

# MAS1802: Problem Solving II (Statistical Computing with R)

**Dr Lee Fawcett**

**School of Mathematics & Statistics**

Semester 2, 2016/2017

# Welcome back!

## Where we're at:

- Assignment 1 (3.33%): Completed, submitted, marked, [feedback ready](#)
- Assignment 2 (3.33%): Due in this coming Wednesday 4pm – hard-copy submission to the homework submission letterboxes, attach a NESS cover-sheet
- Assignment 3 (3.33%): Will start this coming Friday in the main practical session, drop-in next Tuesday 10-11, deadline next Wednesday 11:45pm
- Group project (20%): Will start this coming Friday in the main practical session; remaining practical sessions dedicated to project work
- Next week: bank holiday Monday! Drop-in Tuesday 10am, lecture Wednesday 10am (Curtis Auditorium)
- Practice CBAs

# Welcome back!

Where we're at:

- Assignment 1 (3.33%): Completed, submitted, marked, [feedback ready](#)
- Assignment 2 (3.33%): Due in this coming Wednesday 4pm – hard-copy submission to the homework submission letterboxes, attach a NESS cover-sheet
- Assignment 3 (3.33%): Will start this coming Friday in the main practical session, drop-in next Tuesday 10-11, deadline next Wednesday 11:45pm
- Group project (20%): Will start this coming Friday in the main practical session; remaining practical sessions dedicated to project work
- Next week: bank holiday Monday! Drop-in Tuesday 10am, lecture Wednesday 10am (Curtis Auditorium)
- Practice CBAs

# Welcome back!

Where we're at:

- Assignment 1 (3.33%): Completed, submitted, marked, [feedback ready](#)
- Assignment 2 (3.33%): Due in this coming Wednesday 4pm – hard-copy submission to the homework submission letterboxes, attach a NESS cover-sheet
- Assignment 3 (3.33%): Will start this coming Friday in the main practical session, drop-in next Tuesday 10-11, deadline next Wednesday 11:45pm
- Group project (20%): Will start this coming Friday in the main practical session; remaining practical sessions dedicated to project work
- Next week: bank holiday Monday! Drop-in Tuesday 10am, lecture Wednesday 10am (Curtis Auditorium)
- Practice CBAs

# Welcome back!

Where we're at:

- Assignment 1 (3.33%): Completed, submitted, marked, [feedback ready](#)
- Assignment 2 (3.33%): Due in this coming Wednesday 4pm – hard-copy submission to the homework submission letterboxes, attach a NESS cover-sheet
- Assignment 3 (3.33%): Will start this coming Friday in the main practical session, drop-in next Tuesday 10-11, deadline next Wednesday 11:45pm
- Group project (20%): Will start this coming Friday in the main practical session; remaining practical sessions dedicated to project work
- Next week: bank holiday Monday! Drop-in Tuesday 10am, lecture Wednesday 10am (Curtis Auditorium)
- Practice CBAs

# Welcome back!

Where we're at:

- Assignment 1 (3.33%): Completed, submitted, marked, [feedback ready](#)
- Assignment 2 (3.33%): Due in this coming Wednesday 4pm – hard-copy submission to the homework submission letterboxes, attach a NESS cover-sheet
- Assignment 3 (3.33%): Will start this coming Friday in the main practical session, drop-in next Tuesday 10-11, deadline next Wednesday 11:45pm
- Group project (20%): Will start this coming Friday in the main practical session; remaining practical sessions dedicated to project work
- Next week: bank holiday Monday! Drop-in Tuesday 10am, lecture Wednesday 10am (Curtis Auditorium)
- Practice CBAs

# Welcome back!

Where we're at:

- Assignment 1 (3.33%): Completed, submitted, marked, [feedback ready](#)
- Assignment 2 (3.33%): Due in this coming Wednesday 4pm – hard-copy submission to the homework submission letterboxes, attach a NESS cover-sheet
- Assignment 3 (3.33%): Will start this coming Friday in the main practical session, drop-in next Tuesday 10-11, deadline next Wednesday 11:45pm
- Group project (20%): Will start this coming Friday in the main practical session; remaining practical sessions dedicated to project work
- Next week: bank holiday Monday! Drop-in Tuesday 10am, lecture Wednesday 10am (Curtis Auditorium)
- Practice CBAs

# Welcome back!

The Big Bang Theory: prizes for smallest absolute error!

1. "Danielle & Sarah" and "A.Epp" (overall absolute error = 9)
2. "Integer Milan" (overall absolute error = 11)
3. "Nee how mah" (overall absolute error = 12)
4. "Charlie's angles" (overall absolute error = 14)
5. "Big Wang Theory", "The Awesom 4some" and "Seven" (overall absolute error = 15)

# Welcome back!

The Big Bang Theory: prizes for smallest absolute error!

1. “Danielle & Sarah” *and* “A.Epp” (overall absolute error = 9)
2. “Integer Milan” (overall absolute error = 11)
3. “Nee how mah” (overall absolute error = 12)
4. “Charlie’s angles” (overall absolute error = 14)
5. “Big Wang Theory”, “The Awesom 4some” *and* “Seven” (overall absolute error = 15)

# Welcome back!

The Big Bang Theory: prizes for smallest absolute error!

1. “Danielle & Sarah” and “A.Epp” (overall absolute error = 9)
2. “Integer Milan” (overall absolute error = 11)
3. “Nee how mah” (overall absolute error = 12)
4. “Charlie’s angles” (overall absolute error = 14)
5. “Big Wang Theory”, “The Awesom 4some” and “Seven” (overall absolute error = 15)

# Welcome back!

The Big Bang Theory: prizes for smallest absolute error!

1. “Danielle & Sarah” *and* “A.Epp” (overall absolute error = 9)
2. “Integer Milan” (overall absolute error = 11)
3. “Nee how mah” (overall absolute error = 12)
4. “Charlie’s angles” (overall absolute error = 14)
5. “Big Wang Theory”, “The Awesom 4some” *and* “Seven” (overall absolute error = 15)

# Welcome back!

The Big Bang Theory: prizes for smallest absolute error!

1. “Danielle & Sarah” *and* “A.Epp” (overall absolute error = 9)
2. “Integer Milan” (overall absolute error = 11)
3. “Nee how mah” (overall absolute error = 12)
4. “Charlie’s angles” (overall absolute error = 14)
5. “Big Wang Theory”, “The Awesom 4some” *and* “Seven” (overall absolute error = 15)

# Welcome back!

The Big Bang Theory: prizes for smallest absolute error!

1. “Danielle & Sarah” *and* “A.Epp” (overall absolute error = 9)
2. “Integer Milan” (overall absolute error = 11)
3. “Nee how mah” (overall absolute error = 12)
4. “Charlie’s angles” (overall absolute error = 14)
5. “Big Wang Theory”, “The Awesom 4some” *and* “Seven” (overall absolute error = 15)

# Part IV

## Control Statements and Functions

## 4.1.1 Basic functions

This function takes in a single argument  $x$  and returns  $x^2$ :

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

## 4.1.1 Basic functions

The key elements in the function call are:

- The word `function`;
- The brackets `()` which enclose the **argument** list. This list may be empty.
- A sequence of statements in curly braces `{ }`.
- A return statement.

## 4.1.1 Basic functions

The key elements in the function call are:

- The word `function`;
- The brackets `()` which enclose the **argument** list. This list may be empty.
- A sequence of statements in curly braces `{ }`.
- A return statement.

## 4.1.1 Basic functions

The key elements in the function call are:

- The word `function`;
- The brackets `()` which enclose the **argument** list. This list may be empty.
- A sequence of statements in curly braces `{ }`.
- A return statement.

## 4.1.1 Basic functions

The key elements in the function call are:

- The word `function`;
- The brackets `()` which enclose the **argument** list. This list may be empty.
- A sequence of statements in curly braces `{ }`.
- A return statement.

## 4.1.1 Basic functions

We 'call' Fun1 in the following manner:

```
> Fun1(5)
[1] 25
```

```
> y = Fun1(10)
```

```
> y
[1] 100
```

```
> z = c(1, 2, 3, 4)
> Fun1(z)
[1] 1 4 9 16
```

## 4.1.1 Basic functions

We 'call' Fun1 in the following manner:

```
> Fun1(5)
```

```
[1] 25
```

```
> y = Fun1(10)
```

```
> y
```

```
[1] 100
```

```
> z = c(1, 2, 3, 4)
```

```
> Fun1(z)
```

```
[1] 1 4 9 16
```

## 4.1.1 Basic functions

We 'call' Fun1 in the following manner:

```
> Fun1(5)  
[1] 25
```

```
> y = Fun1(10)
```

```
> y  
[1] 100
```

```
> z = c(1, 2, 3, 4)  
> Fun1(z)  
[1] 1 4 9 16
```

## 4.1.1 Basic functions

We 'call' Fun1 in the following manner:

```
> Fun1(5)  
[1] 25
```

```
> y = Fun1(10)
```

```
> y  
[1] 100
```

```
> z = c(1, 2, 3, 4)  
> Fun1(z)  
[1] 1 4 9 16
```

## 4.1.1 Basic functions

Of course, the old saying 'Garbage in, garbage out' is true:

```
> Fun1()  
Error in Fun1() : argument "x" is missing, no default
```

```
> Fun1("5")  
Error in x * x : non-numeric argument to binary  
operator
```

# Question 1

What is the value of y?

```
> fun1 = function(x) {  
+     return(2*x)  
+ }  
> y = fun1(2)
```

- 5
- 10
- 2
- 4

# Question 1

What is the value of y?

```
> x = 5  
> fun1 = function(x) {  
+       return(2*x)  
+ }  
> (y= fun1(2))  
[1] 4
```

- 5
- 10
- 2
- 4

## Question 2

What is the value of y?

```
> fun1 = function(x) {  
+   z = 3  
+   return(x*z)  
+ }  
> y = fun1(2)
```

- 2
- 3
- 4
- 6

## Question 2

What is the value of y?

```
> fun1 = function(x) {  
+   z = 3  
+   return(x*z)  
+ }  
> (y = fun1(2))  
[1] 6
```

- 2
- 3
- 4
- 6

## Question 3

What is the value of y?

```
> fun1 = function(x) {  
+   z = 3  
+   return(x*z)  
+ }  
> k = 5  
> y = fun1(k)
```

## Question 3

What is the value of y?

```
> fun1 = function(x) {  
+   z = 3  
+   return(x*z)  
+ }  
> k = 5  
> (y = fun1(k))  
[1] 15
```

