%%%
%%%%%

Computational Psychiatry Course 2021

METACOGNITION TUTORIAL

%%%
%%%%%

Adapted from O. Faull and S. Fleming

INSTRUCTIONS:

Run through this tutorial at your own pace. It is suggested that you complete steps 1-6 in order, and then you may choose which extension problem you are most interested in to continue with - or you are welcome to complete all of them if you have time.

REQUIRED:

To run Exercises 1-5 you will need to have the folder and subfolders supplied for this tutorial added to your Matlab path. If you would like to run Exercise 6 onwards on your local device, you will need to have JAGS (an MCMC language similar to BUGS) installed on your machine. See here for further details: http://mcmc-jags.sourceforge.net/

NOTE: There are compatibility issues between matjags and JAGS 4.X**. To run the MATLAB code you will need to install JAGS 3.4.0 rather than the latest version.

You can type "path" in the Matlab console to check if the HMeta-d-master toolbox has been added to your path.

You will need the Matlab optimization toolbox installed.

For Extension problem 3, you will also need the Matlab statistical toolbox installed.

----

1) EXPLORING THE DATA: WHAT METACOGNITION LOOKS LIKE

First, we will simulate data with high and low internal noise, which should lead to higher and lower metacognition. This added noise represents a potential loss of information between type 1 task discrimination

and 'metacognitive' information, i.e. type 2 confidence scores. We will then plot what the confidence scores would look like for each of these two scenarios (high and low internal noise).

Set up the parameters:

```
sim1.d = 2; % Set task performance (d')
sim1.c = 0; % Set task bias (c)
sim1.Ntrials = 1000; % Set the number of trials performed
sim1.Nratings = 4; % Choose the rating scale to use
sim1.noise.low = 0; % Set a low value for internal noise
sim1.noise.high = 0.7; % Set a high value for internal noise
```

Simulate the responses for low and high internal noise:

FUNCTION: ss_type2_SDT_sim(d, noise, c, Nratings, Ntrials);

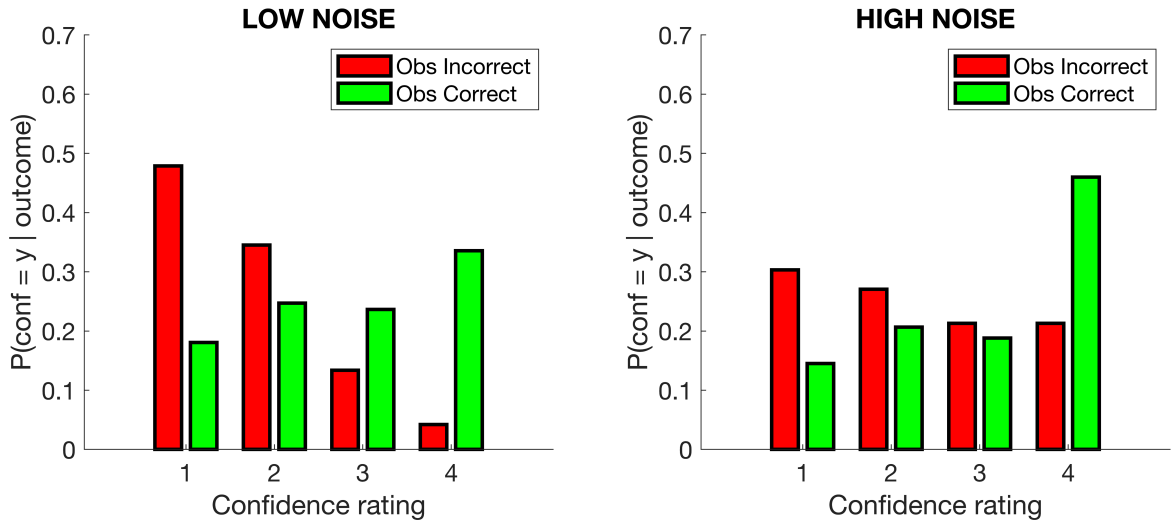
```
sim1.lowNoise.responses = ss_type2_SDT_sim(sim1.d, sim1.noise.low, sim1.c, sim1.Nratings, sim1.
sim1.highNoise.responses = ss_type2_SDT_sim(sim1.d, sim1.noise.high, sim1.c, sim1.Nratings, sim
```

Plot the confidence results for the two simulations:

FUNCTION: ss_plot_confidence(data, title, type)

NOTE: type = 'obs' or 'est' (observed or estimated data from model)

```
figure;
set(gcf, 'Position', [200 200 800 300])
subplot(1,2,1)
ss_plot_confidence(sim1.lowNoise, 'LOW NOISE', 'obs');
subplot(1,2,2)
ss_plot_confidence(sim1.highNoise, 'HIGH NOISE', 'obs');
```



EXPLANATION:

You can see in the figure that the low internal noise likely results in less overlap between the confidence distributions for correct and incorrect responses. It is actually the difference between these two distributions (one for correct responses, one for incorrect responses) that tells us how good metacognition is --> If you have higher confidence scores when you are correct and lower confidence scores when you are incorrect (and not much mixing between the two), the better metacognition will be. Another way of thinking about it is that the less 'noise' that is added as information passes from type 1 performance (e.g. correct vs incorrect perceptions) to type 2 performance (e.g. confidence in the decision that has been made), the less 'mixing' will occur between these two distributions and the larger (or 'better') metacognition will be.

EXERCISE 1:

Run the simulation and plot again a few times. Do you get the same results every time? Why / why not?

You can also change the high noise parameter and see the effect of the separation between the two distribution for correct and incorrect responses.

---

2) EXPLORING THE DATA: HOW DOES PERFORMANCE CHANGE METACOGNITION?

