BAYESIAN ANALYSIS WITH EXCEL AND R

CONRAD CARLBERG

Bayesian Analysis with Excel and R

Conrad G. Carlberg

Contents at a Glance

To access bonus materials, please register your book at informit.com/register and enter ISBN 9780137580989.

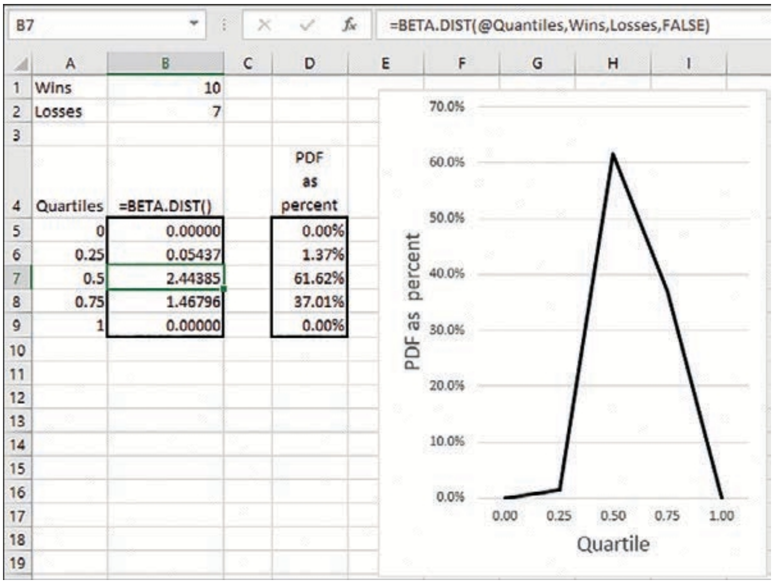


Figure 3.4
The chart can be misleading if it shows too few quantiles.

B7 =BETA.DIST(@Quantiles,Wins,Losses,FALSE)

	A	B	C	D
1	Wins	10		
2	Losses	7		
3				
4	Quartiles	=BETA.DIST()		PDF as percent
5	0	0.00000		0.00%
6	0.25	0.05437		1.37%
7	0.5	2.44385		61.62%
8	0.75	1.46796		37.01%
9	1	0.00000		0.00%

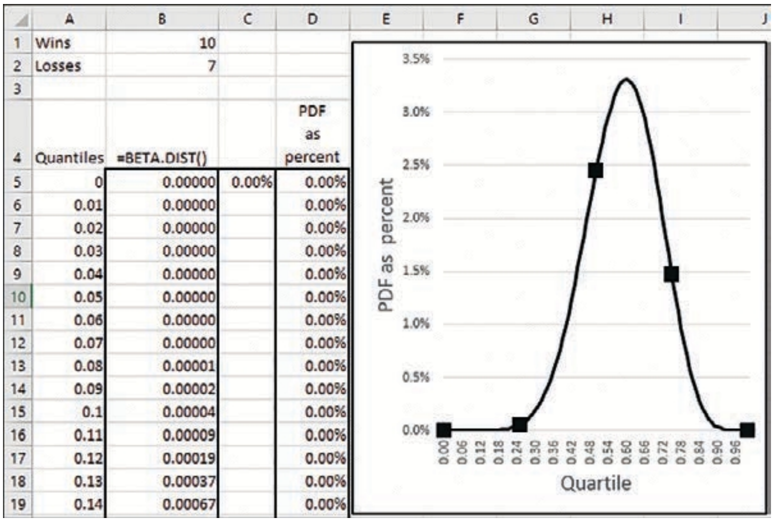


Figure 3.5
This chart is a much better depiction of what the beta distribution with 10 successes in 17 trials looks like.

