DR LEE FAWCETT

MAS1802 PROBLEM SOLVING II: COMPUTING WITH R

NEWCASTLE UNIVERSITY

Useful calendar:

Week beginning...	**Mon**	**Tues**	**Wed**	**Fri**
5th March	Intro lecture 5-6 *(Herschel LT1)*	Intro Practical 3-4 *(Herschel PC/ Armstrong 2.96)*	Lecture 1 11-12 *(Herschel LT1)*	Practical 1 11-1 *(Herschel PC/ Bedson Side PC)*
12th March	Practical 1 DI 12-1 *(Herschel PC)*	✘	Lecture 2 11-12 *(Herschel LT1)*	**Practical 1 deadline: 11:00** Practical 2 11-1 *(Herschel PC/ Bedson Side PC)*

Week beginning...	**Mon**	**Tues**	**Wed**	**Fri**
16th April	Practical 2 DI 12-1 *(Herschel PC)*	✘	Lecture 3 11-12 *(Herschel LT1)*	**Practical 2 deadline: 11:00** Practical 3 11-1 *(Herschel PC/ Bedson Side PC)*
23rd April	Practical 3 DI 12-1 *(Herschel PC)*	✘	Lecture 4 11-12 *(Herschel LT1)*	**Practical 3 deadline: 11:00** Project 11-1 *(Herschel PC/ Bedson Side PC)*
30th April	Project DI 12-1 *(Herschel PC)*	✘	Lecture 5 11-12 *(Herschel LT1)*	Project 11-1 *(Herschel PC/ Bedson Side PC)*
7th May	Bank holiday! ✘	Project DI 3-4 *(Herschel PC)*	Lecture 6 11-12 *(Herschel LT1)*	Project 11-1 *(Herschel PC/ Bedson Side PC)*
14th May	Project DI 12-1 *(Herschel PC)*	✘	✘	**Project deadline: 11:00**

Contents

1 *Introduction and Housekeeping* 5

2 *Introduction to R* 10

3 *Data Summaries* 19

4 *Control Statements and Functions* 27

5 *Random Number Generation* 35

6 *Monte Carlo Methods* 43

1
Introduction and Housekeeping

1.1 Housekeeping

The computing part of MAS1802 will run during the second half of semester 2 (unfortunately broken up by the Easter vacation!). In direct response to your feedback on the `MatLab` part of MAS1801, we will have weekly lectures, with printed notes, to support the computer practical sessions. Generally, a new topic will be introduced each week and will fit around the following schedule[1]:

Type of session	Day	Time	Place
Lecture	Wednesday	11–12	Herschel LT1
Office hour	Thursday	12–1	My office (Herschel 2.07)
Practical	Friday	11–1	Herschel PC/Bedson Side PC
Drop-in	Monday	12–1	Herschel PC
Deadline:	*Friday*	*11am*	*General office/online*

[1] To kick-start things, we will have two lectures in the first week, with an additional lecture being held on Monday 5pm. The very first practical session takes place before the first assignment has been set, on Tuesday 3pm, and will be an important introductory practical to R.

1.1.1 Lecturer information

The module leader, and lecturer for the computing part of MAS1802, is Dr. Lee Fawcett. If you have any questions, comments or feedback on this course I can be contacted via email at `lee.fawcett@ncl.ac.uk`. More complicated questions about the course are best dealt with face-to-face.

1.1.2 Lectures

Given the nature of this course we will have fewer lectures than the other first year modules – as you might imagine, there will be a strong emphasis on practical skills in the computer cluster. I will hand out copies of the lecture notes and practical worksheets in class.

Slides will be used in lectures to accompany the notes. Lectures will also be recorded using the ReCap system. You can access all course materials, including the ReCap recordings, through Blackboard (BB) or via the dedicated course webpage:

http://www.mas.ncl.ac.uk/~nlf8/teaching/mas1802/

1.1.3 Computing practicals

During each computer practical, your general computer skills will be assessed. You are more likely to get high marks for the coursework if:

- You have attended the lecture;
- You don't panic, but read through the question, and the notes, carefully;
- You ask for help[2];
- You are organised! For example, if:
  - your files are organised in a sensible manner – i.e. if you have a directory for MAS1802 and sub-directories for each practical;
  - your files are given convenient and meaningful names;
  - Your code "looks nice".

