change the type of analysis:

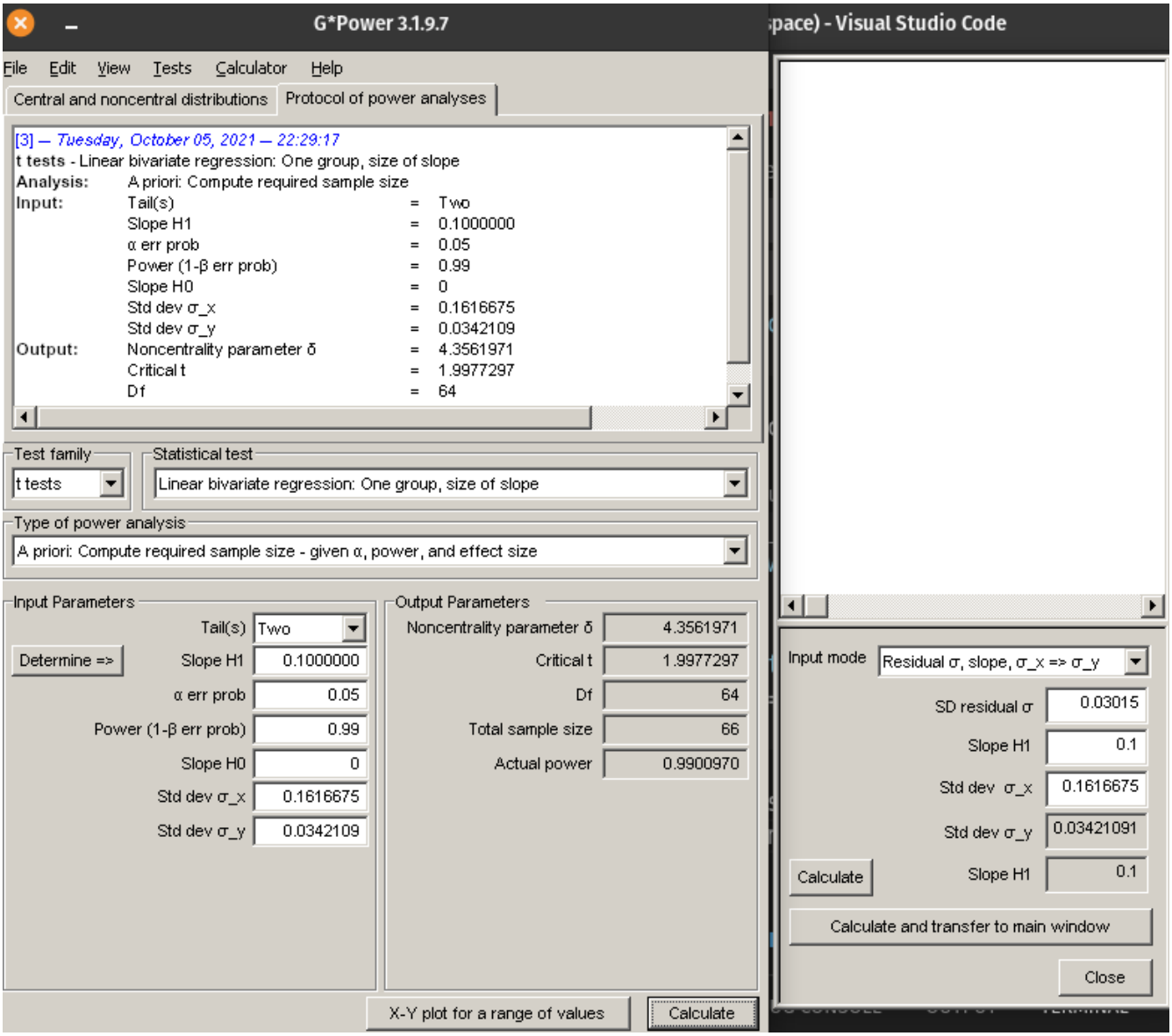


Figure 3.4.: A priori power analysis to estimate the sample size needed to have a power of 0.99

In R you can simply do:

```
library(pwr)

# analyse de puissance
pwr.t.test(power = 0.99, d = 0.54, sig.level = 0.05, type = "one.sample")
```

```
     One-sample t test power calculation

              n = 64.96719
              d = 0.54
```

```
      sig.level = 0.05
          power = 0.99
    alternative = two.sided
```

By increasing sample size to 66, with the same assumptions as before, power increases to 99%.

3.10. Bootstrapping the simple linear regression

A non-parametric test for the intercept and slope of a linear regression can be obtained by bootstrapping.

```
# load boot
library(boot)
# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices, ] # allows boot to select sample
  fit <- lm(formula, data = d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(
  data = sturgeon.male,
  statistic = bs,
  R = 1000, formula = log10(fklngth) ~ log10(age)
)
# view results
results
```

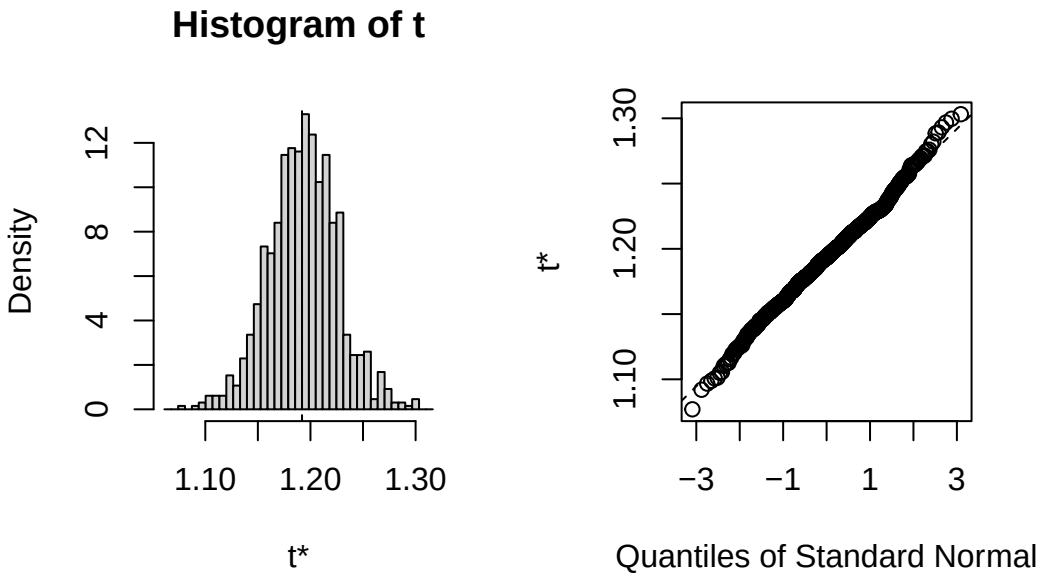
```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = sturgeon.male, statistic = bs, R = 1000, formula = log10(fklngth) ~
    log10(age))


Bootstrap Statistics :
     original        bias    std. error
t1* 1.1919926  0.0012889012  0.03287109
t2* 0.3408557 -0.0008819341  0.02595628
```

For each parameter in the model (here the intercept is labeled `t1\*` and the slope of the regression line is labeled `t2\*`) , you obtain:

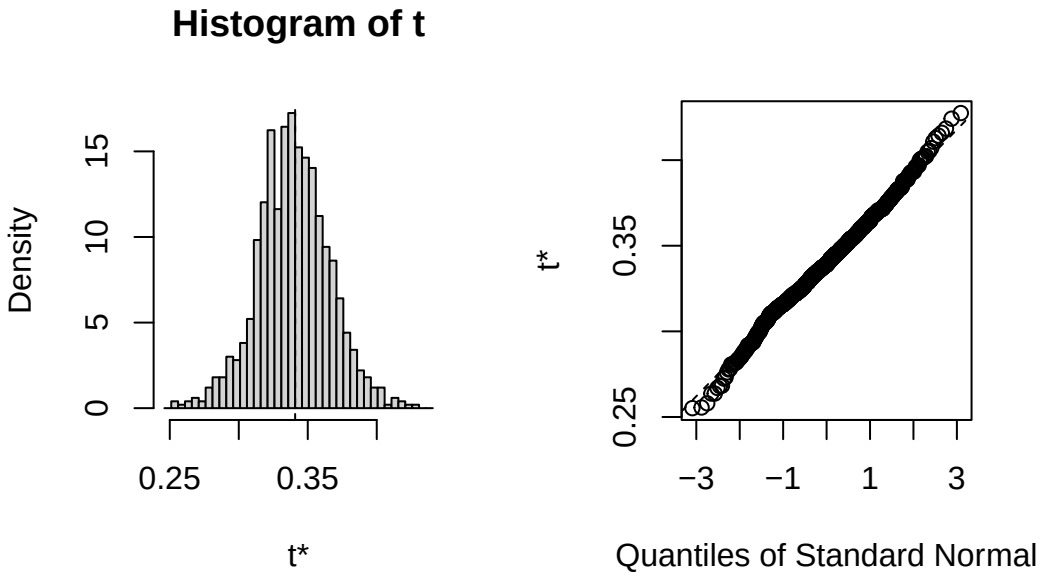
Pour chaque paramètre du modèle (ici l'ordonnée à l'origine est appelée t1* et la pente de la régression `t2\*`), R imprime :

1. `original` original parameter estimate (on all non-bootstrapped data)
2. `bias` the difference between the mean value of all bootstrap estimates and the original value
3. `std. error` standard error of the bootstrap estimate

```
par(mfrow = c(2, 2))
plot(results, index = 1) # intercept
```




```
plot(results, index = 2) # log10(age)
```



The distribution of the bootstrapped estimates is rather Gaussian, with only small deviations in the tails (where it counts for confidence intervals…). One could use the standard error of the bootstrap estimates to calculate a symmetrical confidence interval as mean +- t SE. But, given that R can as easily calculate a bias-corrected adjusted (BCa) confidence interval, or one based on the actual distribution, (Percentile) why not have it do it all:

```
# interval de confiance pour l'ordonnée à l'origine
boot.ci(results, type = "all", index = 1)
```

```
Warning in boot.ci(results, type = "all", index = 1): bootstrap variances
needed for studentized intervals

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "all", index = 1)

Intervals :
Level      Normal              Basic
95%   ( 1.126,  1.255 )   ( 1.120,  1.258 )

Level     Percentile            BCa
95%   ( 1.126,  1.264 )   ( 1.112,  1.249 )
Calculations and Intervals on Original Scale
```

```
# intervalle de confiance pour la pente
boot.ci(results, type = "all", index = 2)
```

```
Warning in boot.ci(results, type = "all", index = 2): bootstrap variances
needed for studentized intervals

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "all", index = 2)

Intervals :
Level      Normal              Basic
95%   ( 0.2909,  0.3926 )   ( 0.2888,  0.3962 )

Level     Percentile            BCa
95%   ( 0.2855,  0.3929 )   ( 0.2942,  0.4028 )
Calculations and Intervals on Original Scale
```

Here the 4 types of CI that R managed to calculate are essentially the same. Had data been violating more strongly the standard assumptions (normality, homoscedasticity), then the different methods (Normal, Basic, Percentile, and BCa) would have diverged more. In that case, which one is best? BCa has the favor of most, currently.

4. Two - sample comparisons

After completing this laboratory exercise, you should be able to:

- Use R to visually examine data.
- Use R to compare the means of two normally distributed samples.
- Use R to compare the means of two non-normally distributed samples.
- Use R to compare the means of two paired samples

4.1. R packages and data

For this la b you need:

- R packages:
  - car
  - lmtest
  - boot
  - pwr
  - ggplot2
  - performance
- data:
  - sturgeon.csv
  - skulldat_2020.csv

You need to load the packages in R with `library()` and if need needed install them first with `install.packages()` For the data, load them using the `read.csv()` function.

4.2. Visual examination of sample data

One of the first steps in any type of data analysis is to visualize your data with plots and summary statistics, to get an idea of underlying distributions, possible outliers, and trends in your data. This often begins with plots of the data, such as histograms, probability plots, and box plots, that allow you to get a feel for whether your data are normally distributed, whether they are correlated one to the other, or whether there are any suspicious looking points that may lead you to go back to the original data file to check for errors.

Suppose we want to test the null hypothesis that the size, as indexed by fork length (`fklngth` in file `sturgeon.csv` - the length, in cm, from the tip of the nose to the base of the fork in the caudal fin), of sturgeon at The Pas and Cumberland House is the same. To begin, we have a look at the underlying distributions of the sample data to get a feel for whether the data are normally distributed in each sample. We will not actually test for normality at this point; the assumption of normality in parametric analyses

refers always to the residuals and not the raw data themselves. However, if the raw data are non-normally distributed, then you usually have good reason to suspect that the residuals also will be non-normally distributed.

An excellent way to visually compare a data distribution to a normal distribution is to superimpose a histogram of the data and a normal curve. To do so, we must proceed in two steps:

1. tell R that we want to make a histogram with a density curve superimposed
2. tell R that we want this to be done for both locations.

- Using the data file `sturgeon.csv` , generate histograms for `fklngth` data at The Pas and Cumberland House.

```
# use "sturgeon" dataframe to make plot called mygraph
# and define x axis as representing fklngth
mygraph <- ggplot(
  data = sturgeon,
  aes(x = fklngth)
) +
  xlab("Fork length (cm)")
# add data to the mygraph ggplot
mygraph <- mygraph +
  geom_density() + # add data density smooth
  geom_rug() + # add rug (bars at the bottom of the plot)
  geom_histogram( # add black semitransparent histogram
    aes(y = ..density..),
    color = "black", alpha = 0.3
  ) +
  # add normal curve in red, with mean and sd from fklength
  stat_function(
    fun = dnorm,
    args = list(
      mean = mean(sturgeon$fklngth),
      sd = sd(sturgeon$fklngth)
    ),
    color = "red"
  )
# display graph, by location
mygraph + facet_grid(. ~ location)
```

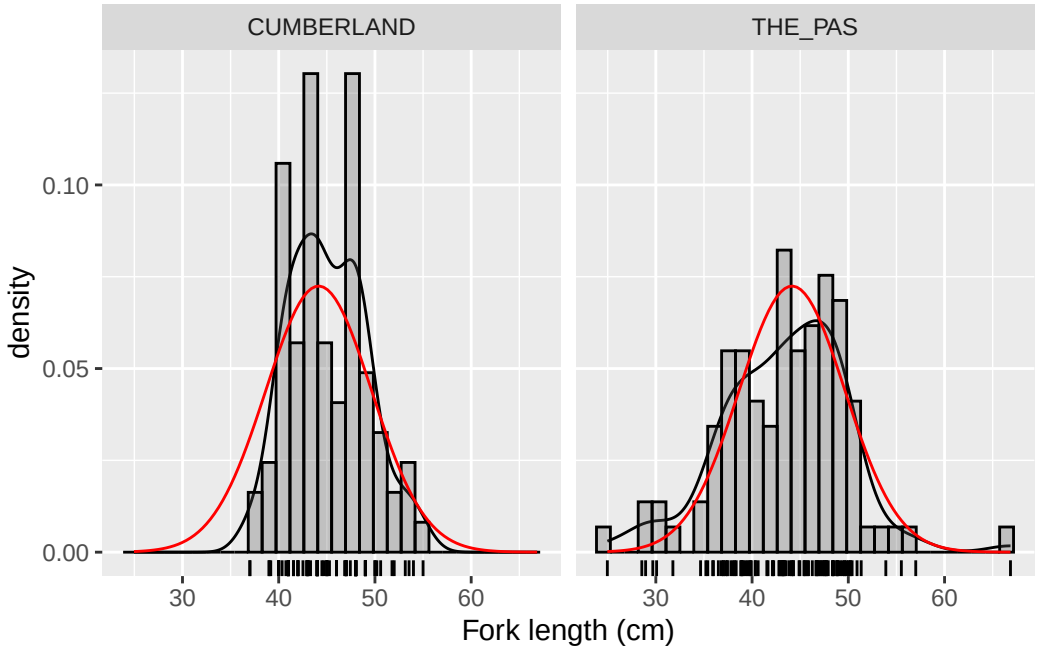


Figure 4.1.: Distribution of sturgeon length at 2 locations

Based on your visual inspection, are the two samples normally distributed? Visual inspection of these plots suggests that this variable is approximately normally distributed in each sample.

Since we are interested in finding out if mean fish size differs among the two locations, it is probably also a good idea to generate a graph that compares the two groups of data. A box plot works well for this.

- Generate a box plot of `fklngth` grouped by `location` . What do you conclude about differences in size among the two locations?

```
ggplot(data = sturgeon, aes(
  x = location,
  y = fklngth
)) +
  geom_boxplot(notch = TRUE)
```

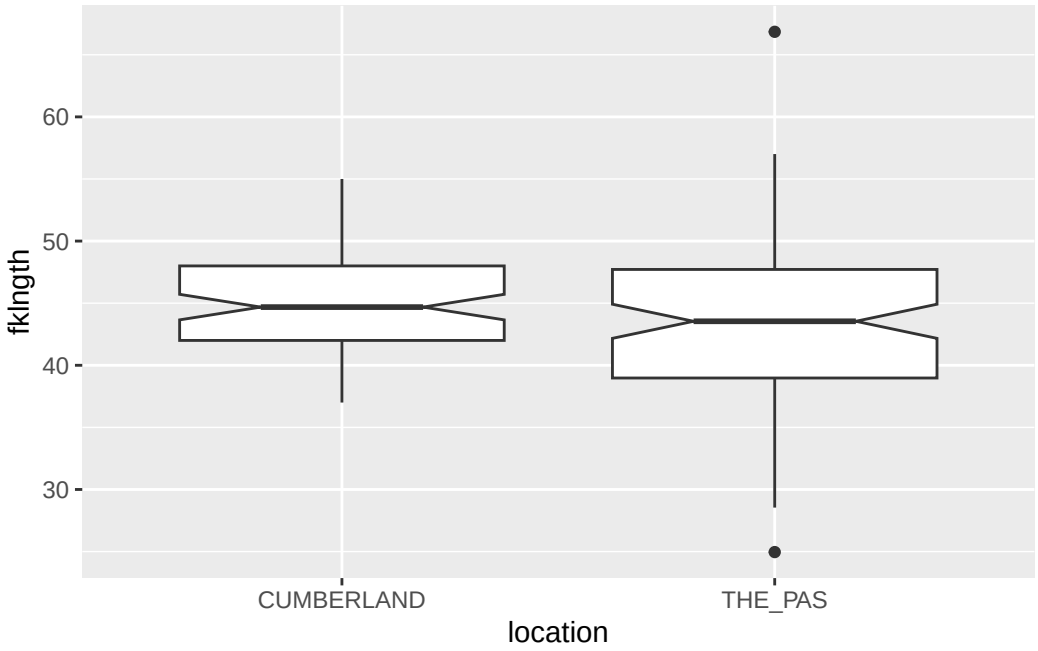


Figure 4.2.: Boxplot of sturgeon legnth at 2 locations

It would appear as though there are not big differences in fish size among the two locations, although fish size at The Pas looks to be more variable, with a bigger range in size and outliers (defined as values > 1.5 * inter-quartile range) at both ends of the distribution.

4.3. Comparing means of two independent samples: parametric and non-parametric comparisons

Test the null hypothesis that the mean fklngth of The Pas and Cumberland House samples are the same. Using 3 different tests:

1. parametric test with equal variances
2. parametric test with unequal variances
3. non-parametric test

What do you conclude?

```
# t-test assuming equal variances
t.test(
  fklngth ~ location,
  data = sturgeon,
  alternative = "two.sided",
  var.equal = TRUE
)
```

```
    Two Sample t-test

data:  fklngth by location
t = 2.1359, df = 184, p-value = 0.03401
alternative hypothesis: true difference in means between group CUMBERLAND and group THE_PAS is no
95 percent confidence interval:
 0.1308307 3.2982615
sample estimates:
mean in group CUMBERLAND    mean in group THE_PAS
                45.08439                 43.36984
```

```r
# t-test assuming unequal variances
t.test(
  fklngth ~ location,
  data = sturgeon,
  alternative = "two.sided",
  var.equal = FALSE
)
```

```
    Welch Two Sample t-test

data:  fklngth by location
t = 2.2201, df = 169.8, p-value = 0.02774
alternative hypothesis: true difference in means between group CUMBERLAND and group THE_PAS is no
95 percent confidence interval:
 0.1900117 3.2390804
sample estimates:
mean in group CUMBERLAND    mean in group THE_PAS
                45.08439                 43.36984
```

```r
# test non paramétrique
wilcox.test(
  fklngth ~ location,
  data = sturgeon,
  alternative = "two.sided"
)
```

```
    Wilcoxon rank sum test with continuity correction

data:  fklngth by location
W = 4973, p-value = 0.06296
alternative hypothesis: true location shift is not equal to 0
```

Based on the *t-test*, we would reject the null hypothesis, *i.e.* there is a significant (but not highly significant) difference in mean fork length between the two populations.

Note that using the Wilcoxon rank sum test, we do not reject the null hypothesis. The two different tests therefore give us two different results. The significant difference obtained using the t-test may, at least in part, be due to deviations from normality or homoscedasticity; on the other hand, the non-significant difference obtained using the U -statistic may be due to the fact that for fixed sample size, the power of a non-parametric test is lower than the corresponding parametric test. Given the p values obtained from both tests, and the fact that for samples of this size (84 and 101), the t-test is comparatively robust with respect to non-normality, I would be inclined to reject the null hypothesis. In practice to avoid P-hacking, you should decide which test is appropriate first and then apply and interpret it, or if you decide to do all you should present results of all and interpret accordingly.

Before accepting the results of the parametric t-test and rejecting the null hypothesis that there is no difference in size between the two locations, one should do some sort of assessment to determine if the model fits the assumption of normally distributed residuals and equal variances. Preliminary examination of the raw data suggested the data appeared roughly normal but there might be problems with variances (since the spread of data for The_Pas was much greater than for Cumberland). We can examine this more closely by looking at the residuals. An easy way to do so, is to fit a linear model and use the residual diagnostic plots:

```
m1 <- lm(fklngth ~ location, data = sturgeon)
par(mfrow = c(2, 2))
plot(m1)
```

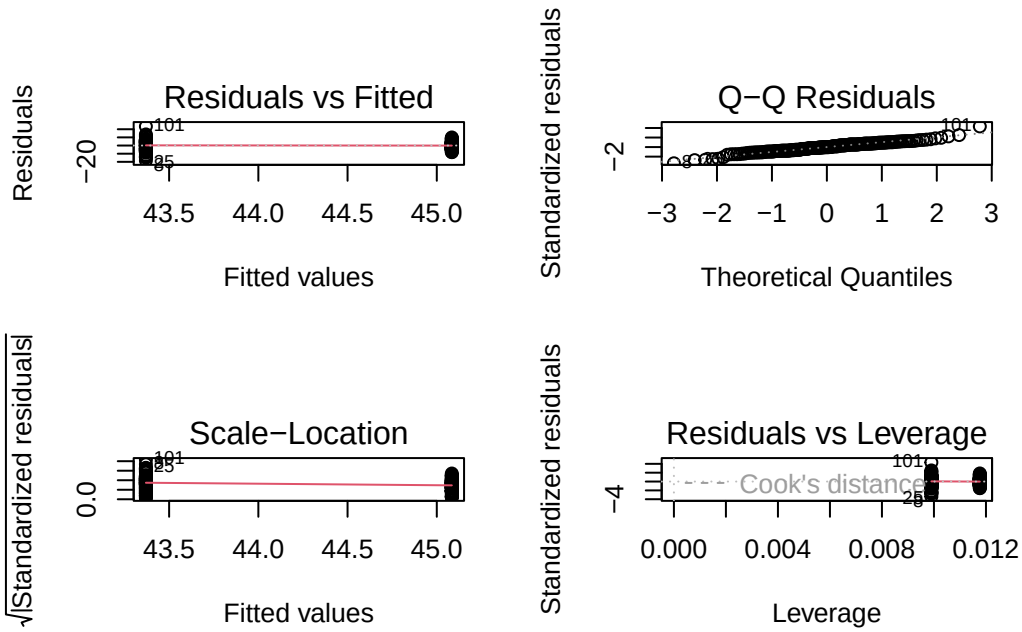


Figure 4.3.: Model assumption checks

The first plot above shows the spread of the residuals around the estimated values for the two groups and allows us to get a feel for whether there are problems with the assumption of homogeneity of variances. If

the variances were equal, the vertical spread of the two clusters of points should be about the same. The above plot shows that the vertical spread of the group with the smaller mean is greater than it is for the larger mean, suggesting again that there are problems with the variances. We can test this formally by examining the mean differences in the absolute value of the residuals.

The second graph above is a normal QQ plot (or probability plot) of the residuals of the model. Note that these generally fall on a straight line, suggesting there is no real problem with normality. We can do a formal test for normality on the residuals using the Shapiro-Wilk test.

```
shapiro.test(residuals(m1))
```

```
    Shapiro-Wilk normality test

data:  residuals(m1)
W = 0.97469, p-value = 0.001857
```

Hummm. The test indicates that the residuals are not normal. But, given that (a) the distribution is not very far (at least visually) from normal, and that (b) the number of observations in each location is reasonably large (i.e. >30), we do not need to be overly concerned with this violation of the normality assumption.

How about equality of variance?

```
library(car)
leveneTest(m1)
```

```
Warning in leveneTest.default(y = y, group = group, ...): group coerced to
factor.

Levene's Test for Homogeneity of Variance (center = median)
       Df F value    Pr(>F)
group   1  11.514 0.0008456 ***
      184
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
bptest(m1)
```

```
    studentized Breusch-Pagan test

data:  m1
BP = 8.8015, df = 1, p-value = 0.00301
```

The above are the results of two tests implemented in R (in the `car` and `lmtest` packages that can be used to test for equal variances in t-tests or linear models involving only discontinuous or categorical independent variables. **Doing the two of them is overkill**. There is not much to prefer one test over another. Levene test is possibly the better known. It tests whether the mean of absolute values of the residuals differs among groups. The Breusch-Pagan test has the advantage of being applicable to more linear models (it can deal with regression-type continuous independent variables, at least to some extent). It tests whether the studentized (i.e. scaled by their sd estimate) squared residuals vary with the independent variables in a linear model. In this case, both indicate that variances are unequal.

On the basis of these results, we conclude that there is evidence (albeit weak) to reject the null hypothesis of no difference in `fklngth` by `location`. We have modified the `t-test` to accommodate unequal variances, and are satisfied that the assumption of normally distributed residuals is sufficiently met. Thus, it appears that `fklngth` at Cumberland is greater than `fklngth` at The Pas.

4.4. Bootstrap and permutation tests to compare 2 means

4.4.1. Bootstrap

Bootstrap and permutation tests can be used to compare means (or other statistics) between pairs of samples. The general idea is simple, and it can be implemented in more ways than I can count. Here, I use existing tools and the fact that a comparison of means can be construed as a test of a linear model. We will be able to use similar code later on when we fit more complex (but fun!) models.

```
library(boot)
```

The first section defines the function that I called bs that simply extracts coefficients from a fitted model:

```
# function to obtain model coefficients for each iteration
bs <- function(formula, data, indices) {
  d <- data[indices, ]
  fit <- lm(formula, data = d)
  return(coef(fit))
}
```

The second section with the `boot()` command is where the real work is done: take data in sturgeon, bootstrap $R = 1000$ times, each time fit the model `fklngth` vs `location`, and keep the values calculated by the `bs()` function.

```
# bootstrapping with 1000 replications
results <- boot(
  data = sturgeon, statistic = bs, R = 1000,
  formula = fklngth ~ location
)
# view results
results
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = sturgeon, statistic = bs, R = 1000, formula = fklngth ~
    location)


Bootstrap Statistics :
     original      bias    std. error
t1* 45.084391 0.012024354   0.4125073
t2* -1.714546 0.001413805   0.7570054
```

So we get the original estimates for the two coefficients in this model: the mean at the first (alphabetical) location, Cumberland, and the difference in means between Cumberland and The Pas). It is the second parameter, the difference between means, which is of interest here.

```
plot(results, index = 2)
```

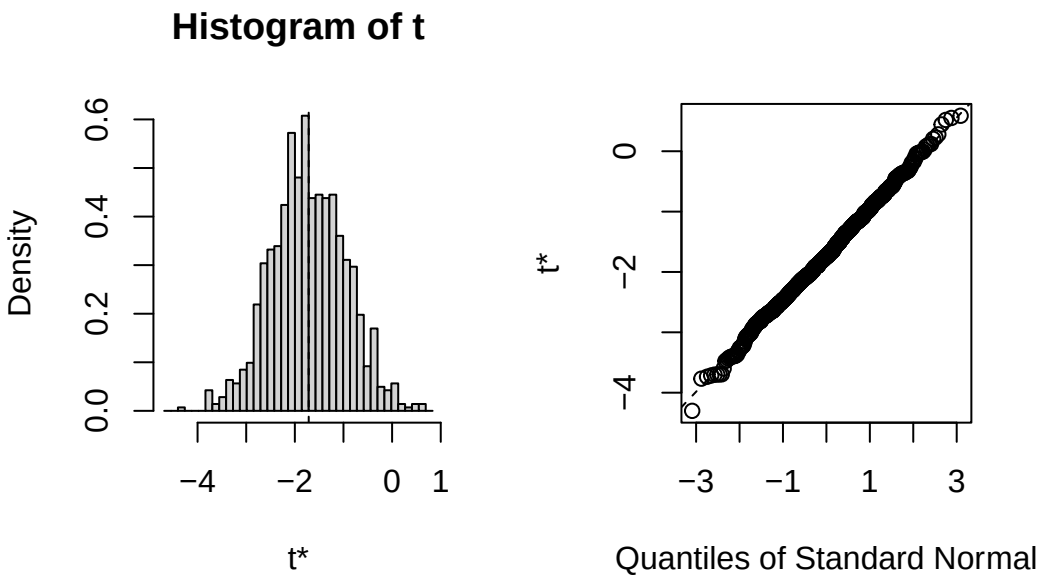


Figure 4.4.: Distribution of bootstrapped mean difference

```
# get 95% confidence intervals
boot.ci(results, type = "bca", index = 2)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
```

```
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 2)

Intervals :
Level       BCa
95%   (-3.104, -0.109 )
Calculations and Intervals on Original Scale
```

The 95% CI for the difference between the two means does not include 0. Hence, the bootstrap test indicates that the two means are not equals.

4.4.2. Permutation

Permutation tests for linear models can easily be done using the `lmPerm` package .

```
m1Perm <- lmp(
  fklngth ~ location,
  data = sturgeon,
  perm = "Prob"
)
```

```
[1] "Settings:  unique SS "
```

The `lmp()` function does all the work for us. Here it is run with the option perm to control the stopping rule used. Option Prob stops the sampling when the estimated standard deviation of the p-value falls below some fraction of the estimated. It is one of many stopping rules that one could use to do permutations on a subset of all the possibilities (because it would take foreeeever to do them all, even on your fast machine).

```
summary(m1Perm)
```

```
Call:
lmp(formula = fklngth ~ location, data = sturgeon, perm = "Prob")

Residuals:
      Min        1Q    Median        3Q       Max
-18.40921  -3.75370  -0.08439   3.76598  23.48055

Coefficients:
          Estimate Iter Pr(Prob)
location1   0.8573 4881   0.0201 *
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.454 on 184 degrees of freedom
Multiple R-Squared: 0.02419,    Adjusted R-squared: 0.01889
F-statistic: 4.562 on 1 and 184 DF,  p-value: 0.03401
```

1. `Iter` coefficient: the Prob stopping rule stopped after 4881 iterations. Note that this number will vary each time you run this snippet of code. These are random permutation results, so expect variability.
2. `Pr(Prob)` coefficient: The estimated probability associated to H0 is 0.0201 . The observed difference in fklngth between the two locations was larger than the permuted differences in about (1 - 0.0201= about 98%) of the 4881 cases. Mind you, 4881 permutations is not a large number, so small p values can't be expected to be very precise. If it is critical that you get more precise p values, more permutations would be needed. Two parameters can be tweaked: maxIter, the maximum number of iterations (default=5000), and Ca, that stops iterations when estimated standard error of the estimated p is less than Ca*p. Default 0.1.
3. `F-statistic`: The rest is the standard output for the model fitted to the data, with the standard parametric test. Here the p-value, assuming all assumptions are met, is 0.034.

