

Practical 1: Data manipulation in R

1 Getting started

Log-in to a PC in the usual way. Now open Rstudio – if you’ve not done this before, you can click on the “Start” button in the bottom left-hand-corner and search for “rstudio”. Once you’ve opened Rstudio you should see four panels – if there is no text editor open in the top-left panel, click on the green button with a white plus symbol and open a new R *Script*. Remember – to execute a line of code in the text editor, hit `Ctrl-Enter`.

In the questions below, the important part is **understanding** what’s going on rather than just typing the **R** commands. Make sure you answer the questions under each piece of **R** code. If you can’t answer the question, then ask! Where possible, write your answer down on the worksheet. You should create the folder `Practical1` in a directory called `MAS1343` – this is where you can save your **R** session and any other files associated with this practical.

Work through *all* of the questions – the practice questions **and** the questions in the assignment!

2 Assessment during the practical

During this practical we will assess that you can correctly use Rstudio.

3 Practice questions

You should try these questions before attempting the assignment.

1. Why does the code in listing 1 produce an error message?

Listing 1: Simple error message

```
x = "1"
y = 2
x*y
```

2. The `seq` command:

Listing 2: The sequence command

```
x = seq(0, 10)
y = seq(0, 1001, by=0.5)
z = seq(1000, 0, by= -exp(1))
```

- a) For each line in listing 2, describe what **R** is doing.
- b) What is the length of each vector?
- c) Generate the vector: -1, -0.5, 0.0, ..., 10.
- d) Generate the vector: 100, 102, ..., 206.

3. Vectors

Listing 3: Vectors

```
#You will find out more about the runif command in a few weeks.  
x = runif(5000, 1, 8)
```

- a) What is the length of x ?
- b) How many values in your vector x are below 2?
- c) How many values in your vector x are above 7?
- d) How many values in your vector x are below 3 or above 8?
- e) How many times does the value 2 occur?
- f) What are the 3400th – 3402th elements in your vector?
- g) What is the smallest value in x ?
- h) What is the largest value in x ?
- i) What are the first and third quartiles of x ?
- j) Create a new vector called y which contains all values of x that are between 2 and 6.
- k) **Tricky:** Which value in x is the closest to 5, i.e.

$$\min(|5 - x_i|)?$$

You can use the **abs** to work out absolute values.

4. Data frames.

```
1 app_ex1 = data.frame(c1=runif(50), c2=rnorm(50), c3=runif(50))
```

- a) What does line 1 do?
- b) How many rows in `app_ex1` are there where `c1 > 0.2`?
- c) How many rows in `app_ex1` are there where `c1 > 0.2` and `c2 > 0.2`?
- d) How many rows in `app_ex1` are there where `c1 > 0.2` and `c2 > 0.2` and `c3 > 0.2`?
- e) How many rows in `app_ex1` are there where `c1 > 0.2` or `c3 < 0.5`?

Assignment

Preliminaries

Before starting, load the `mas1343` package:

```
> library(mas1343)
```

For questions 1 to 4, submit your answers using Blackboard.

- Log onto BB
- Go to MAS1343
- On the left hand menu click assignments
- Click “Practical 1 - Questions 1 to 4”

Warning: Only go to BB once you have answers for all four questions.

Question difficulty

Some of the questions are fairly straightforward, others are fiendishly difficult.

- **Straight forward:** Q1: {1 – 5}; Q2: {1, 4}; Q3: {1 – 2}; Q4: {1 – 3, 7, 8}.
- **Doable but tricky:** Q1: {6, 7}; Q2: {2, 3}; Q4: {4, 5, 6}.
- **Fiendish:** Q2: 5; Q3: 3; Q4: 9.

Look at the commands in table 2.2 of your notes for ideas. The `table` command may also be useful.

Question 1

Run the following **R** code

```
> x1 = GetNumericVector(STUDENT_ID)
> #Where STUDENT_ID is your student id. For example
> #x1 = GetNumericVector("b1234567")
```

1. What is length of `x1`?
2. What is the 55th element of `x1`?
3. What is the final element of `x1`?
4. What is the smallest value of `x1`?
5. How many values are greater than –11 but less than 2.2?
6. How many values fall in the region: –17.86, 22.26?
7. **Tricky:** What is the 4000th smallest value in `x1`?

