LINKÖPINGS UNIVERSITET
Department of information and computer science
Statistics and Machine learning
Måns Magnusson

Exam

Exam solutions (to 732G36 and 732A50)

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

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]  4 10  3  5  4

mean(rdices(100000, dices = 1))

[1] 3.50759
```

Suggested solution:

```
function(n, dices=2, eyes=6){
  res <- integer(n)
```

```
  for(i in 1:n){
    res[i] <- sum(sample(1:eyes, dices, replace = TRUE))
  }
  res
}
```

---

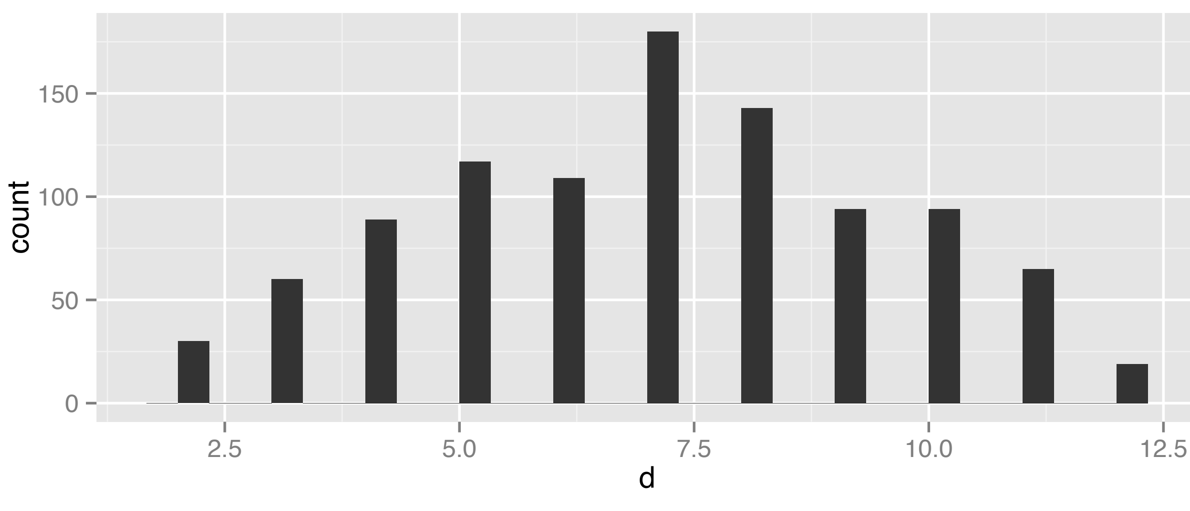
b) What is the complexity of this algorithm ith regard to `n`. Assume that drawing a random draw is a constant operation.

Suggested solution:
The complexity is linear, $O(n)$, in the input size.

---

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

Suggested solution:



Problem 2

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

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

```r
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
```

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

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

Suggested solution:

```r
function(A,B,C){
  zero_mat <- matrix(0, ncol=ncol(A), nrow=nrow(C))
  top_block <- cbind(solve(A), -solve(A)%*%B%*%solve(C))
  low_block <- cbind(zero_mat, solve(C))
  rbind(top_block, low_block)
}
```

---

b) Implement a test suite withs 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.

Suggested solution:

```r
library(testthat)
test_that("my inverse block matrix function is working", {
  mat <- inverse_triangular_block_matrix(diag(1), -1*diag(1), 5*diag(1))
  expect_is(mat, "matrix")
  expect_equal(dim(mat), c(2,2))
  expect_equal(as.vector(mat)*10, c(10, 0, 2, 2))
})
```

Problem 3

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 $\mathbf{x}$ from the N-dimensional multivariate normal distribution with mean vector μ and covariance matrix Σ works as follows:

1. Find any real matrix $\mathbf{A}$ 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 $\mathbf{A}$ is obtained. [...]

2. Let $\mathbf{z} = (z_1, \ldots, z_N)^T$ be a vector whose components are N independent standard normal variates.

3. Let $\mathbf{x}$ 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,] 2.08474 4.13519
[2,] 2.81584 4.37357
[3,] 1.55596 4.16468

var(rmvn(100000, mu, Sigma))

         [,1]     [,2]
[1,] 1.001406 0.502222
[2,] 0.502222 1.004252
```

Suggested solution:

```
function(n, mu, Sigma){
  x <- matrix(0.0, ncol=length(mu), nrow=n)
  A <- chol(Sigma)
  for(i in 1:n){
    z <- matrix(rnorm(length(mu)), ncol=2)
    x[i,] <- z %*% A + mu
  }
  x
}
```

---

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

Suggested solution:

```r
#' @title Multivariate draws
#' @param n The number of draws
#' @param mu A vector of expected values
#' @param Sigma A covariance matrix
#' @description A function that draws multivariate variables using cholesky decomposition.
#' @value Returns a matrix of size n * length(mu)
```

Good luck!