MT 4113: Computing in Statistics, Assignment 1

Setter: Len Thomas

Set: 17 September 2018. Due: 1 October 2018

The purpose of this assignment is to give you some experience writing simple functions in R, making use of various control structures, as well as get you to think a bit about how pseudo-random numbers are generated.

- Create functions that use the uniform random number generator in R to generate random normal deviates,
- From random normal deviates, create χ^2-distributed deviates,
- From random normal deviates and χ^2-distributed deviates, create random numbers with a Student's t-distribution.
- Marks are out of 50: the result will count 10% towards your final grade.

Background

Later in this course we will discuss the bootstrap and other computer-intensive statistical methods that rely on random re-ordering of data to make inferences. As another example, if you are taking the Bayesian Inference course, you'll see that almost all modern Bayesian analysis uses simulation-based methods, which rely on generation of random numbers. The `random numbers` generated by computers are not really random – they use deterministic algorithms that produce a set of numbers that have the proper distributional properties (e.g., the right moments) and no discernible pattern. Because the algorithms are deterministic, they always produce numbers in the same order, and so to be useful they need to be set off at a `random` start point – the `seed`. More about the generation of random numbers can be found in books shown in the module reading list.

There are many distributions from which we might want to generate random values. Computer algorithms begin by generating numbers from the uniform distribution and then use various tricks that use these uniform deviates to generate values from the target distribution. For this assignment, assume that the random uniform number generator in `R`, `runif()`, does a good job of creating uniformly distributed deviates.

From the output `runif()`, you will write functions that produce random deviates conforming to three distributions (normal, χ^2 and Student's t).

The algorithms are as follows:

Algorithm for normally-distributed deviates

The Box-Muller method (Maindonald 1984) generates a **pair** of independent normally-distributed variates, X_1 and X_2.

Algorithm: Box-Muller method.

- 1. Generate $A \sim \text{uniform}(0,1)$ and $B \sim \text{uniform}(0,1)$
- 2. Let $X_1 = \sin(2\pi A)\sqrt{-2\log_e B}$ and $X_2 = \cos(2\pi A)\sqrt{-2\log_e B}$
- 3. Deliver X_1 and X_2

Generating χ^2-distributed deviates

Larsen and Marx (1981:284) give the following theorem, from which an algorithm immediately follows:

Let Z_1, Z_2, ..., Z_n be n independent standard normal random variables. Then

$$\sum_{i=1}^{n} Z_i^2 \sim \chi_n^2$$

Generating Student's t-distributed deviates

Mood, Graybill and Boes (1974:250) give the following theorem, from which an algorithm immediately follows. If Z has a standard normal distribution ($\sim N(0,1)$) and U has a chi-squared distribution with k degrees of freedom ($\sim \chi_k^2$), and if Z and U are independent, then

$$t = \frac{Z}{\sqrt{U/k}}$$

is distributed with a Student's t-distribution with k degrees of freedom.

Your assignment

Problem statement

- Write an `R` function, `my.rnorm()` that return a vector of pseudo-random values from a normal distribution using the Box-Muller algorithm.
  - The algorithm delivers values with mean 0 and sd 1. You can transform them into values with mean μ and sd σ by multiplying the values by σ and then adding μ.
  - The algorithm describes how to generate `pairs` of normally distributed deviates; think about how your code should behave to produce an odd number of deviates.
  - `rnorm()` accepts vector arguments, but your functions should be designed to accept only scalars. This makes it easier for you to code.
  - Just like the `R` function `rnorm()`, the function should have the following arguments:

Argument name	Description	Default
n	number of values to return	none
mean	mean of values to return	0
sd	standard deviation of values to return	1

- Write an `R` function `my.rchisq()` returning a vector of pseudo-random χ^2-distributed deviates.

Argument name	Description	Default
n	number of values to return	none
df	degrees of freedom of the distribution	1

- Write an `R` function `my.rt()` returning a vector of pseudo-random t-distributed deviates.

Argument name	Description	Default
n	number of values to return	none
df	degrees of freedom of the distribution	1

- These last two functions differ from their `R` counterparts `rchisq()` and `rt()`, in that your functions will not use the non-centrality parameter argument.