Question 2

Run the following R code

```
> x2 = GetLogicalVector(STUDENT_ID)
```

1. How many times does TRUE appear in x2?
2. What is the position in the vector of the first occurrence of TRUE?
3. What is the position in the vector of the last occurrence of TRUE?
4. What is the value of the 4000th element of x2?
5. **Tricky:** What position in the vector is the 2406th occurrence of TRUE?

Question 3

Run the following R code

```
> x3 = GetCharacterVector(STUDENT_ID)
```

1. How many times does "A" appear in x3?
2. Which letter appears the least? If more than one letter appears, just give the first letter (if the letters were sorted in alphabetical order).
3. **Very tricky:** How many pairs of letters are there in the x3? For example, in AABCCC we would have 3 pairs of letters.

Question 4

Run the following R code

```
> y = GetDataFrame(STUDENT_ID)
```

The data frame y is a subset of the movie data we use in the lectures.

1. How many rows does y have?
2. How many columns does y have?
3. How many movies are there where the budget is known, i.e. where Budget != -1?
4. How many movies are there where the rating is less than 2.5 or greater than 7.5 (including 2.5 and 7.5)?
5. How many movies are there where the length is greater than 120 and have a rating of more than 7.5 (not including 120 and 7.5)?
6. How many movies were made in 1980 and have a rating above 5.0?
7. How many movies are classified as Action?
8. How many movies were classified as both Action and Animation?
9. **Very tricky:** How many movies were made in even years?

Practical 2: Plotting

Complete this sheet as you work through it. If you run into problems, then ask for help - don't skip sections! Open Rstudio and store any files you download or create in a directory called `Practical2`.

1 The data

Downloading the data

In today's practical, we are going to examine the big bang data that was collected in Monday's lecture.

Use the `library` command to load the `mas1343` package:

```
> library(mas1343)

> bb1415=read.table("bb1415.txt",header=TRUE)
```

Now to load the data from Monday's quiz, type the following:

```
> bb1415 = read.table("http://www.mas.ncl.ac.uk/~nlf8/
+ teaching/mas1343/prac/bb1415.txt",header=TRUE)
```

Do not enter the + sign. I have done this because the code would not fit on a single line in this handout – you should enter the code in a single line!

This downloads the data from the MAS1343 webpage and stores it in the dataframe `bb1415`. The argument `header=TRUE` tells **R** to use the column headings given in the file.

Provided this has all worked smoothly, you should now be able to view the data by typing:

```
> bb1415
```

However, the file might be too big for you to view conveniently on the screen. You can get an idea of what the data looks like by using the `head` function, as was demonstrated in the very first lecture:

```
> head(bb1415)
```

Each row in the dataframe corresponds to a team from Monday's lecture, with their guesses of the ages of the actors/actresses in each of the columns labelled with a character's name. "Watched" takes the value `TRUE` if most of the team watch The Big Bang Theory; "Gender" takes the values "Male", "Female" or "Both", depending on the make-up of the team. Before you move on, please make sure you familiarise yourself with the dataframe in **R**!

Initial investigations

Since we are going to alter the data set, it's a good idea to create a copy, so assign the dataset to another object – say `d`:

```
> d = bb1415
```

We can inspect the column names using

```
> colnames(d)
```

and we can also look at individual columns, using

```
> d$Name  
> d$Shel
```

The dimensions of `d` are found using

```
> dim(d)
```

Complete: Number of columns: _____ Number of rows: _____

2 Scatter plots

2.1 The basics

Let's start with a scatter plot for Group ID against the Sheldon guesses:

```
> plot(d$ID, d$Shel)
```

The default x- and y-axis labels aren't very good, so we use `xlab` and `ylab` arguments, i.e.

```
> plot(d$ID, d$Shel, xlab="Group ID", ylab="Guess")
```

We can highlight the actual age using **abline**, i.e.

```
> abline(h=41)
```

Other commands for **abline** are

```
> abline(h=37, col=1)  
> abline(h=38, col=2)  
> abline(h=39, col=3, lty=2)
```

Complete: What does the `col` argument do? _____

Complete: What does the `lty` argument do? _____

Complete: What does the `h` argument do? _____

Complete: What happens if you change the `h` argument to `v`? _____

2.2 Separating by Gender

We will now use scatter plots to investigate differences between male and female guesses. First we need to separate the guesses into different categories. The following piece of code extracts the female teams...

```
> x_female = d$ID[d$Gender=="Female"]  
> y_female = d$Shel[d$Gender=="Female"]
```

... which we can then plot

```
> plot(x_female, y_female, pch="F")
```

Of course we can add in the `xlab` and `ylab` arguments.

