Statistics with R: a hands-on approach

Myriam Luce

September 18, 2018

Contents

Chapter 1

Data review

1.1 Possibly interesting extra tidbits

In the making of this tutorial, I used several tools that you might like to access as well. Being the tedium-averse programmer I am, I use a reference manager program, in my case Zotero. You can find the full bibliography for this project, including a few entries that did not make it into the references section here because I did not cite them, at the Zotero project page.

I also want to point out awesome-public-datasets, where I foraged for the examples in this tutorial. It has several interesting datasets in the public health domain.

1.2 Variables

When "doing science", you will be taking measurements, usually in the hopes of understanding a phenomena often in the shape of a relationship between things you are measuring. When working with this data, the nature of the things you measure (*variables*) will influence the presentation and analysis that are appropriate.

A *qualitative variable* refers to categories, or a variable recorded with words, as opposed to a *quantitative variable*, which is measured with numbers. Examples of qualitative variables would include US state, restaurant chains and college major. Quantitative variables could be time to execute a task, waist circumference, disease rate or spending amounts.

R refers to *qualitative variables* as *factors*. *Qualitative variables* can further be divided into non ordinal and *ordinal variables*, depending on whether there is a natural order among the categories. For example, dog breed (chihuahua, husky, labrador) is a non ordinal variable, whereas level of satisfaction (dissatisfied, neutral, satisfied) is ordinal.

Quantitative variables are either *discrete variables*, where measurements are done in integers, or *continuous variables*, where they come in real numbers

(you could get an infinity of decimals with a theoretical instrument of infinite precision). *Discrete variables* could be number of children, cancer deaths, or wedding age. *Continuous variables* include temperature, blood sugar level, and weight.

Furthermore, when studying variables in relationship with one another, changes in a *dependent variable* are driven or explained by an *independent variable*. Typically, this means the "x" axis of a graph will be the *independent variable*, while the "y" axis will be the *dependent variable*.

1.3 Gotchas

1.3.1 Tidy data

When working with data in R, analysis will be easier if your data is *tidy*, that is, each column in your data set contains one and only one variable.

For example, in a cancer dataset that we will use later, the original data is presented as in table 1.1. Here, we have four variables: cancer type, sex, number of cases, and number of deaths. While the first column is one and only one variable, the other columns mix sex with number of cases or sex with number of deaths. If you would like to analyze deaths by sex or cases per cancer type, some data manipulation will be necessary to combine the relevant columns.

Cancer Type	Cases		Deaths	
	Male	Female	Male	Female
...	...	...	...	...

Table 1.1: Cancer data set format.

If you keep your data tidy, R can usually do the combining for you, if you know how to ask nicely. As such, it is recommended that the first thing you do after successfully importing data into R is to verify it is tidy. Tools to divide or merge columns will be discussed in section 2.1.

1.3.2 Correlation is not causation, or of the importance of DoE

DoE refers to design of experiments. The common trope that "correlation is not causation" refers to the fact that because two variables vary together does not necessarily mean that one causes the other to change; for instance, they might both be responding to a common cause, when it's not just plain old coincidence. My personal favorite exemplification of "correlation is not causation" is that the divorce rate in Maine correlates with the per capita consumption of margarine [11].

To distinguish between correlation and causation with certainty, a controlled experiment must be run. Depending on the phenomena studied, this might

mean using a control group, a placebo, or directly controlling environmental conditions. For example, to determine the effect of low oxygen concentration in water on cod growth, several tanks can be set up where individual cods are randomly distributed and where oxygen levels are controlled. If you were to simply measure oxygen levels in water and cod growth at different locations and subsequently find a correlation between the two, you couldn't tell for certain whether the difference is due to oxygen levels, or another factor like water temperature, or even the fact that cod compete with one another and that the runts, who would grow slower anyway, end up pushed into less desirable low-oxygen environments. Of course, sometimes running a controlled experiment is not feasible for practical reasons (one can't control amount of natural sunlight, for example) or ethical reasons (having an untreated control group of people with a serious condition, when a potentially life-saving treatment might exist, is questionable).

When dealing with humans, to determine whether medication has a positive effect on an health issue, the health issue can be measured for a group who took the medication (treatment group), a group who took a placebo, and a group who took no medication at all (control group). In humans, particular effort must be placed on controlling or measuring the placebo effect, for the test subjects as well as the professionals. A recent spectacular example is the recommendation to abandon arthroscopic surgeries for knee pain because it did not show better results than physical therapy in randomized trials, despite it being the most common orthopaedic procedure in several countries [9].

Where experimental design is concerned, key factors are randomization, blocking, and replication. If terms like Completely Randomized Design, Latin Squares or Factorial Design are not familiar, I would recommend investing some time into learning the basics of experimental design before embarking in an experiment, in the interest of avoiding some common and easily remedied mistakes ([3] appears to be a well-rounded textbook).

Chapter 2

Descriptive statistics

2.1 Data: import, export, tidy

2.1.1 R plumbing: packages

R [7] should be relatively straightforward to install: download and execute, follow the wizard instructions.

Where things get a bit more complicated is when it comes to packages. While the basic R program has a lot of functions built-in, there will come a time when you will need something that is not offered out of the box. Thankfully, R has a very dynamic community with a ton of packages. For instance, a very popular package to produce figures is ggplot2. Let's install it to see how packages are managed in R.