We won't try to catch you out!

[2] As in MAS1801, there will be lots of help available in computer practicals: Dr. Fawcett, Dr. Cowburn, Dr. Henderson and Dr. Newman will all be on-hand, as will several Ph.D students:

In this course, most of the computer practical sessions are held in the Herschel PC cluster[3], and all machines in the cluster have the necessary software and packages installed. However, you can use computers in other parts of the University while doing your coursework. A list of free computers can be found at:

`http:m.ncl.ac.uk/itservice/#clusfree`

If you use a machine in a cluster and it is missing the necessary software, contact Christian Lawson-Perfect: `christian.perfect@ncl.ac.uk`.

[3] Although, due to the size of the class, some of you will be in a smaller satellite cluster for the main Friday practical

1.1.4 Office hours

The scheduled office hour for this module is Thursday 12–1pm; I will always be free at this time, in my office, to see students who need extra help with the work. I also have office hours for other courses which are usually quiet (see noticeboard outside my office), so feel free to drop in then too. In general, I am happy to see students anytime (if I'm free), except on Wednesdays and Friday after 2pm.

1.1.5 Assessment

- End of semester examination worth **40%** of your overall mark. This will be based on material from the pencil-and-paper part of the course with Evgenios.
- Assessment of coursework worth **60%** of your overall mark:
  - Weekly computing exercises with Lee (10%)
  - Group computing project with Lee (30%)
  - Problem-solving (pencil-and-paper) exercises with Evgenios (20%)

The following deadlines relate to the courswork for the computing part of this module:

- Practical 1: Friday 16th March, 11:00am (online submission)

- Practical 2: Friday 20th April, 11:00am (hard-copy submission to the General Office)
- Practical 3: Friday 27th April, 11:00am (online submission)
- Group computing project: Friday 18th May, 11:00am (hard-copy submission to the General Office)

For the group work you will be in the same groups as you were allocated in the problem solving part of the course[4].

[4] Although I might change my mind on this one...

1.1.6 Quiz questions

Last summer, we wrote lots of quiz questions in `Numbas` to help develop your computing skills for this part of the module. These quiz questions are written within the usual computer-based assessment (CBA) framework – if you're interested, go to:

`www.mas.ncl.ac.uk/~nlf8/basicR`

1.1.7 Late work policy

It is not possible to extend submission deadlines for coursework in this module and no late work can be accepted. For details of the policy (including procedures in the event of illness etc.) please look at the School web site

`http://www.ncl.ac.uk/maths-physics/students/resources`

1.2 What is R?

R is a computer package that is widely used for statistical software development and data analysis[5]. R uses a command line interface, though several graphical user interfaces are available. The system provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, and others) and graphical techniques.

[5] For example Google, IBM, Shell, Thomas Cook, Facebook. The recent NY times article at `http://tinyurl.com/9m5qyh` is worth a read.

R is highly extensible through the use of user-submitted libraries for specific functions or specific areas of study. A particular strength of R is its graphical facilities, which produce quality graphs (better than MatLab!) that can include mathematical symbols. Although R is mostly used by statisticians and other practitioners requiring an environment for statistical computation and software development, it can also be used as a general matrix calculation toolbox with comparable benchmark results to many other software packages (e.g. MatLab!). Why is it called R? There are a few reasons, one of which is that it's named after the two people who developed the software: Ross Ihaka and Robert Gentleman, both from Auckland University in New Zealand.

