Binomial Distribution in R

Therese Donovan, Lew Coggins, Jim Hines

7/2/2020

Contents

Introduction

The binomial distribution is widely used for problems where there are a fixed number of tests or trials (n) and when each trial can have only one of two outcomes (e.g., success or failure, live or die, heads or tails).

The formula is written below, and was introduced in depth in our spreadsheet tutorial:

$$f(y|n,p) = \binom{n}{k} p^y (1-p)^{n-y}$$

The number of successes is usually denoted as y, and the probability of success is usually denoted as p. A typical example considers the probability of getting 3 heads, given 10 coin flips and given that the coin is fair ($p = 0.5$). The left side of the binomial probability function is written $f(3|10, 0.5)$, where the vertical bar | means "given" and is read, "the probability of getting 3 heads, given 10 coin flips and the probability of a head (success) is 0.5." Let's break the right hand side of the binomial probability function into pieces. The portion p^y and $(1-p)^{n-y}$ gives p (the probability of success, or heads) raised to the number of times the success occurred (y) and $1-p$ (the probability of a failure, or tails) raised to the number of times the failures occurred.

But if you flip a fair coin 10 times, there are many ways you could end up with three heads. For instance, the first three tosses could be heads and the rest could be tails (HHHTTTTTTT). Or the first seven could be tails and the last three could be heads (TTTTTTTHHH). Or you could alternate getting heads and tails (e.g., THTHTHTTTT). The portion of the binomial probability function in brackets is called the binomial coefficient, and accounts for ALL the possible ways in which three heads and seven tails could be obtained.

Another example considers the probability of 15 sites are occupied by a species of interest out of a 59 total sites, given $p = 0.3$ (the probability of survival). This example assumes that detection probability (the probability of detecting a species that occurs on site, given the site is occupied) is perfect and can be

ignored. The left side of the binomial probability function would be written: $f(15|59, 0.3)$ or more precisely, $f(y = 15|n = 59, p = 0.3)$

The binomial probability can easily be computed in a R. Here, we will focus on four R functions that are associated with the binomial distribution.

The binomial functions in R

The easiest way to search for functions that are new to you is to use the `help.search()` function, which searches by keyword.

```
# search based on keyword "binomial"
help.search("binomial")
```

The `help()` function can also be used. The binomial functions (and many other probability functions in R) are located in the "stats" package. We can search for the topic "Binomial" and specify that the search is limited to the stats package with the following code:

```
help(package = "stats", topic = "Binomial")
```

You should see a page that looks something like this in your R Studio Help tab:

```
## Registered S3 method overwritten by 'printr':
##   method                from
##   knit_print.data.frame rmarkdown
```

```
## The Binomial Distribution
##
## Description:
##
##      Density, distribution function, quantile function and random
##      generation for the binomial distribution with parameters 'size'
##      and 'prob'.
##
##      This is conventionally interpreted as the number of 'successes' in
##      'size' trials.
##
## Usage:
##
##      dbinom(x, size, prob, log = FALSE)
##      pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
##      qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
##      rbinom(n, size, prob)
##
## Arguments:
##
##     x, q: vector of quantiles.
##
##        p: vector of probabilities.
##
```

```
##        n: number of observations. If 'length(n) > 1', the length is
##           taken to be the number required.
##
##     size: number of trials (zero or more).
##
##     prob: probability of success on each trial.
##
## log, log.p: logical; if TRUE, probabilities p are given as log(p).
##
## lower.tail: logical; if TRUE (default), probabilities are P[X <= x],
##           otherwise, P[X > x].
##
## Details:
##
##      The binomial distribution with 'size' = n and 'prob' = p has
##      density
##
##                         p(x) = choose(n, x) p^x (1-p)^(n-x)
##
##      for x = 0, ..., n.  Note that binomial _coefficients_ can be
##      computed by 'choose' in R.
##
##      If an element of 'x' is not integer, the result of 'dbinom' is
##      zero, with a warning.
##
##      p(x) is computed using Loader's algorithm, see the reference
##      below.
##
##      The quantile is defined as the smallest value x such that F(x) >=
##      p, where F is the distribution function.
##
## Value:
##
##      'dbinom' gives the density, 'pbinom' gives the distribution
##      function, 'qbinom' gives the quantile function and 'rbinom'
##      generates random deviates.
##
##      If 'size' is not an integer, 'NaN' is returned.
##
##      The length of the result is determined by 'n' for 'rbinom', and is
##      the maximum of the lengths of the numerical arguments for the
##      other functions.
##
##      The numerical arguments other than 'n' are recycled to the length
##      of the result.  Only the first elements of the logical arguments
##      are used.
##
## Source:
##
##      For 'dbinom' a saddle-point expansion is used: see
##
##      Catherine Loader (2000). _Fast and Accurate Computation of
##      Binomial Probabilities_; available from <URL:
##      http://www.herine.net/stat/software/dbinom.html>.
```

```
##
##      'pbinom' uses 'pbeta'.
##
##      'qbinom' uses the Cornish-Fisher Expansion to include a skewness
##      correction to a normal approximation, followed by a search.
##
##      'rbinom' (for 'size < .Machine$integer.max') is based on
##
##      Kachitvichyanukul, V. and Schmeiser, B. W. (1988) Binomial random
##      variate generation.  _Communications of the ACM_, *31*, 216-222.
##
##      For larger values it uses inversion.
##
## See Also:
##
##      Distributions for other standard distributions, including
##      'dnbinom' for the negative binomial, and 'dpois' for the Poisson
##      distribution.
##
## Examples:
##
##      require(graphics)
##      # Compute P(45 < X < 55) for X Binomial(100,0.5)
##      sum(dbinom(46:54, 100, 0.5))
##
##      ## Using "log = TRUE" for an extended range :
##      n <- 2000
##      k <- seq(0, n, by = 20)
##      plot (k, dbinom(k, n, pi/10, log = TRUE), type = "l", ylab = "log density",
##            main = "dbinom(*, log=TRUE) is better than  log(dbinom(*))")
##      lines(k, log(dbinom(k, n, pi/10)), col = "red", lwd = 2)
##      ## extreme points are omitted since dbinom gives 0.
##      mtext("dbinom(k, log=TRUE)", adj = 0)
##      mtext("extended range", adj = 0, line = -1, font = 4)
##      mtext("log(dbinom(k))", col = "red", adj = 1)
```

