DR LEE FAWCETT

MAS1343 COMPUTATIONAL PROBABILITY AND STATISTICS (WITH R)

NEWCASTLE UNIVERSITY

Timetable

- Times:
  - Monday @ 9am to 11am: Lecture in LT3
  - Wednesday @ 9am to 10am: Computer lab in Herschel PC cluster
  - Thursday @ 3pm to 4pm: Office hour/Drop-in in LT2
- Attendance will be taken in all labs
- Any box with a ✘ indicates no class that day
- Feedback and Problems classes: Monday will not *always* be a two hour lecture. It will often include time for problems and feedback
- This timetable may change, so check your University email regularly
- CBAs: This module has a single CBA

Practicals: Solutions to all practical work should be handed in by 4pm at the School Office. 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
Jan 26th **1** **Lecture**: §1	28th **2** ✘	29th **3** **Lecture**: §2
Feb 2nd **4** **Lecture**: §3	4th **5** Lab: Practical 1	5th **6** *Office hour*
9th **7** **Lecture**: §4 & §5 Hand-in: Practical 1	11th **8** Lab: Practical 2	12th **9** *Office hour*
16th **10** **Lecture**: §5 Hand-in: Practical 2	18th **11** Lab: Practical 3	19th **12** *Office hour*
23rd **13** **Lecture?**	25th **14** Lab: *Prac. 3 ctd...*	26th **15** Drop-in: Practical 3
Mar 2nd **16** **Lecture**: §6 Hand-in: Practical 3	4th **17** Lab: Practical 4	5th **18** **Lecture**: Monopoly
9th **19** **Lecture**: §7	11th **20** Lab: *Prac. 4 ctd...*	12th **21** Drop-in: Practical 4

TERM 2 SCHEDULE.

CBA 1 practice week

CBA 1 assessed week

MONDAY	WEDNESDAY	THURSDAY
Apr 13th **1** **Lecture**: §8 Practical 5 given out Hand-in: Practical 4	15th **2** ✘	16th **3** *Office hour*
20th **4** **Lecture**: §9 Hand-in: Practical 5	22nd **5** Lab: Practical 6	23rd **6** *Office hour*
27th **7** **Lecture?**	29th **8** Lab: *Prac. 6 ctd...*	30th **9** Drop-in: Practical 6
May 4th **10** **Lecture:** Revision Hand-in: Practical 6	6th **11** ✘	7th **12** ✘
11th **13** **Lecture:** Revision	13th **14** ✘	14th **15** ✘

TERM 3 SCHEDULE.

Contents

1
Introduction and Housekeeping

1.1 Housekeeping

This course will use lectures, problem classes, drop-in sessions, computing practicals and feedback sessions. Since this course contains a higher component of course work, we will have **fewer** lectures.

1.1.1 Lecturer information

The lecturer for this module 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 Thursday 3–4. I also have office hours for MAS2317 (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 3pm.

1.1.3 Lectures

We will have fewer lectures than the other first year modules, but will have more computer labs. I will hand out copies of the lecture notes and practicals in class.

A copy of the notes and practicals can also be downloaded from Blackboard.

Lectures will also be recorded. You can access the recordings through BB or at the following webpage:

`http://www.mas.ncl.ac.uk/~nlf8/teaching/mas1343/`

The lecture slides will (eventually) be put on both the course webpage and BB.

1.1.4 Assessment

- End of semester examination worth 60% of your overall mark. In order to pass this module you must get at least 35% in the exam.
- Assessment of course-work worth 40%:

– A **single** CBA - due week 3;
– Practicals;
– Demonstrations of computer skills at the lab.

Warning: last year a few students passed the exam, but failed the module due to poor course-work marks.

Not all practicals are worth the same marks:

Practical	1	2	3	4	5	6
Weight(%)	5	4	9	9	5	5

Table 1.1: Approximate coursework weightings. On average, 1% of course-work will take 1 hour (assuming you are familiar with your lecture notes). CBA 1 is worth 3% of your final mark.

1.1.5 Computing practicals

During each computer practical, your general computer skills will be assessed. In particular, we will address the following questions:

- Are your files organised in a sensible manner? For example, do you have a directory for MAS1343 and sub-directories for each practical?
- Are your files named in a structured manner?
- Do you use the correct text editor?
- Does your code "look nice"?

In this course, all our computer practicals are held in the Herschel cluster. 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`.

1.1.6 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 analysis.[1] 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.

[1] 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

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

is it's graphical facilities, which produce quality graphs 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.

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 IDE.[2] Both R and Rstudio are free, so you can install them on you own computer.[3] See

http://www.ncl.ac.uk/maths/students/teaching/installingr/

for more details.

[2] Integrated development environment (IDE).

