# Stats 102C - Lecture 4-2: Box-Mueller Sampling and Multivariate Sampling

Michael Tsiang (adapted by Miles Chen)

Department of Statistics

Week 4 Wednesday

UCLA

Section 1

# Box-Muller Transform

# Box-Muller Transform for Normal Distribution

- The Box-Muller transform (George Box and Mervin Muller, 1958) is a method to transform two uniform random variables into a pair of independent standard normal random variables.
- The main idea is to change coordinates from Cartesian to polar coordinates.

# Box-Muller Transform

Let X and Y be independent standard normal random variables.

The joint PDF of X and Y is given by

$$\begin{aligned} f_{xy}(x,y) &= f(x)f(y) \\ &= \frac{1}{\sqrt{2\pi}}e^{-x^2/2} \cdot \frac{1}{\sqrt{2\pi}}e^{-y^2/2} \\ &= \frac{1}{2\pi}e^{-\frac{x^2+y^2}{2}} \end{aligned}$$

# Box-Muller Transform

The relationship between Cartesian coordinates $(x, y)$ and polar coordinates $(r, \theta)$

$$x = r\cos\theta$$

$$y = r\sin\theta$$

We change $f_{xy}(x, y)$ to polar coordinates:

$$f_{XY}(x, y)dxdy = f_{R\Theta}(r, \theta)drd\theta$$

# Box-Muller Transform

It's probably been a while, but from multivariate calculus, the change of variables is given by:

$$\begin{aligned} f_{R\Theta}(r,\theta) &= f_{XY}(x,y)\frac{dxdy}{drd\theta} \\ &= f_{XY}(x,y)\underbrace{\left|\frac{\partial(x,y)}{\partial(r,\theta)}\right|}_{=J}, \end{aligned}$$

where $J$ is the Jacobian

$$J = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix} = \begin{vmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{vmatrix} = r\cos^2\theta + r\sin^2\theta = r$$

Since $x^2 + y^2 = r^2$, then

$$f_{R\Theta}(r, \theta) = f_{XY}(x, y) \cdot r = \frac{1}{2\pi} e^{-\frac{r^2}{2}} \cdot r$$

For $r \geq 0$ and $\theta \in [0, 2\pi)$, we have

$$f_{R\Theta}(r, \theta) dr d\theta = \frac{1}{2\pi} e^{-\frac{r^2}{2}} \cdot r dr d\theta$$

Applying another change of variables from $(r, \theta)$ to $(r^2, \theta)$, we have

$$dr^2 = 2rdr, \text{ so } rdr = \frac{1}{2} dr^2$$

The joint PDF $f_{R\Theta}(r, \theta)drd\theta$, for $r \geq 0$ and $\theta \in [0, 2\pi)$, can now be written as

$$
\begin{aligned}
f_{R\Theta}(r, \theta)drd\theta &= f_{R^2\Theta}(r^2, \theta)dr^2d\theta \\
\frac{1}{2\pi}e^{-\frac{r^2}{2}} \cdot rdrd\theta &= \frac{1}{2\pi}e^{-\frac{r^2}{2}} \cdot \frac{1}{2}dr^2d\theta \\
&= \left(\frac{1}{2}e^{-\frac{r^2}{2}}dr^2\right)\left(\frac{1}{2\pi}d\theta\right) \\
&= f_{R^2}(r^2)dr^2 \cdot f_{\Theta}(\theta)d\theta,
\end{aligned}
$$

which shows:

- $R^2 \perp \Theta$ (i.e., $R^2$ and $\Theta$ are independent)
- $\Theta \sim \text{Unif}(0, 2\pi)$
- $R^2 \sim \text{Exp}(\lambda = \frac{1}{2})$