Complete: Create new variables `x_male` and `y_male`. Add these points to your scatter plot, i.e.

```
> points(x_male, y_male, pch="M")
```

Make sure you include all teams in your scatter plot. To extend the x-axis, you may have to explicitly state the x-limits:

```
> ## Change XX & YY to something sensible
> plot(x_female, y_female, pch="F", xlim=c(XX, YY))
```

Complete: What happens if you omit the `pch` argument? _____

Complete: What happens if you change `pch="F"` to `pch=1` or `pch=2`? _____

Complete: Do you think that males guess the age of Sheldon better than females? _____

3 Summary Statistics

Use the commands **mean**, **median**, **quantile** and **sd** to calculate summary statistics for the `Sheldon` column. *Hint: look at Chapter 3 in your lecture notes!*

Complete: Mean: _____, Median: _____

Complete: Standard deviation: _____, Q1: _____, Q3: _____.

4 Histograms

We will now investigate the distribution of guesses using histograms. To plot a histogram of guesses for Sheldon we use the following code – **use the `xlab` and `ylab` commands to add labels to the axes, and also the `main` command to add a main title to your plots.**

```
> hist(d$Shel)
```

We can also add in vertical lines to show the mean and median of guesses:

```
> abline(v=mean(d$Shel))
> #What does the lty argument do?
> abline(v=median(d$Shel), col=2, lty=2)
```

We can generate two histograms on the same plot using the `par` command, e.g.

```
> par(mfrow=c(1, 2))
> hist(d$Shel, main="Sheldon Guesses")
> abline(v=mean(d$Shel), col=2)
> hist(d$Penny, main="Penny Guesses")
> abline(v=mean(d$Penny), col=2)
```

Complete: How are the graphs different? _____

Hint: Think about summary statistics.

Usually **R** makes a good initial guess when constructing histograms, but we often have to vary the number of bins to achieve the best result. We can do this using the `breaks` argument, i.e.

```
> hist(d$Shel, breaks=5)
> #You can also use the "scott" or the "fd" options as in lectures
> #Typically, these are better than the default.
> hist(d$Shel, breaks="scott")
> hist(d$Shel, breaks="fd")
```

Which rule would you use for the Shel data set – i.e. Default, Scott or FD? _____

5 Boxplots

Let's now investigate the error in age guessing:

$$|\text{actual age} - \text{estimated age}|$$

Sheldon's true age is 41, so the absolute error is:

```
> shel_err = abs(41-d$Shel)
```

To do a boxplot of the error, we use the command:

```
> boxplot(shel_err)
```

We can also investigate if people who have watched BB have an advantage when guessing:

```
> #Remember you can use xlab & ylab for labels!
> boxplot(shel_err ~ d$Watched)
> boxplot(shel_err ~ d$Watched, col="bisque") #In colour
```

Do you think watching BB increased guessing precision? _____

Assignment

Before you begin

Points to note:

- Remember to label your axis correctly.
- For this practical, copy your graphs into Word and type up your comments.
- If you are asked to submit histograms, try different bin number rules.
- Don't go overboard with the number of decimal places you quote.
- If you have more than five pages you will lose marks.
- If you have fewer than five pages you will lose marks.

What I want

The marks for each question are indicated below.