[3] 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. Overall, the programming related aspects probably accounts for 50% of the final mark in this course - there are programming questions in the exam! It is crucial that you come to all the practicals for this course.

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.

1.3 Movie data set

The internet movie database, 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 about 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 were selected for inclusion if they had a known length, had been rated by at least one imdb user and had an mpaa 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.
- 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 1.2.

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

There are a total of 24 variables and 4847 films. The first few rows are given in Table 1.2. We will use this dataset to illustrate the concepts covered in this class.

1.4 Module R package

This module has an associated R package. Installing this package is straightforward:

```
> 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)
```

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 subtraction:[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.

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 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 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())
```

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.

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

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 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 type of a variable:

```
> 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	`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 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 MAS1341. So for example,[4]

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

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

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 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
==	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`

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]

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

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

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
```

3
Summary Statistics

3.1 Measures of location

3.1.1 Sample mean

One of the most important and widely used measures of location is the (arithmetic) mean:

$$\bar{x} = \frac{x_1 + x_2 + \ldots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i \, .$$

So if our data set was $\{0, 3, 2, 0\}$, then $n = 4$. Hence,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{0+3+2+0}{4} = 1.25 \, .$$

3.1.2 Sample median

The sample median is the middle observation when the data are ranked in ascending order. Denote the ranked observations as $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$. The sample median is defined as:

Remember that $x_{(n+1)/2}$ is the $(n+1)/2^{\text{th}}$ ordered observation.

$$\text{Sample median} = \begin{cases} x_{(n+1)/2}, & n \text{ odd;} \\ \frac{1}{2}x_{(n/2)} + \frac{1}{2}x_{(n/2+1)}, & n \text{ even} \, . \end{cases}$$

The median is more robust than the sample mean, but has less useful mathematical properties.

For our simple data set $\{0, 3, 2, 0\}$, to calculate the median we re-order it to: $\{0, 0, 2, 3\}$; then take the average of the middle two observations, to get 1.

3.1.3 Sample mode

The mode is the value which occurs with the greatest frequency. It only makes sense to calculate or use it with discrete data. In R we use the `table` function to calculate the mode.

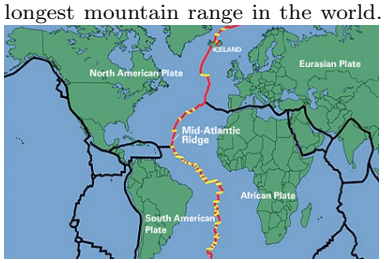
Warning: in R the function `mode` doesn't give you the sample mode. Use `table` instead.

3.1.4 Examples

The number of earth tremours recorded for five randomly chosen locations in Iceland, close to the *Mid-Atlantic ridge*, is recorded below:

Location	1	2	3	4	5
# tremours	7	1	14	13	20

The Mid-Atlantic Ridge is a divergent tectonic plate along the floor of the Atlantic Ocean, and is part of the longest mountain range in the world.



- Calculate the mean and median number of earth tremours for this sample.
  Solution

  The mean is:

$$\bar{x} = \frac{1}{5}(1+7+13+14+20) = 11$$

  The median is the middle value, so we get 13.

- Suppose doubt is cast over the reliability of the observed value at the second location. Find new values for the mean and median, excluding this observation.
  Solution

- Suppose, now, that we trust the observation at the second location. However, there was a recording error for location 5, which is closest to the Mid-Atlantic Ridge; this observation *should have* been 200, and not 20. Find new values for the mean and median, and comment.

  Solution

The mean can be distorted by unusually high or low values.

3.2 Measures of spread

As well as knowing the location statistics of a data set, we also need to know how variable or 'spread-out' our data are.

3.2.1 Range

The range is easy to calculate. It is simply the largest minus the smallest.

When you get a new data set, calculating the range is useful when checking for obvious data-inputting errors.

$$\text{Range} = x_{(n)} - x_{(1)} .$$

So for our data set of $\{0,3,2,0\}$, the range is $3-0=3$. It is very useful for data checking purposes, but in general it's not very robust.

Obviously, the range can be distorted by outliers or extreme observations.

3.2.2 Sample variance and standard deviation

In statistics, the mean and variance are used most often. This is mainly because they have nice mathematical properties, unlike the median, say.

The sample variance, s^2, is defined as

$$\begin{aligned} s^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2 \\ &= \frac{1}{(n-1)}\left\{\left(\sum_{i=1}^{n}x_i^2\right) - n\bar{x}^2\right\} . \end{aligned}$$

The second formula is easier for calculations. So for our data set, $\{0,3,2,0\}$, we have

The divisor is $n-1$ rather than n in order to correct for the bias which occurs because we are measuring deviations from the sample mean rather than the "true" mean of the population we are sampling from - more on this in MAS1341.

$$\sum_{i=1}^{4}x_i^2 = 0^2+3^2+2^2+0^2 = 13.$$

So:

$$s^2 = \frac{1}{n-1}\left\{\left(\sum_{i=1}^{n}x_i^2\right) - n\bar{x}^2\right\} = \frac{1}{3}\left(13-4\times 1.25^2\right) = 2.25.$$

The sample standard deviation, s, is the square root of the sample variance, i.e. for our toy example $s=\sqrt{2.25}=1.5$.

The standard deviation is preferred as a summary measure as it is in the units of the original data. However, it is often easier from a theoretical perspective to work with variances.

3.2.3 Examples

- Calculate the variance of $\{1,7,13,14,20\}$.

  Solution

  We calculate

  $$\sum_{i=1}^{5}x_i^2 = 1^2+7^2+13^2+14^2+20^2 = 815$$

  and $n\bar{x}^2 = 605$. So

  $$s^2 = \frac{815-605}{4} = 52.5 .$$

- Calculate the variance of $\{7, 13, 14, 20\}$.
  Solution

- Calculate the variance of $\{1, 7, 13, 14, 200\}$.
  Solution

3.2.4 Quartiles and the interquartile range

The upper and lower quartiles are defined as follows:

- $Q1$: Lower quartile = $(n+1)/4^{\text{th}}$ smallest observation.
- $Q3$: Upper quartile = 3 $(n+1)/4^{\text{th}}$ smallest observation.

We can linearly interpolate between adjacent observations if necessary. The interquartile range is the difference between the third and first quartile, i.e.

$$IQR = Q3 - Q1.$$

Examples

For the following data sets, calculate the inter-quartile range:

1. $\{5,6,7,8\}$[1] The lower quartile is the $5/4 = 1.25^{\text{th}}$ smallest observation, i.e. $x_{(1.25)}$. The value of $x_{(1)} = 5$ and $x_{(2)} = 6$. So

   $$x_{(1.25)} = x_{(1)} + 0.25 \times (x_{(2)} - x_{(1)}) = 5 + \frac{6-5}{4} = 5.25.$$

   Similarily, the upper quartile is the $3 \times (4+1)/4 = 3.75^{\text{th}}$ smallest observation. So

   $$x_{(3.75)} = x_{(3)} + 0.75 \times (x_{(4)} - x_{(3)}) = 7.75.$$

   Therefore,

   $$IQR = 7.75 - 5.25 = 2.5.$$

   [1] So $n = 4$.

2. $\{10,15,20,25,50\}$[2]. The lower quartile is the $(5+1)/4 = 1.5^{\text{th}}$ smallest observation, i.e. $x_{1.5}$, so

   $$x_{(1.5)} = x_{(1)} + 0.5 \times (x_{(2)} - x_{(1)}) = 12.5.$$

   Similarily, the upper quartile is the $3 \times (5+1)/4 = 4.5^{\text{th}}$ smallest observation. So

   $$x_{(4.5)} = x_{(4)} + 0.5 \times (x_{(5)} - x_{(4)}) = 37.5.$$

   Thus,

   $$IQR = 37.5 - 12.5 = 15.$$

   [2] So $n = 5$

3. $\{1,7,13,14,20\}$.

4. $\{7,13,14,20\}$.

3.3 Using R

For the movie data in section §1.3, we can easily use R to calculate the summary statistics. First we load the data from the `mas1343` package:

```
> library(mas1343)
> data(movies)
```

To calculate location measures, we use the `mean` and `median` functions:

```
> mean(movies$Budget)
[1] 10286893
> median(movies$Budget)
[1] -1
```

Notice that the budget mean and median are substantially different... why?

We can also calculate measures of spread:

```
> range(movies$Budget)
[1] -1e+00  2e+08
> var(movies$Budget)
[1] 5.308283e+14
> sd(movies$Budge)
[1] 23039711
```

To get the quartiles from R we use the `quantile` command, i.e.

```
> quantile(movies$Rating, type=6)
  0%  25%  50%  75% 100%
 1.0  4.6  5.7  6.6  9.1
> summary(movies$Rating)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   4.600   5.700   5.523   6.600   9.100
```

Note the default in R gives you a slightly different quartile range, i.e. if you don't enter the `type=6` argument. As $n \to \infty$, the different quartile functions converge.

Command	Comment	Example
`mean`	Calculates the mean of a vector	`mean(x)`
`sd`	Calculates the standard deviation of a vector	`sd(x)`
`var`	Calculates the variance of a vector	`var(x)`
`quantile`	The vector quartiles. Make sure you use `type=6`	`quantile(x, type=6)`
`range`	Calculates the vector range	`range(x)`
`summary`	Calculates the quartiles **However,** it doesn't use `type=6` quartiles.	`summary(x)`

Table 3.1: Summary of R commands in this chapter.