First off, to install packages in R, you will need to launch it as an administrator. If you don't, you will get the rather unhelpful message shown in figure 2.1. To launch with administrator rights, right-click on your R launcher and find the option "Run as administrator". How to do so from the Windows 10 start menu is shown in figure 2.2. (As a note, you should launch as administrator *only* when installing packages, as opposed to modifying your shortcut to always launch as administrator.)

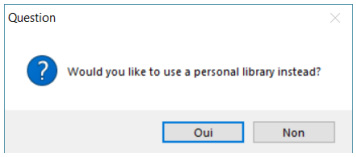


Figure 2.1: Error message displayed by R if trying to install packages without administrator rights.

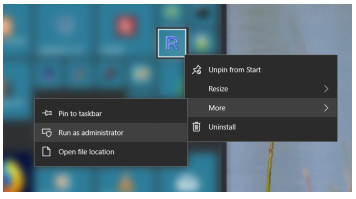
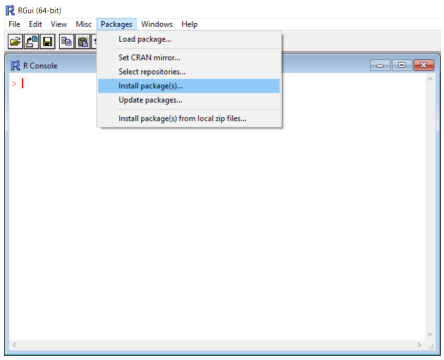


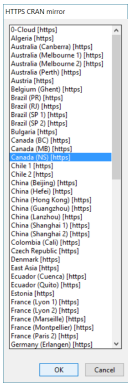
Figure 2.2: How to launch R with administrative rights.

Now that R is launched as administrator, you can install ggplot2 using either

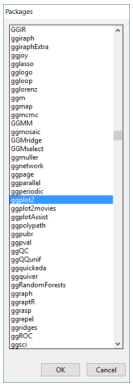
the convenient Packages menu or the command line if you're that hardcore. Personally I use the menu; after choosing a mirror (different mirrors offer different packages; Canada NS has a wide selection and is vaguely geographically close), you can select your desired package and hit "install", as shown in figure 2.3. Then you only need to wait until R is done doing its thing. If all the lines say "successfully unpacked", all good; otherwise, an error has occurred and you will have to decipher the message to figure out how to remedy the situation. (If you run into any trouble, I would first recommend updating to the latest release of R.)



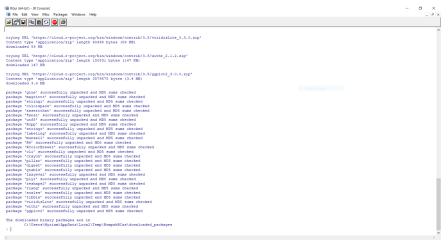
(a) Packages menu in R Gui.



(b) List of mirrors.



(c) List of packages.

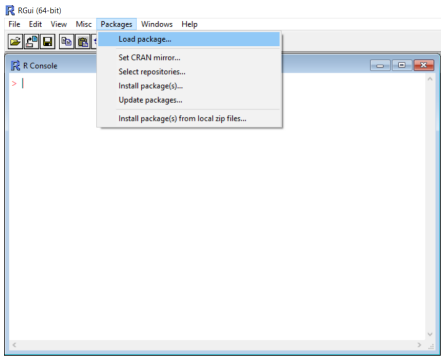


(d) All successfully unpacked: successful installation.

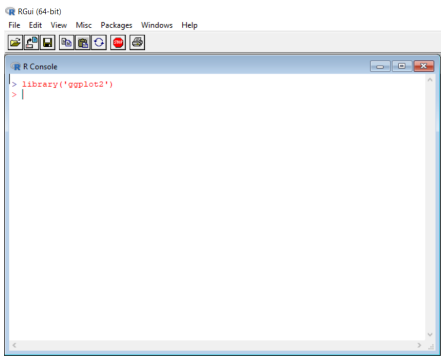
Figure 2.3: Package installation process.

Note that installing a package is not enough to use it; you must also load it. Again this can be done either from the menu or with the command line, as shown in figure 2.4. This operation must be repeated *every time* you restart R.

Now let's say that `ggplot2` becomes your favoritest package in the whole



(a) Packages menu in R Gui.



(b) Command to load a package in R Gui.

Figure 2.4: Loading packages in R.

world and you end up using it every day. After a week or so you will probably be very annoyed having to reload the package every time you open R. You can add packages to be loaded automatically in either of two files: `Rprofile.site` or `.Rprofile`. `Rprofile.site` is located in your R installation folder in the etc folder and is always executed. `.Rprofile` is located in the user home folder and is applied after the changes made by `Rprofile.site`. As the appropriate location to put `.Rprofile` seems to change from R version to version, I will showcase the `Rprofile.site` here. At the bottom of the file, add the following:

```
.First <- function(){
        library('ggplot2')
        # other libraries go here
}
```

2.1.2 Importing data: `read.csv` is your friend

A typical workflow to get data from wherever into R would be as follows:

1. Copy-paste the data into your favorite spreadsheet software (Microsoft Excel, LibreOffice Calc, Google Sheets, etc.).

2. If necessary, transpose your data so that variables are in columns (rather than rows).

3. Tweak column names so they have no spaces and no special characters (é, $, etc).

4. Assign a reasonable format (text, number, thousand separators, etc.) to all columns.

5. With your operating systems using an English locale, save as csv.

6. Use `read.csv` in R with the appropriate options.

Let's cover each of these with a few examples

2.1.3 Export

2.1.4 Tidy

2.1.5 R plumbing: saving aka scripts

As long as you're doing simple things fitting on two or three lines, you probably won't feel the need to "save your file". However, as you start doing more elaborate data treatment or

2.2 Tables and figures

2.2.1 Frequency table (1D) or contingency table (2D)

If you feel the need to make a table with your data, use a spreadsheet software (Microsoft Excel, LibreOffice Calc, Google Sheets). ;) R is superior in statistics and (arguably) in figures, but spreadsheets definitely have their uses when it comes to tables.

2.2.2 Pie chart

A pie chart is a graph that can be used to visually represent proportions of a *discrete variable*[1]. Note that they have their critics, who recommend never using them, as our brain is bad at comparing the size of slices [5].

As an example data set, let's use ebola deaths by country [4]. An excerpt giving the source data is shown in figure 2.5. Enter the data in your favorite spreadsheet software and save it as a csv (thankfully for you English speakers, there is no need to fiddle with decimal symbol (is it a dot or a comma?) and whether the data is really comma-separated). You should get the following:

```
Country,Deaths
Guinea,2543
Liberia,4809
Sierra Leone,3956
Mali,6
Nigeria,8
United States of America,1
```

R offers various data import options. The most useful I have found were `read.csv`[2] to import csv data and `read.fwf` to import fixed-width data. To

[1] Words in italics are defined in the glossary.

[2] Words in monospace font refer to R commands. The cheat sheet at the end of the tutorial lists most of those used in this document.

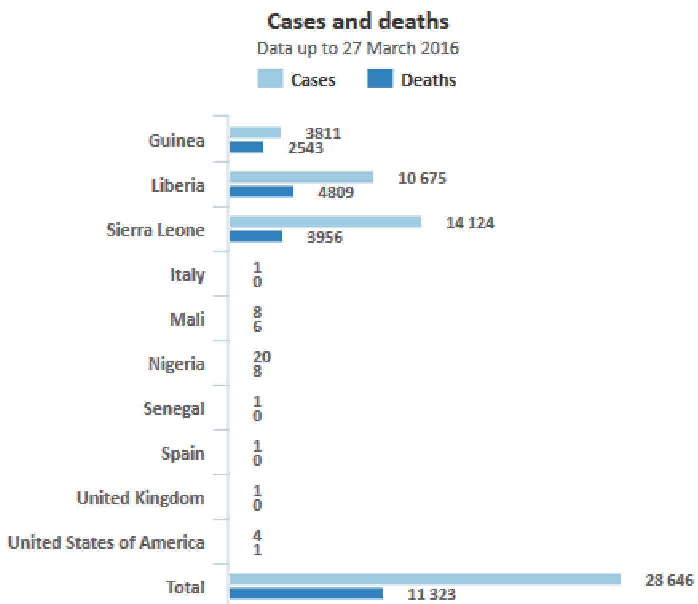


Figure 2.5: Excerpt from [4].

demonstrate, figure 2.6 shows what csv (delimited) and fixed-width data look like side by side.

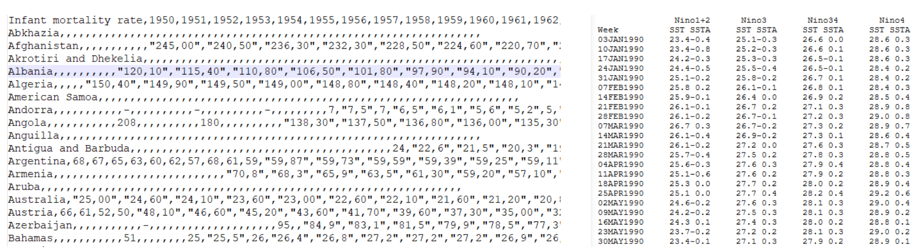


Figure 2.6: Delimited data (left) and fixed-width data (right).

Go ahead and load your small csv into R with `read.csv('C:/......data.csv', header=TRUE)`. To avoid messing with default working folder in R settings, I recommend always using the full absolute file path (i.e. starting with C:). Note that you should use the *forward slash "/"* as a path separator, even on Windows. The second parameter, `header=TRUE`, tells R that the first line in your file corresponds to the column headers, not actual data. You can then use the function `pie(counts, labels)` to produce a pie chart. However, as shown below, a naive approach might displease.

```
> ebola = read.csv('D:/megha/Documents/r-tutorial/ebola.csv', header=TRUE)
```

```
> ebola
                  Country Deaths
1                  Guinea   2543
2                 Liberia   4809
3            Sierra Leone   3956
4                    Mali      6
5                 Nigeria      8
6 United States of America      1
> pie(ebola)
Error in pie(ebola) : 'x' values must be positive.
```