- **Page 1:** On separate lines I would like:
 - Module Name - *MAS1343*
 - My name - *Dr Lee Fawcett*
 - The practical number - *Practical 2*
 - Your name: surname and initial(s). For example, *Smith, J.H.*
 - Your student id. For example, *b9123456*
- **Page 2 [25%]:** Choose a Big Bang Theory actor, and plot a scatter plot of Group ID vs Guess.
 - Add a line to indicate the mean guess.
 - Add two other lines to represent the mean ± 2 standard deviations. *Hint: use **sd** to get the standard deviations, then use **abline**.*
 - Use the `lty` argument to distinguish between the lines.
- **Page 3 [25%]:** Produce a boxplot of the error against Gender for the Big Bang actor you previously chose (not Sheldon!). Do you think that females are better than males at guessing? What about mixed sex groups?
- **Page 4 [25%]:** Produce histograms of the error for four different characters. Put the histograms on the same plot. Use the **abline** command to indicate your group's guess on each of the plots. Your scales on the four histograms should be the same.
 - If you can't remember your group, choose the first group on the list.
- **Page 5 [25%]:** Guesses from two years ago can be found by typing:

```
> data(bb1213)
```

 - Using some graphs and summary statistics compare this year's guesses to those from two years ago. Do you think the class in 2012/13 are better at guessing than the this year's class? I don't want every possible plot and summary statistic. Just a brief overview.

Practical 3: Functions and loops

In the questions below, the important part is **understanding** what's going on rather than just typing the **R** commands. Make sure you answer the questions under each piece of **R** code. If you can't answer the question, then ask! Where possible, write your answer down on the worksheet. You might also wish to create a directory called `Practical3` in your `MAS1343` directory.

1 Practice questions

1. Default arguments

```
> Fun1 = function(x=10) {  
+   return(x)  
+ }  
> Fun2 = function(x) {  
+   return(x)  
+ }
```

- Why does `Fun1()` work, but `Fun2()` give an error message?
- Change `Fun1` so that it returns $\sqrt{x}$.

2. Basic functions

```
> v = 5  
> Fun3 = function() {  
+   v = 0  
+   return(v)  
+ }  
> Fun3()  
> v
```

- Why does line 7 above give 5 and not zero?
- Delete line 3 in the above listing. Now change the function to allow `v` to be passed as an argument, i.e. we can write `Fun3(5)`. Call this function to make sure it works.
- Now make the argument have a default value of 0.

- This question makes sure that you can save a file and upload it to NESS. In this and future practicals, it is crucial that you do this step correctly - or you will get a mark of zero.

Save the function `Fun2` in a file called `practice.R`. You should only save `Fun2` - nothing more, nothing less! You will now upload this file to NESS using the following steps - first of all, you must log-in to NESS. Then...

- Select "MAS1343" at the top of the screen.
- Select "Practice" from the drop-down "Coursework" menu at the left-hand-side.
- Click on "Coursework" underneath this, and then choose "Submit".
- "Browse" for the R program in the "Files to be submitted box".
- Click "Submit your work".

4. if statements

```
> Fun4 = function(x) {  
+   if(x==5) {  
+     y = 0  
+   } else {  
+     y = 1  
+   }  
+   return(y)  
+ }
```

- Call `Fun4` a few times with different values of `x`.
- Change `Fun4` so that it returns 0, if `x` is greater than 5.
- Change `Fun4` so that it returns 0, if `x` is greater than or equal to 5.
- Change `Fun4` so that it:
 - returns 1 if `x` is positive;
 - returns -1 if `x` is negative;
 - returns 0 if `x` is zero.
- Suppose that `x` is a vector. Change the function so that it returns 0 if the **mean**(`x`) is greater than 0.
- Change `Fun4` so that if `x` is positive, it returns $\sqrt{x}$.

5. for loops

```
> total = 0  
> for(i in 1:5) {  
+   total = total + i  
+ }  
> total
```

The **for** loop above calculates

$$\sum_{i=1}^5 i$$

- What is the final value of `total` in the above piece of code?
- Change the code above to calculate the following summations:

$$(i) \sum_{i=1}^{20} (i+1) \quad (ii) \sum_{j=-10}^{15} j \quad (iii) \sum_{j=-10}^{10} \exp(j^{1/j}) \quad (iv) 0, 3, 6, 9, \dots, 24 \quad (v) \sum_{j=0}^{20} \frac{1}{2^j}.$$

6. More for loops

```
> a = 2  
> total = 0  
> for(blob in a:5) {  
+   total = total + blob  
+ }
```

- Delete line 1 above and put the code in a function called `Fun5`, where `a` is passed as an argument, i.e. we can call `Fun5(1)`

- b) Alter the code so that the **for** loop goes from a to b, rather than a to 5. Allow b to be passed as an argument, i.e. we can call `Fun5(1, 5)`.
- c) Change `Fun5` so that it has default arguments of `a=1` and `b=10`.

