# Stat 343: Displaying discrete distributions

T.Scofield

## Binomial distributions

### Binomial variable simulation

Let's suppose you draw an iid sample of size $n$ from a binomial categorical variable whose values are, generically, **S**uccess and **F**ailure. Suppose, too, that the probability of a success on any draw is $\pi$, making the probability of failure $1 - \pi$. Taking our random variable $X$ to be the count of successes, we might simulate values of $X$, given a particular choice of $n$ and $\pi$.

For instance, if $n = 10$ and $\pi = 0.3$, `rflip()` can do the job:

```
rflip(10, prob=0.3)     # only available with the mosaic package
```

```
Flipping 10 coins [ Prob(Heads) = 0.3 ] ...

T T T T T T T H T

Number of Heads: 1 [Proportion Heads: 0.1]
```

We might also make a virtual bag with 7 slips containing 0 and 3 slips containing 1, sampling from that bag with replacement. The `sum()` command produces a count, in this case.

```
bag <- c( rep(0,7), rep(1, 3) )
sum(resample(bag))
```

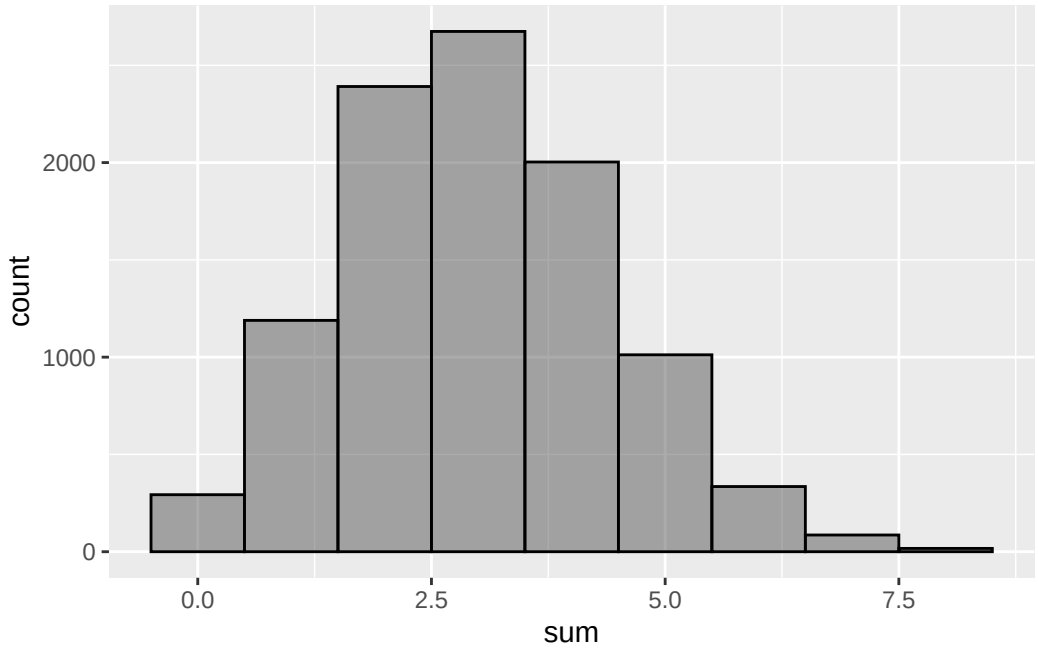
```
[1] 4
```

To simulate the probability distribution, we would need a lot of numbers $X$ produced in this fashion.

```
manyXs <- do(10000) * sum(resample(bag))      # do() also provided by mosaic
head(manyXs)
```

```
  sum
1   2
2   6
3   1
4   5
5   1
6   3
```

```
gf_histogram(~sum, data=manyXs, bins=9, color="black")
```




Not surprisingly, $X = 3$ seems the most-frequently-observed value.

