Psychophysics and Noninvasive Methods: MATLAB SDT exercises

Felix A. Wichmann & David H. Janssen

Summer Term 2014
Graduate Schools of Neural Information Processing and
Neural & Behavioural Sciences

2nd of July 2015

Throughout the manuscript we have tried to adhere to the following convention:
Built-in MATLAB functions are typeset in orange as follows: `cdf()` .
Solutions to the programming exercises are set in blue: `SDT()` .
You find all of the exercise functions uploaded on ILIAS. Please try and programme the functions first before looking at the solutions/files on ILIAS.

1 Computing d' and λ in Yes-No

Write a function named `SDT` that takes the *number* of hits, false alarms, misses, and correct rejections as input and returns d' and λ. Check your routine using the values you calculated with the z-table you were supplied with on ILIAS.
Hint: you can use the function `icdf()` to compute the inverse cumulative Gaussian. What you are required to programme is simply a MATLAB implementation of the steps used to solve for d' and λ after you have observed hits, false alarms, misses and correct rejections after a Yes-No experiment.

2 Simulating an observer doing a Yes-No task

Write a function named `simYesNo` that simulates an observer in a Yes-No task. The function takes the following inputs: the probabilities of hits and false alarms, the number of signal present trials, and the number of noise only trials. The output is four numbers: the simulated numbers of hits, false alarms, misses, and correct rejections.
In order to do this, you should first write yourself a little sub-function named `binom` which simulates the outcome of each individual trial in a Yes-No task as a binary variable that represents whether the subject reported "signal present" (1) or "signal absent" (0). Therefore, the number of outcomes of a series of N independent binary trials taking on value "signal present" with probability p is a binomial variable. To simulate the number of "signal present" trials in a set of N trials given some probability p, we simply count the number of N uniform variables that have values less than p, which will be a binomial variable with parameters N and p. Hint: You can use `rand()` for your function `binom` to generate

N uniformly distributed values between 0 and 1. Now to simulate an observer during a Yes-No experiment, we simply use `binom()` to simulate the number of hits on *nsignal* trials given the probability of a hit, pHit, and the number of false alarms on *nnoise* trials given the probability of a false alarm, pFA. The number of misses and correct rejections simply follow from these values.

3 Bootstrapped confidence intervals for d'

At the end of the exercise you write a function `dPrimeVariability()` to compute bootstrapped 95% confidence intervals for the d' values of five observers example observers (standard Yes-No experiment, probability of a signal trial $p_s = 0.5$):

- Observer 1: hit rate = 0.81 and false alarm rate = 0.19
- Observer 2: hit rate = 0.96 and false alarm rate = 0.50
- Observer 3: hit rate = 0.50 and false alarm rate = 0.04
- Observer 4: hit rate = 0.77 and false alarm rate = 0.23
- Observer 5: hit rate = 0.96 and false alarm rate = 0.42

3.1 Support function no. 1: calcPC.m

First write an easy little function `calcPC()` to calculate *percent correct* from the hit and false alarm rates (note that signal and noise trials were equally frequent, $p_s = 0.5 = 1 - p_s$).

What do you find? Think about your results—given the percent correct scores being equal for the second and third, and the last two observers do you think the second and third observer and the last two observers are really equally sensitive to the signal? What about the first one? Is she more sensitive than the second and third? All of them?

3.2 Support function no. 2: SDTboot.m

To estimate the variability of your calculated d' value, use a Monte Carlo approach, i.e. "What would we expect if ... ?". You want to know how variable the distribution of d' values are for your observer, who achieved some hit rate and some false alarm rate. Therefore, simulate a large number of experiments where you assume that the true (generating) hit and false alarm rates are those you observed. For each simulated experiment, you simulate the number of hits and the number of false alarms such an observer would achieve and then compute d'^*. The variability of all d'^*s you simulated is an estimate of the variability of your observed d'. Write, `SDTboot()` , simulating a large number of Yes-No experiments given your subject's observed performance and the number of signal present and signal absent trials. Of course you should re-use your functions `simYesNo()` and `SDT()` .

3.3 Main function: dPrimeVariability.m

Finally you need to write the function `dPrimeVariability.m` looping over different numbers of signal present trials, calling `SDTboot.m` every time. Remember, we assume here that the number of signal absent trials is equal to the number of signal present trials, as is common practice in many SDT-based Yes-No experiments. Because the exercise asks you to evaluate how confidence intervals change size for the five different observers listed above, the outer loop also loops over the observers. Use vectors `pHit` and `pFA` which are simply the observed hit and false alarm rates indexed by observer. Run the function assuming three different experiments, with 100 signal present trials and 100 noise only trials, 1000 signal trials and 1000 noise only trials as well as 10,000 of each. Observe how this changes the size of the confidence intervals. To estimate 95% confidence intervals on the observed d', we simply use the percentile method, where we calculate the 2.5^{th} percentile of the distribution of simulated d' values as the lower bound and the 97.5^{th} percentile as the upper bound of the interval. Hint: you can use the function `prctile()` to obtain percentiles from a sample. The simulations demonstrate the following:

- Clearly, as the number of signal present (and signal absent) trials is increased in the experiment, the size of the confidence intervals you can estimate on the observed d' (and λ) value(s) become smaller and smaller, i.e. your confidence increases.
- Furthermore, it appears that for the first *three* observers d' is actually identically 1.75—thus their different percent correct scores simply reflect different criterion settings.
- The criterion setting has an influence on the size of the confidence intervals for d': The intervals for the first observer (optimal criterion placement at $\lambda = d'/2$, i.e. highest possible percent correct, c.f. with your calculations above) is always smaller than those for observers two and three.
- Observers 4 and 5 had the same percent correct score but are actually very different in their sensitivity: Observer 4 is the least sensitive of all five, observer 5 the most sensitive of all of them!