That's quite a lot of information packed into a single helpfile! For now, look in Usage section, and note that there are 4 binomial functions listed. We will go through each one.

The dbinom() function

We'll start with the dbinom() function. This function corresponds with Excel's BINOM.DIST() function with cumulative = FALSE. The `dbinom()` function returns the probability of x successes, given trials (size) and the probability of success on each trial (prob). We could use this function, for example, to learn the probability of 4 heads out of 10 coin flips, given the probability of heads (success) is 0.5.

Let's look at this function's arguments:

```
args("dbinom")
```

```
## function (x, size, prob, log = FALSE)
## NULL
```

As shown, there are 4 arguments:

- x is a value of interest. Here, x would be 4 heads.
- size is the number of trials. Here, size would be 10 trials.
- prob is the probability of a success. Here, prob would be 0.5.
- log is a logical argument (TRUE or FALSE) indicating whether the probabilities that are returned should be given on the log scale (the natural log scale). The default is FALSE.

```
# set trials to 10
trials <- 10

# set probability of success
p <- 0.5

# return the probability of observing 4 successes, given trials and p
dbinom(x = 4, size = trials, prob = p)
```

```
## [1] 0.2050781
```

We can use the `dbinom()` function to create a binomial probability distribution - we just need to provide the parameters of the function. We'll start by creating a vector called **successes** and let it run from 0 to the total trials:

```
# create the binomial distribution given trials and p
successes <- seq(from = 0, to = trials, by = 1)

# look at the vector
successes
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

Next, we can use the `dbinom()` function, and pass in our vector of successes for the **x** argument. R will evaluate the binomial probability for each and every alternative in our **successes** vector, a process known as "vectorization" in R.

```
# get the binomial probabilities for each x
binomial.prob <- dbinom(x = successes, size = trials, prob = p)

# look at the binomial probabilities
binomial.prob
```

```
##  [1] 0.0009765625 0.0097656250 0.0439453125 0.1171875000 0.2050781250
##  [6] 0.2460937500 0.2050781250 0.1171875000 0.0439453125 0.0097656250
## [11] 0.0009765625
```

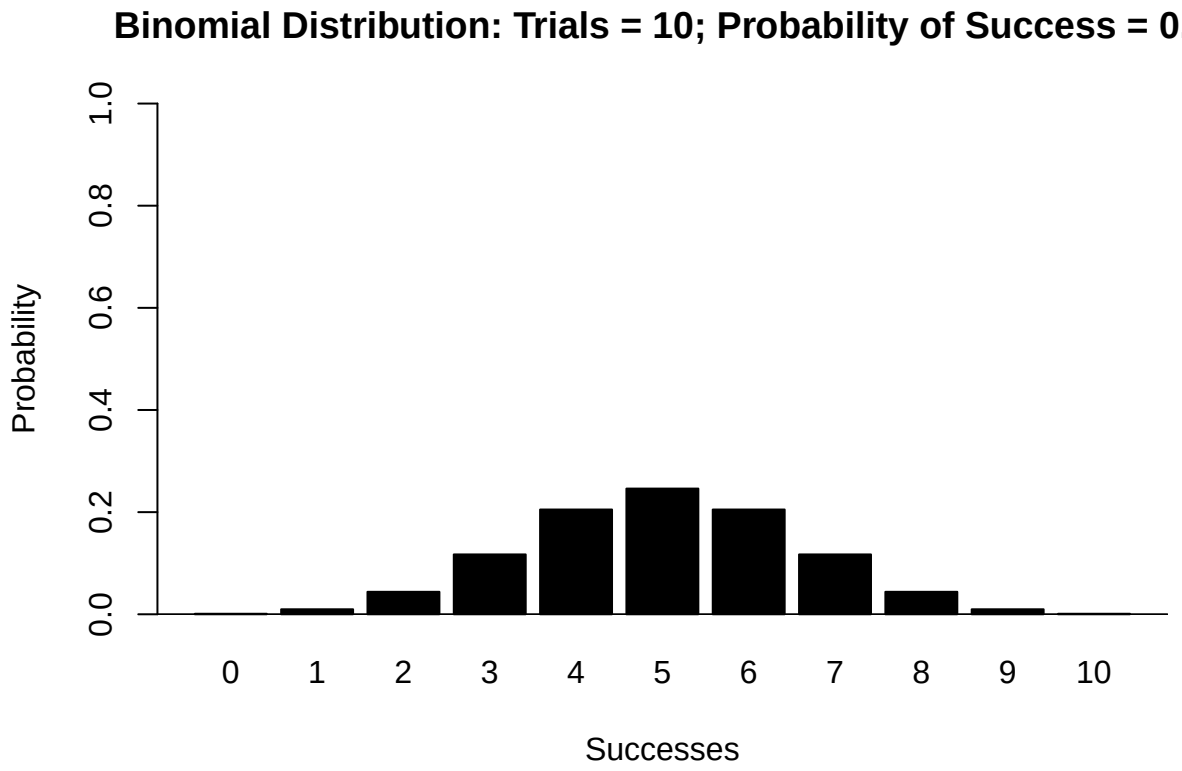
This is a particular binomial distribution, namely one with 10 trials and a probability of success of 0.5. A property of all binomial distributions is that the sum of the probabilities is 1 – this is true of all probability mass functions, not just the binomial function. Let's confirm this:

```r
sum(binomial.prob)
```

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

A plot of the all binomial outcomes is called the Binomial Distribution, and here we will use the built-in `barplot()` function, which is one of the base R plotting functions. Note: you may prefer other graphing packages in R, but we will limit ourselves to the built-in base plotting functions).

```r
# graph the distribution with barplot()
barplot(binomial.prob,
  main = paste0("Binomial Distribution: Trials = ", trials, "; Probability of Success = ", p),
  xlab = "Successes",
  names.arg = successes,
  ylab = "Probability",
  ylim = c(0,1),
  col = "black")

  # add a horizontal line to the x axis
  abline(h = 0)
```



The pbinom() function

The `pbinom()` function will return cumulative binomial probabilities. The function corresponds with Excel's BINOM.DIST() function where the cumulative argument is set to TRUE. Once again, we need to tell R

which binomial distribution we are talking about. We'll stick with our previous example in which the number of trials is 10 and the probability of success is 0.5. Let's have a look at this function's arguments:

```
args('pbinom')
```

```
## function (q, size, prob, lower.tail = TRUE, log.p = FALSE)
## NULL
```

Here, five arguments are required (the last two have default values and can be ignored if you want to use the defaults):

- q is vector of quantiles
- size is the number of trials. Here, size would be 10 trials.
- prob is the probability of a success. Here, prob would be 0.5.

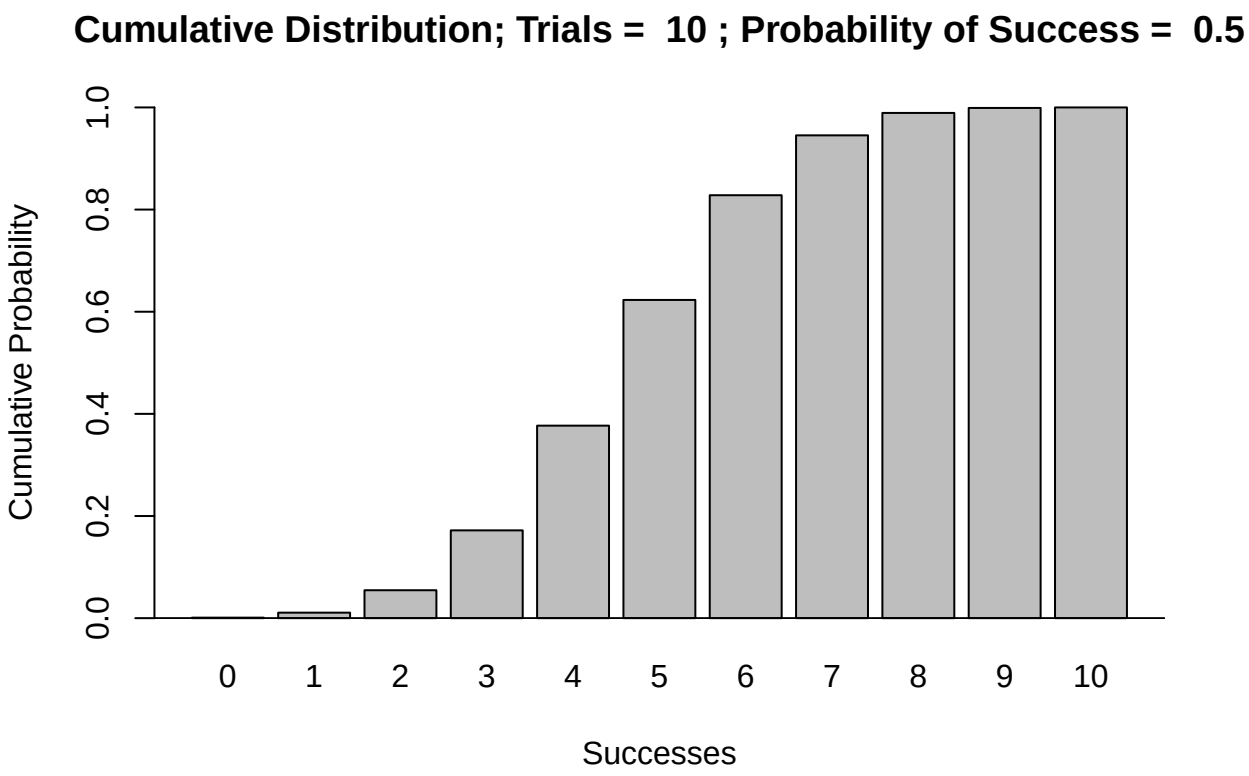
We can use `pbinom` now to evaluate the cumulative probability of observing *at least* each value in the **successes** vector:

```
cumulative.prob <- pbinom(q = successes, size = trials, prob = p)
cumulative.prob
```

```
##  [1] 0.0009765625 0.0107421875 0.0546875000 0.1718750000 0.3769531250
##  [6] 0.6230468750 0.8281250000 0.9453125000 0.9892578125 0.9990234375
## [11] 1.0000000000
```

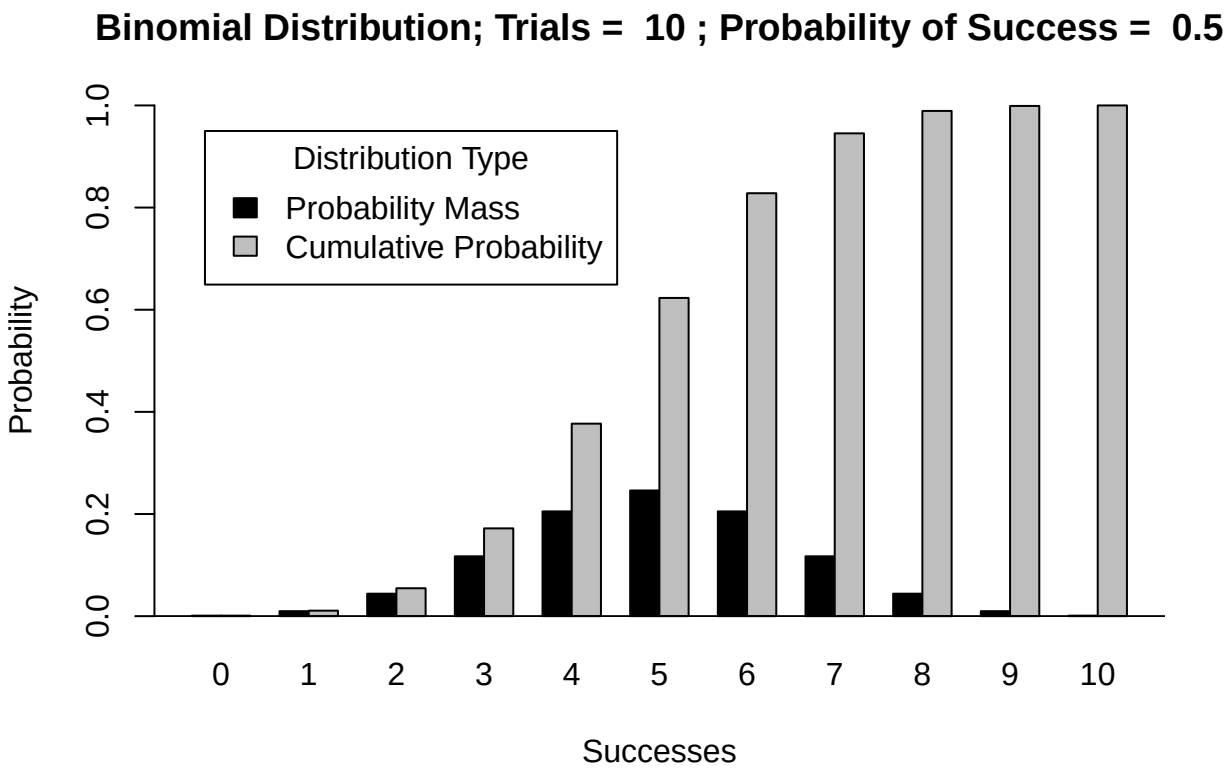
Again, this is an example of vectorization in R: it is evaluating each entry in the successes vector and returning the cumulative probability. For example, the second entry in the successes vector is the number 1, and the value returned is 0.0107421875. This is the probability of observing 1 or fewer heads out of 10 coin flips. Let's graph this distribution:

```
# graph the cumulative binomial
barplot(cumulative.prob,
        main = paste("Cumulative Distribution; Trials = ",
                     trials, "; Probability of Success = ", p),
        xlab = "Successes",
        names.arg = successes,
        ylab = "Cumulative Probability",
        ylim = c(0,1),
        col = "gray")

        # add horizontal line to fill in x-axis
        abline(h = 0)
```



The cumulative distribution can be thought of as stacking the bars from left to right. You should be able to see this more clearly by graphing both distributions side by side:

```
# first, combine the vectors into a matrix; notice the structure in 2 rows
data <- rbind(binomial.prob, cumulative.prob)

# plot
barplot(data, beside = T,
        col = c("black", "gray"),
        main = paste("Binomial Distribution; Trials = ",
                     trials,
                     "; Probability of Success = ",
                     p),
        xlab = "Successes",
        ylab = "Probability",
        ylim = c(0,1),
        names.arg = successes)

        # add line to fill in x axis
        abline(h = 0)

        # add a legend
        legend("topleft", title = "Distribution Type",
               legend = c("Probability Mass", "Cumulative Probability"),
               inset = 0.05,
               fill = c("black", "gray"))
```



The qbinom() function

The `qbinom()` function returns the value associated with a given percentile of a given binomial distribution. Once again, we need to tell R *which* binomial function we are dealing with (here, 10 coin flips and a probability of heads = 0.5). We specify a given percentile of interest in the argument called "p". Suppose we are interested in the 50th percentile of our binomial function:

```
# find the value associated with the 50th percentile of our binomial distribution
qbinom(p = 0.5, size = trials, prob = p)
```

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

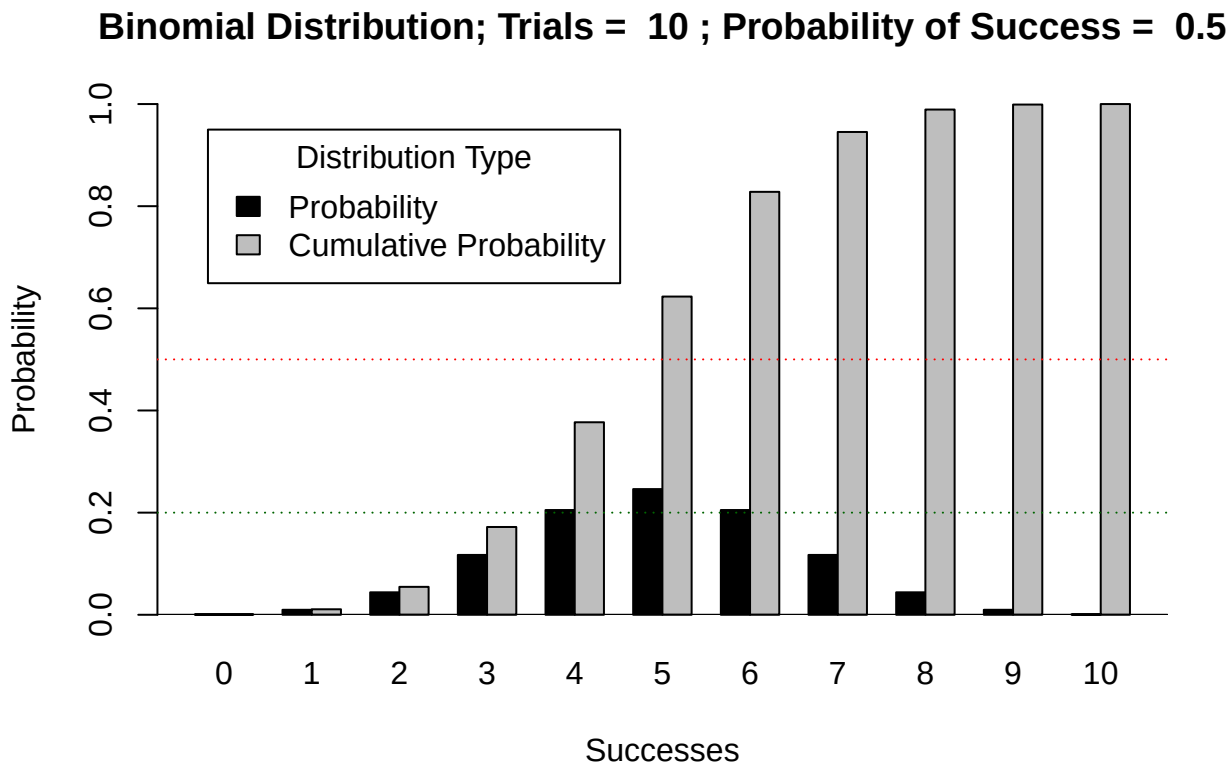
R returns the value of 5, indicating the 5 heads is dead center of our distribution. Let's try the 20th percentile:

```
# find the value associated with the 20th percentile of the above binomial distribution
qbinom(p = 0.2, size = trials, prob = p)
```

```
## [1] 4
```

These values are easy to find on the cumulative distribution. Below, we add horizontal lines at our two percentiles (0.5 in red and 0.2 in green). If you follow the red horizontal line (p = 0.5) from left to right, you should see that hits the "5" successes bar in the cumulative distribution, which is the value returned by `qbinom()`. Similarly, if you follow the green horizontal line (p = 0.2) from left to right, you should see that hits the "4" successes bar in the cumulative distribution, which is the value returned by `qbinom()`.

```r
barplot(data, beside = T,
        col = c("black", "gray"),
        main = paste("Binomial Distribution; Trials = ", trials, "; Probability of Success = ", p),
        xlab = "Successes",
        ylab = "Probability",
        ylim = c(0,1),
        names.arg = successes)

        # add line to fill in x axis
        abline(h = 0)

        # add a legend
        legend("topleft", title = "Distribution Type",
               legend = c("Probability", "Cumulative Probability"),
               inset = 0.05,
               fill = c("black", "gray"))

abline(h = 0.5, col = 'red', lty = 'dotted')
abline(h = 0.2, col = 'darkgreen', lty = 'dotted')
```



The rbinom() function

The final binomial function listed is the `rbinom()` function, which can be used to draw a random value from a given binomial distribution. This function corresponds with Excel's BINOM.INV() function. Again we

need to tell R *which* binomial distribution we are talking about (size = 10, prob = 0.5). And we further need to tell R how many random values we'd like to pull from this distribution (the "n" argument):

```
args('rbinom')
```

```
## function (n, size, prob)
## NULL
```

Let's draw 1 random value from the binomial distribution specified.

```
rbinom(n = 1, size = trials, prob = p)
```

```
## [1] 6
```

If we repeat this command to draw a new value, we will likely get a different value. Let's try it:

```
rbinom(n = 1, size = trials, prob = p)
```

```
## [1] 6
```

We will be using the `rbinom()` function later to simulate occupancy patterns. If you wish to force R to use the same sequence of random values (for repeatability), you can use the `set.seed()` function.

