Lab100 Week 16: Simulation in R

By the end of the session you should be able to:
- generate random numbers,
- construct a histogram,
- use seeds correctly.
- simulate coin tosses and dice rolls

New R commands
\( \text{runif hist set.seed} \)

Random numbers

There is a philosophical problem here. By definition random numbers are entirely unpre-
dictable. Any computer generates numbers using a deterministic algorithm (or rule). How
then can a computer generate random numbers?

The problem is intriguing, sounds difficult, and is impossible!

However we can get the computer to generate numbers that can pass any test of randomness
that an external observer can devise. These are the ones we use.

Q 16.1 WS: Uniform random numbers. We can generate random numbers using the \text{runif}
command. This defaults to creating numbers between 0 and 1. Try
\( \text{runif(1)} \)

Repeat \( \text{runif(1)} \) to get a new number. Repeat \( \text{runif(3)} \) to get a vector.

We can change the defaults \( \text{runif(n,a,b)} \). \( n \) is how many numbers you want to generate,
\( a \) is the lower bound, \( b \) is the upper bound.

\[
\begin{align*}
x &= \text{runif(50,0,10)} \quad \# 50 \text{ random variables between 0 and 10} \\
y &= \text{runif(50,10,20)} \quad \# 50 \text{ random variables between 10 and 20}
\end{align*}
\]

Uniform random numbers are equally likely to be any number between the bounds. This
means that when we draw a histogram of our simulated random numbers, it should be almost
flat.

Q 16.2 WS: Histograms. A histogram describes the distribution of a set of numbers. It
displays the empirical (observed) frequencies that the numbers fall in specific intervals of the
real line. These numbers are deterministic:

\[
\begin{align*}
x &= \text{seq(1,400)/400} \\
\text{hist(x,20, col='yellow')} & \quad \# \text{gives 20 intervals} \\
y &= x^2 \; ; \; \text{hist(y,20)}
\end{align*}
\]
These numbers are random:

```r
x = runif(400)
hist(x, 20, col='yellow')
y = x^2 ; hist(y, 20)
```

Describe the difference between the two sets of numbers.

To get a $3 \times 2$ matrix:

```r
matrix( runif(6), 3, 2 )
```

Repeat to get a different matrix.

The mean of default uniform random numbers should be near $0.5$.

```r
mean( runif(5) ) ; mean( runif(500) )
```

What effect does increasing the number of observations have? Try 10, 100, 1000, 10000.

**Q 16.3 WS: set.seed** Sometimes it is helpful to be able to generate the same set of random numbers again. For example, when debugging (error fixing) code you might want to use the same set of random numbers that created an error again. When you start R it automatically chooses a set of random numbers, then from that set, the second set is pre-determined. If we pre-select one of the billions of starting points, then it makes the numbers more random. We can do this using `set.seed(seedno)`.

```r
set.seed(20)
x=runif(200); mean(x)
set.seed(40)
x=runif(200); mean(x)
set.seed(20)
x=runif(200); mean(x)
```

Notice how the first and third mean are the same but the last is different.

**Q 16.4 WS: Unbiased Coin Toss** We can use the uniform random variables to simulate tossing a coin. If the value of the default uniform random variable is less than 0.5 we will say it is a Head (H) and if it is greater than 0.5 then we call it a Tail (T).

To do this we need to get R to check whether our simulated value is greater than or less than 0.5, we do this using an IF statement. An IF statement will only complete the action IF a condition is met. A general form of an IF statement is

```
IF(condition){then do this}
ELSE IF(condition){then do this}
ELSE{do this}
```
You don’t need to include all these options in every statement. In our example the ‘condition’ is that our simulated value is less than 0.5 and our ‘then do this’ is setting the value of our coin toss to Heads. Now we can simulate the coin toss using the following R code.

```r
x=runif(1)
y='T'
if(x<0.5){y='H'}
y
```

The above code simulates a uniform random variable ‘x’. We then set the value of the coin toss to Tails. The final line changes the value of the coin toss to Heads if our uniform random variable is less than 0.5. Note that if the random variable is equal to 0.5 then the coin toss stays as Tails. NOTE: We have to put the quotation marks (’ and ’) around the T and H so that R understands that y is a character rather than a numeric value.

Q 16.5 WS: Biased Coin Toss To simulate tossing a biased coin we just need to change the value in our IF statement. To simulate a coin that has a probability of 0.6 for Tail and 0.4 for Heads we use the following code.

```r
x=runif(1)
y='T'
if(x<0.4){y='H'}
y
```

Try changing the probabilities to bias the coin both ways (don’t forget the two probabilities always have to sum to 1).

