Lab3

Cui Qingxuan, Nisal Amashan

2025-02-11

Contents

1 Collaborations

Nisal Amashan: Responsible for the question 1.

Cui Qingxuan: Responsible for the question 2.

2 Question 1

3 Question 2

3.1 Generate a Random Vector using the 2 Methods

3.1.1 Box-Muller Method

Measure the time for generating 10 000 000 numbers:

```
##    user  system elapsed
##    2.90    0.50    3.58
```

3.1.2 Package mvtnorm

The reason I use it:

For generating correlated multivariate normal vectors at scale, mvtnorm provides a streamlined, efficient, and statistically rigorous solution. It eliminates manual matrix operations, ensures correctness, and scales effortlessly to large datasets—making it the optimal choice for this task.

Measure the time for generating 10 000 000 numbers:

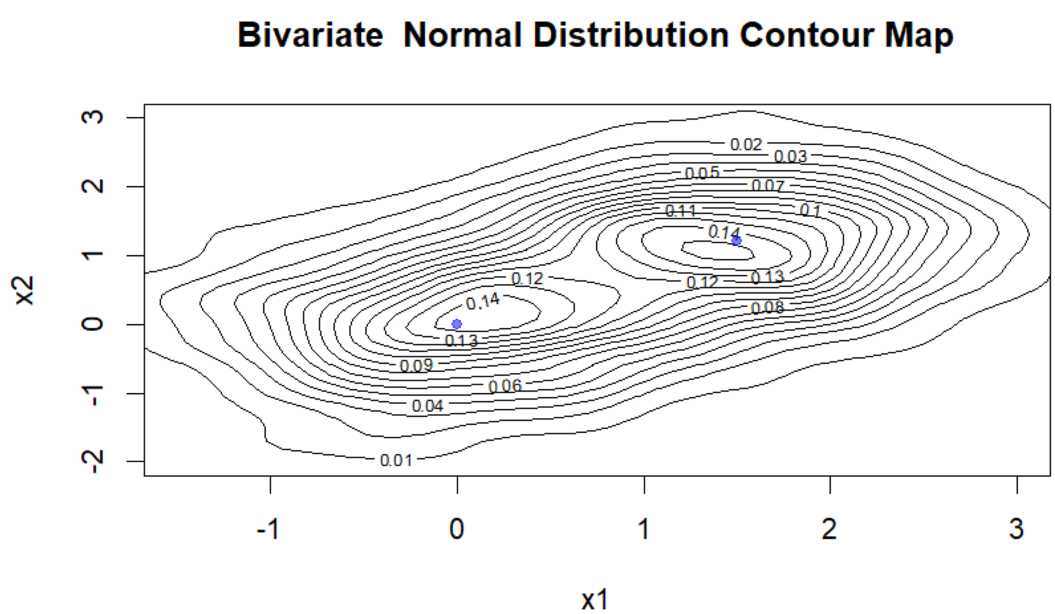
```
##    user  system elapsed
##    1.08    0.32    1.53
```

3.1.3 Compare the Time of 2 Methods

Box-Muller method takes roughly 50% more time than mvnorm package to generate a random variable.

3.2 Generate the Random Vectors from 2 Distributions

3.2.1 Plot the Random Variables



3.2.2 Discussion

It looks satisfactory.

Based on the contour map, we can see the contour lines in the graph are inclined ellipses, and the interval between contour lines becomes progressively wider as the height increases, and is symmetrical about the mean point (the blue point), which in summary is consistent with the properties of a Bivariate Normal Distribution image.

4 Appendix

```r
normRVgen = function(n, mu, sigma){
  # create 2 rv vector uniformly distribute
  # u1 represents R
  # u2 represents angle
  u1 = runif(n, min=0, max=1)
  u2 = 2*pi*runif(n, min=0, max=1)

  # Generate X=[x1, x2] ~ N(0,1)
  x1 = sqrt(-2*log(u1))*cos(u2)
  x2 = sqrt(-2*log(u1))*sin(u2)
  x = rbind(x1, x2)
  # Transform to Z via Z = A.T @ X + mu
  # Cholesky Decomposition -> get A from Sigma = A.T@A
  At = t(chol(sigma))
  Z = At %*% x + mu

  return (Z)
}

n =  1000
mu = c(0,0)
sigma = matrix(c(0.6,0,0,0.6), nrow=2)
start = proc.time()
rv = normRVgen(n,mu,sigma)
end = proc.time()
print(end - start)

library(mvtnorm)
start = proc.time()
rv_buildin <- rmvnorm(n, mean = mu, sigma = sigma)
end = proc.time()
print(end-start)

# Generate 500 values for each distribution
n = 500
rv_d1 = normRVgen(n, mu, sigma)

mu_d2 = c(1.5, 1.2)
sigma_d2 = matrix(c(0.5,0,0,0.5), nrow=2)
rv_d2 = normRVgen(n, mu_d2, sigma_d2)
# Bind them and shuffle
rv_mix = as.data.frame(t(cbind(rv_d1, rv_d2)))
set.seed(123)
rv_shuffled = rv_mix[sample(nrow(rv_mix)), ]
colnames(rv_shuffled) = c("x1", "x2")

# Generate 500 values for each distribution
n = 500
rv_d1 = normRVgen(n, mu, sigma)

mu_d2 = c(1.5, 1.2)
```

```r
sigma_d2 = matrix(c(0.5,0,0,0.5), nrow=2)
rv_d2 = normRVgen(n, mu_d2, sigma_d2)
# Bind them and shuffle
rv_mix = as.data.frame(t(cbind(rv_d1, rv_d2)))
set.seed(123)
rv_shuffled = rv_mix[sample(nrow(rv_mix)), ]
colnames(rv_shuffled) = c("x1", "x2")

# Plot the Function
library(MASS)
z <- kde2d(rv_shuffled$x1, rv_shuffled$x2, n = 50)
contour(z, xlab = "x1", ylab = "x2", main = "Binary Normal Distribution Contour Map")
# points(rv_shuffled$x1, rv_shuffled$x2, col = rgb(0, 0, 1, 0.5), pch = 19,cex = 0.3)
```