

Practical 1: R Refresher

Main Aim: To recap **functions**, **for loops** and **if statements** in R.

There are three activities to complete:

1. **The sticker album:** Self-directed reading and computing, covering Section 2.6 of the course notes
2. A look at **sampling distributions:** Independent work on for loops and functions, with 4 questions to answer
3. **Monte Carlo integration:** Guided example, with some questions to answer

All exercises should get you up-to-speed with functions, for loops and if statements. You can either follow the instructions on this handout, or use the associated slides.

Note that you should not attempt this practical unless you have completed all activities from week 1, including:

- Attend (or watch the recording for) the first synchronous online lecture, held on Wednesday 21st October
- Complete all work associated with Home Study videos 1 and 2

Of course, you must also have access to RStudio. If you still don't have RStudio installed on your machine, visit our student resources webpage:

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

From this link, click on the link for our "Computing page", log-in using your University details, and then click on the link for "R". Here you will find some basic resources aimed at students who have never used R before; scroll down, and you'll find information about how to install R on your home PC. Alternatively, Chris Graham has posted instructions on installing RStudio to the stage 2 induction Canvas page.

Suggested study times have been attached to each of the three activities below. However, we do not expect everyone to complete all tasks within the 90 minute practical session – you must make every effort to complete all unfinished work by the end of this week.

1 The sticker album

Instructions: Read through Section 2.6 of the course notes carefully, trying out all of the R code and making sure you understand the output produced.

You will be taken to your breakout room to complete this work, in which you can ask your tutor (Lee/Amy/Kieran/Nicola) for any help you might need.

Suggested study time: Around 30 minutes.

2 Distribution of the sample mean

The results of this exercise support material we will consider towards the end of the semester, but the exercise itself lends itself well to checking your basic programming skills in R!

You will be taken to your breakout room to complete this work, in which you can ask your tutor (Lee/Amy/Kieran/Nicola) for any help you might need.

Suggested study time: Around 30 minutes.

- (a) Commit the following code carefully to RStudio, and then produce a histogram of x . What can you say about the shape of the distribution you see?

```
1 x = rexp(10000, 2)
```

Do you know what the function `rexp` is actually doing? (don't worry if not!) Can you hazard a guess?

- (b) We now want to see what the distribution of the sample mean from x looks like.
- (i) Use the `sample` command to draw a random sample from x of length 30. Find the mean of this sample, and write it down.
 - (ii) Repeat what you did in (i) a further four times.
 - (iii) Now you should repeat what you did in part (i) 1000 times... but you should automate the process by using a for loop!

First of all, type the following code into RStudio, which will set up a vector of length 1000 to take sample means from repeated random samples from x :

```
1 xbar = vector("numeric", 1000)
```

Now at each iteration i in your for loop, $i = 1, \dots, 1000$, store the sample mean in position i of the vector `xbar`.

- (c) The *Central Limit Theorem*, which we will come back to later in this module, states that – regardless of the distribution of the population – the distribution of the sample mean will be Normal, with mean μ and variance σ^2/n , where
- μ is the population mean
 - σ^2 is the population variance
 - n is the sample size (e.g. $n = 30$ in part (b))

Suppose your vector `x` contains the population. Can you verify the Central Limit Theorem for the distribution of your sample means obtained in (b)(iii)?

- (d) Write a function that explores the Central Limit Theorem for different size samples from x , where the sample size n is an argument to be passed in the function.

3 Monte Carlo integration

You will be taken to your breakout room to complete this work, in which you can ask your tutor (Lee/Amy/Kieran/Nicola) for any help you might need.

Suggested study time: Around 30 minutes.

Suppose the random variable Z follows a **standard Normal distribution**, that is, $Z \sim N(0, 1)$. Then Z has probability density function given by

$$f_Z(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}, \quad z \in \mathcal{R}.$$

Commit the following code to RStudio, and make sure you understand what's going on.

```
1 z = seq(-4, 4, 0.001)
2 d = 1/(sqrt(2*pi))*exp(-(z^2)/2)
3 plot(z, d, xlab = "z", ylab = "density", type = "l")
```

Now suppose we want to find $\Pr(-2 < Z < 1)$. The old-school way of doing this might be to annotate a diagram such as that produced on the previous slide, and then use **statistical tables**! We're not going to do this. We're going to use *Monte Carlo integration*!

Consider throwing random points at the plot you produced earlier, across a "simulation grid" that covers the area you're interested in. First, draw the "simulation grid":

```
1 # Draw the simulation grid on your plot
2 # Lower bound for Pr(-2 < Z < 1)
3 abline(v = -2, lty = 2, col = "red")
4
5 # Upper bound for Pr(-2 < Z < 1)
6 abline(v = 1, lty = 2, col = "red")
7
8 # Covers the top of the Normal PDF
9 abline(h = 0.4, lty = 2, col = "red")
```

Now throw random points at the grid, repeatedly – and watch how your plot updates.

```
1 # Generate a sequence for the x co-ordinate (between -2 and 1)
2 xseq = seq(-2, 1, 0.0001)
3
4 # Generate sequence for the y co-ordinate (between 0 and 0.4)
5 yseq = seq(0, 0.4, 0.0001)
6
7 # Throw your first point at the grid!
8 lines(sample(xseq, 1), sample(yseq, 1), type="p", pch="X")
9 # And another!
10 lines(sample(xseq, 1), sample(yseq, 1), type="p", pch="X")
11 # Keep going...
12 lines(sample(xseq, 1), sample(yseq, 1), type="p", pch="X")
```

Observations

- As you throw random points at the grid — some will lie under the curve, some will lie above the curve
- What's the probability that a point will lie under the curve (call this A)?

$$\Pr(A) = \frac{\text{Area under the curve}}{\text{Area of simulation grid}}.$$

- Therefore, by rearranging, we can see that

$$\begin{aligned} \text{Area under the curve} &= \Pr(A) \times \text{Area of simulation grid} \\ \Pr(-2 < Z < 1) &\approx \frac{\text{\#points under the curve}}{\text{\#points simulated}} \times 1.2. \end{aligned}$$

Study the following block of code, and then try it out – make sure you understand how the function works, and how it is replicating the process described above.

```

1 MCIntegrate = function(N){
2   no_of_hits = 0
3   for(i in 1:N){
4     # Generate the x/y co-ordinates in a better way:
5     x = runif(1, -2, 1)
6     y = runif(1, 0, 0.4)
7
8     # Evaluate the Normal density at x:
9     d = 1/(sqrt(2*pi))*exp(-(x^2)/2)
10    if(y<d){      # Is the point under the curve?
11      no_of_hits = no_of_hits+1
12    }
13  }
14  P = no_of_hits/N          # Find P
15  area_under_curve = P*(3*0.4) # Find the area:
16  return(area_under_curve)}

```

- Run the function `MCIntegrate` using $N = 10, 100, 1000$ and $10,000$.
- Can you generalise the function so that it can be used to find probabilities from the standard Normal distribution between *any* two points (as opposed to just between -2 and 1)?
- Explore the R's built-in function `pnorm` to evaluate the integral exactly, and compare to your answers in part (a).