

MAS115: Mathematical Investigation Skills Semester 2

Paul Blackwell

The University of Sheffield
School of Mathematics and Statistics

The course this semester

Programming

- ▶ R as a programming language
- ▶ pseudocode for planning and structuring your coding

Presentation

- ▶ markdown as a tool for combining maths and code

Mathematics - some new ideas and types of problem

What is R?

You've seen some R before

- ▶ Tool to permit data entry and manipulation
- ▶ Lots of built in functions to do statistical analyses:
 - ▶ Means and variances
 - ▶ Fitting linear models
 - ▶ More complex methods
- ▶ Powerful functions for producing graphics

What is R?

You've seen some R before

- ▶ Tool to permit data entry and manipulation
- ▶ Lots of built in functions to do statistical analyses:
 - ▶ Means and variances
 - ▶ Fitting linear models
 - ▶ More complex methods
- ▶ Powerful functions for producing graphics

but that's not all it does

- ▶ R is a programming language

Programming this term

So far we've been introduced to Python but

- ▶ Only knowing a single specific language isn't always that useful
- ▶ We'd like to be able to code in many languages
- ▶ We aim to show that many programming ideas are universal
- ▶ You can transfer things you learnt in Python to pick up any new language easily

We will do this in two ways

Programming this term II

Lectures:

- ▶ Consider **pseudocode** as recipe for algorithms
- ▶ Not a specific language but creates step-by-step instructions for implementation
- ▶ Can then take and implement in any language
- ▶ Don't need to worry about language specificity **but** must still be logical
- ▶ Helps to think about process **before** starting the coding

Programming this term III

Lab classes:

- ▶ Specifically see how Python ideas can transfer to R easily
- ▶ Implement some of our pseudocode
- ▶ You will learn a new language with much less effort
- ▶ Obtain similar proficiency to Python in half the time

Final aim: Feel confident can learn other languages too

Presentation

R markdown

- ▶ Enables combining maths, code and results in one document
- ▶ Almost no new syntax necessary
- ▶ Exploits strength of \LaTeX
- ▶ Ideal for homework write-ups

The principles of programming

All programming based on similar concepts:

- ▶ Splitting up big problems into smaller ones and using logic
- ▶ Once you learn one language you can then transfer these easily
- ▶ The aim of this course is to show that

Why Python then R?

- ▶ Last term learnt Python
- ▶ Good for learning basic principles and building functions from scratch
- ▶ R slightly higher level - lots of built in functions
- ▶ R also has some special features

Still need to be careful: learning the syntax

While the basic ideas of programming is the same each language has some of its own intricacies that you need to be aware of when programming.

- ▶ Assignment operator (**Python** $x = 0$ but **R** $x \leftarrow 0$)
- ▶ Numbering of elements in vectors e.g. $\mathbf{x} = (x[1], x[2], \dots)$
- ▶ Default data types - double or integer and applying operators e.g. $2/3 = ?$
- ▶ Vectorising operations or working elementwise? $a + b$ or $a[i] + b[i]$
- ▶ Quantity (and quality) of built-in functions to use

What this course entails

- ▶ This course is **not** going to teach statistical analyses using R
- ▶ Concentrate entirely on its use as programming language
- ▶ Show how ideas learnt last semester can be transferred across simply
- ▶ As you know ideas already, we're going to move faster and do more involved programming

Lectures and Assessment

Lectures (6 lectures — 1 per week)

R programming

- ▶ Some key ideas in R
- ▶ R markdown

Pseudocode

- ▶ Semi-formal way to describe algorithms
- ▶ Not language specific and **should** be understandable by anyone
- ▶ Step-by-step instruction can be transferred to any language

Mathematics e.g.

- ▶ Monte-Carlo estimation of integrals
- ▶ Newton-Raphson (iterative solution of equations)
- ▶ Finding primes (Sieve of Eratosthenes)

Labs (6 classes — 1 per week)

Practicals

- ▶ 1 computer practical per week
- ▶ Transfer Python concepts and language to R
- ▶ Implement pseudocode

These aim to implement ideas. **This is the only way to learn.** Make sure you do all questions on problem sheet

Any parts of lab sheet not completed in class should be done in external study

Assessment I

Homework (10% of final grade):

- ▶ Each lab class will have a homework
- ▶ Write up using Rmarkdown
- ▶ Submit in hard copy at following week's class

Programming Test (10% of final grade):

- ▶ Interpret pseudocode and R
- ▶ Identify logical errors in algorithms/code
- ▶ Write pseudocode (and perhaps very simple R)

Assessment II

Individual mini-project (10% of final grade):

- ▶ R programming task
- ▶ Similar format to last term
- ▶ Released in time to be completed over Easter

Group project (30% of final grade):