You might be scratching your head and wondering which part of 2543 or 6 is not positive, and you'd be justified to do so. Here, one must dive into computer programming concerns to understand what is going on. The "not positive" message hints at a problem with the format of the input data. Let's demonstrate:

```
> values = c(2543, 4809, 3956, 6, 8, 1
> labels = c('Guinea', 'Liberia', 'Sierra Leone', 'Mali',
↪  'Nigeria', 'United States of America')
> pie(values, labels)        # works! produces figure 1.3
> typeof(values)
[1] "double"
> typeof(ebola)
[1] "list"
> class(values)
[1] "numeric"
> class(ebola)
[1] "data.frame"
> class(ebola$Country)
[1] "factor"
> class(ebola$Deaths)
[1] "integer"
> pie(ebola$Deaths, ebola$Country)        # works too now!
```

Technically, `read.csv` returns a `data.frame`, while `pie` only accepts numbers. Let's take a painful tangent into R types that will hopefully help you later on.

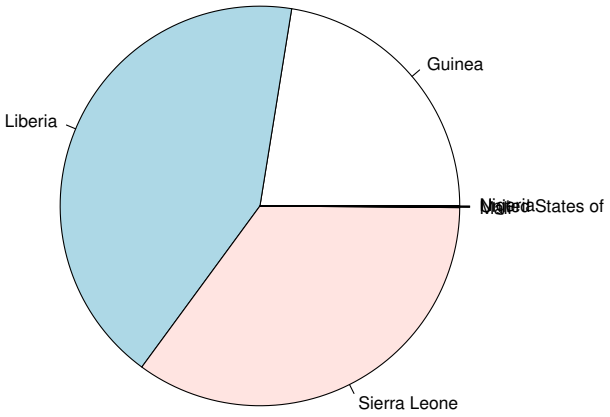


Figure 2.7: Ebola deaths in 2015-2016 by country.

R types

Logical:	TRUE or FALSE
Numeric:	real, by the math definition (ex. 12.3). Double is a numeric with better precision.
Integer:	integer, by the math definition (ex. 12).
Character:	text of any length
Factor:	a type that represents a *discrete variable*
Ordered:	a type that represents an *ordinal variable*
List:	a 1D collection of "things" (may be strings, numbers, or a mix of them)
Vector:	a 1D collection of things of *one type*
Matrix:	a 2D collection of things of *one type*
Array:	a nD collection of things of *one type*
Data Frame:	a (mostly) 2D collection of things, where each column can be of a different type

For future reference, Quick R gives an excellent introduction on the subject [6]. You can convert a variable to anything reasonable (R will turn "2" into an integer, but not "abc") using the host of `as.xyz` functions.

Accessing collection elements

Let's say you have a data frame, for example your ebola deaths by country data:

```
> ebola
                   Country Deaths
1                   Guinea   2543
2                  Liberia   4809
3             Sierra Leone   3956
4                     Mali      6
5                  Nigeria      8
6 United States of America      1
```

Notice how the line with "Country" and "Deaths" is not numbered in the output? It means R is aware it's a header and not data. Data frame columns can be accessed by their name using the `$` operator, like so:

```
> ebola$Country
[1] Guinea                    Liberia
↪  Sierra Leone
[4] Mali                      Nigeria
↪  United States of America
6 Levels: Guinea Liberia Mali Nigeria ... United States of
↪  America
```

If you want to access lines, you can use the `[row, column]` operator, like so:

```
> ebola[1,]
  Country Deaths
1  Guinea   2543
> ebola[1,2]
[1] 2543
```

While the `$` operator is exclusive to data frames, the `[]` is used for all collections. Vectors, lists, matrices and arrays can be accessed with the `[index]` operator for 1D structure, `[row, column]` operator for 2D structures, and `[i, j, k...]` for nD structures.
In the case you want to access several items at once, you can use a colon inside the brackets, i.e. `[begin:end]` like so:

```
> ebola[1:3,]
       Country Deaths
1       Guinea   2543
2      Liberia   4809
3 Sierra Leone   3956
```

Now that we have our basic pie chart, you might be thinking, "That squiggle on the right with the tiny pie slices is quite unseemly". In addition, you might want to tweak other aspects of the graph, like adding a title or choosing colors. We will discuss common graph properties in a following section, to keep it all in the same place. Let's just deal with the pie-chart specific problem of small slices here (I reiterate, you should run away, run away into the arms of a bar chart.), and add a percent annotation, as that is a common occurence. R does not offer an option to deal with small slices out of the box (probably because it tells you in its own manual to use bar charts instead), so let's just manually tweak the labels:

```
> labels = as.character(ebola$Country)
> labels[4]='Others'
> labels[5:6]=''
> labels
[1] "Guinea"       "Liberia"      "Sierra Leone" "Others"
↪  ""
[6] ""
> percents = ebola\$Deaths/sum(ebola\$Deaths)*100
> percents
[1] 22.458712355 42.471076570 34.937737349  0.052989490
↪  0.070652654
[6]  0.008831582
> percents[4] = sum(percents[4:6])
> percents
[1] 22.458712355 42.471076570 34.937737349  0.132473726
↪  0.070652654
[6]  0.008831582
> percents = round(percents, 2)
> percents
[1] 22.46 42.47 34.94  0.13  0.07  0.01
> labels[1:4] = paste(labels[1:4], percents[1:4], '\%')
> labels
[1] "Guinea 22.46 %"       "Liberia 42.47 %"      "Sierra Leone
↪  34.94 %"
[4] "Others 0.13 %"
> pie(ebola\$Deaths, labels)
```

Hacky, but it works, and no more time should be dedicated to pie charts, so let's move on.