4.5. Comparing the means of paired samples

In some experimental designs, individuals are measured twice: common examples are the measurement of the same individual at two different times during development, or of the same individual subjected to two different experimental treatments. In these cases, the two samples are not independent (they include the same individuals), and a paired comparison must be made.

The file `skulldat_2020.csv` shows measurements of lower face width of 15 North American girls measured at age 5 and again at age 6 years (data from Newman and Meredith, 1956).

- Let's first run a standard t-test comparing the face width at age 5 and 6, not taking into account that the data are not independent and that they are consecutive measurements on the same individuals.

```
skull <- read.csv("data/skulldat_2020.csv")
t.test(width ~ age,
  data = skull,
  alternative = "two.sided",
  paired = FALSE
)
```

```
    Welch Two Sample t-test

data:  width by age
t = -1.7812, df = 27.93, p-value = 0.08576
alternative hypothesis: true difference in means between group 5 and group 6 is not equal to 0
95 percent confidence interval:
 -0.43002624  0.03002624
```

```
sample estimates:
mean in group 5 mean in group 6
       7.461333        7.661333
```

So far, we specified the t-test using a `formula` notation as `y ~ x` where `y` is the variable for which we want to compare the means and `x` is a variable defining the groups. This works really well when the samples are not paired and when the data is presented in a *long* format. For example the`skull` data is presented in a long format and contains 3 variables:

- `width`: head width for each observations
- `age`: age at measurement 5 or 6
- `id`: person identity

```
head(skull)
```

```
  width age id
1  7.33   5  1
2  7.53   6  1
3  7.49   5  2
4  7.70   6  2
5  7.27   5  3
6  7.46   6  3
```

When data are paired, we need to indicate how they are paired. In the `skull`data, samples are paired by an individual identity, `id`, with mearurement taken at different ages. However, the function `t.test` does not cope well with this data structure. We need to transpose the data from a *long* to a *wide* format where we have a column per group, with the data of a given individual on the same line. Here is how we can do it.

```
skull_w <- data.frame(id = unique(skull$id))
skull_w$width5 <- skull$width[match(skull_w$id, skull$id) & skull$age == 5]
skull_w$width6 <- skull$width[match(skull_w$id, skull$id) & skull$age == 6]
head(skull_w)
```

```
  id width5 width6
1  1   7.33   7.53
2  2   7.49   7.70
3  3   7.27   7.46
4  4   7.93   8.21
5  5   7.56   7.81
6  6   7.81   8.01
```

Now, let's run the appropriate paired t-test. What do you conclude? Compare this with the previous result and explain any differences.

```
t.test(skull_w$width5, skull_w$width6,
  alternative = "two.sided",
  paired = TRUE
)
```

```
    Paired t-test

data:  skull_w$width5 and skull_w$width6
t = -19.72, df = 14, p-value = 1.301e-11
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.2217521 -0.1782479
sample estimates:
mean difference
           -0.2
```

The first analysis above assumes that the two samples of girls at age 5 and 6 are independent samples, whereas the second analysis assumes that the same girl is measured twice, once at age 5 and once at age 6 years.

Note that in the former case, we accept the null based on $p = 0.05$, but in the latter we reject the null. In other words, the appropriate (paired sample) test shows a very significant effect of age, whereas the inappropriate one does not. The reason is because there is a strong correlation between face width at age 5 and face width at age 6:

```
graphskull <- ggplot(data = skull_w, aes(x = width5, y = width6)) +
  geom_point() +
  labs(x = "Skull width at age 5", y = "Skull width at age 6") +
  geom_smooth() +
  scale_fill_continuous(low = "lavenderblush", high = "red")
graphskull
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

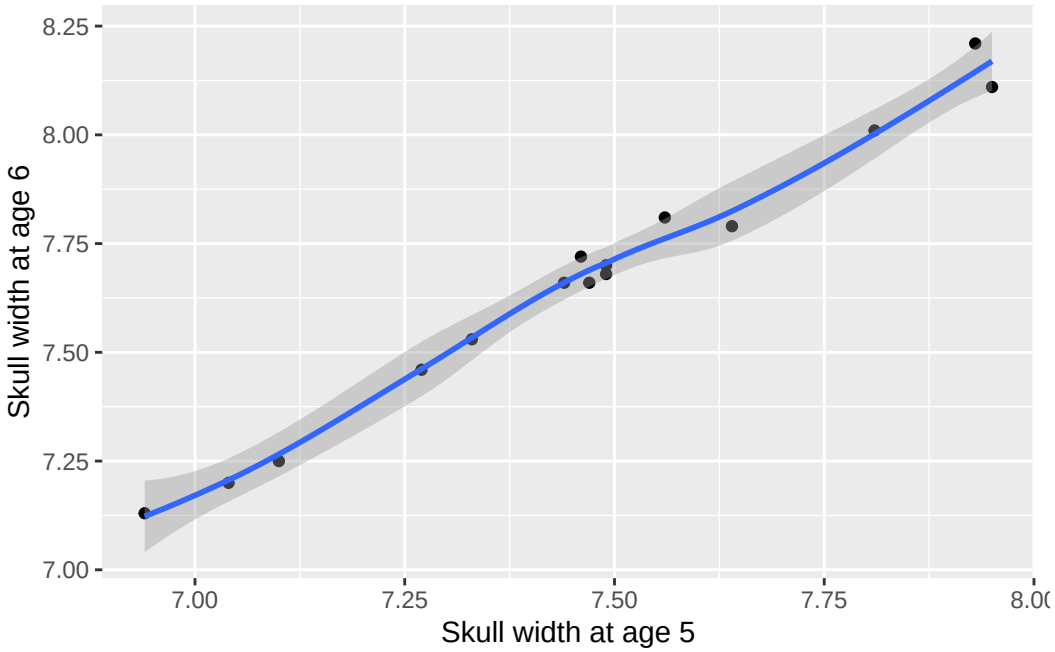


Figure 4.5.: Relation between head width at age 5 and 6

With $r = 0.9930841$. In the presence of correlation, the standard error of the pairwise difference in face width at age 5 and 6 is much smaller than the standard error of the difference between the mean face width at age 5 and 6. Thus, the associated t-statistic will be much larger for a paired sample test, *i.e.* the power of the test is much greater, and the p values are smaller.

- Repeat the above procedure with the nonparametric alternative, the Wilcoxon signed-rank test. What do you conclude?

```
wilcox.test(skull_w$width5, skull_w$width6,
  alternative = "two.sided",
  paired = TRUE
)
```

```
Warning in wilcox.test.default(skull_w$width5, skull_w$width6, alternative =
"two.sided", : cannot compute exact p-value with ties
```

```
    Wilcoxon signed rank test with continuity correction

data:  skull_w$width5 and skull_w$width6
V = 0, p-value = 0.0007193
alternative hypothesis: true location shift is not equal to 0
```

So, we reach the same conclusion as we did using the paired sample t-test and conclude there are significant differences in skull sizes of girls aged 5 and 6 (what a surprise!).

But, wait a minute. We have used two-tailed tests here. But, given what we know about how children grow, a one-tail hypothesis would be preferable. This can be done by changing the alternative option. One uses the alternative hypothesis to decide if it is "less" or greater". Here, we expect that if there is an effect (i.e the alternative hypothesis), width5 will be less than width6

```
t.test(skull_w$width5, skull_w$width6,
  alternative = "less",
  paired = TRUE
)
```

```
    Paired t-test

data:  skull_w$width5 and skull_w$width6
t = -19.72, df = 14, p-value = 6.507e-12
alternative hypothesis: true mean difference is less than 0
95 percent confidence interval:
       -Inf -0.1821371
sample estimates:
mean difference
           -0.2
```

```
wilcox.test(skull_w$width5, skull_w$width6,
  alternative = "less",
  paired = TRUE
)
```

```
Warning in wilcox.test.default(skull_w$width5, skull_w$width6, alternative =
"less", : cannot compute exact p-value with ties
```

```
    Wilcoxon signed rank test with continuity correction

data:  skull_w$width5 and skull_w$width6
V = 0, p-value = 0.0003597
alternative hypothesis: true location shift is less than 0
```

> **Note**
>
> Note that instead of rerunning the t-test specifying a one-tailed test, you can:
>
> - if the sign of the estimate goes in the same direction as the alternative hypothesis, simply divide by 2 the probability you obtain with the two-tailed test
> - if not the sign of the estimate is in the opposite direction of the alternative hypothesis, use $1 - p/2$

To estimate the power of a paired t-test in R, we can use the function `power.t.test()`as for other t-tests but we need to specify the argument `type = "paired"`. You need to specify the mean diference within the pairs as the `delta`and standard deviance of difference within pairs as `sd`.

```
skull_w$diff <- skull_w$width6 - skull_w$width5
power.t.test(
  n = 15,
  delta = mean(skull_w$diff),
  sd = sd(skull_w$diff),
  type = "paired")
```

```

     Paired t test power calculation

              n = 15
          delta = 0.2
             sd = 0.03927922
      sig.level = 0.05
          power = 1
    alternative = two.sided

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs
```

4.6. Bibliography

Bumpus, H.C. (1898) The elimination of the unfit as illustrated by the introduced sparrow, Passer domesticus. Biological Lectures, Woods Hole Biology Laboratory, Woods Hole, 11 th Lecture: 209 - 226.

Newman, K.J. and H.V. Meredith. (1956) Individual growth in skeletal bigonial diameter during the childhood period from 5 to 11 years of age. Amer. J. Anat. 99: 157 - 187.

5. One-way ANOVA

After completing this laboratory exercise, you should be able to:

- Use R to do a one-way parametric ANOVA with multiple comparisons
- Use R to test the validity of the parametric ANOVA assumptions
- Use R to perform a one-way non-parametric ANOVA
- Use R to transform your data so that the assumptions of parametric ANOVA are met.

5.1. R packages and data

For this lab you need:

- R packages:
  - ggplot2
  - multcomp
  - car
- data
  - dam10dat.csv

```
library(ggplot2)
library(car)
library(multcomp)
```

5.2. One-way ANOVA with multiple comparisons

The one-way ANOVA is the multi-group analog of the *t*-test, which is used to compare two groups/levels. It makes essentially the same assumptions, and in the case of two groups/levels, is in fact mathematically equivalent to the *t*-test.

In 1960-1962, the Grand Rapids Dam was built on the Saskatchewan River upstream of Cumberland House. There are anecdotal reports that during dam construction, a number of large sturgeon were stranded and died in shallow pools. Surveys of sturgeon were carried out in 1954, 1958, 1965 and 1966 with fork length (`fklngth`) and round weight (`rdwght`) being recorded (not necessarily both measurements for each individual). These data are in the data file `Dam10dat.csv`.

5.2.1. Visualiser les données

- Using `Dam10dat.csv`, you must first change the data type of the numerical variable year , so that R recognizes that we wish to treat this variable as a factor variable and not a continuous variable.

Solution

```
dam10dat <- read.csv("data/Dam10dat.csv")
dam10dat$year <- as.factor(dam10dat$year)
str(dam10dat)
```

```
'data.frame':   118 obs. of  21 variables:
 $ year    : Factor w/ 4 levels "1954","1958",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ fklngth : num  45 50 39 46 54.5 49 42.5 49 56 54 ...
 $ totlngth: num  49 NA 43 50.5 NA 51.7 45.5 52 60.2 58.5 ...
 $ drlngth : logi  NA NA NA NA NA NA ...
 $ drwght  : num  16 20.5 10 17.5 19.7 21.3 9.5 23.7 31 27.3 ...
 $ rdwght  : num  24.5 33 15.5 28.5 32.5 35.5 15.3 40.5 51.5 43 ...
 $ sex     : int  1 1 1 2 1 2 1 1 1 1 ...
 $ age     : int  24 33 17 31 37 44 23 34 33 47 ...
 $ lfkl    : num  1.65 1.7 1.59 1.66 1.74 ...
 $ ltotl   : num  1.69 NA 1.63 1.7 NA ...
 $ ldrl    : logi  NA NA NA NA NA NA ...
 $ ldrwght : num  1.2 1.31 1 1.24 1.29 ...
 $ lrdwght : num  1.39 1.52 1.19 1.45 1.51 ...
 $ lage    : num  1.38 1.52 1.23 1.49 1.57 ...
 $ rage    : int  4 6 3 6 7 7 4 6 6 7 ...
 $ ryear   : int  1954 1954 1954 1954 1954 1954 1954 1954 1954 1954 ...
 $ ryear2  : int  1958 1958 1958 1958 1958 1958 1958 1958 1958 1958 ...
 $ ryear3  : int  1966 1966 1966 1966 1966 1966 1966 1966 1966 1966 ...
 $ location: int  1 1 1 1 1 1 1 1 1 1 ...
 $ girth   : logi  NA NA NA NA NA NA ...
 $ lgirth  : logi  NA NA NA NA NA NA ...
```

- Next, have a look at the fklngth data, just as we did in the last lab for t-tests. Create a histogram with density line grouped by year to get a feel for what's happening with your data and a boxplot of length per year. What can you say about these data?

Solution

```
mygraph <- ggplot(dam10dat, aes(x = fklngth)) +
  labs(x = "Fork length (cm)") +
  geom_density() +
  geom_rug() +
  geom_histogram(aes(y = ..density..),
    color = "black",
    alpha = 0.3
  ) +
  stat_function(
    fun = dnorm,
    args = list(
      mean = mean(dam10dat$fklngth),
      sd = sd(dam10dat$fklngth)
    ),
    color = "red"
  )

# display graph, by year
mygraph + facet_wrap(~year, ncol = 2)
```

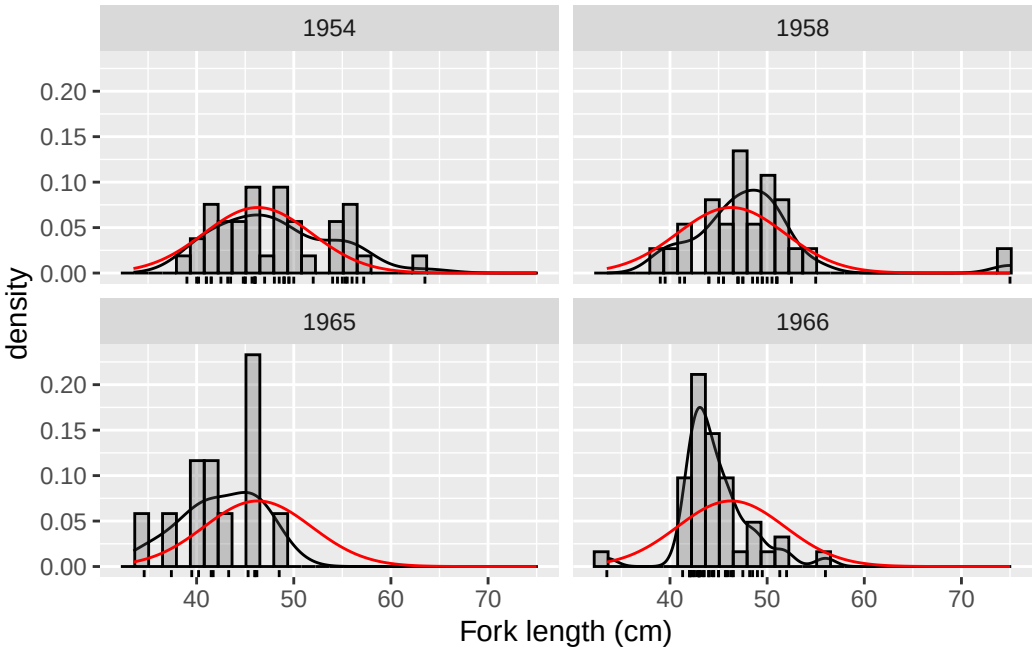


Figure 5.1.: Distribution of sturgeon length per year

```
boxplot(fklngth ~ year, data = dam10dat)
```

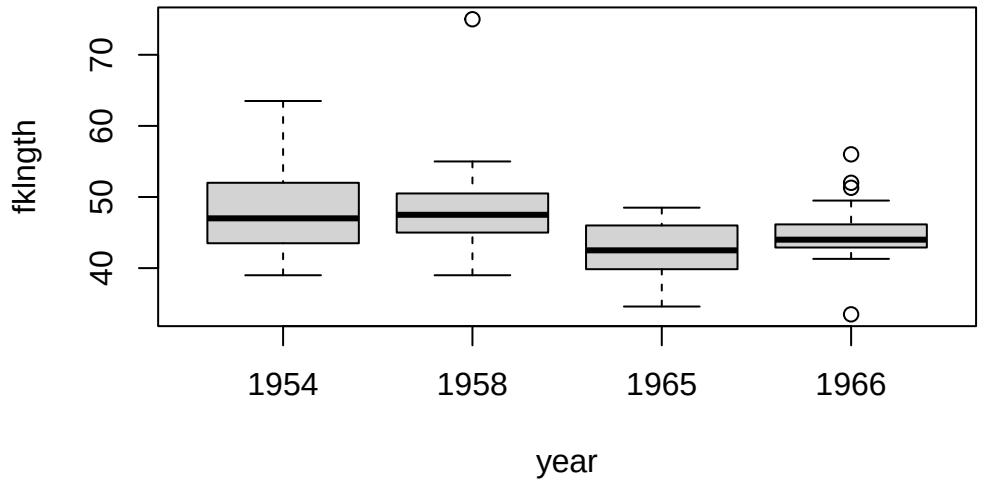


Figure 5.2.: Boxplot of sturgeon length per year

It appears as though there may have been a small drop in `fklngth` after the construction of the dam, but the data are variable and the effects are not clear. There might also be some problems with normality in the 1954 and 1966 samples, and it looks as though there are outliers in the 1958 and 1966 samples. Let's proceed with testing the assumptions of the ANOVA by running the analysis and looking at the residuals.

5.2.2. Testing the assumptions of a parametric ANOVA

Parametric one-way ANOVAs have three major assumptions:

1. the residuals are normally distributed
2. the error variance is the same for all groups (homoscedasticity)
3. the residuals are independent.

These assumptions must be tested before we can accept the results of any parametric ANOVA.

- Carry out a one-way ANOVA on fklngth by year and produce the residual diagnostic plots

```
# Fit anova model and plot residual diagnostics
anova.model1 <- lm(fklngth ~ year, data = dam10dat)
par(mfrow = c(2, 2))
plot(anova.model1)
```

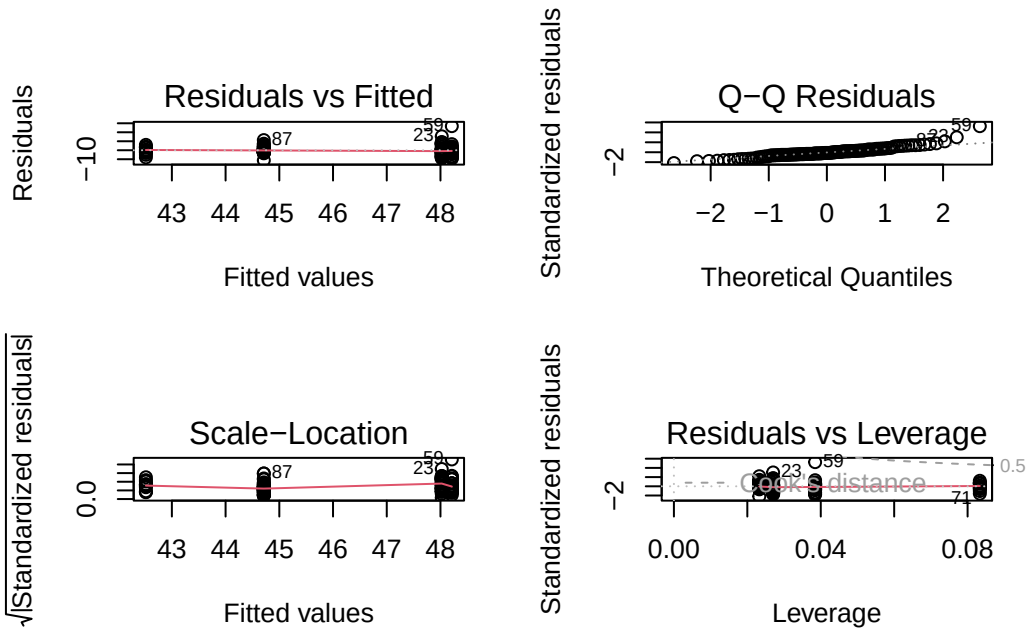


Figure 5.3.: Diagnostic plots for a one-way ANOVA

> **Warning**
>
> Double check that the independent variable is a **factor**. If the dependent variable is a **character**, then you will obtain only 3 graphs and an error message like:
> 'hat values (leverages) are all = 0.1
> and there are no factor predictors; no plot no. 5'

D'après les graphiques, on peut douter de la normalité et de l'homogénéité des variances. Judging from the plots, it looks as though there may be problems with both normality and variance heterogeneity. Note that there is one point (case 59) with large expected values and a large residual that appear to lie well off the line: this is the outlier we noted earlier. This point might be expected to inflate the variance for the group it belongs to. Formal tests may also provide some insight as to whether we should be concerned about normality and variance heterogeneity.

- Perform a normality test on the residuals from the ANOVA.

```
shapiro.test(residuals(anova.model1))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model1)
W = 0.91571, p-value = 1.63e-06
```

This test confirms our suspicions from the probability plot: the residuals are not normally distributed. Recall, however, that the power here is high, so only small deviations from normality are required to reject the null.

- Next, test for homoscedasticity:

```
leveneTest(fklngth ~ year, data = dam10dat)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value  Pr(>F)
group   3  2.8159 0.04234 *
      114
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The probability value tells you that you can reject the null hypothesis that there is no difference in variances among years. Thus, we conclude there is evidence that the variances in the groups are not equal.

5.2.3. Performing the ANOVA

Let's look at the results of the ANOVA, assuming for the moment that assumptions are met well enough.

```
summary(anova.model1)
```

```
Call:
lm(formula = fklngth ~ year, data = dam10dat)

Residuals:
     Min       1Q   Median       3Q      Max
-11.2116  -2.6866  -0.7116   2.2103  26.7885

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  48.0243     0.8566  56.061  < 2e-16 ***
year1958      0.1872     1.3335   0.140  0.88859
year1965     -5.5077     1.7310  -3.182  0.00189 **
year1966     -3.3127     1.1684  -2.835  0.00542 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.211 on 114 degrees of freedom
Multiple R-squared:  0.1355,    Adjusted R-squared:  0.1128
F-statistic: 5.957 on 3 and 114 DF,  p-value: 0.0008246
```

- *Coefficients: Estimates* Note the 4 coefficients printed. They can be used to obtain the predicted values for the model (i.e. the group means). The mean fklngth for the first year (1954) is 48.0243. The coefficients for the 3 other years are the difference between the mean for that year and for

1954. So, the mean for 1965 is (48.0243-5.5077=42.5166). For each estimated coefficient, there is a standard error, a t-value and associated probability (for H0 that the coefficient is 0). Note here that coefficients for 1965 and 1966 are both negative and significantly less than 0. Fish were smaller after the construction of the dam than in 1954. Take these p-values with a grain of salt: these are not corrected for multiple comparisons, and they constitute only a subset of the possible comparisons. In general, I pay little attention to this part of the output and look more at what comes next.

- *Residual standard error*: The square root of the variance of the residuals (observed minus fitted values) corresponds to the amount of variability that is unexplained by the models (here an estimate of how much size varied among fish, once corrected for differences among years)
- *Mutiple R-squared* The R-squared is the proportion of the variance of the dependent variable that can be explained by the model. Here the model explains only 13.5% of the variability. Size differences among year are relatively small compared to the ranges of sizes that can occur within years. This corresponds well to the visual impression left by the histograms of `fklngth` per `year`

4. *F-Statistic* This is the p-value for the "omnibus" test, the test that all means are equal. Here it is much smaller than 0.05 and hence we would reject H0 and conclude that fklngth varies among the years

The `anova()` command produces the standard ANOVA table that contains most of the same information:

```
anova(anova.model1)
```

```
Analysis of Variance Table

Response: fklngth
           Df  Sum Sq Mean Sq F value    Pr(>F)
year        3  485.26 161.755  5.9574 0.0008246 ***
Residuals 114 3095.30  27.152
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The total variability in fklngth sums of square is partitioned into what can be accounted for by year (485.26) and what is left unexplained as residual variability (3095.30). Year indeed explains (485.26/(3095.30 + 485.26) = .1355 or 13.55% of the variability). The mean square of the residuals is their variance.

5.2.4. Performing multiple comparisons of means test

- The `pairwise.t.test()` function can be used to compare means and adjust (or not) probabilities for multiple comparisons by choosing one of the options for the argument `p.adj`:

Comparing all means without corrections for multiple comparisons.

```
pairwise.t.test(dam10dat$fklngth, dam10dat$year,
  p.adj = "none"
)
```

```
    Pairwise comparisons using t tests with pooled SD

data:  dam10dat$fklngth and dam10dat$year

     1954   1958   1965
1958 0.8886 -      -
1965 0.0019 0.0022 -
1966 0.0054 0.0079 0.1996

P value adjustment method: none
```

Option `"bonf"` adjusts the p-values according to the Bonferroni correction. In this case, since there are 6 p-values calculated, it amounts to simply multiplying the uncorrected p-values by 6 (unless the result is above 1, in that case the adjusted p-value is 1).

```
pairwise.t.test(dam10dat$fklngth, dam10dat$year,
  p.adj = "bonf"
)
```

```
    Pairwise comparisons using t tests with pooled SD

data:  dam10dat$fklngth and dam10dat$year

     1954  1958  1965
1958 1.000 -     -
1965 0.011 0.013 -
1966 0.033 0.047 1.000

P value adjustment method: bonferroni
```

Option `"holm"` is the sequential Bonferroni correction, where the p-values are ranked from (i=1) smallest to (N) largest. The correction factor for p-values is then $(N - i + 1). Here, for example, we have N=6 pairs that are compared. The lowest uncorrected p-value is 0.0019 for 1954 vs 1965. The corrected p-value becomes $0.0019 * (6 - 1 + 1) = 0.011$. The second lowest p-value is 0.0022. The corrected p/value is therefore $0.0022 * (6 - 2 + 1) = 0.011$. For the highest p-value, the correction is $(N - N + 1) = 1$, hence it is equal to the uncorrected probability.

```
pairwise.t.test(dam10dat$fklngth, dam10dat$year,
  p.adj = "holm"
)
```

```
    Pairwise comparisons using t tests with pooled SD
```

```
data:  dam10dat$fklngth and dam10dat$year

     1954  1958  1965
1958 0.889 -     -
1965 0.011 0.011 -
1966 0.022 0.024 0.399

P value adjustment method: holm
```

The "fdr" option is for controlling the false discovery rate.

```
pairwise.t.test(dam10dat$fklngth, dam10dat$year,
  p.adj = "fdr"
)
```

```
    Pairwise comparisons using t tests with pooled SD

data:  dam10dat$fklngth and dam10dat$year

     1954   1958   1965
1958 0.8886 -      -
1965 0.0066 0.0066 -
1966 0.0108 0.0119 0.2395

P value adjustment method: fdr
```

The four post-hoc tests here tell us the same thing: differences are all between two groups of years: 1954/58 and 1965/66, since all comparisons show differences between the 50's and 60's but no differences within the 50's or 60's. So, in this particular case, the conclusion is not affected by the choice of adjustment method. But in other situations, you will observe contradictory results.

Which one to choose? Unadjusted p-values are certainly suspect when there are multiple tests. On the other hand, the traditional *Bonferroni* correction is very conservative, and becomes even more so when there are a large number of comparisons. Recent work suggest that the *fdr* approach may be a good compromise when there are a lot of comparisons. The *Tukey* method of multiple comparisons is one of the most popular and is easily performed with R (note, however, that there is a pesky bug that manifests itself when the independent variable can look like a number rather than a factor, hence the little pirouette with `paste0()` to add a letter `m` before the first digit):

```
dam10dat$myyear <- as.factor(paste0("m", dam10dat$year))
TukeyHSD(aov(fklngth ~ myyear, data = dam10dat))
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level
```

```
Fit: aov(formula = fklngth ~ myyear, data = dam10dat)

$myyear
                   diff        lwr        upr     p adj
m1958-m1954  0.1872141  -3.289570  3.6639986 0.9990071
m1965-m1954 -5.5076577 -10.021034 -0.9942809 0.0100528
m1966-m1954 -3.3126964  -6.359223 -0.2661701 0.0274077
m1965-m1958 -5.6948718 -10.436304 -0.9534397 0.0116943
m1966-m1958 -3.4999106  -6.875104 -0.1247171 0.0390011
m1966-m1965  2.1949612  -2.240630  6.6305526 0.5710111
```

```
par(mar = c(4, 7, 2, 1))
plot(TukeyHSD(aov(fklngth ~ myyear, data = dam10dat)), las = 2)
```

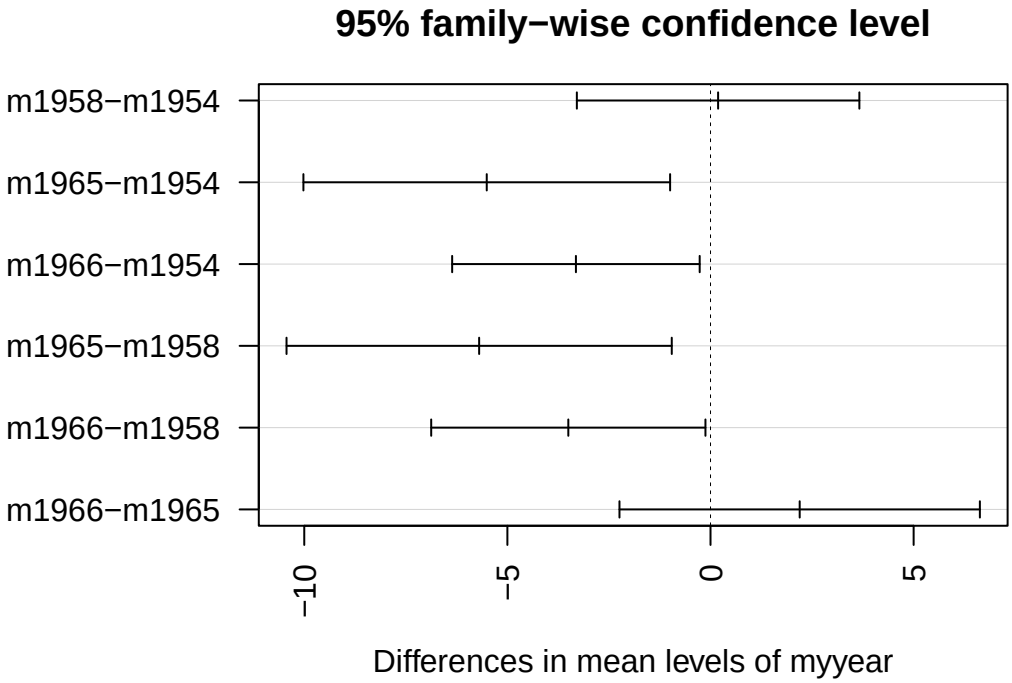


Figure 5.4.: Inter-annual differences in sturgeon length

The confidence intervals, corrected for multiple tests by the Tukey method, are plotted for differences among years. Unfortunately, the labels are not all printed because they would overlap, but the order is the same as in the preceding table. The `multcomp` can produce a better plot version, but requires a bit more code:

```
# Alternative way to compute Tukey multiple comparisons
# set up a one-way ANOVA
anova_fkl_year <- aov(fklngth ~ myyear, data = dam10dat)
# set up all-pairs comparisons for factor `year'
```

