Simulation13

Raphael Carvalho

04/06/2019

Simulation

Let's simulate rolling four six-sided dice: sample(1_6, 4, replace = TRUE)

```
sample(1:6, 4, replace = TRUE)
```

```
## [1] 5 5 5 1
```

Now sample 10 numbers between 1 and 20, WITHOUT replacement. To sample without replacement, simply leave off the 'replace' argument.

```
sample(1:20, 10)
```

```
##  [1] 13  2 14  1 17 19  7  3 20 16
```

The sample() function can also be used to permute, or rearrange, the elements of a vector. For example, try sample(LETTERS) to permute all 26 letters of the English alphabet.

```
sample(LETTERS)
```

```
##  [1] "C" "E" "W" "V" "N" "G" "D" "R" "Y" "U" "O" "M" "B" "K" "F" "X" "J"
## [18] "L" "T" "Z" "A" "H" "I" "P" "Q" "S"
```

Now, suppose we want to simulate 100 flips of an unfair two-sided coin. This particular coin has a 0.3 probability of landing 'tails' and a 0.7 probability of landing 'heads'. Let the value 0 represent tails and the value 1 represent heads. Use sample() to draw a sample of size 100 from the vector c(0,1), with replacement. Since the coin is unfair, we must attach specific probabilities to the values 0 (tails) and 1 (heads) with a fourth argument, prob = c(0.3, 0.7). Assign the result to a new variable called flips.

```
flips <- sample(c(0, 1), 100, prob = c(0.3, 0.7), replace = TRUE)
flips
```

```
##   [1] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1
##  [36] 1 1 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 1 0
##  [71] 1 0 1 1 1 1 1 1 1 0 1 1 0 0 1 0 1 0 0 1 1 1 1 0 1 1 1 1 1 1
```

Since we set the probability of landing heads on any given flip to be 0.7, we'd expect approximately 70 of our coin flips to have the value 1. Count the actual number of 1s contained in flips using the sum() function.

```
sum(flips)
```

```
## [1] 73
```

A coin flip is a binary outcome (0 or 1) and we are performing 100 independent trials (coin flips), so we can use rbinom() to simulate a binomial random variable.

Each probability distribution in R has an r*** function (for "random"), a d*** function (for "density"), a p*** (for "probability"), and q*** (for "quantile").

A binomial random variable represents the number of 'successes' (heads) in a given number of independent 'trials' (coin flips). Therefore, we can generate a single random variable that represents the number of heads in 100 flips of our unfair coin using rbinom(1, size = 100, prob = 0.7). Note that you only specify the probability of 'success' (heads) and NOT the probability of 'failure' (tails). Try it now.

```
rbinom(1, size = 100, prob = 0.7)
```

```
## [1] 68
```

Equivalently, if we want to see all of the 0s and 1s, we can request 100 observations, each of size 1, with success probability of 0.7. Give it a try, assigning the result to a new variable called flips2.

```
flips2 <- rbinom(100, size = 1, prob = 0.7)
flips2
```

```
##   [1] 0 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 0
##  [71] 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1
```

Now use sum() to count the number of 1s (heads) in flips2. It should be close to 70!

```
sum(flips2)
```

```
## [1] 73
```

The standard normal distribution has mean 0 and standard deviation 1. As you can see under the 'Usage' section in the documentation, the default values for the 'mean' and 'sd' arguments to rnorm() are 0 and 1, respectively. Thus, rnorm(10) will generate 10 random numbers from a standard normal distribution. Give it a try.

```
rnorm(10)
```

```
##  [1] -0.20016961 -0.20944667  1.93458812 -0.61987998 -0.86334965
##  [6] -0.57722933 -0.09456287  0.26263289  0.75566201 -0.86398396
```

Now do the same, except with a mean of 100 and a standard deviation of 25.

```
rnorm(10, 100, 25)
```

```
##  [1] 111.20317 159.74736  91.89851 141.95823 112.30899  80.88118  65.47176
##  [8]  88.21803  53.37742  61.08285
```

Finally, what if we want to simulate 100 *groups* of random numbers, each containing 5 values generated from a Poisson distribution with mean 10? Let's start with one group of 5 numbers, then I'll show you how to repeat the operation 100 times in a convenient and compact way.

Generate 5 random values from a Poisson distribution with mean 10.

```
rpois(5, lambda = 10)
```

```
## [1] 11  8 15 12 10
```

Now use replicate(100, rpois(5, 10)) to perform this operation 100 times. Store the result in a new variable called my_pois.

