Special Module: Methods in Mol Ecol and Pop Gen

Rebecca S. Chen

Table of contents

1 Methods in Mol Ecol and Pop Gen

2 Introduction

This webpage/document contains all the practical materials for the Spezial Modul entitled "Methods in Molecular Ecology and Population Genetics".

The order of the practicals/tests on this webpage are in the order of the way we will discuss them throughout the course. Please don't hestitate to ask David or Rebecca any questions throughout the course.

If anything is unclear or not working, please let us know!

Each session will contain text, exercises and answers. We encourage you to really think before you reveal the answer, otherwise you won't learn. If you need any help getting to the answer, you can also ask us questions to put you in the right direction. The same applies to helping you troubleshoot, of course.

Remember, even the experts will google how to code things. Once you get more fluent in the language, you can do simple things by heart. But just like learning any langauge, this takes time and practise. So don't be afraid to look into your notes or look up how to do things, this is how every person works in R

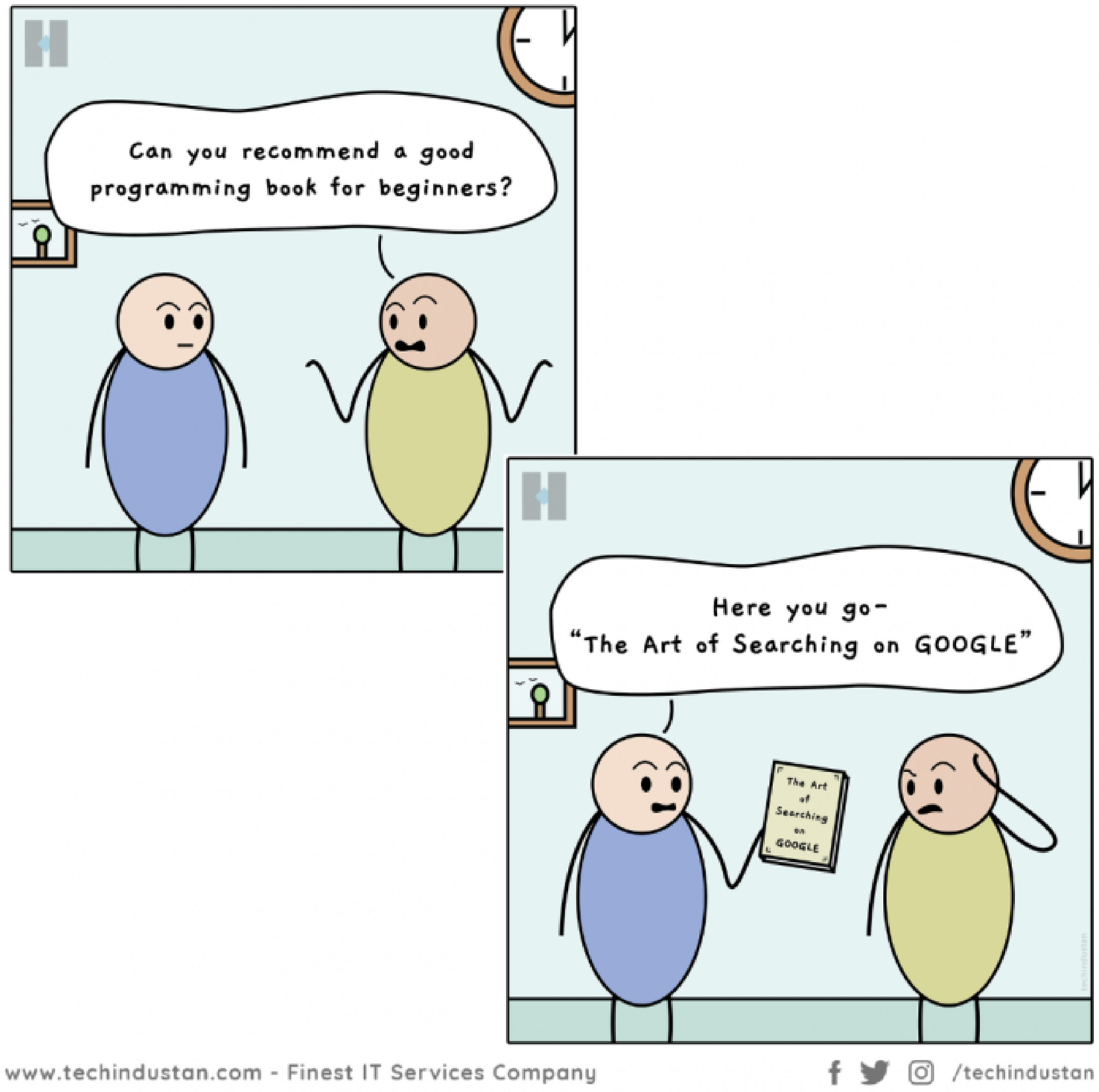
Can you recommend a good programming book for beginners?
Here you go- "The Art of Searching on GOOGLE"
The Art of Searching on GOOGLE
www.techindustan.com - Finest IT Services Company
/techindustan

