Introduction to Simulation for Biologists: Probability Distributions

Jun Ishigohoka

2024-06-11

Contents

Summary

- Binomial distribution: discrete number of successes
  - n: number of trials (`size` in `rbinom()`)
  - p: prob. of success
  - `rbinom(1, size = n, prob = p)` for discrete random variable k successes $k \sim B(n, p)$
- Geometric distribution: discrete number of failures before the first success
  - p: prob. of success
  - `rgeom(1, prob = p)` for discrete random variable k trials $k \sim G(p)$
- Poisson distribution: discrete number of events
  - λ: rate of events per unit of continuous time/space.
  - `rpois(1, lambda = lambda)` for discrete random variable k events $k \sim Pois(\lambda)$
- Exponential distribution: continuous waiting time until the first event
  - λ: rate of events per unit of continuous time/space.
  - `rexp(1, rate = lambda)` for continuous random variable x units $x \sim Exp(\lambda)$

Binomial distribution

Throw a die once. The outcome is success if the side is 1, failure otherwise. This type of trial where the outcome is either success or failure with a certain probability is called a **Bernoulli trial**. The distribution of the number of successes x out of n independent Bernoulli trials ($0 \leq x \leq n$) is the **binomial distribution**.

In `R`, we can use the `rbinom` function for binomial sampling. Confusingly, in R, the number of trials is `size` and the number of binomial samplings is `n`.

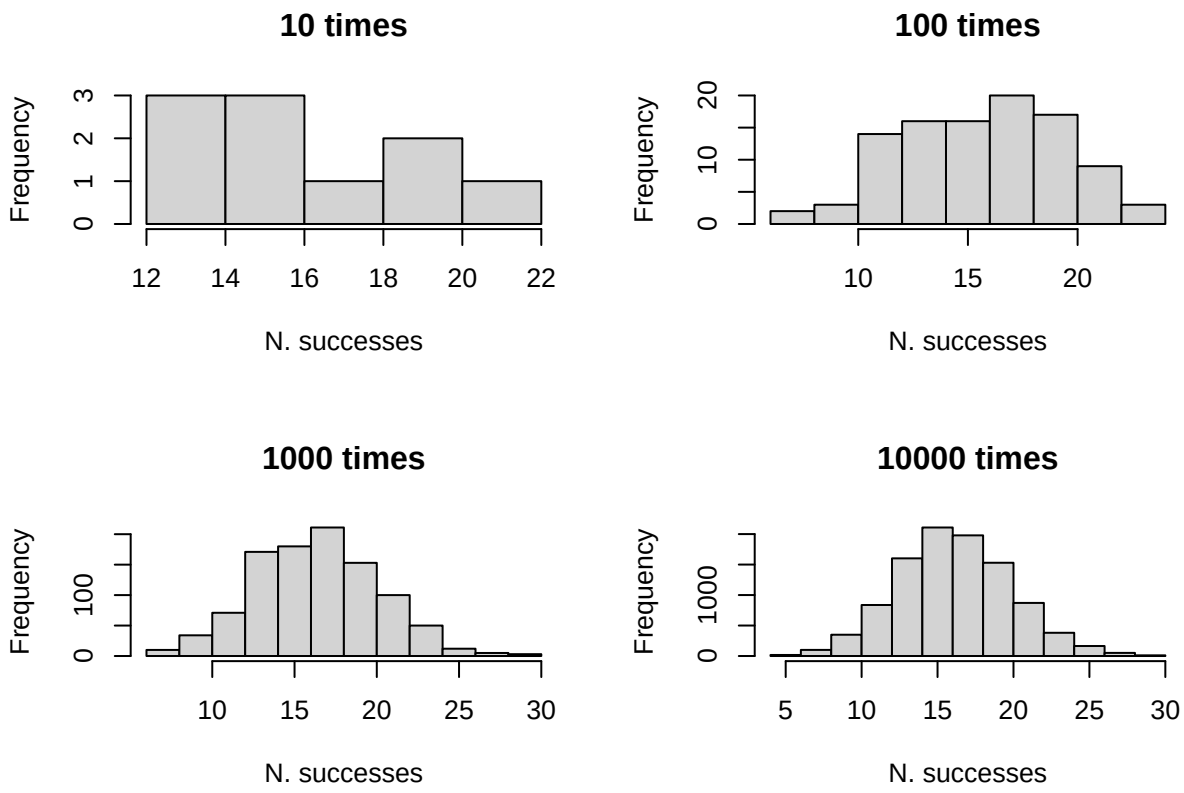
Let's throw a die 100 times and count the number of successes (e.g. the side of 1).

```
rbinom(n = 1, size = 100, prob = 1/6) # B(100, 1/6)
```

```
## [1] 18
```