2.2.3 Bar chart

A bar chart, sometimes called a line graph, is used to represent a *discrete variable*, and the bars *do not touch*. As an example, data on infant mortality by country can be found at Gapminder [2]. If you simply `read.csv` the file you just downloaded and converted to csv using your favorite Spreadsheet software,

and then examine the structure of your data inside R, you might notice the somewhat strange following occurence:

```
> infant = read.csv('D:/megha/Documents/r-tutorial/infant.csv',
↪  header=TRUE)
> class(infant)
[1] "data.frame"
> str(infant)
'data.frame':   260 obs. of  217 variables:
 $ Infant.mortality.rate: Factor w/ 260 levels
 ↪  "Abkhazia","Afghanistan",..: 1 2 3 5 6 7 8 9 10 11 ...
 $ X1800                : int  NA NA NA NA NA NA NA NA NA NA ...
 $ X1801                : int  NA NA NA NA NA NA NA NA NA NA ...
# ...
 $ X1861                : int  NA NA NA NA NA NA NA NA NA NA ...
 $ X1862                : Factor w/ 11 levels
 ↪  "","."","110","131",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ X1863                : Factor w/ 13 levels
 ↪  "","."","106","113",..: 1 1 1 1 1 1 1 1 1 1 ...
# ...
```

Wait, what? The country is of type factor, we all agree on that, but infant mortality rate for 1862 is a factor? Using good ol' Notepad or a better text editor, open your csv and scroll around, you should spot the problem illustrated in figure 2.8.

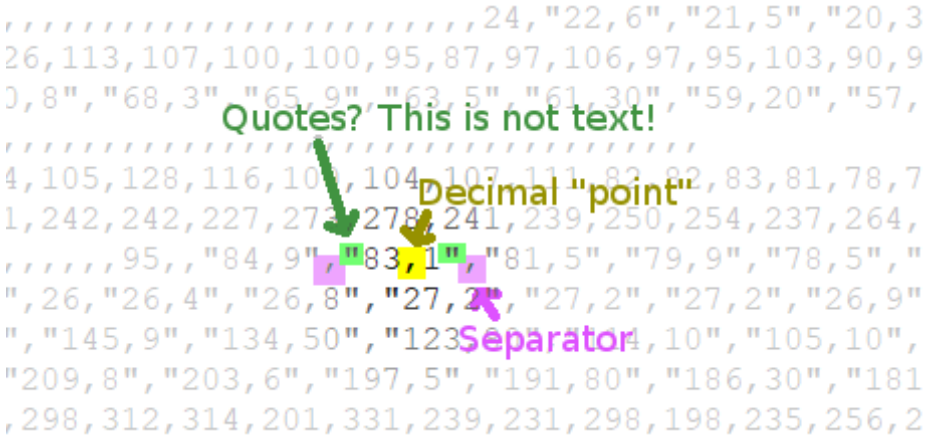


Figure 2.8: Welcome to the real world. CSV sucks internationally.

In other languages, like French for instance, the decimal "point" is a comma, and therefore the "comma-separated" part of comma-separated value leads to some issues. Other gems in this data include "-" for I assume missing data, and some lonely dots, assuming again, for missing data. Gods forbid the authors

hit a snag while importing data and their software didn't warn them something foul was afoot and they just pasted it in the global csv without noticing. (This is why science should be in databases. Real databases. They don't let you put a dash in a number field, they just don't.)

Thankfully, `read.csv` comes with some handy options. You can see them all in the manual by typing `?read.csv` in a R console. So, we need to tell R that our `dec` point is a comma, and dash and dot stand for missing data, i.e. `na.strings`. Technically, R also thinks that things surrounded by quotes are text, but is smart enough to automatically check if the "text" looks like a number and, if it does, it converts it automatically to a number. You therefore get the appropriate following:

```
> infant = read.csv('D:/megha/Documents/r-tutorial/infant.csv',
↪  header=TRUE, dec=",", na.strings=c('-', '.'))
> str(infant)
'data.frame':   260 obs. of  217 variables:
 $ Infant.mortality.rate: Factor w/ 260 levels
 ↪  "Abkhazia","Afghanistan",..: 1 2 3 5 6 7 8 9 10 11 ...
 $ X1800                : int  NA NA NA NA NA NA NA NA NA NA ...
 $ X1801                : int  NA NA NA NA NA NA NA NA NA NA ...
 # ...
```

A barplot is relatively straightforward to produce with R, but we will see all "common" (imho) plot options here, so tie your winter hat down with wire, you'll be sitting here a while. Let's start by simply plotting infant mortality rate by country. To keep the plot readable, let's choose a subset of G8 countries: Canada, France, Germany, Italy, Japan, Russia, United Kingdom and United States of America. Let's also start by studying the mortality rate in 2000. First, we will select each of the countries by its row number, then we will stitch the G8 back together with a function called `rbind`, which binds data frames together by row, as long as all data frames have the same columns.

```
> canada = infant[38,]
> france = infant[77,]
> germany = infant[83,]
> italy = infant[109,]
> japan = infant[111,]
> russia = infant[186,]
> uk = infant[240,]
> usa = infant[241,]
> g8 = rbind(canada, france, germany, italy, japan, russia, uk,
↪  usa)
```