You will use R throughout your degree at Newcastle and the best part is, it's **completely free to download to your own computer!**

				Voting statistics					Movie genre							
Title	Year	Length	Budget	Rating	Votes	r1	...	r10	mpaa	Action	Animation	Comedy	Drama	Documentary	Romance	Short
A.k.a. Cassius	1970	85	NA	5.7	43	4.5	...	14.5	PG	0	0	0	0	1	0	0
AKA	2002	123	NA	6.0	335	24.5	...	1.5	R	0	0	0	1	0	0	0
Alien Vs. Pred	2004	102	45000000	5.4	14651	4.5	...	4.5	PG-13	1	0	0	0	0	0	0
Abandon	2002	99	25000000	4.7	2364	4.5	...	4.5	PG-13	0	0	0	1	0	0	0
Abendland	1999	146	NA	5.0	46	14.5	...	24.5	R	0	0	0	0	0	0	0
Aberration	1997	93	NA	4.8	149	14.5	...	4.5	R	0	0	0	0	0	0	0
Abilene	1999	104	NA	4.9	42	0.0	...	24.5	PG	0	0	0	1	0	0	0
Ablaze	2001	97	NA	3.6	98	24.5	...	14.5	R	1	0	0	1	0	0	0
Abominable Dr	1971	94	NA	6.7	1547	4.5	...	14.5	PG-13	0	0	0	0	0	0	0
About Adam	2000	105	NA	6.4	1303	4.5	...	4.5	R	0	0	1	0	0	1	0

Table 1.1: The first ten rows of the movie data set. **Credit:** This data set was initially constructed by Hadley Wickham at http://had.co.nz/.

1.2.1 *Accessing R*

R is installed on all University machines. In this class we will be using Rstudio, which is an R IDE[6]. Both R and Rstudio are free, so you can install them on you own computer[7]. See

http://www.ncl.ac.uk/maths-physics/students/resources/

for more details.

[6] Integrated development environment (IDE).

[7] Versions of R and Rstudio are available for Windows, Apple Mac and Linux.

1.2.2 *Previous computing knowledge*

This course is intended to teach you the basics of programming. **No** previous programming knowledge is assumed, although the `MatLab` part of MAS1801 should have helped your general skills. It is crucial that you come to all the practicals for this course; it is also vital that you let me/other staff know, in the practical sessions, if you're not sure about anything. Burying your head in the sand will not help. Neither will crying, or getting angry[8].

[8] Please don't do this.

1.2.3 *Recommended R textbooks*

On the course webpage I have a list of suitable books on R programming. Since you will be using R throughout your degree, then it may be worthwhile buying a good R textbook. A really good, free-to-download book written by Venables and Smith, can be found here:

https://cran.r-project.org/doc/manuals/R-intro.pdf

1.3 *Movies dataset*

The internet movie database, (IMDB) http://imdb.com/, is a website devoted to collecting movie data supplied by studios and fans. It claims to be the biggest movie database on the web and is run by *Amazon*. More information `imdb.com` can be found online:

http://imdb.com/help/show_leaf?about

including information about the data collection process

http://imdb.com/help/show_leaf?infosource

IMDB makes their raw data available at http://uk.imdb.com/interfaces/.

Movies are selected for inclusion if they had a known length, had been rated by at least one IMDB user and had an MPAA (*Motion Picture Assocation of America*) rating. Amongst other things, the dataset contains the following fields:

- Title: Title of the movie.
- Year: Year of release.
- Budget: Total budget in US dollars. If the budget isn't known, then it is stored as 'NA'.
- Length: Length in minutes.
- Rating: Average IMDB user rating.
- Votes: Number of IMDB users who rated this movie.
- r1-10: Percentage(to nearest 5%) of users who rated this movie 1, ..., 10
- mpaa: MPAA rating.
- Genre: Action, Animation, Comedy, Drama, Documentary, Romance, Short. Binary variables representing if movie was classified as belonging to that genre. A movie can belong to more one genre. See for example the film *Ablaze* in Table 1.1.

This is **only a subset** of the data, the actual data set contains information on over 50,000 movies.

There are a total of 23 variables and 4988 films. The first few rows are given in Table 1.1. We will use this dataset to illustrate the concepts covered in this class.

1.4 Useful R package

We have written an add-on R package for the module which will allow easy access to many interesting datasets (including the movies dataset). Installing this package is straightforward, and will make loading datasets used in lectures really easy:

```
> install.packages("ncldata",
+                  repos="http://R-Forge.R-project.org",
+                  type="source")
```

When typing in code from the lecture notes, **never** include the + symbols (unless they're actually part of a calculation!). This just indicates, in the notes, that the code has 'wrapped' onto another line.