```
meandiff <- glht(anova_fkl_year, linfct = mcp(
  myyear =
    "Tukey"
))
confint(meandiff)
```

```
     Simultaneous Confidence Intervals

Multiple Comparisons of Means: Tukey Contrasts


Fit: aov(formula = fklngth ~ myyear, data = dam10dat)

Quantile = 2.5931
95% family-wise confidence level


Linear Hypotheses:
                    Estimate lwr      upr
m1958 - m1954 == 0   0.1872  -3.2706   3.6451
m1965 - m1954 == 0  -5.5077  -9.9965  -1.0189
m1966 - m1954 == 0  -3.3127  -6.3426  -0.2828
m1965 - m1958 == 0  -5.6949 -10.4105  -0.9793
m1966 - m1958 == 0  -3.4999  -6.8567  -0.1431
m1966 - m1965 == 0   2.1950  -2.2165   6.6064
```

```
par(mar = c(5, 7, 2, 1))
plot(meandiff)
```

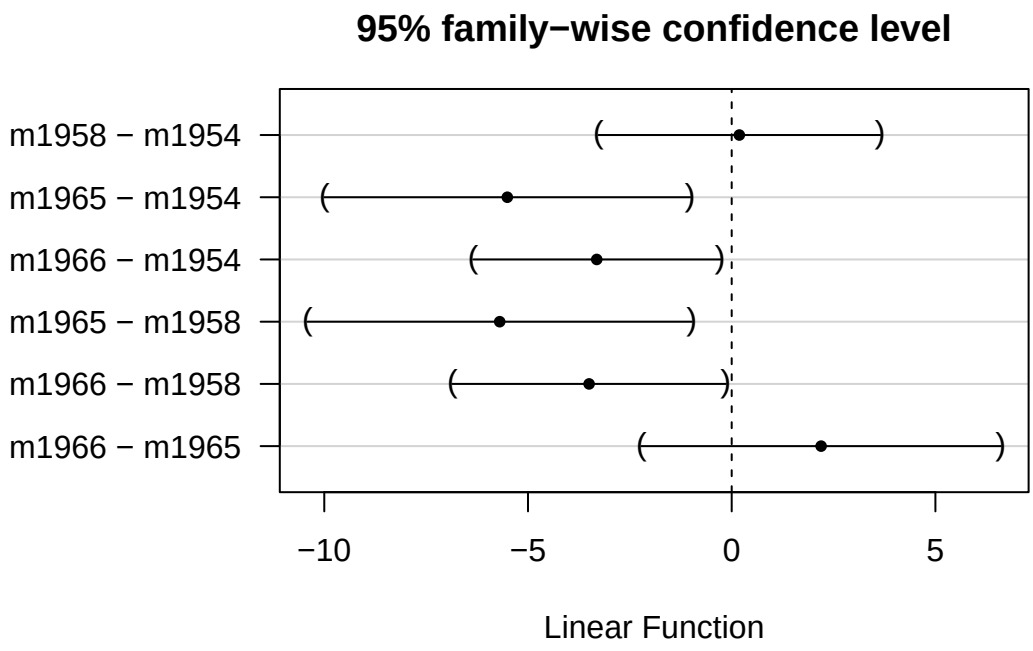


Figure 5.5.: Inter-annual differences in sturgeon length

This is better. Also useful is a plot the means and their confidence intervals with the Tukey groupings shown as letters above:

```
# Compute and plot means and Tukey CI
means <- glht(
  anova_fkl_year,
  linfct = mcp(myyear = "Tukey")
)
cimeans <- cld(means)
# use sufficiently large upper margin
# plot
old_par <- par(mai = c(1, 1, 1.25, 1))
plot(cimeans)
```

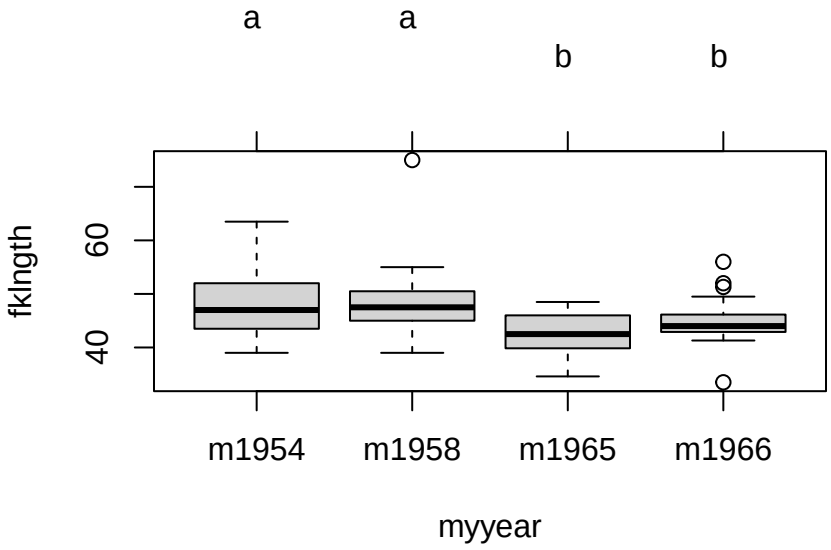


Figure 5.6.: Inter-annual differences in sturgeon length

Note the letters appearing on top. Years labelled with the same letter do not differ significantly.

5.3. Data transformations and non-parametric ANOVA

In the above example to examine differences in fklngth among years , we detected evidence of non-normality and variance heterogeneity. If the assumptions underlying a parametric ANOVA are not valid, there are several options:

1. if sample sizes in each group are reasonably large, parametric ANOVA is reasonably robust with respect to the normality assumption, for the same reason that the t-test is, so the results are probably not too bad;
2. we can transform the data;
3. we can go the non-parametric route.

- Repeat the one-way ANOVA in the section above, but this time run the analysis on the log 10 fklngth . With this transformation, do some of the problems encountered previously disappear?

```
# Fit anova model on log10 of fklngth and plot residual diagnostics
par(mfrow = c(2, 2))
anova.model2 <- lm(log10(fklngth) ~ year, data = dam10dat)
plot(anova.model2)
```

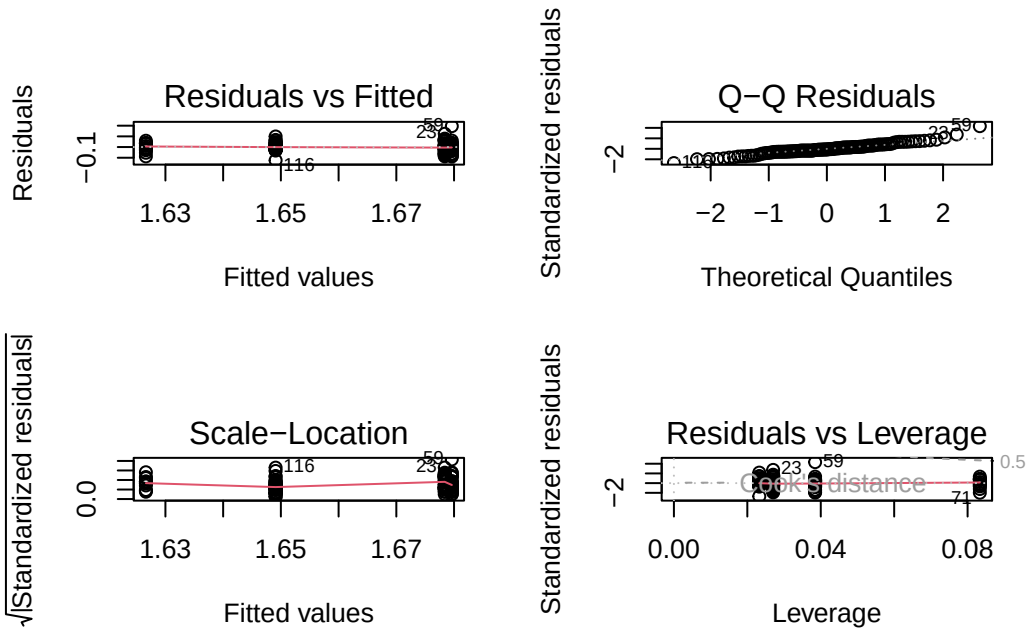


Figure 5.7.: Diagnostic plots for the ANOVA of sturgeon length by year

Looking at the residuals, things look barely better than before without the log transformation. Running the Wilks-Shapiro test for normality on the residuals, we get:

```
shapiro.test(residuals(anova.model2))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model2)
W = 0.96199, p-value = 0.002048
```

So, it would appear that we still have some problems with the assumption of normality and are just on the border line of meeting the assumption of homogeneity of variances. You have several choices here:

1. try to find a different transformation to satisfy the assumptions,
2. assume the data are close enough to meeting the assumptions, or
3. perform a non-parametric ANOVA.

- The most commonly used non-parametric analog of the parametric one-way ANOVA is the Kruskall-Wallis one-way ANOVA. Perform a Kruskall-Wallis one-way ANOVA of `fklngth` , and compare these results to the parametric analysis above. What do you conclude?

```
kruskal.test(fklngth ~ year, data = dam10dat)
```

```
    Kruskal-Wallis rank sum test

data:  fklngth by year
Kruskal-Wallis chi-squared = 15.731, df = 3, p-value = 0.001288
```

So, the conclusion is the same as with the parametric ANOVA: we reject the null that the mean rank is the same for each year. Thus, despite violation of one or more assumptions, the parametric analysis is telling us the same thing as the non-parametric analysis: the conclusion is, therefore, quite robust.

5.4. Dealing with outliers

Our preliminary analysis of the relationship between `fklngth` and `year` suggested there might be some outliers in the data. These were evident in the box plots of `fklngth` by `year` and flagged as cases 59, 23 and 87 in the residual probability plot and residual-fit plot. In general, you have to have very good reasons for removing outliers from a data set (e.g., you know there was a mistake made in the data collection/entry). However, it is often useful to know how the analysis changes if you remove the outliers from the data set.

- Repeat the original ANOVA of `fklngth` by `year` but work with a subset of the data without the outliers. Have any of the conclusions changed?

```
damsubset <- dam10dat[-c(23, 59, 87), ] # removes obs 23, 59 and 87
aov_damsubset <- aov(fklngth ~ as.factor(year), damsubset)
summary(aov_damsubset)
```

```
                Df Sum Sq Mean Sq F value   Pr(>F)
as.factor(year)   3  367.5  122.50   6.894 0.000267 ***
Residuals       111 1972.4   17.77
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
shapiro.test(residuals(aov_damsubset))
```

```
    Shapiro-Wilk normality test

data:  residuals(aov_damsubset)
W = 0.98533, p-value = 0.2448
```

```
leveneTest(fklngth ~ year, damsubset)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value   Pr(>F)
group   3  4.6237 0.004367 **
      111
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Elimination of three outliers, in this case, makes things better in terms of the normality assumption, but does not improve the variances. Moreover, the fact that the conclusion drawn from the original ANOVA with outliers retained does not change upon their removal reinforces the fact that there is no good reason to remove the points. Instead of a Kruskall-Wallis rank-based test, a permutation test could be used.

5.5. Permutation test

This is an example for a more complex way of doing permutation that we used when `lmPerm` was not available.

```
############################################################
# Permutation Test for one-way ANOVA
# modified from code written by David C. Howell
# http://www.uvm.edu/~dhowell/StatPages/
# More_Stuff/Permutation%20Anova/PermTestsAnova.html
# set desired number of permutations
nreps <- 500
# to simplify reuse of this code, copy desired dataframe to mydata
mydata <- dam10dat
# copy model formula to myformula
myformula <- as.formula("fklngth ~ year")
# copy dependent variable vector to mydep
mydep <- mydata$fklngth
# copy independent variable vector to myindep
myindep <- as.factor(mydata$year)
################################################
# You should not need to modify code chunk below
################################################
# Compute observed F value for original sample
mod1 <- lm(myformula, data = mydata) # Standard Anova
sum_anova <- summary(aov(mod1)) # Save summary to variable
obs_f <- sum_anova[[1]]$"F value"[1] # Save observed F value
# Print standard ANOVA results
cat(
  " The standard ANOVA for these data follows ",
  "\n"
)

print(sum_anova, "\n")
cat("\n")
```

```
cat("\n")
print("Resampling as in Manly with unrestricted sampling of observations. ")

# Now start resampling
boot_f <- numeric(nreps) # initalize vector to receive permuted
values
boot_f[1] <- obs_f
for (i in 2:nreps) {
  newdependent <- sample(mydep, length(mydep)) # randomize dep
  var
  mod2 <- lm(newdependent ~ myindep) # refit model
  b <- summary(aov(mod2))
  boot_f[i] <- b[[1]]$"F value"[1] # store F stats
}
permprob <- length(boot_f[boot_f >= obs_f]) / nreps
cat(
  " The permutation probability value is: ", permprob,
  "\n"
)
# end of code chunk for permutation
```

Version `lmPerm` du test de permutation.

```
## lmPerm version of permutation test
library(lmPerm)
# for generality, copy desired dataframe to mydata
# and model formula to myformula
mydata <- dam10dat
myformula <- as.formula("fklngth ~ year")
# Fit desired model on the desired dataframe
mymodel <- lm(myformula, data = mydata)
# Calculate permutation p-value
anova(lmp(myformula, data = mydata, perm = "Prob", center = FALSE, Ca = 0.001))
```

6. Multiway ANOVA: factorial and nested designs

After completing this laboratory exercise, you should be able to:

- Use R to do parametric ANOVAs for 2-way factorial designs with replication.
- Use R to do 2-way factorial design ANOVA without replication
- Use R to do parametric ANOVAs for nested designs with replication.
- Use R to do non-parametric 2-way ANOVAs
- Use R to do multiway pairwise comparisons

Be aware that there are a large number of possible ANOVA designs, many of which can be handled by R: this laboratory is

6.1. R packages and data needed

For this lab you need:

- R packages:
  - tidyverse
  - multicomp
  - car
  - effects
- data files:
  - Stu2wdat.csv
  - Stu2mdat.csv
  - nr2wdat.csv
  - nestdat.csv
  - wmcdat2.csv
  - wmc2dat2.csv

```
library(multcomp)
library(car)
library(tidyverse)
library(effects)
```

6.2. Two-way factorial design with replication

Many experiments are designed to investigate the joint effects of several different factors: in a two-way ANOVA, we examine the effect of two factors, but in principle the analysis can be extended to three, four or even five factors, although interpreting the results from 4- and 5-way ANOVAs can be very difficult.

Suppose that we are interested in the effects of two factors: `location` (Cumberland House and The Pas) and `sex` (male or female) on sturgeon size (data can be found in `Stu2wdat.csv`). Note that because the sample sizes are not the same for each group, this is an unbalanced design. Note also that there are missing data for some of the variables, meaning that not every measurement was made on every fish.

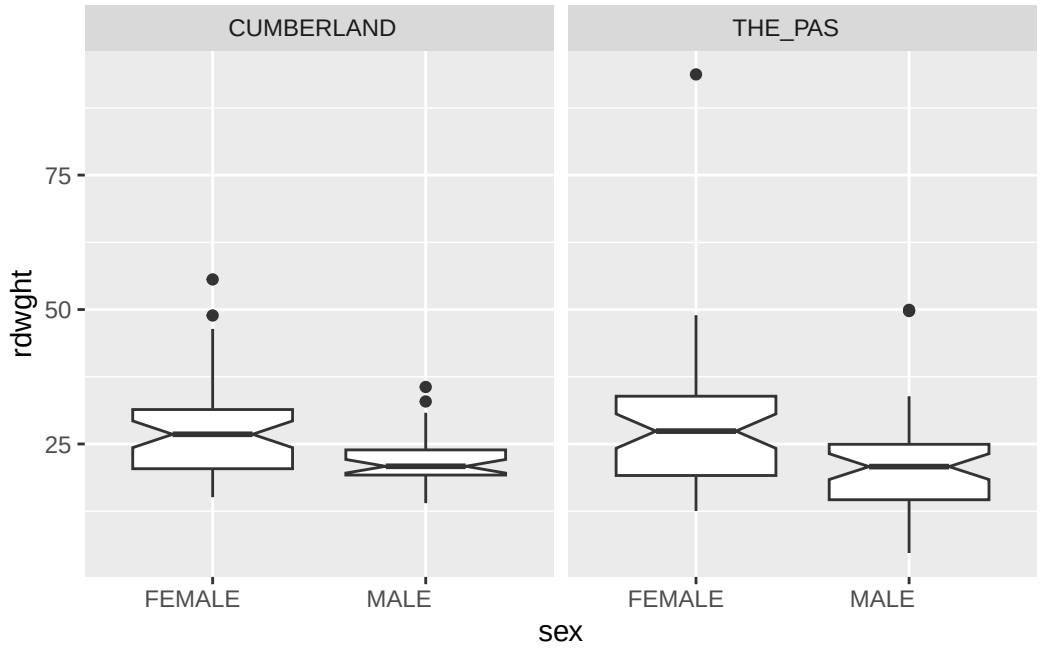
6.2.1. Fixed effects ANOVA (Model I)

- Begin by having a look at the data by generating box plots of `rdwght` for `sex` and `location` from the file `Stu2wdat.csv` .

Solution

```
Stu2wdat <- read.csv("data/Stu2wdat.csv")

ggplot(Stu2wdat, aes(x = sex, y = rdwght)) +
geom_boxplot(notch = TRUE) +
facet_grid(~location)
```

```
Warning: Removed 4 rows containing non-finite values (`stat_boxplot()`).
```



From this, it appears as though females might be larger at both locations. It's difficult to get an idea of whether fish differ in size between the two locations. The presence of outliers on these plots suggests there might be problems meeting normality assumptions for the residuals.

- Generate summary statistics for rdwght by sex and location .

```
Stu2wdat <- read.csv("data/Stu2wdat.csv")
aggregate(rdwght ~ sex + location, data = Stu2wdat, FUN = "summary")
```

```
           sex     location rdwght.Min. rdwght.1st Qu. rdwght.Median
1 FEMALE       CUMBERLAND      15.10000       20.40000      26.80000
2 MALE         CUMBERLAND      14.00000       19.22500      20.85000
3 FEMALE       THE_PAS         12.54000       19.14000      27.39000
4 MALE         THE_PAS          4.73000       14.63000      20.79000
  rdwght.Mean rdwght.3rd Qu. rdwght.Max.
1    27.37347       31.40000    55.60000
2    22.14118       23.90000    35.60000
3    27.97717       33.88000    93.72000
4    20.64652       24.94250    49.94000
```

The summary statistics confirm our interpretation of the box plots: females appear to be larger than males, and differences in fish size between locations are small.

- Using the file `Stu2wdat.csv` , do a two-way factorial ANOVA:

```
# Fit anova model and plot residual diagnostics
# but first, save current par and set graphic page to hold 4 graphs
opar <- par(mfrow = c(2, 2))
anova.model1 <- lm(rdwght ~ sex + location + sex:location,
  contrasts = list(sex = contr.sum, location = contr.sum),
  data = Stu2wdat
)
anova(anova.model1)
```

```
Analysis of Variance Table

Response: rdwght
              Df  Sum Sq Mean Sq F value    Pr(>F)
sex            1  1839.6 1839.55 18.6785 2.569e-05 ***
location       1     4.3    4.26  0.0433    0.8355
sex:location   1    48.7   48.69  0.4944    0.4829
Residuals    178 17530.4   98.49
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

> **Warning**
>
> Be careful here. R gives you the sequential sums of squares (Type I) and associated Mean squares and probabilities. These are not to be trusted unless the design is perfectly balanced. In this case, there are varying numbers of observations across sex and location combinations and therefore the design is not balanced.

What you want are the partial sums of squares (type III). The easiest way to get them is to use the Anova() function in the `car` package (note the subtle difference, `Anova()` is not the same as `anova()`, remember case matters in R.). However, this is not enough by itself. To get the proper values for the type III sums of square, one also needs to specify contrasts, hence the cryptic `contrasts = list(sex = contr.sum,location = contr.sum)`.

```
library(car)
Anova(anova.model1, type = 3)
```

```
Anova Table (Type III tests)

Response: rdwght
             Sum Sq  Df   F value    Pr(>F)
(Intercept)  106507   1 1081.4552 < 2.2e-16 ***
sex            1745   1   17.7220 4.051e-05 ***
location          9   1    0.0891    0.7656
sex:location     49   1    0.4944    0.4829
Residuals     17530 178
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On the basis of the ANOVA, there is no reason to reject two null hypotheses: (1) that the effect of sex (if any) does not depend on location (no interaction), and (2) that there is no difference in the size of sturgeon (pooled over sex) between the two locations . On the other hand, we reject the null hypothesis that there is no difference in size between male and female sturgeon (pooled over location), precisely as expected from the graphs.

```
par(mfrow = c(2, 2))
plot(anova.model1)
```

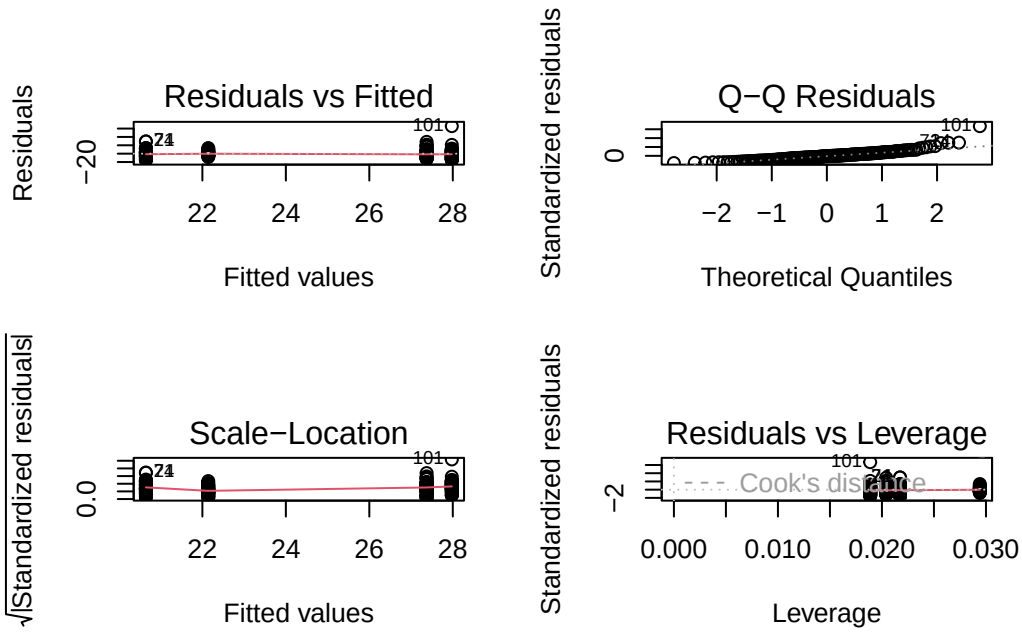


Figure 6.1.: Checking model assumptions for ANOVA model1

As usual, we cannot accept the above results without first ensuring that the assumptions of ANOVA are met. Examination of the residuals plots above shows that the residuals are reasonably normally distributed, with the exception of three potential outliers flagged on the QQ plot (cases 101, 24, & 71; the latter two are on top of one another). However, Cook's distances are not large for these (the 0.5 contour is not even visible on the plot), so there is little indication that these are a concern.The residuals vs fit plot shows that the spread of residuals is about equal over the range of the fitted values, again with the exception of a few cases. When we test for normality of residuals we get:

```
shapiro.test(residuals(anova.model1))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model1)
W = 0.87213, p-value = 2.619e-11
```

So, there is evidence of non-normality in the residuals.

We will use the Levene's test to examine the assumption of homogeneity of variances, just as we did with the 1-way anova.

```
leveneTest(rdwght ~ sex * location, data = Stu2wdat)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value  Pr(>F)
group   3  3.8526 0.01055 *
      178
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If the assumption of homogeneity of variances was valid, we would be accepting the null that the mean of the absolute values of residuals does not vary among levels of sex and location (i.e., group). The above table shows that the hypothesis is rejected and we conclude there is evidence of heteroscedascticity. All in all, there is some evidence that several important assumptions have been violated. However, whether these violations are sufficiently large to invalidate our conclusions remains to be seen.

Exercise

Repeat this procedure using the data file Stu2mdat.Rdata . Now what do you conclude? Suppose you wanted to compare the sizes of males and females: in what way would these comparisons differ between Stu2wdat.Rdata and Stu2mdat.Rdata ?

Solution

```r
Stu2mdat <- read.csv("data/Stu2mdat.csv")
anova.model2 <- lm(
  formula = rdwght ~ sex + location + sex:location,
  contrasts = list(sex = contr.sum, location = contr.sum),
  data = Stu2mdat
)
summary(anova.model2)
Anova(anova.model2, type = 3)
```

```
Call:
lm(formula = rdwght ~ sex + location + sex:location, data = Stu2mdat,
    contrasts = list(sex = contr.sum, location = contr.sum))

Residuals:
    Min      1Q  Median      3Q     Max
-15.917  -6.017  -0.580   4.445  65.743

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     24.5346     0.7461  32.885  < 2e-16 ***
sex1            -0.5246     0.7461  -0.703    0.483
location1        0.2227     0.7461   0.299    0.766
sex1:location1   3.1407     0.7461   4.210 4.05e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 9.924 on 178 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared:  0.09744,   Adjusted R-squared:  0.08223
F-statistic: 6.405 on 3 and 178 DF,  p-value: 0.0003817
```

Note that in this case, we see that at Cumberland House, females are larger than males, whereas the opposite is true in The Pas (you can confirm this observation by generating summary statistics). What happens with the ANOVA (remember, you want Type III sum of squares)?

```
Anova Table (Type III tests)

Response: rdwght
             Sum Sq  Df   F value    Pr(>F)
(Intercept)  106507   1 1081.4552 < 2.2e-16 ***
sex             49   1    0.4944    0.4829
location         9   1    0.0891    0.7656
sex:location  1745   1   17.7220 4.051e-05 ***
Residuals    17530 178
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case, the interaction term `sex:location` is significant but the main effects are not significant.

- You might find it useful here to generate plots for the two data files to compare the interactions between `sex` and `location`. The effect plot shows the relationship between means for each combination of factors (also called cell means). Generate an effect plot for the two models using the `allEffects()` command from the `effects` package:

```
library(effects)
allEffects(anova.model1)
```

```
 model: rdwght ~ sex + location + sex:location

 sex*location effect
             location
sex            CUMBERLAND   THE_PAS
  FEMALE           27.37347     27.97717
  MALE             22.14118     20.64652
```

```
plot(allEffects(anova.model1), "sex:location")
```

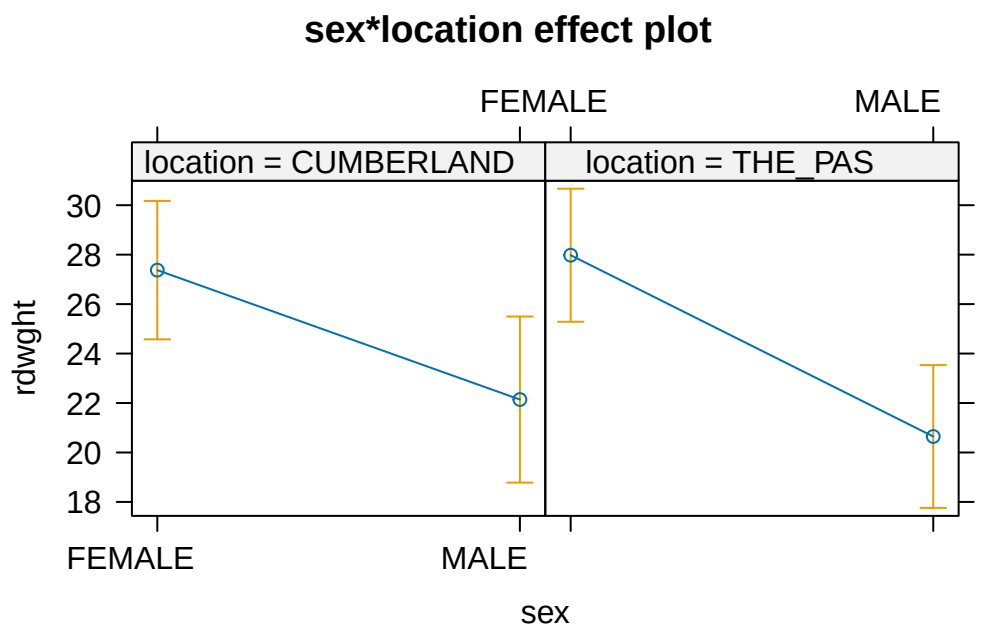


Figure 6.2.: Effet du sexe et du lieu sur le poids des esturgeons

```
allEffects(anova.model2)
```

```
 model: rdwght ~ sex + location + sex:location

 sex*location effect
              location
sex            CUMBERLAND   THE_PAS
  FEMALE          27.37347     20.64652
  MALE            22.14118     27.97717
```

```
plot(allEffects(anova.model2), "sex:location")
```

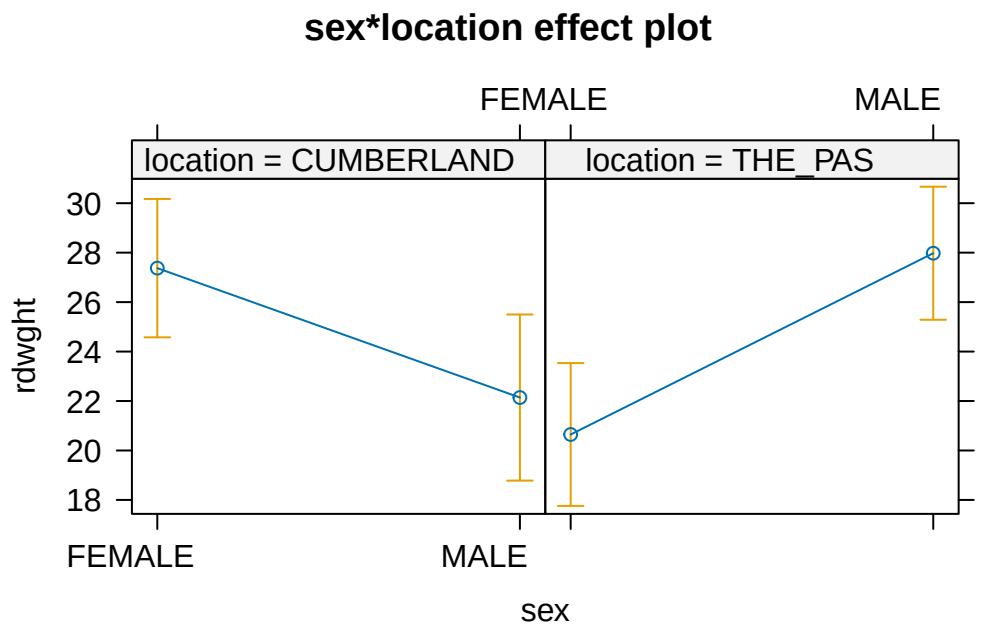


Figure 6.3.: Effet du sexe et du lieu sur le poids des esturgeons

There is a very large difference between the results from `Stu2wdat` and `Stu2mdat`. In the former case, because there is no significant interaction, we can essentially pool over the levels of factor 1 (`sex`, say) to test for the effects of location , or over the levels of factor 2 (`location`) to test for the effects of sex . In fact, if we do so and simply run a one-way ANOVA on the `Stu2wdat` data with sex as the grouping variable, we get:

```
Anova(aov(rdwght ~ sex, data = Stu2wdat), type = 3)
```

```
Anova Table (Type III tests)

Response: rdwght
            Sum Sq  Df F value    Pr(>F)
(Intercept)  78191   1 800.440 < 2.2e-16 ***
sex           1840   1  18.831 2.377e-05 ***
Residuals    17583 180
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that here the residual sum of squares (17583) is only slightly higher than for the 2-way model (17530), simply because, in the 2-way model, only a small fraction of the explained sums of squares is due to the location main effect or the sex:LOCATION interaction. On the other hand, if you try the same trick with `stu2mdat`, you get:

```
Anova(aov(rdwght ~ sex, data = Stu2mdat), type = 3)
```

```
Anova Table (Type III tests)

Response: rdwght
             Sum Sq  Df  F value Pr(>F)
(Intercept)  55251   1 515.0435 <2e-16 ***
sex            113   1   1.0571 0.3053
Residuals    19309 180
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here, the residuals sum of squares (19309) is much larger than in the 2-way model (17530), because most of the explained sums of squares is due to the interaction. Note that if we did this, we would conclude that male and female sturgeons don't differ in size. But in fact they do: it's just that the difference is in different directions, depending on location. This is why it is always dangerous to try and make too much of main effects in the presence of interactions!