It is important to note that looking at the difference between the confidence distributions is an 'absolute' form of metacognition, which means that it will be affected by task performance (measured by d'). For example, if d' is higher (and the task is easier), there is more type 1 information on which to base metacognitive decisions than if d' is lower. We will explore this here by comparing the confidence scores with higher and lower d'.

Set up the parameters:

```
sim2.dValueHigh = 2; % Set task performance to be higher, with a larger d' value
sim2.dValueLow = 1; % Set task performance to be lower, with a smaller d' value
sim2.c = 0; % Set task bias (c)
sim2.Ntrials = 1000; % Set the number of trials performed
sim2.Nratings = 4; % Choose the rating scale to use
sim2.noise.value = 0.2; % Set a value for the noise
```

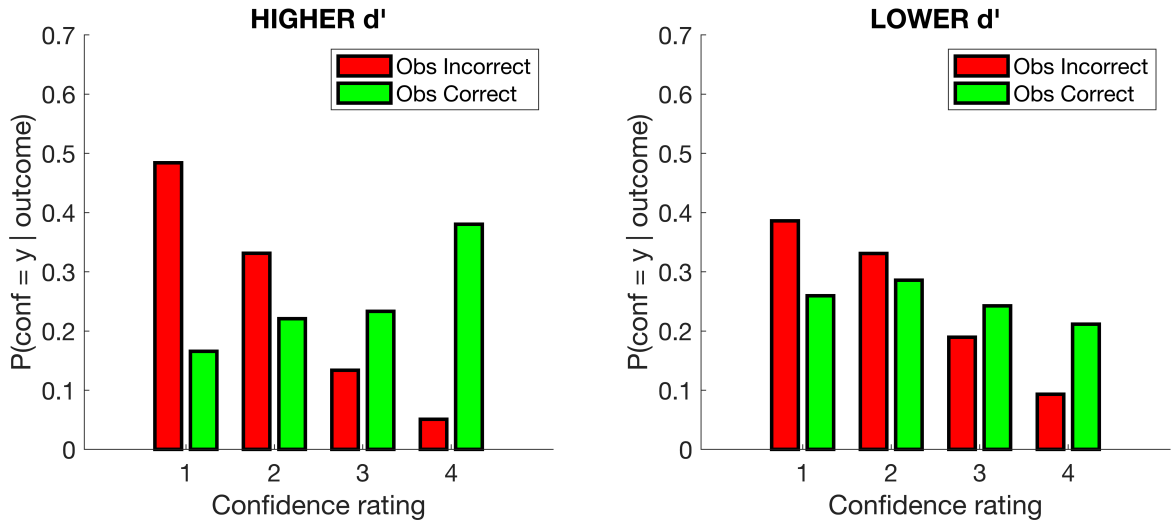
Simulate the responses for low and high task performance:

```
sim2.dHigh.responses = ss_type2_SDT_sim(sim2.dValueHigh, sim2.noise.value, sim2.c, sim2.Nrating
```

```
sim2.dLow.responses = ss_type2_SDT_sim(sim2.dValueLow, sim2.noise.value, sim2.c, sim2.Nratings,
```

Plot the confidence results for the two simulations:

```
figure;
set(gcf, 'Position', [200 200 800 300])
subplot(1,2,1)
ss_plot_confidence(sim2.dHigh, 'HIGHER d'''', 'obs');
subplot(1,2,2)
ss_plot_confidence(sim2.dLow, 'LOWER d'''', 'obs');
```



EXPLANATION:

You can see in this figure that a higher task performance (d') results in a bigger difference between the confidence distributions, despite the same amount of internal noise. Later on we will look at how we can correct for the level of d' to get a 'relative' measure of metacognition.

EXERCISE 2:

Change any of the parameters above and re-run this section to get an idea as to how each parameter may change the observed confidence scores.

---

3) CALCULATING METACOGNITION: THE TYPE2ROC CURVE

Another way of visualising type 2 performance (that will help us towards quantifying metacognition) is to plot the data using a Receiver Operating Characteristic (ROC) curve. Here, we plot the distribution of correct responses against the distribution of incorrect responses. The resulting type2ROC curve tells us how separate these confidence distributions are, and thus the area under the curve can be used as a measure of the level of type 2 performance. We will go through this now.

Set up the parameters:

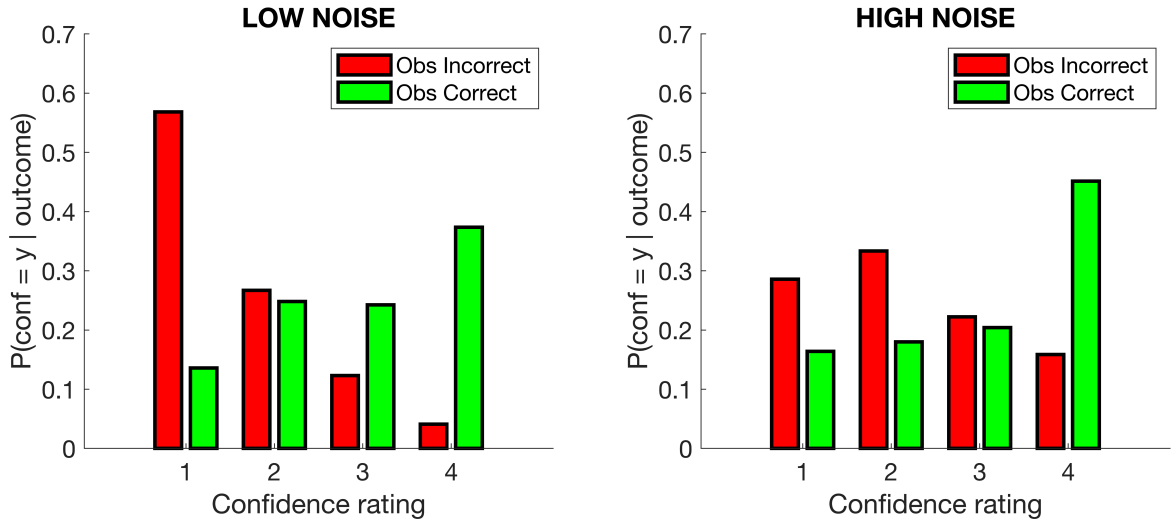
```
sim3.d = 2; % Set task performance(d')
sim3.c = 0; % Set task bias (c)
sim3.Ntrials = 1000; % Set the number of trials performed
sim3.Nratings = 4; % Choose the rating scale to use - any will do
sim3.noise.low = 0; % Set a low value for internal noise
sim3.noise.high = 0.8; % Set a high value for internal noise
```

