MAS115: R programming
Lecture 2: Pseudocode
Lab Class and Technical Material: Matrices, Factors, Data frames, Lists

Paul Blackwell

The University of Sheffield
School of Mathematics and Statistics

Problems Class Aims

Introduce other data structures:

- Matrices
- Factors
- Data Frames
- Lists

Will show how to create them and how to use operators on them.

In this lecture we concentrate on a key idea to help us program:

Pseudocode

and we'll try some practical examples.

Pseudocode

- The fundamental idea of coding is being able to split a big problem into several small problems.
- 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: How computers work

- When we write a program, your computer will work sequentially
- It will start at the beginning of your code
- Implement each command one after the other in sequence as it comes along it
- Pseudocode explains (in words) each step of an algorithm in terms of actions required and the order they must be done

Pseudocode: Making a cup of tea

We can write pseudocode for the process of making a cup of tea:

```
Collect together teapot, teabags and mugs;
Plug in kettle;
Place teabags in teapot;
Put water in kettle;
Boil kettle:
Add boiling water to teapot;
Wait until tea brewed;
Pour tea into mugs;
Add milk/sugar;
Serve;
```

Pseudocode: Adding Conditionality

Often we will want our program to make a choice and do different things in different circumstances. We learnt how to do this last term using an `IF` statement.
Suppose we want a choice as to whether to add milk:

```
IF (milk wanted)
    THEN add milk;
    ELSE don't add milk;
ENDIF;
```

or more generally

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

Make sure that you indent the code for ease of reading

Pseudocode: Making a cup of tea II

```
Collect together teapot, teabags and mugs;
Plug in kettle;
Place teabags in teapot;
Put water in kettle;
Boil kettle:
Add boiling water to teapot;
Wait until tea brewed;
Pour tea into mugs;
IF (milk wanted)
    THEN add milk;
    ELSE don't add milk;
ENDIF;
Serve;
```

Pseudocode: Including Loops `WHILE`

We might also want our computer to keep doing some part of our program until some condition occurs. We have learnt about ways to do this; two examples are `while` and `for` loops.
For example we need to add water to the kettle until it is full

```
WHILE (kettle is not full)
    DO add water to kettle;
ENDWHILE;
```

or more generally

```
WHILE (condition)
    DO statements;
ENDWHILE;
```

Make sure that you indent the code for ease of reading

Pseudocode: Making a cup of tea III

```
Collect together teapot, teabags and mugs;
Plug in kettle;
Place teabags in teapot;
WHILE (kettle is not full)
    DO add water to kettle;
ENDWHILE;
Boil kettle:
Add boiling water to teapot;
Wait until tea brewed;
Pour tea into mugs;
IF (milk wanted)
    THEN add milk;
    ELSE don't add milk;
ENDIF;
Serve;
```

Pseudocode: Including Loops FOR

Another way of writing loops is to use a FOR statement. For example suppose we wanted to fill each mug separately:

```
FOR (each mug in set of mugs)
    DO pour tea into mug;
ENDFOR;
```

or more generally

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

Make sure that you indent the code for ease of reading

Pseudocode: Making a cup of tea IV

```
Collect together teapot, teabags and mugs;
Plug in kettle;
Place teabags in teapot;
WHILE (kettle is not full)
    DO add water to kettle;
ENDWHILE;
Boil kettle:
Add boiling water to teapot;
Wait until tea brewed;
FOR (each mug in set of mugs)
    DO pour tea into mugs;
    IF (milk wanted)
        THEN add milk;
        ELSE don't add milk;
    ENDIF;
ENDFOR;
Serve;
```

Pseudocode: Class Exercise

Write pseudocode for a simple recipe: making a cheese sandwich, boiling some eggs (make sure they are fresh!), baking some potatoes, the omelette recipe from Semester 1, ...

Make sure to include:

- At least one conditional statement
- At least one loop

Think about what assumptions you are making about the 'user'. How can you generalize to allow for cooking for different numbers of people, or similar?

Pseudocode: Class Exercise

```
IF (bread is sliced)
   THEN get out two slices of bread;
   ELSE cut two slices of bread;
ENDIF;
FOR slice in {first, second}
   butter this slice;
ENDFOR;
slice cheese;
put cheese on first slice;
IF (lettuce required)
   THEN put lettuce on top of cheese;
ENDIF;
put second slice on top
```

Pseudocode: Class Exercise

```
SET n = number of sandwiches required
IF (bread is sliced)
   THEN get out 2n slices of bread;
   ELSE cut 2n slices of bread;
ENDIF;
FOR j in 1,...,2n
   butter the jth slice;
ENDFOR;
slice cheese;
FOR k in 1,...,n
   assemble kth sandwich;
ENDFOR;
```

Pseudocode: Starting to be more formal

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 $+, -, <, >, AND, OR, \ldots$. Could use $\leftarrow$ for assignment.

Pseudocode: Starting to be more formal

We have seen the different loops/conditional statements already

- WHILE — a loop (iteration that has a condition at the beginning)
- FOR — a counting loop (iteration)
- IF - THEN - ELSE a decision (selection) in which a choice is made
- REPEAT - UNTIL — a loop (iteration) that has a condition at the end

Make sure that for loops/conditioning you indent the statements within the loop/conditioning

Pseudocode: Starting to be more formal

So far we've considered real-life examples but for mathematical problems the idea is exactly the same. Suppose we wanted to write write pseudocode for the following examples.

Pseudocode: Reading and writing

Pseudocode for an algorithm which reads in a number and then prints it to the screen:

```
INPUT n;
PRINT n;
```

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 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)
```

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;
```

Downloading R and RStudio at home

- If you want to download R then you can do so from: `https://cran.r-project.org/`
- If you want to download RStudio then go to: `https://www.rstudio.com/`

There are versions for Windows, Mac and Linux