6.2.2. Mixed effects ANOVA (Model III)

We have neglected an important component in the above analyses, and that is related to the type of ANOVA model we wish to run. In this example, Location could be considered a random effect, whereas sex is a fixed effect (because it is "fixed" biologically), and so this model could be treated as a mixed model (Model III) ANOVA. Note that in these analyses, R treats analyses by default as Model I ANOVA, so that the main effects and the interaction are tested over the residuals mean square. Recall, however, that in a Model III ANOVA, main effects are tested over the interaction mean square or the pooled interaction mean square and residual mean square (depending on which statistician you consult!)

- Working with the `Stu2wdat` data, rebuild the ANOVA table for `rdwght` for the situation in which `location` is a random factor and `sex` is a fixed factor. To do this, you need to recalculate the F-ratio for sex using the `sex:location` interaction mean square instead of the residual mean square. This is most easily accomplished by hand, making sure you are working with the Type III Sums of squares ANOVA table.

Solution

```
Anova(anova.model1, type = 3)
```

```
Anova Table (Type III tests)

Response: rdwght
              Sum Sq  Df   F value    Pr(>F)
(Intercept)  106507   1 1081.4552 < 2.2e-16 ***
sex            1745   1   17.7220 4.051e-05 ***
location          9   1    0.0891    0.7656
sex:location     49   1    0.4944    0.4829
Residuals     17530 178
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For `sex`, the new ratio of mean squares is

$$F = \frac{(1745/1)}{(49/1)} = 35.6$$

To assign a probability to the new `F-value`, enter the following in the commands window: `pf(F, df1, df2, lower.tail = FALSE)` , where `F` is the newly calculated `F-value`, and `df1` and `df2` are the degrees of freedom of the numerator (`sex`) and denominator (`SEX:location`), respectively.

```
pf(35.6, 1, 1, lower.tail = FALSE)
```

```
[1] 0.1057152
```

Note that the *p value* for `sex` is now non-significant. This is because the error MS of the initial ANOVA is smaller than the interaction MS, but mostly because the number of degrees of freedom of the denominator of the F test has dropped from 178 to 1. In general, a drop in the denominator degrees of freedom makes it much more difficult to reach significance.

> **Note**
>
> Mixed model which are a generalisation of mixed-effect ANOVA are now really developped and are to be favoured intead of doing it by hand.

6.3. 2-way factorial ANOVA without replication

In some experimental designs, there are no replicates within data cells: perhaps it is simply too expensive to obtain more than one datum per cell. A 2-way ANOVA is still possible under these circumstances, but there is an important limitation.

> **Warning**
>
> Because there is no replication within cells, there is no error variance: we have simply a row sum of squares, a column sum of squares, and a remainder sum of squares. This has important implications: if there is an interaction in a Model III ANOVA, only the fixed effect can be tested (over the remainder MS); for Model I ANOVAs, or for random effects in Model III ANOVAs, it is not appropriate to test main effects over the remainder unless we are sure there is no interaction.

A limnologist studying Round Lake in Algonquin Park takes a single temperature (`temp`) reading at 10 different depths (`depth` , in m) at four times (date) over the course of the summer. Her data are shown in `Nr2wdat.csv`.

- Do a two-way unreplicated ANOVA using temp as the dependent vari able, date and depth as the factor variables (you will need to recode depth to tell R to treat this variable as a factor). Note that there is no interaction term included in this model.

```
nr2wdat <- read.csv("data/nr2wdat.csv")
nr2wdat$depth <- as.factor(nr2wdat$depth)
anova.model4 <- lm(temp ~ date + depth, data = nr2wdat)
Anova(anova.model4, type = 3)
```

```
Anova Table (Type III tests)

Response: temp
             Sum Sq Df  F value    Pr(>F)
(Intercept) 1511.99  1 125.5652 1.170e-11 ***
date         591.15  3  16.3641 2.935e-06 ***
depth       1082.82  9   9.9916 1.450e-06 ***
Residuals    325.12 27
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Assuming that this is a Model III ANOVA (`date` random, `depth` fixed), what do you conclude? (Hint: you may want to generate an interaction plot of temp versus depth and month, just to see what's going on.)

```
interaction.plot(nr2wdat$depth, nr2wdat$date, nr2wdat$temp)
```

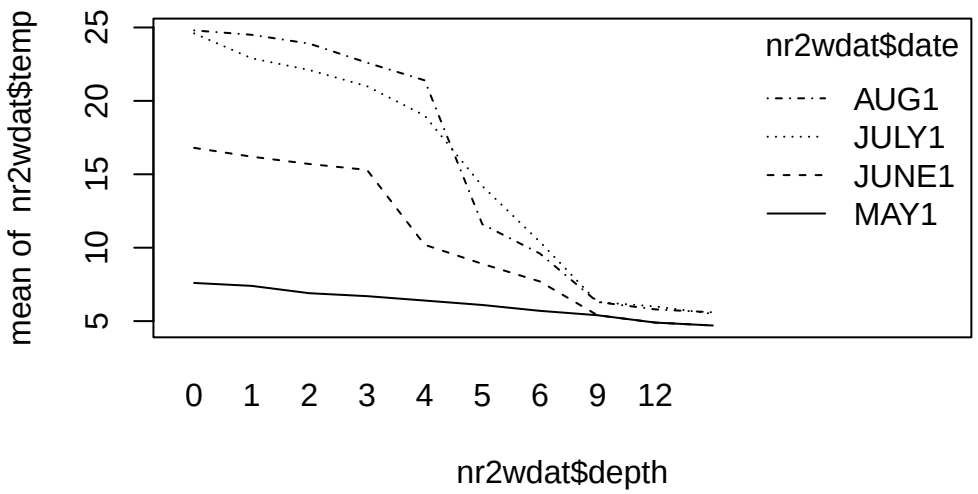


Figure 6.4.: Effet du mois et de la profondeur sur la température

There is a highly significant decrease in temperature as depth increases. To test the effect of month (the (assumed) random factor), we must assume that there is no interaction between depth and month, i.e. that

the change in temperature with depth is the same for each month. This is a dubious assumption: if you plot temperature against depth for each month, you should see that the temperature profile becomes increasingly non-linear as the summer progresses (i.e. the thermocline develops), from almost a linear decline in early spring to what amounts to a step decline in August. In other words, the relationship between temperature and depth does change with month, so that if you were to use the above fitted model to estimate, say, the temperature at a depth of 5 m in July, you would not get a particularly good estimate.

In terms of residual diagnostics, have a look at the residuals probability plot and residuals vs fitted values plot.

```
par(mfrow = c(2, 2))
plot(anova.model4)
```

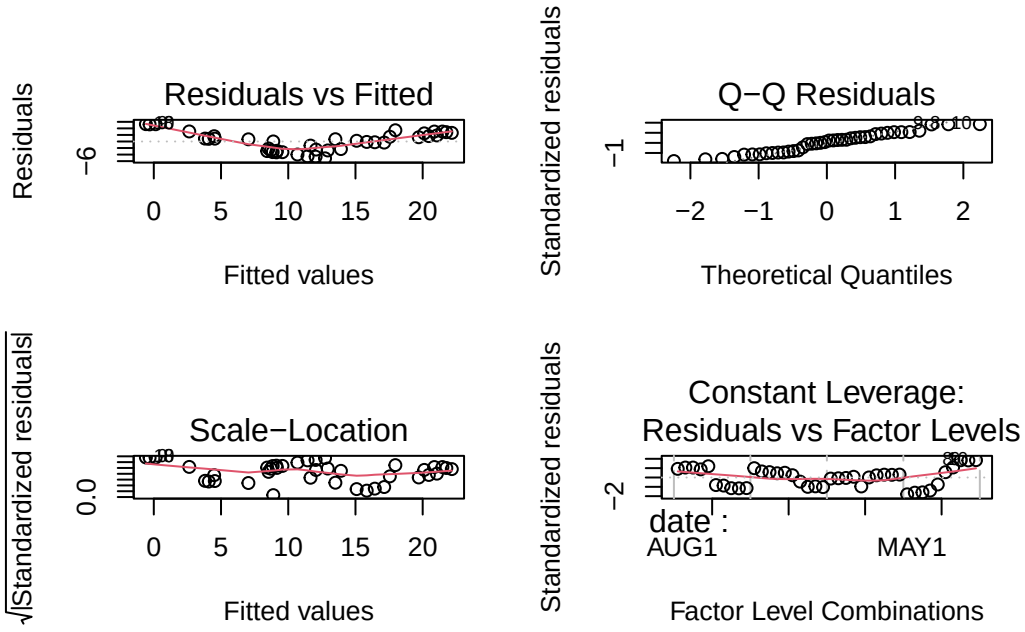


Figure 6.5.: Conditions d'applications du modèle anova.model4

```
shapiro.test(residuals(anova.model4))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model4)
W = 0.95968, p-value = 0.1634
```

Testing the residuals for normality, we get `p = 0.16`, so that the normality assumption seems to be O.K. In terms of heteroscedasticity, we can only test among months, using depths as replicates (or among depths using months as replicates). Using depths as replicates within months, we find

```
leveneTest(temp ~ date, data = nr2wdat)
```

```
Warning in leveneTest.default(y = y, group = group, ...): group coerced to
factor.
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value    Pr(>F)
group  3  17.979 2.679e-07 ***
      36
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So there seems to be some problem here, as can be plainly seen in the above plot of residuals vs fit. All in all, this analysis is not very satisfactory: there appears to be some problems with the assumptions, and the assumption of no interaction between depth and date would appear to be invalid.

6.4. Nested designs

A common experimental design occurs when each major group (or treatment) is divided into randomly chosen subgroups. For example, a geneticist interested in the effects of genotype on desiccation resistance in fruit flies might conduct an experiment with larvae of three different genotypes. For each genotype (major group), she sets up three environmental chambers (sub-groups, replicates within groups) with a fixed temperature humidity regime, and in each chamber, she has five larvae for which she records the number of hours each larvae survived.

- The file `Nestdat.csv` contains the results of just such an experi ment. The file lists three variables: genotype , chamber and survival . Run a nested ANOVA with survival as the dependent variable, genotype/chamber as the independent variables (this is the shorthand notation for a chamber effect nested under genotype).

```
nestdat <- read.csv("data/nestdat.csv")
nestdat$chamber <- as.factor(nestdat$chamber)
nestdat$genotype <- as.factor(nestdat$genotype)
anova.nested <- lm(survival ~ genotype / chamber, data = nestdat)
```

What do you conclude from this analysis? What analysis would (should) you do next? (Hint: if there is a non-significant effect of chambers within genotypes, then you can increase the power of between-genotype comparisons by pooling over chambers within genotypes, although not everyone (Dr. Rundle included) agrees with such pooling.) Do it! Make sure you check your assumptions!

Solution

```
anova(anova.nested)
```

```
Analysis of Variance Table

Response: survival
                 Df  Sum Sq Mean Sq  F value Pr(>F)
genotype          2 2952.22 1476.11 292.6081 <2e-16 ***
genotype:chamber  6   40.65    6.78   1.3432 0.2639
Residuals        36  181.61    5.04
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mfrow = c(2, 2))
plot(anova.nested)
```

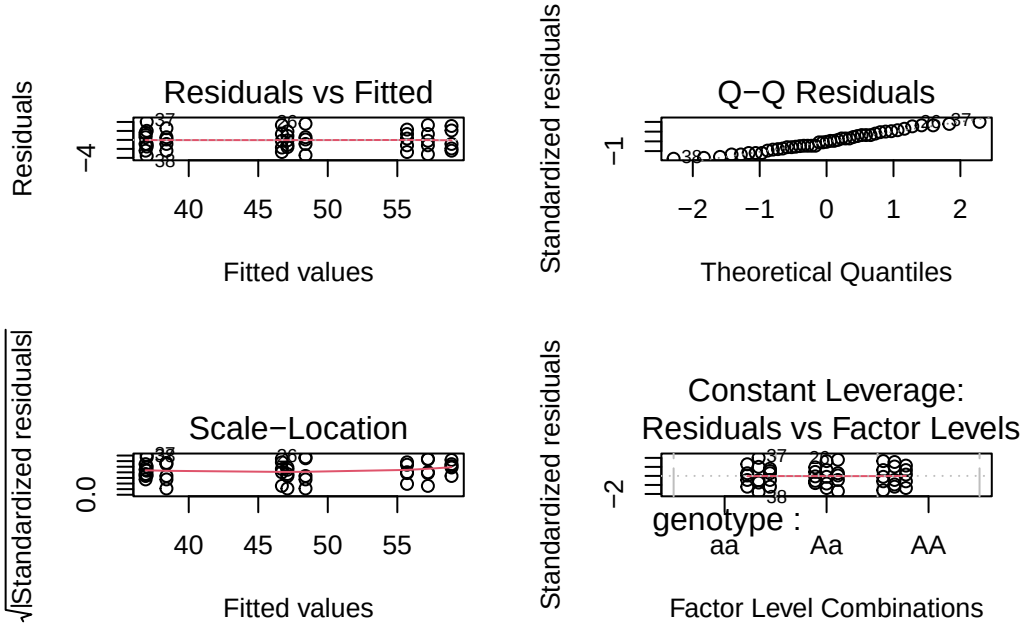


Figure 6.6.: Conditions d'applications du modèle anova.nested

We conclude from this analysis that there is no (significant) variation among chambers within genotypes, but that the null hypothesis that all genotypes have the same dessiccation resistance (as measured by survival) is rejected (Test of genotype using MS genotype:chamber as denominator: F = 1476.11/6.78 = 217.7153, P<0.0001). In other words, genotypes differ in their survival.

Since the chambers within genotypes effect is non-significant, we may want to pool over chambers to increase our degrees of freedom:

```
anova.simple <- lm(survival ~ genotype, data = nestdat)
anova(anova.simple)
```

```
Analysis of Variance Table

Response: survival
          Df  Sum Sq Mean Sq F value    Pr(>F)
genotype   2 2952.22 1476.11  278.93 < 2.2e-16 ***
Residuals 42  222.26    5.29
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus, we conclude that there is significant variation among the three genotypes in dessiccation resistance.

A box plot of survival across genotypes shows clearly that there is significant variation among the three genotypes in dessiccation resistance. This can be combined with a formal Tukey multiple comparison test:

```
par(mfrow = c(1, 1))
# Compute and plot means and Tukey CI
means <- glht(anova.simple, linfct = mcp(
  genotype =
    "Tukey"
))
cimeans <- cld(means)
# use sufficiently large upper margin
old.par <- par(mai = c(1, 1, 1.25, 1))
# plot
plot(cimeans, las = 1) # las option to put y-axis labels as God intended them
```

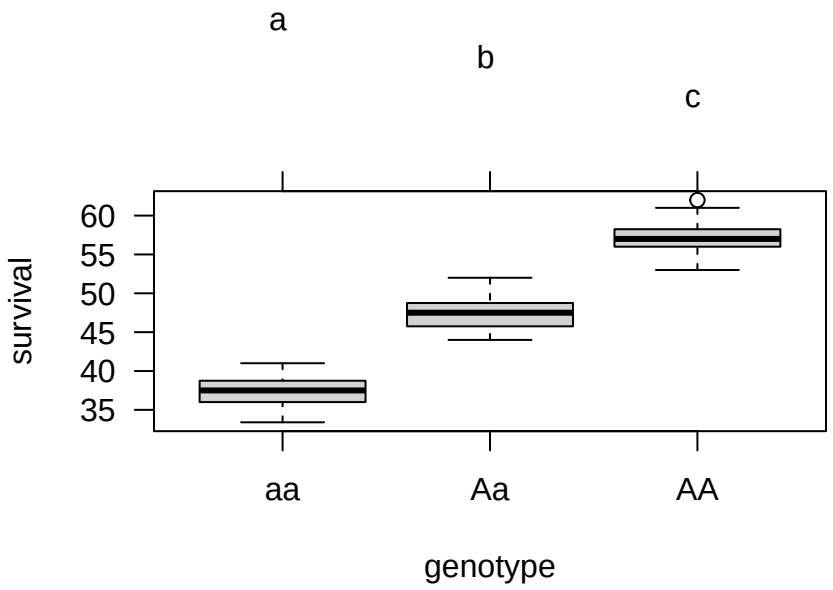


Figure 6.7.: Effet du genotype sur la résistance à la dessication avec un test de Tukey

So, we conclude from the Tukey analysis and plot that dessiccation resistance (R) , as measured by larval survival under hot, dry conditions, varies significantly among all three genotypes with R(AA) > R(Aa) > R(aa).

Before concluding this, however, we must test the assumptions. Here are the residual plots and diagnostics for the one-way (unnested) design:

Solution

```
par(mfrow = c(2, 2))
plot(anova.simple)
```

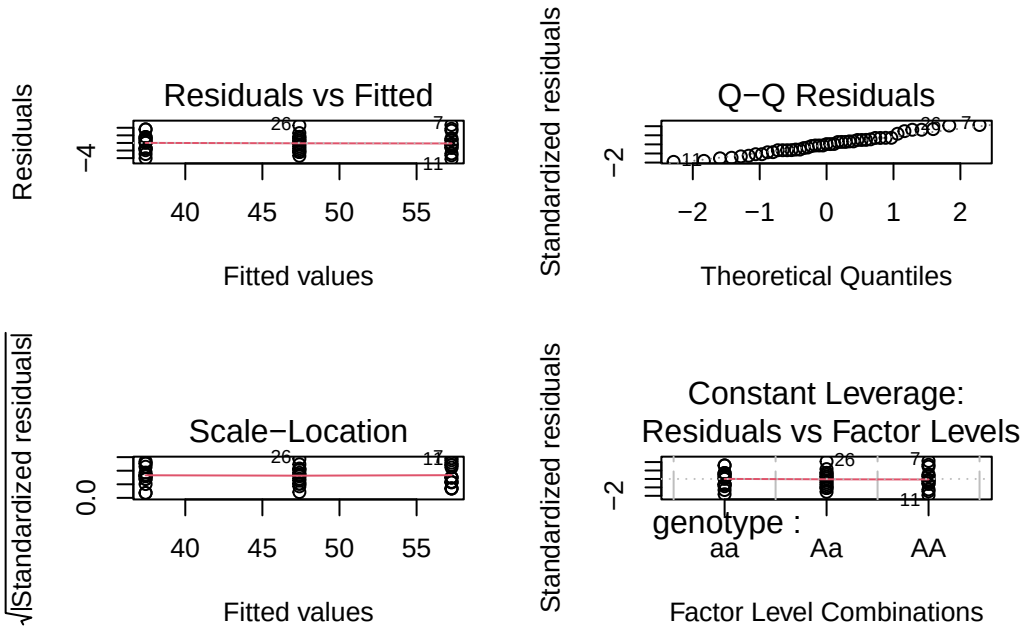


Figure 6.8.: Conditions d'applications du modèle anova.simple

So, all the assumptions appear to be valid, and the conclusion reached above still holds. Note that if you compare the residual mean squares of the nested and one-way ANOVAs (5.04 vs 5.29), they are almost identical. This is not surprising, given the small contribution of the chamber %in% genotype effect to the explained sum of squares.

6.5. Two-way non-parametric ANOVA

Two-way non-parametric ANOVA is an extension of the non-parametric one-way methods discussed previously. The basic procedure is to rank all the data in the sample from smallest to largest, then carry out a 2-way ANOVA on the ranks. This can be done either for replicated or unreplicated data.

Using the data file `Stu2wdat.csv` , do a two-factor ANOVA to examine the effects of `sex` and `location` on `rank(rdwght)`.

```
aov.rank <- aov(
  rank(rdwght) ~ sex * location,
  contrasts = list(
    sex = contr.sum, location = contr.sum
  ),
  data = Stu2wdat
)
```

The Scheirer-Ray-Hare extension of the Kruskall-Wallis test is done by computing a statistic H given by the effect sums of squares (SS) divided by the total MS. The latter can be calculated as the variance of the ranks. We compute an H statistic for each term. The H-statistics are then compared to a theoretical

χ^2 (chi-square) distribution using the command line: `pchisq(H, df, lower.tail = FALSE)` , where `H` and `df` are the calculated H-statistics and associated degrees of freedom, respectively.

- Use the ANOVA table based on ranks to test the effects of `sex` and on `rdwght`. What do you conclude? How does this result compare with the result obtained with the parametric 2-way ANOVA done before?

```
Anova(aov.rank, type = 3)
```

```
Anova Table (Type III tests)

Response: rank(rdwght)
              Sum Sq  Df  F value    Pr(>F)
(Intercept)  1499862   1 577.8673 < 2.2e-16 ***
sex            58394   1  22.4979 4.237e-06 ***
location        1128   1   0.4347    0.5105
sex:location    1230   1   0.4738    0.4921
Residuals     472383 182
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To calculate the Scheirer-Ray-Hare extension to the Kruskall-Wallis test, you must first calculate the total mean square (MS), i.e. the variance of the ranked data. In this case, there are 186 observations, their ranks are therefore the series 1, 2, 3, ..., 186. The variance can be calculated simply as var(1:186) (Isn't R neat? Cryptic maybe, but neat). So we can compute the `H` statistic for each term:

```
Hsex <- 58394 / var(1:186)
Hlocation <- 1128 / var(1:186)
Hsexloc <- 1230 / var(1:186)
```

And convert these statistics into p-values:

```
# sex
Hsex
```

```
[1] 20.14628
```

```
pchisq(Hsex, 1, lower.tail = FALSE)
```

```
[1] 7.173954e-06
```

```
# location
Hlocation
```

```
[1] 0.3891668
```

```
pchisq(Hlocation, 1, lower.tail = FALSE)
```

```
[1] 0.5327377
```

```
# sex:location
Hsexloc
```

```
[1] 0.4243574
```

```
pchisq(Hsexloc, 1, lower.tail = FALSE)
```

```
[1] 0.5147707
```

Note that these results are the same as those obtained in our original two-way parametric ANOVA. Despite the reduced power, we still find significant differences between the sexes, but still no interaction and no effect due to location.

There is, however, an important difference. Recall that in the original parametric ANOVA, there was a significant effect of sex when we considered the problem as a Model I ANOVA. However, if we consider it as Model III, the significant sex effect could in principle disappear, because the df associated with the interaction MS are much smaller than the df associated with the Model I error MS. In this case, however, the interaction MS is about half that of the error MS. So, the significant sex effect becomes even more significant if we analyze the problem as a Model III ANOVA. Once again, we see the importance of specifying the appropriate ANOVA design.

6.6. Multiple comparisons

Further hypothesis testing in multiway ANOVAs depends critically on the outcome of the initial ANOVA. If you are interested in comparing groups of marginal means (that is, means of treatments for one factor pooled over levels of the other factor, e.g., between male and female sturgeon pooled over location), this can be done exactly as outlined for multiple comparisons for one-way ANOVAs. For comparison of individual cell means, you must specify the interaction as the group variable.

The file `wmcdat2.csv` shows measured oxygen consumption (`o2cons`) of two species (`species` = A, B)) of limpets at three different concentrations of seawater (`conc` = 100, 75, 50%) taken from Sokal and Rohlf, 1995, p. 332.

- Run a 2-way factorial ANOVA on `wmcdat2` data, using `o2cons` as the dependent variable and species and conc as the factors. What do you conclude?

Solution

```
wmcdat2 <- read.csv("data/wmcdat2.csv")
wmcdat2$species <- as.factor(wmcdat2$species)
wmcdat2$conc <- as.factor(wmcdat2$conc)
anova.model5 <- lm(o2cons ~ species * conc, data = wmcdat2)
Anova(anova.model5, type = 3)
```

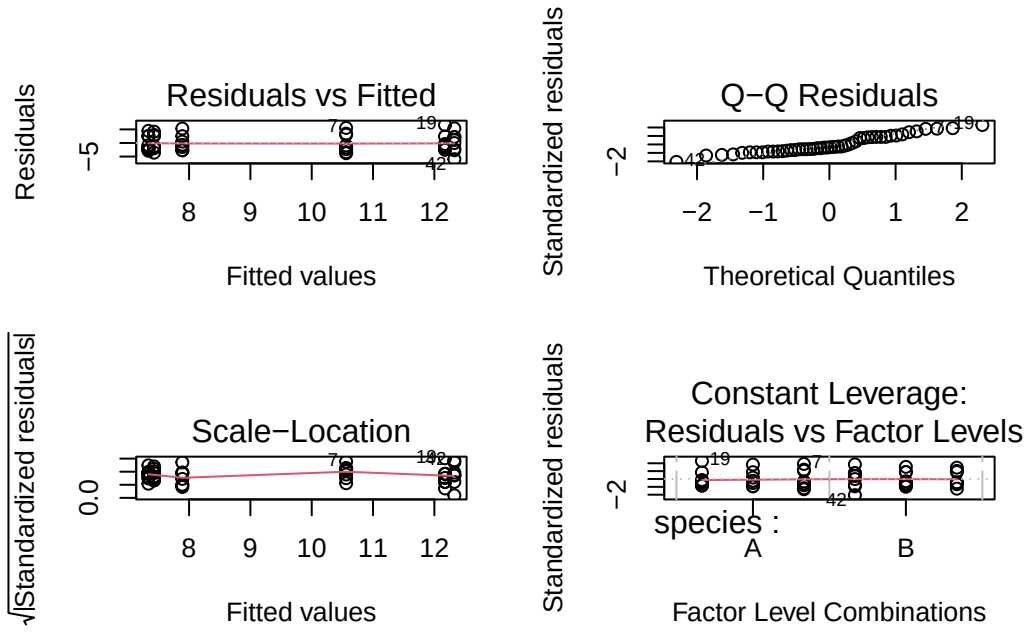
The ANOVA table is shown below. Technically, because the sample sizes in individual cells are rather small, this analysis should be repeated using a non-parametric ANOVA. For the moment, let's stick with the parametric analysis.

```
Anova Table (Type III tests)

Response: o2cons
              Sum Sq Df  F value    Pr(>F)
(Intercept)  1185.60  1 124.0165 4.101e-14 ***
species         0.09  1   0.0097   0.92189
conc           74.90  2   3.9172   0.02755 *
species:conc   23.93  2   1.2514   0.29656
Residuals     401.52 42
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Look at the diagnostic plots:

Solution

```
par(mfrow = c(2, 2))
plot(anova.model5)
```



Homoscedasticity looks ok, but normality less so.. Testing for normality, we get:

Solution

```
shapiro.test(residuals(anova.model5))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model5)
W = 0.93692, p-value = 0.01238
```

So there is evidence of non-normality, but otherwise everything looks O.K. Since the ANOVA is relatively robust with respect to non-normality, we proceed, but if we wanted to reassure ourselves, we could run a non-parametric ANOVA, and get the same answer.

- On the basis of the ANOVA results obtained above, which means would you proceed to compare? Why?

Solution

Need to add an explnation here

Overall, we conclude that there are no differences among species, and that the effect of concentration does not depend on species (no interaction). Since there is no interaction and no main effect due to species, the only comparison of interest is among salinity concentrations:

```
# fit simplified model
anova.model6 <- aov(o2cons ~ conc, data = wmcdat2)
# Make Tukey multiple comparisons
TukeyHSD(anova.model6)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = o2cons ~ conc, data = wmcdat2)

$conc
           diff       lwr        upr     p adj
75-50  -4.63625 -7.321998 -1.9505018 0.0003793
100-50 -3.25500 -5.940748 -0.5692518 0.0141313
100-75  1.38125 -1.304498  4.0669982 0.4325855
```

```
par(mfrow = c(1, 1))
# Graph of all comparisons for conc
tuk <- glht(anova.model6, linfct = mcp(conc = "Tukey"))
# extract information
tuk.cld <- cld(tuk)
# use sufficiently large upper margin
old.par <- par(mai = c(1, 1, 1.25, 1))
# plot
plot(tuk.cld)
```

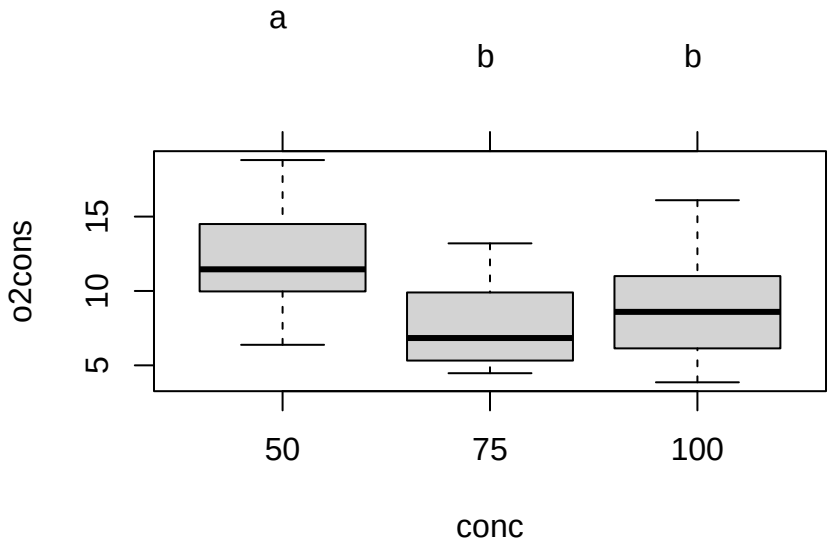


Figure 6.9.: Comparaison de Tukey des moyennes de consommation d'oxygèn en fonction del la concentration

```
par(old.par)
```

So there is evidence of a significant difference in oxygen consumption at a reduction in salinity to 50% of regular seawater, but not at a reduction of only 25%.

- Repeat the analysis described above using `wmc2dat2.csv` . How do your results compare with those obtained for `wmcdat2.csv` ?

