MAS115: R programming
Lecture 2: Pseudocode and Monte Carlo

Lab Class and Technical Material:
Loops and control statements

February 16, 2021

Aims

Problems Class

- `for` loops
- `if` statements
- `break` and `next` commands

Lecture

- Pseudocode
- Monte Carlo estimation

Pseudocode

- Looking at raw code sometimes is hard to understand.
- Pseudocode is an informal way to explain how a computer program is designed.
- Describes in less formal terms the steps needed to perform a task.
- Aims to get the basic ideas across quickly and simply to the reader without all technical details.
- Can consider it like a recipe
- No fixed rules but generally a few defined conventions.

Pseudocode: Fizz-buzz

Fizz-buzz for the first 100 numbers:

```
FOR (i in 1 to 100)
    SET print_number = TRUE
    IF (i is divisible by 3)
      THEN PRINT "Fizz"
           SET print_number = FALSE
    ENDIF
    IF (i is divisible by 5)
      THEN PRINT "Buzz"
           SET print_number = FALSE
    ENDIF
    IF (print_number = TRUE)
      THEN PRINT i
           PRINT a newline
    ENDIF
ENDFOR
```

Pseudocode: Fizz-buzz

Fizz-buzz for the first 100 numbers:

```
FOR (i in 1 to 100)
    SET print_number = TRUE
    IF (i is divisible by 3)
      THEN PRINT "Fizz"
           SET print_number = FALSE
    ENDIF
    IF (i is divisible by 5)
      THEN PRINT "Buzz"
           SET print_number = FALSE
    ENDIF
    IF (print_number = TRUE)
      THEN PRINT i
           PRINT a newline
    ENDIF
ENDFOR
```

Pseudocode: Fizz-buzz

Fizz-buzz for the first 100 numbers:

```
FOR (i in 1 to 100)
    SET print_number = TRUE
    IF (i is divisible by 3)
      THEN PRINT "Fizz"
           SET print_number = FALSE
    ENDIF
    IF (i is divisible by 5)
      THEN PRINT "Buzz"
           SET print_number = FALSE
    ENDIF
    IF (print_number = TRUE)
      THEN PRINT i
           PRINT a newline
    ENDIF
ENDFOR
```

Pseudocode: Fizz-buzz

Fizz-buzz for the first 100 numbers:

```
FOR (i in 1 to 100)
    SET print_number = TRUE
    IF (i is divisible by 3)
       THEN PRINT "Fizz"
            SET print_number = FALSE
    ENDIF
    IF (i is divisible by 5)
      THEN PRINT "Buzz"
           SET print_number = FALSE
    ENDIF
    IF (print_number = TRUE)
      THEN PRINT i
           PRINT a newline
    ENDIF
ENDFOR
```

Pseudocode: Fizz-buzz

Fizz-buzz for the first 100 numbers:

```
FOR (i in 1 to 100)
    SET print_number = TRUE
    IF (i is divisible by 3)
      THEN PRINT "Fizz"
           SET print\_number = FALSE
    ENDIF
    IF (i is divisible by 5)
      THEN PRINT "Buzz"
           SET print_number = FALSE
    ENDIF
    IF (print_number = TRUE)
      THEN PRINT i
           PRINT a newline
    ENDIF
ENDFOR
```

Pseudocode: Fizz-buzz

Fizz-buzz for the first 100 numbers:

```
FOR (i in 1 to 100)
    SET print_number = TRUE
    IF (i is divisible by 3)
      THEN PRINT "Fizz"
           SET print\_number = FALSE
    ENDIF
    IF (i is divisible by 5
      THEN PRINT "Buzz"
           SET print\_number = FALSE
    ENDIF
    IF (print_number = TRUE))
      THEN PRINT i
           PRINT a newline
    ENDIF
ENDFOR
```

Pseudocode: Fizz-buzz

Fizz-buzz for the first N numbers:

```
INPUT N
FOR (i in 1 to N)
    SET print_number = TRUE
    IF (i is divisible by 3)
      THEN PRINT "Fizz"
           SET print_number = FALSE
    ENDIF
    IF (i is divisible by 5)
      THEN PRINT "Buzz"
           SET print_number = FALSE
    ENDIF
    IF (print_number = TRUE)
      THEN PRINT i
           PRINT newline
    ENDIF
ENDFOR
```

Pseudocode: format

```
IF (condition)
    THEN statements
    ELSE statements
ENDIF

WHILE (condition)
    DO statements;
ENDWHILE;

FOR (iteration bounds)
    DO statements
ENDFOR;
```

Make sure that you indent the code for ease of reading

Pseudocode: notation

There is no set of standard pseudocode notation, but some of the most widely used are:

- SET/INITIALIZE/ASSIGN — create a variable and set it to be a certain value
- SET/ASSIGN — change the value of an existing variable
- DO — carry out some task
- INPUT — indicates that here the user will be inputting something
- PRINT/OUTPUT — indicates that an output will appear on the screen
- RETURN — often used in functions to denote the values that will be returned to the main program

You can also use standard mathematical operators such as $+, -, <, >$, AND, OR, ... Could use $\leftarrow$ for assignment.

