

Exam

R programming (732G36)

Time: 8-12, 2015-11-25
Material: The extra material is included in the zip-file **exam_material.zip**.
Grades: A = 19-20 points.
B = 17-18 points.
C = 12-16 points.
D = 10-11 points.
E = 8-9 points.
F = 0-7 points.

Instructions

Write your code in an R script file named **Main.R**. The R code should be complete and readable code, possible to run by copying directly into a script. Comment directly in the code whenever something needs to be explained or discussed. Follow the instructions carefully.

Problem 1 (6 p)

a) Create a function you call `rdices()` with three arguments, `n`, `eyes` and `dices`. The functions should simulate throwing dices. The `eyes` argument should specify the number of eyes of the dice (six should be the default value), `dices` should specify the number of dices that is beeing throwned (two should be the default) and `n` is the number of throws that has been done. The function should return a vector of length `n` with the sum of the eyes in the thrown dices.

```
rdices(5)

[1] 9 5 8 6 7

mean(rdices(100000, dices = 1))

[1] 3.50439
```

b) What is the complexity of this algorithm ith regard to `n`. Assume that drawing a random draw is a constant operation.

c) Visualize 1000 draws from your function (with the default values) as histogram using `ggplot2`.

Problem 2 (7 p)

a) Create a function called `inverse_triangular_block_matrix()` that takes matrices **A**, **B** and **C** and return their inverse as follows:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{C}^{-1} \\ \mathbf{0} & \mathbf{C}^{-1} \end{bmatrix}$$

```
inverse_triangular_block_matrix(diag(2), 2*diag(2), 3*diag(2))
```

```
      [,1] [,2]      [,3]      [,4]
[1,]     1     0 -0.666667  0.000000
[2,]     0     1  0.000000 -0.666667
[3,]     0     0  0.333333  0.000000
[4,]     0     0  0.000000  0.333333
```

```
inverse_triangular_block_matrix(diag(1), -1*diag(1), 5*diag(1))
```

```
      [,1] [,2]
[1,]     1  0.2
[2,]     0  0.2
```

b) Implement a test suite with unit tests that check that the result is of the correct class, of the right size/dimensions and that the function correctly return one of the examples above.

Problem 3 (7 p)

a) Create a function to simulate draws from a multivariate normal distribution. The function should be called `rmvn()` and take the arguments **n** (number of draws), **mu** (a vector of means of length m) and **Sigma** (a square matrix of size $m \times m$). Below is the description of how to do a multivariate draw taken from Wikipedia:

A widely used method for drawing (sampling) a random vector **x** from the N -dimensional multivariate normal distribution with mean vector μ and covariance matrix Σ works as follows:

1. Find any real matrix **A** such that $\mathbf{A}\mathbf{A}^T = \Sigma$. When Σ is positive-definite, the Cholesky decomposition is typically used, and the extended form of this decomposition can always be used (as the covariance matrix may be only positive semi-definite) in both cases a suitable matrix **A** is obtained. [...]
2. Let $\mathbf{z} = (z_1, \dots, z_N)^T$ be a vector whose components are N independent standard normal variates.
3. Let **x** be $\mu + \mathbf{A}\mathbf{z}$. This has the desired distribution [...].

```
Sigma <- matrix(c(1,0.5,0.5,1), ncol=2)
mu <- c(2,5)
rmvn(3, mu, Sigma)
```

```
      [,1]      [,2]
[1,] 1.88421 5.06680
[2,] 1.28087 4.24243
[3,] 1.56805 5.54725

var(rmvn(100000, mu, Sigma))

      [,1]      [,2]
[1,] 0.998719 0.498536
[2,] 0.498536 1.005740
```

b) Document your function using **roxygen2**. The documentation should contain the title, description, the arguments and the resulting value of the function.

Good luck!