## 4.1.1 Basic functions

Other variations to this simple function are:

```
> Fun2 = function(x=1) {  
+   return (x*x)  
+ }
```

```
> Fun2()  
[1] 1
```

```
> Fun2(4)  
[1] 16
```

## 4.1.1 Basic functions

Other variations to this simple function are:

```
> Fun2 = function(x=1) {  
+   return (x*x)  
+ }
```

```
> Fun2()  
[1] 1
```

```
> Fun2(4)  
[1] 16
```

## 4.1.1 Basic functions

Other variations to this simple function are:

```
> Fun2 = function(x=1) {  
+   return (x*x)  
+ }
```

```
> Fun2()  
[1] 1
```

```
> Fun2(4)  
[1] 16
```

## 4.1.1 Basic functions

```
> Fun3 = function(x, y) {  
+   return (x*y)  
+ }
```

```
> Fun3(3, 4)  
[1] 12
```

## 4.1.1 Basic functions

```
> Fun3 = function(x, y) {  
+   return (x*y)  
+ }
```

```
> Fun3(3, 4)  
[1] 12
```

## Question 4

What is the value of y?

```
> fun1 = function(x=3) {  
+     return(2*x)  
+ }  
> y = fun1(2)
```

- ☒ 3
- ☐ 4
- ☐ 6

## Question 4

What is the value of y?

```
> fun1 = function(x=3) {  
+     return(2*x)  
+ }  
> (y = fun1(2))  
[1] 4
```

- 3
- 4
- 6

## Question 5

What is the value of y?

```
> fun1 = function(x=3) {  
+   return(2*x)  
+ }  
> y = fun1()
```

- ☒ 3
- ☐ 4
- ☐ 6

## Question 5

What is the value of y?

```
> fun1 = function(x=3) {  
+   return(2*x)  
+ }  
> (y = fun1())  
[1] 6
```

- 3
- 4
- 6

## Question 6

What is the value of y?

```
> fun1 = function(x=3) {  
+   x = 2  
+   return(2*x)  
+ }  
> y = fun1()
```

- 3
- 4
- 6

## Question 6

What is the value of y?

```
> fun1 = function(x=3) {  
+   x = 2  
+   return(2*x)  
+ }  
> (y = fun1())  
[1] 4
```

- 3
- 4
- 6

## 4.1.2 A more useful function

Here the function below takes in a vector, plots a histogram and returns a vector containing the mean and standard deviation:

```
> Investigate = function(values) {  
+   hist(values)  
+   m_std = c(mean(values), sd(values))  
+   return(m_std)  
+ }
```

## 4.1.2 A more useful function

Once we have created our function, we can put it to good use:

```
> Investigate(movies$Rating)
[1] 5.522715 1.451864
```

```
> Investigate(movies$Length)
[1] 100.87807 17.34152
```

```
> Investigate(movies$Budget)
[1] 10286893 23039711
```

## 4.1.2 A more useful function

Once we have created our function, we can put it to good use:

```
> Investigate(movies$Rating)
[1] 5.522715 1.451864
```

```
> Investigate(movies$Length)
[1] 100.87807 17.34152
```

```
> Investigate(movies$Budget)
[1] 10286893 23039711
```

## 4.1.2 A more useful function

Once we have created our function, we can put it to good use:

```
> Investigate(movies$Rating)
[1] 5.522715 1.451864
```

```
> Investigate(movies$Length)
[1] 100.87807 17.34152
```

```
> Investigate(movies$Budget)
[1] 10286893 23039711
```

## 4.1.3 Variable scope

## 4.1.3 Variable scope

- When we call a function, R first looks for *local* variables, then *global* variables.
- For example, Fun4 uses a global variable:

```
> blob = 5  
> Fun4 = function() {  
+   return(blob)  
+ }
```

```
> Fun4()  
[1] 5
```

## 4.1.3 Variable scope

- When we call a function, R first looks for *local* variables, then *global* variables.
- For example, Fun4 uses a global variable:

```
> blob = 5  
> Fun4 = function() {  
+   return(blob)  
+ }
```

```
> Fun4()  
[1] 5
```

## 4.1.3 Variable scope

But Fun5 uses a local variable:

```
> blob = 5  
> Fun5 = function() {  
+   blob = 6  
+   return(blob)  
+ }
```

```
> Fun5()  
[1] 6
```

```
> blob  
[1] 5
```

## 4.1.3 Variable scope

But Fun5 uses a local variable:

```
> blob = 5  
> Fun5 = function() {  
+   blob = 6  
+   return(blob)  
+ }
```

```
> Fun5()  
[1] 6
```

```
> blob  
[1] 5
```

## 4.1.3 Variable scope

But Fun5 uses a local variable:

```
> blob = 5  
> Fun5 = function() {  
+   blob = 6  
+   return(blob)  
+ }
```

```
> Fun5()  
[1] 6
```

```
> blob  
[1] 5
```

## Question 7

What is the value of y?

```
> x = 1  
> fun1 = function(x) {  
+   x = 2  
+   return(x)  
+ }  
> y= fun1(3)
```

- 1
- 2
- 3

## Question 7

What is the value of y?

```
> x = 1
> fun1 = function(x) {
+   x = 2
+   return(x)
+ }
> (y= fun1(3))
[1] 2
```

- 1
- 2
- 3

## Question 8

What is the value of y?

```
> x = 1  
> fun1 = function(x=3) {  
+   x = 2  
+   return(x)  
+ }  
> y= fun1()
```

- 1
- 2
- 3

## Question 8

What is the value of y?

```
> x = 1  
> fun1 = function(x=3) {  
+   x = 2  
+   return(x)  
+ }  
> (y= fun1())  
[1] 2
```

- 1
- 2
- 3

## Question 9

What is the value of y?

```
> x = 1  
> fun1 = function(x=3) {  
+   return(2*x)  
+ }  
> x = fun1(x)  
> y = fun1(x)
```

## Question 9

What is the value of y?

```
> x = 1
> fun1 = function(x=3) {
+   return(2*x)
+ }
> x = fun1(x)
> (y = fun1(x))
[1] 4
```

## 4.2 Conditionals

- Conditional statements are features of a programming language which perform different computations or actions depending on whether a condition evaluates to TRUE or FALSE.
- They are used in almost all computer programs.

## 4.2.1 If statements

The basic structure of an `if` statement is:

```
> if(expr) {  
+   ##do something  
+ }
```

where `expr` is evaluated to be either `TRUE` or `FALSE`.

## 4.2.1 If statements

The following example illustrates `if` statements in R:

```
> x = 5  
> y = 5  
> if(x<5) {  
+   y = 0  
+ }
```

```
> y  
[1] 5
```

In this code chunk, `x < 5` evaluates to be `FALSE` so the following brackets are not evaluated.

## 4.2.1 If statements

The following example illustrates `if` statements in R:

```
> x = 5  
> y = 5  
> if(x<5) {  
+   y = 0  
+ }
```

```
> y  
[1] 5
```

In this code chunk, `x < 5` evaluates to be `FALSE` so the following brackets are not evaluated.

## 4.2.1 If statements

We test for greater than in a similar manner:

```
> x = 5
> y = 5
> if(x > 0) {
+   y = 0
+ }
> y
[1] 0
```

Here  $x > 0$  evaluates to be TRUE so,  $y$  is set equal to 0. If we wanted to test for equality with zero, then we would use  $=$ .

## 4.3.2 If else statements

We can link together a number of if statements:

```
> x = 0
> if(x > 0) {
+   cat("x is greater than zero")
+ } else if(x < 0) {
+   cat("x is less than zero")
+ } else {
+   cat("x must be zero!")
+   cat("\n")
+ }
x must be zero!
```

The final else is optional.

## 4.2.2 If else statements

```
> IsNegative = function(value) {  
+   is_neg = FALSE  
+   if(value < 0) {  
+     is_neg = TRUE  
+   }  
+   return(is_neg)  
+ }
```

```
> IsNegative(1)  
[1] FALSE
```

```
> IsNegative(-4.6)  
[1] TRUE
```

## 4.2.2 If else statements

```
> IsNegative = function(value) {  
+   is_neg = FALSE  
+   if(value < 0) {  
+     is_neg = TRUE  
+   }  
+   return(is_neg)  
+ }
```

```
> IsNegative(1)  
[1] FALSE
```

```
> IsNegative(-4.6)  
[1] TRUE
```

## 4.2.2 If else statements

```
> IsNegative = function(value) {  
+   is_neg = FALSE  
+   if(value < 0) {  
+     is_neg = TRUE  
+   }  
+   return(is_neg)  
+ }
```

```
> IsNegative(1)  
[1] FALSE
```

```
> IsNegative(-4.6)  
[1] TRUE
```

## 4.2.2 If else statements

A more sophisticated function could be:

```
> IsGreaterThan = function(value1, value2) {  
+   is_greater_than = FALSE  
+   if(value1 > value2) {  
+     is_greater_than = TRUE  
+   }  
+   return(is_greater_than)  
+ }
```

## 4.2.2 If else statements

```
> IsGreaterThan(-5, -6)  
[1] TRUE
```

```
> IsGreaterThan(10, 10)  
[1] FALSE
```

## 4.2.2 If else statements

```
> IsGreaterThan(-5, -6)  
[1] TRUE
```

```
> IsGreaterThan(10, 10)  
[1] FALSE
```

# Question 1

What is the value of y?

```
> x = 5  
> if(x==6) {  
+   y = FALSE  
+ } else {  
+   y = TRUE  
+ }
```

- TRUE
- FALSE

# Question 1

What is the value of y?

```
> x = 5
> if(x==6) {
+   y = FALSE
+ } else {
+   y = TRUE
+ }
> y
[1] TRUE
```

- TRUE
- FALSE

## Question 2

What is the value of y?

```
> x = 5  
> if(x > 6){  
+   y = 0  
+ } else if(x >= 5) {  
+   y = 1  
+ } else {  
+   y = 2  
+ }
```

- 0
- 1
- 2

## Question 2

What is the value of y?

```
> x = 5
> if(x > 6){
+   y = 0
+ }else if(x >= 5) {
+   y = 1
+ } else {
+   y = 2
+ }
> y
[1] 1
```

- 0
- 1
- 2

## Question 3

What is the value of z?

```
> x = 5
> y = "male"
> if(x > 6 & y == "Female"){
+   z = 0
+ }else if(x < 5 & y == "Male") {
+   z = 1
+ } else {
+   z = 2
+ }
```

- 0
- 1
- 2

## Question 3

What is the value of z?

```
> x = 5
> y = "male"
> if(x > 6 & y == "Female"){
+   z = 0
+ }else if(x < 5 & y == "Male") {
+   z = 1
+ } else {
+   z = 2
+ }
> z
[1] 2
```