Pseudocode Class Example: Fibonacci Numbers

Pseudocode to print the first fifteen numbers in the Fibonacci sequence,

$$f_{i+1} = f_i + f_{i-1}$$

Pseudocode Class Example: Fibonacci Numbers

Pseudocode to print the first fifteen numbers in the Fibonacci sequence,

$$f_{i+1} = f_i + f_{i-1}$$

```
SET sum = 0
SET fA and fB = 1 # Two current Fibonacci numbers
SET counter = 2 # Fibonacci numbers found so far
WHILE (counter < 15)
    sum = fA + fB
    PRINT sum
    ASSIGN fA's value to fB
    ASSIGN sum's value to fA
    counter = counter + 1
ENDWHILE
```

Pseudocode Class Example: Fibonacci Numbers

```
SET sum ← 0
SET N ← 15
SET fA ← 1, fB ← 1
SET counter ← 2
WHILE (counter < N)
      sum ← fA + fB
      PRINT sum
      fB ← fA
      fA ← sum
      counter ← counter + 1
ENDWHILE
```

Pseudocode Class Example: Fibonacci Sequence

```
SET N ← 15
SET F ← empty vector of length N
SET F[1] ← 1
SET F[2] ← 1
FOR (i in 3:N)
    F[i] ← F[i−1] + F[i−2]
    PRINT F[i]
ENDFOR
```

Pseudo-code Task: Finding prime numbers

Problem:
Given a natural number $n > 3$ find out if it is prime.

Method:
Initialise a flag variable `prime <- TRUE`. Then try dividing `n` by possible factors, keep flag variable as `TRUE` unless you find a number that divides n in which case you should change flag to `FALSE`. Otherwise, having tried all possible numbers we know number is prime.

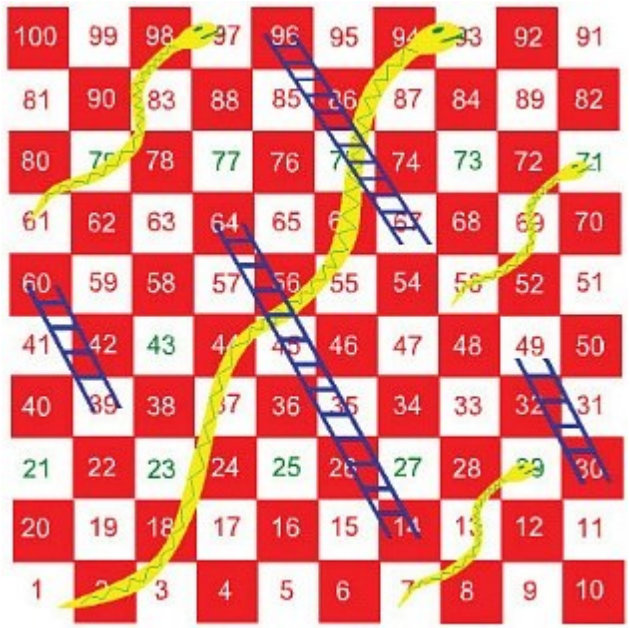
Hint: I used a `FOR` loop with a `BREAK` statement

Pseudo-code Task - Prime numbers - Solution

```
INITIALIZE variable prime ← TRUE
FOR (each natural number i = 2,...,⌊√n⌋)
    IF n mod i = 0
        THEN set prime ← FALSE
             BREAK for loop
    ENDIF
ENDFOR
RETURN prime
```

Monte-Carlo Estimation

Question:
Suppose you are playing snakes and ladders. To finish playing you need to land exactly on the winning tile. If you roll more than the number needed to finish then you will bounce off the end tile and back however many moves you have left.



How long do you expect it will take you to finish?