Solution

```
wmc2dat2 <- read.csv("data/wmc2dat2.csv")
wmc2dat2$species <- as.factor(wmc2dat2$species)
wmc2dat2$conc <- as.factor(wmc2dat2$conc)
anova.model7 <- lm(o2cons ~ species * conc, data = wmc2dat2)
```

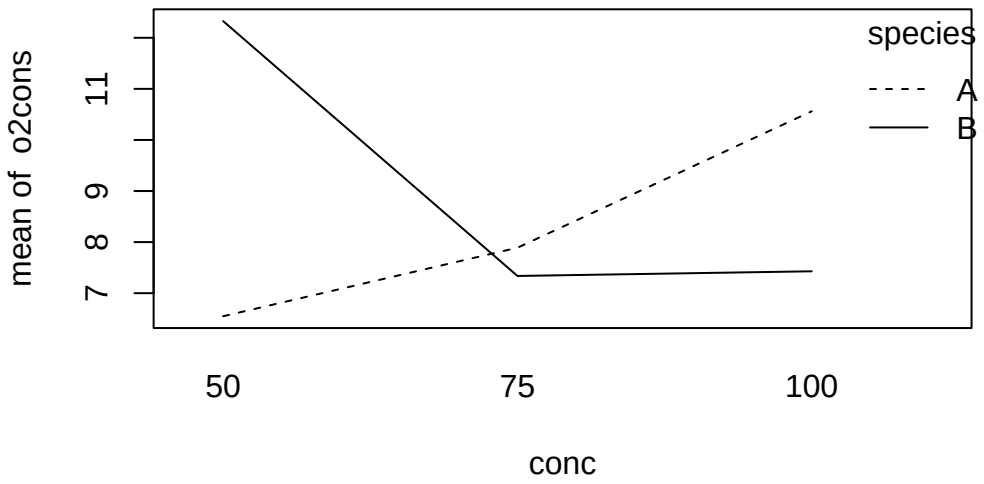
Using `wmc2dat2.csv`,we get:

```
Anova Table (Type III tests)

Response: o2cons
             Sum Sq Df F value    Pr(>F)
(Intercept)  343.09  1 36.2132 3.745e-07 ***
species      133.52  1 14.0929 0.0005286 ***
```

```
conc          66.76  2  3.5232 0.0385011 *
species:conc 168.15  2  8.8742 0.0006101 ***
Residuals    397.91 42
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here there is a large interaction effect, and consequently, there is no point in comparing marginal means. This is made clear by examining an interaction plot:

```
with(wmc2dat2, interaction.plot(conc, species, o2cons))
```



- Working still with the `wmc2dat2` data set, compare individual cell means (6 in all), with the Bonferonni adjustment. To do this, it is helpful to create a new variable to indicate all the combinations of `species` and `conc`:

```
wmc2dat2$species.conc <- as.factor(paste0(wmc2dat2$species, wmc2dat2$conc))
```

Then we can conduct pairwise bonferroni comparisons:

```
with(wmc2dat2, pairwise.t.test(o2cons, species.conc, p.adj = "bonf"))
```

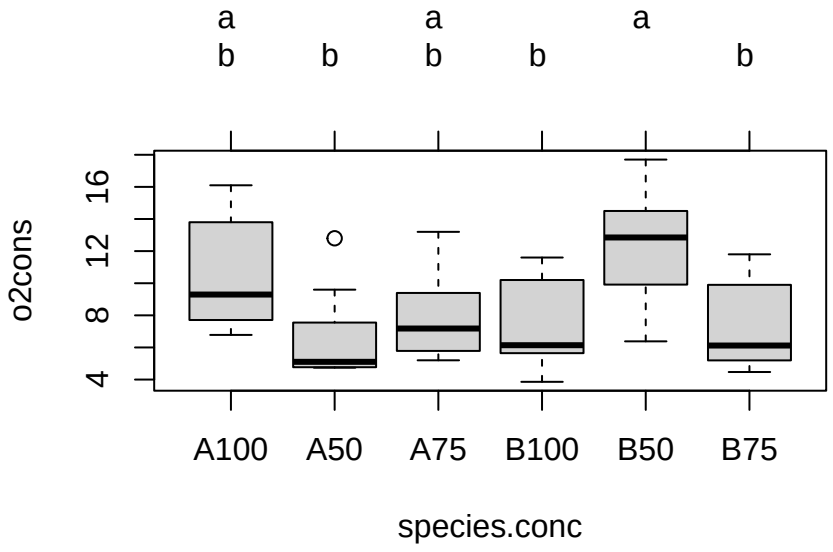
```
    Pairwise comparisons using t tests with pooled SD

data:  o2cons and species.conc
```

```
     A100   A50    A75    B100   B50
A50  0.1887 -      -      -      -
A75  1.0000 1.0000 -      -      -
B100 0.7223 1.0000 1.0000 -      -
B50  1.0000 0.0079 0.0929 0.0412 -
B75  0.6340 1.0000 1.0000 1.0000 0.0350

P value adjustment method: bonferroni
```

These comparisons are a little more difficult to interpret, but the analysis essentially examines for differences among seawater concentrations within species A and for differences among concentrations within species B. We see here that the o2Cons at 50% seawater for species B is significantly different from that of 75% and 100% seawater for species B, whereas there are no significant differences in `o2cons` for species A across all seawater concentrations.

I find these outputs rather unsatisfying because they show only p-values, but no indication of effect size. One can get both the conclusion from the multiple comparison procedure and an indication of effect size from the graph produced with the following code:

```
# fit one-way anova comparing all combinations of species.conc combinations
anova.modelx <- aov(o2cons ~ species.conc, data = wmc2dat2)
tuk2 <- glht(anova.modelx, linfct = mcp(species.conc = "Tukey"))
# extract information
tuk2.cld <- cld(tuk2)
# use sufficiently large upper margin
old.par <- par(mai = c(1, 1, 1.25, 1))
# plot
plot(tuk2.cld)
```



```
par(old.par)
```

Note that in this analysis, we have used the error MS = 9.474 from the original model to contrast cell means. Recall, however, that this assumes that in fact we are dealing with a Model I ANOVA, which may or may not be the case (conc is certainly a fixed factor, but species might be either fixed or random).

6.7. Test de permutation pour l'ANOVA à deux facteurs de classification

When data do not meet the assumptions of the parametric analysis in two- and multiway ANOVA, as an alternative to the non-parametric ANOVA, it is possible to run permutation tests to calculate p-values. The lmPerm package does this easily.

```
#########################################################################
## lmPerm version of permutation test
library(lmPerm)
# for generality, copy desired dataframe to mydata
# and model formula to myformula
mydata <- Stu2wdat
myformula <- as.formula("rdwght ~ sex+location+sex:location")
# Fit desired model on the desired dataframe
mymodel <- lm(myformula, data = mydata)
# Calculate permutation p-value
anova(lmp(myformula, data = mydata, perm = "Prob", center = FALSE, Ca = 0.001))
```

`lmPerm` was orphaned for a while and the code below, while clunkier, provided an alternative way of doing it. You would have to adapt it for other situations.

```
#############################################################
# Permutation test for two way ANOVA
# Ter Braak creates residuals from cell means and then permutes across
# all cells
# This can be accomplished by taking residuals from the full model
# modified from code written by David C. Howell
# http://www.uvm.edu/~dhowell/StatPages/More_Stuff/Permutation%20Anova/PermTestsAnova.html
nreps <- 500
dependent <- Stu2wdat$rdwght
factor1 <- as.factor(Stu2wdat$sex)
factor2 <- as.factor(Stu2wdat$location)
my.dataframe <- data.frame(dependent, factor1, factor2)
my.dataframe.noNA <- my.dataframe[complete.cases(my.dataframe), ]
mod <- lm(dependent ~ factor1 + factor2 + factor1:factor2,
  data = my.dataframe.noNA
)
res <- mod$residuals
```

```
TBint <- numeric(nreps)
TB1 <- numeric(nreps)
TB2 <- numeric(nreps)
ANOVA <- summary(aov(mod))
cat(
  " The standard ANOVA for these data follows ",
  "\n"
)
F1 <- ANOVA[[1]]$"F value"[1]
F2 <- ANOVA[[1]]$"F value"[2]
Finteract <- ANOVA[[1]]$"F value"[3]
print(ANOVA)
cat("\n")
cat("\n")
TBint[1] <- Finteract
for (i in 2:nreps) {
  newdat <- sample(res, length(res), replace = FALSE)
  modb <- summary(aov(newdat ~ factor1 + factor2 +
    factor1:factor2,
  data = my.dataframe.noNA
  ))
  TBint[i] <- modb[[1]]$"F value"[3]
  TB1[i] <- modb[[1]]$"F value"[1]
  TB2[i] <- modb[[1]]$"F value"[2]
}
probInt <- length(TBint[TBint >= Finteract]) / nreps
prob1 <- length(TB1[TB1 >= F1]) / nreps
prob2 <- length(TB2[TB1 >= F2]) / nreps
cat("\n")
cat("\n")
print("Resampling as in ter Braak with unrestricted sampling
of cell residuals. ")
cat(
  "The probability for the effect of Interaction is ",
  probInt, "\n"
)
cat(
  "The probability for the effect of Factor 1 is ",
  prob1, "\n"
)
cat(
  "The probability for the effect of Factor 2 is ",
  prob2, "\n"
)
```

6.8. Bootstrap for two-way ANOVA

In most cases, permutation tests will be more appropriate than bootstrap in ANOVA designs. However, for the sake of completedness, I have a snippet of code to do bootstrap for you::

```
############################################################
############
# Bootstrap for two-way ANOVA
# You possibly want to edit bootfunction.mod1 to return other values
# Here it returns the standard coefficients of the fitted model
# Requires boot library
#
nreps <- 5000
dependent <- Stu2wdat$rdwght
factor1 <- as.factor(Stu2wdat$sex)
factor2 <- as.factor(Stu2wdat$location)
my.dataframe <- data.frame(dependent, factor1, factor2)
my.dataframe.noNA <- my.dataframe[complete.cases(my.dataframe), ]
library(boot)
# Fit model on observed data
mod1 <- aov(dependent ~ factor1 + factor2 + factor1:factor2,
  data = my.dataframe.noNA
)



# Bootstrap 1000 time using the residuals bootstraping methods to
# keep the same unequal number of observations for each level of the indep. var.
fit <- fitted(mod1)
e <- residuals(mod1)
X <- model.matrix(mod1)
bootfunction.mod1 <- function(data, indices) {
  y <- fit + e[indices]
  bootmod <- lm(y ~ X)
  coefficients(bootmod)
}
bootresults <- boot(my.dataframe.noNA, bootfunction.mod1,
  R = 1000
)
bootresults
## Calculate 90% CI and plot bootstrap estimates separately for each model parameter
boot.ci(bootresults, conf = 0.9, index = 1)
plot(bootresults, index = 1)
boot.ci(bootresults, conf = 0.9, index = 3)
plot(bootresults, index = 3)
boot.ci(bootresults, conf = 0.9, index = 4)
plot(bootresults, index = 4)
boot.ci(bootresults, conf = 0.9, index = 5)
plot(bootresults, index = 5)
```

7. Multiple regression

After completing this laboratory exercise, you should be able to:

- Use R to fit a multiple regression model, and compare the adequacy of several models using inferential and information theoretic criteria
- Use R to test hypotheses about the effects of different independent variables on the dependent variable of interest.
- Use R to evaluate multicollinearity among (supposedly) independent variables and its effects.
- Use R to do curvilinear (polynomial) regression.

7.1. R packages and data

For this lab you need:

- R packages:
  - ggplot2
  - car
  - lmtest
  - simpleboot
  - boot
  - MuMIn
- data files:
  - Mregdat.csv

7.2. Points to keep in mind

Multiple regression models are used in cases where there is one dependent variable and several independent, continuous variables. In many biological systems, the variable of interest may be influenced by several different factors, so that accurate description or prediction requires that several independent variables be included in the regression model. Before beginning, be aware that multiple regression takes time to learn well. Beginners should keep in mind several important points:

1. An overall regression model may be statistically significant even if none of the individual regression coefficients in the model are (caused by multicollinearity)
2. A multiple regression model may be "nonsignificant" even though some of the individual coefficients are "significant" (caused by overfitting)
3. Unless "independent" variables are uncorrelated in the sample, different model selection procedures may yield different results.

7.3. First look at the data

The file Mregdat.Rdata contains data collected in 30 wetlands in the Ottawa-Cornwall-Kingston area. The data included are

- the richness (number of species) of:
  - birds (`bird` , and its log transform `logbird`),
  - plants (`plant`, `logpl`),
  - mammals (`mammal`, `logmam`),
  - herptiles (`herptile`, `logherp`)
  - total species richness of all four groups combined (`totsp`, `logtot`)
- GPS coordinates of the wetland (`lat` , `long`)
- its area (`logarea`)
- the percentage of the wetland covered by water at all times during the year (`swamp`)
- the percentage of forested land within 1 km of the wetland (cpfor2)
- the density (in m/hectare) of hard-surface roads within 1 km of the wetland (`thtden`).

We will focus on herptiles for this exercise, so we better first have a look at how this variable is distributed and correlated to the potential independent variables:

```
mydata <- read.csv("data/Mregdat.csv")
scatterplotMatrix(
  ~ logherp + logarea + cpfor2 + thtden + swamp,
  regLine = TRUE, smooth = TRUE, diagonal = TRUE,
  data = mydata
)
```

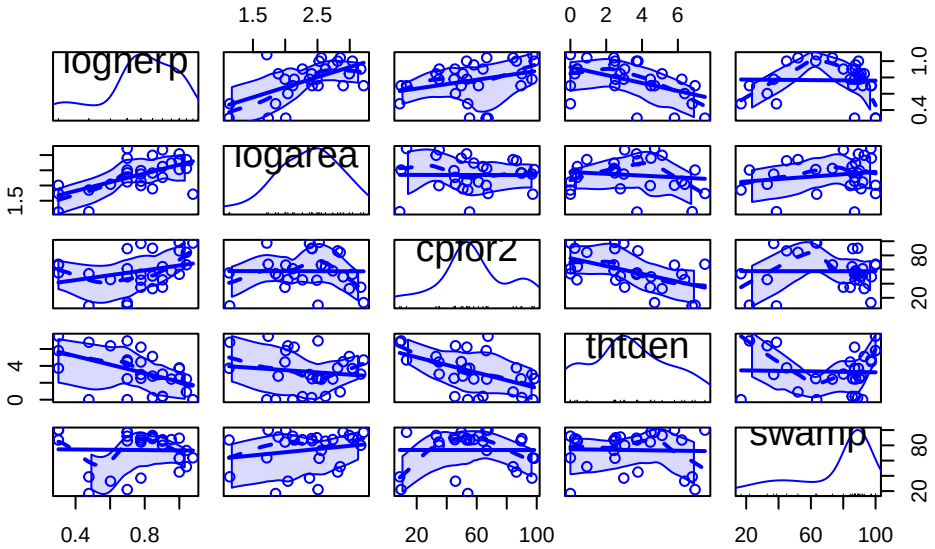


Figure 7.1.: Matrice de rélation et densité pour la richesse spécifique des amphibiens et reptiles

7.4. Multiple regression models from scratch

We begin the multiple regression exercise by considering a situation with one dependent variable and three (possibly) independent variables. First, we will start from scratch and build a multiple regression model based on what we know from building simple regression models. Next, we will look at automated methods of building multiple regressions models using simultaneous, forward, and backward stepwise procedures.

> Exercise
>
> Using the subset of the `Mregdat.csv` data file, regress `logherp` on `logarea`.

On the basis of the regression, what do you conclude?

```
model_loga <- lm(logherp ~ logarea, data = mydata)
summary(model_loga)
```

```
Call:
lm(formula = logherp ~ logarea, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.38082 -0.09265  0.00763  0.10409  0.46977

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.18503    0.15725   1.177 0.249996
logarea      0.24736    0.06536   3.784 0.000818 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1856 on 26 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.3552,    Adjusted R-squared:  0.3304
F-statistic: 14.32 on 1 and 26 DF,  p-value: 0.0008185
```

```
par(mfrow = c(2, 2))
plot(model_loga)
```

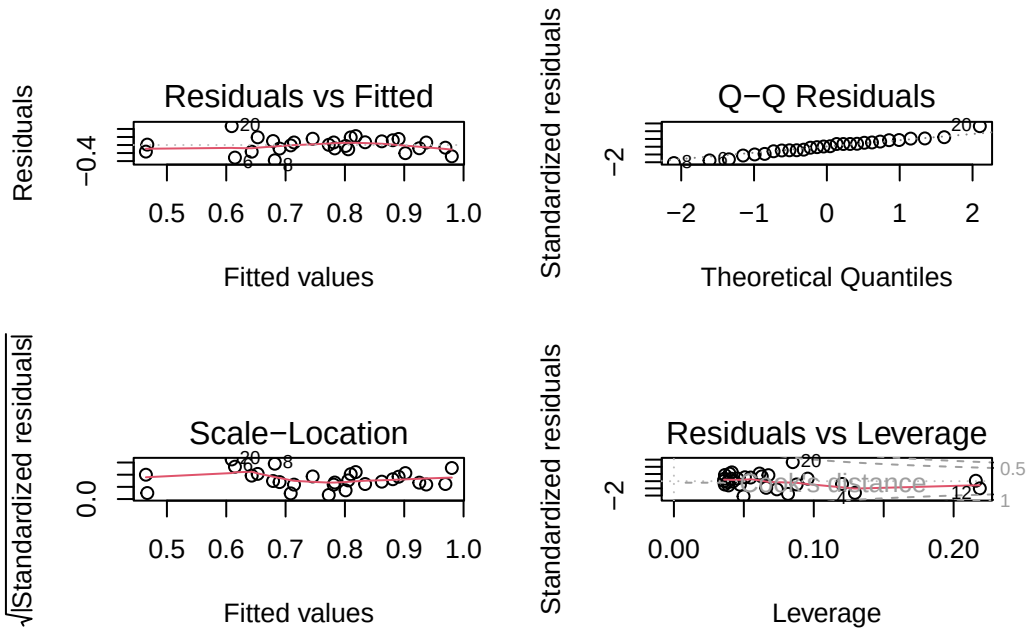


Figure 7.2.: Checking model asusmptions for regression of *logherp* as a function of *logarea*

It looks like there is a positive relationship between herptile species richness and wetland area: the larger the wetland, the greater the number of species. Note, however, that about 2/3 of the observed variability in species richness among wetlands is not "explained" by wetland area ($R^2 = 0.355$). Residual analysis shows no major problems with normality, heteroscedasticity or independence of residuals.

Exercise

Rerun the above regression, this time replacing `logarea` with `cpfor2` as the independent variable, such that the expression in the formula field reads: `logherp ~ cpfor2` . What do you conclude?

Solution

```
model_logcp <- lm(logherp ~ cpfor2, data = mydata)
summary(model_logcp)
```

```
Call:
lm(formula = logherp ~ cpfor2, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.49095 -0.10266  0.05881  0.16027  0.25159

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.609197   0.104233   5.845 3.68e-06 ***
```

```
cpfor2      0.002706   0.001658   1.632    0.115
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2202 on 26 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.09289,   Adjusted R-squared:  0.058
F-statistic: 2.662 on 1 and 26 DF,  p-value: 0.1148
```

According to this result, we would accept the null hypothesis, and conclude that there is no relationship between herptile density and the proportion of forest on adjacent lands. But what happens when we enter both variables into the regression simultaneously?

Exercise

Rerun the above regression one more time, this time adding both independent variables into the model at once, such that `logherp ~ logarea + cpfor2` . What do you conclude?

Solution

```
model_mcp <- lm(logherp ~ logarea + cpfor2, data = mydata)
summary(model_mcp)
```

```

Call:
lm(formula = logherp ~ logarea + cpfor2, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.40438 -0.11512  0.01774  0.08187  0.36179

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.027058   0.166749   0.162 0.872398
logarea     0.247789   0.061603   4.022 0.000468 ***
cpfor2      0.002724   0.001318   2.067 0.049232 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.175 on 25 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.4493,    Adjusted R-squared:  0.4052
F-statistic:  10.2 on 2 and 25 DF,  p-value: 0.0005774
```

Now we reject both null hypotheses that the slope of the regression of `logherp` on logarea is zero and that the slope of the regression of `logherp` on `cpfor2` is zero.

Why is `cpfor2` a significant predictor of `logherp` in the combined model when it was not significant in the simple linear model? The answer lies in the fact that it is sometimes necessary to control for one variable in order to detect the effect of another variable. In this case, there is a significant relationship between `logherp` and `logarea` that masks the relationship between `logherp` and `cpfor2` . When both variables are entered into the model at once, the effect of logarea is controlled for, making it possible to detect a `cpfor2` effect (and vice versa).

Exercise

Run another multiple regression, this time substituting `thtden` for `cpfor2` as an independent variable (`logherp ~ logarea + thtden`).

Solution

```
model_mden <- lm(logherp ~ logarea + thtden, data = mydata)
summary(model_mden)
```

```
Call:
lm(formula = logherp ~ logarea + thtden, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.31583 -0.12326  0.02095  0.13201  0.31674

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.37634    0.14926   2.521 0.018437 *
logarea      0.22504    0.05701   3.947 0.000567 ***
thtden      -0.04196    0.01345  -3.118 0.004535 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1606 on 25 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.5358,    Adjusted R-squared:  0.4986
F-statistic: 14.43 on 2 and 25 DF,  p-value: 6.829e-05
```

In this case we reject the null hypotheses that there are no effects of wetland area (`logarea`) and road density (`thtden`) on herptile richness (`logherp`). Note here that road density has a negative effect on richness, whereas wetland area and forested area (`cpfor2`; results from previous regression) both have positive effects on herptile richness.

The R^2 of this model is even higher than the previous multiple regression model, reflecting a higher correlation between logherp and thtden than between logherp and `cpfor2` (if you run a simple regression between logherp and thtden and compare it to the `cpfor2` regression you should be able to detect this).

Thus far, it appears that herptile richness is related to wetland area (`logarea`), road density (`thtden`), and possibly forest cover on adjacent lands (`cpfor2`). But, does it necessarily follow that if we build

a regression model with all three independent variables, that all three will show significant relationships? No, because we have not yet examined the relationship between Logarea , `cpfor2` and thtden . Suppose, for example, two of the variables (say, `cpfor2` and `thtden`) are perfectly correlated. Then the `thtden` effect is nothing more than the `cpfor2` effect (and vice versa), so that once we include one or the other in the regression model, none of the remaining variability would be explained by the third variable.

Exercise

Fit a regression model with logherp as the dependent variable and logarea , cpfor2 and thtden as the independent variables. What do you conclude?

Solution

```
model_mtri <- lm(logherp ~ logarea + cpfor2 + thtden, data = mydata)
summary(model_mtri)
```

```
Call:
lm(formula = logherp ~ logarea + cpfor2 + thtden, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.30729 -0.13779  0.02627  0.11441  0.29582

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.284765   0.191420   1.488 0.149867
logarea      0.228490   0.057647   3.964 0.000578 ***
cpfor2       0.001095   0.001414   0.774 0.446516
thtden      -0.035794   0.015726  -2.276 0.032055 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1619 on 24 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.5471,    Adjusted R-squared:  0.4904
F-statistic: 9.662 on 3 and 24 DF,  p-value: 0.0002291
```

Several things to note here:

1. The regression coefficient for `cpfor2` has become non-significant: once the variability explained by `logarea` and `thtden` is removed, a non-significant part of the remaining variability is explained by `cpfor2`.
2. The R^2 for this model (.547 is only marginally larger than the R^2 for the model with only logarea and thtden (.536, which is again consistent with the non-significant coefficient for `cpfor2`.

Note also that although the regression coefficient for `thtden` has not changed much from that obtained when just `thtden` and `logarea` were included in the fitted model (-.036 vs -.042, the standard error for

the regression coefficient for `thtden` has increased slightly, meaning the estimate is less precise. If the correlation between `thtden` and `cpfor2` was greater, the change in precision would also be greater.

We can compare the fit of the last two models (*i.e.*, the model with all 3 variables and the model with only `logarea` and `thtden` to decide which model is best to include.

```
anova(model_mtri, model_mden)
```

```
Analysis of Variance Table

Model 1: logherp ~ logarea + cpfor2 + thtden
Model 2: logherp ~ logarea + thtden
  Res.Df     RSS Df Sum of Sq     F Pr(>F)
1     24 0.62937
2     25 0.64508 -1 -0.015708 0.599 0.4465
```

Note that this is the identical result we obtained via the t-test of the effect of `cpfor2` in the model with all 3 variables above as they are testing the same thing (this should make sense to you). From this analysis, we would conclude that the full model with all three variables included does not offer a significant improvement in fit over the model with only `logarea` and `thtden`. This isn't surprising given that we already know that we cannot reject the null hypothesis of no effect of `cpfor2` in the full model. Overall, we would conclude, on the basis of these analyses, that:

1. Given the three variables `thtden` , `logarea` and `cpfor2` , the best model is one that includes the first two variables.
2. There is evidence of a negative relationship between herptile richness and the density of roads on adjacent lands.
3. There is evidence that the larger the wetland area, the greater the herptile species richness. Note that by "best", I don't mean the best possible model, I mean the best one given the three predictor variables we started with. It seems pretty clear that there are other factors controlling richness in wetlands, since even with the "best" model, almost half of the variability in richness is unexplained.

7.5. Stepwise multiple regression procedures

There are a number of techniques available for selecting the multiple regression model that best suits your data. When working with only three independent variables it is often sufficient to work through the different combinations of possible variables yourself, until you are satisfied you have fit the best model. This is, essentially, what we did in the first section of this lab. However, the process can become tedious when dealing with numerous independent variables, and you may find an automatic procedure for fitting models to be easier to work with.

Stepwise regression in R relies on the Akaike Information Criterion, as a measure of goodness of fit

$$AIC = 2k + 2ln(L))$$

where k is the number of regressors, and L is the maximized value of the likelihood function for the model). This is a statistic that rewards prediction precision while penalizing model complexity. If a new model has an AIC lower than that of the current model, the new model is a better fit to the data.

Exercise

Still working with the `Mregdat` data, run a stepwise multiple regression on the same set of variables:

```
# Stepwise Regression
step_mtri <- step(model_mtri, direction = "both")
```

```
Start:  AIC=-98.27
logherp ~ logarea + cpfor2 + thtden

          Df Sum of Sq     RSS     AIC
- cpfor2   1   0.01571 0.64508 -99.576
<none>                 0.62937 -98.267
- thtden   1   0.13585 0.76522 -94.794
- logarea  1   0.41198 1.04135 -86.167

Step:  AIC=-99.58
logherp ~ logarea + thtden

          Df Sum of Sq     RSS     AIC
<none>                 0.64508 -99.576
+ cpfor2   1   0.01571 0.62937 -98.267
- thtden   1   0.25092 0.89600 -92.376
- logarea  1   0.40204 1.04712 -88.013
```

```
step_mtri$anova # display results
```

```
      Step Df   Deviance Resid. Df Resid. Dev       AIC
1          NA         NA        24  0.6293717 -98.26666
2 - cpfor2  1 0.01570813        25  0.6450798 -99.57640
```

Examining the output, we find:

1. R calculated the AIC for the starting model (here the full model with the 3 independent variables.
2. The AIC for models where terms are deleted. Note here that the only way to reduce the AIC is to drop 2.
3. The AIC for models where terms are added or deleted from the model selected in the first step (i.e. `logherp ~ logarea + thtden`. Note that none of these models are better.

Instead of starting from the full (saturated) model and removing and possibly re-adding terms (i.e. direction = "both"), one can start from the null model and only add terms:

```
# Forward selection approach
model_null <- lm(logherp ~ 1, data = mydata)
step_f <- stepAIC(
  model_null,
  scope = ~ . + logarea + cpfor2 + thtden, direction = "forward"
)
```

```
Start:  AIC=-82.09
logherp ~ 1

          Df Sum of Sq    RSS     AIC
+ logarea  1   0.49352 0.8960 -92.376
+ thtden   1   0.34241 1.0471 -88.013
+ cpfor2   1   0.12907 1.2605 -82.820
<none>                 1.3895 -82.091

Step:  AIC=-92.38
logherp ~ logarea

         Df Sum of Sq     RSS     AIC
+ thtden  1   0.25093 0.64508 -99.576
+ cpfor2  1   0.13078 0.76522 -94.794
<none>                0.89600 -92.376

Step:  AIC=-99.58
logherp ~ logarea + thtden

         Df Sum of Sq     RSS     AIC
<none>                0.64508 -99.576
+ cpfor2  1  0.015708 0.62937 -98.267
```

```
step_f$anova # display results
```

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
logherp ~ 1

Final Model:
logherp ~ logarea + thtden

       Step Df  Deviance Resid. Df Resid. Dev       AIC
1                              27  1.3895281 -82.09073
2 + logarea  1 0.4935233        26  0.8960048 -92.37639
3  + thtden  1 0.2509250        25  0.6450798 -99.57640
```

You should first notice that the final result is the same as the default stepwise regression and as what we got building the model from scratch. In forward selection, R first fits the least complex model (i.e, with only an intercept), and then adds variables, one by one, according to AIC statistics. Thus, in the above example, the model was first fit with only an intercept. Next, `logarea` was added, followed by `thtden`. `cpfor2` was not added because it would make AIC increase to above that of the model fit with the first two variables. Generally speaking, when doing multiple regressions, it is good practice to try several different methods (e.g. all regressions, stepwise, and backward elimination, etc.) and see whether you get the same results. If you don't, then the "best" model may not be so obvious, and you will have to think very carefully about the inferences you draw. In this case, regardless of whether we use automatic, or forward/backward stepwise regression, we arrive at the same model.

When doing multiple regression, always bear in mind the following:

1. Different procedures may produce different "best" models, i.e. the "best" model obtained using forward stepwise regression needn't necessarily be the same as that obtained using backward stepwise. It is good practice to try several different methods and see whether you end up with the same result. If you don't, it is almost invariably due to multicollinearity among the independent variables.
2. Be wary of stepwise regression. As the authors of SYSTAT, another commonly used statistical package, note:

   > Stepwise regression is probably the most abused computerized statistical technique ever devised. If you think you need automated stepwise regression to solve a particular problem, you probably don't. Professional statisticians rarely use automated stepwise regression because it does not necessarily find the "best" fitting model, the "real" model, or alternative "plausible" models. Furthermore, the order in which variables enter or leave a stepwise program is usually of no theoretical significance. You are always better off thinking about why a model could generate your data and then testing that model.

3. Remember that just because there is a significant regression of Y on X doesn't mean that X causes Y: correlation does not imply causation!

7.6. Detecting multicollinearity

Multicollinearity is the presence of correlations among independent variables. In extreme cases (perfect collinearity) it will prevent you from fitting some models.

> ⚠ Warning
>
> When collinearity is not perfect, it reduces your ability to test for the effect of individual variables, but does not affect the ability of the model to predict.

The help file for the `HH` package contains this clear passage about one of the indices of multicollinearity, the variance inflation factors:

> A simple diagnostic of collinearity is the variance inflation factor, VIF one for each regression coefficient (other than the intercept). Since the condition of collinearity involves the predictors but not the response, this measure is a function of the X's but not of Y. The VIF for predictor i is
>
> $$1/(1 - R_i^2)$$

where R_i^2 is the R^2 from a regression of predictor `i` against the remaining predictors. If R_i^2 is close to 1, this means that predictor `i` is well explained by a linear function of the remaining predictors, and, therefore, the presence of predictor `i` in the model is redundant. Values of VIF exceeding 5 are considered evidence of collinearity: The information carried by a predictor having such a VIF is contained in a subset of the remaining predictors. If, however, all of a model's regression coefficients differ significantly from 0 (p-value $< .05$), a somewhat larger VIF may be tolerable.

VIFs indicate by how much the variance of each regression coefficient is increased by the presence of collinearity.

> **Note**
>
> There are several `vif()` functions (I know of at least three in the packages `car`, `HH` and `DAAG`) and I do not know if and how they differ.

To quantify multicollinarity, one can simply call the `vif()` function from the package `car`:

```
library(car)
vif(model_mtri)
```

```
 logarea   cpfor2   thtden
1.022127 1.344455 1.365970
```

Here there is no evidence that multicollinearity is a problem since all vif are close to 1.

7.7. Polynomial regression

In the regression models considered so far, we have assumed that the relationship between the dependent and independent variables is linear. If not, in some cases it can be made linear by transforming one or both variables. On the other hand, for many biological relationships no transformation in the world will help, and we are forced to go with some sort of non-linear regression method.

The simplest type of nonlinear regression method is polynomial regression, in which you fit regression models that include independent variables raised to some power greater than one, e.g. X^2, X^3, etc.

> **Exercise**
>
> Plot the relationship between the residuals of the `logherp ~ logarea` regression and `swamp`.