Producing a barplot now is easy:

```
> barplot(g8$X2000, names.arg=g8$Infant.mortality.rate)
```

Several things are wrong with this graph. Glaringly, a bar should not extend beyond its axis. Axes are set as plot options with `xlim` and `ylim`. Also, should

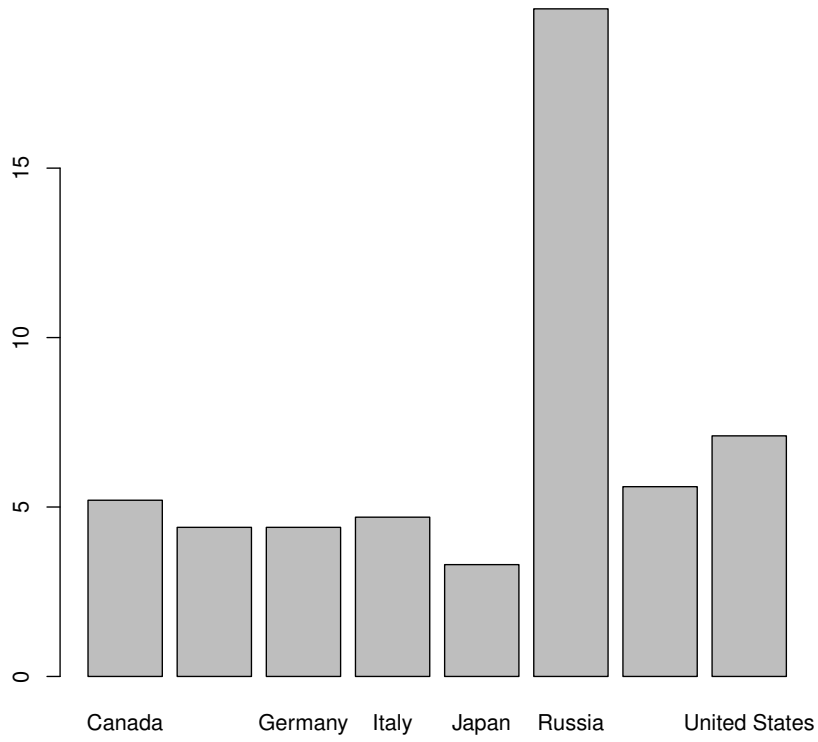


Figure 2.9: Simplest bar plot: infant mortality rate per country.

you want a box around the graph, `bty` takes care of that. Usually. Bar plots are special and you need to all an extra function after your plot appears. See all graph options with `?par`, which we will use a lot more as we customize our graphs.

```
barplot(g8$X2000, names.arg=g8$Infant.mortality.rate,
↪  ylim=c(0,20), bty='o')        # why oh why won't bty work
↪  like everywhere else!
box()
```

You probably also want all country names to show up. Easiest way to do that is to tilt the axis label text. Enter `par`, used *before* your graph function to set general settings. For this next iteration, let's do a few things at once. First, let's make all labels perpendicular to their axis with `par` and `las`. Let's also

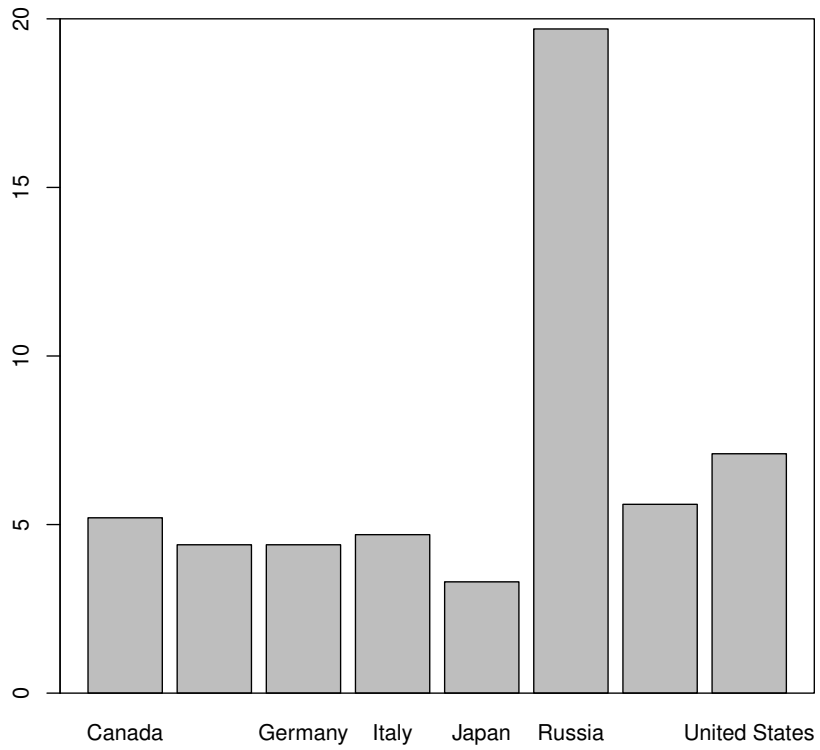


Figure 2.10: Simple bar plot: infant mortality rate per country, axis set.

demonstrate color manipulation by making each country's bar the dominant color on their flag (this might lead to arbitrary choices) with `col`.

```
> colors = c('red', 'blue', 'black', 'green', 'white', 'snow',
↪  'purple', 'purple4')
> par(las=2)                  # axis labels: perpendicular
> barplot(g8$X2000, names.arg=g8$Infant.mortality.rate,
↪  ylim=c(0,20), col=colors)
> box()
```

Colors in R

Colors in R can be specified by their names, if they are among R's list of approved colors, which you can see by calling `colors()`.

