MAS115: R programming
Lecture 2: Pseudo-Code
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:

PSEUDO-CODE

and we'll try some practical examples.

Pseudo-Code

- 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.
- Pseudo-code 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.

Pseudo-Code: 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
- Pseudo-code explains (in words) each step of an algorithm in terms of actions required and the order they must be done

Pseudo-Code: Making a cup of tea

We can write pseudo-code 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;
```

Pseudo-Code: 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

Pseudo-Code: 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;
```

Pseudo-Code: 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

Pseudo-Code: 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;
```

Pseudo-Code: 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

Pseudo-Code: 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;
```

Pseudo-Code: Class Exercise

Write pseudo-code for a simple recipe: 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?

Pseudo-Code: Class Exercise

```
INGREDIENTS
3 eggs, as fresh as possible
2 knobs unsalted butter
Salt and pepper
INSTRUCTIONS
Crack the eggs into a small bowl and whisk.
Add some salt and pepper.
Heat the butter in a 9-inch non-stick frying pan.
Pour in the eggs.
In the first 30 seconds, create 6-10 small cuts.
When the top is nearly set turn off the heat.
Don't worry if the centre isn't quite set.
Use your spatula to flip one half over the other.
Serve immediately.
```

Pseudo-Code: 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.

Pseudo-Code: 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

Pseudo-Code: 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 pseudo-code for the following examples.

Pseudo-Code: Reading and writing

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

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

Pseudo-Code: Checking a number is odd or even

Pseudo-code for an algorithm which reads in a number, checks if it is even or odd and then tells you:

```
INPUT n;
IF (n/2 gives a remainder)
    THEN PRINT "It is odd";
    ELSE PRINT "It is even";
ENDIF;
```

Pseudo-Code: 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
```

Class Examples: The minimum of two numbers

Write pseudo-code using an `IF` statement (and the $<$ operator) for an algorithm which reads two numbers `A` and `B` and tells you which is smaller

```
INPUT A, B;
```

Class Examples: The minimum of two numbers

Write pseudo-code using an `IF` statement (and the $<$ operator) for an algorithm which reads two numbers `A` and `B` and tells you which is smaller

```
INPUT A, B;

IF (A < B)
  THEN PRINT A "is smaller";
  ELSE PRINT B "is smaller";
ENDIF;
```

Can you modify it to tell you if they are equal?

Pseudo-Code: Fibonacci Numbers

Pseudo-code which prints the first fifteen numbers in the Fibonacci sequence:

```
SET sum = 0;
SET fA and fB = 1; # Two current Fibonacci numbers
SET counter = 2; # How many Fibonacci numbers we've found

WHILE (counter < 15)
```

Pseudo-Code: Fibonacci Numbers

Pseudo-code which prints the first fifteen numbers in the Fibonacci sequence:

```
SET sum = 0;
SET fA and fB = 1; # Two current Fibonacci numbers
SET counter = 2; # How many Fibonacci numbers we've found

WHILE (counter < 15)

sum = fA + fB;
  PRINT sum;
  ASSIGN fA's value to fB;
  ASSIGN sum's value to fA;
  counter = counter + 1;
```

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