Random Variable Generation

Michael Rose

November 18, 2018

Reader's Guide

In this chapter, practical techniques that produce random variables from both standard and nonstandard distributions are shown.

Given the availability of a uniform generator in R, we do not deal with the specific production of uniform random variables. The most basic techniques relate the distribution to be simulated to a uniform variate by a transform or a particular probabilistic property.

2.1 | Introduction

The methods in this book summarized under the denomination of *Monte Carlo Methods* mostly rely on the possibility of producting a supposedly endless flow of random variables for well-known or new distributions. Such a simulation is, in turn, based on the production of uniform random variables on the interval (0, 1). In a sense, the uniform distribution $\sim \mathrm{U}(0,1)$ provides the basic probabilistic representation of randomness on a computer and the generators for all other distributions require a sequence of uniform variables to be simulated.

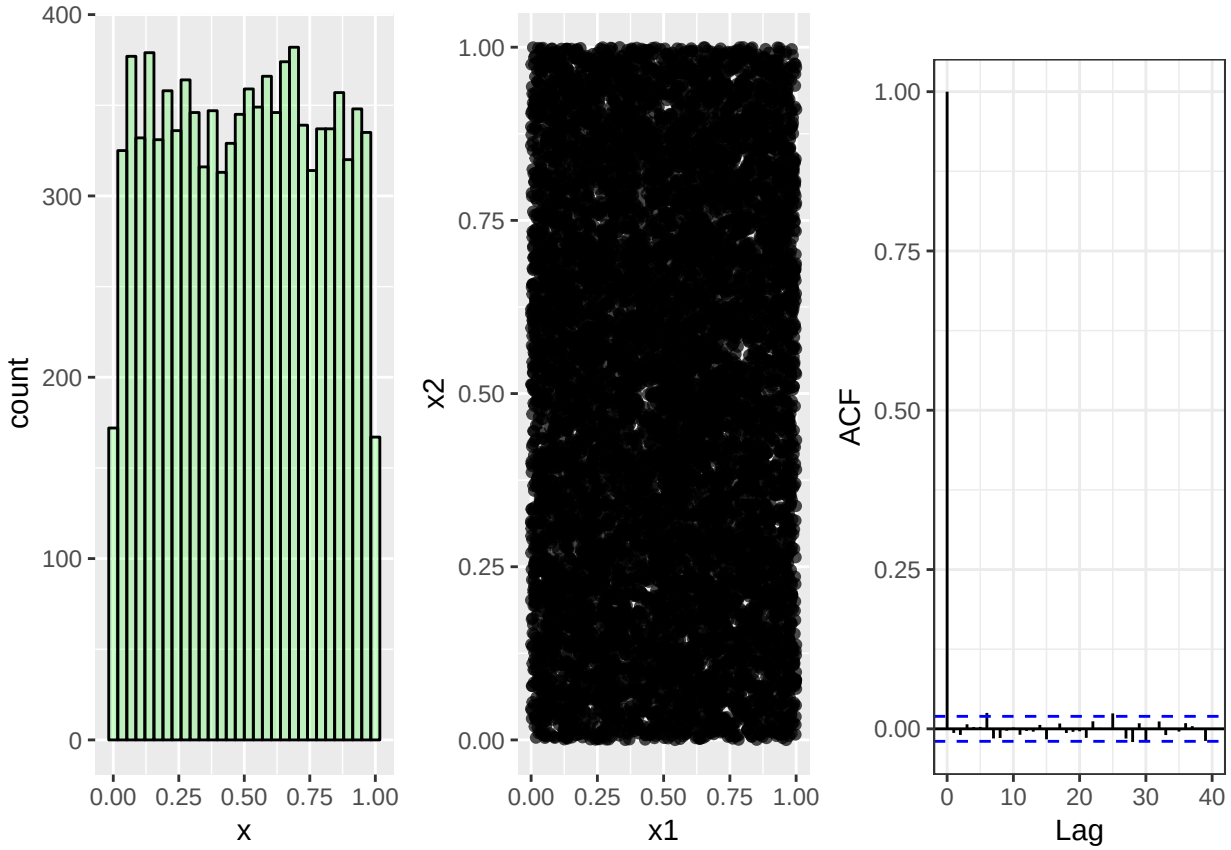
2.1.1 | Uniform Simulation

```
# helper function to make ACF plots in ggplot
ggacf <- function(x, ci=0.95, type="correlation", xlab="Lag", ylab=NULL,
                  ylim=NULL, main=NULL, ci.col="blue", lag.max=NULL) {

    x <- as.data.frame(x)

    x.acf <- acf(x, plot=F, lag.max=lag.max, type=type)

    ci.line <- qnorm((1 - ci) / 2) / sqrt(x.acf$n.used)

    d.acf <- data.frame(lag=x.acf$lag, acf=x.acf$acf)

    g <- ggplot(d.acf, aes(x=lag, y=acf)) +
        geom_hline(yintercept=0) +
        geom_segment(aes(xend=lag, yend=0)) +
        geom_hline(yintercept=ci.line, color=ci.col, linetype="dashed") +
        geom_hline(yintercept=-ci.line, color=ci.col, linetype="dashed") +
        theme_bw() +