To load the package, use

```
> library(ncldata)
```

Then to load the movies dataset, for example, type:

```
> data(movies)
```

Alternatively, datasets can be downloaded directly from the course webpage/BB page (see practical work this/next week). We will explore the movies dataset in Chapter 2.

2
Introduction to R

In this chapter we will play about with R and learn about the basics.

2.1 A simple R session

At its most basic R, can be used as a calculator, for example for multiplication and division:[1]

```
> 5*5
[1] 25
> 10.2/6
[1] 1.7
```

And more 'advanced' operations:

```
> 2^3
[1] 8
> exp(1.5)
[1] 4.481689
> log(10)
[1] 2.302585
> 4 %% 3
[1] 1
```

[1] The # symbol is used for commenting. We use comments to describe what a piece of code is doing. That way, when we look at the code in a few months/years we can figure out what is going on... For example:

```
> #Multiplication
> 5*5
[1] 25

> #Logarithms
> log(10)
[1] 2.302585
```

2.1.1 Assignment operations

In the practicals we will use assignment, i.e. `x = 5`. However, computer assignment is different from typical mathematical assignment. For example:

```
> x = 5
> x = x + 1
> x
[1] 6
```

In mathematics, $x = x + 1$ implies that $0 = 1$.

Notice that when we type `x = 5`, R doesn't display or print any output to the screen.[2] If we want to see what value has been assigned to the variable, we type `x`. An equivalent way is to surround the expression with brackets.[3] For example:

```
> (x = 2*x)
[1] 12
```

[2] Don't confuse this with R not "doing anything."

[3] In these notes, I will surround expressions with brackets so you can see what R has done.

2.1.2 Data types

R has a variety of data types:

```
> (v = TRUE)
[1] TRUE
> (w = "fred")
[1] "fred"
> (x = 5.0)
[1] 5
```

and also some "special" data types:

```
> (y = 5/0)
[1] Inf
> (z = y-y)
[1] NaN
```

Another important data type in R is `NA`. This is used to represent missing values. A list of data tables is given in Table 2.1.

Type	Example 1	Example 2	Example 3	Example 4
Doubles	2	3.1242	-45.6	4e-10
Logicals/Boolean	TRUE	FALSE		
Characters	"FRED"	"x"	"Male"	"TRUE"
Infinity	Inf	5/0		

Table 2.1: Summary of some data types in R.

2.2 The R workspace

Once you create a variable, R stores that variable in memory for reuse. You can view available variables with the `ls()` command:

```
> rm(list = ls())
> library(ncldata)
> data(movies)
```

```
> ls()
[1] "movies"
```

To delete a variable in R, we use the `rm` function. For example:

```
> x = 0
> y = 1
> z = 2
> ls()
[1] "movies" "x"      "y"      "z"
> rm(x)
> ls()
[1] "movies" "y"      "z"
```

We can remove everything in the workspace using `rm(list=ls())`:

```
> rm(list=ls())
```

I would recommend running `rm(list=ls())` at the beginning of each new R session. This stops you relying on previously stored variables and makes your code more portable.

2.3 Vectors

Vectors are the most basic of all data structures, but are used in almost all R code. An R vector contains n values of the same type, where n can be zero. For example:

```
> c(0, 1, 2, 3, 4, 5)
[1] 0 1 2 3 4 5
> (my_first_vec = c(0, 1, 2, 3, 4, 5))
[1] 0 1 2 3 4 5
> (my_second_vec = c("Male", "Female", "Male"))
[1] "Male"   "Female" "Male"
```

In the above code, we create a vector of **doubles**; in line 2 we then assign the vector to the variable `my_first_vec`. We can create vectors of any data type. For example, `my_second_vec` is a vector of **characters**. In R, when we type:

```
> x = 5
> y = "Fred"
```

we have actually created a vector of doubles and characters (of length one). There are special functions in R to determine variable type:

```
> x = 5
> is.double(x)
[1] TRUE
> is.character(x)
[1] FALSE
> is.vector(x)
[1] TRUE
```

To determine the length of a vector in R, we use the `length` function:

```
> length(my_first_vec)
[1] 6
> length(my_second_vec)
[1] 3
```

To create sequences of numbers we use the `seq` command. For example:

```
> (x1 = seq(1, 6))
[1] 1 2 3 4 5 6
> (x2 = seq(-4, 4, by=2))
[1] -4 -2  0  2  4
```

Table 2.2 gives a few more useful R functions.

