BIO4158 Applied biostats with R

Laboratory manual

Julien Martin

21-09-2021

Table des Matières

Note

Development version. Lab material will appear slowly during the Fall 2021 term.

Preface

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.

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"))
```

G*Power

G*Power est un programme gratuit, développé par des psychologues de l'Université de Dusseldorf en Allemagne. Le programme existe en version Mac et Windows. Il peut cependant être utilisé sous linux via Wine. G*Power vous permettra d'effectuer une analyse de puissance pour la majorité des tests que nous verrons au cours de la session sans avoir à effectuer des calculs complexes ou farfouiller dans des tableaux ou des figures décrivant des distributions ou des courbes de puissance.

Téléchargez le programme sur le site https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower.html

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,

warnings,

warnings,

important points

notes

and tips

Resources {-}

This document was developped using the excellent bookdown de Yihui Xie. The manual is based on the previous lab manual *Findlay, Morin and Rundle, BIO4158 Laboratory manual for BIO4158.*

License

Figure 1: License Creative Commons

Chapitre 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

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 define as the based 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/BI`
   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()`

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.

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")
```

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()`:

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" "params"
```

```
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.

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)
```

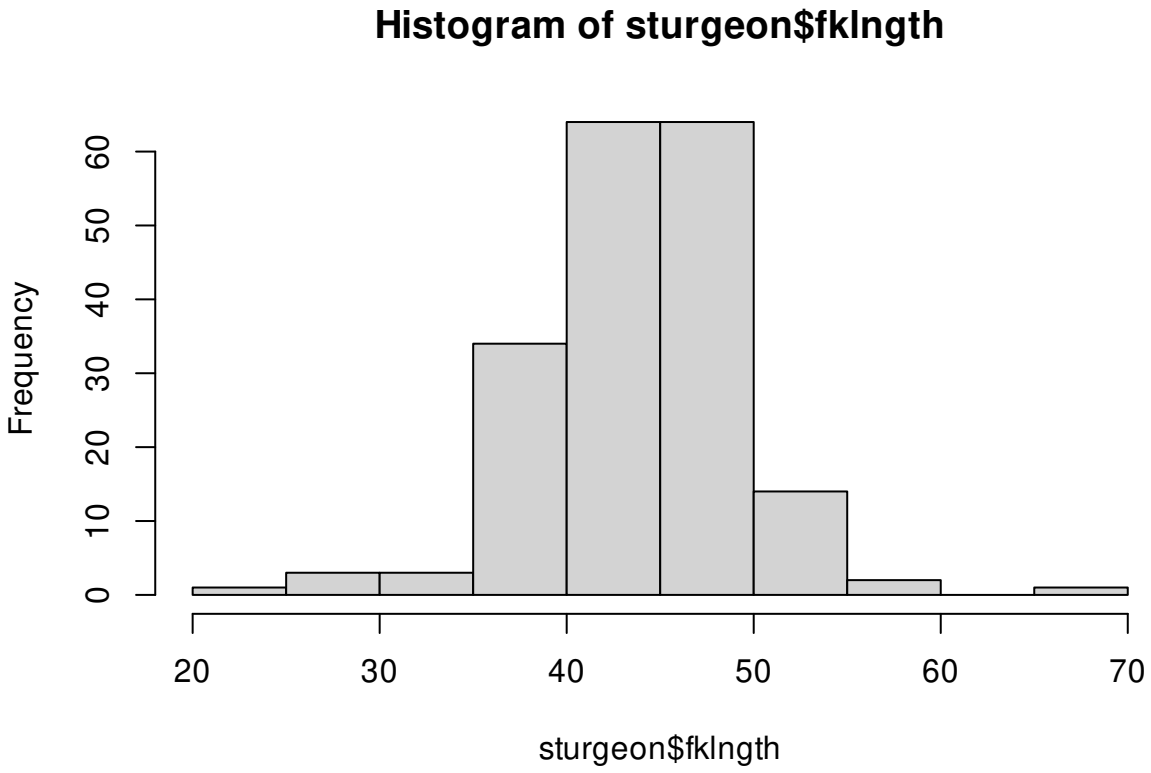


Figure 1.1: Histogram of fluke length of sturgeons

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

Note 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. 1.1) 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
```