7. The following function calculates the sum of a vector:

```
> Fun6 = function(x) {  
+   n = length(x)  
+   total = 0  
+   for(i in 1:n) {  
+     total = total + x[i]  
+   }  
+   return(total)  
+ }
```

Alter `Fun6` to calculate the mean of the vector `x`. Do not use the built-in mean function. Test your function.

Assignment

Preliminaries

For the questions below you will need to save your **R** code in a file called

`practical3.R`

You will upload this file to NESS using the following steps:

- Choose “Coursework” from the top-level menu.
- Choose “MAS1343” from the drop-down Module box.
- Choose “Practical 3” from the drop-down Coursework box.
- Click “Submit” on the side menu.
- Browse for the R program in the files to be submitted box.
- Click “Submit your work”.

Warning: If you save your file incorrectly, you will receive zero marks for this practical!

Questions

In this section do not use the built-in **R** functions **sum**, **mean**, **median**, **sd**, or **var**.

1. Write a function called `MySD` that calculates the sample standard deviation, i.e.

$$\sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

This function should have a single argument `x`.

2. Write a function called `MyDifference` that takes two vector arguments, `x` and `y`, and calculates the differences in means, i.e.

$$\bar{x} - \bar{y}.$$

3. Write a function called `MyMean2`. This function has two arguments, `x` and `nonzero`, where `nonzero` has the default value `TRUE`. This function should return

- $\bar{x}$ if `nonzero = FALSE`
- $\bar{x}$ of all $x' \text{'s} > 0$ if `nonzero = TRUE`

4. Write a function called `IsEven`. This function has a single argument `x` and returns `TRUE` if the number is even and `FALSE` if the number is odd.
5. Write a function called `MyMedian`. This function has a single argument `x` and returns the median value of a vector. You may use the built-in function **sort** to help you.

Example function calls

```
> set.seed(1)
> x = runif(20, -1, 1)
> MySD(x)

[1] 0.5722358

> y = runif(20, -1, 1)
> MyDifference(x, y)

[1] 0.1625699

> MyMean2(x, nonzero=FALSE)

[1] 0.1103342

> MyMean2(x, nonzero=TRUE)

[1] 0.5559767

> IsEven(5)

[1] FALSE

> IsEven(6)

[1] TRUE

> MyMedian(x)

[1] 0.2019674
```

Checking your function

In the `mas1343` package, there are two useful functions to check your file

```
> library(mas1343)
> ##Make sure you file is named correctly first
> CheckFunctionName("practical3.R")
> CheckFunctionName("C:/My Documents/R/practical3.r")
> CheckFunctionName("practical3.r")
> ##Can your file be loaded
> ##This doesn't check if your functions are correct
> CheckFunctionLoading("practical3.R")
```

What should (and shouldn't) be in your file

1. Your file should only include functions. It can include additional, helper functions such as `MyMean`
2. Your file should not include function checks
3. Your file should not include `set.seed`
4. Your file should have any R statements outside of your function definitions

Practical 4



Complete this sheet as you work through it. Make sure you nicely indent your code.

1 Monopoly

1.1 Description of the problem

We wish to determine which properties a player is most likely to land on during a game of monopoly. To simplify things, we assume there is only a single player, we ignore everything to do with money, and we also ignore the 'Get out of Jail Free Cards'. The algorithm we will use is:

1. Begin the game on GO;
2. `current := current + dice roll`
3. Make a note of the new position.
 - If we land on 'Chance' or 'Community Chest', draw a card;
 - If we land on 'Go To Jail', move to Jail;
 - If we move, make a note of the new position;
4. Go back to step 2

After rolling the dice 100,000 times or so, stop.

1.2 Dice rolling

When we roll a single die, the probability mass function is:

$$\Pr[X = k] = 1/6 \quad \text{for } k = 1, \dots, 6$$

i.e. all sides of the dice are equally likely. This means we can use the **sample** function to simulate a die roll:

```
sample (c(1, 2, 3, 4, 5, 6), 1)
#Or just the same
sample (seq(1, 6), 1)
#Or
sample (1:6, 1)
```

To roll two dice, we simply call this function:

Listing 1: Rolling two dice

```
RollTwoDice = function () {
  total = sample (1:6, 1) + sample (1:6, 1)
  return (total)
}
```

Complete: Call the function `RollTwoDice` four times, what values did you get? _____

Complete: What are the range of possible values `RollTwoDice` could return? _____

Complete: If we tried to simulate rolling two dice using `sample(2:12, 1)`, why would this be wrong?

