STA 531 HW1 (RNG)

Daniel Truver

1/23/2018

(1) The RANDU generator

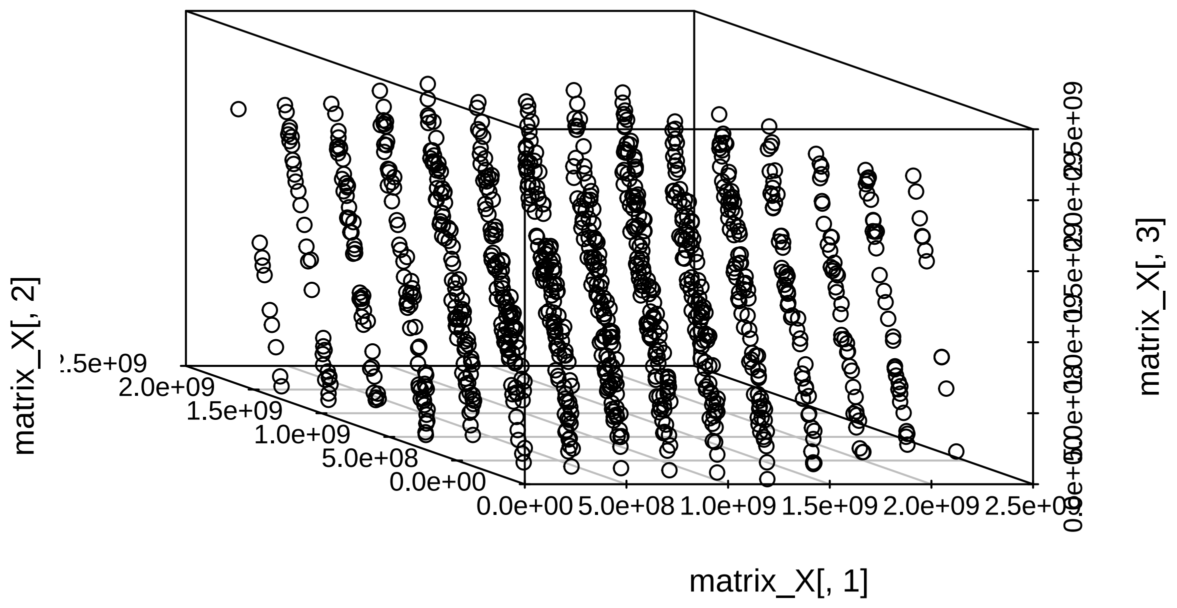
```
N = 1000
X = rep(NA, N)
X[1] = 65000
for (n in 2:N){
  X[n] = (65539 * X[n-1]) %% 2^31
}
matrix_X = matrix(NA, nrow = N, ncol = 4)
matrix_X[,1] = X
matrix_X[,2] = c(X[2:N], NA)
matrix_X[,3] = c(X[3:N], NA, NA)
matrix_X[,4] = (6*matrix_X[,2] - 9*matrix_X[,1]) %% 2^31
matrix_X = matrix_X[1:(N-2),]
equality = matrix_X[,4] == matrix_X[,3]
cat("Out of", nrow(matrix_X), "iterations of the algorithm,",
    sum(equality), "follow the given pattern.")
```

```
## Out of 998 iterations of the algorithm, 998 follow the given pattern.
```

From the above script, we know that the following pattern emerges in the RANDU generator.

$$X_{t+1} = (6X_t - 9X_{t-1}) \mod 2^31$$

```
scatterplot3d(x = matrix_X[,1], y = matrix_X[,2], z = matrix_X[,3], angle = 150)
```



(2) Box-Muller algorithm

Let $X_1.X_2$ be independent standard normal random variables.

(a) Finding the distribution of $R = \sqrt{X_1^2 + X_2^2}$ and $\theta = \tan^{-1} X_1/X_2$.