	A	B	C	D
1	Wins	10		
2	Losses	7		
3				
4	Quantiles	=BETA.DIST()		PDF as percent
5	0	0.00000	0.00%	0.00%
6	0.01	0.00000		0.00%
7	0.02	0.00000		0.00%
8	0.03	0.00000		0.00%
9	0.04	0.00000		0.00%
10	0.05	0.00000		0.00%
11	0.06	0.00000		0.00%
12	0.07	0.00000		0.00%
13	0.08	0.00001		0.00%
14	0.09	0.00002		0.00%
15	0.1	0.00004		0.00%
16	0.11	0.00009		0.00%
17	0.12	0.00019		0.00%
18	0.13	0.00037		0.00%
19	0.14	0.00067		0.00%

The chart in Figure 3.5 shows the 100 quantiles in column A as a curved line. It also shows the five quantiles from Figure 3.4 as individual squares that fall directly on the curve. If all you saw were the squares, your mind's eye might well fill in the missing data very differently.

Decoding Excel's Help Documentation for `BETA.DIST`

Here's what Excel's Help has to say about the second and third arguments to the `BETA.DIST` function:

- **Alpha:** Required. A parameter of the distribution.
- **Beta:** Required. A parameter of the distribution.

When I first read that, I couldn't tell what it was talking about. Granted that alpha and beta are parameters of the beta distribution, which ones are they? The mean and the variance? The median and the skewness? P-bar and theta? Now I ask you.

After some experimenting (and an unenlightening tour of Google) I figured out that alpha and beta are related to the mean, dispersion, and other descriptive measures, but only just barely. With the binomial distribution, we're still working with two values, A and Not-A, so True and False, Win and Loss, Sale and No Sale, but we supply number of trials rather than number of losses as the second argument to `BINOM.DIST`. In the beta distribution, *Alpha* refers to one of those values and *Beta* to the other, so it's 10 True and 7 False, or 10 Wins and 7 Losses, or 10 Sales and 7 No Sales.

NOTE The mean of a beta distribution is $\alpha / (\alpha + \beta)$. Its variance is $\alpha\ \beta / [\ (\alpha + \beta)^2\ (\ \alpha + \beta + 1)]$.

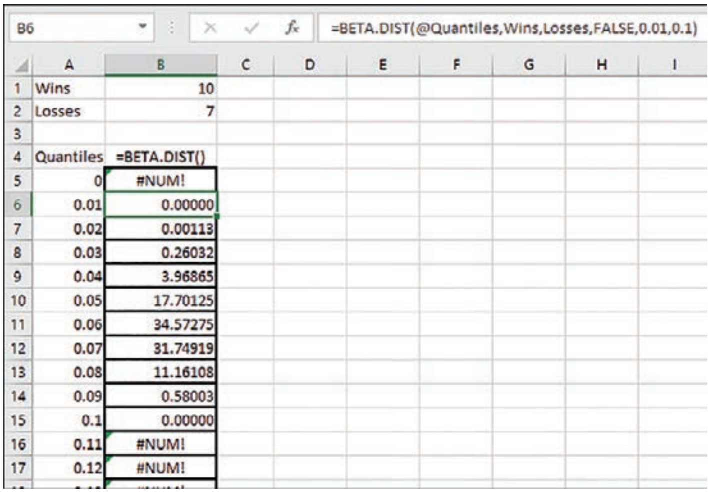
Excel's A and B arguments (the fifth and sixth arguments to the `BETA.DIST` function) can also seem a little mysterious at first. They are optional, and you can use them to restrict the analysis to a subset of quantiles. For example, consider the analysis in Figure 3.5. The formulas in column B are

```
=BETA.DIST(@Quantiles,Wins,Losses,FALSE)
```

Notice that the A and B arguments are missing; in that case, they default to 0.0 and 1.0. To restrict the analysis to the quantiles from .01 to .10, use these formulas on the same rows as the quantiles of interest:

```
=BETA.DIST(@Quantiles,Wins,Losses,FALSE,0.01,0.10)
```

Excel returns the #NUM! error value whenever the associated quantile is outside the range bracketed by the A and B arguments (see Figure 3.6).