Complete: If we tried to simulate rolling two dice using `2*sample(1:6, 1)`, why would this be wrong?

1.3 The Monopoly board

In Monopoly there are forty properties, or squares; see table 1 at the end of this practical for a complete list. If we set the first square 'Go' to be number 1, then we can represent all forty squares as a vector in R. For example

```
#This creates a vector of 40 values;
#All values are initially zero
landings = numeric(40)
```

Then, when we land on a square we simply increase the associated `landings` entry by one. Suppose we landed on 'Old Kent Rd', we would represent this as:

```
landings[2] = landings[2] + 1
```

since 'Old Kent Road' is square 2 (see table 1).

Complete: Write down the code if we landed on 'Free Parking'? _____

Complete: Write down the code if we landed on 'Mayfair'? _____

[Please turn over]

1.4 Going round the board

Our first go at simulating Monopoly will ignore the community chest, chance cards, and the 'Go To Jail' square. This means that we are simply going round the board. The code in listing 2 rolls the dice `no_of_rolls` times, and stores the squares that are landed on in the vector `landings`.

Listing 2: Basic Monopoly

```
1 SimulateMonopoly = function(no_of_rolls) {  
2   landings = numeric(40)  
3   #Start at GO  
4   current = 1  
5   for(i in 1:no_of_rolls) {  
6     current = current + RollTwoDice()  
7     if(current > 40) {  
8       current = current - 40  
9     }  
10    landings[current] = landings[current] + 1  
11  }  
12    return(landings)  
13  }  
14  no_of_rolls = 50000  
15  sim = SimulateMonopoly(no_of_rolls)  
16  plot(sim/sum(sim), ylim=range(0.01, 0.04), type='l')
```

Complete: What does `landings = numeric(40)` do? _____

Complete: Explain the rationale for lines 9-12: _____

Complete: Why do we divide by `sum(sim)` above? _____

Complete: What happens to the plot as you increase `no_of_rolls`? _____

Complete: What values are the probabilities converging to (we will call this value the **baseline** probability)

Complete: Add a horizontal line to your plot showing the **baseline** probability.

[Please turn over]

1.5 Incorporating Community Chest Cards

There are three community chest squares on the board - squares 3, 18 and 34. In the code below we will just consider square 3. There are sixteen cards in total, hence the probability of drawing any particular card is 1/16. In the code below we will **only implement the first two community chest cards**:

Listing 3: Basic Community Chest function

```
CommunityChest = function (current) {  
  goto = current  
  u = runif(1)  
  if (u < 1/16) {  
    goto = 1#Move to Go  
  } else if (u < 2/16) {  
    goto = 11#Go To Jail  
  }  
  return (goto)  
}
```

This function takes in the `current` position, with probability 1/16 we 'Move to Go', with probability 1/16 we 'Go to Jail' and with probability 14/16 we stay in our current position. We now alter Listing 2 to incorporate the `communityChest` function:

Listing 4: Basic Monopoly

```
1 SimulateMonopoly1 = function (no_of_rolls) {  
2   landings = numeric(40)  
3   #Start at GO  
4   current = 1  
5   for (i in 1:no_of_rolls) {  
6     current = current + RollTwoDice()  
7     if (current > 40) {  
8       current = current - 40  
9     }  
10    landings[current] = landings[current] + 1  
11    if (current == 3) {  
12      cc_move = CommunityChest(current)  
13      if (cc_move != current){  
14        current = cc_move  
15        landings[current] = landings[current] + 1  
16      }  
17    }  
18  }  
19  return (landings)  
20 }  
21 no_of_rolls = 200000  
22 sim2 = SimulateMonopoly1(no_of_rolls)  
23 plot (sim2/sum(sim2), ylim=range(0.01, 0.04))
```

Complete: Explain the rationale for lines 14-20: _____

Complete: Add a horizontal line to your plot with the **baseline** probabilities.

