

CHAPTER 5

R

Copyright 2008 William M. Briggs, wmbriggs.com

1. R

R is a fantastic, hugely supported, rapidly growing, infinitely extensible, operating-system agnostic, free and open source statistical software platform. Nearly everybody who is anybody uses R, and since I want you to be somebody, you will use it, too. Some things in R are incredibly easy to do; other tasks are bizarrely difficult. Most of what makes R hard for the beginner is the same stuff that makes any piece of software hard; that is, getting used to expressing your statistical desires in computerese. As such an environment can be strange and perplexing at first, some students experience a kind of peculiar stress that is best described by example. Here is a video from a Germany showing a young statistics student who experienced trouble understanding R:

<http://youtube.com/watch?v=PbcctWbC8Q0>

Be sure that this doesn't happen to you. Remember what Douglas Adams said: *Don't panic*.

The best way to start is by going to r-project.org and click the CRAN under the Download heading. You can't miss it. After that, you have to choose a *mirror*, which means one of the hundreds of computers around the world that host the software. Obviously, pick a site near you. Once that's done, and choose your platform (your operating system, like Linux or one of the others), and then choose the **base** package. Step-by-step instructions are at this book's website: wmbriggs.com/book. It is no more difficult to install than any other piece of software.

This is not the place to go over all the possibilities of R; just the briefest introduction will be given, because there are far better

places available online (see the book website for links). But there are a few essential commands that you should not do without. These are

Command	Description
<code>help(command)</code>	Does the obvious: always scroll down to the bottom of the help to see examples of the command.
<code>?command</code>	Same as <code>help()</code>
<code>apropos('string')</code>	If you cannot remember the name of a command—and I always forget—but remember is started with <code>co</code> —something, then just type <code>apropos('co')</code> and you'll get a complete list of commands that have <code>co</code> anywhere in their names.
<code>c()</code>	This is the concatenation function: typing <code>c(1,2)</code> concatenates a 2 to 1, or sticks on the end 1 the number 2, so that we have a <i>vector</i> of numbers.

The Appendix gives a fuller list of R commands.

It is important to understand that R is a *command-line* language, which we may interpret as meaning that all commands in R are *functions* which must be typed into the console. These are objects that are a command name plus a left and right parenthesis, with variables (called arguments) stuck in between, thus: `plot(x,y)`. Remember that you are dealing with computers, which are literal, intolerant creatures (much like the people who want to ban smoking), and so cannot abide even the slightest deviation from its expectations. That means, if instead of `plot(x,y)`, you type `lot(x,y)`, or `plot x,y)`, or `plot(,y)`, or `plot(x,y` things will go awry. R will try to give you an idea of what went wrong by giving you an error message. Except in cases like that last typo, which will cause you to develop stress lines, because all you'll see is this

+

and every attempt of yours to type anything new, or hit `enter` 100 times, will not do a thing except give you more lines of + or other screwy errors. Because why? Because you typed `plot(x,y;`

that is, you typed a left parenthesis (right before the `x`) and you never “closed” it with a right parenthesis, and R will simply wait forever for you to type one in.

The solution is to enter a right parenthesis, or hit

`ctrl+c`

which means the control key plus the `c` key simultaneously, which “breaks” the current computation.

Using R means that you have to memorize (!) and type in commands instead of using a graphical user interface (GUI), which is the standard point-and-click screen with which you are probably familiar. It is my experience that students who are not used to computers start freaking out at this point; however, there is no need to. I have made everything very, very easy and all you have to do is copy what you see in the book to the R screen. All will be well. I promise.

GUIs are very nice things, incidentally, and R has one that you can download and play with. It is called the **R Commander**. Like all GUIs, some very basic functionality is included that allows you to, well, point and click and get a result. Problem is, the very second you want to do something different than what is available from the GUI, you are stuck. With statistics, we often want to do something differently, so we will stick with the command line.

2. R binomially

By now, you are eagerly asking yourself: “Can R help up with those binomial calculations like in the Thanksgiving example?” Let’s type `apropos('bino')` and see, because, after all, ‘bino’ is something like binomial. The most likely function is called `binomial`, so let’s type `?binomial` and see how it works. Uh oh. Weird words about “family objects” and the function `glm()`, and that doesn’t sound right. What about one of the functions like `dbinom()`? Jackpot. We’ll look at these in detail, since it turns out that this structure of four functions is the same for every distribution. The functions are in this table:

- dbinom** The probability of *density* function: given the **size**, or n , and **prop**, or p , this calculates the probability that we see x successes; this is equation (11).
- pbinom** The distribution function, which calculates the probability that the number of successes is less than or equal to some **a**.
- qbinom** This is the “quantile” function, which calculates, given a probability from the distribution function, which value of **q** it is associated with. This will be made clear with some examples with the normal distribution later.
- rbinom** This generates a “random” binomial number; and since random means unknown, this means it generates a number that is unknown in some sense; we’ll talk about this later.