```
# to enable repeatability, use a random seed
set.seed(10)
rbinom(n = 1, size = trials, prob = p)
```

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

Now let's run this line again. Let's use the same seed though before running the code:

```
# notice that when the seed is set to the same value, the same random value is returned
set.seed(10)
rbinom(n = 1, size = trials, prob = p)
```

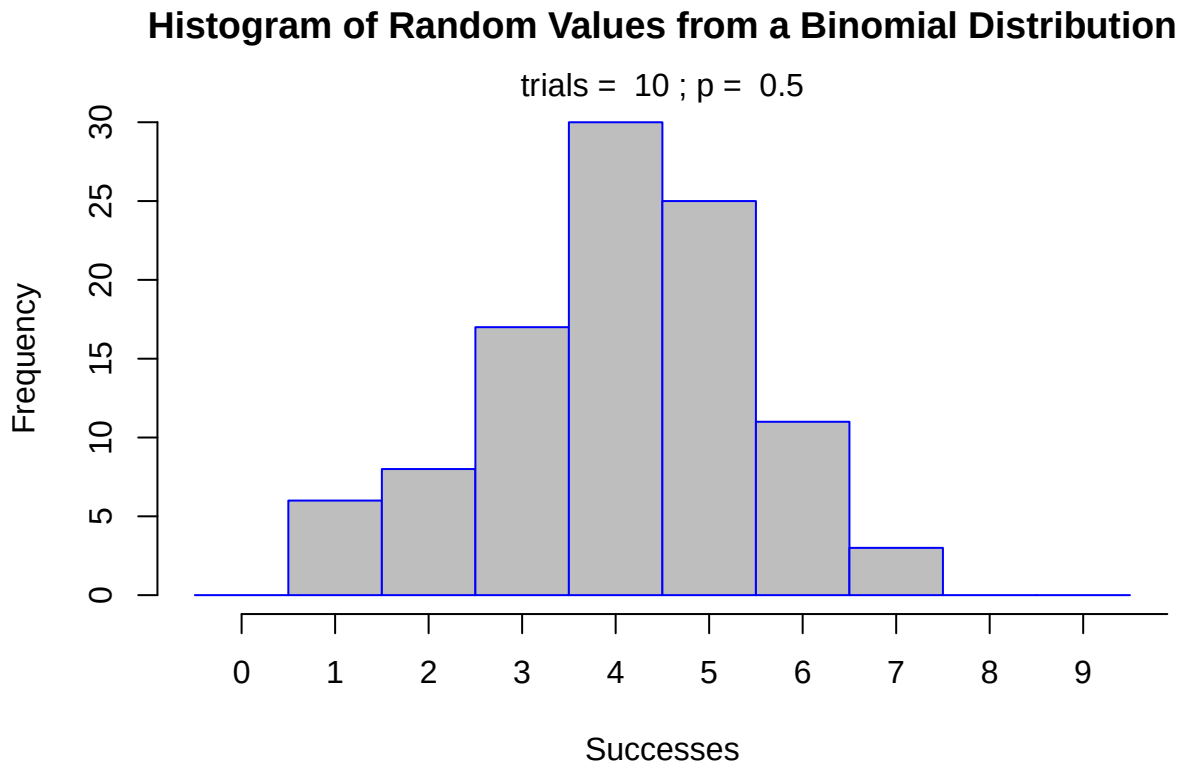
```
## [1] 5
```

```
# draw 100 random values from the binomial distribution above
set.seed(189)
random.values <- rbinom(n = 100, size = trials, prob = p)
random.values
```

```
##   [1] 6 5 5 6 4 2 5 5 5 4 7 6 6 6 4 3 6 3 5 5 7 5 4 7 3 7 7 6 6 4 4 4 5 3 6
##  [36] 6 5 5 5 6 6 4 6 2 7 5 5 5 5 6 6 2 4 3 4 6 2 5 8 2 6 5 5 4 5 6 5 2 6 5
##  [71] 3 4 5 4 5 5 6 7 3 7 6 8 5 3 7 5 4 6 5 6 6 7 4 7 8 6 4 5 5 4
```

```
# create a histogram of the values
help("hist")
```

```r
# customize the histogram with hist()
hist1 <- hist(random.values,
     breaks = successes,
     xaxt = 'n',
     main = "Histogram of Random Values from a Binomial Distribution",
     xlab = "Successes",
     ylab = "Frequency",
     border = "blue",
     col = "gray",
     xlim = c(min(successes), max(successes)))

    # add a subtitle
    mtext(text = paste("trials = ", trials, "; p = ", p), side = 3)

    # customize the x axis
    axis(side = 1, at = successes + 0.5, labels = successes)
```




```r
# look at the stored histogram
hist1
```

```
## $breaks
##  [1]  0  1  2  3  4  5  6  7  8  9 10
##
## $counts
##  [1]  0  6  8 17 30 25 11  3  0  0
```

```
##
## $density
##  [1] 0.00 0.06 0.08 0.17 0.30 0.25 0.11 0.03 0.00 0.00
##
## $mids
##  [1] 0.5 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5
##
## $xname
## [1] "random.values"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

Likelihood

So far, we've been exploring the binomial functions in R where the number of trials is 10 and the probability of a success is 0.5.