- Generate $\Theta \sim \text{Unif}(0, 2\pi)$
- Generate $V \sim \text{Exp}(\lambda = \frac{1}{2})$ (i.e., $V = R^2$) and compute

$$R = \sqrt{V}$$

- Compute

$$X = R\cos\Theta$$
$$Y = R\sin\Theta$$

Then $X, Y \sim \mathcal{N}(0, 1)$

Note that we can use $U \sim \text{Unif}(0, 1)$ to sample from $\text{Exp}(\lambda = \frac{1}{2})$ using the inverse CDF method:

$$-\frac{1}{\lambda}\log U = -2\log U \sim \text{Exp}(\lambda = \frac{1}{2})$$

# Box-Muller algorithm:

1. Generate $U \sim \text{Unif}(0, 1)$ and compute

$$\Theta = 2\pi U$$

2. Generate $V \sim \text{Unif}(0, 1)$ and compute

$$R = \sqrt{-2 \log V}$$

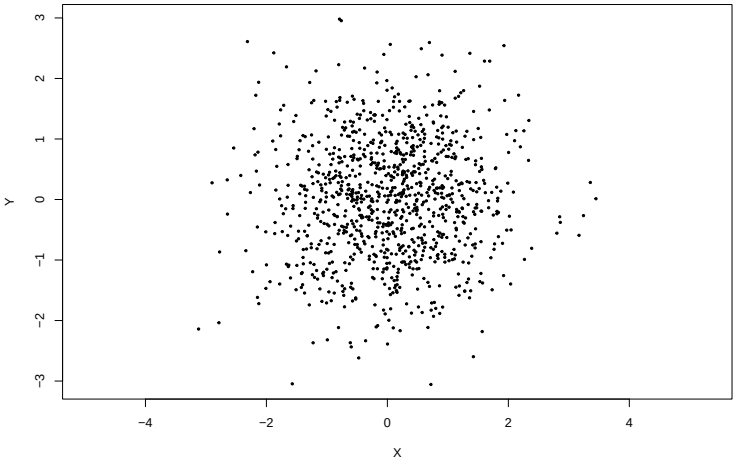
3. Compute

$$X = R \cos \Theta$$
$$Y = R \sin \Theta$$

Then $X, Y \sim \mathcal{N}(0, 1)$

```r
set.seed(9999)
n <- 1000
# Generate n points from Unif(0,1)
U <- runif(n,0,1)
# Compute Theta
Theta <- 2*pi*U
# Generate n points from Unif(0,1)
V <- runif(n,0,1)
# Compute R
R <- sqrt(-2*log(V))
# Compute X and Y
X <- R*cos(Theta)
Y <- R*sin(Theta)
```

```r
plot(X, Y, cex = 0.4, asp = 1, pch = 19)
```



https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform

https://upload.wikimedia.org/wikipedia/commons/1/1f/Box-Muller_transform_visualisation.svg

# Normal Distribution

We can now assume we can generate samples from $N(0, 1)$.

How can we use samples from $N(0, 1)$ to sample from $W \sim N(\mu, \sigma^2)$, for any $\mu \in R, \sigma^2 > 0$?

# Normal Distribution

- Generate $Z \sim N(0, 1)$.
- Then $W = \mu + \sigma Z \sim N(\mu, \sigma^2)$.

Why?

- $W$ is normally distributed.
- $E(W) = E(\mu + \sigma Z) = \mu + \sigma E(Z) = \mu$
- $Var(W) = Var(\mu + \sigma Z) = Var(\sigma Z) = \sigma^2 Var(Z) = \sigma^2$

Section 2

# Generating a Multivariate Sample

# Multivariate Sampling

Sometimes we need to generate random values from a joint distribution of multiple random variables.

We'll look at generating data from bivariate distributions.

Eventually, we'll want to generate samples from higher dimensional distributions.

# Bivariate Discrete Distribution

Let's say random variables $X$ and $Y$ have the following joint PMF:

| X/Y | y = 0 | y = 1 |
|---|---|---|
| x = 0 | 0.2 | 0.6 |
| x = 1 | 0.1 | 0.1 |

How can we generate samples from this distribution?

# Bivariate Discrete Distribution

We map the pairs $X$ and $Y$ onto interval (0, 1).