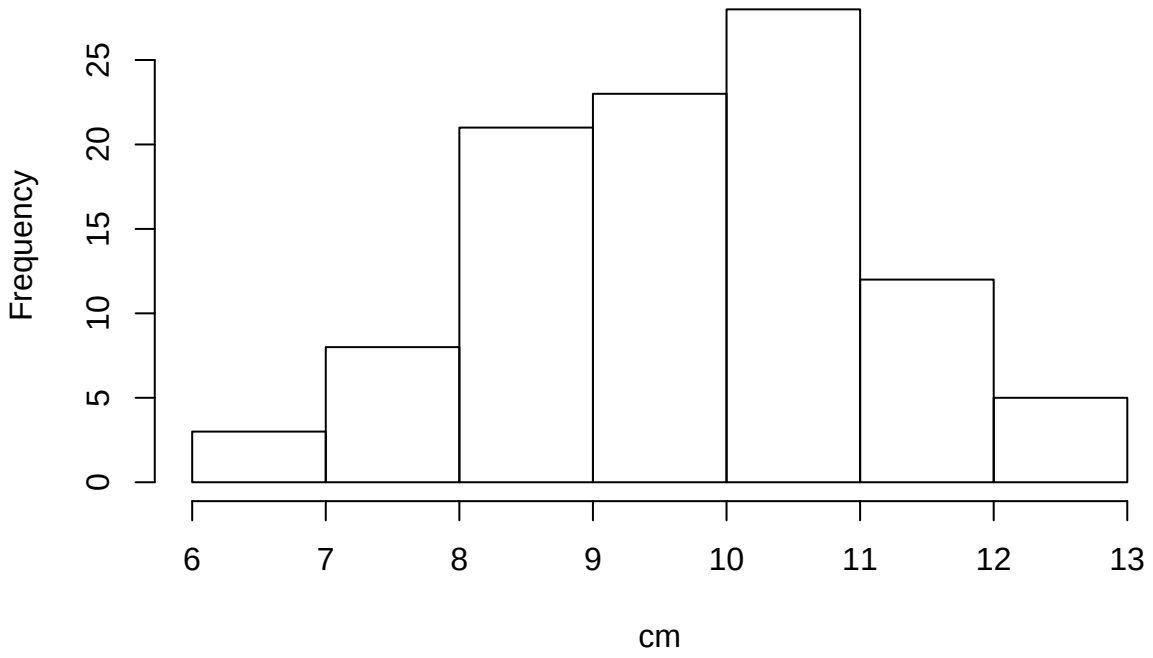
```
my_pois <- replicate(100, rpois(5, 10))
my_pois
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]   10    8    6   12   12    7   10   12   12     9     6     9    12
## [2,]   11    8   14   10    7    4    9   11    7     6    13    10    11
## [3,]    8   10   14    8    8   11   10    8   12    11    11    11    11
## [4,]   10   16   11   15    9    6    7   16    6     5    12    20     9
## [5,]    9   12    9   10    8    8    8   15   10     4    11     9    10
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]    11    11    11    17     7     5     4     3    17     8     7
## [2,]    11     8     8     8     8     6     7    18     7     6    13
```

```
## [3,]     7    17     7    11     6    13    14     7    12    10    11
## [4,]    11     4    16    12    12     8     4    17     8    17     4
## [5,]     9    13     4     7     9     6    10    12    11    12     9
##      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
## [1,]    16     9    11    13     8     9     7    11     9    11    18
## [2,]    10     9    10     7     9    16    12    10    12    12    11
## [3,]     9     8    11    15    11    10    12     9    12     4    12
## [4,]     9     9     6     9    12     5    10    10    13     7     6
## [5,]    13     9     7     7     9    12    11     9    13     9    13
##      [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
## [1,]    14    10    11     9    17     8     7    10     8     9     8
## [2,]     8     7    12     7     8     8     4    11    10     2    13
## [3,]     8     9    10    10    16    15     8     6    15    11     9
## [4,]     8     3     9     7    10     7    10     7    12    12     6
## [5,]    13    13    11     8     4    11    15    12    10     4     3
##      [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57]
## [1,]     8    13    15     7    14    11     5     4    17     9    10
## [2,]    11     6     6    16     8     8     9    15    14    13     9
## [3,]     9    10    10    15     9     7     8    13    10     5     9
## [4,]     7    12    12    11     8    10    16     8    10     6     9
## [5,]    15     7     8    12     9     6     7    13    12    15     7
##      [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68]
## [1,]     7    13    14     8    10    15     8     9    11    13    11
## [2,]     6    12    11    17     8     9     9     8    15    13     9
## [3,]     6    12     6     9    12     7     9    12    11     9     6
## [4,]    14    10     9     9    10    15     7    10     7     3     3
## [5,]    10    13     8     9    11     7    14     3    12     6    13
##      [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78] [,79]
## [1,]    12     9     8    14    12    10    12    14     6    13     6
## [2,]     8    10    10     9     6     9     9     8     7    10    15
## [3,]    10     8     8     6    10     5     9     3     8     6    14
## [4,]     5    12    12     8    12    11     7     8     4     6     7
## [5,]    14    16     9    13    10     7     9     9     9    21    14
##      [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90]
## [1,]    10    14     8    21    13    14    14     5    10     9     8
## [2,]    14     9    11    10     8     5    12     8    16     6    10
## [3,]     6    11     9    14     6    13     8     7    12    10     5
## [4,]     8    10    11    10    10     8     9    10    11     8     5
## [5,]    10     9    15    10    10    18    14     9     5     9     6
##      [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
## [1,]     7    11     9     7     9    13     9     8     5      9
## [2,]    16    10     6    12    12    13     7     7     9     11
## [3,]     9     7     7    16    13     7     7    13    11      7
## [4,]    18    10    13    12    11    19     5     8    12     10
## [5,]    10     9    10     7     9    11    11     2    16     13
```

