Q1 - Generating Random Variates

November 21, 2021

1 Q1 - Generating Random Variates

Arman Rezaei - 9723034

1.1 Part 1: The Cauchy Distribution

We want to generate 5 samples from $C(0,1)$ with pdf

$$f(X = x) = \frac{1}{\pi}\frac{1}{1+x^2}$$

We will use the inverse transform method. The CDF is:

$$F(X = x) = \frac{\arctan(x)}{\pi}$$

Please note that the code to calculate $F(X) = \int f(x)\,dx$ using **SymPy** is written below:

```
# python code
from sympy import *
init_session()

f = 1/pi * 1/(1+x**2)
F = integrate(f, x)
#############
```

[1]:
```
n <- 5
uniform_samples <- runif(n)
cauchy_samples <- atan(uniform_samples) / pi

print(cauchy_samples)
```

```
[1] 0.15596752 0.03390587 0.23876154 0.21155400 0.07074715
```

1.2 Part 2: The Beta Distribution

We will be using the inverse transform method, again.

$$\text{Beta}(3,4) = \frac{\Gamma(3+4)}{\Gamma(3)\Gamma(4)}x^{3-1}(1-x)^{4-1} = 60x^2(1-x)^3$$

Therefore the CDF is

$$F(X = x) = \int f(x)\,dx = -10x^6 + 36x^5 - 45x^4 + 20x^3$$

Again, we have used SymPy to calculate the CDF:

```
# python code
from sympy import *
init_session()

f =  60 * x**2 * (1-x)**3
F = integrate(f, x)
#############
```

[2]:
```
n <- 5
uniform_samples <- runif(n)
beta_samples <- -10*uniform_samples^6 + 36*uniform_samples^5 -␣
↪45*uniform_samples^4 + 20*uniform_samples^3

print(beta_samples)
```

```
[1] 0.320471181 0.007018565 0.998772726 0.496930811 0.966521687
```

1.3 Part 3: The Chi-squared Distribution

There is a very clean method to generate Chi-squared samples from the Normal distribution called *Convolution*.

Suppose $Z_1, \ldots, Z_\nu$ are iid $N(0,1)$ random variables, then $V = Z_1^2 + \cdots + Z_\nu^2$ has the $\chi^2(\nu)$ distribution.

Algorithm

1. Fill an $n \times \nu$ matrix with $n\nu$ random $N(0,1)$ variates.
2. Square each entry in the matrix.
3. Compute the row sums of the squared normals. Each row sum is one random observation from the $\chi^2(\nu)$ distribution.
4. Deliver the vector of row sums.

[3]:
```
n <- 5
nu <- 4
X <- matrix(rnorm(n*nu), n, nu)^2 # matrix of sq. normals

# sum the squared normals across each row
y <- rowSums(X)

print(y)
```

```
[1] 2.321693 3.397430 3.789409 1.013914 2.833370
```

1.4 Part 4: The Binomial Distribution

The inverse transform may be a bit tricky here. Here is the algorithm.

Algorithm

1. Generate n iid rvs $U_1, U_2, \ldots, U_n \sim \text{Unif}(0, 1)$.
2. For each $1 \leq i \leq n$, set $Y_i = 0$ if $U_i \leq 1-p$; $Y_i = 1$ if $U_i > 1-p$. (This yields n iid Bernoulli(p) rvs.)
3. Set $X = \sum_{i=1}^{n} Y_i$

[4]:
```
# binomial parameters
n_param <- 4
p <- 0.6

# function to generate binom according to algorithm
generate_binom <- function(n_param, p) {
    unif_samples <- runif(n_param)
    y <- rep(0, n_param)

    for (i in 1:n_param) {
        if (unif_samples[i] <= 1 - p)
            y[i] = 0
        else
            y[i] = 1
    }

    return(sum(y))
}

# number of samples to generate
n <- 5
binom_samples <- rep(0, n)

for (i in 1:n) {
    binom_samples[i] <- generate_binom(n_param, p)
}

print(binom_samples)
```

```
[1] 2 3 2 3 2
```