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

RTB wrote it, 23 April 2011 as "binodemo.m"
RTB revised it, 23 May 2011 as "binorat.m"

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. Is the rat learning the task or might he still be guessing?

Let's say that we know nothing about probability theory or the binomial distribution and 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 "head" (or a "tail", take your pick). So our previous question can be put as how often one would expect to get 31 (or more) heads 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?

Ex. 1: Write a single line of MATLAB code to simulate a coin toss.

How many ways can you think of to do this?

To analyze our particular case, we need to ask questions about the occurrence of heads in runs of 50 tosses.

Ex. 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 heads is likely to occur in a run of 50, so we need to repeat the simulation many times. Then we can ask how often a given result (31 or more heads) happens.

Start by initializing some variables:

```
p = 0.5;        % our "null hypothesis" probability
n = 50;         % the number of coin tosses in one simulation
x = 31;         % the number of "tails" we obtained in real life
nsim = 1000;    % the number of simulations we're going to run
```

Mike has been allowing you 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
results = zeros(nsim,1);      % results of each simulation
```

Why did we initialize the variables with zeros?

Ex. 3: Code up 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*.

What is the probability of getting 31 or more heads from 50 coin tosses? If you run this simulation again, do you get the same answer? How can you get a progressively "better" answer?

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.

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

This is much more satisfying code (fewer lines!), and it would 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 it!

You might vaguely recall that events like coin tosses, where there are only two possible outcomes, are described by the binomial distribution. 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?

Ex. 5: Using 'binopdf', write code that will give the probability of getting 5 or fewer heads.

Is there a single built-in function that can do this for us?

This is called the "cumulative distribution function" and for binomial data, we would use 'binocdf'. What is the equivalent cdf command for the pdf version of the code above?

One of the wonderful things about MATLAB's statistics toolbox is that it provides pdf's and cdf's for just about any statistical function you might imagine.

```
lookfor 'cumulative distribution'
```

Finally, let's return to our rat question.

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

Ex. 7: 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?