2 Assignment

Each question adds an additional layer of complexity to your code.

1. Run the simulation function `SimulateMonopoly1`. [14%]
2. Add in the two other community squares, i.e. squares 18 and 34 into the `SimulateMonopoly1` code; [+7%]
3. Add in 'Go to Old Kent Road' into your `CommunityChest` function; [+7%]
4. Square 31 is 'Go To Jail.' Implement this in your main simulation function; [+7%]
5. Create a `Chance` function, that implements the first six Chance cards. When you land on a Chance square, call this function; [+10%]
6. Add in Community Chest card four; [+10%]
7. Add in Chance card 8. [+10%]
8. Add in Chance card 7, 'Go back 3 spaces'. [+10%]
9. Rolling a double (a pair of 1's, 2's, ..., 6's) is special: [+10%]
 - a) Roll two dice (`total1`): `total_score = total1`
 - b) If you get a double, roll again (`total2`) and `total_score = total1 + total2`
 - c) If you get a double, roll again (`total3`) and `total_score = total1 + total2 + total3`
 - d) If roll three is a double, Go To Jail, otherwise move `total_score`

Other marks are awarded for code presentation and style [+15%]

What to hand in

- a A front page as described in Practical 2.
- b Submit your nicely formatted code. This can be printed directly from RStudio or from Word. If you use Word, make sure the formatting isn't messed up.
 - There should only be a single `SimulateMonopoly1` function.
 - Of course, there will be other functions. For example `Chance`, `CommunityChest`.
 - If you submit more than one `SimulateMonopoly1`, marks will be deducted.
 - If you submit a function that **produces** an error, i.e. it doesn't even run without crashing, you may get zero marks.
 - Don't submit your plot commands. Just submit your R functions.
- c Plot the probability of being on a certain square, against property based on your code. Add in a **dashed** horizontal line with the **baseline** probabilities.
- d State clearly how many simulations you used. When working through the practical use about 10,000 simulations, but your final results should probably be at least 100,000 or even more depending on your computer speed.

Remember to label all axis and put your name & student id on your work. Also, put your student number on all plots.

3 Additional Information

Community Chest Cards

There are three community chest areas on the board (see Table 1). In total, there are 16 community chest cards.

1. Advance to Go;
2. Go to jail;
3. Go to Old Kent Road;
4. Take a Chance card instead;

Chance Cards

A Chance card is most likely to move players. There are three chance areas on the board (see Table 1). There are 16 chance cards in total, of which eight cards move the player:

1. Advance to Go;
2. Advance to Trafalgar Square;
3. Advance to Pall Mall;
4. Go directly to Jail;
5. Take a trip to Marylebone Station
6. Advance to Mayfair
7. Go back 3 spaces;
8. Advance token to nearest Utility. The utility squares are the water works and the electric company.

Square Number	Name	Square Number	Name
1	Go	11	Jail
2	Old Kent Road	12	Pall Mall
3	Community Chest	13	Electric Company
4	WhiteChapel Road	14	Whitehall
5	Income tax	15	Northumberland Avenue
6	King's Cross Station	16	Marylebone station
7	The Angel Islington	17	Bow Street
8	Chance	18	Community Chest
9	Euston Road	19	Marlborough Street
10	Pentonville Road	20	Vine Street
21	Free Parking	31	Go To Jail
22	Strand	32	Regent Street
23	Chance	33	Oxford Street
24	Fleet Street	34	Community Chest
25	Trafalgar Square	35	Bond Street
26	Fenchurch Street Station	36	Liverpool St Station
27	Leicester Square	37	Chance
28	Coventry St	38	Park Lane
29	Water Works	39	Super Tax
30	Piccadilly	40	Mayfair

Table 1: *Monopoly squares with associated square numbers*

Practical 5

Pencil & paper questions

There is no lab associated with this practical.

1. Show that

$$\sum_{i=1}^n (x_i - \bar{x})^2 = \sum_{i=1}^n x_i^2 - n\bar{x}^2.$$

2. Consider the congruential generator

$$r_{i+1} = (ar_i + b) \bmod m \quad \text{for } i = 0, 1, \dots$$

with seed $r_0 = 8$, a multiplier of 17, an additive constant of 5 and a modulo of 16.

