

Developing R Programming Skills

18th-19th October 2018

Helen Lockstone and Ben Wright
Bioinformatics Core





Developing R Programming Skills

- Two-day workshop style course – self-paced and tailored to level of background knowledge
- Main course tutorial <https://tinyurl.com/ya9ce5ub>
- Additional material at:
http://www.well.ox.ac.uk/bioinformatics/training/Developing_R_Skills_Oct2018
- Key Course Aims
 - Introduce the R software through practical sessions
 - Help make R and associated resources accessible to novice programmers
 - Emphasise good programming practice

Housekeeping

- Part of teaching for DPhil programme in Genomic Medicine and Statistics at WHG, opened out to Medical Sciences Division
- No fire alarm test due – if alarm sounds, please leave by main building door
- Tea/coffee provided at 11am and lunch available from the canteen (12.30-1.30)
- Wrap up around 3.30 each day

Introductory Remarks



Why learn R?

- The ability to handle and analyse large-scale datasets is, and will continue to be, a key skill in modern biological research, as well as many other fields
- Generating data far outstrips our ability to interpret and make sense of it – and it is still hard to recruit good bioinformaticians
- R has become a key programming language for genomics due to the diverse set of packages available through BioConductor
<https://www.bioconductor.org/>



The downsides to R

- R is not an intuitive language to learn, even for those who are familiar with programming concepts
- It can take many months to start to feel comfortable writing your own R code, so be prepared for some investment of time, hard work and a steep learning curve
- R's extensive functionality and versatility are its main attributes but also the reason it can be hard to know where to begin...



Tips for Successful Programming

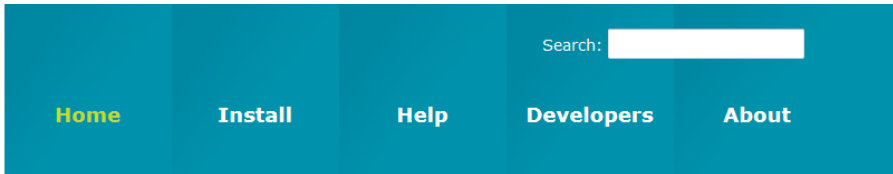
- Logical thinking and problem solving skills are vital
 - Decide what steps needed to solve a task
 - Troubleshooting error messages
 - Testing code to ensure it does what it should
- Consistency, accuracy and attention to detail
 - Easy to make unintentional mistakes (which R may well execute with no warning message)
 - Check what code is doing at every step
 - Mentally predict what should happen, so you can spot potential errors
- Accept that it can be a frustrating process!



What is R?

- R is a powerful software package for statistical analysis and also a high-level programming language
- Originally written in the 1990s for teaching statistics by Ross Ihaka and Robert Gentleman at the Department of Statistics of the University of Auckland
- It's open-source, available for free for Win, Mac, Linux and regularly updated (maintained by R Core development team)
- <http://www.r-project.org/index.html>
- Has become a mainstream research tool
 - Users have contributed hundreds of add-on packages (CRAN)
 - Bioconductor resource is invaluable for genomic data
<https://www.bioconductor.org/>

Bioconductor



About *Bioconductor*

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language, and is open source and open development. It has two releases each year, [1295 software packages](#), and an active user community. Bioconductor is also available as an [AMI](#) (Amazon Machine Image) and a series of [Docker](#) images.

News

- Bioconductor [3.4](#) is available.
- Bioconductor [F1000 Research Channel](#) launched.
- Orchestrating high-throughput genomic analysis with *Bioconductor* ([abstract](#)) and other [recent literature](#).
- Read our latest [newsletter](#) and [course material](#).
- Use the [support site](#) to get help installing, learning and using Bioconductor.