- 0
- 1
- 2

## 4.3 Control statements

- At times we would like to perform some operation on a vector or a data frame.
- Often R has built-in functions that will do this for you, e.g. mean, sd, other times we have to write our own functions.
- For example, suppose we want to calculate  $\sum_{i=1}^{10} i^2$ .
- In R we can use a for loop:

```
> x = 0  
> for(i in 1:10) {  
+   x = x + i^2  
+ }
```

```
> x  
[1] 385
```

## 4.3 Control statements

- At times we would like to perform some operation on a vector or a data frame.
- Often R has built-in functions that will do this for you, e.g. mean, sd, other times we have to write our own functions.
- For example, suppose we want to calculate  $\sum_{i=1}^{10} i^2$ .
- In R we can use a for loop:

```
> x = 0  
> for(i in 1:10) {  
+   x = x + i^2  
+ }
```

```
> x  
[1] 385
```

## 4.3 Control statements

$$\sum_{j=-5}^{-1} e^j / j^2 ,$$

```
> total = 0  
> for(j in -5:-1) {  
+   total = total + exp(j)/j^2  
+ }
```

```
> total  
[1] 0.4086594
```

## 4.3 Control statements

$$\sum_{j=-5}^{-1} e^j / j^2 ,$$

```
> total = 0  
> for(j in -5:-1) {  
+   total = total + exp(j)/j^2  
+ }
```

```
> total  
[1] 0.4086594
```

## 4.3 Control statements

A more tricky example is: calculate  $\sum e^k/k^2$ , for  $k = 3, 6, 9, \dots, 21$ :

```
> total = 0
> for(i in 1:7) {
+   k = i*3
+   total = total + exp(k)/k^2
+ }
> total
[1] 3208939
```

## Question 4

What is the value of x?

```
> total = 0  
> for(blob in 1:4) {  
+   total = total + 1  
+ }  
> x = total
```

## Question 4

What is the value of x?

```
> total = 0  
> for(blob in 1:4) {  
+   total = total + 1  
+ }  
> (x = total)  
[1] 4
```

## Question 5

What is the value of x?

```
> total = 0
> for(blob in 1:10) {
+   if(blob > 9) {
+     total = total + 1
+   }
+ }
> x = total
```

## Question 5

What is the value of x?

```
> total = 0
> for(blob in 1:10) {
+   if(blob > 9) {
+     total = total + 1
+   }
+ }
> (x = total)
[1] 1
```

## 4.4 Putting it all together

Rather than have to constantly write R code to solve the summations in §4.3 we can create a function to solve the general form:

$$\sum_{i=i_s}^{i_e} \frac{e^j}{j^2} \quad \text{for } i = i_s, i_s + j, i_s + 2j, \dots, i_e.$$

## 4.4 Putting it all together

So in R we have:

```
> Summation1 = function(i_s, i_e, j) {  
+   total = 0  
+   for(i in i_s:(i_e/j)) {  
+     k = i*j  
+     total = total + exp(k)/k^2  
+   }  
+   return(total)  
+ }
```

```
> Summation1(-5, -1, 1)  
[1] 0.4086594
```

```
> Summation1(3, 21, 3)  
[1] 3208925
```

## 4.4 Putting it all together

So in R we have:

```
> Summation1 = function(i_s, i_e, j) {  
+   total = 0  
+   for(i in i_s:(i_e/j)) {  
+     k = i*j  
+     total = total + exp(k)/k^2  
+   }  
+   return(total)  
+ }
```

```
> Summation1(-5, -1, 1)  
[1] 0.4086594
```

```
> Summation1(3, 21, 3)  
[1] 3208925
```

## 4.4 Putting it all together

So in R we have:

```
> Summation1 = function(i_s, i_e, j) {  
+   total = 0  
+   for(i in i_s:(i_e/j)) {  
+     k = i*j  
+     total = total + exp(k)/k^2  
+   }  
+   return(total)  
+ }
```

```
> Summation1(-5, -1, 1)  
[1] 0.4086594
```

```
> Summation1(3, 21, 3)  
[1] 3208925
```

## 4.5.1 The apply function

- We use the `apply` function when we want to *apply* the same function to every row or column of a data frame.
- For example, suppose we have a data frame with three columns:

```
> (df4 = data.frame(c1 = 1:4, c2 = 4:7, c3 = 2:5))
```

|   | c1 | c2 | c3 |
|---|----|----|----|
| 1 | 1  | 4  | 2  |
| 2 | 2  | 5  | 3  |
| 3 | 3  | 6  | 4  |
| 4 | 4  | 7  | 5  |

## 4.5.1 The apply function

- The `apply` function takes (at least) three arguments.
- The first argument is the data frame, the second the number 1 or 2 indicating row or column and the third a function to apply to each row or column.
- So

```
> apply(df4, 1, mean)
[1] 2.333333 3.333333 4.333333 5.333333
```

calculates the mean value of every row.

## 4.5.1 The apply function

What about the standard deviation for every column? Easy!

```
> apply(df4, 2, sd)
      c1      c2      c3
1.290994 1.290994 1.290994
```

## 4.5.1 The apply function

What about the standard deviation for every column? Easy!

```
> apply(df4, 2, sd)
      c1      c2      c3
1.290994 1.290994 1.290994
```

Suppose one of the columns was non-numeric:

```
> (df5 = data.frame(c1 = 1:3, c2 = 4:6,  
+   c3 = LETTERS[1:3]))  
  c1 c2 c3  
1  1  4  A  
2  2  5  B  
3  3  6  C
```

Then taking the mean doesn't really make sense:

```
> apply(df5, 1, mean)  
[1] NA NA NA
```

Instead, we remove the column, then calculate the mean:

```
> apply(df5[,1:2], 1, mean)  
[1] 2.5 3.5 4.5
```

Suppose one of the columns was non-numeric:

```
> (df5 = data.frame(c1 = 1:3, c2 = 4:6,  
+   c3 = LETTERS[1:3]))  
  c1 c2 c3  
1  1  4  A  
2  2  5  B  
3  3  6  C
```

Then taking the mean doesn't really make sense:

```
> apply(df5, 1, mean)  
[1] NA NA NA
```

Instead, we remove the column, then calculate the mean:

```
> apply(df5[,1:2], 1, mean)  
[1] 2.5 3.5 4.5
```

## Question 6

What type of variable is z?

```
> apply(z, 1, mean)
```

- Data frame
- Double
- String
- logical

## Question 6

What type of variable is z?

```
> apply(z, 1, mean)
```

- Data frame
- Double
- String
- logical

## Question 7

Which of these statements is correct?

```
> apply(z, 2, median)
```

- We calculate the median of z
- We calculate the median for each row of z
- We calculate the median for each column of z
- We calculate the mean of z

## Question 7

Which of these statements is correct?

```
> apply(z, 2, median)
```

- We calculate the median of z
- We calculate the median for each row of z
- We calculate the median for each column of z
- We calculate the mean of z

## 4.5.2 The tapply function

- The function `tapply` is very useful, and is a natural extension of `apply`
- It's best described using an example:

```
> tapply(movies$Length, movies$mpaa, mean)
      NC-17      PG      PG-13      R 
110.18750  97.23384 104.97877 100.18818
```

- In the above code, we have calculated the average movie length conditional on its MPAA rating.
- So the average length of a PG movie is 97 minutes and the average NC-17 movie length is 110 minutes
- With `tapply` we can do we very interesting things...

## 4.5.2 The `tapply` function

- The function `tapply` is very useful, and is a natural extension of `apply`
- It's best described using an example:

```
> tapply(movies$Length, movies$mpaa, mean)
      NC-17      PG      PG-13      R 
110.18750  97.23384 104.97877 100.18818
```

- In the above code, we have calculated the average movie length conditional on its MPAA rating.
- So the average length of a PG movie is 97 minutes and the average NC-17 movie length is 110 minutes
- With `tapply` we can do we very interesting things...

## 4.5.2 The `tapply` function

- The function `tapply` is very useful, and is a natural extension of `apply`
- It's best described using an example:

```
> tapply(movies$Length, movies$mpaa, mean)
      NC-17      PG      PG-13      R
110.18750  97.23384 104.97877 100.18818
```

- In the above code, we have calculated the average movie length conditional on its MPAA rating.
- So the average length of a PG movie is 97 minutes and the average NC-17 movie length is 110 minutes
- With `tapply` we can do we very interesting things...

## 4.5.2 The `tapply` function

- The function `tapply` is very useful, and is a natural extension of `apply`
- It's best described using an example:

```
> tapply(movies$Length, movies$mpaa, mean)
      NC-17      PG      PG-13      R
110.18750  97.23384 104.97877 100.18818
```

- In the above code, we have calculated the average movie length conditional on its MPAA rating.
- So the average length of a PG movie is 97 minutes and the average NC-17 movie length is 110 minutes
- With `tapply` we can do we very interesting things...

## 4.5.2 The `tapply` function

- For example, in the next piece of code, we plot the average movie length conditional on its rating:

```
> tapply(movies$Length, movies$Rating, mean)[1:6]
 1  1.2  1.3  1.4  1.5  1.6
85.5 93.0 87.5 85.0 67.0 86.0
```

```
> rating_by_len = tapply(movies$Length, movies$Rating,
+   mean)
> plot(names(rating_by_len), rating_by_len)
```

## 4.5.2 The `tapply` function

- For example, in the next piece of code, we plot the average movie length conditional on its rating:

```
> tapply(movies$Length, movies$Rating, mean)[1:6]
      1  1.2  1.3  1.4  1.5  1.6 
85.5 93.0 87.5 85.0 67.0 86.0
```

```
> rating_by_len = tapply(movies$Length, movies$Rating,
+   mean)
> plot(names(rating_by_len), rating_by_len)
```

## 4.5.2 The tapply function

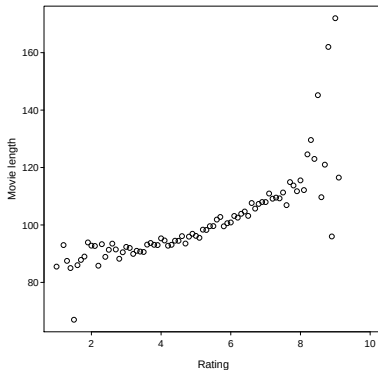


Figure: Plot of movie length conditional on it's rating.

## 4.6 Help

- R has a very good help system.
- If you need information about a particular function, say `plot`.
- Then typing `?plot` in a R terminal will bring up the associated help page.
- The internet is another very good source of R help.
- Unfortunately, using Google isn't particularly useful since the letter "R" appears on most web pages!
- However, you can use

`http://www.rseek.org/`

Using this search engine limits searches to R web-pages.