We will use the method of transformations.

$$X_1 = r\cos\theta\tan\theta = r\sin\theta$$

$$X_2 = r\sin\theta/\tan\theta = r\cos\theta$$

$$f(x_1, x_2) = (1/2\pi)\exp(-(x_1 + x_2)/2)$$

Then, we get

$$\begin{aligned} f(r,\theta) &= \frac{1}{2\pi}\exp\left(-\frac{1}{2}R^2\right)\left|\det\begin{bmatrix}\sin\theta & \cos\theta \\ r\cos\theta & -r\sin\theta\end{bmatrix}\right| \\ &= \left(\frac{1}{2\pi}\right)\cdot R\exp\left(-\frac{1}{2}R^2\right)\end{aligned}$$

This factors nicely into the product of the distributions for θ and R^2.

$$\theta \sim U(0, 2\pi), \quad R^2 \sim \chi_2^2 \implies R \sim \chi_2$$

(b) Generating R, θ by the inverse cdf method. Take F, G to be the cdf's of R, θ, respectively.

We draw $U_1, U_2 \sim \text{unif}(0, 1)$ then take $\theta = 2\pi U_1 = F^{-1}(U_1)$ and $R^2 = -2\log(1 - U_2) = G^{-1}(U_2)$.

We now have precisely the form we need to implement the Box-Muller algorithm.

$$X_1 = \sqrt{-2\log(1-U_2)}\cos(2\pi U_1), \quad X_2 = \sqrt{-2\log(1-U_2)}\sin(2\pi U_1)$$

Fortunately, $1 - U_2$ also has a uniform distribution, so we fit the bill for use in the Box-Muller algorithm.

(c) Comparing Box-Muller to 12 independent uniform r.v.'s

```r
Nsim = 10000
set.seed(2018)
sim.norm.clt = function(nsim){
  normal_clt = rep(NA, nsim)
  for (i in 1:nsim){
    U = runif(12)
    X = sum(U) - 6
    normal_clt[i] = X
  }
  return(normal_clt)
}
sim.norm.box = function(nsim){
  normal_box_1 = rep(NA, nsim/2)
  normal_box_2 = rep(NA, nsim/2)
  for (i in 1:(nsim/2)){
    U = runif(2)
    R = sqrt(-2*log(U[2]))
    theta = 2*pi*U[1]
    X_1 = R*cos(theta)
    X_2 = R*sin(theta)
    normal_box_1[i] = X_1
    normal_box_2[i] = X_2
  }
  return(c(normal_box_1, normal_box_2))
}
normal_clt = sim.norm.clt(Nsim)
normal_box = sim.norm.box(Nsim)
clt.time = system.time(sim.norm.clt(Nsim))
box.time = system.time(sim.norm.box(Nsim))
library(ggplot2)
ggplot(data = data.frame(normal_clt, normal_box)) +
  geom_density(aes(normal_clt, color = "CLT")) +
  geom_density(aes(normal_box, color = "BoxMuller")) +
  ggtitle("The pretty much indistiguishable normal simulations")
```