> **Solution**

```r
# problème avec les données de manquantes dans logherp
mysub <- subset(mydata, !is.na(logherp))
# ajouter les résidus dans les donnée
mysub$resloga <- residuals(model_loga)
ggplot(data = mysub, aes(y = resloga, x = swamp)) +
  geom_point() +
  geom_smooth()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

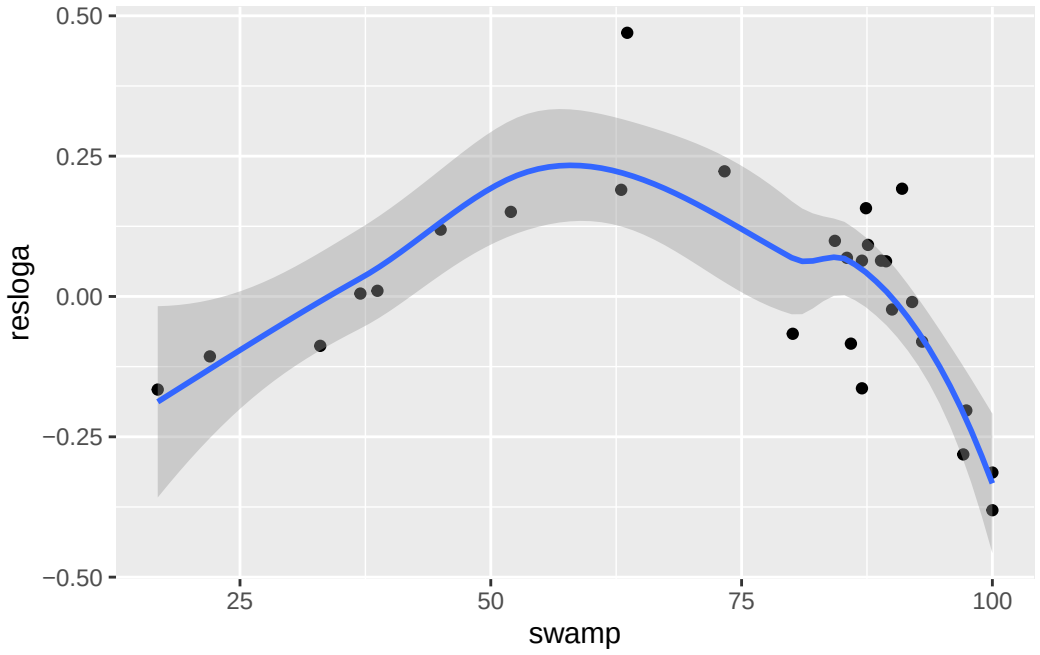


Figure 7.3.: Relation entre swamp et les résidus de la régression entre logherp et logarea

Visual inspection of this graph suggests that there is a strong, but highly nonlinear, relationship between these two variables.

Exercise

Try regressing the residuals of the `logherp ~ logarea` regression on `swamp`. What do you conclude?

Solution

```r
model_resloga <- lm(resloga ~ swamp, mysub)
summary(model_resloga)
```

```
Call:
```

```
lm(formula = resloga ~ swamp, data = mysub)

Residuals:
     Min       1Q   Median       3Q      Max
-0.35088 -0.13819  0.00313  0.10849  0.45802

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.084571   0.109265   0.774    0.446
swamp       -0.001145   0.001403  -0.816    0.422

Residual standard error: 0.1833 on 26 degrees of freedom
Multiple R-squared:  0.02498,   Adjusted R-squared:  -0.01252
F-statistic: 0.666 on 1 and 26 DF,  p-value: 0.4219
```

In other words, the fit is terrible, even though you can see from the graph that there is in fact quite a strong relationship between the two - it's just that it is a non-linear relationship. (If you look at model assumptions for this model, you will see strong evidence of nonlinearity, as expected) The pattern might be well described by a quadratic relation.

Exercise

Rerun the above regression but add a second term in the Formula field to represent swamp^2 . If you simply add swamp^2 in the model R won't fit a quadratic effect, you need to use the function`I()` which indicates that the formula within should be evaluated before fitting the model.
The expression should appear as:

$$residuals\ swamp + I(swamp^2)$$

.
What do you conclude? What does examination of the residuals from this multiple regression tell you?

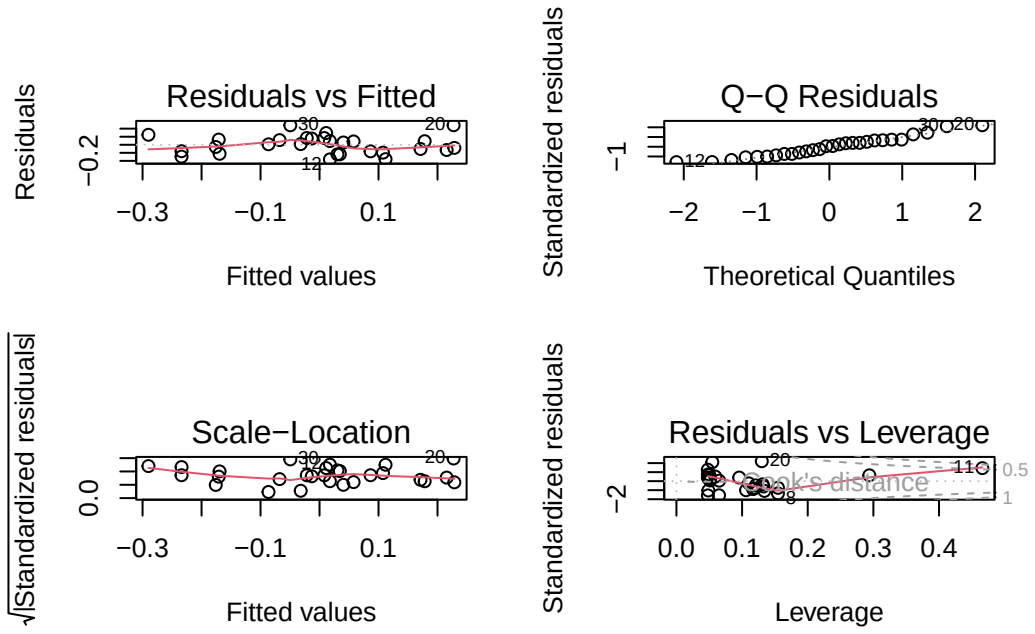
Solution

```
model_resloga2 <- lm(resloga ~ swamp + I(swamp^2), mysub)
summary(model_resloga2)
```

```
Call:
lm(formula = resloga ~ swamp + I(swamp^2), data = mysub)

Residuals:
      Min        1Q    Median        3Q       Max
-0.181185 -0.085350  0.007377  0.067327  0.242455

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -7.804e-01  1.569e-01  -4.975 3.97e-05 ***
swamp        3.398e-02  5.767e-03   5.892 3.79e-06 ***
I(swamp^2)  -2.852e-04  4.624e-05  -6.166 1.90e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1177 on 25 degrees of freedom
Multiple R-squared:  0.6132,    Adjusted R-squared:  0.5823
F-statistic: 19.82 on 2 and 25 DF,  p-value: 6.972e-06
```

```r
par(mfrow = c(2, 2))
plot(model_resloga2)
```



It is clear that once the effects of area are controlled for, a considerable amount of the remaining variability in herptile richness is explained by `swamp` , in a nonlinear fashion. If you examine model assumptions, you will see that compared to the linear model, the fit is much better.

Based on the results from the above analyses, how would you modify the regression model arrived at above? What, in your view, is the "best" overall model? Why? How would you rank the various factors in terms of their effects on herptile species richness?

In light of these results, we might want to try and fit a model which includes `logarea`, `thtden`, `cpfor2`, `swamp` and `swamp^2^` :

Solution

```r
model_poly1 <- lm(
  logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2),
  data = mydata
)
summary(model_poly1)
```

```
Call:
lm(formula = logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2),
    data = mydata)

Residuals:
      Min        1Q    Median        3Q       Max
-0.201797 -0.056170 -0.002072  0.051814  0.205626

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.203e-01  1.813e-01  -1.766   0.0912 .
logarea      2.202e-01  3.893e-02   5.656 1.09e-05 ***
cpfor2      -7.864e-04  9.955e-04  -0.790   0.4380
thtden      -2.929e-02  1.048e-02  -2.795   0.0106 *
swamp        3.113e-02  5.898e-03   5.277 2.70e-05 ***
I(swamp^2)  -2.618e-04  4.727e-05  -5.538 1.45e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1072 on 22 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.8181,    Adjusted R-squared:  0.7767
F-statistic: 19.78 on 5 and 22 DF,  p-value: 1.774e-07
```

Note that on the basis of this analysis, we could potentially drop `cpfor2` and refit using the remaining variables:

Solution

```r
model_poly2 <- lm(
  logherp ~ logarea + thtden + swamp + I(swamp^2),
  data = mydata
)
summary(model_poly2)
```

```
Call:
lm(formula = logherp ~ logarea + thtden + swamp + I(swamp^2),
    data = mydata)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.19621 -0.05444 -0.01202  0.07116  0.21295

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.461e-01  1.769e-01  -1.957   0.0626 .
logarea      2.232e-01  3.842e-02   5.810 6.40e-06 ***
thtden      -2.570e-02  9.364e-03  -2.744   0.0116 *
swamp        2.956e-02  5.510e-03   5.365 1.89e-05 ***
I(swamp^2)  -2.491e-04  4.409e-05  -5.649 9.46e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1063 on 23 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.8129,    Adjusted R-squared:  0.7804
F-statistic: 24.98 on 4 and 23 DF,  p-value: 4.405e-08
```

How about multicollinearity in this model?

```
vif(model_poly2)
```

```
   logarea     thtden      swamp I(swamp^2)
  1.053193   1.123491  45.845845  45.656453
```

VIF for the two swamp terms are much higher than the standard threshold of 5. However, this is expected for polynomial terms, and not really a concern given that both terms are highly significant in the model. The high VIF means that these two coefficients are not estimated precisely, but using both in the model still allows to make a good prediction (i.e. account for the response to swamp).

7.8. Checking assumptions of a multiple regression model

All the model selection techniques or the manual model crafting assumes that the standard assumptions (independence, normality, homoscedasticity, linearity) are met. Given that a large number of models can be fitted, it may seem that testing the assumptions at each step would be an herculean task. However, it is generally sufficient to examine the residuals of the full (saturated) model and of the final model. Terms not contributing significantly to the fit do not affect residuals much, and therefore, the residuals to the full model, or the residuals to the final model, are generally sufficient.

Let's have a look at the diagnostic plots for the final model. Here we use the `check_model()` function from the `performance` .

Solution

```
library(performance)
check_model(model_poly2)
```

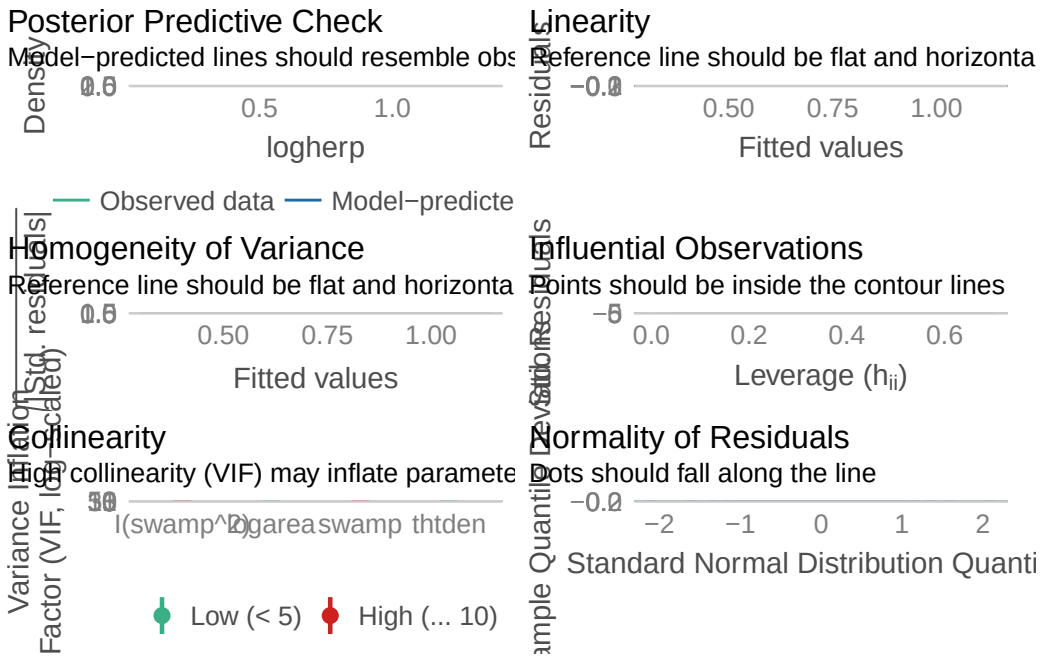


Figure 7.4.: Conditions d'application du modèle `model_poly2`

Alternatively it can be done with the classic method

Code

```
par(mfrow = c(2, 2))
plot(model_poly2)
```

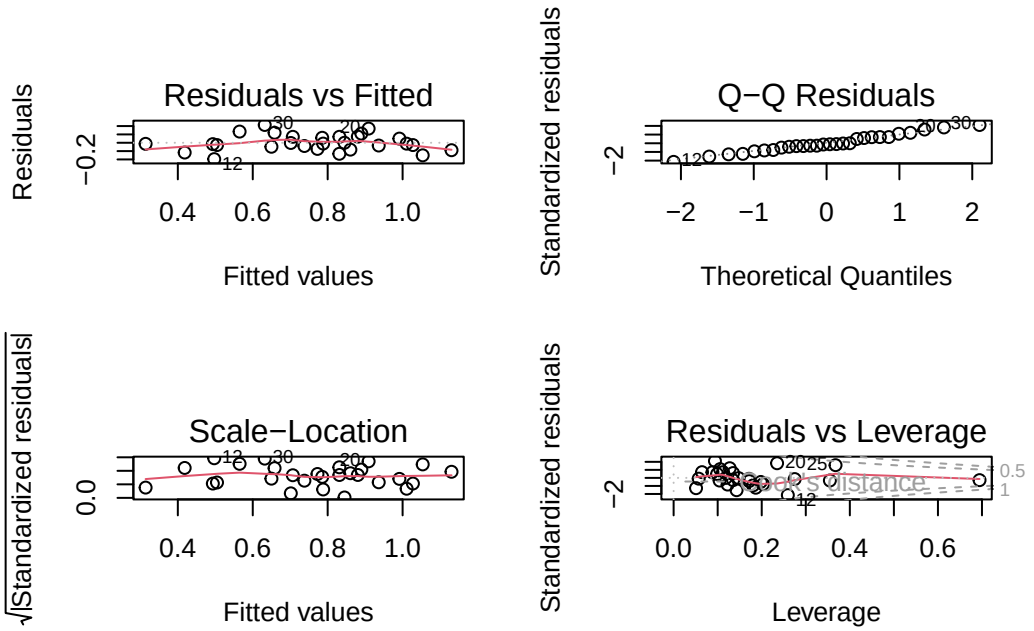


Figure 7.5.: Conditions d'application du modèle `model_poly2`

Everything looks about right here. For the skeptic, let's run the formal tests.

```
shapiro.test(residuals(model_poly2))
```

```
    Shapiro-Wilk normality test

data:  residuals(model_poly2)
W = 0.9837, p-value = 0.9278
```

The residuals do not deviate from normality. Good.

```
library(lmtest)
bptest(model_poly2)
```

```
    studentized Breusch-Pagan test

data:  model_poly2
BP = 3.8415, df = 4, p-value = 0.4279
```

No deviation from homoscedasticity either. Good.

```
dwtest(model_poly2)
```

```
    Durbin-Watson test

data:  model_poly2
DW = 1.725, p-value = 0.2095
alternative hypothesis: true autocorrelation is greater than 0
```

No serial correlation in the residuals, so no evidence of non-independence.

```
resettest(model_poly2, type = "regressor", data = mydata)
```

```
    RESET test

data:  model_poly2
RESET = 0.9823, df1 = 8, df2 = 15, p-value = 0.4859
```

And no significant deviation from linearity. So it seems that all is fine.

7.9. Visualizing effect size

How about effect size? How is that measured or viewed? The regression coefficients can be used to measure effect size, although it may be better to standardize them so that they become independent of measurement units. But a graph is often useful as well. In this context, some of the most useful graphs are called partial residual plots (or component + residual plots). These plots show how the dependent variable, corrected for other variables in the model, varies with each individual variable. Let's have a look:

```
# Evaluate visually linearity and effect size
# component + residual plot
crPlots(model_poly2)
```

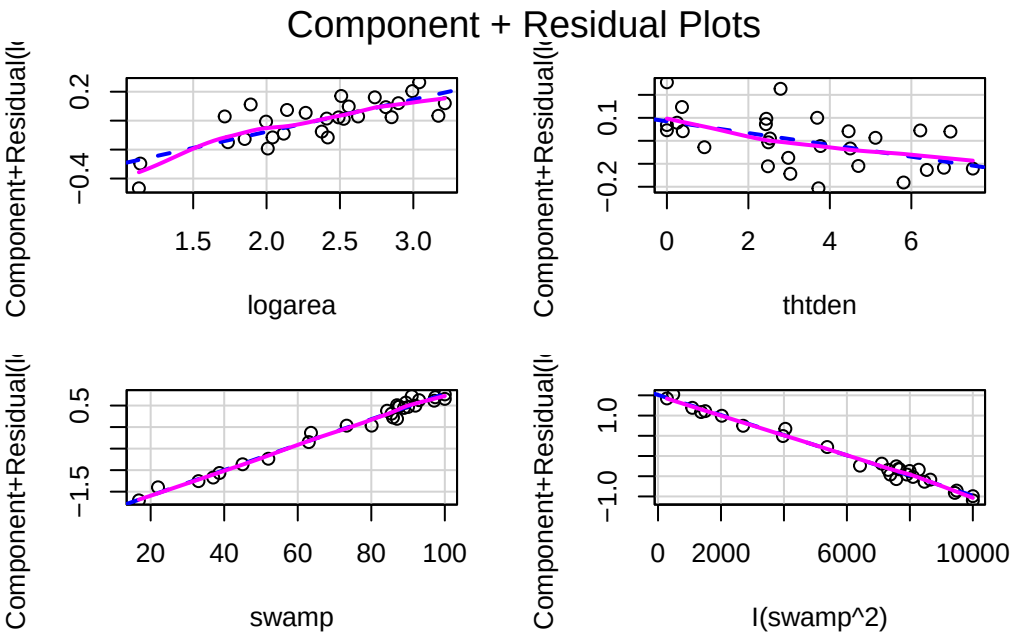


Figure 7.6.: Graphiques de résidus partiels du modèle `model_poly2`

Note that the vertical scale varies among plots. For `thtden`, the dependent variable (log10(herptile richness)) varies by about 0.4 units over the range of `thtden` in the sample. For `logarea`, the variation is about 0.6 log units. For `swamp`, it is a bit tricky since there are two terms and they have opposite effect (leading to a peaked relationship), so the plots are less informative. However, there is no deviation from linearity to be seen.

To illustrate what these graphs would look like if there was deviation from linearity, let's drop swamp^2 term and produce the graphs and run the RESET test

Solution

```
model_nopoly <- lm(
  logherp ~ logarea + thtden + swamp,
  data = mydata
)
crPlots(model_nopoly)
```

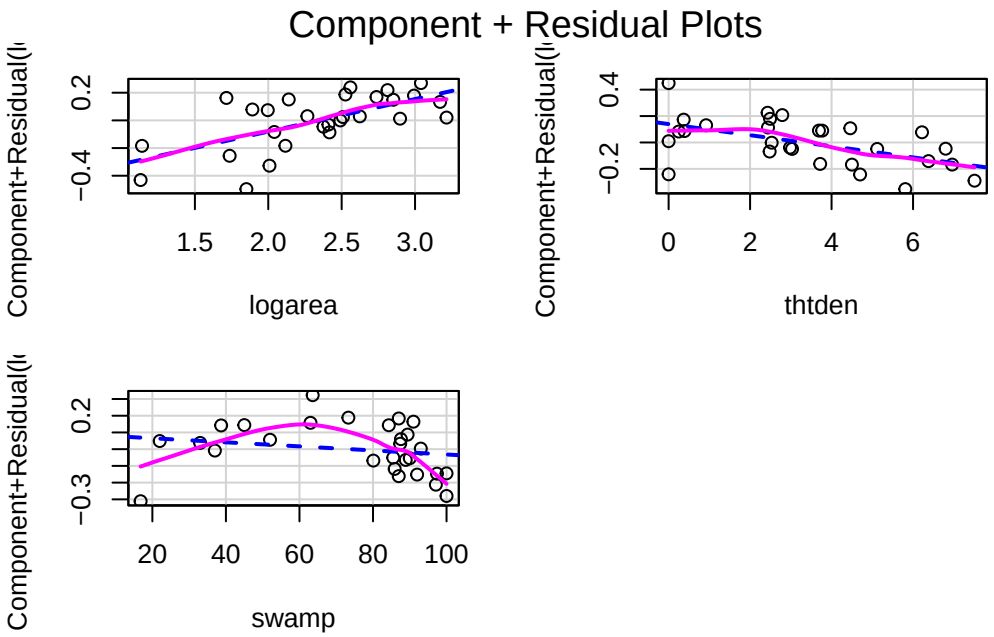


Figure 7.7.: Graphiques de résidus partiels du modèle `model_nopoly`

The lack of linearity along the gradient of swamp becomes obvious. The RESET test also detects a violation from linearity:

```
resettest(model_nopoly, type = "regressor")
```

```
    RESET test

data:  model_nopoly
RESET = 6.7588, df1 = 6, df2 = 18, p-value = 0.0007066
```

7.10. Testing for interactions

When there are multiple independent variables one should always be ready to assess interactions. In most multiple regression contexts this is somewhat difficult because adding interaction terms increases overall multicollinearity and because in many cases there are not enough observations to test all interactions, or the observations are not well balanced to make powerful tests for interactions. Going back to our final model, see what happens if one tries to fit the fully saturated model with all interactions:

```
fullmodel_withinteractions <- lm(
  logherp ~ logarea * cpfor2 * thtden * swamp * I(swamp^2),
  data = mydata
)
summary(fullmodel_withinteractions)
```

```
Call:
lm(formula = logherp ~ logarea * cpfor2 * thtden * swamp * I(swamp^2),
    data = mydata)

Residuals:
ALL 28 residuals are 0: no residual degrees of freedom!

Coefficients: (4 not defined because of singularities)
                                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                           -5.948e+03        NaN     NaN      NaN
logarea                                3.293e+03        NaN     NaN      NaN
cpfor2                                 7.080e+01        NaN     NaN      NaN
thtden                                 9.223e+02        NaN     NaN      NaN
swamp                                  1.176e+02        NaN     NaN      NaN
I(swamp^2)                            -3.517e-01        NaN     NaN      NaN
logarea:cpfor2                        -3.771e+01        NaN     NaN      NaN
logarea:thtden                        -4.781e+02        NaN     NaN      NaN
cpfor2:thtden                         -1.115e+01        NaN     NaN      NaN
logarea:swamp                         -7.876e+01        NaN     NaN      NaN
cpfor2:swamp                          -1.401e+00        NaN     NaN      NaN
thtden:swamp                          -1.920e+01        NaN     NaN      NaN
logarea:I(swamp^2)                     5.105e-01        NaN     NaN      NaN
cpfor2:I(swamp^2)                      3.825e-03        NaN     NaN      NaN
thtden:I(swamp^2)                      7.826e-02        NaN     NaN      NaN
swamp:I(swamp^2)                      -2.455e-03        NaN     NaN      NaN
logarea:cpfor2:thtden                  5.359e+00        NaN     NaN      NaN
logarea:cpfor2:swamp                   8.743e-01        NaN     NaN      NaN
logarea:thtden:swamp                   1.080e+01        NaN     NaN      NaN
cpfor2:thtden:swamp                    2.620e-01        NaN     NaN      NaN
logarea:cpfor2:I(swamp^2)             -5.065e-03        NaN     NaN      NaN
logarea:thtden:I(swamp^2)             -6.125e-02        NaN     NaN      NaN
cpfor2:thtden:I(swamp^2)              -1.551e-03        NaN     NaN      NaN
logarea:swamp:I(swamp^2)              -4.640e-04        NaN     NaN      NaN
cpfor2:swamp:I(swamp^2)                3.352e-05        NaN     NaN      NaN
thtden:swamp:I(swamp^2)                2.439e-04        NaN     NaN      NaN
logarea:cpfor2:thtden:swamp           -1.235e-01        NaN     NaN      NaN
logarea:cpfor2:thtden:I(swamp^2)       7.166e-04        NaN     NaN      NaN
logarea:cpfor2:swamp:I(swamp^2)               NA         NA      NA       NA
logarea:thtden:swamp:I(swamp^2)               NA         NA      NA       NA
cpfor2:thtden:swamp:I(swamp^2)                NA         NA      NA       NA
logarea:cpfor2:thtden:swamp:I(swamp^2)        NA         NA      NA       NA

Residual standard error: NaN on 0 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:      1, Adjusted R-squared:    NaN
F-statistic:   NaN on 27 and 0 DF,  p-value: NA
```

Indeed, it is not possible to include all 32 terms with only 28 observations. There are not enough data points, R square is one, and the model perfectly overfits the data.

If you try to use an automated routine to "pick" the best model out of this soup, R complains:

```
step(fullmodel_withinteractions)
```

```
Error in step(fullmodel_withinteractions): AIC is -infinity for this model, so 'step' cannot proc
```

Does this mean you can forget about potential interactions and simply accept the final model without a thought? No. You simply do not have enough data to test for all interactions. But there is a compromise worth attempting, comparing the final model to a model with a subset of the interactions, say all second order interactions, to check whether the inclusion of these interactions improves substantially the fit:

```
full_model_2ndinteractions <- lm(
  logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2)
    + logarea:cpfor2
    + logarea:thtden
    + logarea:swamp
    + cpfor2:thtden
    + cpfor2:swamp
    + thtden:swamp,
  data = mydata
)
summary(full_model_2ndinteractions)
```

```
Call:
lm(formula = logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2) +
    logarea:cpfor2 + logarea:thtden + logarea:swamp + cpfor2:thtden +
    cpfor2:swamp + thtden:swamp, data = mydata)

Residuals:
      Min        1Q    Median        3Q       Max
-0.216880 -0.036534  0.003506  0.042990  0.175490

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     4.339e-01  6.325e-01   0.686 0.502581
logarea        -1.254e-01  2.684e-01  -0.467 0.646654
cpfor2         -9.344e-03  7.205e-03  -1.297 0.213032
thtden         -1.833e-01  9.035e-02  -2.028 0.059504 .
swamp           3.569e-02  7.861e-03   4.540 0.000334 ***
I(swamp^2)     -3.090e-04  7.109e-05  -4.347 0.000500 ***
logarea:cpfor2  2.582e-03  2.577e-03   1.002 0.331132
logarea:thtden  7.017e-02  3.359e-02   2.089 0.053036 .
logarea:swamp  -5.290e-04  2.249e-03  -0.235 0.816981
cpfor2:thtden  -2.095e-04  6.120e-04  -0.342 0.736544
cpfor2:swamp    4.651e-05  5.431e-05   0.856 0.404390
```

```
thtden:swamp    2.248e-04  4.764e-04   0.472 0.643336
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.108 on 16 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.8658,    Adjusted R-squared:  0.7735
F-statistic: 9.382 on 11 and 16 DF,  p-value: 4.829e-05
```

This model fits the data slightly better than the "final" model (it explains 86.6% of the variance in `logherp`, compared to 81.2% for the "final" model without interactions), but has twice as many parameters.

If you look at the individual coefficients, some weird things happen: for example, the sign for logarea has changed. This is one of the symptoms of multicollinearity. Let's look at the variance inflation factors:

```
vif(full_model_2ndinteractions)
```

```
there are higher-order terms (interactions) in this model
consider setting type = 'predictor'; see ?vif
```

```
       logarea         cpfor2         thtden          swamp     I(swamp^2)
      49.86060       78.49622      101.42437       90.47389      115.08457
logarea:cpfor2 logarea:thtden  logarea:swamp  cpfor2:thtden   cpfor2:swamp
      66.97792       71.69894       67.27034       14.66814       29.41422
  thtden:swamp
      20.04410
```

Ouch. All VIF are above 5, not only the ones involving the swamp terms. This model is not very satisfying it seems. Indeed the AIC for the two models indicate that the model with interactions has less information than the full model (remember, models with the lowest AIC value are to be preferred):

```
AIC(model_poly1)
```

```
[1] -38.3433
```

```
AIC(full_model_2ndinteractions)
```

```
[1] -34.86123
```

The `anova()` command can be used to test whether the addition of all interaction terms improves the fit significantly:

```
anova(model_poly1, full_model_2ndinteractions)
```

```
Analysis of Variance Table

Model 1: logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2)
Model 2: logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2) + logarea:cpfor2 +
    logarea:thtden + logarea:swamp + cpfor2:thtden + cpfor2:swamp +
    thtden:swamp
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     22 0.25282
2     16 0.18651  6  0.066314 0.9481  0.489
```

This test indicates that the addition of interaction terms did not reduce significantly the residual variance around the full model. How about a comparison with the final model without `cpfor2`?

```
anova(model_poly2, full_model_2ndinteractions)
```

```
Analysis of Variance Table

Model 1: logherp ~ logarea + thtden + swamp + I(swamp^2)
Model 2: logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2) + logarea:cpfor2 +
    logarea:thtden + logarea:swamp + cpfor2:thtden + cpfor2:swamp +
    thtden:swamp
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     23 0.25999
2     16 0.18651  7  0.073486 0.9006 0.5294
```

And this comparison suggests that our final model does not make worse predictions than the full model with interactions.

7.11. Dredging and the information theoretical approach

One of the main critiques of stepwise methods is that the p-values are not strictly correct because of the large number of tests that are actually done. This is the multiple testing problem. In building linear models (multiple regression for example) from a large number of independent variables, and possibly their interactions, there are so many possible combinations that if one were to use Bonferroni type corrections, it would make tests very conservative.

An alternative, very elegantly advocated by Burnham and Anderson (2002, Model selection and multimodel inference: a practical information-theoretic approach. 2nd ed), is to use AIC (or better the AICc that is more appropriate for samples where the number of observations is less that about 40 times the number of variables) to rank potential models, and identify the set of models that are the best ones. One can then average the parameters across models, weighting using the probability that it is the best model to obtain coefficients that are more robust and less likely to be unduly affected by multicollinearity.

> **Warning**
>
> To compare models using *AIC*, models need to be fitted using the exact same data for each model. You thus need to be careful that there are no missing data when using an AIC based approach to model selection

The approach of comparing model fit using AIC was first developed to compare a set of model carefully build and chosen by the person doing the analysis based on a-priori knowledge and biological hypotheses. Some, however, developped an approach that I consider brainless and brutal to fit all potential models and then compare them using *AIC*. This approach has been implemented in the package MuMIn.

> **Note**
>
> I do not support the use of stepwise AIC or data dredging which are going against the philosophy of AIC and parsimony. Develop a model based on biological hypothesis and report all the results significant or not without dredging the data.