## Part V

# Random Number Generation

# Some thoughts...

- When writing big chunks of code, indent nicely! Generally, it is good practice for your code to be structured (bug-spotting, typos etc.). Certainly don't type everything onto a single line.
- `if/else` statements – feel free to put each `if` on a single line – e.g.:
- Plus symbol on the console window – R is expecting something else – most likely a close bracket!
- Running functions – e.g. `IsEven(5)`
- Commenting in code is very important! Use the `#` symbol to let us know what you're doing:

# Some thoughts...

- When writing big chunks of code, indent nicely! Generally, it is good practice for your code to be structured (bug-spotting, typos etc.). Certainly don't type everything onto a single line.
- `if/else` statements – feel free to put each `if` on a single line – e.g.:
- Plus symbol on the console window – R is expecting something else – most likely a close bracket!
- Running functions – e.g. `IsEven(5)`
- Commenting in code is very important! Use the `#` symbol to let us know what you're doing:

## Some thoughts...

- When writing big chunks of code, indent nicely! Generally, it is good practice for your code to be structured (bug-spotting, typos etc.). Certainly don't type everything onto a single line.
- `if/else` statements – feel free to put each `if` on a single line – e.g.:
- Plus symbol on the console window – R is expecting something else – most likely a close bracket!
- Running functions – e.g. `IsEven(5)`
- Commenting in code is very important! Use the `#` symbol to let us know what you're doing:

## Some thoughts...

- When writing big chunks of code, indent nicely! Generally, it is good practice for your code to be structured (bug-spotting, typos etc.). Certainly don't type everything onto a single line.
- `if/else` statements – feel free to put each `if` on a single line – e.g.:
- Plus symbol on the console window – R is expecting something else – most likely a close bracket!
- Running functions – e.g. `IsEven(5)`
- Commenting in code is very important! Use the `#` symbol to let us know what you're doing:

# Some reminders...

- Practical 2 currently being marked, won't be long...
- Practical 3 due in **today, 11.45pm**
- This Friday's practical: Group project time! Make sure you sit with the other members of your group – we will try to shepherd you to an area
- Next week's lecture: finish the syllabus, plus Group projects hints!

## Some reminders...

- Practical 2 currently being marked, won't be long...
- Practical 3 due in **today, 11.45pm**
- This Friday's practical: Group project time! Make sure you sit with the other members of your group – we will try to shepherd you to an area
- Next week's lecture: finish the syllabus, plus Group projects hints!

# Some reminders...

- Practical 2 currently being marked, won't be long...
- Practical 3 due in **today, 11.45pm**
- This Friday's practical: Group project time! Make sure you sit with the other members of your group – we will try to shepherd you to an area
- Next week's lecture: finish the syllabus, plus Group projects hints!

## Some reminders...

- Practical 2 currently being marked, won't be long...
- Practical 3 due in **today, 11.45pm**
- This Friday's practical: Group project time! Make sure you sit with the other members of your group – we will try to shepherd you to an area
- Next week's lecture: finish the syllabus, plus Group projects hints!

## 5.1 Randomness: quantifying uncertainty

- The concepts of uncertainty and randomness have intrigued humanity for a long time:
  - Chinese: formalised odds/chance some 3000 years ago
  - Greek philosophers discussed randomness at length
  - 16th Century: Italian mathematicians formalised odds associated with games of chance
  - 19th Century: Physics
  - Early 20th Century: Rapid growth in the formal analysis of randomness

## 5.1 Randomness: quantifying uncertainty

- The concepts of uncertainty and randomness have intrigued humanity for a long time:
  - Chinese: formalised odds/chance some 3000 years ago
  - Greek philosophers discussed randomness at length
  - 16th Century: Italian mathematicians formalised odds associated with games of chance
  - 19th Century: Physics
  - Early 20th Century: Rapid growth in the formal analysis of randomness

## 5.1 Randomness: quantifying uncertainty

- The concepts of uncertainty and randomness have intrigued humanity for a long time:
  - Chinese: formalised odds/chance some 3000 years ago
  - Greek philosophers discussed randomness at length
  - 16th Century: Italian mathematicians formalised odds associated with games of chance
  - 19th Century: Physics
  - Early 20th Century: Rapid growth in the formal analysis of randomness

## 5.1 Randomness: quantifying uncertainty

- The concepts of uncertainty and randomness have intrigued humanity for a long time:
  - Chinese: formalised odds/chance some 3000 years ago
  - Greek philosophers discussed randomness at length
  - 16th Century: Italian mathematicians formalised odds associated with games of chance
  - 19th Century: Physics
  - Early 20th Century: Rapid growth in the formal analysis of randomness

## 5.1 Randomness: quantifying uncertainty

- The concepts of uncertainty and randomness have intrigued humanity for a long time:
  - Chinese: formalised odds/chance some 3000 years ago
  - Greek philosophers discussed randomness at length
  - 16th Century: Italian mathematicians formalised odds associated with games of chance
  - 19th Century: Physics
  - Early 20th Century: Rapid growth in the formal analysis of randomness

## 5.1 Randomness: quantifying uncertainty

- The concepts of uncertainty and randomness have intrigued humanity for a long time:
  - Chinese: formalised odds/chance some 3000 years ago
  - Greek philosophers discussed randomness at length
  - 16th Century: Italian mathematicians formalised odds associated with games of chance
  - 19th Century: Physics
  - Early 20th Century: Rapid growth in the formal analysis of randomness

## 5.1 Randomness: quantifying uncertainty

- The concepts of uncertainty and randomness have intrigued humanity for a long time:
  - Chinese: formalised odds/chance some 3000 years ago
  - Greek philosophers discussed randomness at length
  - 16th Century: Italian mathematicians formalised odds associated with games of chance
  - 19th Century: Physics
  - Early 20th Century: Rapid growth in the formal analysis of randomness

## 5.1 Randomness: quantifying uncertainty

- The world around us is not deterministic and we are faced continually with chance occurrences.
- Uncertainty is inherent in nature; for example, the behaviour of fundamental physical particles, genes and chromosomes in biology, and individuals in society under stress or strain.
- The methodology for exploring uncertainty involves the use of *random numbers*.

## 5.2 Pseudo-random numbers

Suppose we need to obtain a list of random digits 0, 1, 2, . . . , 9. How might we go about this? There are several options:

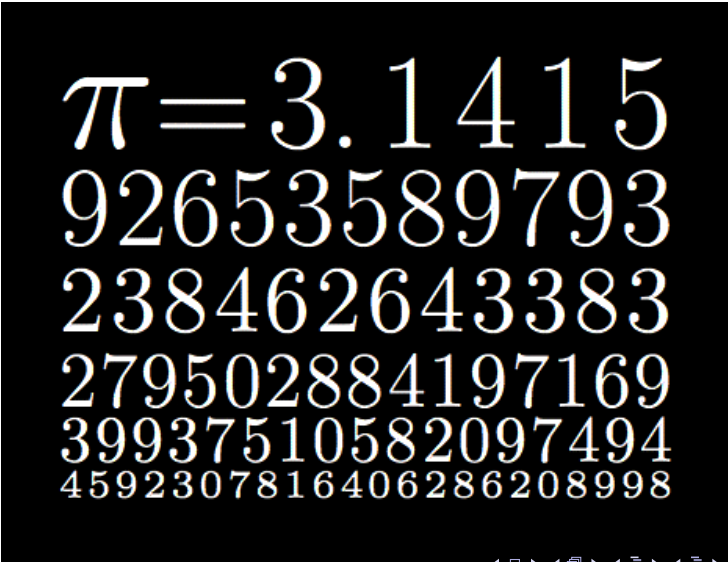
## Fair ten-sided die

If the sides are labelled from 0 to 9 then tosses of this die will yield the required digits.



# Decimal expansion of $\pi$

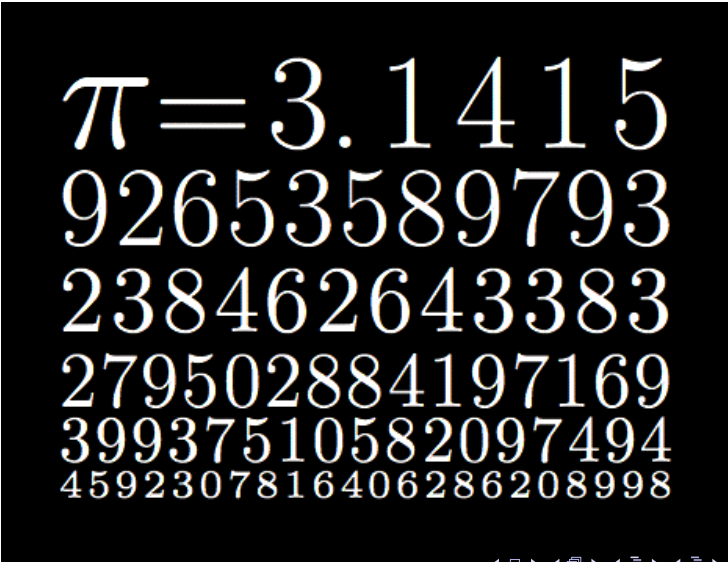
Irrational number – decimal expansion goes on forever, with no pattern!



$\pi = 3.1415$   
92653589793  
238462643383  
279502884197169  
39937510582097494  
4592307816406286208998

# Decimal expansion of $\pi$

Irrational number – decimal expansion goes on forever, with no pattern!



$\pi = 3.1415$   
92653589793  
238462643383  
279502884197169  
39937510582097494  
4592307816406286208998

# Tosses of a fair coin

Toss a fair coin four times. The following equally likely outcomes could correspond to the integers shown:

|      |   |      |   |
|------|---|------|---|
| HHHH | 0 | HTHT | 5 |
| HHHT | 1 | HTTH | 6 |
| HHTH | 2 | HTTT | 7 |
| HHTT | 3 | THHH | 8 |
| HTHH | 4 | THHT | 9 |

The coin has no 'memory', and so each block of 4 tosses is independent of any other. If any outcome other than those listed occurs, we can ignore and toss a new set of 4.

This method is rather inefficient as a lot of the time a combination of outcomes is rejected!

# Tosses of a fair coin

Toss a fair coin four times. The following equally likely outcomes could correspond to the integers shown:

|         |   |         |   |
|---------|---|---------|---|
| H H H H | 0 | H T H T | 5 |
| H H H T | 1 | H T T H | 6 |
| H H T H | 2 | H T T T | 7 |
| H H T T | 3 | T H H H | 8 |
| H T H H | 4 | T H H T | 9 |

The coin has no 'memory', and so each block of 4 tosses is independent of any other. If any outcome other than those listed occurs, we can ignore and toss a new set of 4.