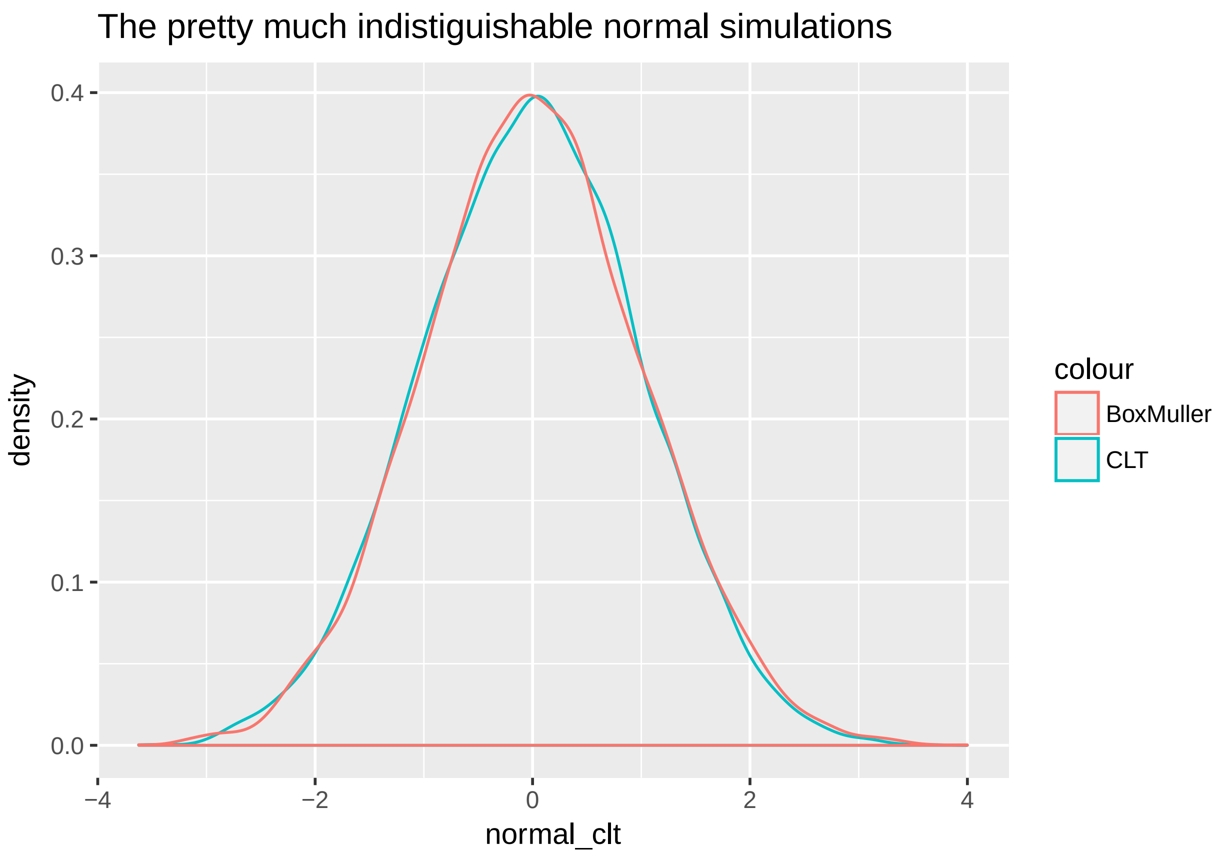


Table 1: Comparison of Run time for CLT and Box-Muller methods, N =10000

	user	system	elapse
CLT	0.031	0.001	0.032
BoxMuller	0.012	0.000	0.013

It appears that the Box-Muller method performs better.

(3) Generating Beta r.v.'s

Let $Y_1 \sim Ga(\alpha, 1), Y_2 \sim Ga(\beta, 1)$. We will derive the desired result using the method of transformations. We first note the joint distribution function of Y_1, Y_2 is

$$f(y_1, y_2) = \frac{1}{\Gamma(\alpha)\Gamma(\beta)} y_1^{\alpha-1} y_2^{\beta-1} \exp(-(y_1 + y_2))$$

Let

$$u_1 = \frac{y_1}{y_1 + y_2}, \quad u_2 = y_1 + y_2$$

For here, we see that

$$y_1 = u_1 u_2, \quad y_2 = u_2(1 - u_1), \quad J = det \begin{bmatrix} u_2 & u_1 \\ -u_2 & 1 - u_1 \end{bmatrix} = u_2$$

