

DR LEE FAWCETT

MAS1802
PROBLEM SOLVING II:
COMPUTING WITH R

NEWCASTLE UNIVERSITY

Timetable

This information refers to the computing part of MAS1802 only – in other words, the first six weeks of the semester. After the Easter vacation, pencil-and-paper problem-solving exercises will begin.

- Times:
 - Monday @ 10am to 11am: Herschel PC cluster
 - Monday @ 1pm to 2pm: Office hour
 - Thursday @ 9am to 10am: Herschel LT2
 - Friday @ 9am to 11am: Herschel PC cluster
- Attendance will be monitored in most sessions
- Any box with a **✕** indicates no class that day/no assignment work due in
- Feedback and quizzes: Thursday will not *always* be a full lecture. It will often include time for quizzes and feedback
- This timetable may change, so check your University email regularly

Practicals: Solutions to all practical work should be handed in by 4pm at the School Office, unless otherwise stated. You must use a NESS cover sheet and put your work in the post box. You can hand your work in early if you wish; however, I would strongly recommend that you hand them in at most 3 days early (otherwise they may get lost in the ‘system’). **Some work will be submitted electronically** – follow the instructions on the handouts carefully!

Class/Assignment Schedule

MONDAY	WEDNESDAY	THURSDAY	FRIDAY
Feb 1st 1 X	3rd 2 X	4th 3 Lecture: §1	5th 4 X
8th 5 Lab: Intro.	10th 6 X	11th 7 Lecture: §2	12th 8 Lab: Prac. 1
15th 9 Lab: Drop-in for Prac. 1	17th 10 <i>Hand-in: Prac. 1</i>	18th 11 Lecture: §3	19th 12 Lab: Prac. 2
22nd 13 Lab: Drop-in for Prac. 2	24th 14 <i>Hand-in: Prac. 2</i>	25th 15 Lecture: §4	26th 16 Lab: Prac. 3/ Start Project
29th 17 Lab: Drop-in Prac. 3/Proj.	Mar 2nd 18 X	3rd 19 Lecture: §5	4th 20 Lab: Prac. 3/ Project
7th 21 Lab: Drop-in Prac. 3/Proj.	9th 22 <i>Hand-in: Prac. 3</i>	10th 23 Lecture: §6	11th 24 Lab: Project

The Group project is due in by no later than 4pm, Monday 18th April (after Easter). Students should feel free to use the drop-in lab sessions to work on their project.

Contents

<i>1</i>	<i>Introduction and Housekeeping</i>	6
<i>2</i>	<i>Introduction to R</i>	11
<i>3</i>	<i>Data Summaries</i>	21
<i>4</i>	<i>Control Statements and Functions</i>	29
<i>5</i>	<i>Random Number Generation</i>	37
<i>6</i>	<i>Monte Carlo Methods</i>	45

1

Introduction and Housekeeping

1.1 Housekeeping

THE COMPUTING PART of MAS1802 will run during the first half of semester 2 (until the Easter vacation). In direct response to your feedback on the MatLab part of MAS1801¹, 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 schedule²:

Type of session	Day	Time	Place
Lecture	Thursday	9–10	Herschel LT2
Practical	Friday	9–11	Herschel PC
Drop-in	Monday	10–11	Herschel PC (Blue zone)
Office hour	Monday	1–2	My office (2.07)
<i>Deadline:</i>	<i>Wednesday</i>	<i>4pm/11.45pm</i>	<i>General office/online</i>

The pencil-and-paper problem-solving for this module will begin after Easter; however, some of the exercises for the computing part will also require you to problem-solve! *Note: these lecture notes refer only to the computing part of the module.*

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 Office hours

The scheduled office hour for this module is Monday 1pm–2pm; I will always be free at this time, in my office, to see students who need extra help with the work. Occasionally, I will hold these office hours in LT2. I also have office hours for MAS8306 (see the timetable on my noticeboard), 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 Many of you said more direct teaching of the computing part of MAS1801 would have helped.

² See the calendar on page 3 of these notes for a full week-by-week schedule.

1.1.3 Lectures

Given the nature of the first part 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 computer skills. I will hand out copies of the lecture notes and practical worksheets in the Thursday 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 BB or via the dedicated course webpage³:

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

1.1.4 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. We won't try to catch you out!
- You ask for help
- 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”.

All our computer practicals are held in the Herschel cluster, 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:

<http://m.ncl.ac.uk/itservice/#clusfree>

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

1.1.5 Assessment

- End of semester examination worth **40%** of your overall mark. This will include material from the pencil-and-paper part of the course after Easter.
- Assessment of course-work worth **60%** of your overall mark:
 - ‘Weekly’ computing exercises (10%)
 - Group computing project (30%)
 - Problem-solving exercises (20%)

³I will try to keep on top of BB/webpage updates, but there will often be a bit of a lag between students receiving materials and these materials appearing online.

As in MAS1801, there will be lots of available help in computer practicals: Dr. Fletcher, Dr. Baggaley, Dr. Kakariadis and Dr. Billam will all be on-hand, as will several Ph.D students.



The weekly computer exercises will be due on on the following dates, all of which are Wednesdays⁴:

17th February (23:45); 24th February (16:00); 9th March (23:45).

The group computing project will be due in by no later than 4pm, 18th April.

⁴ Online submissions, via NESS, are due in at 23:45; standard ‘hard-copy’ submissions are submitted in the usual way, by 16:00.