This method is rather inefficient as a lot of the time a combination of outcomes is rejected!

# Tosses of a fair coin

Toss a fair coin four times. The following equally likely outcomes could correspond to the integers shown:

|         |   |         |   |
|---------|---|---------|---|
| H H H H | 0 | H T H T | 5 |
| H H H T | 1 | H T T H | 6 |
| H H T H | 2 | H T T T | 7 |
| H H T T | 3 | T H H H | 8 |
| H T H H | 4 | T H H T | 9 |

The coin has no 'memory', and so each block of 4 tosses is independent of any other. If any outcome other than those listed occurs, we can ignore and toss a new set of 4.

This method is rather inefficient as a lot of the time a combination of outcomes is rejected!

## Other physical devices: Wheels of fortune



## Other physical devices: Lottery machines



## Other physical devices: Gamma ray counters

- Quantum mechanics predicts that the nuclear decay of atoms is random
- **Idea:** Use a geiger counter to generate random numbers!

## Other physical devices: Gamma ray counters

- Quantum mechanics predicts that the nuclear decay of atoms is random
- **Idea:** Use a geiger counter to generate random numbers!

## 5.2 Pseudo-random numbers

- Mechanical and electronic devices are not reproducible...
- So we use *Pseudo-random numbers generators* (PRNG)

## 5.2 Pseudo-random numbers

- Mechanical and electronic devices are not reproducible...
- So we use *Pseudo-random numbers generators* (PRNG)

## 5.2 Pseudo-random numbers

The German Federal Office for Information Security (*Bundesamt für Sicherheit in der Informationstechnik*, or BSI) has established criteria for quality RNG:

- 1 A sequence of random numbers has a high probability of containing no identical consecutive elements
- 2 A sequence of numbers which is indistinguishable from 'true random' numbers (tested using statistical tests)
- 3 It should be impossible to calculate – or guess – from any given sub-sequence, any previous or future values in the sequence
- 4 It should be impossible, for all practical purposes, for an attacker to calculate, or guess, the values used in the random number algorithm

Points 3 and 4 are crucial for many applications.

## 5.2 Pseudo-random numbers

The German Federal Office for Information Security (*Bundesamt für Sicherheit in der Informationstechnik*, or BSI) has established criteria for quality RNG:

- 1 A sequence of random numbers has a high probability of containing no identical consecutive elements
- 2 A sequence of numbers which is indistinguishable from 'true random' numbers (tested using statistical tests)
- 3 It should be impossible to calculate – or guess – from any given sub-sequence, any previous or future values in the sequence
- 4 It should be impossible, for all practical purposes, for an attacker to calculate, or guess, the values used in the random number algorithm

Points 3 and 4 are crucial for many applications.

## 5.2 Pseudo-random numbers

As we will see in this course, it is important to be able to reproduce a sequence of random numbers.

The advent of digital computers gave rise to a number of techniques that use a **recursive relation**, that is, the number  $r_i$  in a sequence is a function of the preceding numbers  $r_{i-1}, r_{i-2}, \dots$

Some of these PRNG's have quite cool names: for example,

- the *Super-Duper* PRNG
- the *Blum-Blum-Shub* PRNG (after Lenore Blum, Manuel Blum and Michael Shub)
- the *Mersenne-Twister* PRNG (as used by R)

## 5.2 Pseudo-random numbers

As we will see in this course, it is important to be able to reproduce a sequence of random numbers.

The advent of digital computers gave rise to a number of techniques that use a **recursive relation**, that is, the number  $r_i$  in a sequence is a function of the preceding numbers  $r_{i-1}, r_{i-2}, \dots$

Some of these PRNG's have quite cool names: for example,

- the *Super-Duper* PRNG
- the *Blum-Blum-Shub* PRNG (after Lenore Blum, Manuel Blum and Michael Shub)
- the *Mersenne-Twister* PRNG (as used by R)

## 5.2 Pseudo-random numbers

As we will see in this course, it is important to be able to reproduce a sequence of random numbers.

The advent of digital computers gave rise to a number of techniques that use a **recursive relation**, that is, the number  $r_i$  in a sequence is a function of the preceding numbers  $r_{i-1}, r_{i-2}, \dots$

Some of these PRNG's have quite cool names: for example,

- the *Super-Duper* PRNG
- the *Blum-Blum-Shub* PRNG (after Lenore Blum, Manuel Blum and Michael Shub)
- the *Mersenne-Twister* PRNG (as used by R)

## 5.2 Pseudo-random numbers

As we will see in this course, it is important to be able to reproduce a sequence of random numbers.

The advent of digital computers gave rise to a number of techniques that use a **recursive relation**, that is, the number  $r_i$  in a sequence is a function of the preceding numbers  $r_{i-1}, r_{i-2}, \dots$

Some of these PRNG's have quite cool names: for example,

- the *Super-Duper* PRNG
- the *Blum-Blum-Shub* PRNG (after Lenore Blum, Manuel Blum and Michael Shub)
- the *Mersenne-Twister* PRNG (as used by R)

## 5.2 Pseudo-random numbers

As we will see in this course, it is important to be able to reproduce a sequence of random numbers.

The advent of digital computers gave rise to a number of techniques that use a **recursive relation**, that is, the number  $r_i$  in a sequence is a function of the preceding numbers  $r_{i-1}, r_{i-2}, \dots$

Some of these PRNG's have quite cool names: for example,

- the *Super-Duper* PRNG
- the *Blum-Blum-Shub* PRNG (after Lenore Blum, Manuel Blum and Michael Shub)
- the *Mersenne-Twister* PRNG (as used by R)

## 5.3.1 The `runif` function

```
> runif(n, min=0, max=1)
```

- This function will generate `n` random numbers between the values of `min` and `max`.
- If the arguments `min` or `max` are omitted, then the default values are 0 and 1 respectively.

## 5.3.1 The runif function

For example,

```
> runif(1)
[1] 0.2729965
> runif(1)
[1] 0.9990863
> runif(5)
[1] 0.8122783 0.9069950 0.1731072 0.3454292 0.6102412
> runif(1, 6, 7)
[1] 6.427512
```

## 5.3.1 The runif function

```
> set.seed(12345)
> runif(1)
[1] 0.7209039
> runif(1)
[1] 0.8757732
```

```
> set.seed(12345)
> runif(1)
[1] 0.7209039
```

## 5.3.1 The runif function

```
> set.seed(12345)
> runif(1)
[1] 0.7209039
> runif(1)
[1] 0.8757732
```

```
> set.seed(12345)
> runif(1)
[1] 0.7209039
```

## 5.3.2 The sample function

Another important R function that we will use is the sample function:

```
> sample(x, size, replace = FALSE, prob = NULL)
```

## 5.3.2 The sample function

This takes the following arguments

- `x`: a list of values
- `size`: non-negative integer giving the number of items to choose.
- `replace`: Should sampling be with replacement? Default: `FALSE`.
- `prob`: A vector of probability weights. Default: All values equally likely.

## 5.3.2 Example usage of sample

Suppose we wish to sample five numbers from  $\{1, 2, 3, 4, 5, 6\}$ , then

```
> set.seed(1)
> x = c(1, 2, 3, 4, 5, 6)
> sample(x, 5)
[1] 2 6 3 4 1
```

We can also sample with replacement:

```
> sample(x, 5, replace=TRUE)
[1] 6 6 4 4 1
```

This means that values **may** appear more than once.

### 5.3.3 Simulating the Capital One Cup draw

We are in the semi-finals of the *Capital One* Cup, and we need to organise the draw for the final stage. The remaining teams are:

Manchester Utd, Manchester City, Sunderland, West Ham.

## 5.3.3 Simulating the Capital One Cup draw

Here's how we do this in R:

```
> set.seed(3)
> teams = c("Man Utd", "Man City", "Sunderland", "West
Ham")
> sample(teams, 4)
[1] "Man Utd"      "Sunderland"  "West Ham"    "Man City"
```

So we have 'Man Utd vs Sunderland' and 'West Ham vs Man City'.

### 5.3.3 Simulating the Capital One Cup draw

However, if we think Sunderland are likely to get beaten by Man Utd, we can rig the voting:

```
> prob_weights = c(0.4, 0.4, 0.05, 0.2)
> sample(teams, 4, prob=prob_weights)
[1] "Man Utd"      "Man City"     "West Ham"
"Sunderland"
```

That's better!

## 5.3.3 Simulating the Capital One Cup draw



## 5.4 Simulating from discrete probability distributions

Suppose we want to simulate a discrete random variable  $X$  which has probability mass function

$$\Pr[X = x_j] = p_j \quad \text{for } j = 1, 2, \dots,$$

and where  $\sum_{\text{all}} p_j = 1$ .

## 5.4 Simulating from discrete probability distributions

To accomplish this we first simulate a value  $u$  from a  $U(0, 1)$  distribution and set

$$X = \begin{cases} x_1 & \text{if } u < p_1 \\ x_2 & \text{if } p_1 \leq u < p_1 + p_2 \\ x_3 & \text{if } p_1 + p_2 \leq u < p_1 + p_2 + p_3 \\ \vdots & \vdots \\ x_j & \text{if } \sum_{i=1}^{j-1} p_i \leq u < \sum_{i=1}^j p_i \\ \vdots & \vdots \end{cases}$$

## 5.4 Simulating from discrete probability distributions

To accomplish this we first simulate a value  $u$  from a  $U(0, 1)$  distribution and set

$$X = \begin{cases} x_1 & \text{if } u < p_1 \\ x_2 & \text{if } p_1 \leq u < p_1 + p_2 \\ x_3 & \text{if } p_1 + p_2 \leq u < p_1 + p_2 + p_3 \\ \vdots & \vdots \\ x_j & \text{if } \sum_{i=1}^{j-1} p_i \leq u < \sum_{i=1}^j p_i \\ \vdots & \vdots \end{cases}$$

## 5.4.1 Example

Simulate a random variable with the following probability mass function:

| $x$          | 1   | 2    | 3    | 4   |
|--------------|-----|------|------|-----|
| $\Pr[X = x]$ | 0.2 | 0.15 | 0.25 | 0.4 |

## 5.4.1 Solution

We calculate the CDF of the distribution:

| $x$             | 1   | 2    | 3    | 4   |
|-----------------|-----|------|------|-----|
| $\Pr[X = x]$    | 0.2 | 0.15 | 0.25 | 0.4 |
| $\Pr[X \leq x]$ | 0.2 | 0.35 | 0.60 | 1.0 |

Then we generate an observation  $u$  from  $U \sim \text{Uniform}(0, 1)$ , so

- if  $u < 0.2$ , set  $X = 1$ ;
- if  $0.2 \leq u < (0.2 + 0.15) = 0.35$ , set  $X = 2$ ;
- if  $0.35 \leq u < (0.2 + 0.15 + 0.25) = 0.6$ , set  $X = 3$ ;
- if  $u \geq 0.6$ , set  $X = 4$ .