Simulate the responses for low and high internal noise:

```
sim3.lowNoise.responses = ss_type2_SDT_sim(sim3.d, sim3.noise.low, sim3.c, sim3.Nratings, sim3.
sim3.highNoise.responses = ss_type2_SDT_sim(sim3.d, sim3.noise.high, sim3.c, sim3.Nratings, sim
```

Plot the confidence results for the two simulations:

```
figure;
set(gcf, 'Position', [200 200 800 300])
subplot(1,2,1)
ss_plot_confidence(sim3.lowNoise, 'LOW NOISE', 'obs');
subplot(1,2,2)
ss_plot_confidence(sim3.highNoise, 'HIGH NOISE', 'obs');
```

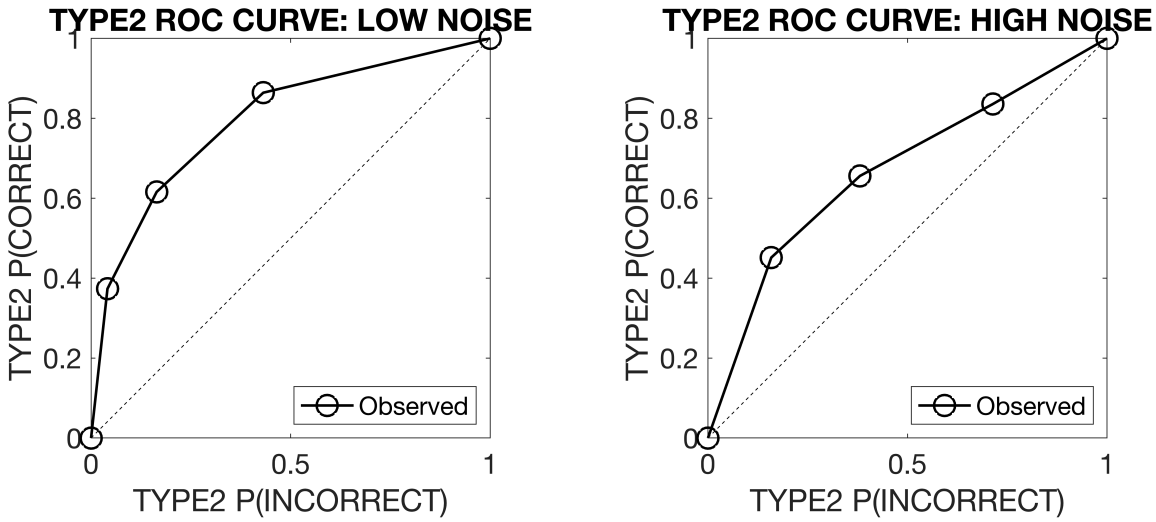


Now plot the observed type2 data in a type2ROC curve:

FUNCTION: ss_plot_type2roc(data, title, type)

NOTE: type = 'obs' or 'est' (observed or estimated data from model)

```
figure;
set(gcf, 'Position', [200 200 800 300])
subplot(1,2,1)
ss_plot_type2roc(sim3.lowNoise, 'LOW NOISE', 'obs');
subplot(1,2,2)
ss_plot_type2roc(sim3.highNoise, 'HIGH NOISE', 'obs');
```



EXPLANATION:

We can now compare the two types of figures (confidence distributions and type2ROC curves) from the same data to see what the type2ROC curve is plotting. The first point above zero on the curve (second from the left) is the proportion of the correct responses that used the highest confidence rating (i.e. 4 on the confidence scale) plotted against the proportion of incorrect responses that used the highest confidence rating. The next point to the right of this is the proportion of the correct responses that used the highest two confidence ratings (i.e. 3 and 4) against the the proportion of incorrect responses that used the highest two confidence ratings. This continues until we reach 1 in the top right corner of the curve, where we have accounted for all of the correct and incorrect responses across all of the confidence ratings. You can see here that when there is lower internal noise, the type2ROC curve has a larger 'bow' - it extends closer towards the top left corner of the graph. This is because there is a greater proportion of the correct responses that use the higher confidence ratings, compared to the proportion of the incorrect responses that used the higher ratings. If metacognition was perfect, and a participant **only** used the higher rating scores when they were correct and the lower rating scores when they were incorrect (and never mixed them), the curve would be a right angle in the top left corner. If metacognition was zero, the curve would be flat and lie on the diagonal dotted line, as there would be approximately the same proportion of correct and incorrect responses no matter what confidence rating was used.

Now we will calculate the area under each of the type2ROC curves:

FUNCTION: ss_calcAU_type2roc(data)

```
low_noise_area = ss_calcAU_type2roc(sim3.lowNoise)
```

```
low_noise_area = 0.7903
```

```
high_noise_area = ss_calcAU_type2roc(sim3.highNoise)
```

```
high_noise_area = 0.6675
```

EXPLANATION + EXERCISE 3:

Now you should be able to see that the area under the type2ROC curve with lower internal noise is larger than when the data are produced with higher internal noise. This metric is often used as an absolute measure of metacognition. We can also check what happens to the area under the type2ROC when we have a higher value of d', and no difference in the internal noise... To do this, try calculating the area under the type2ROC for the data we simulated using different levels of d' from step 2. How does higher d' affect the area under the typeROC?