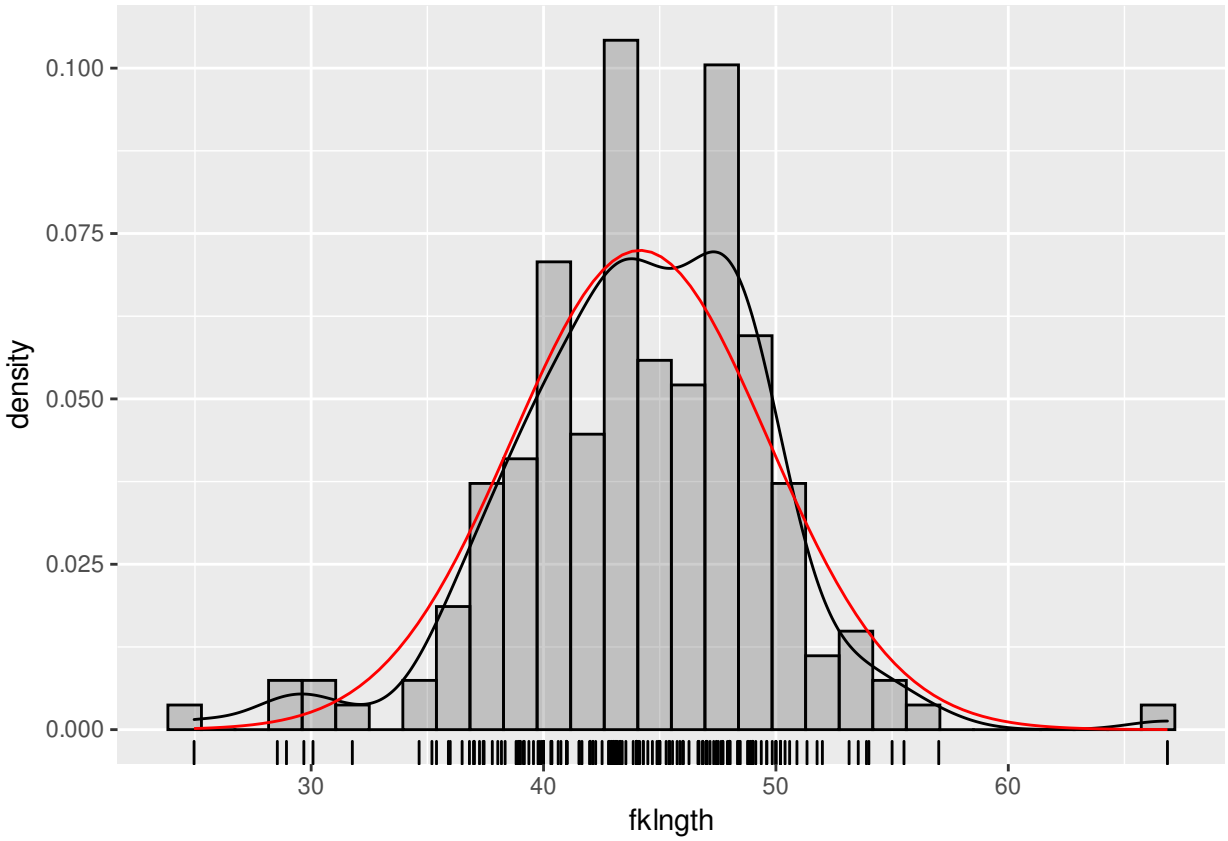
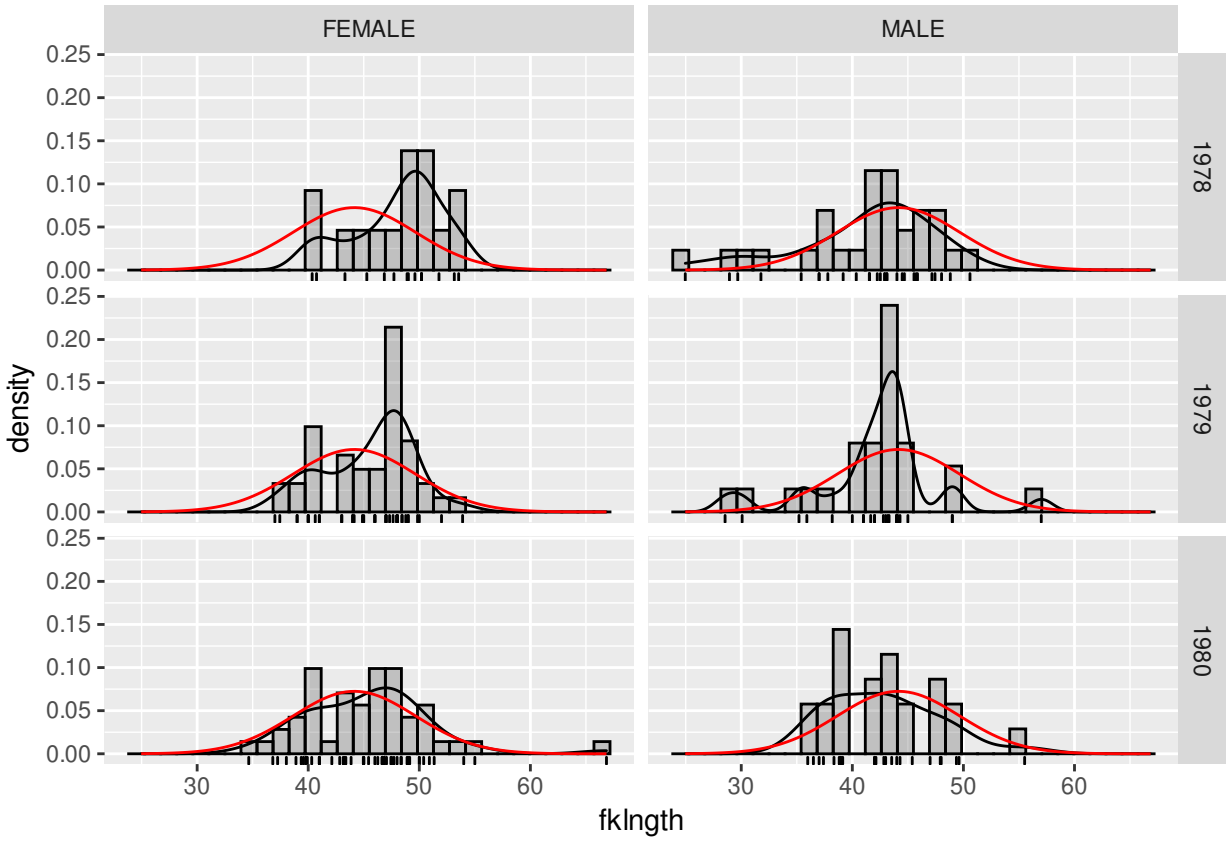


Figure 1.2: Distribution of fluke length in sturgeon plotted with ggplot

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:

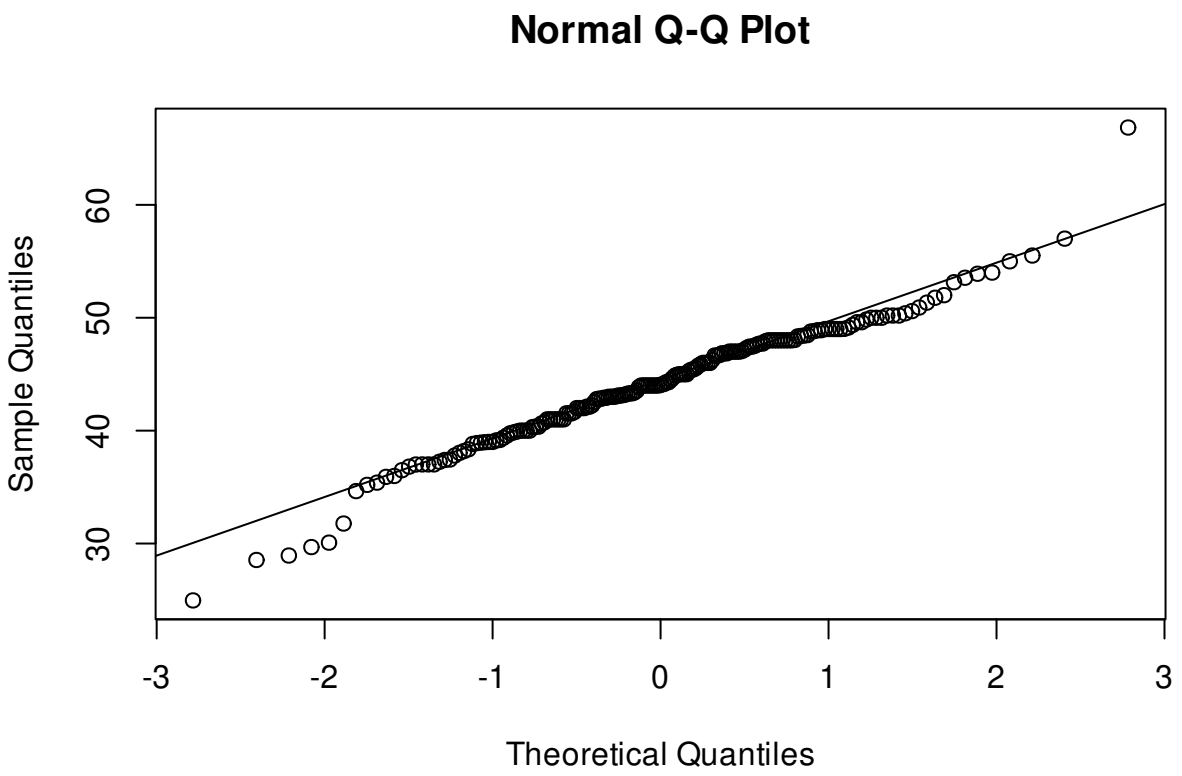
```
mygraph + facet_grid(year ~ sex)
```



