

# Introduction to R

## Reading, writing and exploring data

R-peer-group

QUB

February 12, 2013



- Review of last weeks exercise
- Introduction to R's read/write system
- Reading data into R
  - ▶ Useful functions
  - ▶ Common problems
  - ▶ Practical using data recorded last week
- Data exploration
  - ▶ Data extraction
  - ▶ Manipulation
  - ▶ Basic analysis (inc. plotting)
  - ▶ practical
- Writing data back out of R
  - ▶ Useful formats
  - ▶ Helpful Packages
  - ▶ Practical



Q Record the data and save them as objects in R

A example answer

```
lgths <- c(46.58, 44.25, ..., 46.55)
mean.lgth <- 45.15
sd.lgth <- 1.73
```

Q Check to make sure that the mean and standard deviation given to you corresponds to the mean and standard deviation of the data you've recorded.

```
A mean(lgths) == mean.lgth    #FALSE
sd(fish.lengths) == sd.lgth  #FALSE
```

Also: `?identical`, `?all.equal`, `?isTRUE`



Q If there is a problem what do you think it is considering the fact that your colleague has told you that all fish were longer than 40 inches? A 4 has been left of the start of one of the lengths in the data. You check with your colleague and they have told you that the problem fish was recorded incorrectly. Its length should be 44.23 inches. How can you fix it?

A example answer

```
lgths[6] <- 44.23
```

Q Using logical tests, demonstrate that the mean and standard deviation calculated from the fixed data are the same as those sent to you by your colleague.

```
A mean(lgths) == mean.lgth    #TRUE  
sd(lgths) == sd.lgth        #TRUE
```



Q Convert your vector of lengths from *inches* to *cm*, saving the results in a new vector. [1 inch = 2.54 cm]

A example answer

```
lgths.cm <- lgths * 2.54
# or
lgths.cm <- cm(lgths)
```

Q Calculate the new mean and standard deviation of the converted data.

```
A new.mean <- mean(lgth.cm)
new.sd <- sd(lgth.cm)
# Alternatives (manual)
# Mean
new.mean <- sum(lgth.cm)/lgth(lgth.cm)
# sd
new.sd <- sqrt(var(lgth.cm))
# or
new.sd <- (sum((lgths.cm-mean(lgths.cm))^2)/(length(lgths.cm)-1))^0.5
```



Q Calculate the variance of both data sets. [ $sd = \sqrt{Var(x)}$ ].

A example answer

```
var.lgths <- var(lgths)
var.lgth.cm <- var(lgth.cm)
# or
var.lgths <- new.sd^2
```



## Getting data into R



- Initially, one of the most challenging aspects of learning R is how to access your data so you can analyse them
- There are many functions which make this easy, you just have to make sure your data are in the correct format
  - ▶ File type
  - ▶ Field delimiter
  - ▶ Headers
- The most commonly used function is `read.table()`
- There are many other similar functions with very similar behaviour and arguments
  - ▶ `read.csv()`, `read.delim()`, `read.fwf()`, `read.DIF()`
- For more complex data files you can use `scan()` directly, which is the backbone of most other read functions





- Imagine we have a data set in a spreadsheet that looks like this:

Name	Height(m)	Weight(kg)	BMI
Steve	1.78	75.2	23.73
Emily	1.56	120.4	49.47
George	2.11	91.3	20.51
Nicola	1.65	45.6	16.75
Pierre	1.95	86.3	22.70

- The simplest way read this data into R would be:
    - > Save the data to .txt or .csv format (lets say we name it 'mydata.txt')
    - > Use `read.table()` or `read.csv()` respectively, to read the data
- ```
> myData <- read.table("mydata.txt", header = TRUE)
```
- This command will load the data set into R so that you can carry out your analyses



- After reading data into R, it is important to check that it has maintained the correct format
- We can inspect the data in a number of different ways
  - > In RStudio we can click on the variable name in the workspace
  - > Type `fix(myData)` into the console (Usually reserved for editing cells)
  - > In RStudio, type `View(myData)` (limited to 100 columns and 1000 rows)
  - > Print to the console (only for small data sets)

```
> myData
```

|   | Name   | Height.m. | Weight.kg. | BMI   |
|---|--------|-----------|------------|-------|
| 1 | Steve  | 1.78      | 75.2       | 23.73 |
| 2 | Emily  | 1.56      | 120.4      | 49.47 |
| 3 | George | 2.11      | 91.3       | 20.51 |
| 4 | Nicola | 1.65      | 45.6       | 16.75 |
| 5 | Pierre | 1.95      | 86.3       | 22.70 |



- R has done something to the header names of our data set
- Some characters/symbols are not permitted in headers as these are treated as variables in R
  - > R replaced the characters '()' with '.' automatically in our data
  - > It will also do this for whitespace characters such as tabs and spaces in headers
- We could look at only the header of `myData` by typing:

```
> names(myData)
```

```
[1] "Name"          "Height.m."    "Weight.kg."  "BMI"
```