HINTS:

- You can plot using: ss_plot_type2ROC(sim2.dHigh, 'HIGH d', 'obs')

- You can calculate the area using: ss_calcAU_type2roc(sim2.dHigh)

---

4) FITTING THE MODEL: META-D'?

In this step, we will consider how we can quantify metacognition using meta-d' rather than the area under a type2ROC curve. Because meta-d' is calculated in the same units as d', we can get rid of the effect of type 1 performance (d') by creating a ratio of meta-d' / d', which we call the "Mratio". The Mratio is then a 'relative' measure of metacognition, and is independent of task performance. For full details of the model please refer to the paper by Maniscalco and Lau (2012: "A signal detection theoretic approach for estimating metacognitive sensitivity from confidence ratings").

The key here is actually the direct relationship between type 1 and type 2 performance that we explored in steps 2 and 3, where we can predict what the type2ROC curve *should* look like for a certain type 1 d' and bias (if no information was lost or gained between type 1 and type 2 performance). Therefore, if we take the observed type2ROC curve, we can theoretically estimate what level of d' would cause the observed type 2 performance... And we call this meta-d' (For details on the relationship between type 1 and type 2 performance,

see Galvin et al., 2003: "Type 2 tasks in the theory of signal detectability: Discrimination between correct and incorrect decisions"). Here we will plot the observed and estimated confidence values and type2ROC curves for our simulated data with different levels of metacognition, and take a closer look at the model values for meta-d'.

Set up the parameters:

```
sim4.d = 2; % Set task performance (d')
sim4.c = 0; % Set task bias (c)
sim4.Ntrials = 1000; % Set the number of trials performed
sim4.Nratings = 4; % Choose the rating scale to use
sim4.metad.low = 0.8; % This time we will simulate with values of meta-d'
sim4.metad.high = 1.5; % Set a high value for metacognition
```

Simulate the responses for low and high metacognition:

FUNCTION: ss_metad_sim(d, meta-d', c, Nratings, Ntrials);

```
sim4.lowMetad.responses = ss_metad_sim(sim4.d, sim4.metad.low, sim4.c, sim4.Nratings, sim4.Ntri
sim4.highMetad.responses = ss_metad_sim(sim4.d, sim4.metad.high, sim4.c, sim4.Nratings, sim4.Nt
```

Now we will fit the model for meta-d':

FUNCTION: fit_meta_d_MLE(data.nR_S1, data.nR_S2)

NOTE: nR_S1 and nR_S2 are the counts of the responses to stimulus 1 (nR_S1) and stimulus 2 (nR_S2) for each confidence level, and are generated by ss_metad_sim

```
sim4.lowMetad.fit = fit_meta_d_MLE(sim4.lowMetad.responses.nR_S1, sim4.lowMetad.responses.nR_S2
```

```
Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

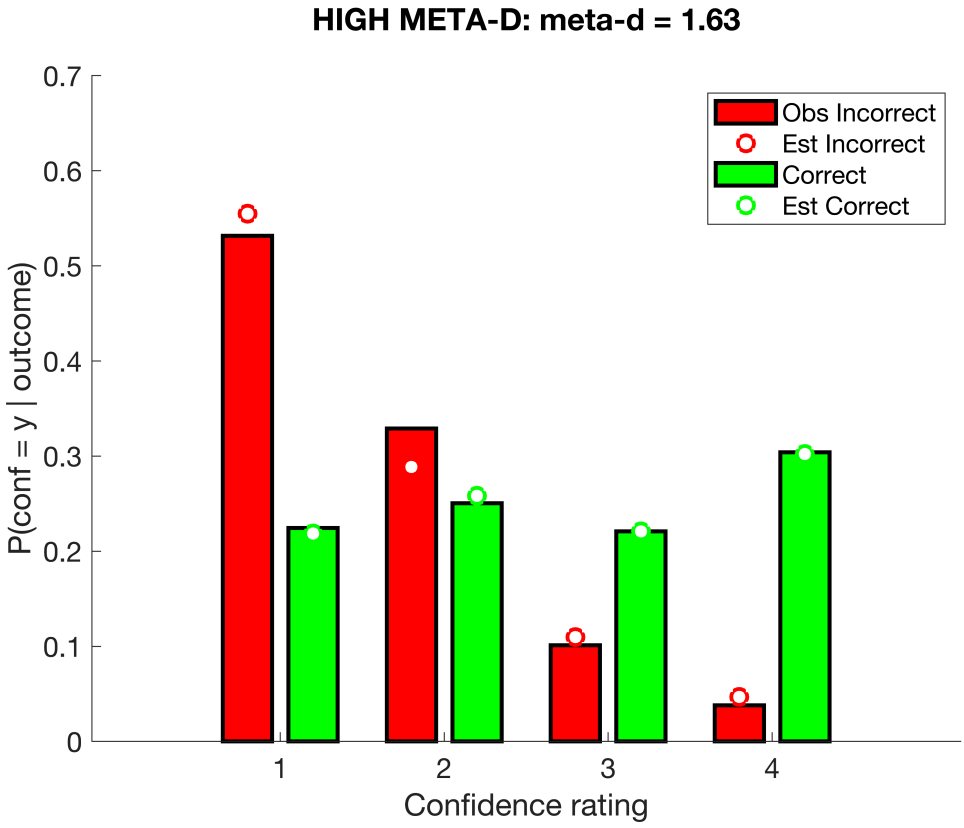
```
sim4.highMetad.fit = fit_meta_d_MLE(sim4.highMetad.responses.nR_S1, sim4.highMetad.responses.nR
```