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)
```

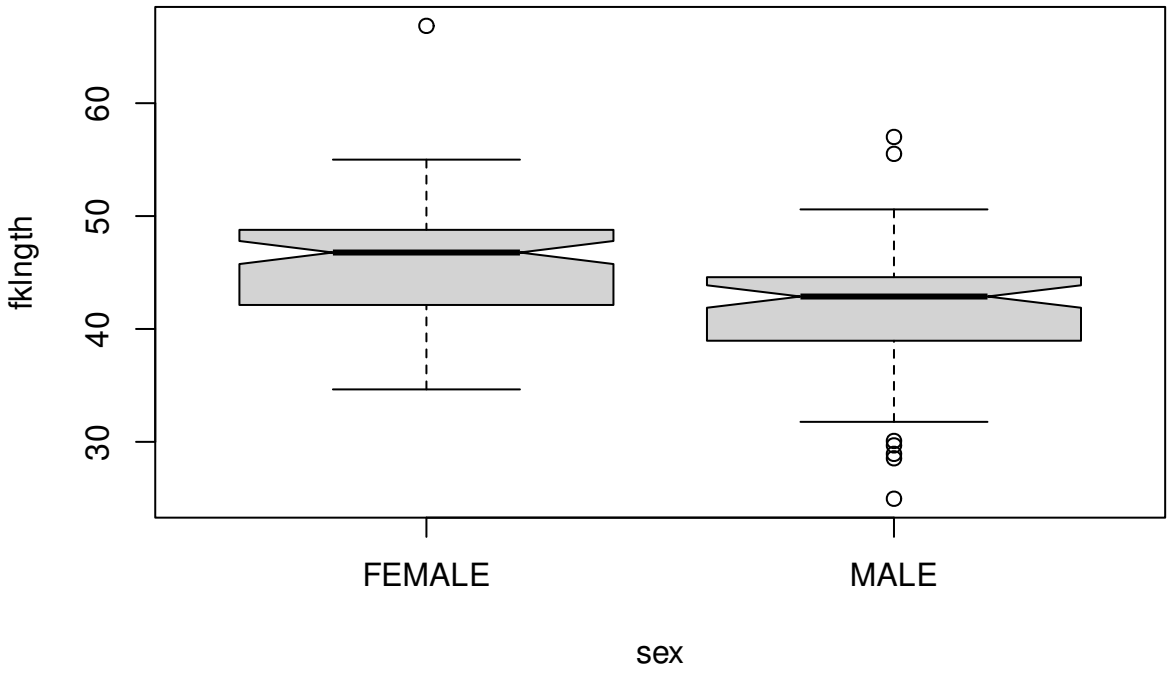


Figure 1.3: Boxplot of fluke length in strugeon by sex

The slightly thicker line inside the box of figure 1.3 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.

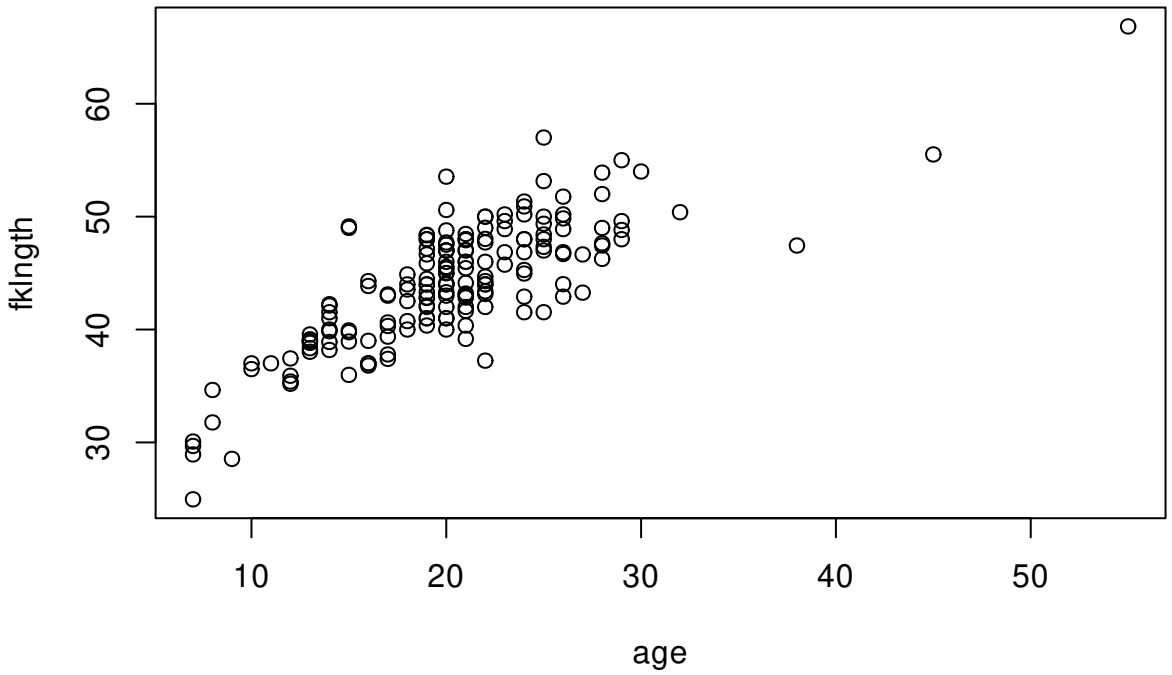
1.3.3 Scatterplots

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).

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

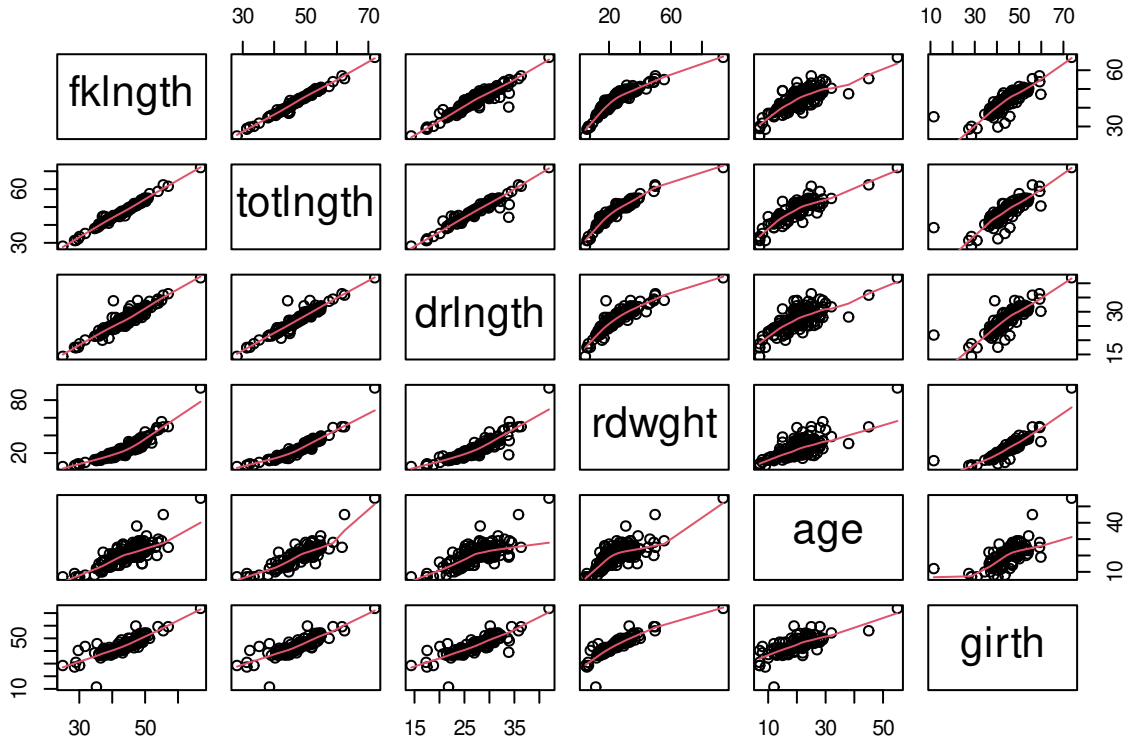
You should obtain:

```
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
```

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.

Operateur	Explication	Operateur	Explication
==	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		

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

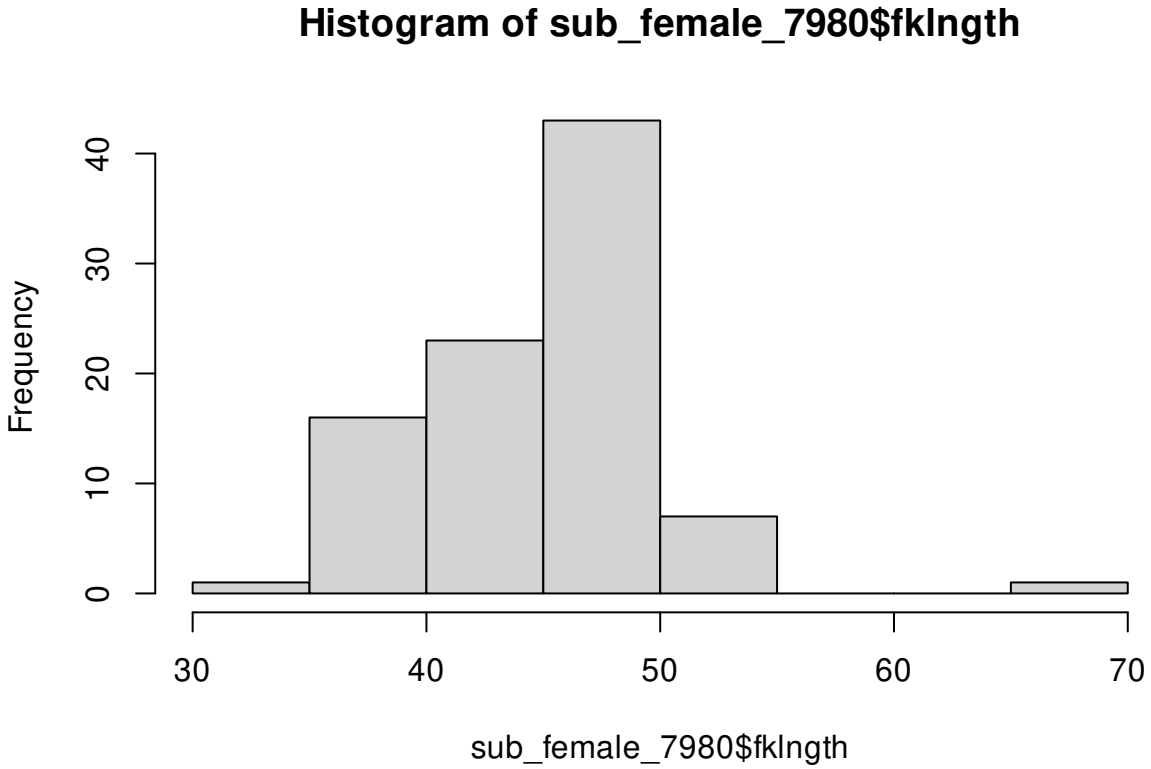


Figure 1.4: Distibution of fluke length of female sturgeons in 1979 and 1980

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.

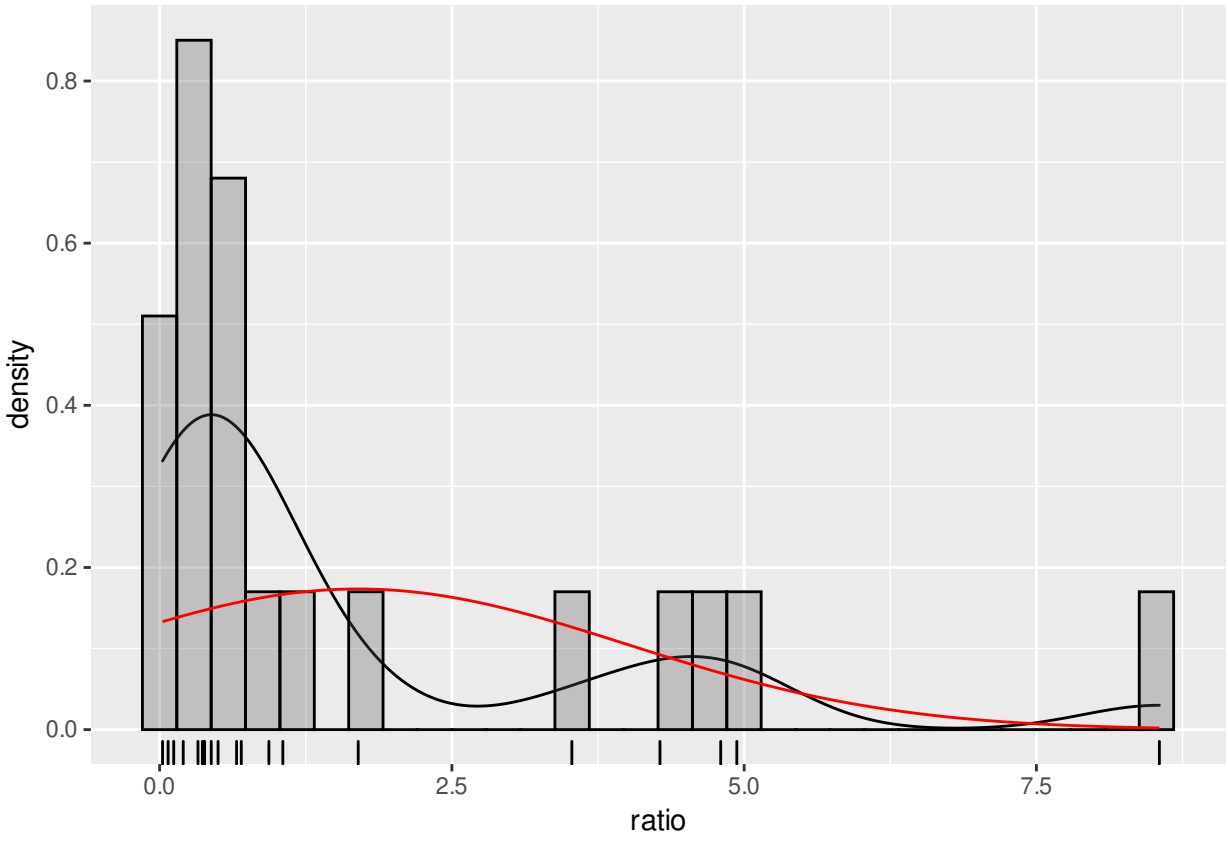


Figure 1.5: Distibution of infection ratios by the wild (SAUVAGE) strain of salmonella

Chapitre 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 sizze

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.

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	f	$f = \frac{\mu_1}{\sigma}$
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

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:

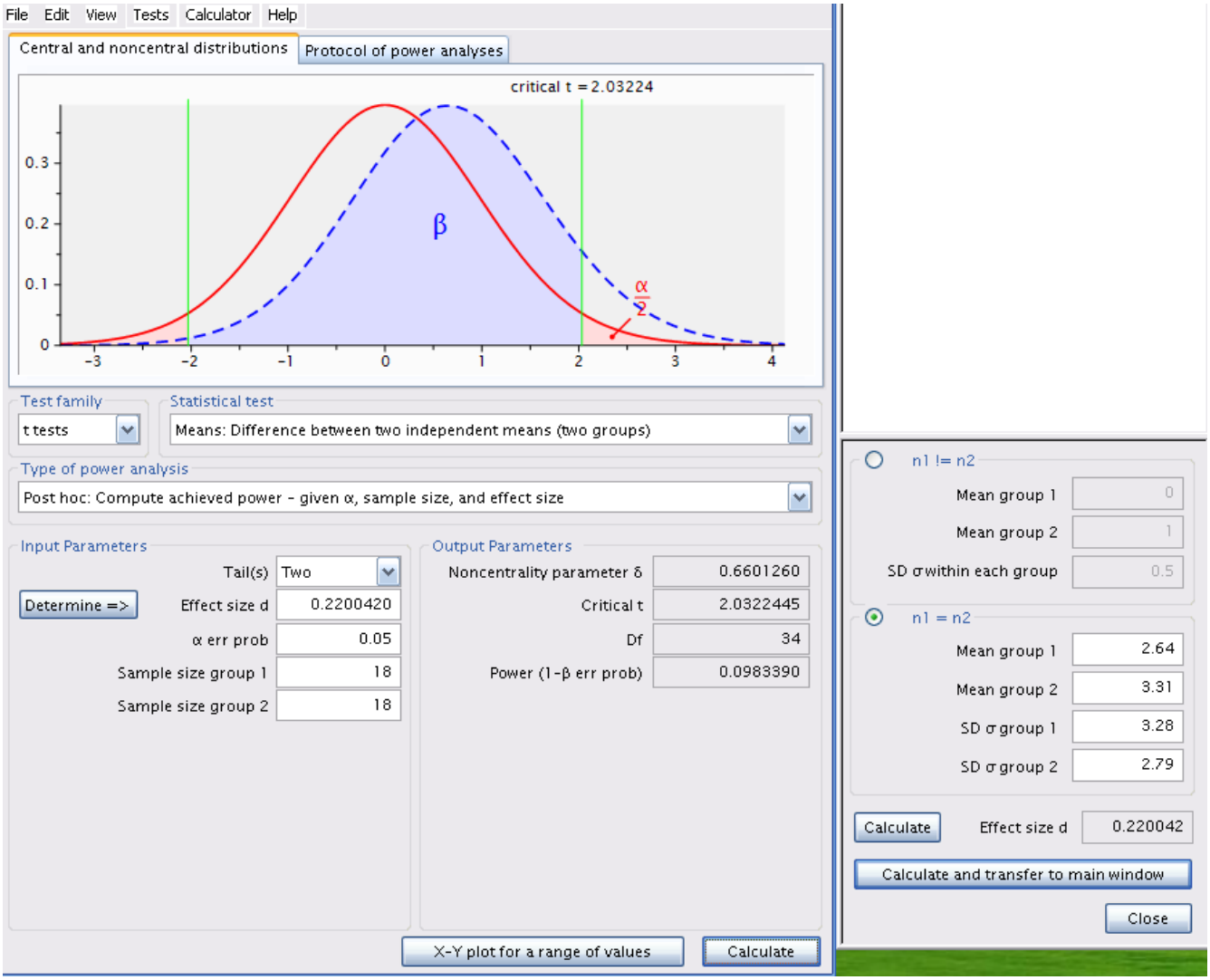


Figure 2.1: Post-hoc analysis with estimated effect size

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,
```

```
##
##      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
```

```r
# 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,
```

