MAS115: Mathematical Investigation Skills
Semester 2

February 18, 2021

The course this semester

Programming

- R as a programming language
- Pseudocode as a tool for planning and structuring your coding

Presentation

- Markdown as a tool for combining written 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 you'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: to feel confident that you 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

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 efficiently (Sieve of Eratosthenes), complexity in general

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 (5% of final grade):

- Each lab class will have a homework
- Write up using Rmarkdown
- Submit an electronic copy to blackboard the following Monday

Class Test (10% of final grade):

- Interpret code and pseudocode
- Test material from lectures
- Identify logical errors in algorithms/code
- Write pseudocode (and perhaps simple code)

Assessment II

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

- R programming task
- Similar format to last term
- Released in Week 5
- Deadline in Week 7

Group project (25% of final grade):

- Webpage — 15%
- Presentation — 10%
- Released Week 7
- Project deadline in Week 10
- Presentations in Week 11

The R language

While the basic ideas of programming are the same, each language has some of its own intricacies.
Details of R will be covered mainly in lab classes; here are some thing to watch out for.

- Assignment operator (Python `x = 0` but R `x <- 0`)
- Default structure in R is a vector, more like a NumPy array than a list in Python
- Numbering of elements in vectors (and lists) starts at 1, not 0
- Default data mode is numeric, either double or integer; modes logical, complex and character also exist.
- Many operations on vectors work elementwise by default, like in NumPy
- Indentation doesn't matter (to the computer); structure relies on {,}

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. Note that `-Inf` also exists.
- `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.

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:

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:

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 - header

```
---
title: "MAS115: minimal R markdown document"
author: PGB
date: ’‘r format(Sys.time())‘’
numbersections: TRUE
output: pdf_document
---
\renewcommand{\thesection}{Q\arabic{section}\!\!}
\renewcommand{\thesubsection}{\alph{subsection})\!\!}
\renewcommand{\thesubsubsection}{\roman{subsubsection})}
\makeatletter
\renewcommand{\subsection}{\@startsection{subsection}{2}{\z@}%
  {1.75ex \@plus 1ex \@minus .2ex}{-0.01em}{\normalfont\normalsi
\makeatother
```

Rmarkdown - maths

```
# Mathematics
Maths can be produced using many \LaTeX\ commands,
without any special flagging, for example $A = \pi r^2$.
That includes displayed equations, such as
\[
\exp(i\pi) + 1 = 0.
\]
If necessary \verb"\verb" can be used to show \LaTeX\
commands (or anything else) verbatim, as above.
```

Rmarkdown - code

````
# Coding
The main reason to use Rmarkdown is to combine code and
the output it produces.

R code is generally included as *chunks*, shown in the
final document as highlighted code followed by the
output produced, though either can be suppressed.

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

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