Suppose  $u = 0.821$ . Then  $X = 4$ .

## 5.4.1 Solution

We calculate the CDF of the distribution:

| $x$             | 1   | 2    | 3    | 4   |
|-----------------|-----|------|------|-----|
| $\Pr[X = x]$    | 0.2 | 0.15 | 0.25 | 0.4 |
| $\Pr[X \leq x]$ | 0.2 | 0.35 | 0.60 | 1.0 |

Then we generate an observation  $u$  from  $U \sim \text{Uniform}(0, 1)$ , so

- if  $u < 0.2$ , set  $X = 1$ ;
- if  $0.2 \leq u < (0.2 + 0.15) = 0.35$ , set  $X = 2$ ;
- if  $0.35 \leq u < (0.2 + 0.15 + 0.25) = 0.6$ , set  $X = 3$ ;
- if  $u \geq 0.6$ , set  $X = 4$ .

Suppose  $u = 0.821$ . Then  $X = 4$ .

## 5.4.1 Solution

We calculate the CDF of the distribution:

| $x$             | 1   | 2    | 3    | 4   |
|-----------------|-----|------|------|-----|
| $\Pr[X = x]$    | 0.2 | 0.15 | 0.25 | 0.4 |
| $\Pr[X \leq x]$ | 0.2 | 0.35 | 0.60 | 1.0 |

Then we generate an observation  $u$  from  $U \sim \text{Uniform}(0, 1)$ , so

- if  $u < 0.2$ , set  $X = 1$ ;
- if  $0.2 \leq u < (0.2 + 0.15) = 0.35$ , set  $X = 2$ ;
- if  $0.35 \leq u < (0.2 + 0.15 + 0.25) = 0.6$ , set  $X = 3$ ;
- if  $u \geq 0.6$ , set  $X = 4$ .

Suppose  $u = 0.821$ . Then  $X = 4$ .

## 5.4.1 Solution

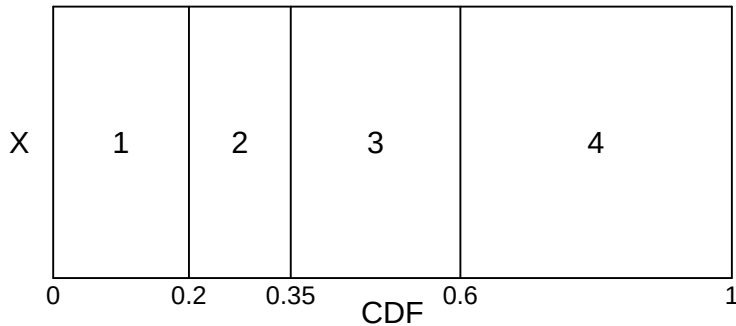


Figure: Simulating discrete random numbers.

## 5.4.1 Using R

To simulate from the above distribution in R, there are two (obvious) methods that we can use. We could use the `sample` command...

```
> x = c(1, 2, 3, 4)
> prob = c(0.2, 0.15, 0.25, 0.4)
> sum(prob)
[1] 1
> sample(x, 1, prob, replace=TRUE)
[1] 4
```

## 5.4.1 Using R

... or we could use a bunch of `if` statements (Monopoly!):

```
> u = runif(1)
> if(u <= 0.2) {
+   X = 1
+ } else if(u <= (0.2+0.15)) {
+   X = 2
+ } else if(u <= (0.2+0.15+0.25)) {
+   X = 3
+ } else {
+   X = 4
+ }
```

## 5.4.2 Simulating a Poisson random variable

For the uniform random numbers

0.253, 0.588, 0.789,

simulate three random numbers from a Poisson distribution with mean  $\lambda = 2$ .

## 5.4.2 Solution

The pdf of the Poisson distribution is

$$Pr[X = x] = \frac{e^{-\lambda} \lambda^x}{x!}$$

where  $\lambda > 0$ . So we have

| $x$            | 0     | 1     | 2     | 3     | 4     |
|----------------|-------|-------|-------|-------|-------|
| $Pr[X = x]$    | 0.135 | 0.271 | 0.271 | 0.180 | 0.090 |
| $Pr[X \leq x]$ | 0.135 | 0.406 | 0.677 | 0.857 | 0.947 |

## 5.4.2 Solution

The pdf of the Poisson distribution is

$$Pr[X = x] = \frac{e^{-\lambda} \lambda^x}{x!}$$

where  $\lambda > 0$ . So we have

| $x$            | 0     | 1     | 2     | 3     | 4     |
|----------------|-------|-------|-------|-------|-------|
| $Pr[X = x]$    | 0.135 | 0.271 | 0.271 | 0.180 | 0.090 |
| $Pr[X \leq x]$ | 0.135 | 0.406 | 0.677 | 0.857 | 0.947 |

## 5.4.2 Solution

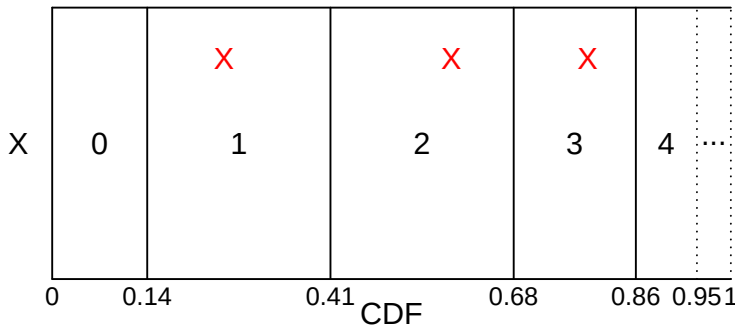


**Figure:** Diagram illustrating on simulating discrete random numbers from the Poisson distribution.

Hence our random numbers are 1, 2 and 3.

## 5.4.2 Using R

```
> rpois(1,2)
[1] 3
```

```
> rpois(10,2)
[1] 0 1 2 0 0 0 2 1 3 1
```

## 5.4.3 Simulating from other distributions

Simulating random values from other standard probability distributions is easy in R.

Just as we have seen the intrinsic functions `runif` and `rpois`, there are other functions such as `rgeom` and `rbinom`.

## 5.4.3 Simulating from other distributions

Simulating random values from other standard probability distributions is easy in R.

Just as we have seen the intrinsic functions `runif` and `rpois`, there are other functions such as `rgeom` and `rbinom`.

## 5.4.3 Simulating from other distributions

For example, suppose  $X \sim \text{Bin}(6, 0.2)$  and  $Y \sim \text{Geom}(0.75)$ , and we want to simulate 5 random values from  $X$  and 3 random values from  $Y$ :

```
> (X=rbinom(5,6,0.2))  
[1] 0 1 1 3 0
```

```
> (Y=rgeom(3,0.75))  
[1] 2 0 1
```

## 5.4.3 Variations on a theme

We have seen, for example, `runif`, `rpois`, `rbinom` and `rgeom`.

Replacing the `r` prefix with:

- `d`, calculates the **density** of the distribution at a particular value.  
For example, `dpois(1,2)` would find  $Pr[X = 1]$  when  $X \sim Po(2)$
- `p`, calculates the **distribution function** at a particular value.  
For example, `pbinom(2,6,0.2)` would find  $Pr[X \leq 2] = Pr[X = 0] + Pr[X = 1] + Pr[X = 2]$  when  $X \sim Bin(6, 0.2)$
- `q`, finds **quantiles** of a distribution for a particular probability.  
For example, `qnorm(0.7,0,1)` finds  $x$  such that  $Pr[X \leq x] = 0.75$ , when  $X \sim N(0, 1)$

## 5.4.3 Variations on a theme

We have seen, for example, `runif`, `rpois`, `rbinom` and `rgeom`.

Replacing the `r` prefix with:

- `d`, calculates the **density** of the distribution at a particular value.

For example, `dpois(1,2)` would find  $\Pr[X = 1]$  when  $X \sim \text{Po}(2)$

- `p`, calculates the **distribution function** at a particular value.

For example, `pbinom(2,6,0.2)` would find  $\Pr[X \leq 2] = \Pr[X = 0] + \Pr[X = 1] + \Pr[X = 2]$  when  $X \sim \text{Bin}(6, 0.2)$

- `q`, finds **quantiles** of a distribution for a particular probability.

For example, `qnorm(0.7,0,1)` finds  $x$  such that  $\Pr[X \leq x] = 0.75$ , when  $X \sim N(0, 1)$

## 5.4.3 Variations on a theme

We have seen, for example, `runif`, `rpois`, `rbinom` and `rgeom`.

Replacing the `r` prefix with:

- `d`, calculates the **density** of the distribution at a particular value.  
For example, `dpois(1,2)` would find  $Pr[X = 1]$  when  $X \sim Po(2)$
- `p`, calculates the **distribution function** at a particular value.  
For example, `pbinom(2,6,0.2)` would find  $Pr[X \leq 2] = Pr[X = 0] + Pr[X = 1] + Pr[X = 2]$  when  $X \sim Bin(6, 0.2)$
- `q`, finds **quantiles** of a distribution for a particular probability.  
For example, `qnorm(0.7,0,1)` finds  $x$  such that  $Pr[X \leq x] = 0.75$ , when  $X \sim N(0, 1)$

## 5.4.3 Variations on a theme

We have seen, for example, `runif`, `rpois`, `rbinom` and `rgeom`.

Replacing the `r` prefix with:

- `d`, calculates the **density** of the distribution at a particular value.  
For example, `dpois(1,2)` would find  $Pr[X = 1]$  when  $X \sim Po(2)$

- `p`, calculates the **distribution function** at a particular value.

For example, `pbinom(2,6,0.2)` would find  
 $Pr[X \leq 2] = Pr[X = 0] + Pr[X = 1] + Pr[X = 2]$  when  
 $X \sim Bin(6, 0.2)$

- `q`, finds **quantiles** of a distribution for a particular probability.  
For example, `qnorm(0.7,0,1)` finds  $x$  such that  $Pr[X \leq x] = 0.75$ ,  
when  $X \sim N(0,1)$

## 5.4.3 Variations on a theme

We have seen, for example, `runif`, `rpois`, `rbinom` and `rgeom`.