        xlab("Lag") +
        ggtitle(ifelse(is.null(main), "", main)) +
        if (is.null(ylab))
            ylab(ifelse(type=="partial", "PACF", "ACF"))
        else
            ylab(ylab)
```

```
    g
}
```

```
# generator 100 random uniform samples
hunnit <- runif(100, min = 2, max = 5)

# check properties of uniform generator

# create vector of randomly generated uniform vars
nsim <- 10^4
x <- runif(nsim)

# vectors to plot
x_1 <- x[-nsim]

# adjacent pairs
x_2 <- x[-1]

# place into dataframes
x <- tibble(x)
x_n <- tibble(x_1, x_2)

# plot
x_hist <- ggplot(x, aes(x)) + geom_histogram(fill = "green", color = "black", alpha = 0.2)
x_plot <- ggplot(x_n, aes(x_1, x_2)) + geom_point(color = "black", alpha = 0.7) + xlab("x1") + ylab("x2
x_acf <- ggacf(x)
gridExtra::grid.arrange(x_hist, x_plot, x_acf, ncol = 3)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



From the simulation above, we see that runif is acceptable.

2.1.2 | The Inverse Transform

There is a simple, useful transformation, known as the probability integral transform which allows us to transform any random variable into a uniform variable and vice versa.

For example, if X has density f and cdf F, then we have the relation

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

and if we set $U = F(X)$, then U is a random variable distributed from a uniform $U(0, 1)$. This is because

$$P(U \leq u) = P[F(X) \leq F(x)] = P[F^{-1}(F(X)) \leq F^{-1}(F(x))] = P(X \leq x)$$

where we have assumed that F has an inverse.

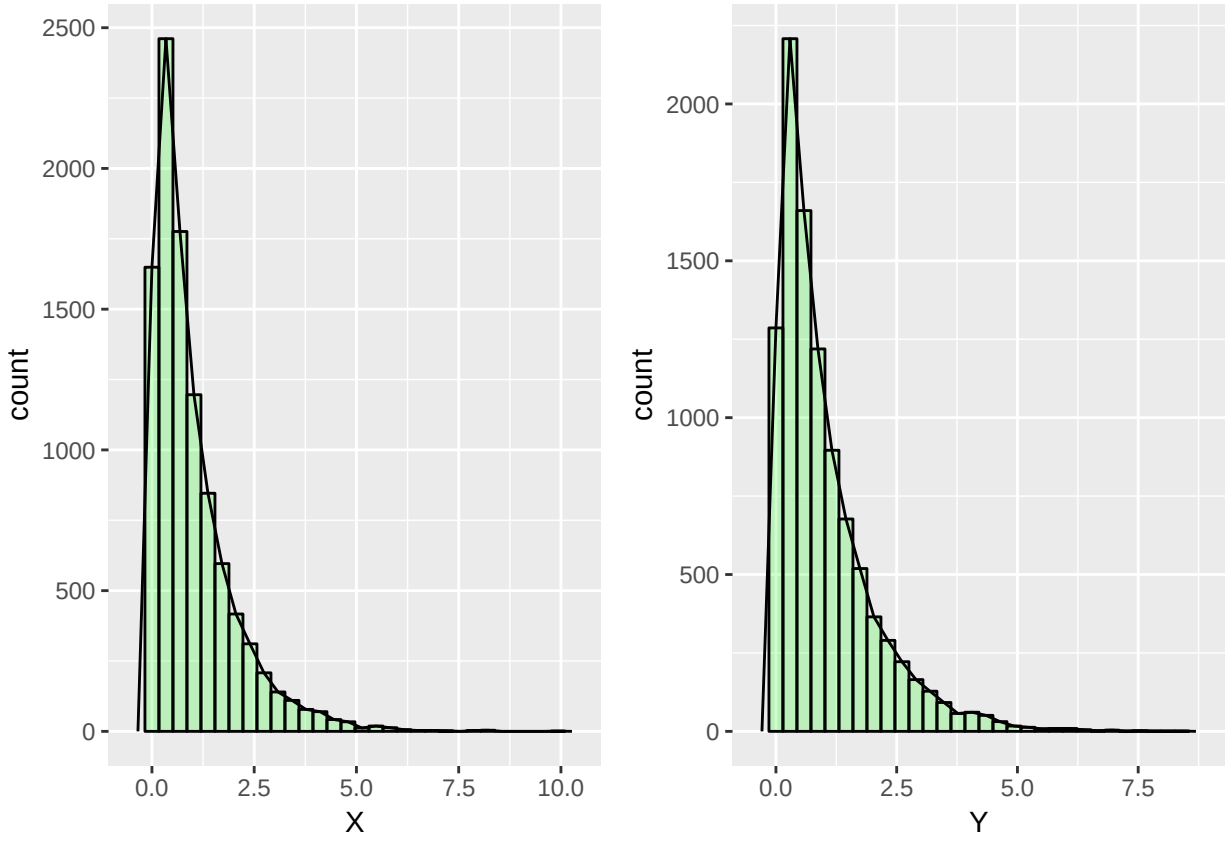
Example 2.1

If $X \sim \text{Exp}(1)$, then $F(x) = 1 - e^{-x}$. Solving for x in $u = 1 - e^{-x}$ gives us $x = -\log(1 - u)$. Therefore, if $U \sim U(0, 1)$, then $X = -\log U \sim \text{Exp}(1)$ as U and $1 - U$ are both uniform.