As a reminder, the binomial probability mass function is written below, and was introduced in depth in our spreadsheet tutorial:

$$f(y|n,p) = \binom{n}{k} p^y (1-p)^{n-y}$$

In an occupancy framework, the number of trials is akin to the number of sites we wish to sample for our target species. We don't know the probability of success. Instead, we survey our sites and record how many sites held the species of interest. In other words, we know y, but we don't know p. The equation is the same: it's just that we have different pieces of information in hand. Here, we can write the likelihood of p, given n and y as:

$$L(p|n,y) = \binom{n}{k} p^y (1-p)^{n-y}$$

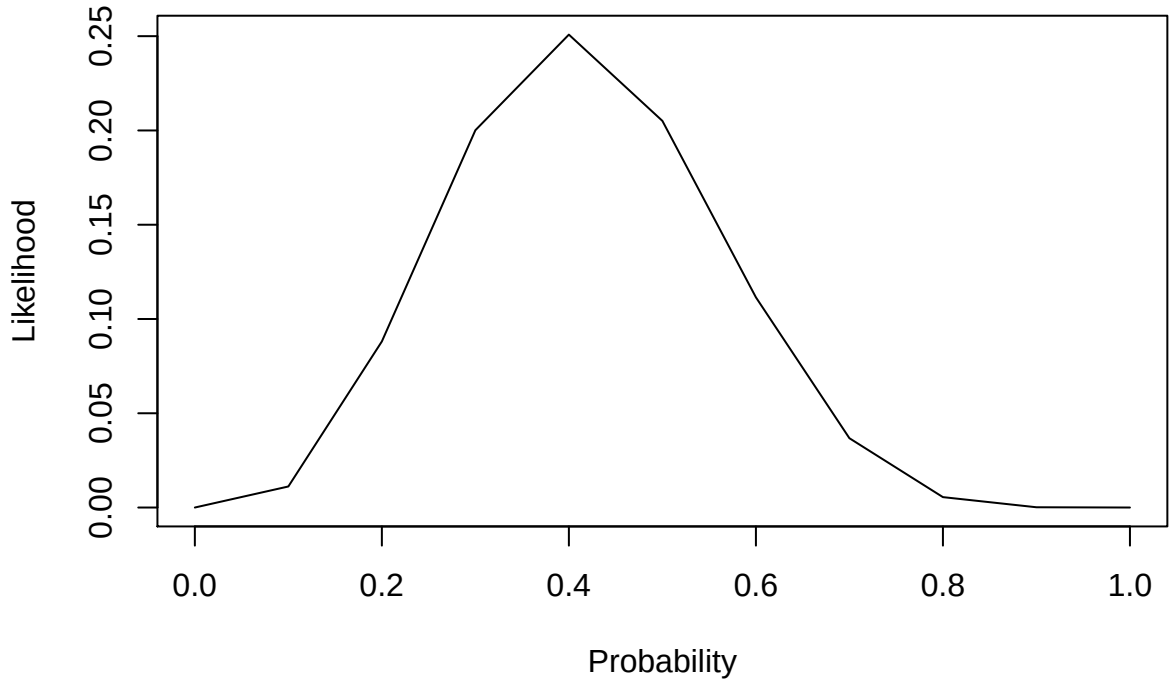
Let's assume we survey 10 sites, and that our target species exists on 4 of them. We'd like to use our information to estimate p, the probability of site occupancy. Let's create some new objects to reflect this scenario:

```
# the species was found in 4 sites
y <- 4

# we don't know p.  Let's set up a vector of possibilities ranging from 0 to 1.
probs <- seq(from = 0, to = 1, by = 0.1)

probs
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

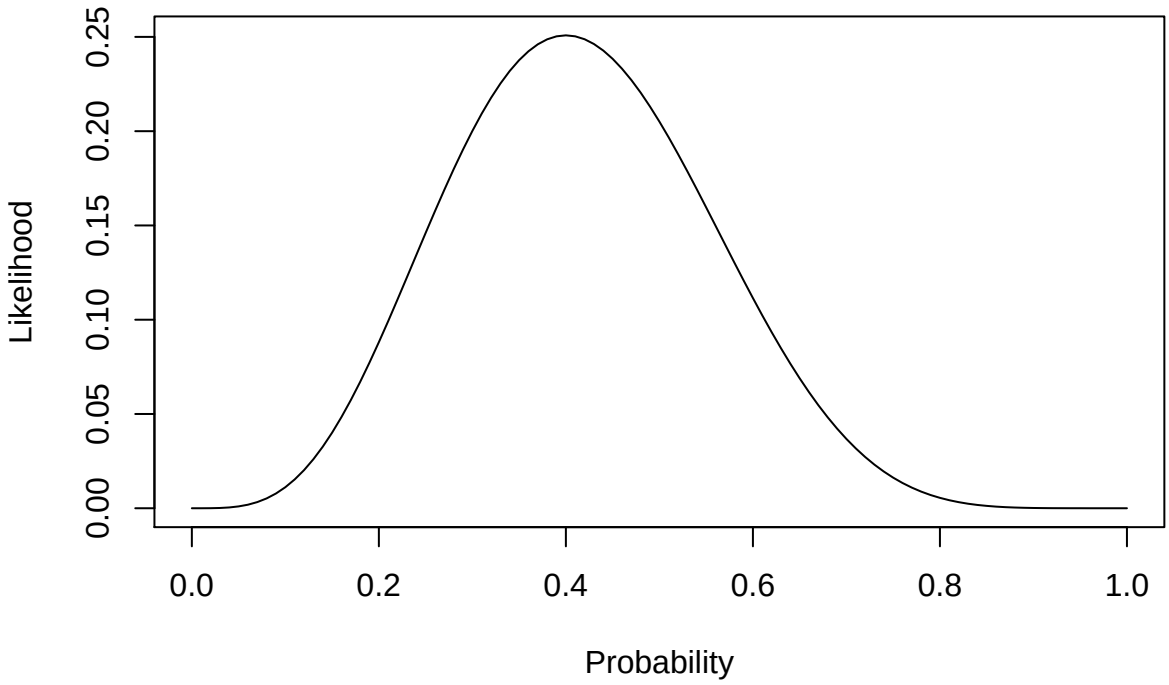
Now, let's use the `dbinom()` function to get the binomial probabilities associated for this challenge. Remember, the function returns probabilities given y (in R this argument is called "x"), n (in R this argument is called "trials"), and p (in R this argument is called "prob").