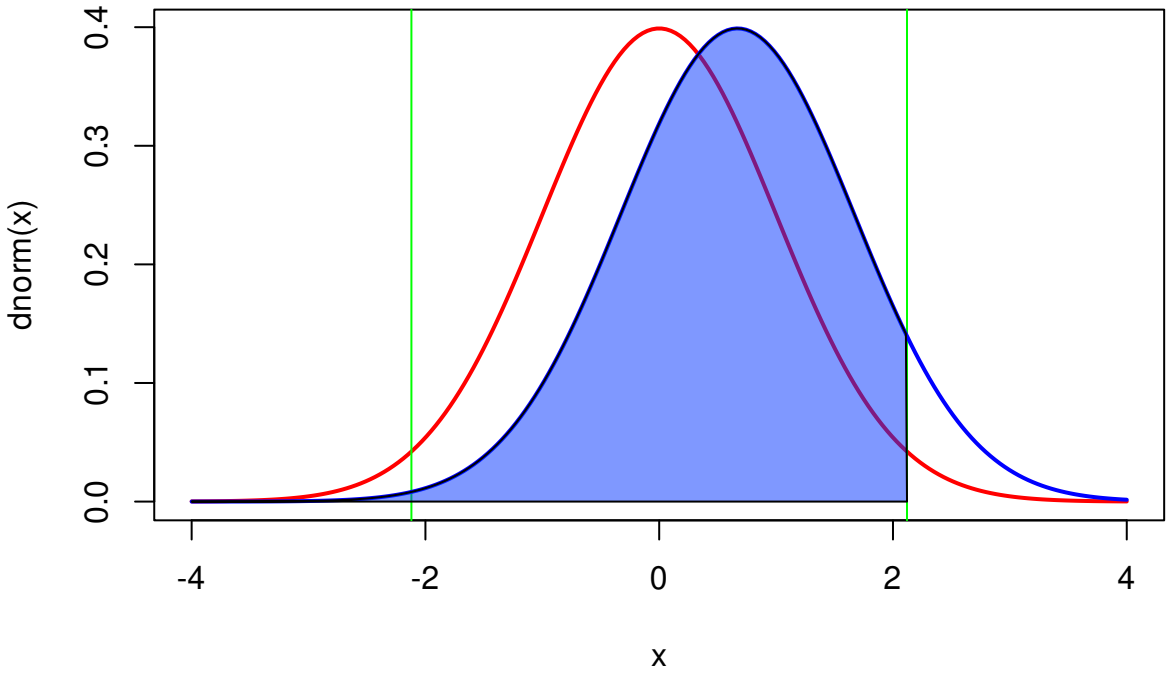


Figure 2.2: Post-hoc analysis with estimated effect size in R

Let's examine the figure 2.1.

- The curve on the left, in red, corresponds to the expected distri- bution 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 dis- tribution 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.

Change the mean of group 2 to 5.28, recalculate effect size, and click on Calculate to obtain figure 2.3.