Install »

Get started with *Bioconductor*

- [Install Bioconductor](#)
- [Explore packages](#)
- [Get support](#)
- [Latest newsletter](#)
- [Follow us on twitter](#)
- [Install R](#)

Learn »

Master *Bioconductor* tools

- [Courses](#)
- [Support site](#)
- [Package vignettes](#)
- [Literature citations](#)
- [Common work flows](#)
- [FAQ](#)
- [Community resources](#)
- [Videos](#)

Use »

Create bioinformatic solutions with *Bioconductor*

- [Software](#), [Annotation](#), and [Experiment](#) packages
- [Amazon Machine Image](#)
- [Latest release announcement](#)
- [Support site](#)

Develop »

Contribute to *Bioconductor*

- [Developer resources](#)
- [Use Bioc 'devel'](#)
- 'Devel' [Software](#), [Annotation](#) and [Experiment](#) packages
- [Package guidelines](#)
- [New package submission](#)
- [Build reports](#)

Bioconductor packages extend functionality of R to all aspects of handling, processing and analysing genomic data
<https://www.bioconductor.org/>

Getting started with R

- Some aspects of R are analogous to tasks you might routinely perform in Excel:
 - sorting/filtering data
 - finding a particular occurrence of text
 - simple data summary statistics and tests
 - plotting graphs
- R provides far more complex functionality besides, particularly for statistical modelling/analysis and data visualisation. It can also perform small useful tasks with a simple line or two of code (where there may not be a quick way to do the equivalent in Excel) e.g. finding the overlap of two lists of genes

Getting started with R

- For those without a programming background, the biggest challenge might be getting used to how the R language is structured, and working with objects to store and manipulate data
- Imagine that instead of using Excel, we are giving R step-by-step instructions of what we want to do (e.g. filter a column for all values <0.05):
 - Our data could be created in Excel and read into R (like a table); it can be stored in an appropriate object (named by the user)
 - We'd need to specify the column of interest somehow (R has more than one way to do this) and set the condition 'values less than 0.05'
 - Finally we'd have to decide whether to display the output or save it in a new variable
- Although some aspects of R can be quite unintuitive and take some time to become familiar with, they provide the flexibility that makes the software so powerful.

Getting started with R

- R is extremely extensive and versatile, quite possibly no two people use it in the same way. So how best to get started?’
- Hopefully this course will help, and R itself is very well documented with a large and active user community (mailing lists, online forums like stack overflow etc) – usually googling any R problem will find relevant threads.
- Learn by doing – follow examples in the manuals or start by running scripts written by others to understand what the code does before moving on to modifying code or writing your own scripts from scratch.
- It’s impossible to remember the details of all R functions (or even know about all of them!) – make use of the help pages to check syntax, arguments, examples of usage etc

Some general advice

- A computer will do exactly what you tell it to do and you need to tell it every little step, in the right order
 - Imagine writing instructions that someone can follow to produce a particular result, including every tiny detail they would need
 - Akin to instructions for an experimental protocol - important to be very precise
- Comment your code as much as possible – if you need to revisit it at a later date, it will make much more sense!
- Check and double check (and check again) that your code is doing what you *think* it is doing – R has some default behaviours that may not always be realised, and can lead to problems if not spotted
 - Inspect the objects created, their length, contents and so on.
 - Manually check a few elements of the output to be sure they are correct

Problem solving

- You will inevitably run into problems when you are programming - try solving them by:
 - reading the error message
 - reading the help file
 - make a list of what could possibly be wrong and check these possibilities one by one
 - break down a complex step into smaller, simpler steps (this is also a useful way to build up a more complex piece of code)
 - If you really cannot solve a problem, remember there are usually alternative ways in R to achieve the same goal (using a related function for example)
 - Google the problem – many helpful forums/lists
 - If all else fails, post a question on the relevant help list (but be sure to read the guidance first!)