- ▶ Webpage — 20%
- ▶ Presentation — 10%
- ▶ Programming should be in R
- ▶ Released in week 6
- ▶ Completed after Easter

An R session - fundamentals

Rstudio

- ▶ We will use RStudio for our programming
- ▶ Integrated front-end where we can have code, terminal and other information all viewable together
- ▶ Autocompletes brackets in script window to reduce errors
- ▶ Comments, special commands highlighted for easy viewing
- ▶ Free and open source
- ▶ Works on Windows, Mac and Linux

Fundamentals - Opening a session

- ▶ Need to have an organised system on your computer to store projects
- ▶ R can do this by creating a workspace where all output is stored
- ▶ We recommend that for each major project you use a separate workspace

Fundamentals - During a session

Writing code:

- ▶ Do not write directly to the command window
- ▶ Create all code used in a source/file window
- ▶ This allows you to fix typos, change parameters, ...
- ▶ A script file consists of R code & comments
- ▶ Use sensible file names to help find each analysis
- ▶ Use clear variable names to understand what is stored

Commenting

- ▶ You **must** document your code
- ▶ In an R script, you use the `#` character for commenting...
- ▶ ...or use Rmarkdown format

Fundamentals - During a session

Getting help:

Fundamentals - During a session

Getting help:

- ▶ If you know the function name use
`?lm` or `help(lm)`

Fundamentals - During a session

Getting help:

- ▶ If you know the function name use

`?lm` or `help(lm)`

- ▶ If just want to find out about a subject try

`??"linear models"` or `help.search("linear models")`

Fundamentals - During a session

Getting help:

- ▶ If you know the function name use

`?lm` or `help(lm)`

- ▶ If just want to find out about a subject try

`??"linear models"` or `help.search("linear models")`

Fundamentals - During a session

Getting help:

- ▶ If you know the function name use
`?lm` or `help(lm)`
- ▶ If just want to find out about a subject try
`??"linear models"` or `help.search("linear models")`

Closing the session:

Fundamentals - During a session

Getting help:

- ▶ If you know the function name use
`?lm` or `help(lm)`
- ▶ If just want to find out about a subject try
`??"linear models"` or `help.search("linear models")`

Closing the session:

- ▶ R does allow you to save the workspace as a `.RData` file

Fundamentals - During a session

Getting help:

- ▶ If you know the function name use
`?lm` or `help(lm)`
- ▶ If just want to find out about a subject try
`??"linear models"` or `help.search("linear models")`

Closing the session:

- ▶ R does allow you to save the workspace as a `.RData` file
- ▶ Not necessarily a good idea

Fundamentals - During a session

Getting help:

- ▶ If you know the function name use
`?lm` or `help(lm)`
- ▶ If just want to find out about a subject try
`??"linear models"` or `help.search("linear models")`

Closing the session:

- ▶ R does allow you to save the workspace as a `.RData` file
- ▶ Not necessarily a good idea
- ▶ Just save the R commands and perhaps the objects

Rmarkdown - \LaTeX emphasis

```
---  
title: "MAS115: minimal R markdown document"  
author: PGB  
date: 'r format(Sys.time())'  
numbersections: TRUE  
output: pdf_document  
---
```

The header is generated by a simple block at the start of t

```
\section{Coding}
```

R code is generally included as `\emph{chunks}`, shown in the

```
''' {r random_normal}  
set.seed(1)  
x <- rnorm(200)  
summary(x)  
'''
```

Rmarkdown - markdown emphasis

```
---  
title: "MAS115: minimal R markdown document"  
author: PGB  
date: 'r format(Sys.time())'  
numbersections: TRUE  
output: pdf_document  
---
```

The header is generated by a simple block at the start of t

Coding

R code is generally included as **chunks**, shown in the fina

```
``` {r random_normal}  
set.seed(1)
x <- rnorm(200)
summary(x)
```
```

Special Values

There are some special values in R, often treated differently in calculations—not character strings but values in themselves.

- ▶ NA - If a particular value is known to be missing (e.g. in a statistical study) then we would denote that value by NA. Some functions can ignore these missing values e.g. `mean(x, na.rm = TRUE)`. The value NA can arise in a calculation too.
- ▶ Inf - The result of dividing a non-zero number by zero e.g. $1/0$, or similar.
- ▶ NaN - An undefined value e.g. $\log(-10)$ or $0/0$.
- ▶ NULL - More esoteric, but has programming uses.

Unexpected appearances often suggest something might be wrong in your code—R often tries to give warnings. Functions `is.na`, `is.finite`, `is.infinite`, `is.nan`, `is.null` exist.

Summary

After the lab class you should have understanding of:

- ▶ Course structure
- ▶ How to organise R workspace and use a script file
- ▶ How to create a basic Rmarkdown document
- ▶ Basic data modes in R
- ▶ Vectors and operations