```
Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

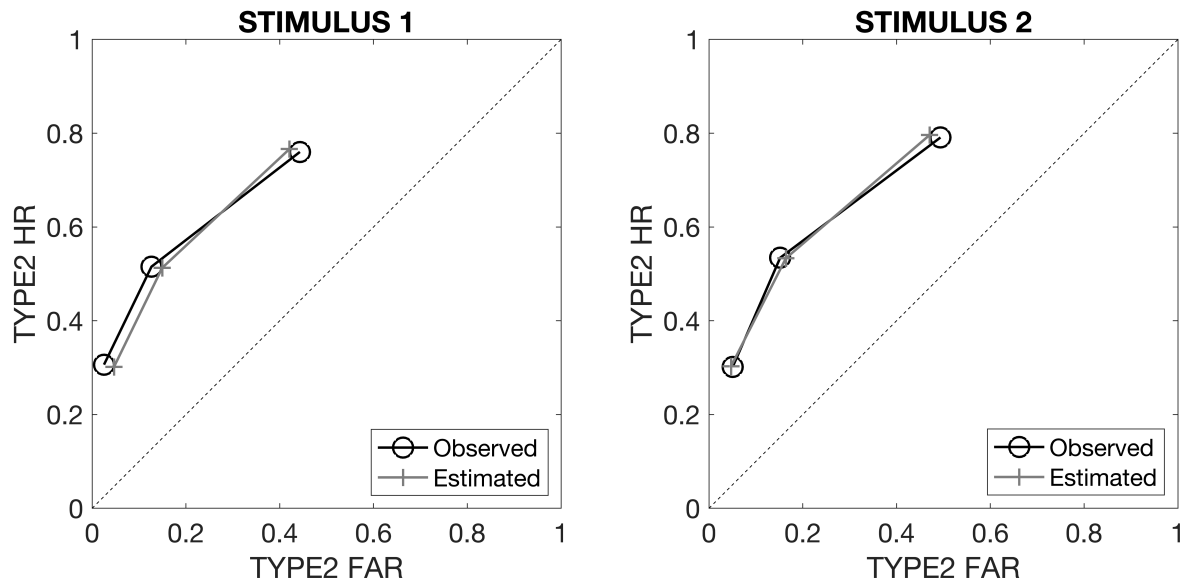
Plot the model estimation of type 2 performance with high metacognition:

--> Use the 'est' flag in these functions this time to plot the estimated model results as well as the real data

```
figure
ss_plot_confidence(sim4.highMetad, 'HIGH META-D', 'est');
```

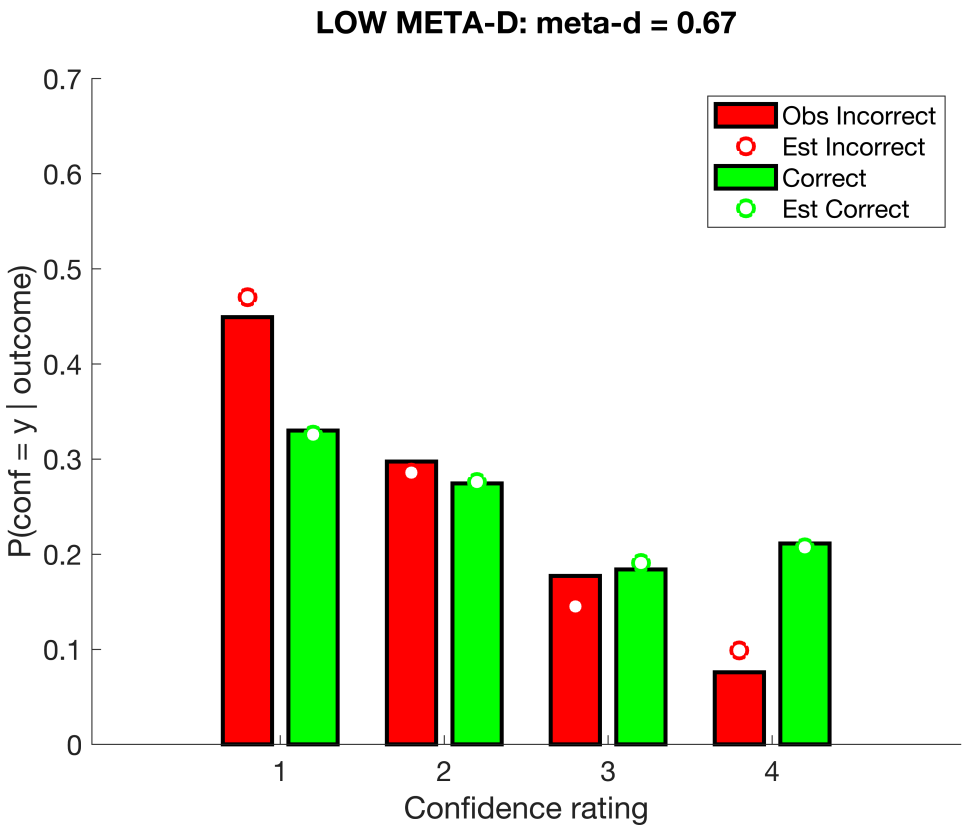



```
ss_plot_type2roc(sim4.highMetad, 'HIGH META-D', 'est');
```