Figure 2.1: Use the resources available to you!

3 Basic skills - a test

3.1 Datacamp

In the first week, you will have almost two full days to learn the basics of R yourself. We have given you access to DataCamp, a great interactive platform to learn R at any level. The courses we would like you to complete because they prepare you with all you need to know to follow the rest of the course are as follows:

- "Introduction to R"
- "Introduction to tidyverse"
- "Introduction to dplyr"

Please follow them at your own speed. If you think something is too easy because you already have some prior experience, you can skip through it and learn at your own speed. Feel free to start other courses too. All we want you to be able to do, is gain the skills and experience to be able to complete the exercises below. These exercises are therefore not graded in any way, it's just a way to make sure you understand the basics to continue with the rest of the course.

! A note on AI

You can find the answers and code when you click on the toggle, but try to do as much as you can without looking at the answers first. You can use Google or your notes from DataCamp to find the answers, but please don't use ChatGPT or other AI tools to help you find the answers directly. For example, don't just put the question as a prompt, but you could ask: how do I calculate the mean in R? Or: how can I make a basic plot with ggplot2 to visualise XYZ? In other words: don't 'cheat', but if you want to use ChatGPT, use it in a way that will help you learn and it doesn't hand the answer to you.

3.1.1 Data description

Let's work with some global height dataset. It was downloaded from the website https://ncdrisc.org/data-downloads-height.html and the file can be found in the `data` directory. The data contain the mean heights of adolescents of both boys and girls at different age groups. The mean height is the mean height of a certain age group in a specific year.

3.1.2 Part 1: exploration

- The dataset is called `NCD_RisC_Lancet_2020_heigh_child_adolescent_global.csv` and is stored in the `data` directory. What does the `.csv` extension mean?

Answer

Csv stands for comma separated file. It is a file format to store tables (such as Excel files), where each line represents a data record, and fields are separated by commas. CSV files are commonly used to store data, but you have to be careful if your data contains text that might include comma's, because it will think the sentence needs to be split up into different fields! Other ways to separate fields are for example by using tabs () or spaces.

- Import the data set and assign it to a variable (called e.g. `data`)

Code and answer

We are working within an R project, so we can use relative path names. If you prefer using the absolute path name, this might look different. Also, path specification is dependent on your operating system. This has been done in iOS (Apple), but Windows looks different.

```
# using read.csv
data <-
↪ read.csv("../data/NCD_RisC_Lancet_2020_height_child_adolescent_global.csv")

# or absolute path name
data <-
↪ read.csv("/Users/vistor/Documents/Work/GitHub/postdoc/special_module_molecol_popgen/dat
↪

# or using read table, make sure you specify there's a header!
data <-
↪ read.table("../data/NCD_RisC_Lancet_2020_height_child_adolescent_global.csv",
↪ sep = ",", header=T)
```

- How many data entries are there?