Make sure each of your functions are clearly written, with a useful amount of commenting. Pay attention to the tips on good programming style given in the lectures and practicals, including things like spacing between blocks of code, indenting and spaces within lines of code, use of clear variable names, etc. Use vectorization where appropriate. Include appropriate error traps on the inputs; when the function detects an incorrect input it should stop program execution[1] with *exactly* the error message `invalid arguments`.

Update 17th Sept 2018: Please do not use any additional `R` libraries in your solution beyond those loaded in a standard `R` installation. Your solution therfore should not include the `library()` or `require()` functions.

What to hand in

Before 11:59am 1 October, upload the following to MMS:

- A single text file named `asmt1.r` containing the three functions (suitably commented) you have written. *Please be careful to exactly follow the instructions about function names, arguments, returned values, etc.* – the functions must be drop-in replacements for `rnorm()`, `rchisq()` and `rt()`. I will be using automated testing to check the functions, if you do not follow the instructions exactly, my tests will fail on your code, and you will not get credit for your work.

Here is an example of an automated test - checking that the function `my.rnorm()` produces a vector of 10 numbers when asked:

```r
source('asmt1.r')
x <- my.rnorm(n=10)
pass.test <- (length(x)==10 & is.numeric(x))
# pass.test takes on the value TRUE if the object 'x' contains 10 numeric values
```

- Also within the text file named `asmt1.r` add one or more additional **functions** (not open code) used to test the `rnorm()`, `rchisq()` and `rt()` functions. The test function(s) will not be part of the automated testing. This part of the assignment is open-ended; I wish to see what creativity you can use to test your own code. However the test function(s) **must** be functions and contain sufficient documentation so that I know what it is the test(s) are trying to accomplish. If your test code is not written in the form of functions, they will interfere with my automated testing procedure, causing possible loss of marks.

If you hand work in late, it will be subject to the late work penalty described on the School's web page [http://www.st-andrews.ac.uk/maths/current/ug/information/latepenalties/] and summarised as *A late piece of work is penalised with an initial penalty of 15% of the maximum available mark, and then a further 5% per 8-hour period, of part thereof.* Do not wait until the morning of the due date to try uploading your work to MMS – if the upload fails at that point, your work will be discounted as late.

Because this assignment counts towards your final grade, it is important that you do not collaborate with others in completing the work. *You should be comfortable with the following statement, which you must include as a comment at the beginning of your code file*

```
# I confirm that the attached is my own work, except where clearly indicated in the text.
```

Please note that failure to include this comment at the beginning of your code file will result in deduction of marks from your assignment.

If you got stuck and needed to ask your peers, please use comments to tell me in your code file where you got help from others. If you are really stuck, email me. For more information, see the *Academic misconduct* section of the university web site [http://www.st-andrews.ac.uk/students/rules/academicpractice/]. This holds for all assessed project work on this course. Plagiarism cannot be tolerated on this or any other assignment for this module.

[1]See the function `stop()` for how to do this.

Marking scheme

There are 50 marks on offer for the assignment.

- Up to 30 marks for producing working functions `my.rnorm()`, `my.rchisq()` and `my.rt()`. As stated above, I will use automated testing to check your functions work, including trapping input errors. This means I will run a battery of tests on your functions and compare the results with what they should produce.
  - If you are unable to create `my.rnorm()` according to the Box-Muller algorithm provided, you may receive partial credit for the assignment by calling the R function, `rnorm()` in your χ^2 and Student's t-distribution random variate generation. Marks you would have earned in the testing of `my.rnorm()` will be lost by adopting this strategy.
- Up to 12 marks for good programming style in each function (presence and usefulness of comments, use of indenting, presence of error checks, etc.). I will assess this by making a visual check of your code, and I will assign marks for style out of 12.
- Up to 8 marks are available for the testing functions you produce. I will check them for accuracy, creativity and ambitiousness. Note the marks available for this portion of the assignment is less than the number of marks available for good programming style; allocate your effort accordingly.

References

- Larsen, R.J. and M.L. Marx. 1981. An introduction to mathematical statistics and its applications. Prentice-Hall, Inc.
- Maindonald, John. 1984. Statistical computation. John Wiley and Sons.
- Mood, A.M., F.A. Graybill, and D.C. Boes. 1974. Introduction to the theory of statistics. Third Edition. McGraw Hill.