Let $p_{xy} = \Pr(X = x, Y = y)$

$$
\begin{aligned}
F_1 &= p_{00} &&= 0.2 \\
F_2 &= p_{00} + p_{10} &&= 0.3 \\
F_3 &= p_{00} + p_{10} + p_{01} &&= 0.9 \\
F_4 &= p_{00} + p_{10} + p_{01} + p_{11} &&= 1.0
\end{aligned}
$$

1. Generate $U \sim \text{Unif}(0, 1)$
2. Then

$$
(X, Y) = \begin{cases} (0,0) & \text{if } 0 < U \leq F_1 \\ (1,0) & \text{if } F_1 < U \leq F_2 \\ (0,1) & \text{if } F_2 < U \leq F_3 \\ (1,1) & \text{if } F_3 < U < 1 \end{cases}
$$

# A more complex distribution

| X / Y | y = 1 | y = 2 | y = 3 | y = 4 |
|---|---|---|---|---|
| x = 1 | 0.02 | 0.06 | 0.03 | 0.05 |
| x = 2 | 0.03 | 0.04 | 0.02 | 0.05 |
| x = 3 | 0.04 | 0.2 | 0.1 | 0.1 |
| x = 4 | 0.01 | 0.1 | 0.05 | 0.1 |

$$
\begin{aligned}
F_1 &= p_{11} &&= 0.02 \\
F_2 &= p_{11} + p_{21} &&= 0.05 \\
F_3 &= p_{11} + p_{21} + p_{31} &&= 0.09 \\
\vdots &= \vdots &&= \vdots \\
F_{15} &= p_{11} + p_{21} + p_{31} + \ldots + p_{34} &&= 0.9 \\
F_{16} &= p_{11} + p_{21} + p_{31} + \ldots + p_{34} + p_{44} &&= 1.0
\end{aligned}
$$

1. Generate $U \sim \text{Unif}(0, 1)$
2. Then

$$(X, Y) = \begin{cases} (1,1) & \text{if } 0 < U \leq F_1 \\ (2,1) & \text{if } F_1 < U \leq F_2 \\ \vdots & \vdots \\ (3,4) & \text{if } F_1 4 < U \leq F_1 5 \\ (4,4) & \text{if } F_1 5 < U < 1 \end{cases}$$

