Getting Started in R

Part 3

by M. Papathomas and E. Rexstad (with material from a document prepared by R. King)

```
## Warning: package 'knitr' was built under R version 3.2.5
```

In the previous Microlab session, we wrote functions in `R`. In this practical, we will look at understanding a given piece of `R` code used to solve a specific problem problem. This will be extended to modify the code to undertake changes in the problem being considered.

Example: Playing a game

You are offered the chance to take part in a game with the following rules:

- You pay £5 to play.
- You throw five dice.
- Any that do not show a six, you throw again.
- Any that still do not show a six, you throw a third time.
- Your winnings are £2^n where n is the final number of sixes. (Your profit is therefore £$(2^n - 5)$.)

Under these rules, determine the following consequences of playing this dice game.

- You play the game once. What is the probability that you throw no sixes at the first throw of five dice? Hence or otherwise, find an expression for the probability that you throw no sixes at all. Use `R` to evaluate your answer to four decimal places.

- Explain carefully what the following `R` code does, relating the `R` commands to the rules of the game.

```
psixes <- rep(0,6)
for (k in 1:1000){
  sixes <- rep(0,5)
  for (i in 1:3){
    for (j in 1:5){
      if(sixes[j]==0){
          x <- rbinom(1,1,1/6)
          if(x==1) sixes[j] <- 1
      }
    }
  }
  y <- sum(sixes)
  psixes[y+1] <- psixes[y+1]+1
}
psixes
```

- Type the preceeding code into the R-Studio editor panel, submit it to the R console and use it to check your manual answer from the previous part. HINT: Until you are confident of what your code is doing, it may be better to substitute `for (k in 1:100){` for the line `for (k in 1:1000){` (resulting in precision to only 3 decimal places). If you are unsure of what an `R` command does, use the `help()` function.

- Given your output obtained from running the code, provide an estimate of the probability mass function for

  - number of sixes thrown and

  - profit.

- Calculate an estimate of the probability that your winnings exceed the original £5 if the game is played a single time.
- Extend your `R` code to plot histograms of the estimated probability mass function of
  - number of sixes thrown and
  - profit.
  - HINT: the variable `psixes` is a set of six frequencies that sum to 1000. To plot a histogram of the number of sixes over 1000 games, you need to set up a variable with 1000 observations; each of these observations will be one of the numbers 0,1,2,3,4,5. The easy way to do this is to store the 1000 values in a vector. For your histograms, you may find the options `freq` and `breaks` useful — see `help(hist)`.
- Further extend your `R` code to obtain an empirical estimate of the expected number of sixes thrown and the expected profit, by using the estimated probability mass function and the definition of expectation.
- Modify the code to play the game with only 4 dice. How does this affect the expected number of sixes thrown, expected profit and probability of your winnings exceeding the original £5 stake?
- Reverting back to the five dice, explore the game where you are only throw the dice for a total of two times. How does this affect the expected number of sixes thrown, expected profit and probability of your winnings exceeding the original £5 stake?