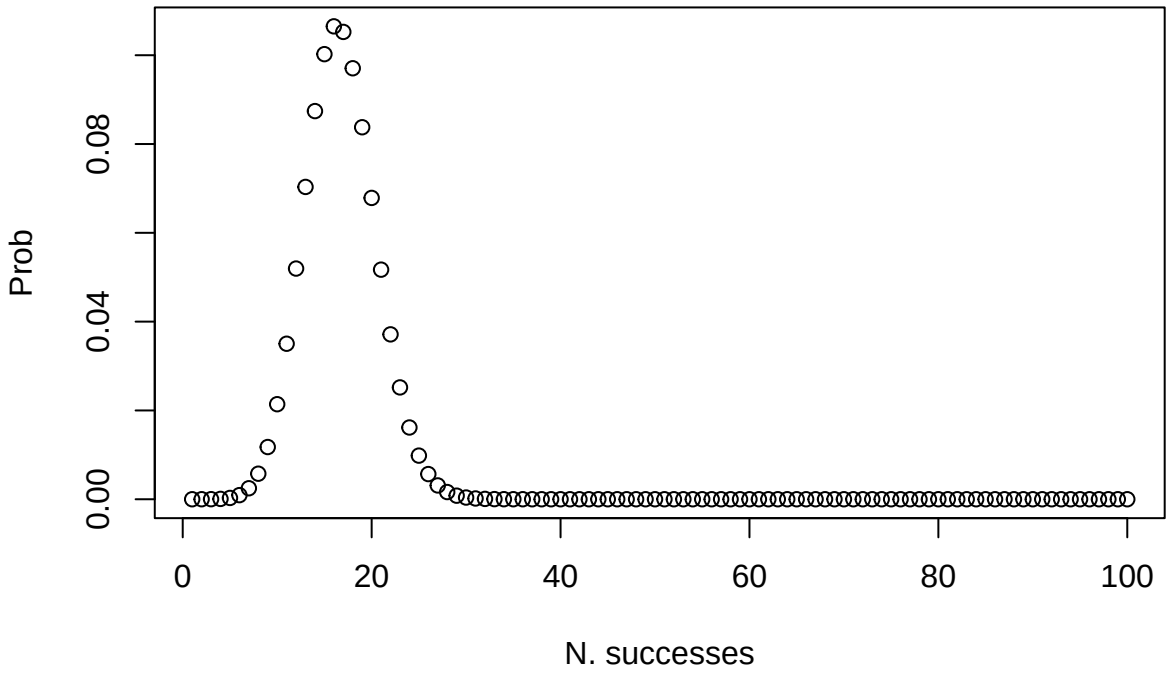
Let's repeat this 10, 100, 1,000, 10,000 times to see the distribution of the number of successes.

```
par(mfrow=c(2,2))
for(n in c(10, 100, 1000, 10000)){
        hist(rbinom(n = n, size = 100, prob = 1/6),
             main = paste0(n, " times"),
             xlab = "N. successes"
        )
}
```



This distribution will approach the probability mass function of the binomial distribution.

```r
x <- 1:100
plot(x, dbinom(x = x, size = 100, prob = 1/6),
     main = "B(100, 1/6)",
     ylab = "Prob",
     xlab = "N. successes"
)
```



Geometric distribution

Let's keep throwing a die until the side is 1. How many times does it take?