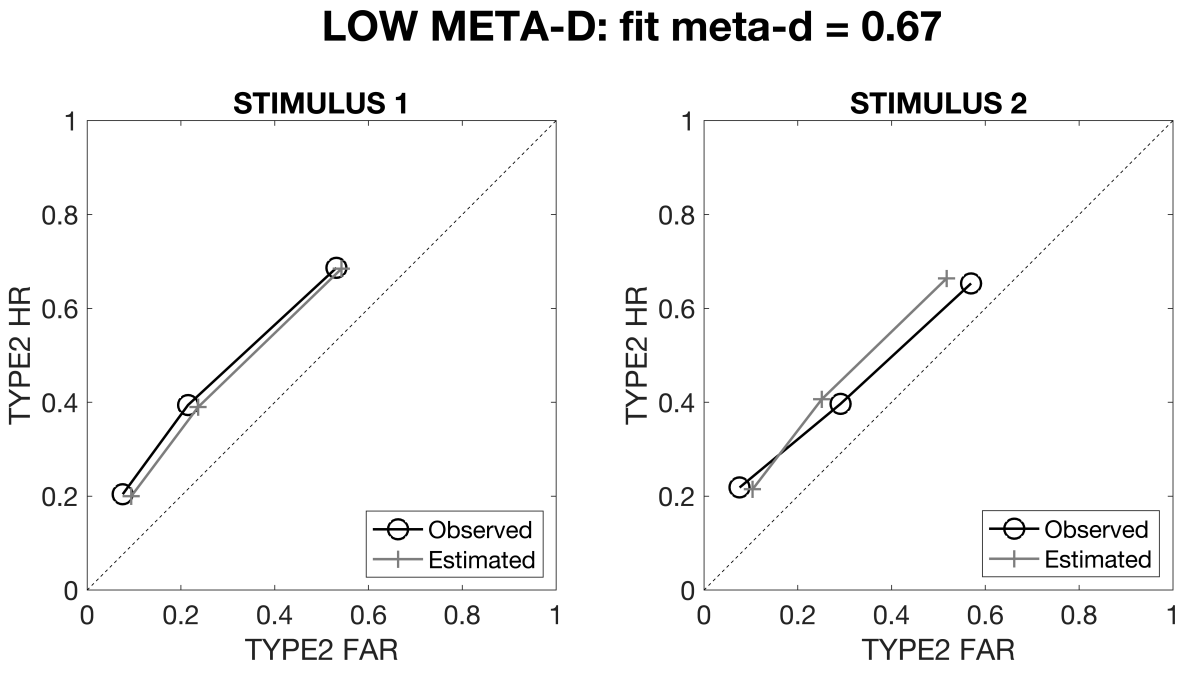


Now plot the same figures for the model estimation of type 2 performance with low metacognition:

```
figure
ss_plot_confidence(sim4.lowMetad, 'LOW META-D', 'est');
```




```
ss_plot_type2roc(sim4.lowMetad, 'LOW META-D', 'est');
```



EXPLANATION:

You can see that the model does a pretty good job estimating the confidence values for correct and incorrect responses, and the estimated type2ROC curves are quite close to the observed values (the model estimates the curves for each of the two stimuli separately). We will now explore some extensions to the model that will help us cope with more difficult data.

EXERCISE 4:

How do the estimated (fit) Mratio values compare to the simulated Mratio values that we used?

HINT: You can find the fitted Mratio im sim4.highMetad.fit, but you will need to calculate the simulated Mratio.

Next, run this whole section again to simulate new data from the same parameters and re-fit the model. Do you get exactly the same meta-d' and Mratio values as before? Why / why not?

---

5) FITTING THE MODEL: PARAMETER RECOVERY

We saw in exercise 4 that the meta-d model we are using doesn't give us a perfect estimation of meta-d. We will now take a closer look at how well this model is able to recover meta-d values, using 20 simulations at different levels of meta-d. This is important because our estimation should be both precise and reliable.

Set up the parameters:

```
sim5.params.d = 2; % Set task performance (d')
sim5.params.d_sigma = 0.1; % Include some between-subject variability
sim5.params.c = 0; % Set task bias (c)
sim5.params.c_sigma = 0.1; % Include some between-subject variability
sim5.params.Ntrials = 1000; % Set the number of trials performed
sim5.params.Nratings = 4; % Choose the rating scale to use
sim5.params.Nsims = 20; % Specify the number of simulations at each meta-d value
sim5.params.meta_d = [0.5 1.25 2.0]; % Specify a range of of meta-d values
sim5.params.meta_d_sigma = 0.1; % Include some between-subject variability
```

Simulate the responses and fit the model for different levels of metacognition:

--> This may take a few minutes...

```
for a = 1:length(sim5.params.meta_d)
    for b = 1:sim5.params.Nsims
        % Generate dprime values
        sim5.values{a}.d(b) = normrnd(sim5.params.d, sim5.params.d_sigma);
```

```
        % Generate bias values
        sim5.values{a}.c(b) = normrnd(sim5.params.c, sim5.params.c_sigma);
        % Generate meta-d values
        sim5.values{a}.meta_d(b) = normrnd(sim5.params.meta_d(a), sim5.params.meta_d_sigma);
        % Simulate data
        sim5.data{b,a} = ss_metad_sim(sim5.values{a}.d(b), sim5.values{a}.meta_d(b), sim5.value
        % A small padding is required to avoid problems in model fit if any
        % confidence ratings aren't used (see Hautus MJ, 1995 for details),
        % thereby creating empty cells
        adj_f = 1/length(sim5.data{b,a}.nR_S1);
        sim5.data{b,a}.nR_S1 = sim5.data{b,a}.nR_S1 + adj_f;
        sim5.data{b,a}.nR_S2 = sim5.data{b,a}.nR_S2 + adj_f;
        % Fit the model
        sim5.fit{b,a} = fit_meta_d_MLE(sim5.data{b,a}.nR_S1, sim5.data{b,a}.nR_S2);
    end
end
```

```
Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.
```

```
<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.
```

```
<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