- We could also use:

```
> colnames(myData)
```

```
[1] "Name"          "Height.m."    "Weight.kg."  "BMI"
```



- Because we will use the headers as variable names for each column in our data set, it is sometimes convenient to change their names to something shorter

```
> names(myData) <- c("nms", "hgt", "wgt", "bmi")  
> # or  
> colnames(myData) <- c("nms", "hgt", "wgt", "bmi")
```

- Now if we look at `myData`, the header names should have changed

```
> myData  
  
      nms  hgt  wgt  bmi  
1 Steve 1.78 75.2 23.73  
2 Emily 1.56 120.4 49.47  
3 George 2.11 91.3 20.51  
4 Nicola 1.65 45.6 16.75  
5 Pierre 1.95 86.3 22.70
```



- We can also inspect the object (our data) in many different ways
  - `typeof()` - Tells you the type/class of an object (e.g. `numeric`, `character`)
  - `str()` - Displays the structure of the object
- We can also ask R specific question about our data object
  - `is.numeric()` - Is my object numeric?
  - `is.data.frame()` - Is my object a dataframe?
  - `length()` - What length is my data object?
- We could ask R how many rows and columns are in `myData` like this:

```
> nrow(myData)
```

```
[1] 5
```

```
> ncol(myData)
```

```
[1] 4
```

```
> # or
```

```
> dim(myData)
```

```
[1] 5 4
```



- We can extract useful information from our data in a number of ways
  - ▶ Index mapping - remember the `[]` symbols from last week
  - ▶ Extraction - done using the `$` symbol

- If we wanted to find out what value was in the 3<sup>rd</sup> row of the 4<sup>th</sup> column using index mapping, we would simply type:

```
> myData[3, 4]
[1] 20.51
```



- To *extract* the BMI column from the data we could type the following:

```
> BMI <- myData$bmi
> BMI
[1] 23.73 49.47 20.51 16.75 22.70
```

- Let's see if our new object BMI is equal to the bmi or 4<sup>th</sup> column of myData

```
> identical(BMI, myData[, 4])
[1] TRUE
```



- 1 Create a new project in a convenient directory on your system called 'Session-2'
- 2 Download [this file](#) and save it to the directory you created your R project in.
- 3 Check what format the data are in (.txt or .csv), and use an appropriate function to read them into R, assigning them to the variable `morph1`. 
- 4 Inspect your data. Notice that on the 31<sup>st</sup> row there are '?' in two columns. These are missing data. In R missing data are usually recorded as `NA`. Can you think of a way, using `index mapping`, that you could replace these elements with `NA`s? 
- 5 Recode the header names to more convenient header names



# Data exploration





- We can inspect the object `morph1` using a few more functions

```
> head(morph1) # displays the first 6 rows
```

|   | sex | col    | hgt | lrh  | llf |
|---|-----|--------|-----|------|-----|
| 1 | F   | Black  | 73  | 18   | 30  |
| 2 | M   | Black  | 76  | 21   | 30  |
| 3 | M   | Brown  | 68  | 18   | 30  |
| 4 | F   | Ginger | 62  | 16.2 | 25  |
| 5 | F   | Brown  | 67  | 16   | 28  |
| 6 | F   | Blonde | 68  | 16   | 25  |

- When R reads data, it treats different variable types in different ways by coercing them to certain classes
- `str()` is the easiest way to inspect how R has treated variables in our dataframe



- We can use this function as follows:

```
> str(morph1)
```

- `morph1$lrh` and `morph1$llf` should be numeric variables, instead R has coerced them to the class 'factor' because of the original missing data values used (i.e. ?).
- To coerce them to numeric variables use the following:

```
> morph1$lrh <- as.character(morph1$lrh)
```

```
> morph1$lrh <- as.numeric(morph1$lrh)
```

```
> morph1$llf <- as.numeric(levels(morph1$llf)[morph1$llf])
```

```
# or
```

```
morph1$lrh <- as.numeric(as.character(morph1$lrh))
```

```
# or
```

```
morph1$lrh <- as.numeric(levels(morph1$lrh)[morph1$lrh])
```



- We can check if our changes worked

```
> is.numeric(morph1$lrh)
```

```
[1] TRUE
```

- We can also inspect our data using the `summary()` function

```
> summary(morph1)
```

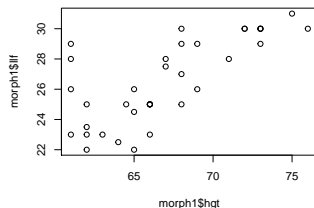
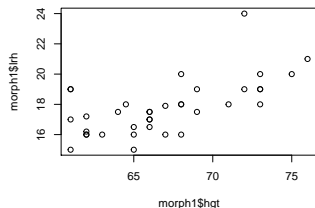
| sex  | col       | hgt           | lrh           | llf           |
|------|-----------|---------------|---------------|---------------|
| F:25 | Black : 5 | Min. :61.00   | Min. :15.00   | Min. :22.00   |
| M:12 | Blonde: 8 | 1st Qu.:64.00 | 1st Qu.:16.43 | 1st Qu.:24.88 |
|      | Brown :21 | Median :66.00 | Median :17.50 | Median :26.00 |
|      | Ginger: 2 | Mean :67.09   | Mean :17.79   | Mean :26.53   |
|      | Grey : 1  | 3rd Qu.:71.00 | 3rd Qu.:19.00 | 3rd Qu.:29.00 |
|      |           | Max. :76.00   | Max. :24.00   | Max. :31.00   |
|      |           |               | NA's :1       | NA's :1       |



