This online appendix provides the MATLAB code for the simulation results in the paper.

A Overconfidence through Disagreement on the Expected Value

The following is the MATLAB code that was used to derive the results in Table 1 ‘Hitrate for 90% Confidence Interval in function of the Ratio of Standard Deviations’:

```matlab
clc;
clear all;
delete md1.out;
diary md1.out;

% I assume that
% 1) The beliefs themselves have a normal distribution.
% 2) The means of these beliefs are distributed standard normal. I will
% indicate these means as u and v.

% I will have u and v be randomly drawn (with n * m datapoints) and then
% see how much confidence one places in the other’s 90% interval

% Let \( \text{sig} \) be the std dev of each player’s belief,
% i.e., the player’s uncertainty. (This is measured relative to the
% std dev of the means distribution, which I normalized to 1.)

% Let \( \text{gam} \) denote how many std dev you need (in each direction) to get the 90%
% confidence interval.

n=50
m=50
sigvec = [.5 1 2 3]

for i = 1:4
    sig = sigvec(i,i);
    % The simulation considers four values for the ratio of std dev’s. The
    % ratio here is the inverse of that in the paper.
```

∗Harvard Business School (evandensteen@hbs.edu).
gam = norminv(.95,0,1);
% gam is the point (on the X-axis) at which the std norm cum distri equals 95%
% The 90% confidence interval runs from \gam to the left of 0
% to \gam to the right of 0.

OneMat=ones(n,m);
ZeroMat=zeros(n,m);
U = normrnd(0,1,n,m);
V = normrnd(0,1,n,m);
L = min(U,V)
H = max(U,V)
% Without loss of generality, I will denote the lower one of u and v as "l"
% and the higher one as "h" and work with these

BigSig=sig*OneMat
% This is a matrix with the standard deviation

BigDev=sig*gam*OneMat
% This is a matrix with the \gam interval scaled by the std dev

ValueAll=normcdf(H+BigDev,L,BigSig)-normcdf(H-BigDev,L,BigSig)
% ValueAll calculates how much confidence the L distribution puts
% on the 90% confidence interval of H
% which is [H-BigDev, H+BigDev]

AvgHitRate=mean(mean(ValueAll))
ValueNorm=ValueAll./.9
AvgVal=mean(mean(ValueNorm))

MFinal(i)=AvgHitRate;
MFinalRel(i)=AvgVal;
end

MFinal
MFinalRel

B Endogenously Generated Overconfidence

The following is the MATLAB code that was used to derive Figure 3 ‘The hitrate for 90% confidence
[...] Chi-squared distribution with mean \nu.’:

% This program shows how the hitrate changes in function of the
% number of data points and other parameters.

% The model assumes that all beliefs are mean-zero normal.
% It calculates the hitrate by normalizing the observer player’s (O) variance to
% standard normal. (The reason is that O’s variance is typically the
Note that the hitrate may be larger than 90% in the cases where F is (relatively) underconfident.

The program calculates the 90% confidence interval of the focal player (F). It then calculates the likelihood of that interval according to the standard normal. That is the hitrate of F. The hitrate divided by .9 gives the degree of overconfidence.

```matlab
clc;
clear all;
delete hr1.out;
diary hr1.out;

% Definition of basic parameters
n = 50000;
m = 20;

for k=1:3
    nu = 1.*k; % Chi-squared parameter
    gam = norminv(.95,0,1)
    % gam is the point (on the X-axis) at which the std norm cum distri equals 95%
    % The 90% confidence interval runs from \gam to the left of 0 to \gam
    % to the right of 0.

% Iteration
% I will use F and O for Focal player and Observer
for i=1:m
    HR(k,i)=0; % i is the number of signals (from 1 to m=20)

    for j=1:n
        Sz = random('chi2',nu); % This is the variance for the common prior
        SsF = random('chi2',nu,1,i+1); % These are the var's of the i data points for F
        SsO = random('chi2',nu,1,i+1); % These are the var's of the i data points for O
        SsF(1,1) = Sz(1,1); % The Ss vector has the prior var and all data point var's
        SsO(1,1) = Sz(1,1); % The Ss vector has the prior var and all data point var's
        Ts = 1./SsF; % This is the resulting vector of precisions (i.e., the squared taus)
        VF = (Ts.^2 * SsF')/(Ts * ones(1,i+1)')^2;
        % This if F's variance.
        % Note that Ts[k]/(Ts * ones(1,i+1)') is the weight put on data point k.
        % The contribution to variance must square this (both numerator and denominator).
        VO = (Ts.^2 * SsO')/(Ts * ones(1,i+1)')^2;
        % This is what F's variance should be acc to O
```

A3
Vrat = sqrt(VF./VO);
% For the confidence interval, you need to convert to std dev
Len = gam.*Vrat;
% Len is half the 90% confidence interval of the focal player F
hr = 2.*(normcdf(Len,0,1)-.5);
% FINALLY "hr" is the relative hit-rate

HR(k,i) = HR(k,i)+hr;
end

HR(k,i) = HR(k,i)/n;
% This is now the average hitrate over all n simulations
end

HRb(k,1)=0.9;
% HRb will be the matrix with the curve data. The .9 initializes for no data points.

for i=1:m
    HRb(k,i+1)=HR(k,i);
end
end

% The rest just plots the graphs.
X=linspace(0,m,m+1);
Calline=.9 .* ones(1,m+1)
figure(1)
plot(X, Calline,’--k’,X,HRb(1,1:m+1),’-k’,X,HRb(2,1:m+1),’-k’,X,HRb(3,1:m+1),...’-k’,’LineWidth’,2)
axis([0,m,0,1])
xlabel(‘Number of Observed Data Points (or Signals)’)
ylabel(‘Hitrate for 90% Confidence Interval’)
text(3,.45,’\nu=1’)
text(6,.5,’\nu=2’)
text(11,.55,’\nu=3’)