4 Source code

```
function [dp,lambda] = SDT(Hit,FA,Miss,CR)
% [dp,lambda] = SDT(Hit,FA,Miss,CR)
%   Gaussian Equal Variance Signal Detection Theory
%
% Inputs:
% Hit     the number of hits
% FA      the number of false alarms
% Miss    the number of misses
% CR      the number of correct rejections
%
% Outputs:
% dp      dprime
% lambda  sensory criterion
%

nsignal = Hit+Miss;
nnoise  = FA+CR;

pHit    = Hit/nsignal;
pMiss   = Miss/nsignal;
pFA     = FA/nnoise;
pCR     = CR/nnoise;

lambda  = icdf( 'norm', pCR,   0, 1 );
l_star  = icdf( 'norm', pMiss, 0, 1 );
dp      = lambda - l_star;
end
```

```
function b = binom(N,p)
% b = binom(N,p)
% generate a binomial variable, the number of successes on N trials with
% binary outcomes and probability of success p
%
b = sum( rand(1,N) < p);
end
```

```
function [Hit,FA,Miss,CR] = simYesNo(pHit,pFA,nsignal,nnoise)
% [Hit,FA,Miss,CR] = simYesNo(pHit,pFA,nsignal,nnoise)
%   Simulated Yes-No experiment assuming independent Bernoulli trials
%
% Inputs:
% pHit     the probability of a hit
% pFA      the probability of a false alarm
% nsignal  the total number of signal present trials
% nnoise   the total number of noise only trials
%
% Outputs:
% Hit      the number of hits
% FA       the number of false alarms
% Miss     the number of misses
% CR       the number of correct rejections
%
Hit = binom(nsignal,pHit);
Miss = nsignal-Hit;

FA = binom(nnoise,pFA);
CR = nnoise - FA;
end
```

```
function pc = calcPC(hr,fa,p_s)
% function pc = calcPC(hr,fa,p_s)
%
% Function calculates percent correct from the hit rate (hr) and the false
% alarm rate (fa) and the probability of a signal trial (p_s). If the third
% argument is not specified, the function assumes an equal number of signal
% and noise trials, i.e. p_s = 0.5
if nargin < 3, p_s = 0.5; end

pc = p_s*hr + (1-p_s)*(1-fa);
end
```

```
function [dp,lambda] = SDTboot(pHit,pFA,nsignal,nnoise,BIG)
% [dp,lambda] = SDTboot(pHit,pFA,nsignal,nnoise,BIG)
%   Simulate BIG number of Yes-No experiments and use SDT to
%   compute dp and lambda each time.
%
% Inputs:
% pHit     the probability of a hit
% pFA      the probability of a false alarm
% nsignal  the total number of signal present trials
% nnoise   the total number of noise only trials
% BIG      the number of simulated experiments to run
%
% Outputs:
% dp      a vector of BIG dprime values, one for each experiment
% lambda a vector of BIG sensory criterion values, one for each experiment
%

dp = zeros(1,BIG);
lambda = zeros(1,BIG);
for t = 1:BIG
    [Hit,FA,Miss,CR] = simYesNo(pHit,pFA,nsignal,nnoise);
    [dp(t),lambda(t)] = SDT(Hit,FA,Miss,CR);
end
end
```

```
function [CI_dp, CI_lambda] = dPrimeVariability(pHit, pFA, BIG, nsignal)
% function [CI_dp, CI_lambda] = dPrimeVariability(pHit, pFA, BIG, nsignal)
%
% Function calculating the confidence interval sizes for d-prime and lambda
% depending on the hit and false alarm rates and the number of trials for
% binomial observers during a Yes-No experiment.
% NOTE: The function assumes a balanced design, i.e. it automatically uses
% 'nsignal' for signal *and* noise trials.

% set defaults
if nargin < 1, pHit = [0.81, 0.96, 0.50 0.77, 0.96]; end
if nargin < 2, pFA =  [0.19, 0.50, 0.04 0.23, 0.42]; end
if nargin < 3, BIG = 1000; end % the number of simulations we will run
if nargin < 4, nsignal = [100 1000 10000]; end % the number of signal

% For each observer
for obs = 1:length(pHit)
    % estimate the 95% CI assuming 'nsignal' signal and 'nsignal' noise
    % trials and that there were the same number of noise only trials
    for expt = 1:length(nsignal);
        nnoise = nsignal(expt);
        [dp,lambda] = SDTboot(pHit(obs),pFA(obs),nsignal(expt),nnoise,BIG);
        CI_dp(obs,:,expt) = [prctile(dp,2.5),prctile(dp,97.5)];
        CI_lambda(obs,:,expt) = [prctile(lambda,2.5),prctile(lambda,97.5)];
    end
    disp(['...␣done␣observer␣no.␣' num2str(obs) '␣of␣' ...
        num2str(length(pHit)) '␣...␣'])
end
end
```