```
# num random variables
nsim <- 10^4

U <- runif(nsim)

# transformation of uniforms
```

```r
X <- -log(U)

# exponentials
Y <- rexp(nsim)

# transform to data frames
X <- tibble(X)
Y <- tibble(Y)

# plot
unif_hist <- ggplot(X, aes(X)) + geom_histogram(fill = "green", color = "black", alpha = 0.2) + geom_fr
exp_hist <- ggplot(Y, aes(Y)) + geom_histogram(fill = "green", color = "black", alpha = 0.2) + geom_fre
gridExtra::grid.arrange(unif_hist, exp_hist, ncol = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The plot above compares the outpit from the probability inverse transform with the output from rexp. The fits of both histograms to their exponential limit are not distinguishable.

2.2 | General Transformation Methods

When a distribution with density f is linked in a relatively simple way to another distribution that is easy to simulate, this relationship can often be exploted to construct an algorithm to simulate variables from f.

Example 2.2

In example 2.1, we saw how to generate an exponential random variable starting from a uniform. Now we illustrate some of the RVs that can be generated starting from an exponential distribution.

For $X_i \sim \text{Exp}(1)$,

$$Y = 2\sum_{j=1}^{v} X_j \sim \chi^2_{2v},\ v \in \mathbb{N}^*$$

$$Y = \beta \sum j = 1^a X_j \sim G(a, \beta),\ a \in \mathbb{N}^*$$

$$Y = \frac{\sum_{j=1}^{a} X_j}{\sum_{j=1}^{a+b} X_j} \sim \beta(a, b),\ a, b \in \mathbb{N}^*$$

where $\mathbb{N}^* = \{1, 2, ...\}$.

```r
# generate chi sq 6 dof
U <- runif(3*10^4)

# matrix for sums
U <- matrix(data = U, nrow = 3)

# uniform to exponential
X <- -log(U)

# sum up to get chi squares
X <- 2 * apply(X, 2, sum)
```

2.2.1 | A Normal Generator