A more visually helpful version can be found at Color Chart [1] which, incidentally, has other fascinating references about the use of color in science (good vs. bad color ramps, color blindness considerations).

Additionally, colors can be specified in other formats like `#RRGGBB` where $00 \leq \text{color} \leq \text{FF}$. These values can be found with graphics software or off a color generator on the internet.

Finally, if color space is a factor, additional functions exist: `rgb`, `hsv`, `hcl`, `gray` and `rainbow`.

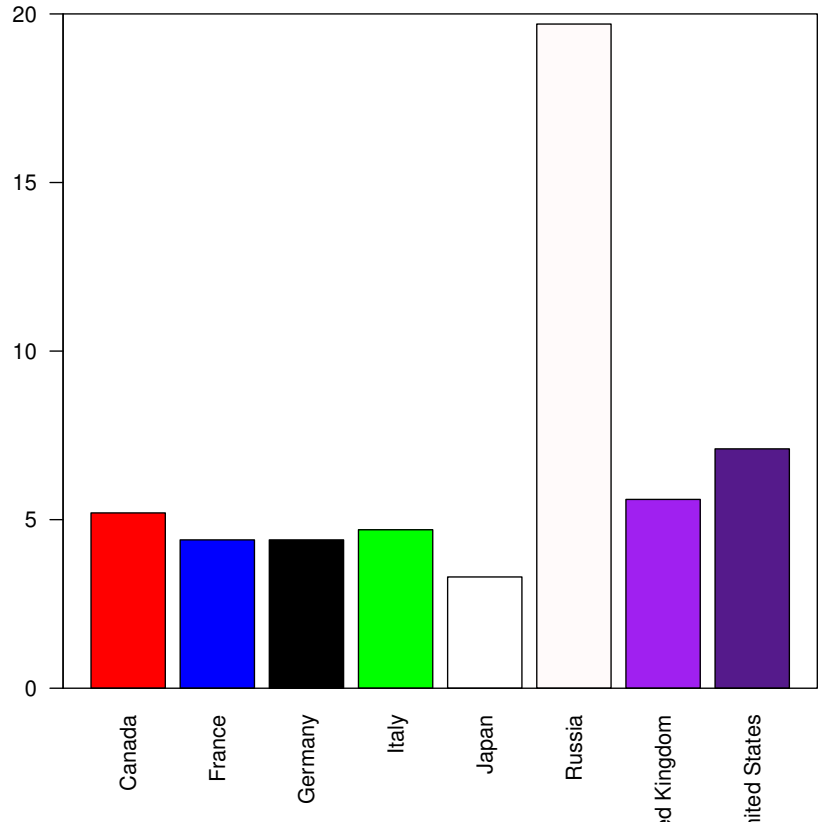


Figure 2.11: Psychedelic bar plot: infant mortality rate per country, axis set, labels perpendicular, colors.

With the country names printed at the vertical, they are running out of space at the bottom of the graph. More margin is needed there. Figures have two types of margins in R: outer and inner. The inner margin is used to draw the figure title and the axis ticks and labels and can be set in inches with `mai=c(bottom, left, top, right)` or in lines with `mar=c(bottom, left, top, right)`. The outer margin is outside the figure; it makes more sense when several plots are displayed together, as we will do a few exercises down the line. The outer margin as well can be set in inches with `omi=c(bottom, left, top, right)` or in lines with `oma=c(bottom, left, top, right)`. As for the appropriate margin necessary to display the full country name, that's a matter of trial and error. Starting with the current parameters' value of inner margin, I found that a value of 8 worked well.

```
par()$mar
[1] 5.1 4.1 4.1 2.1
> par(mar=c(8, 4.1, 4.1, 2.1))
> barplot(g8$X2000, names.arg=g8$Infant.mortality.rate,
↪  ylim=c(0,20), col=colors)
> box()
```

Now, bar charts often use stacked bars. For example, let's use cancer rates [10]. This data includes number of cases and number of deaths by sex and cancer type. If the thing you would most like to compare is the number of cancer by type, you would stack the sexes into one bar, and make one bar for new cases and one bar for deaths. Data can most easily be copy-pasted into a Spreadsheet software from Table 1 of the peer-reviewed article version of the report [8]. After removing sum lines and columns, removing the thousand separator, tweaking header names, saving as csv and importing into R, you can produce your bar chart:

```
> cancer =
↪  read.csv('C:/Users/Myriam/Documents/r-tutorial/cancer.csv')
> cancer
                                cancer cases_male cases_female
                                ↪  deaths_male deaths_female
1                               Tongue      12490         4620
↪  1750           760
2                                Mouth       7980         5600
↪  1770           880
...
47 Other & unspecified primary sites      16520        15290
↪  23950         20610
> str(cancer)
'data.frame':   47 obs. of  5 variables:
 $ cancer       : Factor w/ 47 levels "Acute lymphocytic
 ↪  leukemia",..: 41 19 32 27 10 38 36 9 34 3 ...
 $ cases_male   : int  12490 7980 14250 2440 13480 16520 ...
```

```
$ cases_female : int  4620 5600 3340 820 3810 9720 ...
$ deaths_male  : int  1750 1770 2480 1280 12850 6510 ...
$ deaths_female: int  760 880 750 360 3000 4290  ...
```