Command description	Example	Result
Length	`length(x)`	4
Reverse order	`rev(x)`	3,5,5,1
Sort	`sort(x)`	1,3,5,5
Sum	`sum(x)`	14
Extract unique elements	`unique(x)`	1,5,3
Indices of particular elements	`which(x==5)`	2,3

Table 2.2: Useful vector functions. In the above examples, `x = c(1,5,5,3)`. Check the associated R help for further information.

2.3.1 Vector operations

When our data is in a vector structure, we can apply standard operations to the entire vector. For example:

```
> (x = seq(-4, 4))
[1] -4 -3 -2 -1  0  1  2  3  4
> x*x
[1] 16  9  4  1  0  1  4  9 16
> x - 5
[1] -9 -8 -7 -6 -5 -4 -3 -2 -1
> x + x
[1] -8 -6 -4 -2  0  2  4  6  8
```

2.3.2 Extracting elements from vectors

R has a number of useful of methods that we can use to extract subsets of our data. For example, to pick out particular elements:

```
> my_first_vec[2]
[1] 1
> my_second_vec[2:3]
[1] "Female" "Male"
> my_first_vec[4:2]
[1] 3 2 1
```

We can also use other arguments. For example, to remove the last entry in the vector, we use the `length` function:

```
> l = length(my_first_vec)
> my_first_vec[1:(l-1)]
[1] 0 1 2 3 4
```

We determine the length of the vector using the `length` function and select particular elements using the `[·]` operator.

2.4 Logical vectors

R supports the logical elements: `TRUE` and `FALSE`. Boolean algebra tells us how to evaluate the truth of compound statements. Table 2.3 gives a summary of R operations and compares them to the notation used in MAS1604. So for example:[4]

```
> A = TRUE
> B = FALSE
> !A
[1] FALSE
> !B
[1] TRUE
> A & B
[1] FALSE
> A | B
[1] TRUE
```

[4] Read `!A` as **NOT** A.
Read `A & B` as A **AND** B.
Read `A | B` as A **OR** B.

Boolean	A	B	$\bar{A}$	$\bar{B}$	$A \cap B$	$A \cup B$
R	`A`	`B`	`!A`	`!B`	`A & B`	`A | B`
	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE
	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE
	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE
	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE

Table 2.3: Truth table for Boolean operations.

2.4.1 Using logicals for sub-setting vectors

We can construct vectors of logical operators and use them to take subsets of vectors. For example:

```
> (logic1 = c(TRUE, FALSE, TRUE, FALSE))
[1]  TRUE FALSE  TRUE FALSE
> (vec1 = seq(1, 4))
[1] 1 2 3 4
> vec1[logic1]
[1] 1 3
```

2.4.2 Relational operators

When programming it is often necessary to test relations for equality and inequality. To do this in R we use the relation operators. First let's define some variables:

```
> x = 5
> y = 7
```

To test for equality we use `==`:

```
> x == 5
[1] TRUE
> x == y
[1] FALSE
```

Similarly, to test for inequality we use `!=`:

```
> x != 5
[1] FALSE
> y != x
[1] TRUE
```

There are also commands for greater/less than:

```
> y > 6
[1] TRUE
> x >= 5
[1] TRUE
> x <= y
[1] TRUE
```

Table 2.4 gives a summary of the main commands.

WE CAN also apply these techniques to vectors. For example:

```
> (vec2 = seq(0, 10, by=2.5))
[1]  0.0  2.5  5.0  7.5 10.0
> vec2 > 3
[1] FALSE FALSE  TRUE  TRUE  TRUE
> vec2 < 9
[1]  TRUE  TRUE  TRUE  TRUE FALSE
> (vec2 > 3) & (vec2 < 9)
[1] FALSE FALSE  TRUE  TRUE FALSE
> !(vec2 > 3)
[1]  TRUE  TRUE FALSE FALSE FALSE
```

Table 2.4: Summary of R relational operators. The example is for `x = 5`.