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/students/teaching/homework/>

1.2 What is R?

R is a computer package that is widely used for statistical software development and data analysis⁵. 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 it’s 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!). You will use R throughout your degree at Newcastle.

1.2.1 Accessing R

R is installed on all University machines. In this class we will be using Rstudio, which is an R IDE⁶. Both R and Rstudio are free, so you can install them on you own computer⁷. See

<http://www.ncl.ac.uk/maths/students/teaching/installingr/>
for more details.

⁶ Integrated development environment (IDE).

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

Title	Year	Length	Budget	Voting statistics					Movie genre							
				Rating	Votes	r1	...	r10	mpaa	Action	Animation	Comedy	Drama	Documentary	Romance	Short
A.k.a. Cassius	1970	85	-1	5.7	43	4.5	...	14.5	PG	0	0	0	0	1	0	0
AKA	2002	123	-1	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	-1	5.0	46	14.5	...	24.5	R	0	0	0	0	0	0	0
Aberration	1997	93	-1	4.8	149	14.5	...	4.5	R	0	0	0	0	0	0	0
Abilene	1999	104	-1	4.9	42	0.0	...	24.5	PG	0	0	0	1	0	0	0
Ablaze	2001	97	-1	3.6	98	24.5	...	14.5	R	1	0	0	1	0	0	0
Abominable Dr	1971	94	-1	6.7	1547	4.5	...	14.5	PG-13	0	0	0	0	0	0	0
About Adam	2000	105	-1	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/>.

1.2.2 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.

1.2.3 Recommended R textbooks

At <http://goo.gl/nzQTK>, I have constructed list of suitable books (with comments) 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:

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

1.3 Movie data set

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 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 Association of America*) rating. 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 '-1'.
- 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 a 1, ..., 10
- mpaa. MPAA rating; e.g. PG, PG-13, R.
- 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.

There are a total of 24 variables and 4847 films. The first few rows are given in Table 1.1. We will use this dataset to illustrate the concepts covered in this class.

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

1.4 Module R package

We have written an R package for another module (MAS1343) which you will find useful in this course. Installing this package is straightforward, and will make loading datasets used in lectures, and some other features of the computer practicals, much easier:

```
> install.packages("mas1343",
+                   repos="http://R-Forge.R-project.org",
+                   type="source")
```

To load the package, use:

```
> library(mas1343)
```

Then to load the movies dataset, type:

```
> data(movies)
```

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

Don't include the plus signs if you wish to execute this code in R - R inserts these automatically if a single line of code wraps over more than one line in the terminal. So this example code represents a single line.

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...

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

...and for more ‘advanced’ operations:

```
> 2^3
[1] 8
> exp(1.5)
[1] 4.481689
> log(10)
[1] 2.302585
> 4 %% 3
[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 computer practicals we will use assignment, i.e. $x = 5$. However, computer assignment is different from typical mathematical assignment¹. For example:

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

Notice that when we type $x = 5$, R doesn’t display or print any output to the screen². 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 brackets³. For example:

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

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

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.

You can also use the `<-` operator for assignment. This is, for almost all situations, identical to the `=` operator.

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 types 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 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(mas1343)
> 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())
```

```
> source("~/Rprofile")
> library(mas1343)
> data(movies)
```

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 created a vector of **doubles** and in line 3 we assigned 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 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	<code>length(x)</code>	4
Reverse order	<code>rev(x)</code>	3,5,5,1
Sort	<code>sort(x)</code>	1,3,5,5
Sum	<code>sum(x)</code>	14
Extract unique elements	<code>unique(x)</code>	1,5,3
Indices of particular elements	<code>which(x==5)</code>	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 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 make use of 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 commonly used in probability. 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 R	A	B	$\bar{A}$!A	$\bar{B}$!B	$A \cap B$ A & B	$A \cup B$ 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 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
```

We can also combine logical operators:

```
> vec2 > 3
[1] FALSE FALSE TRUE TRUE TRUE
> !(vec2 > 3)
[1] TRUE TRUE FALSE FALSE FALSE
```

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

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

2.4.3 Vector partitions

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
```

```
> vec2
[1] 0.0 2.5 5.0 7.5 10.0
> vec2 > 3 & vec2 < 9
[1] FALSE FALSE TRUE TRUE FALSE
> vec2[vec2 > 3 & vec2 < 9]
[1] 5.0 7.5
```

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

```
> length(Budget)
[1] 4847
```

⁵ To load movie budgets, use the following commands: `library(mas1343); data(Budget); data(Length)`

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

```
> non_zero_b = Budget[Budget != -1]
> length(non_zero_b)
[1] 1785
```

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

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

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"
```

WHEN WE download the movies data set, we automatically create a data frame:

```
> dim(movies)
[1] 4847  24
> colnames(movies)[1:4]
[1] "Title" "Year"  "Length" "Budget"
> movies[4061, 1:4]
              Title Year Length  Budget
4061 Star Wars: Episode I - The Phantom Menace 1999    133 1.15e+08
```

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] 1738  24
```

...or movies that cost more than \$100,000 but are not R rated...

```
> m2 = movies[movies$Budget > 100000
+   & movies$mpaa != "R",]
> dim(m2)
[1] 727  24
```

...or movies that are either PG or PG-13:

```
> m3 = movies[
+   movies$mpaa == "PG" | movies$mpaa == "PG-13",]
> dim(m3)
[1] 1515  24
```