2.2.4 Histogram

2.2.5 Line graph

2.2.6 Scatter graph

2.2.7 Box and whiskers graph

2.3 Numbers

2.3.1 Center

Mean

Median

Mode

2.3.2 Dispersion

Range

Variance

Standard deviation

Coefficient of variation

Quartiles and percentiles

2.3.3 Shape

Skewness

Kurtosis

L-moments

Chapter 3

Probabilities

Chapter 4

Statistics

Chapter 5

Inferential statistics

Chapter 6

Cheat sheet

6.1 Plumbing

?	?exact_function_name
??	??keyword
typeof	typeof(R_variable)
class	class(R_variable)
str	str(R_variable)
colnames	colnames(R_variable)
as.integer	as.integer(R_variable)

6.2 Data import and export

read.csv	read.csv('delimited_data.csv', header=TRUE, sep=",", dec=".")
read.fwf	read.fwf('fixed_width_data.txt', widths=c(10, 5, 4), header=TRUE, skip=2)
write.csv	write.csv(R_variable, file='desired_file_name.csv', append=FALSE)

Bibliography

[1] Earl F. Glynn. *Chart of R Colors*. Apr. 2005. URL: http://research.stowers.org/mcm/efg/R/Color/Chart/ (visited on 09/10/2018).

[2] Klara Johannson, Mattias Lindgren, and Ola Rosling. *Infant mortality rate (per 1,000 live birth)*. Oct. 2015. URL: https://docs.google.com/spreadsheets/d/1OHMMuHbSFKDolNHXsmgHYlkjSKfAZyyY1P-ddMu_Fz0/pub# (visited on 09/10/2018).

[3] Douglas C. Montgomery. *Design and Analysis of Experiments*. English. 9th ed. Wiley, Apr. 2017. ISBN: 978-1-119-32093-7.

[4] World Health Organization. *Ebola Situation Reports | Ebola*. URL: http://apps.who.int/ebola/ebola-situation-reports (visited on 09/10/2018).

[5] *Pie chart*. en. Page Version ID: 856409948. Aug. 2018. URL: https://en.wikipedia.org/w/index.php?title=Pie_chart&oldid=856409948 (visited on 09/09/2018).

[6] *Quick-R: Data Types*. URL: https://www.statmethods.net/input/datatypes.html (visited on 09/10/2018).

[7] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2016. URL: https://www.R-project.org/.

[8] Rebecca L. Siegel, Kimberly D. Miller, and Ahmedin Jemal. "Cancer statistics, 2018". en. In: *CA: A Cancer Journal for Clinicians* 68.1 (Jan. 2018), pp. 7–30. ISSN: 1542-4863. DOI: 10.3322/caac.21442. URL: https://onlinelibrary.wiley.com/doi/abs/10.3322/caac.21442 (visited on 09/16/2018).

[9] Reed A. C. Siemieniuk et al. "Arthroscopic surgery for degenerative knee arthritis and meniscal tears: a clinical practice guideline". en. In: *BMJ* 357 (May 2017), j1982. ISSN: 1756-1833. DOI: 10.1136/bmj.j1982. URL: https://www.bmj.com/content/357/bmj.j1982 (visited on 09/18/2018).

[10] American Cancer Society. *Cancer Facts & Figures 2018*. en. Tech. rep. Atlanta, 2018. URL: https://www.cancer.org/research/cancer-facts-statistics/all-cancer-facts-figures/cancer-facts-figures-2018.html (visited on 09/13/2018).

[11] Tyler Vigen. *Spurious Correlations*. English. Hachette Books, May 2015. ISBN: 978-0-316-33943-8.

Glossary

continuous variable A variable that refers to continuous data, i.e. that can take on an infinite number of values (ex. height in mm), as opposed to categorical data (ex. color of eyes). 3, 4

dependent variable The "explained" variable in a relationship, the one we try to understand as a consequence of another factor. For example, when studying the effect of smoking on lung cancer, lung cancer is the dependent variable.. 4

discrete variable A variable that refers to categorical data (ex. color of eyes), as opposed to continuous data (ex. height in mm). 3, 4, 9, 12, 14

independent variable The "explaining" variable in a relationship, the one that drives a phenomena. For example, when trying to understand the causes of diabetes, body mass index would be an independent variable.. 4

ordinal variable A qualitative variable where the values can be ordered (ex. small, medium, large). 3, 12

qualitative variable A variable that is recorded with words rather than numbers (ex. color of eyes, state of mind). 3

quantitative variable A variable that is measured with numbers (ex. number of cases, height in mm). 3