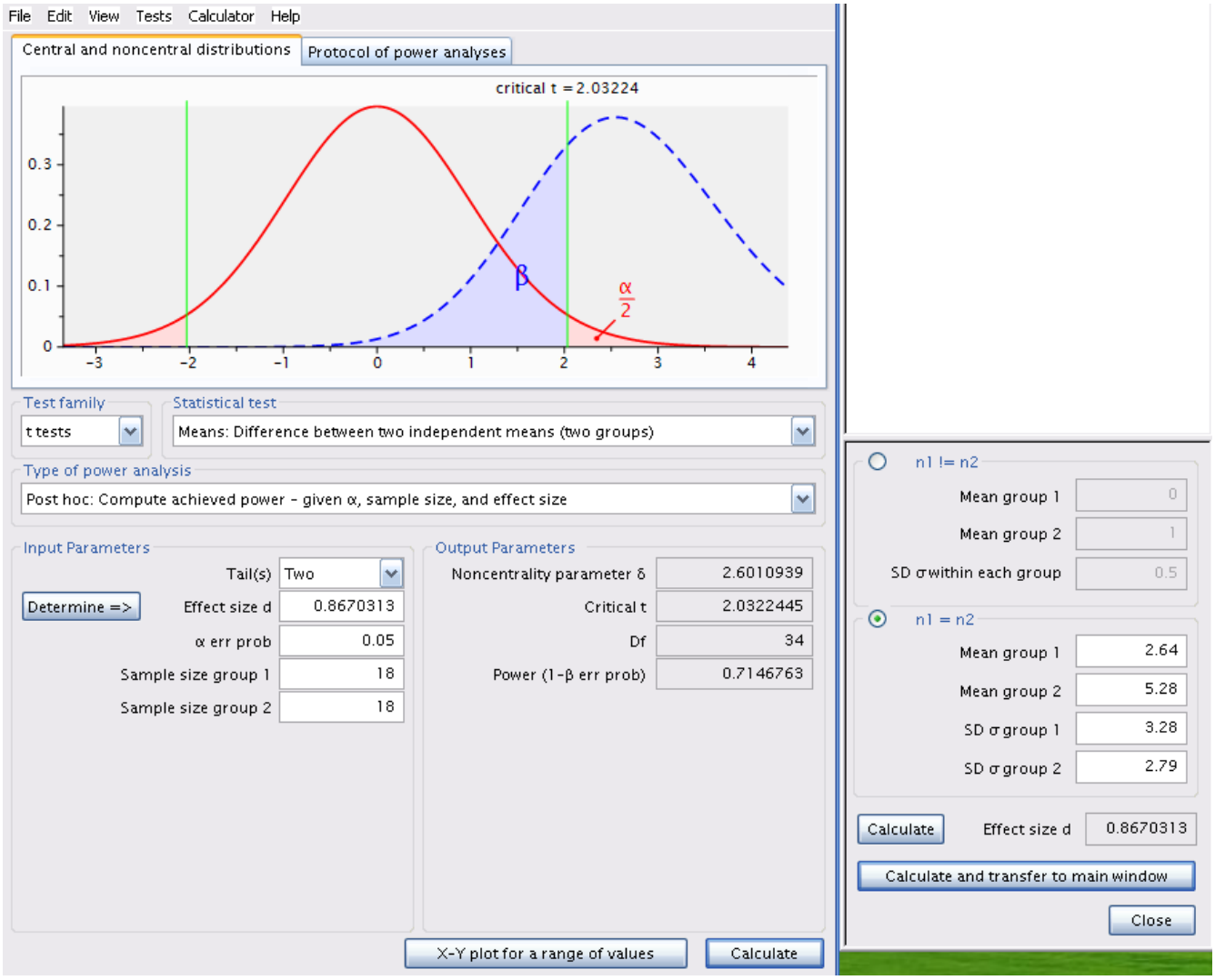


Figure 2.3: Post-hoc analysis using an effect size different from the one estimated

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,
```

```
##
##      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)?

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 (2.4.