Replacing the `r` prefix with:

- `d`, calculates the **density** of the distribution at a particular value.  
For example, `dpois(1,2)` would find  $Pr[X = 1]$  when  $X \sim Po(2)$
- `p`, calculates the **distribution function** at a particular value.  
For example, `pbinom(2,6,0.2)` would find  $Pr[X \leq 2] = Pr[X = 0] + Pr[X = 1] + Pr[X = 2]$  when  $X \sim Bin(6, 0.2)$
- `q`, finds **quantiles** of a distribution for a particular probability.  
For example, `qnorm(0.7,0,1)` finds  $x$  such that  $Pr[X \leq x] = 0.75$ , when  $X \sim N(0, 1)$

## 5.4.3 Variations on a theme

We have seen, for example, `runif`, `rpois`, `rbinom` and `rgeom`.

Replacing the `r` prefix with:

- `d`, calculates the **density** of the distribution at a particular value.

For example, `dpois(1,2)` would find  $Pr[X = 1]$  when  $X \sim Po(2)$

- `p`, calculates the **distribution function** at a particular value.

For example, `pbinom(2,6,0.2)` would find

$Pr[X \leq 2] = Pr[X = 0] + Pr[X = 1] + Pr[X = 2]$  when  $X \sim Bin(6, 0.2)$

- `q`, finds **quantiles** of a distribution for a particular probability.

For example, `qnorm(0.7,0,1)` finds  $x$  such that  $Pr[X \leq x] = 0.75$ , when  $X \sim N(0, 1)$

## 5.4.3 Variations on a theme

We have seen, for example, `runif`, `rpois`, `rbinom` and `rgeom`.

Replacing the `r` prefix with:

- `d`, calculates the **density** of the distribution at a particular value.  
For example, `dpois(1,2)` would find  $Pr[X = 1]$  when  $X \sim Po(2)$
- `p`, calculates the **distribution function** at a particular value.  
For example, `pbinom(2,6,0.2)` would find  $Pr[X \leq 2] = Pr[X = 0] + Pr[X = 1] + Pr[X = 2]$  when  $X \sim Bin(6, 0.2)$
- `q`, finds **quantiles** of a distribution for a particular probability.  
For example, `qnorm(0.7,0,1)` finds  $x$  such that  $Pr[X \leq x] = 0.75$ , when  $X \sim N(0, 1)$

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly: Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by 4pm, Monday 18th April
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly:Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by **4pm, Monday 18th April**
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly:Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by **4pm, Monday 18th April**
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly:Sticker album – 70%:30%, ish...

# Announcements

- Last week of term – last lecture with me (for some of you...)
- EvaSys questionnaires
- Prize question winner: Alex Hayward!
- **Assignments**
  - Assignment 1 – feedback sheet – please download!
  - Assignment 2 – the Big Bang Theory quiz – ready to collect
  - Assignment 3 – deadline 23:45 last night – will be marked and ready to collect in around a weeks' time
- **Group project**
  - Last scheduled session tomorrow, 9–11
  - Some of you have yet to attend any of the sessions – last chance tomorrow!
  - If you feel like anyone in your group has not contributed, email me
  - WebPA for peer assessment
  - Work due in by **4pm, Monday 18th April**
  - Follow instructions carefully!
  - I will schedule some office hours in the first week back after Easter
  - Monopoly:Sticker album – 70%:30%, ish...

# Monopoly: Some hints

## • Debugging

- For question 2, do each part (a)–(h) one at a time, checking the graph at each stage...
- ...then, if you get something weird, you'll have more idea where you went wrong
- Each time you edit your code, do so in a different script, so you don't screw up the code that you know works!
- Indent your code – the Rstudio text editor usually does this automatically – indenting makes your code more structured and easier to spot where things might be wrong
- Check your brackets, and make sure each thing you add to the main code is in the right 'section'

# Monopoly: Some hints

## • Debugging

- For question 2, do each part (a)–(h) one at a time, checking the graph at each stage...
- ...then, if you get something weird, you'll have more idea where you went wrong
- Each time you edit your code, do so in a different script, so you don't screw up the code that you know works!
- Indent your code – the Rstudio text editor usually does this automatically – indenting makes your code more structured and easier to spot where things might be wrong
- Check your brackets, and make sure each thing you add to the main code is in the right 'section'

# Monopoly: Some hints

## • Debugging

- For question 2, do each part (a)–(h) one at a time, checking the graph at each stage...
- ...then, if you get something weird, you'll have more idea where you went wrong
- Each time you edit your code, do so in a different script, so you don't screw up the code that you know works!
- Indent your code – the Rstudio text editor usually does this automatically – indenting makes your code more structured and easier to spot where things might be wrong
- Check your brackets, and make sure each thing you add to the main code is in the right 'section'

# Monopoly: Some hints

## • Debugging

- For question 2, do each part (a)–(h) one at a time, checking the graph at each stage...
- ...then, if you get something weird, you'll have more idea where you went wrong
- Each time you edit your code, do so in a different script, so you don't screw up the code that you know works!
- Indent your code – the Rstudio text editor usually does this automatically – indenting makes your code more structured and easier to spot where things might be wrong
- Check your brackets, and make sure each thing you add to the main code is in the right 'section'

# Monopoly: Some hints

## • Debugging

- For question 2, do each part (a)–(h) one at a time, checking the graph at each stage...
- ...then, if you get something weird, you'll have more idea where you went wrong
- Each time you edit your code, do so in a different script, so you don't screw up the code that you know works!
- Indent your code – the Rstudio text editor usually does this automatically – indenting makes your code more structured and easier to spot where things might be wrong
- Check your brackets, and make sure each thing you add to the main code is in the right 'section'

# Monopoly: Some hints

## • Debugging

- For question 2, do each part (a)–(h) one at a time, checking the graph at each stage...
- ...then, if you get something weird, you'll have more idea where you went wrong
- Each time you edit your code, do so in a different script, so you don't screw up the code that you know works!
- Indent your code – the Rstudio text editor usually does this automatically – indenting makes your code more structured and easier to spot where things might be wrong
- Check your brackets, and make sure each thing you add to the main code is in the right 'section'

# Monopoly: Some hints

- **What to look for**

- Initially, all probabilities should hover around  $1/40$
- Once you start adding in extra layers (chance, community chest etc) this will change for some squares
- Lots of things send you to Jail! So expect a spike here...

- **Rolling doubles – this bit is hard!**

- Unless you're confident your code for part (h) works, leave it out!
- Including a function for this which doesn't work properly could cost you dearly!

# Monopoly: Some hints

- **What to look for**

- Initially, all probabilities should hover around  $1/40$
- Once you start adding in extra layers (chance, community chest etc) this will change for some squares
- Lots of things send you to Jail! So expect a spike here...

- **Rolling doubles – this bit is hard!**

- Unless you're confident your code for part (h) works, leave it out!
- Including a function for this which doesn't work properly could cost you dearly!

# Monopoly: Some hints

## ● What to look for

- Initially, all probabilities should hover around  $1/40$
- Once you start adding in extra layers (chance, community chest etc) this will change for some squares
- Lots of things send you to Jail! So expect a spike here...

## ● Rolling doubles – this bit is hard!

- Unless you're confident your code for part (h) works, leave it out!
- Including a function for this which doesn't work properly could cost you dearly!

# Monopoly: Some hints

## ● What to look for

- Initially, all probabilities should hover around  $1/40$
- Once you start adding in extra layers (chance, community chest etc) this will change for some squares
- Lots of things send you to Jail! So expect a spike here...

## ● Rolling doubles – this bit is hard!

- Unless you're confident your code for part (h) works, leave it out!
- Including a function for this which doesn't work properly could cost you dearly!

# Monopoly: Some hints

## ● What to look for

- Initially, all probabilities should hover around  $1/40$
- Once you start adding in extra layers (chance, community chest etc) this will change for some squares
- Lots of things send you to Jail! So expect a spike here...

## ● Rolling doubles – this bit is hard!

- Unless you're confident your code for part (h) works, leave it out!
- Including a function for this which doesn't work properly could cost you dearly!

# Monopoly: Some hints

- **What to look for**

- Initially, all probabilities should hover around  $1/40$
- Once you start adding in extra layers (chance, community chest etc) this will change for some squares
- Lots of things send you to Jail! So expect a spike here...

- **Rolling doubles – this bit is hard!**

- Unless you're confident your code for part (h) works, leave it out!
- Including a function for this which doesn't work properly could cost you dearly!

# Monopoly: Some hints

- **What to look for**

- Initially, all probabilities should hover around  $1/40$
- Once you start adding in extra layers (chance, community chest etc) this will change for some squares
- Lots of things send you to Jail! So expect a spike here...

- **Rolling doubles – this bit is hard!**

- Unless you're confident your code for part (h) works, leave it out!
- Including a function for this which doesn't work properly could cost you dearly!

# Monopoly: Some hints

- **What to look for**

- Initially, all probabilities should hover around  $1/40$
- Once you start adding in extra layers (chance, community chest etc) this will change for some squares
- Lots of things send you to Jail! So expect a spike here...

- **Rolling doubles – this bit is hard!**

- Unless you're confident your code for part (h) works, leave it out!
- Including a function for this which doesn't work properly could cost you dearly!

# Rolling doubles

```
> df = data.frame(d1=sample(1:6,3,replace=TRUE),d2=sample(1:6,3,replace=TRUE))
> df$Total = apply(df,1,sum)
> df$IsDouble = df$d1==df$d2
> df
```

|   | d1 | d2 | Total | IsDouble |
|---|----|----|-------|----------|
| 1 | 6  | 3  | 9     | FALSE    |
| 2 | 4  | 3  | 7     | FALSE    |
| 3 | 2  | 2  | 4     | TRUE     |

```
> if(df$IsDouble[1]==TRUE & df$IsDouble[2]==TRUE & df$IsDouble[3]==TRUE) {
  current = 11 #Go To Jail
} else if (df$IsDouble[1]==TRUE & df$IsDouble[2]==TRUE & df$IsDouble[3]!=TRUE) {
  current = current + sum(df$Total[1:3])
}
```

# Rolling doubles

```
> df = data.frame(d1=sample(1:6,3,replace=TRUE),d2=sample(1:6,3,replace=TRUE))
> df$Total = apply(df,1,sum)
> df$IsDouble = df$d1==df$d2
> df
```

|   | d1 | d2 | Total | IsDouble |
|---|----|----|-------|----------|
| 1 | 6  | 3  | 9     | FALSE    |
| 2 | 4  | 3  | 7     | FALSE    |
| 3 | 2  | 2  | 4     | TRUE     |

```
> if(df$IsDouble[1]==TRUE & df$IsDouble[2]==TRUE & df$IsDouble[3]==TRUE) {
  current = 11 #Go To Jail
} else if (df$IsDouble[1]==TRUE & df$IsDouble[2]==TRUE & df$IsDouble[3]!=TRUE) {
  current = current + sum(df$Total[1:3])
}
```

# Sticker album: Some hints

Ultimate goals:

- Find out how many packets of stickers are needed to fill the album!
- How much will this cost?
- Histograms and other numerical summaries...

