

Exercises: Introduction to R

Licence

This manual is © 2011-18, Laura Biggins and Simon Andrews.

This manual is distributed under the creative commons Attribution-Non-Commercial-Share Alike 2.0 licence. This means that you are free:

- to copy, distribute, display, and perform the work
- to make derivative works

Under the following conditions:

- Attribution. You must give the original author credit.
- Non-Commercial. You may not use this work for commercial purposes.
- Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a licence identical to this one.

Please note that:

- For any reuse or distribution, you must make clear to others the licence terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.
- Nothing in this license impairs or restricts the author's moral rights.

Full details of this licence can be found at

<http://creativecommons.org/licenses/by-nc-sa/2.0/uk/legalcode>

Exercise 1: Simple Calculations

- Use R to calculate the following:
 - $31 * 78$
 - $697 / 41$
- Assign the value of 39 to `x`
- Assign the value of 22 to `y`
- Make `z` the value of `x - y`
- Display the value of `z` in the console
- Calculate the square root of 2345, and perform a \log_2 transformation on the result.

Exercise 2: Working with Vectors

- Create a vector called `vec1` containing the numbers 2,5,8,12 and 16
- Use `[lower]:[upper]` notation to make a second vector called `vec2` containing the numbers 5 to 9
- Subtract `vec2` from `vec1` and look at the result
- Use `seq()` to make a vector of 100 values starting at 2 and increasing by 3 each time
- Extract the values at positions 5,10,15 and 20 in the vector of 100 values you made
- Extract the values at positions 10 to 30 in the vector of 100 values you made

Exercise 3: Lists and Data Frames

- Enter the following into a vector with the name `'mouse.colour'`. Remember to surround each piece of text with quotes.


```
purple
red
yellow
brown
```
- Display the 2nd element in the vector (red) in the console.
- Enter the following into a vector with the name `'mouse.weight'`:


```
23
21
18
26
```
- Join the 2 vectors together using the `data.frame` function to make a data frame named `mouse.info` with 2 columns and 4 rows. Call the first column 'colour' and the second one 'weight'.
- Display the data frame in the console.
- Display just row 3 in the console
- Display just column 1 in the console
- Display the item of data in row 4, column 1.

Exercise 4: Reading in data from a file

Set your working directory to where the data files are stored. Make sure that the folder of data files has been unzipped. e.g. `setwd("D:/Data_folder")`

4a

- Read the file 'small_file.txt' into a new data structure. This is a tab delimited file so you should use `read.delim()`.
- View the data set to check that it has imported correctly.

4b

- Read the file 'Child_Variants.csv' into a new data structure. This is a comma separated file so you should use `read.csv()`.
- Use `head` and `View` to look at the data set to check that it has imported correctly.
- Display row 11.
- Calculate the `mean` of the column named `MutantReadPercent`. Think about how you are going to access a single column first, then once you can access the data pass it to the `mean` function.

Exercise 5: Filtering

- Create a filtered version of the child variants dataset which only includes rows where the `MutantReadPercent` is ≥ 70 . Save this under a new name. The process you should use is:
 - Make sure you can access the `MutantReadPercent` column from the data frame
 - Do a logical test on this data for ≥ 70 to obtain a boolean vector
 - Use this boolean vector as the row selector to filter the original dataset. Remember that you need to say which rows AND columns you want, so as you want all columns you'll need to put a comma after the boolean vector in your data frame filter.
- From the filtered data frame we want to know the frequency with which G, A, T and C bases were mutated. You will need to use an exact match filter against the text in the `REF` column. To get the count you can simply `sum()` the boolean array values (true counts as 1, false counts as 0). Create a vector called '`mutation.counts`' of the counts for the different mutation frequencies (you will need to make this manually having calculated the values).

Exercise 6: Histograms, Boxplots and Barplots

- In the original child variants dataset draw a histogram (`hist`) of the `MutantReadPercent` column, try increasing the number of categories (`breaks`) to 50.
- Plot a `boxplot` of the `MutantReadPercent` values from both the original child variants and the same column from the filtered dataset you made in Exercise 5 (`MutantReadPercent` ≥ 70). Check that the distributions look different. To do this you can just pass two vectors to the `boxplot` function.
- Plot the results of the vector created in Exercise 5 ('`mutation.counts`') as a `barplot`. Use the `names.arg` function to show which mutation is which and make a vector of colours (`c("black", "green", "red", "blue")`) to give the bars different colours (using the `col` parameter).

[Optional] Exercise 7: Scatterplots and Line graphs

- Read in the file `'neutrophils.csv'`. This is a comma-delimited file so use `read.csv()`.
- Create a `boxplot` of the 4 samples and put a suitable title on the plot (you should be able to pass the entire data frame to `boxplot` since you're plotting all of the data there).
- Use the `colMeans()` function to calculate the mean of each dataset. Note: the data contains missing values (NA) so look at the help file for `colMeans` to find out how to ignore these. Plot the means as a `barplot`.
- Read in the file `'brain_bodyweight.txt'`. This is a tab delimited file so you can use `read.delim()`. The first column contains species names, not data, so use `row.names=1` to set these up correctly in your data frame.
- Log transform the data (base 2).
- Create a scatterplot with default parameters with the log transformed data.
- Read in the file `'chr_data.txt'`.
- Remove the first column (you can simply select the second and third columns and then overwrite the original variable)
- Create a line graph like the one below. The process will be:
 - Use `plot()` to plot the genome position vs the ABL1 dataset. You will need to set the `type` parameter to `l` (lower case L) to get this to be a line graph.
 - Use `legend` to add in a legend to the plot, and `title` to add a title to the plot (you could also do this by adding a main parameter to the original plot function)