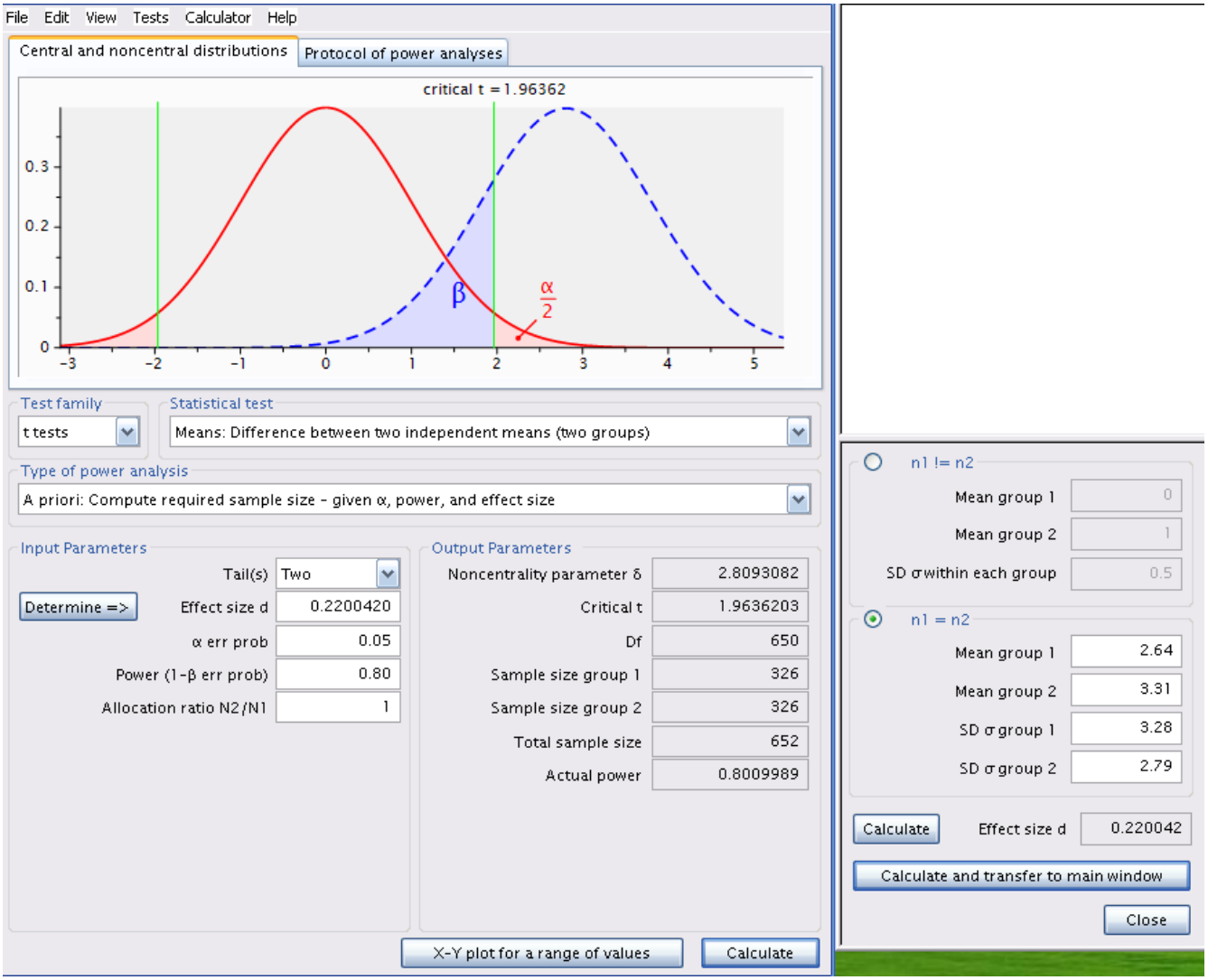


Figure 2.4: A priori analysis

```
pwr.t.test(power = 0.8, d = d(u1 = 2.64, sd1 = 3.28, u2 = 3.31, sd2 = 2.79), sig.level = 0
```

```
##
##      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 =
```

```
##
##      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 $\beta = 0.2$)?

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.

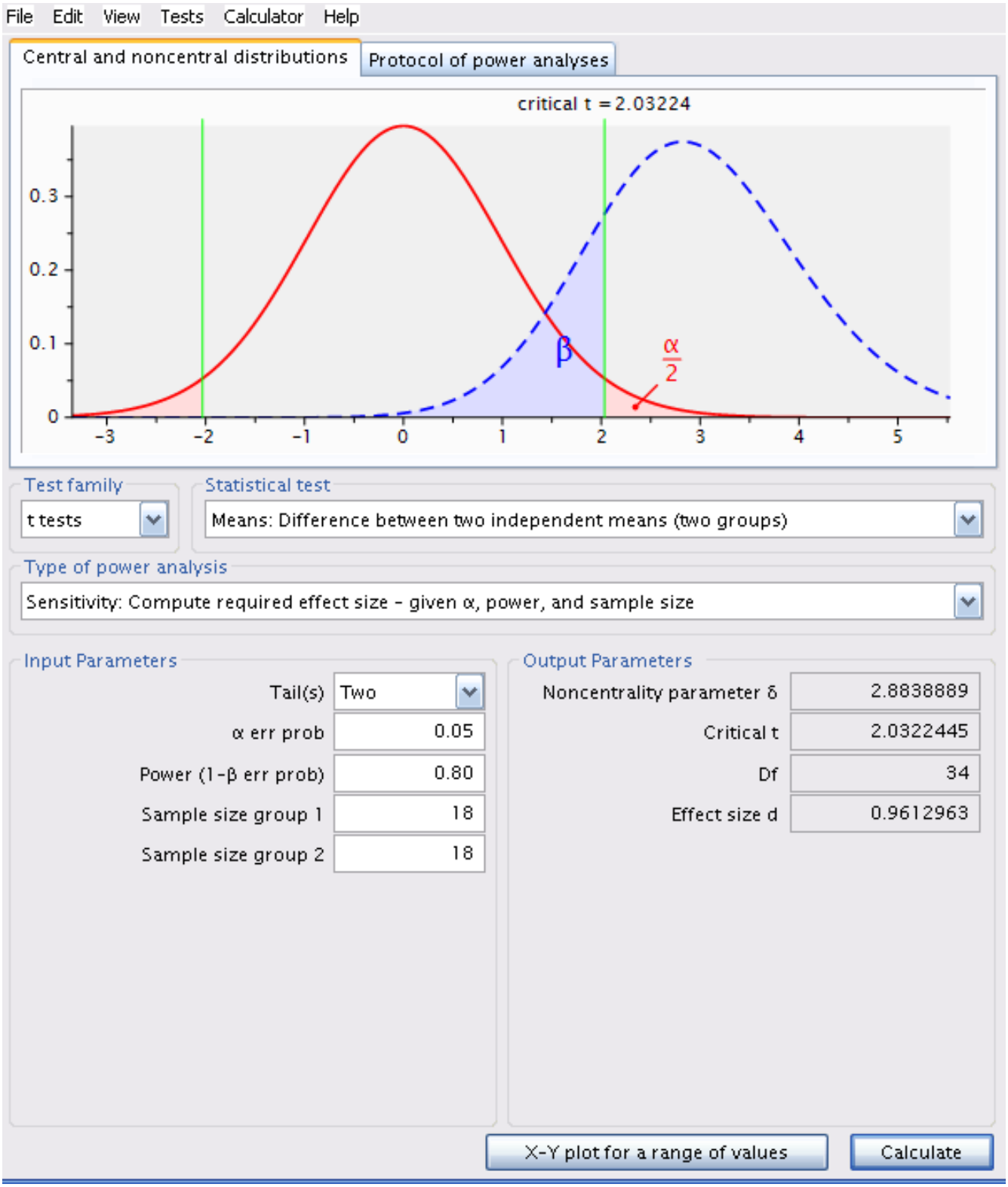


Figure 2.5: Analyse de sensitivité

```
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.

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 +
```

```
## [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.

Appendix 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] '2.9.2.1'
```

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 3.14159265-2.6-1.40.21 (TeX Live 2020/Debian)
## kpathsea version 6.3.2
## Copyright 2020 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 4.02
```

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