Lab 0: Introduction to **R** Programming

Math 2820L : Spring 2018 : Jon Ashbrock

1 Introduction

In class today we will write a program which will let the user input an integer and the program will output the factorial of that number. After today's lab, the students will be able to

1. How to follow the logic of a computer program
2. How to do mathematics in **R**
3. How to write comments in a program
4. How to write a for-loop
5. Use the print statement

2 The logic of a computer program

Every programming language has a set of words and symbols it recognizes to mean specific things. Examples in **R** include

1. Symbol: + Operation: Add two numbers together
2. Symbol: `<-` Operation: Assign a variable a specific value
3. Symbol: == Operation: Return TRUE if two values are equal and FALSE otherwise

When you write a computer program, you tell the computer what values to store in what variables and how to manipulate those values to get the result that you want. The computer reads code from top to bottom and runs each line of code it sees in order. After knowing this, it is best to see examples rather than explaining what coding is.

Discussion. *We want to write the factorial program I described above. Let us outline exactly the steps we need to take to complete this task.*

3 Mathematics in R

R understands the following mathematical operations: $+, -, *, /, \%\%, \wedge$. The first four do exactly what you would expect them to. The symbol %% returns the remainder when dividing the first number by the second. The carrot ($\wedge$) symbol is used for exponents. We also need to know the symbol `<-` takes the value on the right side of the symbol and stores it in the variable name on the left side of the symbol. Lets see an example: Typing these

```
Sum <- 4+5
Rem <- 20% % 3
a <- 2 ∧ 4
```

into the **R** console will force the variable names ”Sum”, ”Rem” and ”a” to have the values of 9, 2, an 16, respectively. The other important note is that once a variable has been assigned a value, that variable can be used to do mathematics operations in the future. Suppose that we typed and entered the lines above then entered then the variable result would store the value

```
result <- Sum*Rem
```

18.

Often we will have to update the value stored in a specific variable. This can be done in a very simply manner. Suppose we want to take the value stored by the variable ”result” and multiply it by 2. The following line will do exactly that.

```
result <- result*2
```

Discussion. *Let us write a line of code (or a few) to evaluate the polynomial* $x^2 - x + 5$ *at the point* $x = 4$*.*

4 How to write comments in a program

This is one of the more important things a programmer will need to know. Sometimes in a coding file it is helpful to write some commentary to help you (or someone reading your code) understand what is going on. A comment in **R** is any line that begins with the symbol #. Anything after that symbol will be skipped by the computer. That means that we can type plain english after the # symbol and the code will still run. As an example:

```
# The following program will store 90 in the variable b
# If we un-comment the middle line, then the variable b will store the value 86

a <- 15*3
# a <- a-2
b a+a
```

You will be expected to comment any code you turn in so I can understand what your code is doing.

5 The For-Loop in R

Often when writing a computer program you may want to execute a command multiple times. This can be done using a for-loop. The syntax (what you type in your code) for a for-loop is as follows

```
for (i in 1:n){
# Enter here the lines of code you want to be repeated
}
```

The above runs whatever lines of code you want n times. On each iteration of the loop the variable i increases its value by one. Oftentimes you will use the variable i to do some computations. Suppose that we wanted to add up all the numbers from 1 to 100 and store that memory in the variable location "result". The following code will do just that

Discussion. *Now that we know enough, let us write the program that calculates the factorial of a number.*

```
result <- 0
for (i in 1:100){
result <- result + i
}
```

6 The print statement

Writing a code that doesn't tell us what it computes is fairly useless. Therefore we often want our code to output some value. This can be done with the simple "print" function in **R** will do just that. If we want to write a program that doesn't just calculate the sum of the first 200 integers but prints out the answer we could write

```
result <- 0
for (i in 1:200){
result <- result + i
}
print(result)
```