### Binomial distribution tools in R

R has built-in tools for working with binomial distributions. The simulation we did above is performed more efficiently using the `rbinom()` command:

```
rbinom(5, 10, 0.3)          # makes a list, not a data frame; change 5 to 10000 if you like
```

```
[1] 2 3 4 1 2
```

In specifying $n = 10$ and $\pi = 0.3$ there is no need for any bag-building. As many counts of successes as we desire will be simulated. (Above, I ask for 5.)

In class, we developed a formula for the $\Pr(X = x)$:

$$\Pr(X = x) \;=\; \binom{n}{x}\pi^x(1-\pi)^{n-x} \;=\; \frac{n!}{x!(n-x)!}\pi^x(1-\pi)^{n-x}.$$

The `dbinom()` command evaluates this formula:

```
dbinom(5, size=10, prob=.2)      # Pr(X=5) when n=10 and pi=0.2
```

```
[1] 0.02642412
```

```
dbinom(50, size=100, prob=.2)    # Pr(X=50) when n=100 and pi=0.2
```

```
[1] 1.621261e-11
```

```
dbinom(23, 100, .2)              # Pr(X=23) when n=100 and pi=0.2
```

```
[1] 0.07198004
```

To see the full list of values of the **probability mass function** $f(x) = \Pr(X = x)$ when $n = 5$, $\pi = 0.3$:

```
dbinom( 0:5, 5, 0.3 )
```

```
[1] 0.16807 0.36015 0.30870 0.13230 0.02835 0.00243
```

This might be displayed in table format

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $\Pr(X = x)$ | 0.16807 | 0.36015 | 0.30870 | 0.13230 | 0.02835 | 0.00243 |

Below, I will use the `gf_dist()` command (one provided with mosaic) to display the probability mass function as a graph.

The `pbinom()` command evaluates the cdf (cumulative distribution function) for a binomial r.v. $X \sim \text{Binom}(n, \pi)$. That is, for any fixed value $x$, the probability of a count of successes no more than $x$,

$$F(x) \;=\; \Pr(X \le x) \;=\; \sum_{k \le x} f(k) \;=\; \sum_{k \le x} \binom{n}{k} \pi^k (1-\pi)^{n-k}$$

is given by `pnorm(x,`$n, \pi$`)`. For example, when $n = 10$ and $\pi = 0.4$, these two commands produce the same value:

```
pbinom(5, 10, .4)
```

```
[1] 0.8337614
```

```
sum( dbinom(0:5, 10, 0.4) )
```

```
[1] 0.8337614
```

To round out the set, there is the `qnorm()` command which attempts to invert the `pnorm()` command. It can do this better when $n$ is large, as in

```
pbinom(55, 100, 0.6)      # Pr(X <= 55) = 0.1789 when n=100, pi=0.6
```

```
[1] 0.1789016
```

```
qbinom(0.17, 100, 0.6)    # Estimate where to stop so that cdf value is 0.17
```

```
[1] 55
```

```
qbinom(0.18, 100, 0.6)    # Estimate where to stop so that cdf value is 0.18
```

```
[1] 56
```

### Other distributional tools in R

The prefixes remain when the distributions change. For negative binomial distributions, there are 4 built-in commands mirroring those described for binomial distributions: `rnbinom()`, `dnbinom()`, `pnbinom()`, and `qnbinom()`.

For **geometric distributions**, we have commands: `rgeom()`, `dgeom()`, `pgeom()`, and `qgeom()`. For **hypergeometric distributions**, we have commands: `rhyper()`, `dhyper()`, `phyper()`, and `qhyper()`. For **Poisson distributions**, there are commands: `rpois()`, `dpois()`, `ppois()`, and `qpois()`. While several of these will be mentioned for the first time later in Section 2.7, each of these is for distributions of a particular kind of discrete random variable.

### Plotting a distribution

Here, I suggest the mosaic command `gf_dist()`. Three different forms of this command used to plot the pdf (twice) and the cdf (once) are displayed in the gray box on p. 63. The corresponding graphs appear in Figure 2.8, at the top of p. 63.