Naively, we can loop binomial sampling until it succeeds.

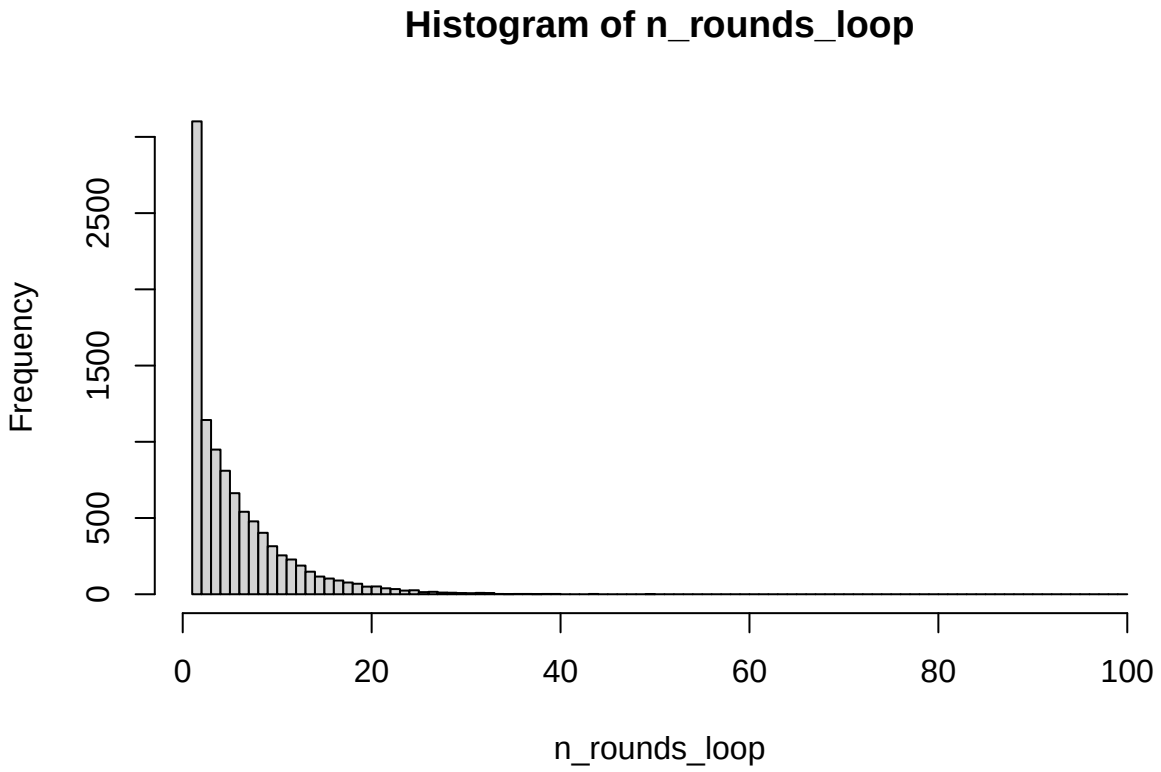
```
get_nrounds <- function(prob){
        c <- 0
        n <- 0
        while(n == 0){
                c <- c + 1
                n <- rbinom(n = 1, size = 1, prob = prob)
        }
        return(c)
}


get_nrounds(prob = 1/6)
```