```
Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

```
fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.
```

```
fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
```

```
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

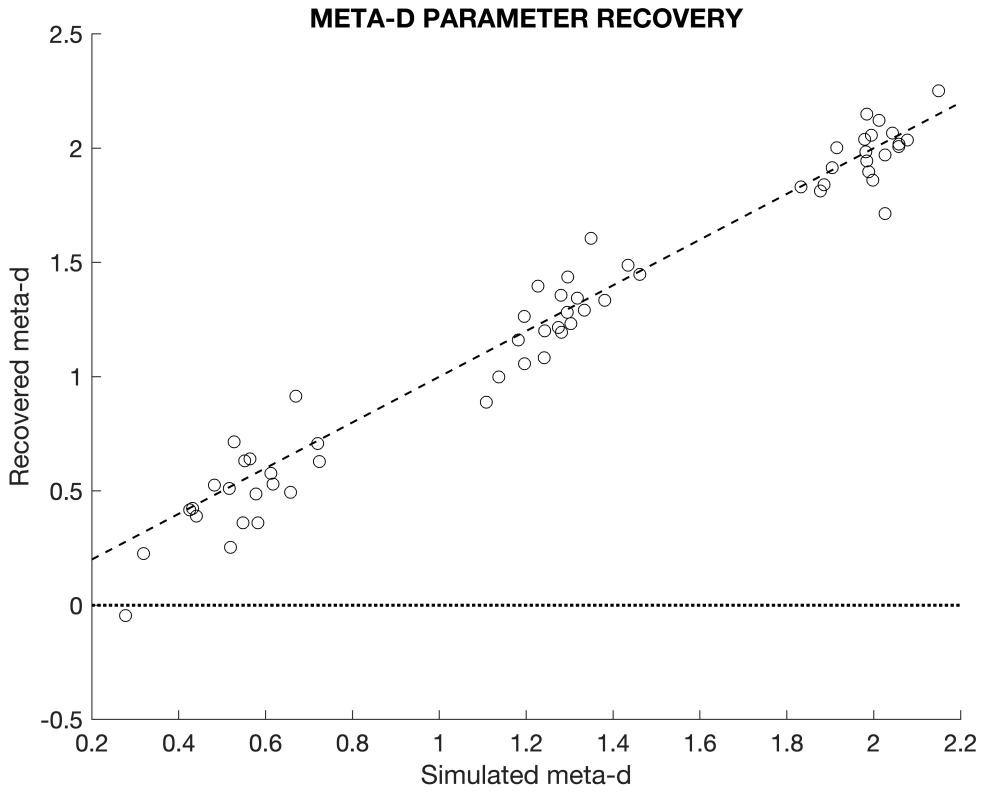
Plot the simulated meta-d (or Mratio) values against the recovered values:

FUNCTION: ss_plot_simVfit(data, data.fit, parameter, model)

NOTE: parameter = 'meta-d' or 'Mratio'

NOTE: model = 'single' or 'group' (model used, here it is 'single')

```
ss_plot_simVfit(sim5, sim5.fit, 'meta-d', 'single');
```



EXPLANATION:

Here we can see that there is some variance in the model fits for each value of meta-d', even when the data were generated from the same underlying parameters. However, they are centred around the generated value, and do not appear to be biased towards being too high or too low.

EXERCISE 5:

Drop the trial numbers (Ntrials) per simulation down to just 100, and re-run the simulation and recovery procedure you just performed. What happens to the recovered values of meta-d with lower numbers of trials per subject? Do you think this may or may not be a problem? Feel free to try other trial numbers if you are interested.

---

6) FITTING THE MODEL: THE HMETA-D ALTERNATIVE

We have now seen in exercise 5 that the original formulation of the meta-d model (that uses a maximum likelihood estimate for model fits) can become highly variable in its estimations when using a low number of trials. Here, we will try an alternative model that uses an hierarchical Bayesian formulation, whereby all single subject values are fit simultaneously and regularised by the overall group value. We will now repeat the exercise from step 5 using this model.

Set up the parameters:

```
sim6.params.d = 2; % Set task performance (d')
sim6.params.d_sigma = 0.1; % Include some between-subject variability
sim6.params.c = 0; % Set task bias (c)
sim6.params.c_sigma = 0.1; % Include some between-subject variability
sim6.params.Ntrials = 100; % Set the number of trials performed
sim6.params.Nratings = 4; % Choose the rating scale to use
sim6.params.Nsims = 20; % Specify the number of simulations at each meta-d value
sim6.params.meta_d = [0.5 1.25 2.0]; % Specify a range of of meta-d values
sim6.params.meta_d_sigma = 0.1; % Include some between-subject variability
```

Simulate the responses and fit the model for different levels of metacognition:

--> This may take a few minutes...