# Code Part 1

```
prob_matrix <- matrix(
  c(0.02, 0.06, 0.03, 0.05,
    0.03, 0.04, 0.02, 0.05,
    0.04, 0.2 ,  0.1, 0.1,
    0.01, 0.1 , 0.05, 0.1),
  byrow = TRUE, nrow = 4)

cum_prob <- cumsum(as.vector(prob_matrix))

cum_prob
```

```
##  [1] 0.02 0.05 0.09 0.10 0.16 0.20 0.40 0.50 0.53 0.55 0.65 0.70 0.75 0.80 0.90 1.00
```

```
prob_matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,] 0.02 0.06 0.03 0.05
## [2,] 0.03 0.04 0.02 0.05
## [3,] 0.04 0.20 0.10 0.10
## [4,] 0.01 0.10 0.05 0.10
```

```
x_vals <- rep(1:4, 4)
y_vals <- rep(1:4, each = 4)
print(rbind(cum_prob, x_vals, y_vals))
```

```
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16]
## cum_prob 0.02 0.05 0.09  0.1 0.16  0.2  0.4  0.5 0.53  0.55  0.65   0.7  0.75   0.8   0.9     1
## x_vals   1.00 2.00 3.00  4.0 1.00  2.0  3.0  4.0 1.00  2.00  3.00   4.0  1.00   2.0   3.0     4
## y_vals   1.00 1.00 1.00  1.0 2.00  2.0  2.0  2.0 3.00  3.00  3.00   3.0  4.00   4.0   4.0     4
```

```r
# one value
u <- 0.01
print(rbind(cum_prob, x_vals, y_vals))
```

```
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16]
## cum_prob 0.02 0.05 0.09  0.1 0.16  0.2  0.4  0.5 0.53  0.55  0.65   0.7  0.75   0.8   0.9     1
## x_vals   1.00 2.00 3.00  4.0 1.00  2.0  3.0  4.0 1.00  2.00  3.00   4.0  1.00   2.0   3.0     4
## y_vals   1.00 1.00 1.00  1.0 2.00  2.0  2.0  2.0 3.00  3.00  3.00   3.0  4.00   4.0   4.0     4
```

```r
position <- min(which( u < cum_prob)) # compare u to cum_prob and finds the first value that is greater
position
```

```
## [1] 1
```

```r
x_vals[position] # value to save for X
```

```
## [1] 1
```

```r
y_vals[position] # value to save for Y
```

```
## [1] 1
```

```r
# one value
u <- 0.03
print(rbind(cum_prob, x_vals, y_vals))
```

```
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16]
## cum_prob 0.02 0.05 0.09  0.1 0.16  0.2  0.4  0.5 0.53  0.55  0.65   0.7  0.75   0.8   0.9     1
## x_vals   1.00 2.00 3.00  4.0 1.00  2.0  3.0  4.0 1.00  2.00  3.00   4.0  1.00   2.0   3.0     4
## y_vals   1.00 1.00 1.00  1.0 2.00  2.0  2.0  2.0 3.00  3.00  3.00   3.0  4.00   4.0   4.0     4
```

```r
position <- min(which( u < cum_prob)) # compare u to cum_prob and finds the first value that is greater
position
```

```
## [1] 2
```

```r
x_vals[position] # value to save for X
```

```
## [1] 2
```

```r
y_vals[position] # value to save for Y
```

```
## [1] 1
```

```r
# one value
u <- 0.89
print(rbind(cum_prob, x_vals, y_vals))
```

```
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16]
## cum_prob 0.02 0.05 0.09  0.1 0.16  0.2  0.4  0.5 0.53  0.55  0.65   0.7  0.75   0.8   0.9      1
## x_vals   1.00 2.00 3.00  4.0 1.00  2.0  3.0  4.0 1.00  2.00  3.00   4.0  1.00   2.0   3.0      4
## y_vals   1.00 1.00 1.00  1.0 2.00  2.0  2.0  2.0 3.00  3.00  3.00   3.0  4.00   4.0   4.0      4
```

```r
position <- min(which( u < cum_prob))
position
```

```
## [1] 15
```

```r
x_vals[position] # value to save for X
```

```
## [1] 3
```

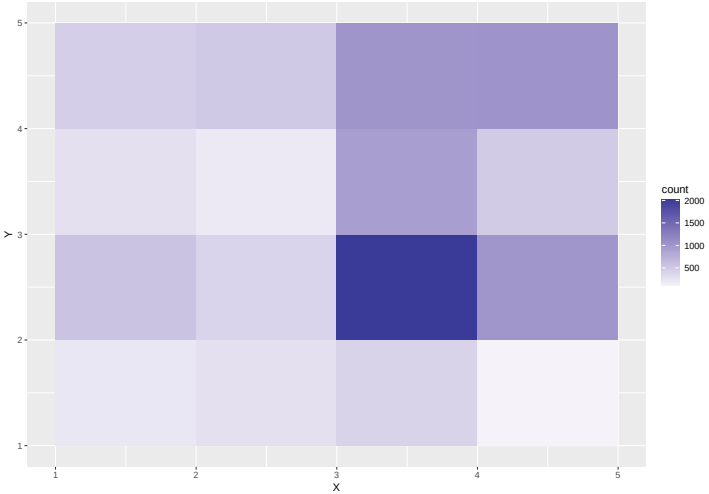
```r
y_vals[position] # value to save for Y
```

```
## [1] 4
```

Code to generate many values

```
set.seed(1)
n <- 10 ^ 4
U <- matrix(runif(n)) # a matrix of U

# a function to find the position based on u
position_func <- function(u){
  min(which( u < cum_prob))
}

positions <- apply(U, 1, position_func)

X <- x_vals[positions]
Y <- y_vals[positions]

df <- data.frame(X, Y, U)
```

```r
library(ggplot2)
# for some reason binwidth = 1 produced only 3 bins
d <- ggplot(df, aes(X, Y)) + geom_bin2d(binwidth = 0.999) + scale_fill_gradient2()
d
```



# Bivariate Normal Distribution

See Professor Tsiang's notes