Monte-Carlo Estimation

Answer:

Working this out is hard but computer simulation can help us:

- The number of rolls we need is a random variable X
- We want to know $\mu = E[X]$

Monte-Carlo Estimation

Answer:

Working this out is hard but computer simulation can help us:

- The number of rolls we need is a random variable X
- We want to know $\mu = E[X]$
- If we could play the game lots of times then we would observe $X_1, \ldots, X_n$
- Could estimate

$$\hat{\mu}_n = \bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i$$

Monte-Carlo Estimation

Answer:

Working this out is hard but computer simulation can help us:

- The number of rolls we need is a random variable X
- We want to know $\mu = E[X]$
- If we could play the game lots of times then we would observe $X_1, \ldots, X_n$
- Could estimate

$$\hat{\mu}_n = \bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i$$

- The Central Limit Theorem tells us that (under some conditions)

$$\bar{X}_n \to N\left(\mu, \frac{\sigma^2}{n}\right).$$

We don't want to play the game loads of times but we can get our computer to simulate games for us

Monte-Carlo Estimation - special case

Let X be a 0-1 random variable - the *indicator* of some event.
Let $X_1, \ldots, X_n$ be independent r.v.s, same distribution as X, so

$$X_i = \begin{cases} 0 & \text{with probability } 1 - p \\ 1 & \text{with probability p} \end{cases}$$

where $p \equiv E[X]$.

Monte-Carlo Estimation - special case

Let X be a 0-1 random variable - the *indicator* of some event.
Let $X_1, \ldots, X_n$ be independent r.v.s, same distribution as X, so

$$X_i = \begin{cases} 0 & \text{with probability } 1-p \\ 1 & \text{with probability p} \end{cases}$$

where $p \equiv E[X]$. Then $\sum_{i=1}^{n} X_i$ has a binomial distribution, and

$$\hat{p}_n = \bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i.$$

Monte-Carlo Estimation - special case

Let X be a 0-1 random variable - the *indicator* of some event.
Let $X_1, \ldots, X_n$ be independent r.v.s, same distribution as X, so

$$X_i = \begin{cases} 0 & \text{with probability } 1-p \\ 1 & \text{with probability p} \end{cases}$$

where $p \equiv E[X]$. Then $\sum_{i=1}^{n} X_i$ has a binomial distribution, and

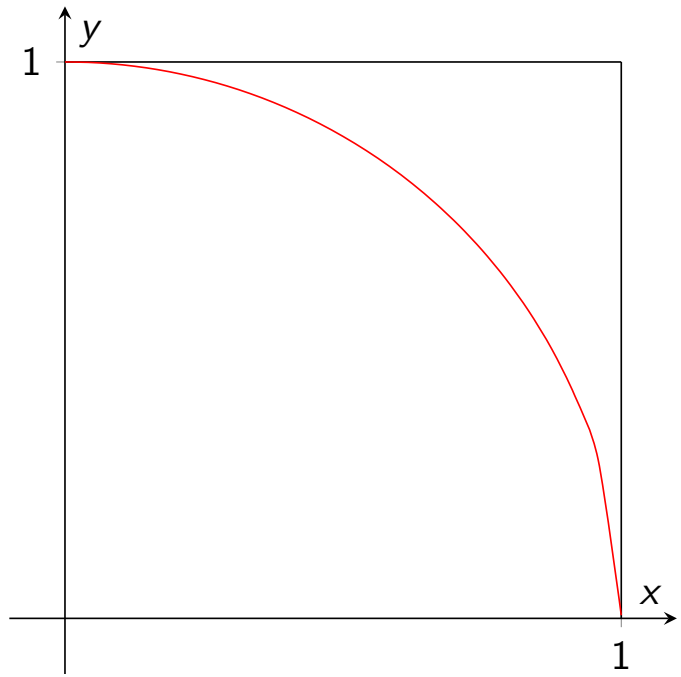
$$\hat{p}_n = \bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i.$$

The Central Limit Theorem tells us that for $0 < p < 1$,

$$\bar{X}_n \to N\left(p, \frac{p(1-p)}{n}\right).$$

Estimating π again

Consider the unit quarter-disc in the first quadrant, sitting inside the unit square



- If you threw a point uniformly at random onto square, what is probability it would lie in the disc?
- Write pseudo-code which generates N points (x, y) uniformly at random in the unit square and hence estimate π.

Solution - ready for vectorization in R

```
INPUT N
SAMPLE N VALUES x[1]...x[N] from UNIF[0,1]
SAMPLE N VALUES y[1]...y[N] from UNIF[0,1]
FOR (i in 1:N)
    SET r[i] ← x[i]^2+y[i]^2
ENDFOR
FOR (i in 1:N)
    IF (r[i] < 1)
        THEN bool[i] ← TRUE
    ELSE
        bool[i] ← FALSE
    ENDIF
ENDFOR
SET n ← number of cases with bool[i] TRUE
RETURN (4*n/N);
```