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

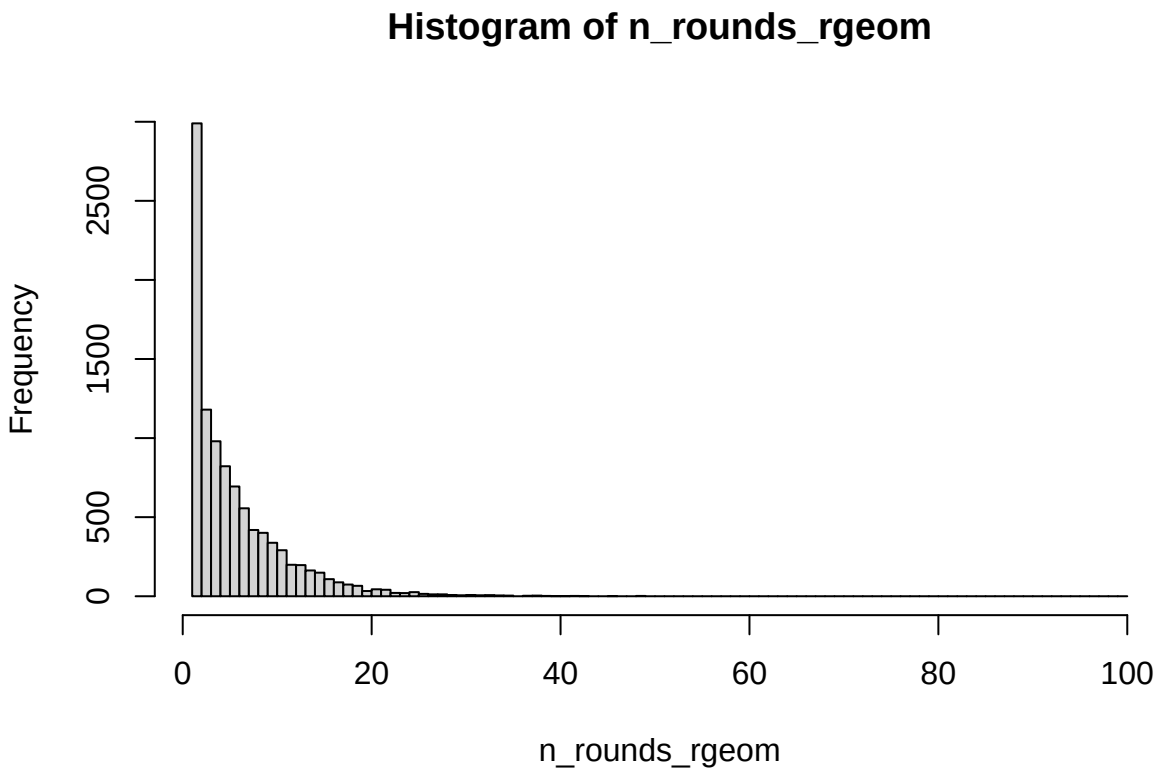
Let's visualise how this waiting time is distributed

```
n_rounds_loop <- sapply(1:10000,
       function(x){
               get_nrounds(prob = 1/6)
       }
)
hist(n_rounds_loop, breaks = seq(1,100))
```



The number of failures before the first success is known to follow a **geometric distribution**. So, instead of looping binomial/Bernoulli trials, we can directly obtain the number of throws from a geometric distribution using `rgeom` function.

```
n_rounds_rgeom <- rgeom(n = 10000, prob = 1/6) + 1 # + 1 for the first success
hist(n_rounds_rgeom, breaks = seq(1,100))
```



Using a geometric sampling instead of looping is more efficient especially when the probability of success is very low. Let's measure the time it takes to obtain the number of throws until the first success 100 times using the loop.

```
system.time(
            sapply(1:10,
                   function(x){
                           get_nrounds(prob = 1e-6)
                   }
            )
)
```

```
##    user  system elapsed
##   3.009   0.007   3.017
```

And if we use geometric sampling

```
system.time(
        rgeom(n = 10000, prob = 1/6) + 1 # + 1 for the first success
)
```

```
##    user  system elapsed
##   0.001   0.000   0.000
```

This is a good example that you can still implement your simulation without knowing probability distributions, yet knowing distributions makes your simulation faster and more efficient!