- a) What are the first five terms of the sequence produced by this generator?
- b) What is the period (cycle length) for this generator?
- c) Is this the maximum possible period for this choice of m ? Justify your answer.

3. In each of the following questions use these $U(0, 1)$ random numbers:

0.4129, 0.7642, 0.9251

- a) Simulate three random numbers from the $Bern(0.6)$ distribution.
- b) Simulate three random numbers from the $Geom(0.5)$ distribution.
- c) Simulate three random numbers from the $Po(5)$ distribution.
- d) Simulate three random numbers from the $Bin(10, 0.5)$ distribution.

Practical 6: Integral estimation

In this practical we will use Monte-Carlo techniques to estimate the area under a curve, i.e. we will perform *numerical integration*. It will help if you have your lecture notes for chapter 8 in front of you. As an example, we will try to estimate the following integral:

$$f(x) = abx^{a-1}(1-x^a)^{b-1} \quad \text{for } 0 \leq x \leq 1,$$

where $a > 0$ and $b > 0$.

In the following example, we will use the values $a = 2.5$ and $b = 4$; then, we will plot the function. Type into R the following code:

```
a = 2.5; b = 4;
EvalF = function(x, a, b) {
  f = a*b*x^(a-1)*(1-x^a)^(b-1)
  return(f)
}
x = seq(0, 1, by=0.01)
y = EvalF(x, a, b)
plot(x, y, type="l")
```

To show the region which we're going to throw random numbers at (see chapter 8 lecture notes), we use the **abline** commands:

Listing 1: Rejection region

```
1 x = seq(0, 1, by=0.1)
2 y = EvalF(x, a, b)
3 plot(x, y, type="l", xlim=range(-1, 2))
4 abline(h=0, col=2, lty=2)
5 abline(h=2, col=2, lty=2)
6 abline(v=0, col=2, lty=2)
7 abline(v=1, col=2, lty=2)
```

Complete: What is the area of the rectangle that we throw random numbers at? _____

Complete: If we threw 1000 random numbers at the rectangle, and 500 landed under the curve, what is our estimate of the area under the curve? _____

To estimate the integral

$$\int_0^1 abx^{a-1}(1-x^a)^{b-1} dx$$

we can use the following function. Type this carefully into R, and then use it to approximate the integral given at the start of this handout.

```
Area = function(n, a, b) {
  no_of_hits = 0
  for(i in 1:n) {
```

```
#Generate a value on the x-axis
x = runif(1, 0, 1)
#Generate a value on the y-axis
y = runif(1, 0, 2)
if (EvalF(x, a, b) > y) {#i.e. does the value land under the curve?
  no_of_hits = no_of_hits + 1
}
}
area_of_rectangle = 2
area_under_curve = area_of_rectangle * no_of_hits/n
return (area_under_curve)
}
```

Assignment

Each person will have their own values of a and b . To generate your values, type the following:

```
set.seed (LOGIN_ID)
a = sample(3:14, 1); b = sample(3:14, 1)
```

where LOGIN_ID is your computer id without the 'a'.

1. For your values of a and b , estimate the area under the curve of $f(x)$, for $0 \leq x \leq 1$;
 - Produce a plot similar to that produced by listing 1, but alter your rejection region accordingly. What is your new area? Make $n = 10^6$ for your estimate.
 - Hand in a single page, with the plot above. State clearly your values of a , b and your estimate for the area under the curve.
2. A special function that comes up a lot in statistics is the standard Normal distribution, that is, $Z \sim N(0, 1)$ – giving a bell-shaped curve with mean 0 and variance 1. Z has the following probability density function:

$$g(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right).$$

Using Monte-Carlo integration, estimate the following integral:

$$\int_{-d}^d g(z) dz$$

where d is the value:

```
set.seed (LOGIN_ID)
d = round(runif(1, 0.5, 2.5), 2)
```

- Produce a plot similar to Listing 1, but alter your rejection region accordingly. What is your new area?
- For this question hand in two pages:
 - Page 1: A plot showing the sampling region and your function. State clearly your value of d and your integral estimate. Make sure $n = 10^5$ or more.
 - Page 2: Nicely formatted computer code for this question.

Your assignment should only be three pages long.