B6 =BETA.DIST(@Quantiles,Wins,Losses,FALSE,0.01,0.1)

	A	B
1	Wins	10
2	Losses	7
3		
4	Quantiles	=BETA.DIST()
5	0	#NUM!
6	0.01	0.00000
7	0.02	0.00113
8	0.03	0.26032
9	0.04	3.96865
10	0.05	17.70125
11	0.06	34.57275
12	0.07	31.74919
13	0.08	11.16108
14	0.09	0.58003
15	0.1	0.00000
16	0.11	#NUM!
17	0.12	#NUM!

Figure 3.6
This analysis has been deliberately limited to the first few quantiles.

Replicating the Analysis in R

R has several functions that pertain to the beta distribution, analogous to those that pertain to the binomial distribution discussed in Chapter 2. The beta distribution functions in R include `dbeta`, `pbeta`, `qbeta`, and `rbeta`.

Using R, you can enter many statements directly in the Console window, where they are executed immediately. Sometimes, though, R will wait to do anything until you have finished a multi-line entry, such as a `For` loop. So, you might decide to enter your code in R's script window. From there, you can choose to execute your code one line at a time or the entire set of commands immediately.

To make that choice, take one of these steps:

- To execute one or more lines of code, click the line (or select the lines) in the script window and choose Edit, Run Line Or Selection.
- To execute all the lines of code in the script window, begin by making sure that the script window is active. (That's to ensure that the necessary commands are available in R's menu.) Then choose Run All. You needn't start by selecting the entire set of statements or the first statement in a block of code.

Understanding `dbeta`

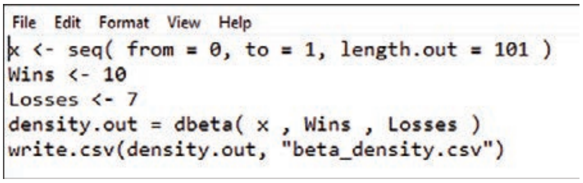
The `dbeta` function returns the probability density function (PDF) for the quantile of a continuous variable. As such, it is analogous to the `dbinom` function, which is used with a discrete variable to return the probability mass function (PMF). The characteristic that most clearly distinguishes the PMF from the PDF is that it's necessary to use integration with a PDF to measure the difference in probability between two quantiles. Nothing more sophisticated than middle-school arithmetic is needed to quantify the difference in probability between two quantiles on a discrete scale.

The argument list for `dbeta` is similar to that for `dbinom`. From Chapter 2:

```
dbinom(x, size, prob, log = FALSE)
```

where *x* is a quantile such as 0.1666, distinguishable and discrete as a side on a six-sided die, *size* is some number of trials, *prob* is the theoretical probability for a single trial (say, 50% for a coin flip or 16.67% for the roll of a single die), and *log*, if FALSE or omitted, returns the resulting number of successes in *size* trials, or the log of that number if TRUE.

You could then enter the following commands into R's script window. When you start R, the console window appears. Choose File, New Script to display a fresh script window, and enter the following code there (see Figure 3.7).

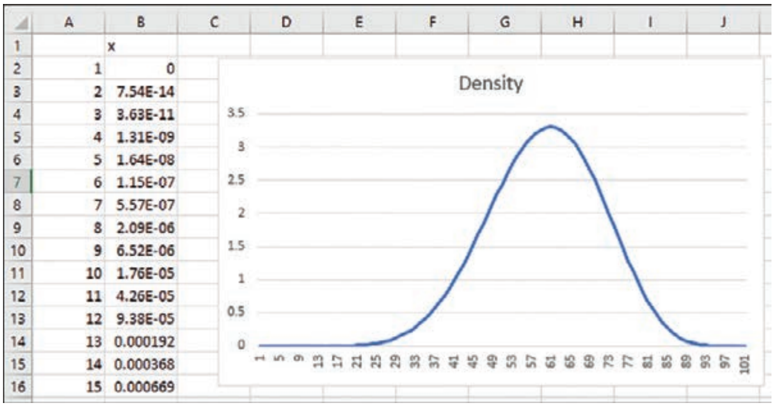


```
x <- seq( from = 0, to = 1, length.out = 101 )
Wins <- 10
Losses <- 7
density.out = dbeta( x , Wins , Losses )
write.csv(density.out, "beta_density.csv")
```

Figure 3.7
Choose File, New Script to open a fresh R Editor window.

