R: Introduction to Basic Features

Introduction to R

Authors: Helen Lockstone and Matthieu Miossec Bioinformatics Core

Workshop aims

- To introduce the R language and environment
- To explain some programming jargon and concepts
- To use R to run some typical tasks
- To be aware of the vital importance of good programming habits

R: What is it?

- Powerful statistical computing environment and programming language
- Developed by Robert Gentleman and Ross Ihaka at the University of Auckland in mid 1990s – originally something to teach statistics that would run on a Mac
- Full story of R's development
- Now widely used all around the world

R: Strengths and Weaknesses

- Incredibly powerful and versatile statistical programming software. . . .but where do I start?
- Features that make life easier in many ways. . . .but potential pitfalls as well
- Open-source, free software with a strong support and development community
- Extensive additional functionality for bioinformatics and genomic data through the Bioconductor project

R: Basic Features

- Go to the workshop link
- This will introduce some features of working in the R environment, in particular the creation of objects and storing information in R using a data type known as 'vectors'

R: Basic Features

- If you are attending a live workshop, we will demonstrate and talk through the content.
- If you are self-learning, the document includes explanatory text and you can type the commands into an R session running on your local machine.
- Note there is an accompanying RMarkdown (.rmd) file, which can be opened in RStudio to run the code directly – however, most commands are short, and it is useful to type them yourself to get used to the syntax. Countdown challenge Once you have worked through the introductory material on creating and manipulating vectors in R, try the exercise on the next slide.

Countdown challenge

- Create a new vector containing a set of any 9 letters, including at least 3 vowels. Give your vector object a suitably descriptive name.
- Check the length of your vector equals 9

- Now extract subsets of letters to create as many new short words as you can
- hint: use the c() and : operators within square brackets

Example solution

```
> random_letters <- c("r", "t", "d", "e", "e",
"i", "a", "s", "n")
> length(random_letters)
[1] 9
> random_letters[c(8,7,6,3)]
[1] "s" "a" "i" "d"
> random_letters[c(1,7,6,8,4)]
[1] "r" "a" "i" "s" "e"
> random_letters[c(8,2,4:5,1)]
[1] "s" "t" "e" "e" "r"
> random_letters[c(2,1,4,4)]
[1] "t" "r" "e" "e"
```

Practical Session

- Next we will read in some example data files and perform a few typical tasks.
- From the course link, you can open either the pdf or rmd version of 'R_basic_features_practical' to work through the exercises.
- Make sure you also have both files – 'inflammation_data.csv' and 'sample.csv' - downloaded locally on your machine and have the working directory in R pointing to that location.

Practical session

- Again, if you are attending a live workshop we will go through the material together (or you can work through yourself if you prefer)
- At the end of the R basic features practical you will find a list of additional online resources to follow up from this introductory workshop
- In addition or in particular, the links on the following page might be usefulFurther R resources
- Online tutorials on using R for genomic data
- 'tidyverse' is set of packages designed to make handling data in R easier and more reliable

Good programming habits

1. Break it down into small steps
   - Write code to do individual parts of your overall task and build it up into a larger script
   - This makes it easier to get right and test as you are going along
   - It is also a good way to solve errors/bugs in your code
2. Pay attention to details and program carefully
   - Commands must be accurate both in their syntax and what they do
   - It is easy to overwrite objects, pull out wrong subsets of data, mix up sample names etc
   - To avoid this, you need to check carefully all the time what you are working with and what R produces after you run a command
   - Inspect object contents, make plots, test examples can all helpGood programming habits
3. Test your code thoroughly
   - You are the only person who can make sure it is doing what you intend it to do
   - R will give an error if a command is incorrectly specified according to the rules of the language but it can't tell you if the command you've written does not do what you think it does...
   - ...and it will go ahead and execute it regardless, because that is what you have instructed R to do!

4. Comment your code
    - Comments are really useful to jog your memory if you come back to some code after a while or want someone else to understand it.

Finally...

Please let us know if you have any questions and good luck on your programming journey!

We are happy to help anytime: bioinformatics@well.ox.ac.uk