```
# redo the model double chekcing there are no "NA"
# specifying na.action

full_model_2ndinteractions <- update(
  full_model_2ndinteractions,
  . ~ .,
  data = mysub,
  na.action = "na.fail"
)

library(MuMIn)
dd <- dredge(full_model_2ndinteractions)
```

```
Fixed term is "(Intercept)"
```

Object `dd` will contain all possible models using the terms of our full model with 2nd order interactions. Then, we can have a look at the subset of models that have an AICc within 4 units from the lowest AICc model. (Burnham and Anderson suggest that models that deviate by more than 2 AICc units have very little empirical support):

```
# get models within 4 units of AICc from the best model
top_models_1 <- get.models(dd, subset = delta < 4)
avgmodel1 <- model.avg(top_models_1) # compute average parameters
summary(avgmodel1) # display averaged model
```

```
Call:
model.avg(object = top_models_1)
```

```
Component model call:
lm(formula = logherp ~ <8 unique rhs>, data = mysub, na.action =
     na.fail)

Component models:
       df logLik   AICc delta weight
23457   7  27.78 -35.95  0.00   0.34
2345    6  25.78 -35.56  0.39   0.28
123457  8  28.30 -33.02  2.93   0.08
234578  8  28.26 -32.95  3.00   0.08
12345   7  26.17 -32.74  3.21   0.07
23458   7  26.06 -32.51  3.44   0.06
234567  8  27.88 -32.17  3.78   0.05
23456   7  25.79 -31.99  3.97   0.05

Term codes:
        cpfor2     I(swamp^2)        logarea          swamp         thtden
             1              2              3              4              5
 logarea:swamp logarea:thtden   swamp:thtden
             6              7              8

Model-averaged coefficients:
(full average)
                 Estimate Std. Error Adjusted SE z value Pr(>|z|)
(Intercept)    -2.075e-01  2.484e-01   2.593e-01   0.800    0.424
logarea         1.314e-01  1.185e-01   1.222e-01   1.076    0.282
swamp           3.193e-02  6.125e-03   6.438e-03   4.960    7e-07 ***
I(swamp^2)     -2.676e-04  4.904e-05   5.154e-05   5.193    2e-07 ***
thtden         -6.843e-02  5.324e-02   5.459e-02   1.254    0.210
logarea:thtden  2.139e-02  2.506e-02   2.565e-02   0.834    0.404
cpfor2         -1.202e-04  4.710e-04   4.886e-04   0.246    0.806
swamp:thtden   -3.277e-05  1.419e-04   1.475e-04   0.222    0.824
logarea:swamp   4.378e-05  5.378e-04   5.676e-04   0.077    0.939

(conditional average)
                 Estimate Std. Error Adjusted SE z value Pr(>|z|)
(Intercept)    -2.075e-01  2.484e-01   2.593e-01   0.800   0.4236
logarea         1.314e-01  1.185e-01   1.222e-01   1.076   0.2820
swamp           3.193e-02  6.125e-03   6.438e-03   4.960    7e-07 ***
I(swamp^2)     -2.676e-04  4.904e-05   5.154e-05   5.193    2e-07 ***
thtden         -6.843e-02  5.324e-02   5.459e-02   1.254   0.2100
logarea:thtden  3.924e-02  2.125e-02   2.251e-02   1.743   0.0813 .
cpfor2         -8.187e-04  9.692e-04   1.027e-03   0.797   0.4253
swamp:thtden   -2.402e-04  3.127e-04   3.313e-04   0.725   0.4684
logarea:swamp   4.462e-04  1.664e-03   1.762e-03   0.253   0.8001
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
confint(avgmodel1) # display CI for averaged coefficients
```

```
                      2.5 %        97.5 %
(Intercept)    -0.7157333646  0.3007147516
logarea        -0.1080048582  0.3708612563
swamp           0.0193158426  0.0445532538
I(swamp^2)     -0.0003686653 -0.0001666418
thtden         -0.1754184849  0.0385545120
logarea:thtden -0.0048800385  0.0833595106
cpfor2         -0.0028313465  0.0011940283
swamp:thtden   -0.0008894138  0.0004090457
logarea:swamp  -0.0030067733  0.0038991294
```

1. **components models:** You first get the list of the models with an AICc within the desired 4 units of the best model. The variables that are included in the model are coded with the key just below.
2. For each model, in addition to the AICc, the Akaike weights are calculated. They represent the relative likelihood of a model, and indicate the relative importance of a model compared to the other models tested.
3. **Mode-averaged coefficients:** For the subset of models, weighted averages (using Akaike weights) for model parameters are calculated, with 95% CI. Note that, by default, terms missing from a model are assumed to have a coefficient of 0.

7.12. Bootstrapping multiple regression

When data do not meet the assumptions of normality and homoscedasticity and it is not possible to transform the data to meet the assumptions, bootstraping can be used to compute confidence intervals for coefficients. If the distribution of the bootstrapped coefficients is symmetrical and approximately Gaussian, then empirical percentiles can be used to estimate the confidence limits.

The following code, using the `simpleboot` has been designed to be easily modifiable and will compute CI using empirical percentiles. Following this is an easier approach using the library boot that will calculate several different bootstrap confidence limits.

```
#############################################################
#######
# Bootstrap analysis the simple way with library simpleboot
# Define model to be bootstrapped and the data source used
mymodel <- lm(logherp ~ logarea + thtden + swamp + I(swamp^2), data = mydata)
# Set the number of bootstrap iterations
nboot <- 1000
library(simpleboot)
# R is the number of bootstrap iterations
# Setting rows to FALSE indicates resampling of residuals
mysimpleboot <- lm.boot(mymodel, R = nboot, rows = FALSE)
# Extract bootstrap coefficients
myresults <- sapply(mysimpleboot$boot.list, function(x) x$coef)
```

```
# Transpose matrix so that lines are bootstrap iterations
# and columns are coefficients
tmyresults <- t(myresults)
```

You can then plot the results using the follwoing code. When run, it will pause to let you have a look at the distribution for each coefficient in the model by producing plots like:

```
# Plot histograms of bootstrapped coefficients
ncoefs <- length(data.frame(tmyresults))
par(mfrow = c(1, 2), mai = c(0.5, 0.5, 0.5, 0.5), ask = TRUE)
for (i in 1:ncoefs) {
  lab <- colnames(tmyresults)[i]
  x <- tmyresults[, i]
  plot(density(x), main = lab, xlab = "")
  abline(v = mymodel$coef[i], col = "red")
  abline(v = quantile(x, c(0.025, 0.975)))
  hist(x, main = lab, xlab = "")
  abline(v = quantile(x, c(0.025, 0.975)))
  abline(v = mymodel$coef[i], col = "red")
}
```

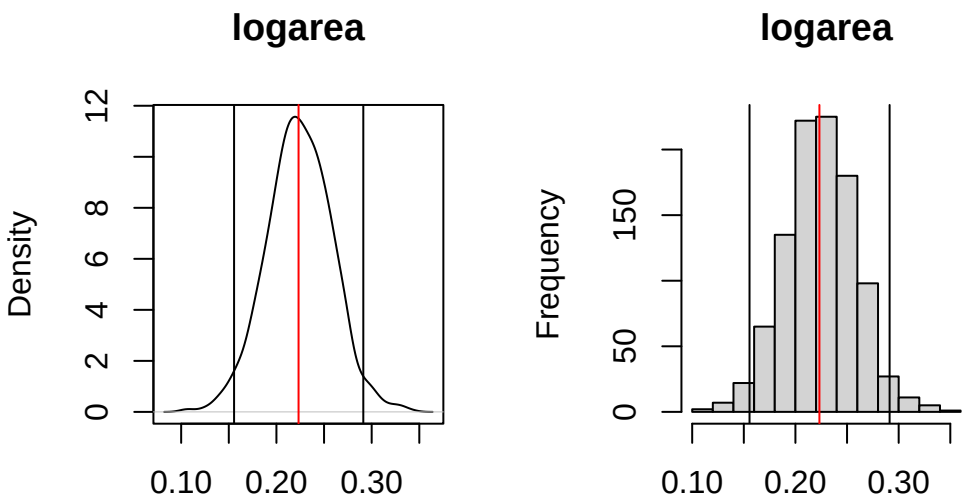


Figure 7.8.: Distribution of bootstrapped estimates for `logarea`

The top plot is the probability density function and the bottom one is the histogram of the bootstrap estimates for the coefficient. On these plots, the red line indicate the value of the parameter in the ordinary analysis, and the two vertical black lines mark the limits of the 95% confidence interval. Here the CI does not include 0 and one can conclude that the effect of logarea on logherp is significantly positive.

Precise values for the limits can be obtained by:

```
# Display empirical bootstrap quantiles (not corrected for bias)
p <- c(0.005, 0.01, 0.025, 0.05, 0.95, 0.975, 0.99, 0.995)
apply(tmyresults, 2, quantile, p)
```

```
      (Intercept)   logarea       thtden      swamp    I(swamp^2)
0.5%  -0.71693552 0.1377196 -0.046925489 0.01704594 -0.0003448064
1%    -0.69848646 0.1418898 -0.044431745 0.01845862 -0.0003318199
2.5%  -0.66306879 0.1555415 -0.040750495 0.01996994 -0.0003241589
5%    -0.61637340 0.1688297 -0.038430996 0.02155766 -0.0003156309
95%   -0.10526658 0.2758776 -0.011414015 0.03791514 -0.0001843812
97.5% -0.05652584 0.2912015 -0.009577238 0.03867714 -0.0001714492
99%   -0.01355566 0.3062894 -0.006870364 0.04045081 -0.0001512678
99.5%  0.03788347 0.3220494 -0.004197562 0.04145126 -0.0001451856
```

These confidence limits are not reliable when the distribution of the bootstrap estimates deviate from Gaussian. If they do„ then it is preferable to compute so-called bias-corrected accelerated (BCa) confidence limits. The following code does just that:

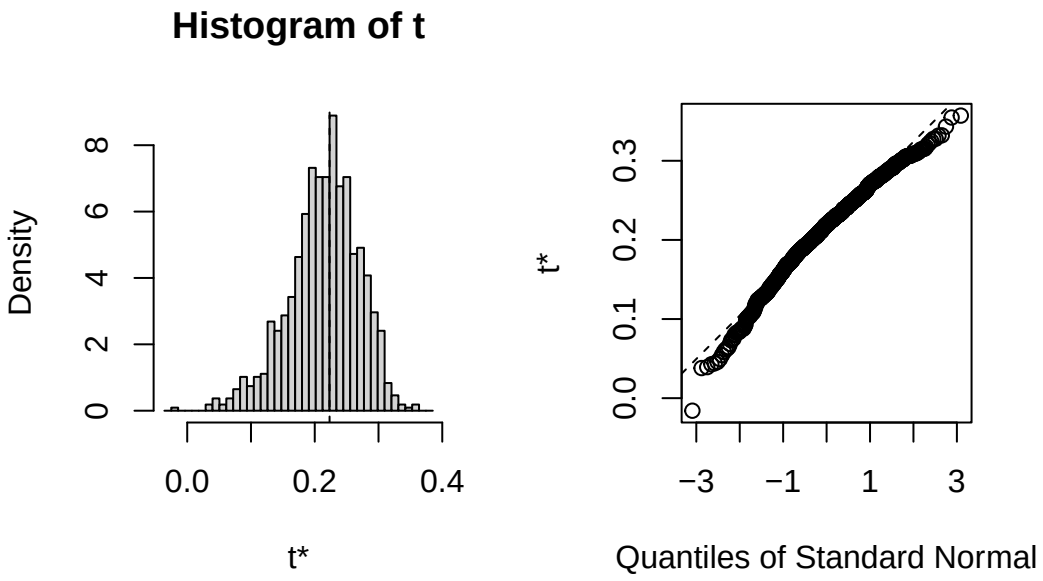
```
################################################
# Bootstrap analysis in multiple regression with BCa confidence intervals
# Preferable when parameter distribution is far from normal
# Bootstrap 95% BCa CI for regression coefficients

library(boot)
# function to obtain regression coefficients for each iteration
bs <- function(formula, data, indices) {
  d <- data[indices, ] # allows boot to select sample
  fit <- lm(formula, data = d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(
  data = mydata, statistic = bs, R = 1000,
  formula = logherp ~ logarea + thtden + swamp + I(swamp^2)
)
# view results
```

To get teh results, the following code will produce the standard graph for each coefficient and the resulting BCa interval.

```
plot(results, index = 1) # intercept
plot(results, index = 2) # logarea
plot(results, index = 3) # thtden
plot(results, index = 4) # swamp
plot(results, index = 5) # swamp2

# get 95% confidence intervals
boot.ci(results, type = "bca", index = 1)
boot.ci(results, type = "bca", index = 2)
boot.ci(results, type = "bca", index = 3)
boot.ci(results, type = "bca", index = 4)
boot.ci(results, type = "bca", index = 5)
```

For `logarea`, we get:

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 2)

Intervals :
Level       BCa
95%   ( 0.1097,  0.3170 )
Calculations and Intervals on Original Scale
```



Note that the BCa interval is from 0.12 to 0.32, whereas the simpler percentile interval is 0.16 to 0.29. BCa interval here is longer on the low side, and shorter on the high side, which it should be given the distribution of bootstrap estimates.

7.13. Permutation test

Permutation tests are more rarely performed in multiple regression contexts than bootstrap. But here is code to do it.

```
############################################################
##########
# Permutation in multiple regression
#
# using lmperm library
library(lmPerm)
# Fit desired model on the desired dataframe
my_model <- lm(logherp ~ logarea + thtden + swamp + I(swamp^2),
  data = mydata
)
my_model_prob <- lmp(
  logherp ~ logarea + thtden + swamp + I(swamp^2),
  data = mydata, perm = "Prob"
)
summary(my_model)
summary(my_model_prob)
```

8. ANCOVA and general linear model

After competing this laboratory exercise, you should be able to:

- Use R to do an analysis of covariance (ANCOVA) and interpret statistical models that have both continuous and categorical independent variables LMs
- Use R to test the assumptions underlying LMs
- Use R to compare model fits
- Use R to do bootstrap and permutation tests on LM type models including both continuous and categorical independent variables.

8.1. R packages and data

This laboratory requires the following:

- R packages:
  - ggplot2
  - car
  - lmtest
- data files
  - anc1dat.csv
  - anc3dat.csv

8.2. Linear models

GLM sometimes stands for General Linear Model, however, it is much more frequently used for *Generalized linear models*. Thus I always rather talk about *Linear models* or *LMs* instead of *General linear models* to avoid confusion in the acronym. LMs are statistical models that can be written as $Y = XB + E$, where Y is a vector (or matrix) containing the dependent variable, `X` is a matrix of independent variables, `B` is a matrix of estimated parameters, and `E` is the vector (or matrix) of independent, normally distributed and homoscedastic residuals. All tests we have seen to date (t-test, simple linear regression, One-Way ANOVA, Multiway ANOVA, and multiple regression) are LMs. Note that all models we have encountered until now contain only one type of variable (either continuous or categorical) as independent variables. In this laboratory exercise, you will fit models that have both type of independent variables. These models are also LMs.

8.3. ANCOVA

ANCOVA stands for Analysis of Covariance. It is a type of LM where there is one (or more) continuous independent variable (sometimes called a covariate) and one (or more) categorical independent variable. In the traditional treatment of ANCOVA in biostatistical textbooks, the ANCOVA model does not contain interaction terms between the continuous and categorical independent variables. Hence, the traditional ANCOVA analysis assumes that there is no interaction, and is preceeded by a test of significance of interactions, equivalent to testing that the slopes (coefficients for the continuous independent variables) do not differ among level of the categorical independent variables (a test for homogeneity of slopes). Some people, me included, use the term ANCOVA a bit more loosely for any LM that involves both continuous and categorical variables. Be aware that, depending on the author, ANCOVA may refer to a model with or without interaction terms.

8.4. Homogeneity of slopes

In many biological problems, a question arises as to whether the slopes of two or more regression lines are significantly different; for example, whether two different insecticides differ in their efficacy, whether males and females differ in their growth curves, etc. These problems call for direct comparisons of slopes. GLMs (ANCOVAs) can test for equality of slopes (homogeneity of slopes).

Remember that there are two parameters that describe a regression line, the intercept and the slope. The ANCOVA model (*sensu stricto*) tests for homogeneity of intercepts, but the starting point for the analysis is a test for homogeneity of slopes. This test can be performed by fitting a model with main effects for both the categorical and continuous independent variables, plus the interaction term(s), and testing for significance of the addition of the interaction terms.

8.4.1. Case 1 - Size as a function of age (equal slopes example)

> Exercise
>
> Using the file `anc1dat.csv` , test the hypothesis that female and male sturgeon at The Pas over the years 1978-1980 have the same observed growth rate, defined as the slope of the regression of $\log_{10}$ of fork `length`, `lfkl` , on the $\log_{10}$ age, `lage` .

First, let's have a look at the data. It would help to draw the regression line and a lowess trace to better assess linearity. Being fancy, one could also use more of R magic to spruce up the axis legends (note the use of `expression()` to get subscripts):

```
anc1dat <- read.csv("data/anc1dat.csv")
anc1dat$sex <- as.factor(anc1dat$sex)
myplot <- ggplot(data = anc1dat, aes(x = lage, y = log10(fklngth))) +
  facet_grid(. ~ sex) +
  geom_point()
myplot <- myplot +
  stat_smooth(method = lm, se = FALSE) +
  stat_smooth(se = FALSE, color = "red") +
```

```
  labs(
    y = expression(log[10] ~ (Fork ~ length)),
    x = expression(log[10] ~ (Age))
  )
myplot
```

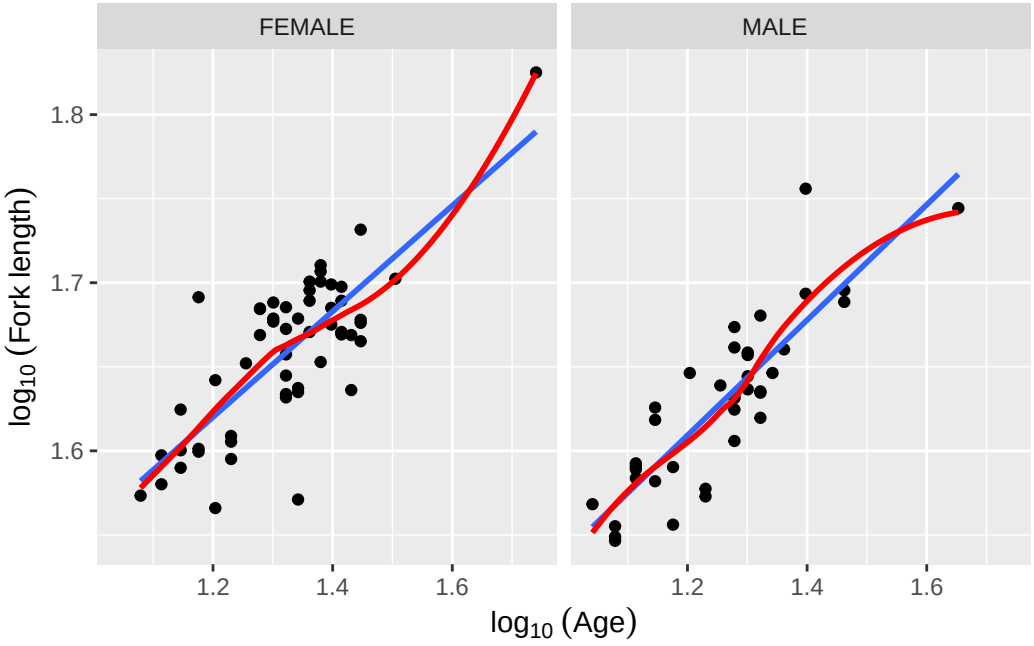


Figure 8.1.: Sturgeon length as a function of age

The log-log transformation makes the relationship linear and, at first glance, there is no issues with assumptions of LMs (although this should be confirmed by appropriate examination of the residuals). Let's fit the full model with both main effects and the interaction:

```
model.full1 <- lm(lfkl ~ sex + lage + sex:lage, data = anc1dat)
Anova(model.full1, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df  F value    Pr(>F)
(Intercept) 0.64444  1 794.8182 < 2.2e-16 ***
sex         0.00041  1   0.5043    0.4795
lage        0.07259  1  89.5312 4.588e-15 ***
sex:lage    0.00027  1   0.3367    0.5632
Residuals   0.07135 88
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the previous output, on line 4, 0.5632277 is the probability of observing an `lage:sex` interaction this strong or stronger under the null hypothesis that slope of the relationship between fork length and age does not vary between the sexes, or equivalently that the difference in fork length between males and females (if it exists) does not vary with age (and providing the assumptions of the analysis have been met).

Note

Note that I used the Anova() function with an uppercase "a" (from the car library) instead of the "built in" anova() (with a lowercase "a") command to get the results using Type III sums of squares. The type III (partial) sums of squares are calculated as if each variable was the last entered in the model and correspond to the difference in explained SS between the full model and a reduced model where only that variable is excluded. The standard anova() function returns Type I (sequential) SS, calculated as each variable is added sequentially to a null model with only an intercept. In rare cases, the type I and type III SS are equal (when the design is perfectly balanced and there is no multicolinearity). In the vast majority of cases, they will differ, and I recommend that you always use the Type III SS in your analyses.

On the basis of this analysis, we would accept the null hypotheses that:

1. the slope of the regression of log(fork length) on log(age) is the same for males and females (the interaction term is not significant)
2. that the intercepts are also the same for the two sexes (the sex term is also not significant).

But before accepting these conclusions, we should test the assumptions in the usual way

Solution

```
par(mfrow = c(2, 2))
plot(model.full1)
```

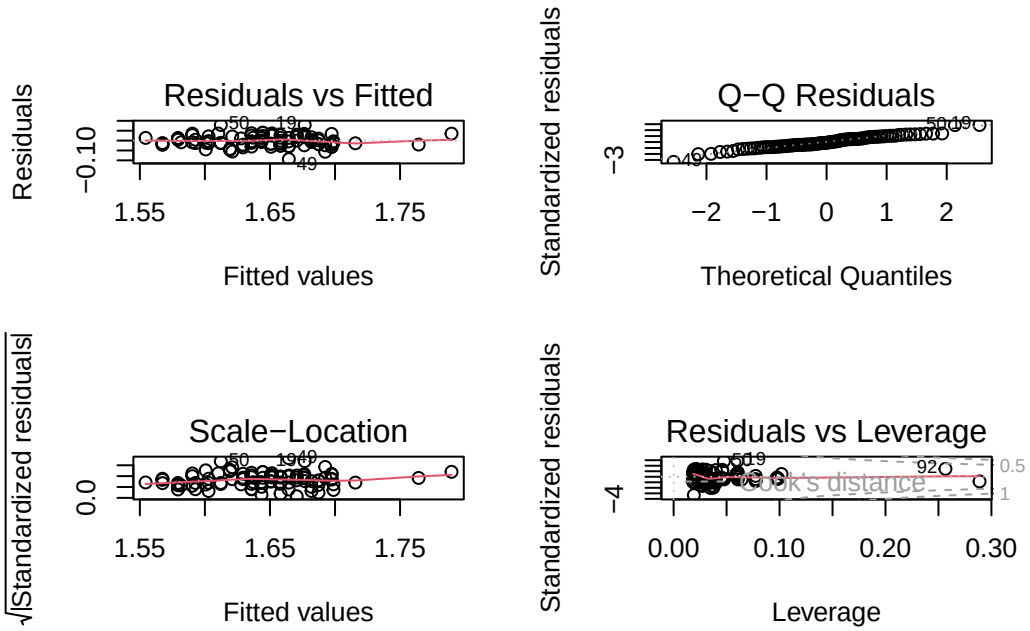


Figure 8.2.: Model assumptions for `model.full1`

With respect to normality, things look O.K., although there are several points in the top right corner that appear to lie off the line. We could also run a Wilk-Shapiro normality test and find $W = 0.9764$, $p = 0.09329$, also suggesting this assumption is valid. Homoscedasticity seems fine too, but if you want further evidence of this, you can run one of the tests. Here I use the Breusch-Pagan test, which is appropriate when some of the independent variables are continuous (Levene's test is for categorical independent variables only):

```
bptest(model.full1)
```

```
    studentized Breusch-Pagan test

data:  model.full1
BP = 0.99979, df = 3, p-value = 0.8013
```

Since the null is that the residuals are homoscedastic, and p is rather large, the test confirm the visual assessment.

Further, there is no obvious pattern in the residuals, which implies there is no problem with the assumption of linearity. This too can be formally tested:

```
resettest(model.full1, power = 2:3, type = "regressor", data = anc1dat)
```

```
    RESET test

data:  model.full1
RESET = 0.59861, df1 = 2, df2 = 86, p-value = 0.5519
```

The last assumption in this sort of analysis is that the covariate (in this case, lage) has no measurement error. We really have no way of knowing whether this assumption is justified, although multiple aging of fish by several different investigators usually gives ages that are within 1-2 years of each other, which is within the 10% considered by most to be the maximum for Type I modelling. Note that there is no "test" that you can do with the data to determine what the error is, at least in this case. If we had replicate ages for individual fish, it could be estimated quantitatively

Exercise

You will notice that there is one datum with a large studentized residual, i.e. an outlier (case 49). Eliminate this datum from your data file and rerun the analysis. Do your conclusions change?

Tip

```
model.full.no49 <- lm(lfkl ~ sex + lage + sex:lage, data = anc1dat[c(-49), ])
Anova(model.full.no49, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df  F value Pr(>F)
(Intercept) 0.64255  1 895.9394 <2e-16 ***
sex         0.00038  1   0.5273 0.4697
lage        0.07378  1 102.8746 <2e-16 ***
sex:lage    0.00022  1   0.3135 0.5770
Residuals   0.06239 87
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So the conclusion does not change if the outlier is deleted (not surprising as Cook's distance is low for this point reflecting its low leverage). Since there is no good reason to delete this data point, and since (at least qualitatively) our conclusions do not change, it is probably best to go with the full data set. A test of the assumptions for the refit model (with the outlier removed) shows that all are met, and no more outliers are detected. (I won't report these analyses, but you can and should do them just to assure yourself that everything is O.K.)

8.4.2. Case 2 - Size as a function of age (different slopes example)

> Exercise
>
> The file `anc3dat.csv` records data on male sturgeon collected at two locations (locate) , Lake of the Woods, in northwestern Ontario, and the Churchill River in northern Manitoba. Using the same procedure as outlined above (with locate as the categorical variable instead of sex), test the null hypothesis that the slope of the regression of `lfkl` on `lage` is the same in the two locations. What do you conclude?

```
anc3dat <- read.csv("data/anc3dat.csv")
myplot <- ggplot(data = anc3dat, aes(x = lage, y = log10(fklngth))) +
  facet_grid(. ~ locate) +
  geom_point() +
  stat_smooth(method = lm, se = FALSE) +
  stat_smooth(se = FALSE, color = "red") +
  labs(
    y = expression(log[10] ~ (Fork ~ length)),
    x = expression(log[10] ~ (Age))
  )
myplot
```

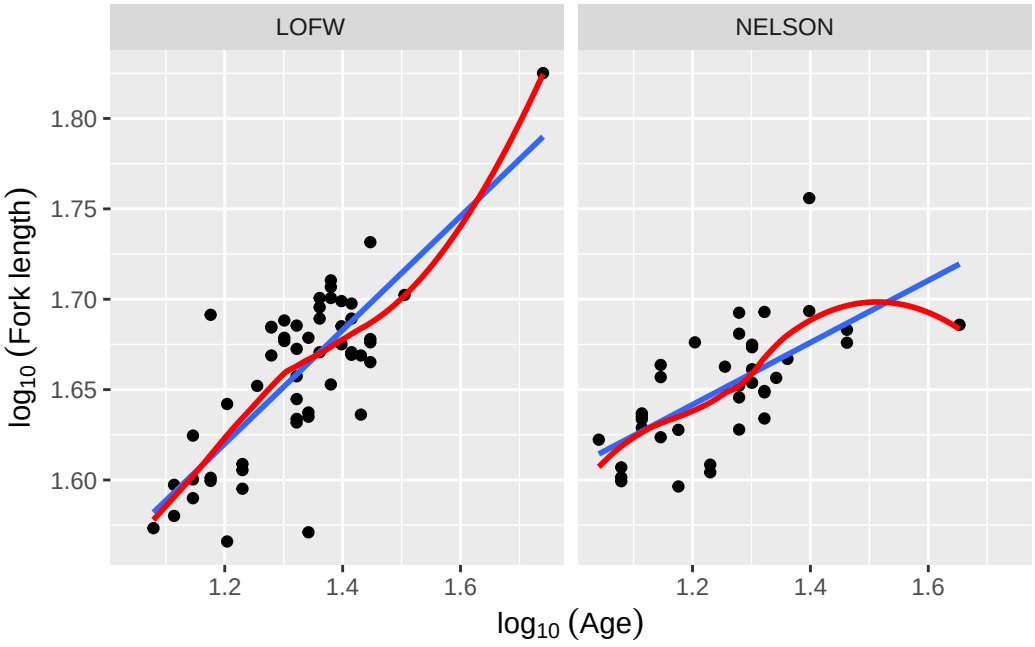


Figure 8.3.: Longueur des esturgeons en fonction de l'age d'après `anc3dat`

```
model.full2 <- lm(lfkl ~ lage + locate + lage:locate, data = anc3dat)
Anova(model.full2, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df  F value    Pr(>F)
(Intercept) 0.62951  1 1078.632 < 2.2e-16 ***
lage        0.07773  1  133.185 < 2.2e-16 ***
locate      0.00968  1   16.591 0.0001012 ***
lage:locate 0.00909  1   15.575 0.0001592 ***
Residuals   0.05136 88
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case, we reject the null hypotheses that (1) the slopes of the regressions are the same in the two locations; and (2) that the intercepts are the same in the two locations. In other words, if we want to predict the fork length of a sturgeon of a particular age (accurately) we need to know from which location it came. The fact that we reject the null hypothesis that the slopes of the lfkl - lage regressions are the same in both locations means that we should be doing individual regressions for each location separately (that is in fact what the full model is fitting). But we are jumping the gun here. Before you can trust these p values, you need to confirm that assumptions are met:

Tip

```
par(mfrow = c(2, 2))
plot(model.full2)
```

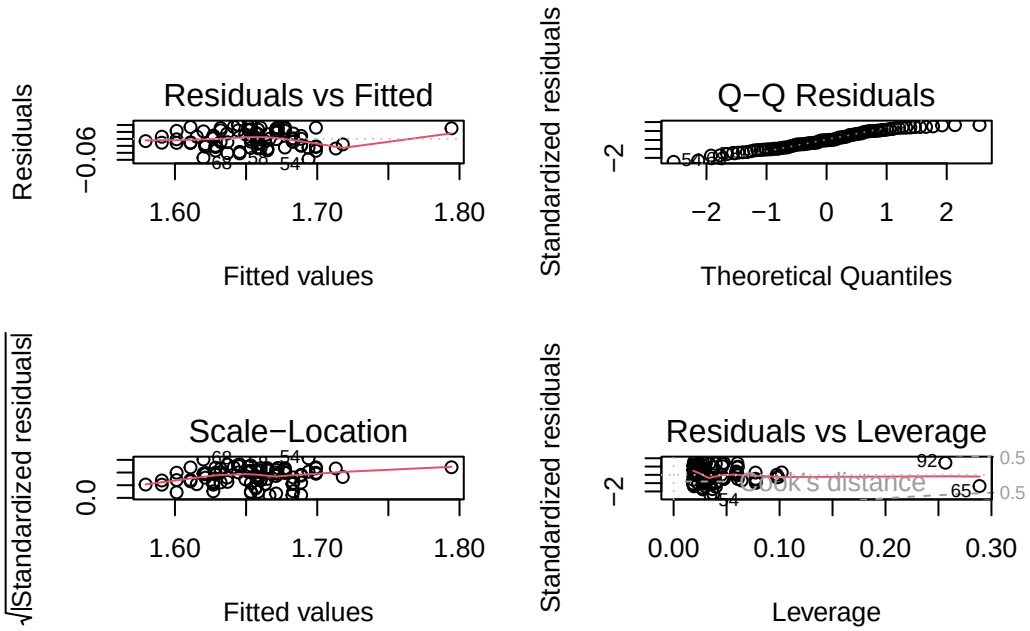


Figure 8.4.: Conditions d'applications du modèle `model.full2`

If you analyze the residuals (in the way described above), you will find that there is no problem with the linearity assumption, nor the homoscedasticity assumption (BP = 1.2267, p = 0.268). However, Wilk-Shapiro test of normality of residuals is suspicious (W=0.97, p = 0.03). Given the relatively large sample size ($N = 92$), this normality test has high power and the magnitude of deviation from normality does not appear to be large. Considering the robustness of GLM to non-normality with large samples, we should not be overly concerned with this violation.

Given that the assumptions appear sufficiently met, we can accept the results as calculated by R. All terms in the model are significant (location, lage, and the interaction). This full model is equivalent to fitting separate regressions for each location. To get the coefficients of these regression, one can either fit the two regressions on data subsets for each location, or extract the fitted coefficients from the full model:

```
model.full2
```

```
Call:
lm(formula = lfkl ~ lage + locate + lage:locate, data = anc3dat)

Coefficients:
            (Intercept)                    lage       locateNELSON
                 1.2284                  0.3253                  0.2207
lage:locateNELSON
                -0.1656
```

By default, the variable `locate` in the model is internally encoded as 0 for the location that comes first alphabetically (LofW) and 1 for the other (Nelson). So the regression equations for each location become:

For LofW:

$$\begin{aligned} lfkl &= 1.2284 + 0.3253 \times lage + 0.2207 \times 0 - 0.1656 \times 0 \times lage \\ &= 1.2284 + 0.3253 \times lage \end{aligned}$$

For Nelson:

$$\begin{aligned} lfkl &= 1.2284 + 0.3253 \times lage + 0.2207 \times 1 - 0.1656 \times 1 \times lage \\ &= 1.4491 + 0.1597 \times lage \end{aligned}$$

You can convince youself that this is the same as fitting 2 regressions separately.