Code and answer

```r
nrow(data)
```

1050 entries because there are 1050 rows, excluding the header

- What is the youngest age group in the dataset, and what is the oldest age group (i.e. how old are those groups)?

Code and answer

First we need to know what columns there are in the data and what they are called, which can be done in different ways.
Remember you can always view the data (by clicking on it in the environment or by calling `View(data)`).

```r
#this shows you just the structure of the table
str(data)

#this tells you the column names of the table
names(data)

# this shows you the first 6 entries and the column names
head(data)
```

From this we can see that the column that stores the age is called `Age.group`

```r
# minimum age i.e. youngest age class
min(data$Age.group)

# maximum age i.e. oldest age class
max(data$Age.group)
```

So 5 and 19, respectively.

3.1.3 Part 2: cleaning the data

- What is the data type / class of the mean height and its standard error?

Code and answer

```r
# this tells us the class of a specific column
class(data$Mean.height)
class(data$Mean.height.standard.error)

# this tells us the class of all columns
str(data)
```

Both are characters/strings.

- Hmm, this data class is probably incorrect… It should be numeric, right? What's causing this?

Code and answer

A classic coding issue, "what's up with my data"? Figuring out why something is not behaving as expected will happen most of the time you are coding. How do we diagnose the problem?
We can start by having a look at the data using e.g. `View(data)` and by doing some tests.

```r
# let's see what happens when we change the data class to numeric, and
↪ store this in a test variable.

test <- as.numeric(data$Mean.height)
```

It gives a warning which indicates it could not change one or more of the values to numeric so it had to be put to NA. Why?
What sometimes works is to sort the column, this might help us diagnose what is happening in such a large table. When we look at `View(data)` / the data tab, we can click on the column `Mean.height` and it will sort it. Click on it again, and it will sort it in reverse order
Whatever way you used, you might have identified the issue!
There are two "ERRORS" at row number 112, a boy of age class 1992 and age 11. This text is causing the error, as R doesn't recognise this column as numeric now.

- Fix the error and change the data class to numeric

Code and answer

We can't correct the value because we don't know what happened with this boy's height measurement. So let's either remove the entire row or change the values to NA. You could do this manually by going into the raw data using e.g. Excel. But a better practise is by doing this in the code - this way we can always track exactly what we did, and others can see it too because it is documented.
In the first method below, we simply remove the 112'th row, so we remove the entire data point. It's always smart to store something in a different variable if you remove something so you do not override the original data, I'm showing you an example here but we'll continue with the 'data' variable as we know it's all good.
In the second and third method, we remove the text "ERROR" and instead replace it with the 'official' NA value that R recognises as NA.

```
## method 1
# remove the row
data_clean <- data[-112,]

# then we change the data class to numeric and we do not get a warning
data_clean$Mean.height <- as.numeric(data_clean$Mean.height)
# same for the SE
data_clean$Mean.height.standard.error <-
 ↪  as.numeric(data_clean$Mean.height.standard.error)

## method 2
# this is what we had done before and automatically put the 'character'
 ↪  ERROR as NA, you'll get a warning but now we know why

data$Mean.height <- as.numeric(data$Mean.height)

# same for the SE
data$Mean.height.standard.error <-
 ↪  as.numeric(data$Mean.height.standard.error)

## method 3

# this is the same as above, but manually which is a bit longer and more
 ↪  complex

data$Mean.height[which(data$Mean.height=="ERROR")] <- NA
data$Mean.height.standard.error[which(data$Mean.height.standard.error=="ERROR")]
 ↪  <- NA
```

```
# change to numeric, now we won't get a warning
data$Mean.height <- as.numeric(data$Mean.height)
data$Mean.height.standard.error <-
 ↪  as.numeric(data$Mean.height.standard.error)
```

There we go! It's always good to check if things worked, so you can use e.g. `str(data)` now to check that R indeed changed the columns to numeric.

3.1.4 Part 3: summary statistics

- What is the mean across years of the mean heights in the dataset?