replicate() created a matrix, each column of which contains 5 random numbers generated from a Poisson distribution with mean 10. Now we can find the mean of each column in my_pois using the colMeans() function. Store the result in a variable called cm.

```
cm <- colMeans(my_pois)
hist(cm)
```




```
my_pois <- replicate(100, rpois(5, 10))
my_pois
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]   11    6   11   16   14   10   14   10    8    15    12    15    12
## [2,]   10   10   14    7   15   10   16   11    8     7     7    12    10
## [3,]    8    4    6   11    8   15   13   10   11     6    17     9     9
## [4,]    8   16    8    8   14    8   25    8    9    11    10    11     7
## [5,]   17    7   12    6    9   10   10    6    8    10     9     7     7
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]    15    14    10     9    11    16    12     8     2    12    10
## [2,]     6    15     7     5    10    11     7     7     5    13    14
## [3,]    13    14    10     9    11    11     9    15    12    12    13
## [4,]    10    10     9     8    13    11    10    11    10    11     9
## [5,]    14    10    17    10     5     8    10    14    12     9    13
##      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
## [1,]    13    12    10     7     9     7    10     6    15    12    11
## [2,]    12     5     9     7    11    14     7     6     4     4    11
## [3,]     8     9     7     5     6     8    14    11     6    13    12
## [4,]     9    11    14    12    10     6    13     7     7     5    10
## [5,]    10     9    13    13    10     6     8    17     8    12     8
##      [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
## [1,]     7     9     6    14     4    18    13     7    12    10    12
## [2,]     8    10     8    17    15    13    10    16    15    16    10
## [3,]    12    11    13     5    12    14    10     7    10     7    13
## [4,]    11    12     6    12    10     8    14    13     9    13    14
## [5,]    10     8     6     7    12     9    14    13    12     9     8
##      [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57]
## [1,]     7    10    13     8     9     6     8     9    10    10     7
## [2,]    10     7     6    12    12    12    16     9    15     8     7
```

```
## [3,]    10    10     7    10     9    11     9    12    11     6    10
## [4,]     8     6     9     5     7     7    14    15    10    13     9
## [5,]    10     8    18     8    11     9     5    12    12    12    16
##      [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68]
## [1,]    11     6    13    10     6    14    10     8     9    15    10
## [2,]    11     8    15     9     8     5     6    12    12    18    10
## [3,]     5     7     8    13     7    11     8     8     9    10    13
## [4,]    15     6     7     3    10    10     9    15    14    12     3
## [5,]     8     9     7    11    11    14    10     8    11    13    14
##      [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78] [,79]
## [1,]    10     5     7     4     7     7     7    11     9     5     8
## [2,]     8    15    13    13     9    12    10     7    11    16     9
## [3,]    13     8     8    13    11     4    10    17    11     8     8
## [4,]    13     5    12    10    16     6    13    11     6     7    14
## [5,]    14    19    14    13     8    11     6     9     9    12    13
##      [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90]
## [1,]     8    12     5    11    17    10     8    13     9     8     7
## [2,]     6    16    10     6    11     7    12    18     6    17    13
## [3,]    11     9    12    12    14    10     8    13    14    10    10
## [4,]    19    10    12    17    11    12    12     7    10    13     8
## [5,]    14     3    11    12     8     8    10     9    10    16     6
##      [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
## [1,]     8    10     9     6    12    11    11     7    16     10
## [2,]    12    16     9    11    17    12     9     4    15     12
## [3,]     8    17    19    10    12    11    14     9    13     15
## [4,]     9    15    10     8    16    14    11     7     8      9
## [5,]    10    10     8     7    11     8     7     8    10      8
```