R documentation and help

- `> help.start()`
 - Most useful if you don't know which command to use
 - Starts a Search Engine in your favourite browser
 - Can get this also from the R GUI Help Menu
- If you do know which command to use but need to check details
 - `> ?plot`
 - or equivalently
 - `> help(plot) # help("plot")` in RStudio
 - Brings up the help page on the function 'plot'
- From the R GUI Help menu or the R website, you can get Manuals (in PDF)
 - An Introduction to R
 - R Reference manual

R documentation and help

- Further reading
 - Michael J Crawley (2005) *Statistics: An Introduction using R*. Wiley: Chichester, England
 - This is an excellent book, and well worth working through
 - Also has an associated web page with datasets, exercises etc
 - <http://www.imperial.ac.uk/bio/research/crawley/statistics/>
 - A selection of recommended online resources, tutorials, local courses and textbooks are collated at:
<https://help.it.ox.ac.uk/courses/R>
 - R-help and BioConductor mailing lists
 - Stack Overflow



<https://www.rstudio.com>

RStudio Interface

The screenshot displays the RStudio interface with the following components:

- Script Editor:** Contains the following R code:

```
1 # Write your script here
2
3 x = rnorm(100, 1, 1)
4 y = rnorm(100, 1, 1)
5
6 p = plot(x, y)
7
8 print(p)
```
- Environment Pane:** Shows the current environment with the following values:

Variable	Value
p	NULL (empty)
x	num [1:100] 2.46 0.287 -0.193 1.568 0.914 ...
y	num [1:100] 0.543 0.696 2.439 2.238 3 ...
- Console:** Shows the output of the script execution:

```
Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

>
>
> source('~/.active-rstudio-document')
NULL
> source('~/.active-rstudio-document')
NULL
```
- Plots Pane:** Displays a scatter plot of the generated data. The x-axis is labeled 'x' and ranges from -1 to 4. The y-axis is labeled 'y' and ranges from -1 to 3. The plot shows a random distribution of points.

An interactive and easy-to-use interface with many features that make working with R easier

RStudio Interface

The screenshot displays the RStudio interface with the following components:

- Script Editor:** Contains the following R code:

```
1 # Write your script here
2
3 x = rnorm(100, 1, 1)
4 y = rnorm(100, 1, 1)
5
6 p = plot(x, y)
7
8 print(p)
```
- Environment Pane:** Shows the Global Environment with the following values:

Variable	Value
p	NULL (empty)
x	num [1:100] 2.46 0.287 -0.193 1.568 0.914 ...
y	num [1:100] 0.543 0.696 2.439 2.238 3 ...
- Console:** Shows the output of the script execution, including a warning about natural language support and the execution of the `source()` command:

```
Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]
>
>
> source('~/.active-rstudio-document')
NULL
> source('~/.active-rstudio-document')
NULL
>
```
- Plots Pane:** Displays a scatter plot of the generated data, with the x-axis labeled 'x' and the y-axis labeled 'y'. Both axes range from approximately -1 to 4.

Console for entering R
commands

RStudio Interface

Script, data tables, etc...

The screenshot displays the RStudio interface with the following components:

- Script Editor (my_script.R):** Contains the following R code:

```
1 # Write your script here
2
3 x = rnorm(100, 1, 1)
4 y = rnorm(100, 1, 1)
5
6 p = plot(x, y)
7
8 print(p)
```
- Console:** Shows the R startup message and the execution of the script:

```
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
>
>
> source('~/.active-rstudio-document')
NULL
> source('~/.active-rstudio-document')
NULL
>
```
- Environment Pane:** Shows the current environment with the following values:

Variable	Value
p	NULL (empty)
x	num [1:100] 2.46 0.287 -0.193 1.568 0.914 ...
y	num [1:100] 0.543 0.696 2.439 2.238 3 ...
- Plots Pane:** Displays a scatter plot of the generated data. The x-axis is labeled 'x' and ranges from -1 to 4. The y-axis is labeled 'y' and ranges from -1 to 3. The plot shows a random distribution of approximately 100 data points.