# Graphical exploration

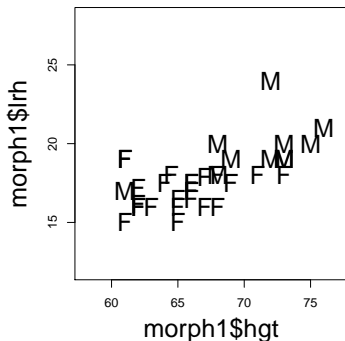
- As insightful biologists, we might make the prediction that we should observe a positive relationship between an individual's height and both their right hand length and their left foot length. It is easy to explore this prediction visually in R

```
> plot(x = morph1$hgt, y = morph1$lrh)
> plot(x = morph1$hgt, y = morph1$llf)
```



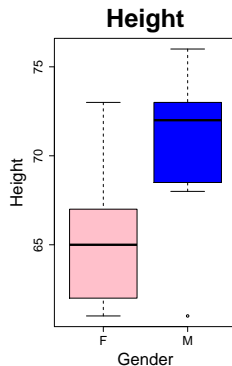
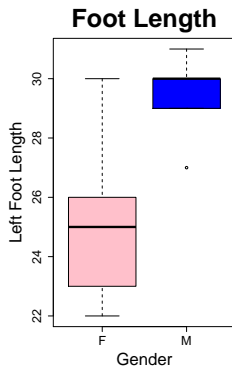
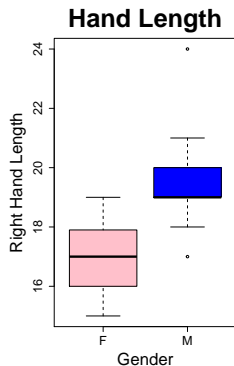
- We can use R's powerful plotting capabilities to explore the data in many ways

```
> plot(x = morph1$hgt, y = morph1$lrh,  
+      pch = as.character(morph1$sex),  
+      cex = 2)
```



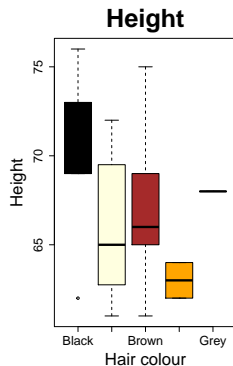
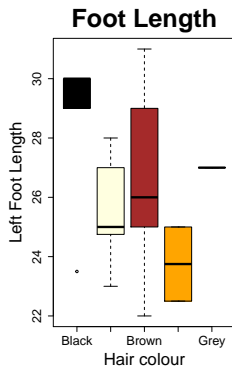
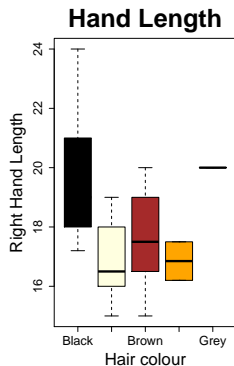
- We can explore the differences between genders for each of the measurements

```
> plot(x = morph1$sex, y = morph1$lrh, col = c("pink", "blue"),  
+       xlab = "Gender", ylab = "Right Hand Length")
```



- What about hair colour?

```
> plot(x = morph1$col, y = morph1$lrh, xlab = "Hair colour",  
+      col = c("black", "lightyellow", "brown", "orange", "lightgrey"),  
+      ylab = "Right Hand Length")
```







Genome size is the amount of DNA contained within a single copy (i.e. a single cell) of a single genome, usually measured in picograms ( $\mu\text{g} = 10^{-12}$ ).

Genome size has been measured for a large number of species on the planet as it was once believed that there was a correlation between genome size and complexity. However, following the discovery of 'non-coding DNA', and observations to the contrary, this idea was understood to be false.

You will be provided with a data set containing genome sizes for various vertebrates grouped into different classes and subclasses. Your task is to follow the instructions to explore the relationship between genome size and taxonomic grouping.



- 1 Download the genome size file [here](#) and save it to your session-2 project directory.
- 2 Inspect the file and read it into R using the appropriate function.
- 3 The data object will have a number of issues. Use the various exploratory functions demonstrated to:
  - a Replace any missing data with NAs
  - b Coerce incorrectly classified variables
  - c Plot the relationship between class and genome size
  - d Plot the relationship between subclass and genome size.
  - e Which class has the most striking difference in genome size among subclasses?
- 4 Create a new variable in the genome size data frame named `basepairs`. This variable should contain the basepair equivalent to the genome size for each species in the data.  
1pg  $\approx$  978,000,000 basepairs