```
by(anc3dat, anc3dat$locate, function(x) lm(lfkl ~ lage, data = x))
```

```
anc3dat$locate: LOFW

Call:
lm(formula = lfkl ~ lage, data = x)

Coefficients:
(Intercept)         lage
     1.2284       0.3253

------------------------------------------------------------
anc3dat$locate: NELSON

Call:
lm(formula = lfkl ~ lage, data = x)

Coefficients:
(Intercept)         lage
     1.4491       0.1597
```

8.5. The ANCOVA model

If the test for homogeneity of slopes indicates that the two or more slopes are not significantly different, i.e. there is no significant interaction between the categorical and continuous variable, then a single slope parameter can be fit. How about the intercepts? Do they differ among levels of the categorical variable? There are two school of thoughts on how to proceed to test for equality of intercepts when slopes are equal:

- The old school fits a reduced model, with the categorical and continuous variables, but no interactions (this is the ANCOVA model, *sensus stricto*) and uses the partitioned sums of squares to test for significance, say with the `Anova()` function. This approach is the one presented in many statistical textbooks.

- Others simply use the full model results, and test significance of each term from the partial sums of squares. This approach has the advantage of being faster as only one model needs to be fitted to make all inferences. However, this approach is less powerful.

In most practical cases, it does not matter unless one has very complex models with a large number of terms and higher level interactions and that many of these terms are not significant. My suggestion is that you use the faster approach first, and use the traditional approach only when you accept the null hypothesis for equal intercepts. Why? Since the faster approach is less powerful, if you nevertheless reject H0, then this conclusion will not be changed, only reinforced, by using the traditional approach.

Here I will compare the old school and the other approach. Recall that we want to assess equality of intercepts once we determined that slopes are equal. Test for equality of intercepts when slopes differ (or, if you prefer, when there is a significant interaction) are rarely directly meaningful, are often misinterpreted, and should rarely be conducted.

Going back to `anc1dat.csv`, comparing the relationships between lfkl and lage among sexes, we obtained the following results for the full model with interactions

```
Anova(model.full1, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df  F value    Pr(>F)
(Intercept) 0.64444  1 794.8182 < 2.2e-16 ***
sex         0.00041  1   0.5043    0.4795
lage        0.07259  1  89.5312 4.588e-15 ***
sex:lage    0.00027  1   0.3367    0.5632
Residuals   0.07135 88
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We already concluded that the slope of the regression for males and females does not differ (the interaction sex:lage is not significant). Note that the p-values associated with sex (0.4795) is not significant either.

For the old-school approach, one would fit a reduced model (the *sensus stricto* ANCOVA model):

```
model.ancova <- lm(lfkl ~ sex + lage, data = anc1dat)
Anova(model.ancova, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df   F value Pr(>F)
(Intercept) 1.13480  1 1410.1232 <2e-16 ***
sex         0.00149  1    1.8513 0.1771
lage        0.14338  1  178.1627 <2e-16 ***
```

```
Residuals   0.07162 89
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model.ancova)
```

```
Call:
lm(formula = lfkl ~ sex + lage, data = anc1dat)

Residuals:
      Min        1Q    Median        3Q       Max
-0.093992 -0.018457 -0.000876  0.022491  0.081161

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.225533   0.032636  37.552   <2e-16 ***
sexMALE         -0.008473   0.006228  -1.361    0.177
lage             0.327253   0.024517  13.348   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02837 on 89 degrees of freedom
Multiple R-squared:  0.696, Adjusted R-squared:  0.6892
F-statistic: 101.9 on 2 and 89 DF,  p-value: < 2.2e-16
```

According to this test, sex is not significant and therefore we can conclude that the intercept does not vary significantly between males and females. Note that the p-value is lower this time (0.1771 vs 0.4795), reflecting the higher power of this old-school approach. However, the conclusion remains qualitatively the same: intercepts do not differ.

So we accept the null hypothesis that the intercepts are the same for the two sexes. Running the residual diagnostics, we find no problems with linearity, independence, homogeneity of variances, and normality.

> **Exercise**
>
> You will notice, in the above analysis that the residuals plots flag three data points (cases 19, 49, and 50) as having high residuals. These points are a bit worrisome, and may be having a disproportionate effect on your analysis. Eliminate these "outliers" and re-run the analysis. Now what do you conclude?

```
model.ancova.nooutliers <- lm(lfkl ~ sex + lage, data = anc1dat[c(-49, -50, -19), ])
Anova(model.ancova.nooutliers, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df   F value  Pr(>F)
(Intercept) 1.09160  1 1896.5204 < 2e-16 ***
sex         0.00232  1    4.0374 0.04764 *
lage        0.13992  1  243.0946 < 2e-16 ***
Residuals   0.04950 86
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model.ancova.nooutliers)
```

```
Call:
lm(formula = lfkl ~ sex + lage, data = anc1dat[c(-49, -50, -19),
    ])

Residuals:
      Min        1Q    Median        3Q       Max
-0.058397 -0.018469 -0.000976  0.020696  0.040288

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.224000   0.028106  43.549   <2e-16 ***
sexMALE         -0.010823   0.005386  -2.009   0.0476 *
lage             0.328604   0.021076  15.591   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02399 on 86 degrees of freedom
Multiple R-squared:  0.7706,    Adjusted R-squared:  0.7653
F-statistic: 144.4 on 2 and 86 DF,  p-value: < 2.2e-16
```

Well, well. Now we would, according to convention, reject the null hypothesis, and conclude that in fact, the intercepts of the regressions for the two sexes are different! This is a qualitatively different result from that obtained using all the data. Why? There are two possible reasons:

1. the "outliers" have significant impacts on the fitted regression lines, so that the intercepts of the lines change depending on whether the "outliers" are included (or not);
2. the exclusion of the outliers increases the precision, i.e. reduces the standard error of the intercept estimates, and therefore increases the likelihood that the two intercepts will in fact be "statistically" different.

(1) is unlikely, since none of the outliers had high leverage (hence Cook's distances were not large), so (2) is more likely, and you can verify this by fitting separate regressions for each sex with and without these three outliers. If you do, you will notice that the estimated intercepts for each sex do not change very much, but the standard errors of these intercepts change quite a lot.

Exercise

Fit separate regresssions by sex with vs. without the outliers. Pay attention to the intercepts.

Including all data.

```
by(
  anc1dat,
  anc1dat[, "sex"],
  function(x) {
    summary(lm(lfkl ~ lage, data = x))
  }
)
```

```
anc1dat[, "sex"]: FEMALE

Call:
lm(formula = lfkl ~ lage, data = x)

Residuals:
      Min        1Q    Median        3Q       Max
-0.093728 -0.020510 -0.000618  0.024066  0.078844

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.24264    0.04660  26.664  < 2e-16 ***
lage         0.31431    0.03512   8.949 4.16e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03011 on 52 degrees of freedom
Multiple R-squared:  0.6063,    Adjusted R-squared:  0.5987
F-statistic: 80.09 on 1 and 52 DF,  p-value: 4.16e-12

------------------------------------------------------------
anc1dat[, "sex"]: MALE

Call:
lm(formula = lfkl ~ lage, data = x)

Residuals:
      Min        1Q    Median        3Q       Max
-0.046663 -0.014875 -0.004275  0.013489  0.078910

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.19730    0.04209   28.45  < 2e-16 ***
lage         0.34300    0.03337   10.28 2.97e-12 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02594 on 36 degrees of freedom
Multiple R-squared:  0.7458,    Adjusted R-squared:  0.7388
F-statistic: 105.6 on 1 and 36 DF,  p-value: 2.972e-12
```

Difference in intercept is indeed really small. Now let's have a look when we exclude outliers.

```
by(
  anc1dat,
  anc1dat[, "sex"],
  function(x) {
    summary(lm(lfkl ~ lage, data = x[c(-49, -50, -19), ]))
  }
)
```

```
anc1dat[, "sex"]: FEMALE

Call:
lm(formula = lfkl ~ lage, data = x[c(-49, -50, -19), ])

Residuals:
      Min        1Q    Median        3Q       Max
-0.092746 -0.020176 -0.000078  0.023779  0.079995

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.24029    0.04815  25.760  < 2e-16 ***
lage         0.31533    0.03614   8.724 1.53e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03021 on 49 degrees of freedom
Multiple R-squared:  0.6083,    Adjusted R-squared:  0.6003
F-statistic: 76.11 on 1 and 49 DF,  p-value: 1.526e-11

------------------------------------------------------------
anc1dat[, "sex"]: MALE

Call:
lm(formula = lfkl ~ lage, data = x[c(-49, -50, -19), ])

Residuals:
      Min        1Q    Median        3Q       Max
-0.047429 -0.012818 -0.005274  0.013495  0.077538

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.19361    0.04188   28.50  < 2e-16 ***
lage         0.34662    0.03325   10.42 2.83e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02574 on 35 degrees of freedom
Multiple R-squared:  0.7563,    Adjusted R-squared:  0.7494
F-statistic: 108.6 on 1 and 35 DF,  p-value: 2.835e-12
```

Differences in intercepts are really small and similar than in previous models but now the precision (i.e. standard error) is much smaller for the models without outliers.

> **Note**
>
> It is often the case that by eliminating outliers, new outliers appear. This is simply because the "outlier" designation is usually based on a standardized residual: if you eliminate a couple of outliers, then the residual sums of squares decreases, i.e. the "average" (absolute) residual decreases. Thus, points which were not "far from the average" when the original average residual was comparatively large (i.e. were not "outliers"), may now become so because the average residual has been decreased. Remember also that as you eliminate outliers, N decreases, and the increase in R^2 may be more than compensated for by decreased power. So be wary of eliminating outliers!

8.6. Comparing model fits

As we have just seen, the process of fitting models to data is usually an iterative one. That is, there are, more often than not, several competing models that may be used to fit the data and it is left to the analyst to decide which model best balances goodness of fit (which we are usually trying to maximize) and complexity (which we are usually trying to minimize). In general, the strategy to use in regression and anova is to choose a simpler model when doing so does not reduce the goodness-of-fit by a significant amount. R can compute an F-statistic to compare the fit of two models. The null hypothesis in this situation is that there is no difference in goodness of fit between the models.

> **Exercise**
>
> Working with the `Anc1dat` data set, compare the fit of the ANCOVA and common simple regression models::

```
model.linear <- lm(lfkl ~ lage, data = anc1dat)
anova(model.ancova, model.linear)
```

```
Analysis of Variance Table

Model 1: lfkl ~ sex + lage
Model 2: lfkl ~ lage
  Res.Df      RSS Df  Sum of Sq      F Pr(>F)
```

```
1     89 0.071623
2     90 0.073113 -1 -0.0014899 1.8513 0.1771
```

The `anova()` function can compare the differences in sum of squares and degrees of freedom between the simpler and more complex models, takes the ratio of these two values to generate a mean square, and divides this by the mean square of the more complex model to generate an F-statistic. In the above case, there is insufficient evidence to reject the null hypothesis and we conclude that the simpler model, which is the simple linear regression, is the best model for these data. (Because these models differ by only the presence vs. absence of a single factor (sex), the P-value is the same as the p-value for sex in model 1.)

> **Exercise**
>
> Repeat the above procedure with the ANC3DAT data, rerunning the full ANCOVA with interaction (`lfkl ~ lage + locate + lage:locate`) and without interaction (`lfkl ~ lage + locate`), saving the model objects as you did above. Compare the fits of the two models. What do you conclude?

```
model.full.anc3dat <- lm(lfkl ~ lage + locate + lage:locate, data = anc3dat)
model.ancova.anc3dat <- lm(lfkl ~ lage + locate, data = anc3dat)
anova(model.full.anc3dat, model.ancova.anc3dat)
```

```
Analysis of Variance Table

Model 1: lfkl ~ lage + locate + lage:locate
Model 2: lfkl ~ lage + locate
  Res.Df      RSS Df  Sum of Sq      F    Pr(>F)
1     88 0.051358
2     89 0.060448 -1 -0.0090901 15.575 0.0001592 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case there is sufficient evidence to reject the null hypothesis and conclude that the full model with interaction is the best model to fit to the `Anc3dat` data. This is as we expected, given the fact that we found the interaction to be significant the in original analysis of the data. While no new information is gain from this model comparison in this case, this approach can be more usefully employed to compared nested models that differ in more than one term.

8.7. Bootstrap

```
############################################################
######
# Bootstrap analysis
#
```

```
# Bootstrap analysis BCa confidence intervals
# Preferable when parameter distribution is far from normal
# Bootstrap 95% BCa CI for regression coefficients
library(boot)

# To simplify future modifications of the code in this file,
# copy the data to a generic mydata dataframe
mydata <- anc3dat

# create a myformula variable containing the formula for the model to be fitted
myformula <- as.formula(lfkl ~ lage + locate + lage:locate)

# function to obtain regression coefficients for each iteration
bs <- function(formula, data, indices) {
  d <- data[indices, ]
  fit <- lm(formula, data = d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(data = mydata, statistic = bs, R = 1000, formula = myformula)

# view results
results
boot_res <- summary(results)
rownames(boot_res) <- names(results$t0)
boot_res

op <- par(ask = TRUE)
for (i in 1:length(results$t0)) {
  plot(results, index = i)
  title(names(results$t0)[i])
}
par(op)

# get 95% confidence intervals
for (i in 1:length(results$t0)) {
  cat("\n", names(results$t0)[i], "\n")
  print(boot.ci(results, type = "bca", index = i))
}
```

8.8. Permutation test

```
############################################################
##########
# Permutation test
```

```
#
# using lmperm library
# To simplify future modifications of the code in this file,
# copy the data to a generic mydata dataframe
mydata <- anc3dat
# create a myformula variable containing the formula for the
# model to be fitted
myformula <- as.formula(lfkl ~ lage + locate + lage:locate)
require(lmPerm2)
# Fit desired model on the desired dataframe
mymodel <- lm(myformula, data = mydata)
# Calculate p-values for each term by permutation
# Note that lmp centers numeric variable by default, so to
# get results that are
# consistent with standard models, it is necessary to set
# center=FALSE
mymodelProb <- lmp(myformula,
  data = mydata, center = FALSE,
  perm = "Prob"
)
summary(mymodel)
summary(mymodelProb)
```

References

Bibliography

Xie, Y. 2015. Dynamic documents with R and knitr. 2nd edition. Chapman; Hall/CRC, Boca Raton, Florida.

List of R packages used in this book

```
  [1] "abind"             "ade4"              "ape"
  [4] "arm"               "arrayhelpers"      "ash"
  [7] "AsioHeaders"       "askpass"           "asreml"
 [10] "assertthat"        "audio"             "av"
 [13] "backports"         "badger"            "base"
 [16] "base64enc"         "bayesplot"         "bayestestR"
 [19] "bdsmatrix"         "beepr"             "benchmarkme"
 [22] "benchmarkmeData"   "BH"                "bibtex"
 [25] "bigD"              "BiocManager"       "bit"
 [28] "bit64"             "bitops"            "blob"
 [31] "blogdown"          "bmp"               "bookdown"
 [34] "boot"              "boot"              "brew"
 [37] "bridgesampling"    "brio"              "brms"
 [40] "Brobdingnag"       "broom"             "broom.helpers"
 [43] "broom.mixed"       "bslib"             "ca"
 [46] "cachem"            "callr"             "car"
 [49] "carData"           "caret"             "caTools"
 [52] "cellranger"        "checkmate"         "chromote"
 [55] "chron"             "citr"              "class"
 [58] "class"             "classInt"          "cli"
 [61] "climwin"           "clipr"             "clock"
 [64] "cluster"           "cmdstanr"          "coda"
 [67] "codetools"         "collections"       "colorspace"
 [70] "colourpicker"      "combinat"          "commonmark"
 [73] "compiler"          "config"            "conflicted"
 [76] "conquer"           "corpcor"           "correlation"
 [79] "corrplot"          "covr"              "cowplot"
 [82] "coxme"             "cpp11"             "crayon"
 [85] "credentials"       "crosstalk"         "crul"
 [88] "cubature"          "cubelyr"           "curl"
 [91] "cyclocomp"         "dagitty"           "data.table"
 [94] "datasets"          "datawizard"        "DBI"
 [97] "dbplyr"            "deldir"            "dendextend"
[100] "DEoptimR"          "Deriv"             "desc"
```

```
[103] "deSolve"           "devtools"          "dfidx"
[106] "DHARMa"            "diagram"           "DiagrammeR"
[109] "dials"             "DiceDesign"        "diffobj"
[112] "digest"            "distill"           "distributional"
[115] "dlstats"           "doParallel"        "dotCall64"
[118] "downlit"           "downloader"        "dplyr"
[121] "dsm"               "DT"                "dtplyr"
[124] "dygraphs"          "e1071"             "effects"
[127] "effectsize"        "egg"               "ellipse"
[130] "ellipsis"          "emmeans"           "emo"
[133] "emoji"             "estimability"      "evaluate"
[136] "evd"               "exactRankTests"    "expm"
[139] "factoextra"        "FactoMineR"        "fansi"
[142] "farver"            "fastmap"           "fastmatrix"
[145] "fda"               "fds"               "fields"
[148] "fitdistrplus"      "flashClust"        "flextable"
[151] "fmesher"           "FNN"               "fontawesome"
[154] "fontBitstreamVera" "fontLiberation"    "fontquiver"
[157] "forcats"           "foreach"           "forecast"
[160] "foreign"           "foreign"           "formatR"
[163] "Formula"           "formula.tools"     "fracdiff"
[166] "fs"                "furrr"             "futile.logger"
[169] "futile.options"    "future"            "future.apply"
[172] "gamm4"             "gap"               "gap.datasets"
[175] "gargle"            "gbRd"              "gdata"
[178] "gdtools"           "generics"          "genetics"
[181] "gert"              "getPass"           "gfonts"
[184] "GGally"            "gganimate"         "ggdag"
[187] "ggdist"            "ggeffects"         "ggforce"
[190] "ggfortify"         "ggfun"             "ggimage"
[193] "ggmcmc"            "ggplot2"           "ggplotify"
[196] "ggpubr"            "ggraph"            "ggrepel"
[199] "ggridges"          "ggsci"             "ggsignif"
[202] "ggstats"           "ggtext"            "ggthemes"
[205] "gh"                "git2r"             "gitcreds"
[208] "glmnet"            "globals"           "glue"
[211] "gmodels"           "gnm"               "googledrive"
[214] "googlesheets4"     "gower"             "GPfit"
[217] "gplots"            "graphics"          "graphlayouts"
[220] "grateful"          "grDevices"         "gremlin"
[223] "greta"             "grid"              "gridExtra"
[226] "gridGraphics"      "gridtext"          "gsg"
[229] "gsubfn"            "gt"                "gtable"
[232] "gtools"            "hardhat"           "haven"
[235] "HDInterval"        "hdrcde"            "here"
[238] "hexbin"            "hexSticker"        "highr"
[241] "HKprocess"         "Hmisc"             "hms"
[244] "HSAUR3"            "htmlTable"         "htmltools"
[247] "htmlwidgets"       "httpcode"          "httpgd"
[250] "httpuv"            "httr"              "httr2"
```

```
[253] "icons"            "ids"              "igraph"
[256] "imager"           "infer"            "influenceR"
[259] "ini"              "inlabru"          "inline"
[262] "insight"          "interp"           "invgamma"
[265] "ipred"            "IRdisplay"        "IRkernel"
[268] "isoband"          "iterators"        "itsadug"
[271] "janitor"          "jpeg"             "jquerylib"
[274] "jsonlite"         "juicyjuice"       "kableExtra"
[277] "kernlab"          "KernSmooth"       "KernSmooth"
[280] "kinship2"         "km.ci"            "KMsurv"
[283] "knitr"            "ks"               "labeling"
[286] "labelled"         "lambda.r"         "languageserver"
[289] "later"            "latexdiffr"       "lattice"
[292] "latticeExtra"     "lava"             "lavaan"
[295] "lazyeval"         "leaflet"          "leaflet.providers"
[298] "leaps"            "LearnBayes"       "learnr"
[301] "lhs"              "lifecycle"        "lintr"
[304] "listenv"          "lme4"             "lmerTest"
[307] "lmPerm"           "lmtest"           "locfit"
[310] "loo"              "lsmeans"          "lubridate"
[313] "magick"           "magrittr"         "manipulateWidget"
[316] "maps"             "maptools"         "markdown"
[319] "marmotdata"       "MASS"             "MASS"
[322] "MasterBayes"      "mathjaxr"         "Matrix"
[325] "Matrix"           "MatrixModels"     "matrixStats"
[328] "maxstat"          "mclogit"          "mclust"
[331] "mcmc"             "MCMCglmm"         "MCMCpack"
[334] "memisc"           "memoise"          "memuse"
[337] "methods"          "metR"             "mgcv"
[340] "mime"             "miniUI"           "minqa"
[343] "misc3d"           "mitools"          "mlogit"
[346] "mnormt"           "modelbased"       "modeldata"
[349] "modelenv"         "ModelMetrics"     "modelr"
[352] "mrds"             "multcomp"         "multcompView"
[355] "multicool"        "MuMIn"            "munsell"
[358] "mvtnorm"          "nadiv"            "NBDA"
[361] "nleqslv"          "nlme"             "nlme"
[364] "nloptr"           "nnet"             "nnet"
[367] "numDeriv"         "officer"          "opdisDownsampling"
[370] "openssl"          "openxlsx"         "operator.tools"
[373] "optimx"           "ordinal"          "orthopolynom"
[376] "packrat"          "pagedown"         "palmerpenguins"
[379] "pamm"             "parallel"         "parallelly"
[382] "parameters"       "parsnip"          "partR2"
[385] "patchwork"        "pbapply"          "pbdZMQ"
[388] "pbivnorm"         "pbkrtest"         "pcaPP"
[391] "pdftools"         "pedantics"        "pedSimulate"
[394] "performance"      "permute"          "piecewiseSEM"
[397] "pillar"           "pingr"            "pixmap"
[400] "pkgbuild"         "pkgconfig"        "pkgdown"
```

```
[403] "pkgload"           "plogr"             "plot3D"
[406] "plotfunctions"     "plotly"            "plotrix"
[409] "pls"               "plyr"              "png"
[412] "polyclip"          "polynom"           "posterior"
[415] "ppcor"             "pracma"            "praise"
[418] "prettyunits"       "pROC"              "processx"
[421] "prodlim"           "profvis"           "progress"
[424] "progressr"         "projpred"          "promises"
[427] "proto"             "proxy"             "ps"
[430] "pspearman"         "purrr"             "pwr"
[433] "qgam"              "QGglmm"            "qpdf"
[436] "quadprog"          "quantmod"          "quantreg"
[439] "quarto"            "questionr"         "QuickJSR"
[442] "qvcalc"            "R.cache"           "R.methodsS3"
[445] "R.oo"              "R.utils"           "R6"
[448] "ragg"              "rainbow"           "rappdirs"
[451] "rARPACK"           "raster"            "rbibutils"
[454] "rcmdcheck"         "RColorBrewer"      "Rcpp"
[457] "RcppArmadillo"     "RcppEigen"         "RcppParallel"
[460] "RcppRoll"          "RcppTOML"          "RCurl"
[463] "Rdpack"            "reactable"         "reactR"
[466] "readbitmap"        "readODS"           "readr"
[469] "readxl"            "recipes"           "RefManageR"
[472] "registry"          "relimp"            "rematch"
[475] "rematch2"          "remotes"           "renderthis"
[478] "renv"              "report"            "repr"
[481] "reprex"            "reshape"           "reshape2"
[484] "reticulate"        "revealjs"          "rex"
[487] "rgdal"             "rgeos"             "rgl"
[490] "rio"               "rJava"             "rlang"
[493] "rmarkdown"         "rngtools"          "ROAuth"
[496] "robustbase"        "ROI"               "roxygen2"
[499] "rpart"             "rprojroot"         "rptR"
[502] "rsample"           "rsconnect"         "rSDP"
[505] "Rsolnp"            "RSpectra"          "RSQLite"
[508] "rstan"             "rstantools"        "rstatix"
[511] "rstudioapi"        "rvcheck"           "rversions"
[514] "rvest"             "s2"                "sandwich"
[517] "sass"              "scales"            "scatterplot3d"
[520] "scholar"           "scico"             "see"
[523] "selectr"           "servr"             "sessioninfo"
[526] "settings"          "sf"                "sfsmisc"
[529] "shape"             "shiny"             "shinyjs"
[532] "shinyMatrix"       "shinystan"         "shinythemes"
[535] "showtext"          "showtextdb"        "simpleboot"
[538] "sjlabelled"        "sjmisc"            "sjPlot"
[541] "sjstats"           "slam"              "slider"
[544] "slidex"            "sn"                "snakecase"
[547] "snow"              "sourcetools"       "sp"
[550] "spam"              "SparseM"           "spatial"
```

```
[553] "spatial"           "SpatialPack"       "splancs"
[556] "splines"           "splines2"          "splitstackshape"
[559] "sqldf"             "SQUAREM"           "squid"
[562] "StanHeaders"       "statmod"           "stats"
[565] "stats4"            "stringi"           "stringr"
[568] "styler"            "survey"            "survival"
[571] "survival"          "survminer"         "survMisc"
[574] "svglite"           "svUnit"            "sys"
[577] "sysfonts"          "systemfonts"       "tcltk"
[580] "tensorA"           "tensorflow"        "terra"
[583] "testthat"          "textshaping"       "tfautograph"
[586] "tfruns"            "TH.data"           "threejs"
[589] "tibble"            "tidybayes"         "tidygraph"
[592] "tidymodels"        "tidymv"            "tidyr"
[595] "tidyselect"        "tidyterra"         "tidyverse"
[598] "tiff"              "timechange"        "timeDate"
[601] "tinytex"           "tmvnsim"           "tools"
[604] "triebeard"         "truncnorm"         "tseries"
[607] "tsfeatures"        "TTR"               "tune"
[610] "tweenr"            "twosamples"        "tzdb"
[613] "ucminf"            "units"             "urca"
[616] "urlchecker"        "urltools"          "usethis"
[619] "utf8"              "utils"             "uuid"
[622] "V8"                "vcd"               "vcdExtra"
[625] "vctrs"             "vegan"             "viridis"
[628] "viridisLite"       "visNetwork"        "visreg"
[631] "vroom"             "waldo"             "warp"
[634] "webshot"           "webshot2"          "websocket"
[637] "whisker"           "whoami"            "withr"
[640] "wk"                "workflows"         "workflowsets"
[643] "writexl"           "xaringan"          "xaringanExtra"
[646] "xaringanthemer"    "xfun"              "xgboost"
[649] "xlsx"              "xlsxjars"          "xml2"
[652] "xmlparsedata"      "xopen"             "xtable"
[655] "xts"               "yaml"              "yardstick"
[658] "ybamaRmot"         "yesno"             "yulab.utils"
[661] "zip"               "zoo"
```

A. Software Tools

For those who are not familiar with software packages required for using R Markdown, we give a brief introduction to the installation and maintenance of these packages.

A.1. R and R packages

R can be downloaded and installed from any CRAN (the Comprehensive R Archive Network) mirrors, e.g., https://cran.rstudio.com. Please note that there will be a few new releases of R every year, and you may want to upgrade R occasionally.

To install the **bookdown** package, you can type this in R:

```
install.packages("bookdown")
```

This installs all required R packages. You can also choose to install all optional packages as well, if you do not care too much about whether these packages will actually be used to compile your book (such as **htmlwidgets**):

```
install.packages("bookdown", dependencies = TRUE)
```

If you want to test the development version of **bookdown** on GitHub, you need to install **devtools** first:

```
if (!requireNamespace("devtools")) install.packages("devtools")
devtools::install_github("rstudio/bookdown")
```

R packages are also often constantly updated on CRAN or GitHub, so you may want to update them once in a while:

```
update.packages(ask = FALSE)
```

Although it is not required, the RStudio IDE can make a lot of things much easier when you work on R-related projects. The RStudio IDE can be downloaded from https://www.rstudio.com.

A.2. Pandoc

An R Markdown document (`*.Rmd`) is first compiled to Markdown (`*.md`) through the **knitr** package, and then Markdown is compiled to other output formats (such as LaTeX or HTML) through Pandoc. This process is automated by the **rmarkdown** package. You do not need to install **knitr** or **rmarkdown** separately, because they are the required packages of **bookdown** and will be automatically installed when you install **bookdown**. However, Pandoc is not an R package, so it will not be automatically installed when you install **bookdown**. You can follow the installation instructions on the Pandoc homepage (http://pandoc.org) to install Pandoc, but if you use the RStudio IDE, you do not really need to install Pandoc separately, because RStudio includes a copy of Pandoc. The Pandoc version number can be obtained via:

```
rmarkdown::pandoc_version()
## [1] '3.1.11.1'
```

If you find this version too low and there are Pandoc features only in a later version, you can install the later version of Pandoc, and **rmarkdown** will call the newer version instead of its built-in version.

A.3. LaTeX

LaTeX is required only if you want to convert your book to PDF. You may see https://www.latex-project.org/get/ for more information about LaTeX and its installation, but we strongly recommend that you install the lightweight and cross-platform LaTeX distribution named TinyTeX and based on TeX Live. TinyTeX can be easily installed through the R package **tinytex** (which should be automatically installed when you install **bookdown**):

```
tinytex::install_tinytex()
```

With TinyTeX, you should never see error messages like this:

```
! LaTeX Error: File `titling.sty' not found.

Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: sty)

Enter file name:
! Emergency stop.
<read *>

l.107 ^^M

pandoc: Error producing PDF
Error: pandoc document conversion failed with error 43
Execution halted
```

The above error means you used a package that contains `titling.sty`, but it was not installed. LaTeX package names are often the same as the `*.sty` filenames, so in this case, you can try to install the `titling` package. If you use TinyTeX with R Markdown, missing LaTeX packages will be installed automatically, so you never need to worry about such problems.

LaTeX distributions and packages are also updated from time to time, and you may consider updating them especially when you run into LaTeX problems. You can find out the version of your LaTeX distribution by:

```
system("pdflatex --version")
## pdfTeX 3.141592653-2.6-1.40.22 (TeX Live 2022/dev/Debian)
## kpathsea version 6.3.4/dev
## Copyright 2021 Han The Thanh (pdfTeX) et al.
## There is NO warranty.  Redistribution of this software is
## covered by the terms of both the pdfTeX copyright and
## the Lesser GNU General Public License.
## For more information about these matters, see the file
## named COPYING and the pdfTeX source.
## Primary author of pdfTeX: Han The Thanh (pdfTeX) et al.
## Compiled with libpng 1.6.37; using libpng 1.6.37
## Compiled with zlib 1.2.11; using zlib 1.2.11
## Compiled with xpdf version 4.03
```

To update TinyTeX, you may run:

```
tinytex::tlmgr_update()
```

From year to year, you may need to upgrade TinyTeX, too (otherwise you cannot install or update any LaTeX packages), in which case you may reinstall TinyTeX:

```
tinytex::reinstall_tinytex()
```

Index

LaTeX, 192

Pandoc, 192