Let’s go back to the Thanksgiving example, which used a binomial. Moe can calculate, given $n = \text{size} = 3, p = \text{prob} = 0.1$, his probabilities using R:

```
dbinom(0,3,.1)
```

which gives the probability of taking nobody along for the ride. The answer is `[1] 0.729`. The “[1]” in front of the number just means that you are only looking at line number 1. If you asked for dozens of probabilities, for example, R would space them out over several lines. Let’s now calculate the probability of taking just 0, just 1, etc.

```
dbinom(c(0,1,2,3),3,.1)
```

where we have “nested” two functions into one: the first is the concatenation function `c()`, where we have stuck the numbers 0 through 3 together, and which shows you the `dbinom()` function can calculate more than one probability at a time. What pops out is

```
[1] 0.729 0.243 0.027 0.001;
```

that is, the exact values we got above for taking 0 or 1 or 2 etc. along for the ride. Now we can look at the distribution function:

```
pbinom(c(0,1,2,3),3,.1);
```

and we get

```
[1] 0.729 0.972 0.999 1.000.
```

This is the probability of taking 0 or less, 1 or less, 2 or less, and 3 or less. The last probability very obviously has to be 1, and will always be 1 for any binomial (as long as the last value in the function `c(0,1,...,n)` equals n).

There turns out to be a shortcut to typing the concatenation function for simple numbers, and here it is:

```
c(0,1,2,...,n) = 0:n.
```

So we can rewrite the first function as `dbinom(0:3,3,.1)` and get the same results.

We can nest functions again and make pretty pictures

```
plot(dbinom(0:3,3,.1))
```

And that's it for any binomial function. Isn't that simple? (The answer is *yes*.) The commands never change for any binomial you want to do.

3. R normally

Can R do normal distributions as well? *Can* it! Let's type in `apropos('normal')` and see what we get. A lot of gibberish, that's what. Where's the normal distribution? Well, it turns out that computer programmers are a lazy bunch, and they often do not use all the letters of a word to name a function (too much typing). Let's try `apropos('norm')` instead (which no matter what should give us at least as many results, right? This is a question of logic, not computers.). Bingo. Among all the rest, we see `dnorm` and `pnorm` etc., just like with the binomial. Now type `?dnorm` so we can learn about our fun function. Same layout as the binomial; only difference being we need to supply a “mean” and “sd” (the m and s). Sigh. This is an example of R being naughty and misusing the terminology that I earlier forbade: m and s are *not* a mean and standard deviation. It's a trap too many fall into. We'll work with it, but just remember “mean” and “sd” actually imply our *parameters* m and s .

You will recall from our discussion of normals that we cannot compute a probability of seeing a single number (and if you don't remember, shame on you: go back and read Chapter 4). The function `dnorm` does *not* give you this number, because that probability is always 0; instead, it gives you a “density”, which

means little to us. But we *can* calculate the probability of values being in some interval using the `pnorm` function. For example, to calculate $\Pr(x < 10|m = 10, s = 20, E_N)$, use

```
pnorm(10,10,20)
```

and you should see [1] 0.5. But you already knew that would be the answer before you typed it in, right? (*Right?*) Let's try a trickier one: $\Pr(x < 0|m = 10, s = 20, E_N)$; type `pnorm(0,10,20)` and get [1] 0.3085375. So what is this probability: $\Pr(x > 0|m = 10, s = 20, E_N)$ (*x greater than 0*)? Think about it. x can either be less than or greater than 0; the probability it is so is 1. So $\Pr(x < 0|m = 10, s = 20, E_N) + \Pr(x > 0|m = 10, s = 20, E_N) = 1$. Thus, $\Pr(x < 0|m = 10, s = 20, E_N) \Pr(x > 0|m = 10, s = 20, E_N) = 1 - \Pr(x < 0|m = 10, s = 20, E_N)$. We can get that in R by typing

```
1-pnorm(0,10,20)
```

and you should get [1] 0.6914625, which is $1 - 0.3085375$ as expected.