Q 16.6 WS: Sequence of coin tosses To simulate more than one coin toss we just need to simulate a different uniform random variable and repeat our test. Just repeating the above code over and over will overwrite our simulated values. Really we want to keep all the values we simulate and so we create an empty vector to store them in.

```r
y=NULL
```

Then we decide how many coin tosses we want to simulate, we shall start with 10. As we don’t want to overwrite our values, we need some way to tell R to move onto the next dimension in our vector. To do this we use a FOR loop. A FOR loop repeats the same code over and over replacing some part of the code with a different value. The general form is

```r
FOR(variable IN range){run this code}
```

So to repeat our coin toss ‘n’ times we use the following code.

```r
FOR(1:10){x=runif(1);y='T';if(x<0.4){y='H'};y}
```
n=10
y=rep('T',n) # fills our vector with Tails
for(i in 1:n){
  x=runif(1)
  if(x<0.5){
    y[i]='H'
  } # ends IF statement
} # ends FOR loop
y

This simply repeats our above simulation ‘n’ times. The key is the \( y[i] \) which tells R to replace the \( i^{th} \) value of the vector \( y \) only. As \( i \) changes through the values from 1 to \( n \) the values in \( y \) are changed or not changed depending on whether they fulfil the condition of the IF statement. We could also do the above using an IF and ELSE statement instead. Try the following code:

n=10
y=NULL # sets up null vector
for(i in 1:n){
  x=runif(1)
  if(x<0.5){
    y[i]='H'
  } # ends IF statement
  else{
    y[i]='T'
  } # ends ELSE statement
} # ends FOR loop
y

Make sure you understand the difference between the two ways to code this. Note that the second option is preferred as if there is an error in the first code you will just get a vector of Tails. However, if there is an error in the second code \( y \) does not have a default value so it will contain NA terms. (You can try replacing \( y[i]='T' \) with \( x='T' \) in the IF statement in both sets of code above to see for yourself.) Repeat this for a biased coin by changing the value in the IF statement.

**Q 16.7 WS: Simulating Die Rolls** We can make the coin toss more complicated by adding more than two outcomes. We shall now simulate rolling a die, where the outcomes are the numbers 1-6 and each outcome is equally likely. We can just extend the above code to 6 outcomes.

x=runif(1)
{ if(x<1/6){y=1}
  else if(x<2/6){y=2}
  else if(x<3/6){y=3}
  else if(x<4/6){y=4}
  else if(x<5/6){y=5}
  else if(x<6/6){y=6}
}
else if(x<4/6){y=4}
else if(x<5/6){y=5}
else{y=6} }
y

The extra brackets around the entire statement tell R this is one expression. Note that we use the ELSE IF statement as it is only activated if all previous IF/ELSE IF statements have failed. Otherwise if we just put all IF statements in, a value of 0.4 would activate the \(x<3/6\) and above conditions and so would always be given the value 5.

We have now simulated one die roll. If we want to simulate several die rolls then we can put our code inside a FOR loop as before.

```r
y=NULL
n=10
for(i in 1:n){
  x=runif(1)
  if(x<1/6){y[i]=1}
  else if(x<2/6){y[i]=2}
  else if(x<3/6){y[i]=3}
  else if(x<4/6){y[i]=4}
  else if(x<5/6){y[i]=5}
  else{y[i]=6}
}
y
```

**Q 16.8 WS: Histogram** Using the above code, simulate \(n = 10\) rolls of a die. We then draw a histogram of the results using

```r
hist(y,prob=T)
```

Repeat the experiment for \(n = 50, 100, 500\). What is happening to the shape?

**Lab100 Quiz Week 16**

At the end remember to quit from R with `q()`.

**Q 16.9 QZ: qz100wk16.i: True or False?** Consider whether the following statements are true or false.

1. `runif(3)` has dimension 3,
2. `log(runif(5))` has all positive elements,
3. in repeated samples `mean(runif(1000))` is within 0.03 of 1/2,
4. in repeated samples `sd(runif(1000))` is within 0.01 of \(\sqrt{1/12}\)
(A) FFTF, (B) FTTF, (C) TFTT, (D) FFFT, (E) TTFT

**Q 16.10**  
*QZ: qz100wk16.ii: Histograms*  
Run the code

```r
x = runif(1000)
y = -log(x)
hist(y, 20)
```

Which curve does the histogram most resemble

(A) rising linear, (B) uniform, (C) falling exponential, (D) bell shaped, (E) falling linear.

**Q 16.11**  
*QZ: qz100wk16.iii: Biased Coin*  
How many heads do you get when you toss a biased coin, for which \( P(\text{head}) = 0.2 \), 80 times using seed 512?

(A) 18 (B) 16 (C) 20 (D) 17 (E) 15

**Q 16.12**  
*QZ: qz100wk16.iv: Dice*  
We play a game where I roll a dice 50 times. If it comes up as 1 or 6 I give you £1, otherwise you give me £1. If you start with £50, how much money do you have left (use set.seed 108)?

(A) 100 (B) 20 (C) 50 (D) -10 (E) 10