Rattus binomialis

MATLAB concepts covered:

1. 'rand'
2. simulations for answering questions of probability
3. 'for' loops
4. indexing arrays
5. 'tic' and 'toc' to time code execution
6. binomial distribution
7. pdf's and cdf's

*See corresponding Matlab code in **BinoRat.m** for help on each step.*

A rat is performing a "2-alternative forced choice" (2AFC) task in which it must identify an odor presented at a central port. If it detects odor 'A' it should choose the right-hand port for a reward; if it detects odor 'B' is should choose the other port.

See movie: uchida-mainen-rat_odor_discrimination.mov

The rat in the movie clip is highly trained and gets it right almost every time. But if we go back in time to the early training period, we would find that the rat seems to get it wrong as often as he gets it right. So you decide to do a test and keep track of his correct rate for a block of 50 trials. After 50 trials, we see that the rat has gotten 31 trials correct (19 trials wrong) for an average of 62% correct. **You want to know if the rat has learned the task or if he is still guessing.**

Let's say that we know nothing about probability theory or the binomial distribution, so we will just use the brute force power of a simple simulation.

What is the null hypothesis?

Since there are only two possible answers, this corresponds to a correct probability of 0.5, which is identical to the rate at which tossing a "fair" coin should produce a "tail" (or a "head"). So our previous question can be put as: **how often would one expect to get 31 (or more) tails out of 50 coin tosses**?

Why is the parenthetical "or more" added to the previous sentence?

Well, we're not really interested in the probability of the rat getting *exactly* 31 of 50 trials correct. We're looking for conditions under which we would *reject* the null hypothesis of guessing. So we would clearly count any performance that was better than 31 as well. To answer with a real coin would take a long time, but computers are fast, so we can let the computer do the flipping. How?

Step 1: Write a single line of MATLAB code to simulate a coin toss. It should return a '1' for heads and a '0' for tails.

How many ways can you think of to do this?

See Step 1 in BinoRat.m for help

To address our particular question, we need to find the number of occurrences of tails in runs of 50 tosses.

How might we do this?

Step 2: Write a 'for' loop to simulate 50 coin tosses and store the results in an array called 'tosses'.

We want to know *how often* a certain number of tails is likely to occur in a run of 50 tosses, so we need to repeat the coin flip 50 times. Then we can ask whether, in those 50 tosses, we observed 31 or more tails. One way to solve this problem is by writing a 'for' loop.

What is a 'for' loop? See Step 2 in BinoRat.m for help

Let's start by initializing some variables.

What types of variables will we need for this simulation?

Thus far, you've learned how to "grow" arrays within 'for' loops, but it's better form to have your arrays pre-declared, provided it's predictable in advance how many values you will be generating. In this case it is, so declare variables to hold the results of your simulations:

```
tosses = zeros(n,1); % results of individual coin tosses on one simulation
```

Why did we initialize this variable with zeros?

So, now we can write the 'for' loop to simulate 50 coin flips. See Step 2 in BinoRat.m for help

Step 3: Code a pair of nested 'for' loops that will do *nsim* simulations of 50 coin tosses and store the resulting number of heads on each simulation in *results*.

Step 2 had us simulate flipping a coin 50 times. But we want to flip a coin 50 times *lots of times* so that we can generate an understanding of how likely we are to turn up 31 tails.

How can we simulate 50 coin flips 1000 times? See Step 3 in BinoRat.m for help

What additional variables will we need?

Now we can ask how often a given result (31 or more tails) happens in 1000 simulations. To do that, we will write a nested for loop. See Step 3 in BinoRat.m for help

If you run your nested for loops again, do you get the same answer? How can you get a progressively "better" answer?

Step 4: Write a script that will do the same thing without 'for' loops.

Because MATLAB is a vector-based language, we can often do things much more efficiently using indexing and operators (such as '>') that work across entire arrays or functions that work along a specific dimension, such as down columns.