RStudio Interface

Clickable list of
variables in memory

The screenshot displays the RStudio interface with the following components:

- Script Editor:** Contains the following R code:

```
1 # Write your script here
2
3 x = rnorm(100, 1, 1)
4 y = rnorm(100, 1, 1)
5
6 p = plot(x, y)
7
8 print(p)
```
- Environment Pane (highlighted in yellow):** Shows the Global Environment with the following variables:

Variable	Value
p	NULL (empty)
x	num [1:100] 2.46 0.287 -0.193 1.568 0.914 ...
y	num [1:100] 0.543 0.696 2.439 2.238 3 ...
- Console:** Shows the output of the script execution, including the R startup message and the execution of the script:

```
Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> source("~/..active-rstudio-document")
NULL
> source("~/..active-rstudio-document")
NULL
```
- Plots Pane:** Displays a scatter plot of x vs y, showing a positive correlation between the two variables.

RStudio Interface

The screenshot displays the RStudio interface with the following components:

- Script Editor:** Contains the following R code:

```
1 # Write your script here
2
3 x = rnorm(100, 1, 1)
4 y = rnorm(100, 1, 1)
5
6 p = plot(x, y)
7
8 print(p)
```
- Environment Pane:** Shows the current environment with the following values:

Variable	Value
p	NULL (empty)
x	num [1:100] 2.46 0.287 -0.193 1.568 0.914 ...
y	num [1:100] 0.543 0.696 2.439 2.238 3 ...
- Console:** Shows the output of the script execution:

```
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]
>
>
>
> source('~/.active-rstudio-document')
NULL
> source('~/.active-rstudio-document')
NULL
>
```
- Plot Window:** A scatter plot titled 'my_script.R' showing a random distribution of points. The x-axis is labeled 'x' and ranges from -1 to 4. The y-axis is labeled 'y' and ranges from -1 to 3. The plot contains approximately 100 data points.

Plots, Files,
Packages,
help etc...

RStudio on Windows

The screenshot displays the RStudio interface with four main panes:

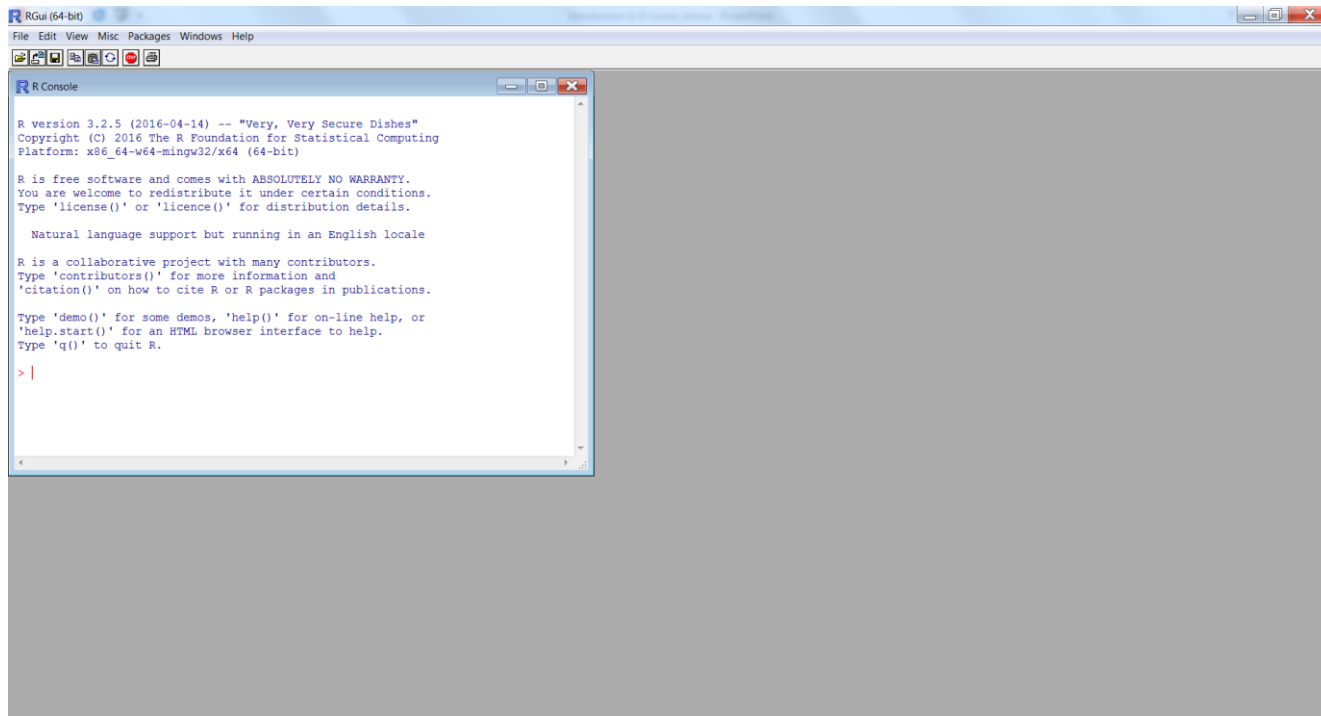
- Source Editor:** Contains R code for creating vectors and plotting a sine wave. The code is as follows:

```
1 x <- 5:20
2
3 1+2
4
5 x <- c(0,2,-1,pi,10)
6 x
7 x <- c(0:5,10:15)
8 x
9
10 y <- c(1, 2, 3, 4, 5)
11 x ~ y
12
```
- Environment:** Shows the Global Environment with the following values:

Variable	Value
bmi	24.5
v	int [1:101] -50 -49 -48 -47 -46 -45 -44 -43 -42 -41 ...
x	num [1:101] -6.28 -6.16 -6.03 -5.91 -5.78 ...
x2	num [1:12] 0 1 4 9 16 25 100 121 144 169 ...
y	num [1:101] 2.45e-16 1.25e-01 2.49e-01 3.68e-01 4.82e-01 ...
z	0.333333333333333
- Console:** Shows the execution of the code and the output of the `plot` function. The output is a vector of 101 values:

```
> par(mfrow=c(1,2))
> plot(x,y)
> plot(x,y, las=1, xlab="x", ylab="sin(x)", type="l", main="Trigonometri
c functions")
>
> help.start()
If nothing happens, you should open
'http://127.0.0.1:21471/doc/html/index.html' yourself
> help("plot")
?plot
> help("plot")
> v <- -50:50
> v
[1] -50 -49 -48 -47 -46 -45 -44 -43 -42 -41 -40 -39 -38 -37 -36 -35
[17] -34 -33 -32 -31 -30 -29 -28 -27 -26 -25 -24 -23 -22 -21 -20 -19
[33] -18 -17 -16 -15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3
[49] -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13
[65] 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
[81] 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
[97] 46 47 48 49 50
> x <- seq(-2*pi,2*pi,length=101)
> y <- sin(x)
> plot(x,y)
> plot(x,y, las=1, xlab="x", ylab="sin(x)", type="l", main="Trigonometri
c functions")
> help("plot")
```
- Documentation:** Shows the R documentation for `plot` (graphics), titled "Generic X-Y Plotting". The description states: "Generic function for plotting of R objects. For more details about the graphical parameter arguments, see [par](#)." It also mentions that for simple scatter plots, `plot.default` will be used, and that there are `plot` methods for many R objects including `function`, `data.frame`, and `density` objects. The usage is `plot(x, y, ...)`. The arguments section lists `x` (coordinates of points), `y` (y coordinates), and `...` (arguments to be passed to methods). The `type` argument is described as "what type of plot should be drawn. Possible types are" with a list:
 - "p" for points,
 - "l" for lines,
 - "b" for both,

The R interface (Windows)



- From here we can start working in R and do things like:
 - Check what packages are already installed
 - Install new packages
 - Check the current working directory or change to a new one
 - Access the help options
- As always, there are several ways to do things