By the way, if you are starting to feel the onset of a freak out, and wonder “Why, O why, can't he give us a point and click way to do this!” Because, dear reader, a point and click way to do this does not exist. Stop worrying so much. You'll get it.

What is $\Pr(15 < x < 18|m = 15, s = 5, E_N)$ (which might be reasonable numbers for the temperature example)? Any interval splits the data into three parts: the part less than the lower bound (15), the part of the interval itself (15-18), and the part larger than the upper bound (18). We already know how to get $\Pr(x < 15|m = 15, s = 5, E_N)$, which is `pnorm(15,15,5)`, and which equals 0.5. We also know how to get $\Pr(x > 18|m = 15, s = 5, E_N)$, which is `1-pnorm(18,15,5)`, and which equals 0.2742531. This means that $\Pr(x < 15 \text{ or } x > 18|m = 15, s = 5, E_N)$, using probability rule number 1, is $0.5 + 0.2742531 = 0.7742531$. Finally, 0.7742531 is the probability of *not* being in the interval, so the probability of being *in* the interval must be one minus this, or $1 - 0.7742531 = 0.2257469$. A lot of work. We could have jumped right to it by typing

```
pnorm(18,15,5)-pnorm(15,15,5).
```

This is the way you write the code to compute the probability of any interval—remembering to input your own m and s of course!

4. Advanced

. You don't need to do this section, because it is somewhat more complicated. Not much, really, but enough that you have to think more about the computer than you do the probability.

Our goal is to plot the picture of a normal density. The function `dnorm(x, 15, 5)` will give you the value of the normal density, with an $m = 15$ and $s = 5$, for some value of x . To picture the normal, which is a picture of densities for a range of x , we somehow have to specify this range. Unfortunately, there is no way to know in advance which range you want to plot, so getting the exact picture you want takes some work. Here is one way:

```
x = seq(-4, 4, .01)
```

which gives us a `sequence` of numbers from -4 to 4 in increments of 0.01. Thus, $x = -4.00, -3.99, -3.99, \dots, 4$. Calculating the density of each of these values of x is easy:

```
dnorm(x)
```

where you will have noticed that I did not type a m or s . Type `?dnorm` again. It reads `dnorm(x, mean=0, sd=1, log = FALSE)`. Ignoring the `log = FALSE` bit, we can see that R supplies helpfully *default* values of the parameters. They are default, because if you are happy with the values chosen, you do not have to type in your own. In this case, $m = 0$ and $s = 1$, which is called a *standard normal*. Anyway, to get the plot is now easy:

```
plot(x, dnorm(x), type='l')
```

This means, for every value of x , plot the value of `dnorm` at that value. I also changed the plot type to a line with `type='l'`, and which makes the graph prettier. Try doing the plot without this argument and see what you get.

5. Homework

- (1) Type `demo(graphics)` then `demo(persp)` in the command window, and follow the instructions.
- (2) Open a text file and call it `Rcode.R`. Save it anywhere you like, but save it as a *text* file. **Do not** save it as a–God help us–Microsoft Word file. In that file, you will type all the commands that you want R to run. When you do want R to run a

particular line, cut and paste that line from the file `Rcode.R` to `R`. Why do this? So you can have a record of all your commands, and so you don't have to retype them over and over again.

- (3) Calculate the probability that the CMU football teams wins $x = 0, x = 1, \dots, x = 12$ games, using the information from the previous Chapter. Round your answers! Please do not try to write out 43 digits for each probability. Cut and paste the results and print them if you like.
- (4) `dbinom(0, 10, .5)` calculates the probability of getting 0 heads in 10 flips of a coin (right?). Write down the `R` code to calculate the probability of *getting at least one head*. HINT: getting at least one head means *not* getting zero heads.
- (5) What is the probability of seeing an $x = 7$ where your uncertainty in x is represented by a normal distribution with parameters 7 and $\sqrt{10}$?
- (6) Suppose before you decide to take anybody for the Thanksgiving trip, you learn that Moe and Curly's brother Shemp showed up, so now there are 4 people who need a ride. Describe the situation in terms of probability and calculate every single thing that can happen.
- (7) What is $\Pr(x < -10 \text{ or } x > 5 | m = -5, s = 3, E_N)$?
- (8) What is `pnorm(0)`? Try to do this in your head first. Either way, explain why the number is what it is.