Poisson distribution

Consider taking 1 mL from a cell suspension at 5,000,000 cells/L (5,000 cell/mL on average). Instead of 1 time of pipetting of 1 mL, let's pipette n times, $1/n$ mL each. When n is very large, the probability that multiple cells are in the taken $1/n$ mL is so low that it contains either one cell at a probability of $5000/n$

or no cells at a probability of $1 - 5000/n$. By considering of having one cell as success, the total number of cells in the n portions (thus 1 mL) should follow the binomial distribution B(n, p), where $p = 5000/n$. When $n \to \infty$, the binomial distribution approaches a **Poisson** distribution. The Poisson distribution has one parameter, the rate λ, which depicts the number of events per unit of time or space (in this case, volume). In this example, the number of cells in a sampled suspension of 1 mL should follow a Poisson distribution with $\lambda = 5000$.

Let's take a 1 mL of cell suspension from 5,000 cells/mL.

```
rpois(n = 1, lambda = 5000)
```

```
## [1] 5077
```

A nice thing about Poisson distribution is that you can define the unit of time/space as you like as long as the rate parameter is linearly scaled. In the example above, if we take 10 mL instead of 1 mL, the number of sampled cells should now follow another Poisson distribution with $\lambda = 5000 \times 10 = 50,000$.

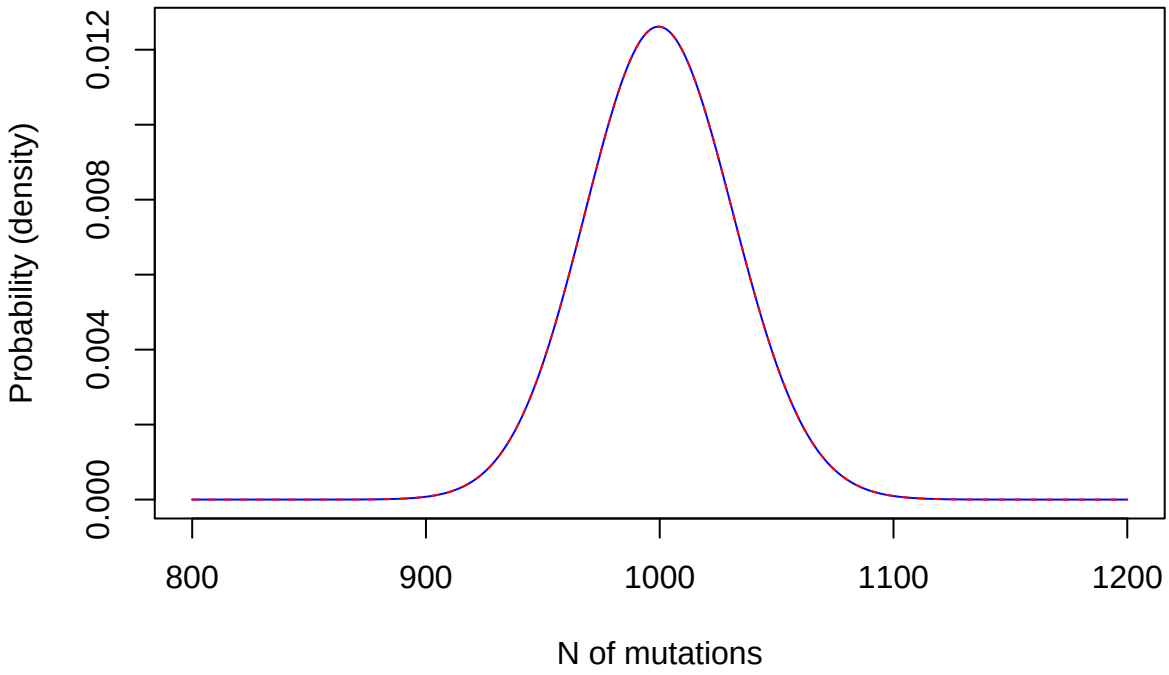
Let's think of the number of mutations (mutation rate of 1×10^{-9} per year per bp) between two homologous sequences of DNA of 1,000,000 bp, where TMRCA is 500,000 years ago. The total branch length between the two sequences is 100,000 years, and mutation occurs at a rate of $1,000,000[\text{years}] \times 1,000,000[\text{bp}] \times 10^{-9}[\text{year}^{-1}\text{bp}^{-1}] = 1,000$. So the number of mutations follows a Poisson distribution with $\lambda = 1,000$.

```
rpois(n = 1, lambda = 1000)
```

```
## [1] 980
```