Code and answer

The function that calculates means is, intuitively, `mean()` which is a base function (i.e. you don't need to install a package for this). If you want to know more details about how to use a function, you can use `?mean`

```
# we have to make sure we remove the NA when calculating the mean though!
mean(data$Mean.height, na.rm = T)

# or we use the data that doesn't contain the NA
mean(data_clean$Mean.height)
```

- What is the mean across years of the mean height, calculated separately for boys and girls?

Code and answer

Here's where your preference can kick in: do you want to use dplyr or something else, like base R or data.table? I will show you multiple options, just see what looks more intuitive to you. To make sure we can use dplyr, we have to load the library.
To answer this question, we have to do multiple steps: divide the data among the sexes somehow, and calculate the mean. This can be done in steps, or in one go. If you're a computational scientists, you would want things to be done in as little code as possible. But since we are learning, just do what makes most sense to you.

```
# method 1 with dplyr
library(dplyr) # we can use dplyr to use the filter option
```

```r
girls <- data %>% filter(Sex == "Girls")
boys <- data %>% filter(Sex == "Boys")

mean(girls$Mean.height, na.rm = T)
mean(boys$Mean.height, na.rm = T)

# method 2 without dplyr
mean(data$Mean.height[which(data$Sex == "Girls")], na.rm=T)
mean(data$Mean.height[which(data$Sex == "Boys")], na.rm=T)
```

- In what year was the standard error lowest for boys and girls separately, for the age class 12? (i.e. the variance smallest?)

Again, we need to take multiple steps here: only look at data from the Age group 12, and then look at what year the variance was the smallest (i.e. the minimal variance).

I'll only show you the dplyr way, but you can use many different methods to get to the same answer here.

Code and answer

```r
# method 1: with dplyr
girls_12yr <- girls %>% filter(Age.group == 12)
boys_12yr <- boys %>% filter(Age.group == 12)

girls_12yr %>% arrange(Mean.height.standard.error) %>% head() # 2003,
 ↪  with a SE of 0.09367336

boys_12yr %>% arrange(Mean.height.standard.error) %>% head() # 2004, with
 ↪  a SE of 0.1342430
```

3.1.5 Part 4: plotting

- Make a plot that shows you whether the mean height of 18 year olds increased or decreased over time. Bonus if you do it seperately per sex!

Code and answer

Let's get plotting! First choose: do you want to use ggplot2 (recommended) or base R to make your plot?
Before you start making code, think what elements should be on the axes. What should

be on the X axis, what variable on the Y axis, and what type of graph would represent the data best? A boxplot, scatterplot, line graph, pie chart? Choose the right element accordingly and then start writing the code.
We also have to use a portion of the data only, again. We can use dplyr (`filter()`) but I will also show you another function here, `subset()` which is a base R function to do something similar to `filter()`.

```r
# method 1: with ggplot
library(ggplot2)

ggplot(subset(data, Age.group == 18), aes(x = Year, y = Mean.height)) +
 ↪  geom_point(aes(col = Sex)) + geom_line(aes(col = Sex)) +
 ↪  theme_classic()

# method 2: base R
# subset the data
age_18 <- subset(data, Age.group == 18)

# set up empty plot
plot(
  age_18$Year, age_18$Mean.height,
  type = "n",
  xlab = "Year",
  ylab = "Mean height"
)

# add points and lines by Sex
for (s in levels(age_18$Sex)) {
  ds <- age_18[age_18$Sex == s, ]

  points(ds$Year, ds$Mean.height,
         col = cols[s], pch = 16)

  lines(ds$Year, ds$Mean.height,
        col = cols[s], lwd = 2)
}

# add legend
legend("topleft",
       legend = levels(as.factor(age_18$Sex)),
       pch = 16,
       lwd = 2,
         bty = "n")
```

As we see, both the mean height of girls and boys aged 18 years increased over time. What is also clear is that base R is a lot longer, and you require a lot of separate elements.

Well done! If you managed to answer all of the above, with or without some help from your notes and the internet, you're ready to explore genetic data and answer some pop gen questions.