APPENDIX V

The probability that a single underlying binomial distribution yields two given success proportions: A hypothesis test

The problem

Binomial distributions are ubiquitous. Examples include a coin toss and questions such as, “Does a locust spend more time in the air or in the odor compartment of a Y-maze?” Whenever two such experiments are performed under different conditions, the question arises: Is the difference in the proportion of success observed in two experiments significant? i.e. What is the probability that a difference at least as large as that between the results of two experiments could be produced by chance if the two conditions have the same underlying success probability? For example, does starving the locusts cause a significant change in their chemoattractant behavior?

Notation

Let us call the number of observations in the first experiment *n1* and the number of observations in experiment 2 *n2*. Let us also call the fraction of successes in each experiment *f1* and *f2*. Finally, let us call the success probabilities of the two underlying distributions *p1* and *p2*. What we wish to calculate is the probability that if the null hypothesis was true, i.e. that both experiments have the same underlying success probability, two experiments with n1 & n2 trials, respectively, would yield success proportions as different as those observed or more.

Alternatives

The usual methods to address this question include the normal approximation to the binomial distribution, which is approximately valid only when the number of observations is significantly greater than 30, and assuming that p1=p2=f1, which is approximately valid only when n1>>n2. But today’s computational technology make approximations an unnecessary compromise.

I present a solution to the problem that can be calculated to arbitrary precision. The probability of getting the empirical results or more extreme results given *a* probability for a common underlying distribution is calculated exactly. The only approximation involved in the test below is in the discretization of such success probabilities used to partition the 0-1 interval to integrate over all possible underlying probabilities. This approximation can be carried to arbitrary precision (given by the *precision* parameter below).

The exact hypothesis test

Given an underlying success probability p0 and the success fractions of two experiments, f1 and f2, we want the probability of obtaining a difference between the success fractions of two experiments that is equal or greater than that observed, namely greater than |f1-f2|. This difference, calculated by the MATLAB function BINODIFCDF below, is given by summing the probabilities of obtaining each of the results of the two binomial experiments which yield a difference between success fractions equal or larger than |f1-f2|. The probability of each result can be calculated with the standard binomial formula:

p(f1 | p=p0) = p0^x . q^y . (n1 choose x). (1)

where x = # of successes and y = # of failures, given an underlying success probability p0.

The probability of two success fractions f1 and f2 given an underlying success probability p0 is given by

p(f1 & f2 | p=p0) = p(f1 | p=p0) . p(f2 | p=p0). (2)

Since we don't know what the underlying success probability is, we need to calculate the above for every possible p0 value (the 0-1 interval is discretized to a desired accuracy). The process above is then repeated for all p0 values ranging from 0 to 1, and the results integrated. But all values for p0 do not have equal probability given f1 and f2. The weight of each, i.e. the probability that the underlying success probability is p0, given that f1 and f2 are the fraction of successes observed in two experiments with N1 and N2 observations is calculated using Bayes' rule (and taking the prior probabilities to be uniform) as the probability of obtaining f1 and f2 given p0:

p(p=p0 | f1 & f2) = p(f1 & f2 | p=p0) (3)

which can in turn be calculated from (2) above.

MATLAB implementation

An electronic copy of the following code is available at http://www.its.caltech.edu/~alex/code/binomialtest.htm:

```
function [realpval] = binomialanal(p1,p2,N1,N2,precision)
% © Alex Bäcker Aug 01
% NS1 & NS2 (integers) are the two numbers of successes being compared to see if they can come from the same underlying distribution.   They represent the two 'conditions'.
% N1 & N2 are the # of experiments for p1 & p2, respectively. Default N2=N1.
%
% REALPVAL returns the p-value for the null hypothesis that both experiments were generated by the same underlying
```

```
distribution.
%
% Note that REALPVAL is not only a function of abs(p1-p2), N1 and N2, but rather also of p1 & p2, because the relative probabilities of different underlying success probabilities are different for different p1 & p2 pairs even for constant difference: More extreme values of p1 & p2 indicate more consistent processes with smaller variances, and thus the probability of observing the same difference by chance gets smaller as p1 & p2 get away from 0.5
%
% Takes p(underlying prob==X | p1 & p2) = p(p1 & p2 | underlying prob==X)

if nargin<4,
   N2=N1;
end
if nargin<5,
   precision=.1;
end

dif=abs(p2-p1); % prob difference observed

k=0;
for m=0:precision:1, % These are continuous, not discrete, because they are the true underlying distribution, which is independent of N1 & N2
   k=k+1;
   p12=m; % p of getting a head:

   % Prob of getting a larger difference of p's than observed given underlying success prob is p12:
   pvalue(k)=binodifcdf(dif,p12,N1,N2);

   % Prob of underlying success prob being p12 given p1 & p2 observed:
   pp1= binopdfab(p1,N1,p12); % If you use binopdf & binocdf, you get NaN when using p12=0 &/or 1
   pp2= binopdfab(p2,N2,p12);
   pp(k)=pp1*pp2;

end
totp=sum(pp);
realpval=sum(pp.*pvalue)/totp;
```

function pvalue=binodifcdf(dif,p,N1,N2)

```
% BINODIFCDF - Binomial difference cumulative distribution
% pvalue=binodifcdf(dif,p,N1,N2)
% © Alex Bäcker Aug 01
% Yields the cumulative probability distribution that the difference b/w 2 # of heads  is greater than or equal to DIF
% if the underlying probability is P12 and N1 & N2 are the # of observations in each experiment
% Default N2 = N1

if nargin<4,
   N2=N1;
end
if N2<N1, % Ensure N2>=N1
   [N1,N2]=swap(N1,N2);
```

```
end

pvalue=0;
for Nh1=0:min(N1,N2-dif), % Nh1<=Nh2
   for d=dif:N2-Nh1,
      Nh2=Nh1+d;
      pv=binopdfab(Nh1,N1,p)*binopdfab(Nh2,N2,p);
      pvalue=pvalue+pv;
  end
end

if dif==0, % To avoid repeating the situation Nh1=Nh2 above and below, do not count it again. Equiv: dif=max(dif,1);
   dif=dif+1;
end
for Nh2=0:N1-dif, % Nh2<Nh1
   for d=dif:N1-Nh2,
      Nh1=Nh2+d;
      pv=binopdfab(Nh1,N1,p)*binopdfab(Nh2,N2,p);
      pvalue=pvalue+pv;
  end
end
```

```
function y = binopdfab(x,n,p)
% BINOPDFab Binomial probability density function.
% © Alex Bäcker Aug 01
%   Y = binopdfab(X,N,P) returns the binomial probability density
%   function with parameters N and P at the values in X.
%   Note that the density function is zero unless X is an integer.
%
%   The size of Y is the common size of the input arguments. A scalar input
%   functions as a constant matrix of the same size as the other inputs.
%
%   The Mathwork's BINOPDF can give warning Log of zero if X=N, this one does not.

% Initialize Y to zero.
y = zeros(size(x));

% Binomial distribution is defined on positive integers less than N.
q=1-p;
ix=n-x;
k = find(x >= 0  &  x == round(x)  &  x <= n);
if any(k),
   for i=1:length(k),
      y(k(i)) = p(k(i)).^x(k(i)).*q(k(i)).^ix(k(i)).*nchoosek(n(k(i)),x(k(i)));
   end
end

k1 = find(n < 0 | p < 0 | p > 1 | round(n) ~= n);
if any(k1)
  tmp = NaN;
  y(k1) = tmp(ones(size(k1)));
end
```

Acknowledgments

Thanks to Natalia Caporale for bringing the problem that motivated this appendix to my attention.