Click in the R Editor window to make sure it's active, and then choose Run All from R's Edit menu. The commands you typed will be repeated, along with any system messages, in the Console. If all goes well, a new file named beta_density.csv is written to your working directory. It contains the results of the `dbeta` functions. You can open a csv file using Excel (just double-click it), or you can use a text editor such as Notepad to open it.

The file is shown (with a chart added in Excel) in Figure 3.8.



	A	B
1		x
2	1	0
3	2	7.54E-14
4	3	3.63E-11
5	4	1.31E-09
6	5	1.64E-08
7	6	1.15E-07
8	7	5.57E-07
9	8	2.09E-06
10	9	6.52E-06
11	10	1.76E-05
12	11	4.26E-05
13	12	9.38E-05
14	13	0.000192
15	14	0.000368
16	15	0.000669

Figure 3.8
I used Excel's Text To Columns command to convert R's output in the beta_density.csv file to figures that could be charted.

Understanding `pbeta`

You may recall from Chapter 2 that you can get the *cumulative* binomial distribution for a discrete variable by using the `pbinom` function. Let's revisit an example from Chapter 2. Begin by opening a fresh script window and entering into it these commands:

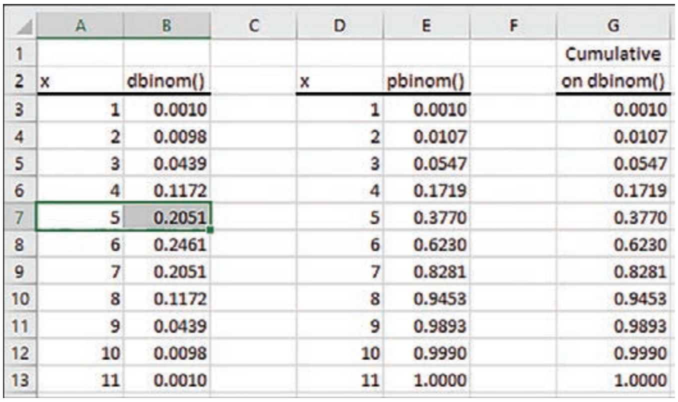
```
successes = seq(0, 10, by = 1)
probabilities = dbinom(successes, size=10,.5)
write.csv(probabilities,"dbinom_out.csv")
```

Then, choose Edit, Run All to execute those commands. A new csv file is written to your working directory.

Now edit your code in the script window so that it reads as follows (I've shown the two changes in boldface):

```
successes = seq(0, 10, by = 1)
probabilities = pbinom(successes, size=10,.5)
write.csv(probabilities,"pbinom_out.csv")
```

If you open the two csv files, dbinom_out.csv and pbinom_out.csv, you'll see what's shown in Figure 3.9. (I have combined the two result files into one worksheet to make it easier to compare them in one figure.) The results of using R's `dbinom` function are in the range B3:B13. They show the expected proportion of the trials that will have the same number of successes as the associated quantile in A3:A13. So, for example, you would expect that 0.2051 of the trials to have five occurrences of events with a 50% probability (Figure 3.9, A7:B7).



	A	B	C	D	E	F	G
1							Cumulative
2	x	dbinom()		x	pbinom()		on dbinom()
3	1	0.0010		1	0.0010		0.0010
4	2	0.0098		2	0.0107		0.0107
5	3	0.0439		3	0.0547		0.0547
6	4	0.1172		4	0.1719		0.1719
7	5	0.2051		5	0.3770		0.3770
8	6	0.2461		6	0.6230		0.6230
9	7	0.2051		7	0.8281		0.8281
10	8	0.1172		8	0.9453		0.9453
11	9	0.0439		9	0.9893		0.9893
12	10	0.0098		10	0.9990		0.9990
13	11	0.0010		11	1.0000		1.0000

Figure 3.9
Differences between `dbinom` and `pbinom`.