# Sticker album: Some hints

Ultimate goals:

- Find out how many packets of stickers are needed to fill the album!
- How much will this cost?
- Histograms and other numerical summaries...

# Sticker album: Some hints

Ultimate goals:

- Find out how many packets of stickers are needed to fill the album!
- How much will this cost?
- Histograms and other numerical summaries...

# Part VI

## Monte Carlo Methods

## 6.1 Introduction

- The term “Monte Carlo” is used to describe any simulation study which involves random numbers.
- The name is a reference to the famous Monte Carlo Casino in Monaco, where repetition of random events is the order of the day!

## 6.2 What is a simulation study?

- For our purposes, a simulation study is any study where we study the properties of a system using random numbers.
- We have already seen a simulation study: for example, the monopoly exercise in your project.
- In general, suppose we do an experiment which has the event  $A$  as one possible outcome.
- We would like to estimate the probability of  $A$ , denoted by  $\Pr[A]$ .

## 6.2 What is a simulation study?

- For our purposes, a simulation study is any study where we study the properties of a system using random numbers.
- We have already seen a simulation study: for example, the monopoly exercise in your project.
- In general, suppose we do an experiment which has the event  $A$  as one possible outcome.
- We would like to estimate the probability of  $A$ , denoted by  $\Pr[A]$ .

## 6.2 What is a simulation study?

- Then by repeatedly simulating the experiment, it is simple to estimate  $\Pr[A]$ , the probability of the event  $A$ , using  $P_F(A)$ , where:

$$P_F(A) = \frac{\text{No. of times } A \text{ occurs}}{\text{Number of times experiment simulated}}.$$

- Here  $P_F(A)$  is the *frequency estimate* of  $\Pr[A]$ .
- Why does this work?

## 6.2 What is a simulation study?

- Well suppose we denote the number of times we simulate the experiment by  $n$ , then  $P_F(A)$  has the following important property:

as  $n \rightarrow \infty$ , then  $P_F(A) \rightarrow \Pr[A]$ .

- This means that the more simulations we do, the more *accurate* our estimate of  $\Pr[A]$ .

## 6.3 Monte Carlo Integration

- We would like to evaluate  $\int_0^1 f(x) dx$ , but the function may be too complicated to integrate.
- We can find an approximate answer by noting that the integral is equal to the area under the curve, and using Monte Carlo methods.
- We design an experiment which would work in general, even if the function was defined on a general range  $(a, b)$ , and if  $f(x) \in (0, c)$ , for any positive value  $c$ .

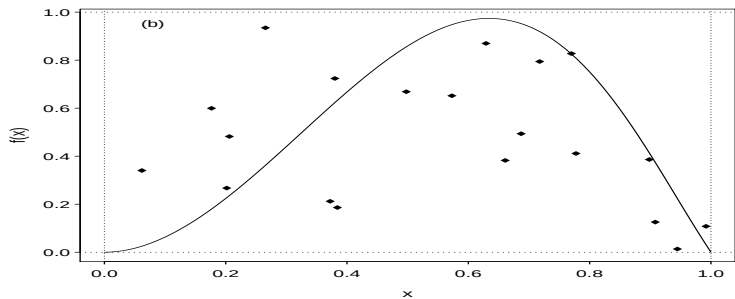
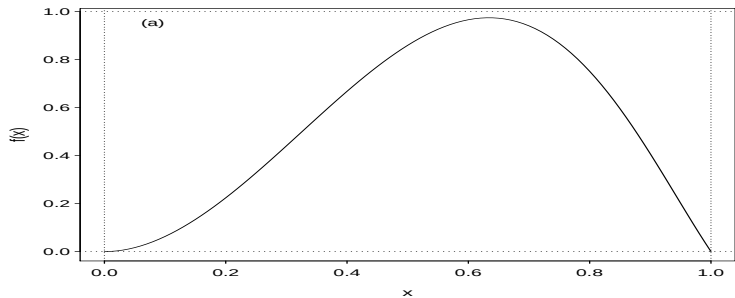


Figure: (a) An example function. (b) Twenty points randomly placed on the graph.

## 6.3 Monte Carlo Integration

We generate a random data point from a *simulation grid*. Let

$$A = \{\text{the data point lies below the curve}\}.$$

Then

$$\Pr[A] = \frac{\text{area under curve}}{\text{area of simulation grid}},$$

so that

$$\begin{aligned}\int_a^b f(x) dx &= [\text{area under curve}] \\ &= \Pr[A] \times [\text{area of simulation grid}].\end{aligned}$$

## 6.3 Monte Carlo Integration

We generate a random data point from a *simulation grid*. Let

$$A = \{\text{the data point lies below the curve}\}.$$

Then

$$\Pr[A] = \frac{\text{area under curve}}{\text{area of simulation grid}},$$

so that

$$\begin{aligned}\int_a^b f(x) dx &= [\text{area under curve}] \\ &= \Pr[A] \times [\text{area of simulation grid}].\end{aligned}$$

## 6.3 Monte Carlo Integration

We generate a random data point from a *simulation grid*. Let

$$A = \{\text{the data point lies below the curve}\}.$$

Then

$$\Pr[A] = \frac{\text{area under curve}}{\text{area of simulation grid}},$$

so that

$$\begin{aligned}\int_a^b f(x) dx &= [\text{area under curve}] \\ &= \Pr[A] \times [\text{area of simulation grid}].\end{aligned}$$

## 6.3.1 Example

- Consider the function we saw earlier in Figure 6.1, defined on  $(0, 1)$ , and with  $f(x) \in (0, 1)$ .
- Estimate  $\int_0^1 f(x) dx$  using Monte Carlo methods.

## 6.3.1 Solution: Step 1

We simulate  $n$  data points from the simulation grid; here this is the unit square  $(0, 1) \times (0, 1)$ .

## 6.3.1 Solution: Step 1

We simulate  $n$  data points from the simulation grid; here this is the unit square  $(0, 1) \times (0, 1)$ .

## 6.3.1 Solution: Step 2

Each of the coordinates  $x$  and  $y$  are generated using a  $U(0, 1)$  random variable. Note the fact that

$$A = \{\text{data point } (x, y) \text{ lies below the curve}\} = \{y < f(x)\}.$$

## 6.3.1 Solution: Step 2

Each of the coordinates  $x$  and  $y$  are generated using a  $U(0, 1)$  random variable. Note the fact that

$$A = \{\text{data point } (x, y) \text{ lies below the curve}\} = \{y < f(x)\}.$$

## 6.3.1 Solution: Step 2

Each of the coordinates  $x$  and  $y$  are generated using a  $U(0, 1)$  random variable. Note the fact that

$$A = \{\text{data point } (x, y) \text{ lies below the curve}\} = \{y < f(x)\}.$$

## 6.3.1 Solution: Step 2

Each of the coordinates  $x$  and  $y$  are generated using a  $U(0, 1)$  random variable. Note the fact that

$$A = \{\text{data point } (x, y) \text{ lies below the curve}\} = \{y < f(x)\}.$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 3

If  $r$  points lie below the curve, then  $\Pr[A] \simeq P_F(a) = r/n$ , and thus

$$\begin{aligned}\int_0^1 f(x) dx &= \Pr[A] \times [\text{area of simulation grid}] \\ &= \Pr[A] \times 1 \\ &= \Pr[A] \\ &\simeq P_F(A) \\ &= r/n.\end{aligned}$$

## 6.3.1 Solution: Step 4

In our case, from the figure above we estimate:

$$\int_0^1 f(x) dx \simeq \frac{r}{n} = \frac{12}{20} = 0.6 .$$

## 6.3.1 Solution: Step 4

In our case, from the figure above we estimate:

$$\int_0^1 f(x) dx \simeq \frac{r}{n} = \frac{12}{20} = 0.6 .$$

## 6.3.1 Solution: Step 4

In our case, from the figure above we estimate:

$$\int_0^1 f(x) dx \simeq \frac{r}{n} = \frac{12}{20} = 0.6 .$$

## 6.3.1 Solution: Step 4

In our case, from the figure above we estimate:

$$\int_0^1 f(x) dx \simeq \frac{r}{n} = \frac{12}{20} = 0.6 .$$

## 6.3.2 Example using R

We can also easily implement the above algorithm in R. As an extreme example, consider the function

$$f(x) = |\sin\{\sin[\sin(x^4)]\}|.$$

We wish to calculate

$$\int_0^2 f(x) dx = \int_0^2 |\sin\{\sin[\sin(x^4)]\}| dx,$$

(which evaluates to approximately 0.71875). First plot the function to determine the necessary region:

## 6.3.2 Example using R

We can also easily implement the above algorithm in R. As an extreme example, consider the function

$$f(x) = |\sin\{\sin[\sin(x^4)]\}|.$$

We wish to calculate

$$\int_0^2 f(x) dx = \int_0^2 |\sin\{\sin[\sin(x^4)]\}| dx,$$

(which evaluates to approximately 0.71875). First plot the function to determine the necessary region:

## 6.3.2 Example using R

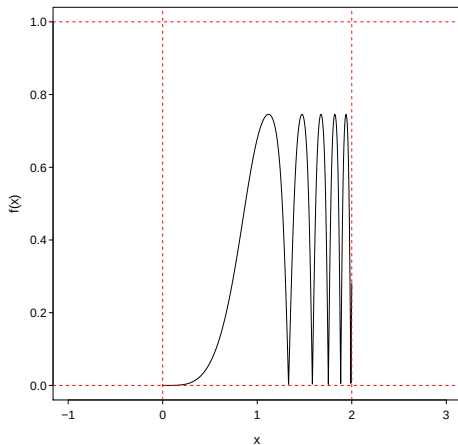


Figure: A plot of  $|\sin\{\sin[\sin(x^4)]\}|$  with the sampling region.

## 6.3.2 Example using R

The plot generated in the code above is shown in figure 6.2. Then we use a for loop to simulate lots of random numbers:

```
> set.seed(1)
> N = 100000
> no_of_hits = 0
> for(i in 1:N) {
+   x = runif(1, 0, 2); y = runif(1)
+   if(abs(sin(sin(sin(x^4)))) > y) {
+     no_of_hits = no_of_hits + 1
+   }
+ }
> area_under_curve = no_of_hits/N*2
> area_under_curve
```