Operator	Tests for	Example	Result
==	Equality	`x == 5`	`TRUE`
! =	Inequality	`x != 5`	`FALSE`
<	Less than	`x < 5`	`FALSE`
<=	Less or equal	`x <= 5`	`TRUE`
>	Greater	`x > 5`	`FALSE`
>=	Greater or equal	`x >= 5`	`TRUE`

Using relational operators allows us to extract subsets of data very easily. Consider the movie budgets:[5]

```
> length(Budget)
[1] 4988
```

To select movies where the budget is known, we use the following command:

```
> non_zero_b = Budget[!is.na(Budget)]
> length(non_zero_b)
[1] 3990
```

and to select movies where the movie length is greater than 60 minutes but shorter than 90 minutes:

```
> m_l = Length[Length > 60 & Length < 90]
```

[5] To extract certain variables from the movies dataframe, use the `attach` command: `library(ncldata); data(movies);attach(movies)`

2.5 Data frames

A data frame is a special kind of object. We use data frames for storing and managing data sets that have a rectangular structure. Typically the rows correspond to *cases* and the columns to *variables*. The crucial difference between a data frame and a `matrix` is that all values in a matrix must be of the same type. The next code segment constructs a simple data frame. First, we construct three vectors:

```
> age = c(24, 26, 25, 21)
> sex = c("Male", "Female", "Male", "Female")
> respond = c(TRUE, FALSE, FALSE, FALSE)
```

Then we combine them using the `data.frame` function:

```
> (df1 = data.frame(age=age, gender=sex, respond=respond))
  age gender respond
1  24   Male    TRUE
2  26 Female   FALSE
3  25   Male   FALSE
4  21 Female   FALSE
```

The data frame `df1` has three columns and four rows. Once we put our data into a data frame, then data manipulation is easier. To calculate the dimensions of a data frame we use `dim`:

```
> dim(df1)
[1] 4 3
```

To extract the first column we use square brackets:

```
> df1[ ,1]
[1] 24 26 25 21
```

Similarly, we can get the first row:

```
> df1[1, ]
  age gender respond
1  24   Male    TRUE
```

The column names are also easily manipulated:

```
> colnames(df1)
[1] "age"     "gender"  "respond"
> (colnames(df1) = c("Age", "Sex", "Respond"))
[1] "Age"     "Sex"     "Respond"
```

The `movies` dataset is actually a data frame[6]:

[6] Here, we're extracting the 3363rd row and columns 1 to 4

```
> dim(movies)
[1] 4988   23
> colnames(movies)[1:4]
[1] "Title"  "Year"   "Rating" "Votes"
> movies[3363, 1:4]
                                          Title Year Rating  Votes
3363 Star Wars: Episode I - The Phantom Menace 1999    6.5 607932
```

2.5.1 *Subsets of data frames*

We can also retrieve subsets from the data frame. For example, if we wanted only female responses, then:

```
> (female_only = df1$Sex=="Female")
[1] FALSE  TRUE FALSE  TRUE
> (df2 = df1[female_only, ])
  Age    Sex Respond
2  26 Female   FALSE
4  21 Female   FALSE
```

Or people 25 and over:

```
> over_25 = df1$Age>=25
> (df3 = df1[over_25, ])
  Age    Sex Respond
2  26 Female   FALSE
3  25   Male   FALSE
```

2.5.2 *Example: movie data*

We can select movies where the budget is greater than $100,000:

```
> m1 = movies[movies$Budget > 100000,]
> dim(m1)
[1] 4881   23
```

Or movies that cost more than $100,000 but are not R rated:

```
> m2 = movies[movies$Budget > 100000
+   & movies$mpaa != "R",]
> dim(m2)
[1] 3174   23
```

Or movies that are either PG or PG-13:

```
> m3 = movies[
+   movies$mpaa == "PG" | movies$mpaa == "PG-13",]
> dim(m3)
[1] 3229   23
```