

Assignments to Lecture 2 – More Introduction into R

Danny Arends

Control structures

1a) Generate a random uniform value (between 0 and 1) using the **runif()** function and store this value in a variable with the name 'unknown'.

1b) Use an if statement and use the **cat()** or **print()** function to print out either: *lower* or *higher* if the variable is smaller than 0.5 or bigger than 0.5

1c) Generate a uniform value between -10 and 30, **round()** it to 0 digits behind the comma, and using an if statement check if the value you generated is between 0 and 10 (inclusive), if the variable is not in this range throw a stop error.

2a) Using a *for* loop and sum up all the numbers from 1 to 1000 (inclusive), you can check if your answer is correct by comparing the result to the result of **sum(1:1000)**

2b) Do the same thing as in assignment 2a, but now use a while loop

3) Create a *for* loop that does the following a 100 times:

- * Generate a random number between 0 and 100 and store it in a variable

- * Using a if/else if/else statement check if the variable is lower/higher or equal to 42

- * Using the **cat()** function print one of these three statements, (replace X by the random number generated), make sure you add a newline to the cat statement:

"X is lower than 42", "X is higher then 42" or "42 is the answer to life the universe and everything"

4) Use a *while* or a *for* loop and the **cat()** function to print out a triangle of #, having 12 lines, each line should have one more hashtag then the previous line:

```
#
##
###
####
...
#####
```

Make sure that there are NO spaces between the hashtags, and that there is NO additional whitespace (called trailing white space) before the end of the line.

Tip: Use \n for going to the next line

Escaping

5) Use character escaping and print the following two sentences literally to a file (test.txt), using the **cat()** function:

I say: "Escaping stuff is 'great', but \ and / might be a nuisance."

You are correct, but I think the \t and \b create more problems then a basic "

Check your results by opening the create test.txt file in notepad++

Random variables

6a) Set your random number generator seed to a number of your choice

6b) Using **runif()** generate a vector containing 15 random numbers, between 0 and 10 and store it in a variable called 'random1'

- 6c) Use the **round()** function to round your random numbers in 'random1'
- 6d) Reset your seed to your number, and generate a single random number using **rnorm()**
- 6e) Now repeat steps 0b, and 0c, store the results in variable 'random2'
- 6f) Why is the content of 'random1' and 'random2' not equal to each other, and what do you observe when looking at the sequence of numbers generated?

Functions

- 7) Create a function that returns the result of a coin flip ("Heads", "Tails")

```
flipCoin <- function(){
  #<your code here>
}
```

- 8) Re-use the code you created in assignment 4, but now make it a function (called triangle) that prints an triangle of which the size can be specified by the user. The function signature will look something like, here *size* is the parameter that lets the user specify the number of rows of the triangle:

```
triangle <- function(size){
  #<your code here>
}
```

- 9) Create a function that calculates the factorial ($5! = 5 * 4 * 3 * 2 * 1$) of a given number, the function signature should look like this, here *x* is the function parameter, that represents/holds the input provided by the user:

```
myfactorial <- function(x){
  #<your code here>
}
```

Assignment: fill in the code to make this a working factorial function.

Additional Assignments:

*Your candle fades, as you walk into darkness.
Suddenly you realize you are on your own.*

- Extra 1) The nice thing in R is that we could call our function "!" this then allows us to type:

```
!5
```

to calculate 5!, the assignment is to create a new function called "!", and inside this function call you myfactorial function, test this new function by calculating the factorial of 5, using !5

- Extra 2) Remember that each computational task (+, -, *, /) requires time. Revisit assignment 2 and find a smarter (more efficient) way to do this.

Note: This is actually a question teachers use in elementary school to test children's ability for mathematics... :-)

- Extra 3) Create a function that checks if a number is a prime number (A prime number is one with exactly two positive divisors) use euclidian division