We could think of the same problem with binomial sampling, where mutation can occur at a very low probability (10^{-9} per year per bp) with many trials (1,000,000 [bp] $\times$ 1,000,000 [years] = 1,000,000,000,000 trials) We can see how close both ways are:

```
plot(800:1200, dpois(800:1200, lambda = 1000),
     type = 'l',
     col = "blue",
     xlab = "N of mutations",
     ylab = "Probability (density)"
)
lines(800:1200, dbinom(800:1200, 1e12, 1e-9), type = 'l', col = "red", lty = 2)
```



Exponential distribution

Continuing on the example of taking cells, let's keep sucking the solution until we get the first cell. The random amount of suspension to take until we get the first cell follows a continuous distribution called **exponential distribution**, which is the limit of geometric distribution where $\lambda = np$ and $n \to \infty$. As is the Poisson distribution, the exponential distribution has one parameter, the rate λ.

Let's sample the amount of suspension at 5,000 cells/mL until you get the first cell. Confusingly, `lambda` in `rgeom()` is the same rate parameter as `rate` in `rexp()`.

```
rexp(1, 5000)
```

```
## [1] 0.0003215086
```

According to Wikipedia, the lifetime probability of a fatal lightning strike is 1/60000. So the time it takes for you to die of a lightning hit can be modelled with an exponential distribution. Let's sample one such event.

```
print(paste(rexp(1, 1/60000) * 70, "years"))
```

```
## [1] "2802819.4011189 years"
```

Other distributions used in the course

Hypergeometric distribution

The hypergeometric distribution describes the probability of k successes (random draws for which the object drawn has a specified feature) in n draws, **without replacement**, from a finite population of size N that contains exactly K objects with that feature, wherein each draw is either a success or a failure. This is similar to binomial distribution, but it is sampling without replacement.

The hypergeometric distribution is often seen in enrichment analysis. For example, in the context of gene ontology analysis, N is the number of all genes, K is the number of all genes of a focal gene ontology term, n is the number of significant genes, k is the number of significant genes of a focal gene ontology.

Let's sample a number of genes out of 100 significant genes that are of one GO term, when there are 2000 genes of this GO term in 10000 genes throughout the genome.

```
rhyper(1,
       m = 2000, # number of genes of a GO type in the genome
       n = 10000 - 2000, # number of genes that are not the GO type in the genome
       k = 100 # number of significant genes
)
```

```
## [1] 20
```

Let's compute the p-value of observing 50 out of 100 significant genes that are of this GO term.

```
phyper(50,  # number of significant genes of the GO type
       m = 2000, # number of genes of the GO type in the genome
       n = 10000 - 2000, # number of genes that are not the GO type in the genome
       k = 100,  # number of significant genes
       lower.tail = F
)
```

```
## [1] 3.839329e-12
```

Normal distribution

According to the central limit theorem, the average of many observations of some random variable is a random variable, whose distribution converges to a normal distribution as the number of samples increases. Quantities that are expected to be the sum of many independent processes, such as quantitative genetic traits and errors, can be modelled to follow a normal distribution.

Let's sample a normal random variable with mean of 175 and variance of standard deviation of 6.

```
rnorm(1,
      mean = 175,
      sd = 6
)
```

```
## [1] 175.2156
```