Therefore, the joint distribution of U_1, U_2 is

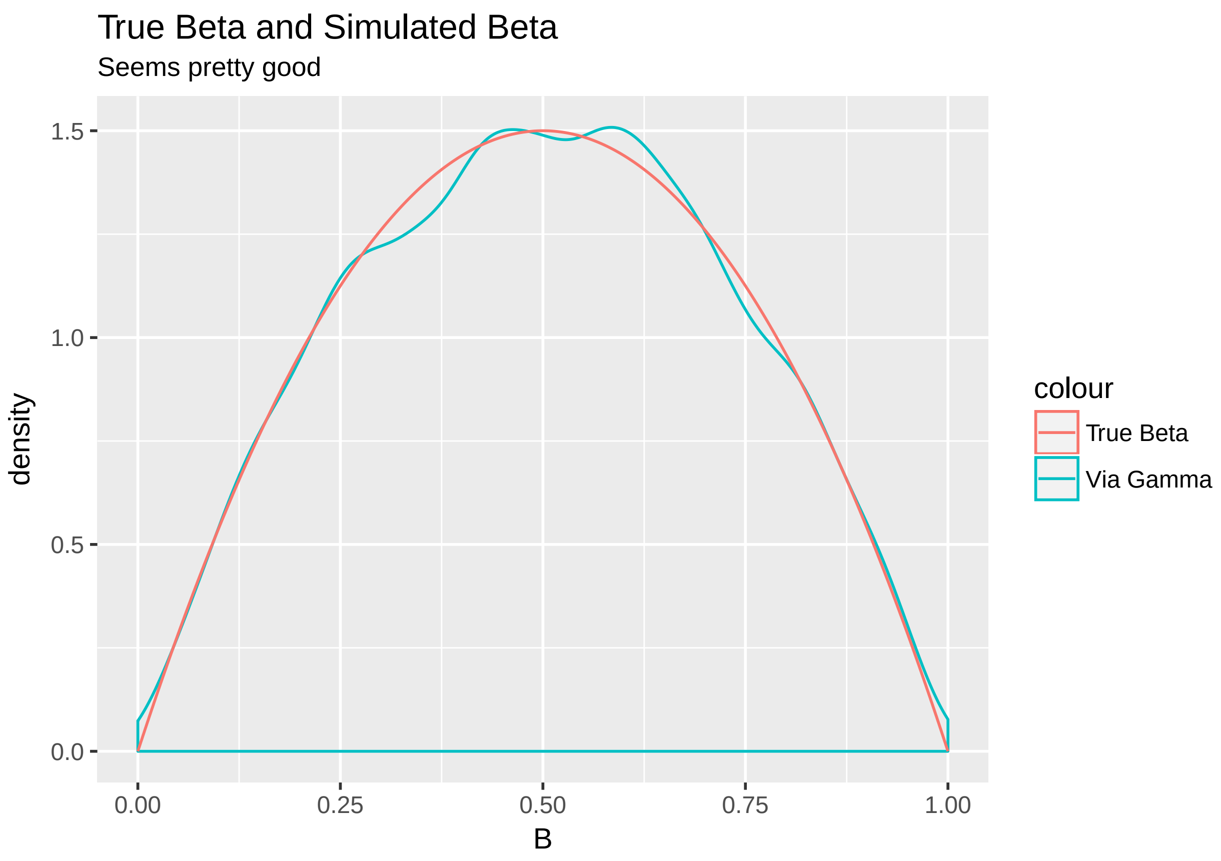
$$\begin{aligned} f(u_1 u_2, u_2(1-u_1))|J| &= \frac{1}{\Gamma(\alpha)\Gamma(\beta)} (u_1 u_2)^{\alpha-1} (u_2(1-u_1))^{\beta-1} \exp(-u_2)|u_2| \\ &= u_1^{\alpha-1}(1-u_1)^{\beta-1} \cdot k(u_2), \quad 0 < u_2, 0 < u_1 < 1 \\ &\implies \int_{U_2} u_1^{\alpha-1}(1-u_1)^{\beta-1} \cdot k(u_2) \mathrm{d}u_2 = k \cdot u_1^{\alpha-1}(1-u_1)^{\beta-1} \end{aligned}$$

We recognize this as the form of the Beta distribution with parameters α, β.

This suggests to us that we may generate Beta random variables by generating two gamma random variables, G_1, G_2 and calculating the ratio

$$B = \frac{G_1}{G_1 + G_2}$$

```r
alpha = 2
beta = 2
g_1 = rgamma(10000, alpha, 1)
g_2 = rgamma(10000, beta, 1)
B = g_1/(g_1 + g_2)
x = seq(0,1, length.out = 10000)
B_true = dbeta(x,alpha,beta)
ggplot(data = data.frame(B, B_true, x)) +
  geom_density(aes(B, color = "Via Gamma")) +
  geom_line(aes(x, B_true, color = "True Beta")) +
  ggtitle("True Beta and Simulated Beta",
          subtitle = "Seems pretty good")
```


True Beta and Simulated Beta
Seems pretty good
density
B
0.0
0.5
1.0
1.5
0.00
0.25
0.50
0.75
1.00
colour
True Beta
Via Gamma