The results of using R's `pbinom` function are in the range E3:E13. The values there are cumulative sums: in other words, the value 0.1719 in cell E6 is the sum of the first four quantiles, from x = 1 to x = 4 in B3:B6.

You can verify this easily enough by examining the cells in the range G3:G13 in Figure 3.9. There, the cumulative probability sums are obtained by adding the prior sum to the probability of the current quantile. So, the formula =G5+B6 in cell G6 is the prior sum in G5 plus the probability for quantile 4 in cell B6. You can save yourself the trouble of accumulating probability totals, as in column G, by using `pbinom` to begin with, before leaving R.

That said, you might use an analogous procedure—`pbeta` rather than `pbinom`—if you were working with a continuous variable instead of a discrete variable. Have a look at Figure 3.10.

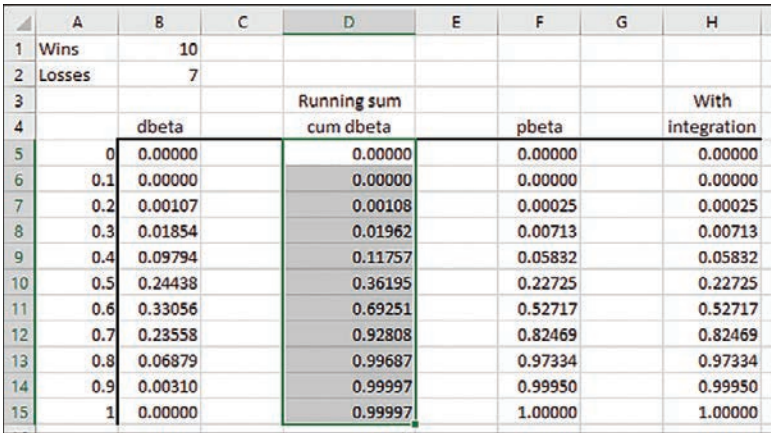


Figure 3.10
To get a cumulative sum of probabilities using a continuous—not a discrete—variable, use integration instead of simple addition.

	A	B	C	D	E	F	G	H
1	Wins	10						
2	Losses	7						
3				Running sum				With
4		dbeta		cum dbeta		pbeta		integration
5	0	0.00000		0.00000		0.00000		0.00000
6	0.1	0.00000		0.00000		0.00000		0.00000
7	0.2	0.00107		0.00108		0.00025		0.00025
8	0.3	0.01854		0.01962		0.00713		0.00713
9	0.4	0.09794		0.11757		0.05832		0.05832
10	0.5	0.24438		0.36195		0.22725		0.22725
11	0.6	0.33056		0.69251		0.52717		0.52717
12	0.7	0.23558		0.92808		0.82469		0.82469
13	0.8	0.06879		0.99687		0.97334		0.97334
14	0.9	0.00310		0.99997		0.99950		0.99950
15	1	0.00000		0.99997		1.00000		1.00000

In Figure 3.10, the range B5:B15 shows the breakdown of the total probability of 17 events (ten successes and seven failures) according to R's `dbeta` function. The range D5:D15 contains the sum of the probabilities in previous quantiles plus the probability of the current quantile—sometimes termed a *running sum*. This is just how we calculated the cumulative probabilities for a discrete variable, using `dbinom` and `pbinom` in Figure 3.9.

The problem is that the running sums in D5:D15 don't match the results of `pbeta` in F5:F15. But the documentation says that `pbeta` returns the cumulative probability density values.

And so it does. What the running sum approach to accumulation omits is all those quantiles in between the ten used in the present example. Bear in mind that you can divide a continuous variable into a theoretically infinite number of values. Each quantile that we add to the analysis makes the running sum a little more accurate, but the arithmetic can never make it a perfectly accurate accumulation. There's always another quantile you can add to a continuum of an infinite number of values. It's a variation of Achilles Paradox.

The `pbeta` function in R is highly accurate (no calculus is perfectly accurate), and the running sum method starts as fairly inaccurate but becomes more accurate as more quantiles are added to the problem. To demonstrate this, you can run a simple set of statements in R,