> *How might you solve this problem without relying on for loops?*
> See Step 4 in BinoRat.m for help

This code will be much more satisfying code (fewer lines!), and it will execute much more quickly. You can test the speed using MATLAB's 'tic' and 'toc' functions. You execute a 'tic' at the beginning of your code, and then a 'toc' at the end, and you'll get the amount of elapsed time. Try this for your loops in Step 3 and compare it to the code in Step 4. See Step 4 in BinoRat.m for help

Step 5: Using a built in binomial function, write code that will give the probability of getting 5 or fewer heads in 10 tosses.

You might vaguely recall that events like coin tosses, where there are only two possible outcomes, are described by the binomial distribution[1]. Based on your knowledge of MATLAB, you might expect that there is an existing function that will make our task easier. Remembering only this key word, 'binomial', find some useful sounding functions.

> *Using one of these functions, what is the probability of getting (exactly) 5 heads from 10 tosses? Is there a single built-in function that can do this for us?*
> See Step 5 in BinoRat.m for help
>
> *What are cumulative distribution functions (CDFs) and probability density functions (PDFs)?[2] Which function would be appropriate to use to determine the probability of getting 5 or fewer heads from ten tosses?*

Step 6: Calculate the probability that the rat is guessing using 'binocdf'.

Finally, let's return to our rat question. Remember, we want to know the probability of getting 31 tails or more out of 50 tosses.

> *How can we use binopdf to solve this question?*
> *How can we use binocdf to solve this question?* See Step 6 in BinoRat.m for help

OK, so what do we think? Is the rat "getting it" or is he just guessing? How many would he need to get right (out of 50) before you believed it?

> *How can we use the above functions to answer this?*

EXTRA Step 7: Visualize the results from your simulation

> *Use the array named 'results' from Step 3 and plot its values.*
> *Now, calculate the mean, standard deviation and standard error of 'results'. How do these values change as a function of the number of simulations?*
> *Remake a histogram plot of 'results' but change the Y-axis values to percent occurrence rather than count number.*

EXTRA Step 8: What if the distribution we cared about wasn't a binomial?

What other functions does Matlab make available? Inspect how they are distinct.
Find the relationship between a poisson distribution's mean and its variance.
Is there an easy way to generate random numbers from non-uniform distributions? Exponential? Normal? Plot them and compare them to the output of rand.
When would these other distribution types be useful?

EXTRA Step 9: Compute the 95% upper and lower confidence bounds for:

a) 30 correct out of 50
b) 60 of 100
c) 300 of 500
Is there a MATLAB function that does this auto-magically?

See Step 10 in BinoRat.m for a solution.

Footnotes

[1] The **binomial distribution** is the discrete probability distribution of the number of successes in a sequence of *n* independent yes/no experiments, each of which yields success with probability *p*.

[2] In probability theory, a **probability density function** (**pdf**), or **density** of a continuous random variable is a function that describes the relative likelihood for this random variable to occur at a given point. The probability for the random variable to fall within a particular region is given by the integral of this variable's density over the region. The probability density function is nonnegative everywhere, and its integral over the entire space is equal to one.

Likewise, the **cumulative distribution function** (**cdf**), describes the probability that a real-valued random variable *X* with a given probability distribution will be found at a value *less than or equal to x*. Intuitively, it is the "*area so far*" function of the probability distribution. Because the area under the pdf must sum to one, we can get the *"area beyond"* as: 1 – cdf(x).

Revisions:
RTB wrote it, 23 April 2011 as "binodemo.m"
RTB revised it, 23 May 2011 as "binorat.m"
DAR updated it, 4 August 2011 as two files. This pdf and corresponding "binorat.m"
JW updated and reformatted on 8 August 2011
RTB added exercise 10 (number picking), 27 May 2012, saved as "BinoRat.m"
RTB separated the four-choice probability exercise into a separate module ("Four-choice probability.doc"), 20 May 2014.