```r
# return the likelihoods for each value of p
likes <- dbinom(x = y, size = trials, prob = probs)

# plot the likelihood surface
plot(x = probs,
     y = likes,
     type = "l",
     xlab = "Probability",
     ylab = "Likelihood")
```



Note the maximum likelihood value is centered at p = 0.4, which is the same as 4/10. This graph looks a bit jagged only because we examined only 11 values of p. Let's up the game a bit:

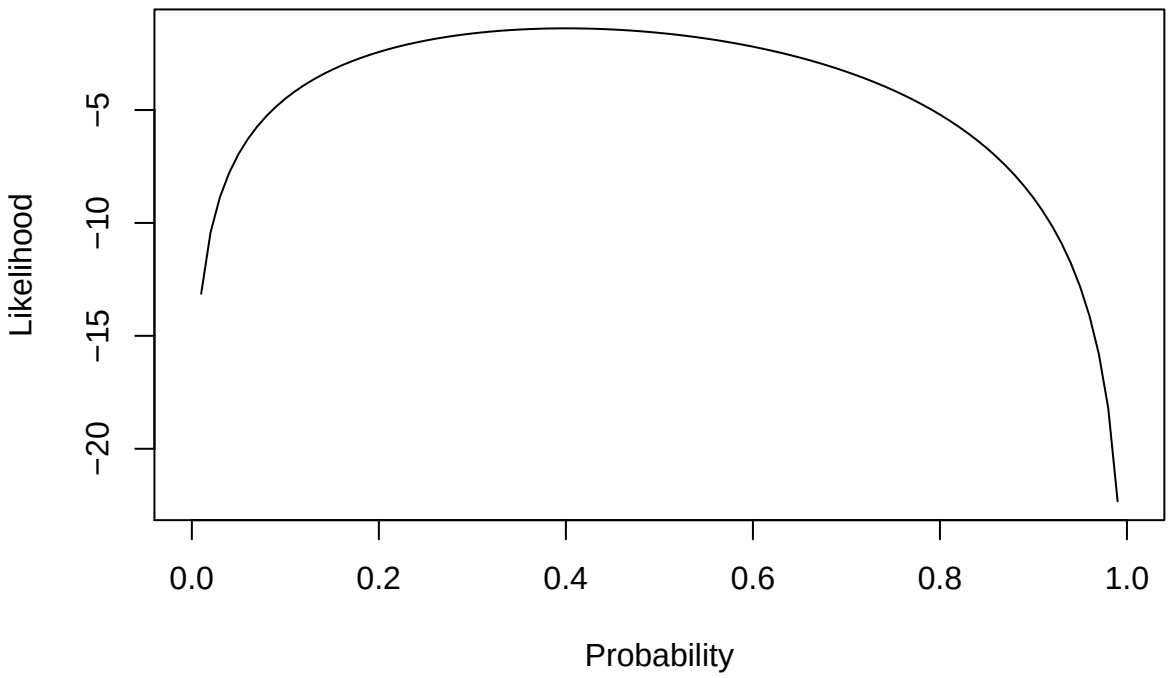
```r
# we don't know p.  Let's set up a vector of possibilities ranging from 0 to 1.
probs <- seq(from = 0, to = 1, by = 0.01)

# return the likelihoods for each value of p
likes <- dbinom(x = y, size = trials, prob = probs)

# plot the likelihood surface
plot(x = probs,
     y = likes,
     type = "l",
     xlab = "Probability",
     ylab = "Likelihood")
```



Notice that, unlike the binomial distribution we explored earlier, the area under this likelihood curve does not integrate to 1. The peak of this curve is called the maximum likelihood estimate, or MLE. The curve itself provides information about how likely site occupancy probability may be, given that we observed our target species at 4 of the 10 sites.

We can also examine the log-likelihood function. In R, this is easily done by setting the "log" argument to TRUE.

```
# return the log likelihoods for each value of p
likes <- dbinom(x = y, size = trials, prob = probs, log = T)

# plot the likelihood surface
plot(x = probs,
     y = likes,
     type = "l",
     xlab = "Probability",
     ylab = "Likelihood")
```



Once again, we see that the top of this curve is centered over 0.4, indicating that this value of p is our most likely estimate of occupancy probability. The curvature of this function contains a wealth of information that we can use to put "bounds" or confidence intervals around our estimate of 0.4. After all, you could easily observe 4 occupied sites out of 10, even if p were really 0.7 or 0.2. We will revisit this topic in later exercises.