```
for a = 1:length(sim6.params.meta_d)
    for b = 1:sim6.params.Nsims
        % Generate dprime values
        sim6.values{a}.d(b) = normrnd(sim6.params.d, sim6.params.d_sigma);
        % Generate bias values
        sim6.values{a}.c(b) = normrnd(sim6.params.c, sim6.params.c_sigma);
        % Generate meta-d values
        sim6.values{a}.meta_d(b) = normrnd(sim6.params.meta_d(a), sim6.params.meta_d_sigma);
        % Simulate data
        % N.B. Data padding is not needed here
        % N.B. Needs a different nR_S1 and nR_S2 layout for this model
        sim6.data{b,a} = ss_metad_sim(sim6.params.d, sim6.params.meta_d(a), sim6.params.c, sim6
        sim6.bayesGroup.data{a}.nR_S1{b} = sim6.data{b,a}.nR_S1;
        sim6.bayesGroup.data{a}.nR_S2{b} = sim6.data{b,a}.nR_S2;
    end
    % Fit the model as a group for each level of meta-d
    sim6.fit{a} = fit_meta_d_mcmc_group(sim6.bayesGroup.data{a}.nR_S1, sim6.bayesGroup.data{a}.
end
```

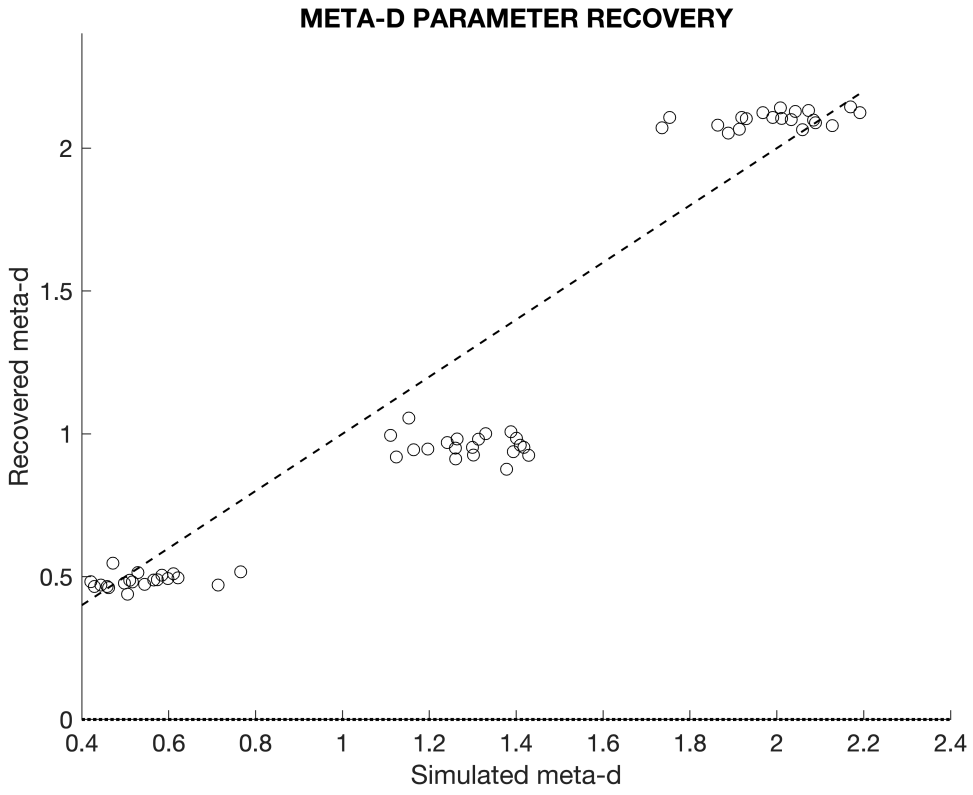
```
----------------------------------------
Hierarchical meta-d' model
https://github.com/smfleming/HMeta-d
----------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 123.338182 seconds.
```

```
----------------------------------------
Hierarchical meta-d' model
https://github.com/smfleming/HMeta-d
----------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 121.539726 seconds.

----------------------------------------
Hierarchical meta-d' model
https://github.com/smfleming/HMeta-d
----------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 112.773369 seconds.
```

Plot the simulated meta-d (or 'Mratio') values against the recovered values:

NOTE: model = 'group'

```
ss_plot_simVfit(sim6, sim6.fit, 'meta-d', 'group');
```



EXPLANATION:

You can see the HMeta-d model appears to greatly reduce the variance in the single subject estimates when compared to the original MLE model, due to the regularisation towards the group mean. This could be particularly useful to help us infer more accurate group mean values when we have low trial numbers per subject.

EXERCISE 6:

Can you think of any potential drawbacks of using an hierarchical model to estimate meta-d and Mratio? Discuss, and feel free to discuss with the tutors as well.

HINTS:

- Can you think of any potential issues with reducing between-subject variance?

- Can you run regular frequentist statistics on hierarchically-fit values?

NOTE: If you are interested in how we might be able to combat these reductions in between-subject variance, please see extension exercise 3...

EXTENSION OPTION 1: INDEPENDENT GROUP COMPARISONS WITH HMETA-D

We will now go through an example whereby we simulate and fit two independent groups, and test for a group difference in Mratio. It is worth noting the type of statistics required to compare groups that are fit within (separate) hierarchical models.

Set up the parameters:

```
simTwoGroupsI.params.d = 2; % Set task performance (d')
simTwoGroupsI.params.d_sigma = 0.1; % Include some between-subject variability
simTwoGroupsI.params.c = 0; % Set task bias (c)
simTwoGroupsI.params.c_sigma = 0.1; % Include some between-subject variability
simTwoGroupsI.params.Ntrials = 100; % Set the number of trials performed
simTwoGroupsI.params.Nratings = 4; % Choose the rating scale to use
simTwoGroupsI.params.Nsims(1) = 35; % Specify the number of simulations for group 1
simTwoGroupsI.params.Nsims(2) = 45; % Specify the number of simulations for group 2
simTwoGroupsI.params.Mratio(1) = 1; % Specify Mratio for group 1
simTwoGroupsI.params.Mratio(2) = 0.6; % Specify Mratio for group 2 = worse
simTwoGroupsI.params.Mratio_sigma = 0.1; % Include some between-subject variability
% N.B. Group sizes do not have to be equal in this instance
```

Simulate data for each of the groups:

```
for a = 1:2
    for b = 1:simTwoGroupsI.params.Nsims(a)
        % Generate dprime values
        simTwoGroupsI.values{a}.d(b) = normrnd(simTwoGroupsI.params.d, simTwoGroupsI.params.d_s
        % Generate bias values
        simTwoGroupsI.values{a}.c(b) = normrnd(simTwoGroupsI.params.c, simTwoGroupsI.params.c_s
        % Generate meta-d values
        simTwoGroupsI.values{a}.meta_d(b) = normrnd((simTwoGroupsI.params.Mratio(a) .* simTwoGr
        % Simulate data
        simTwoGroupsI.data{a}.sims{b} = ss_metad_sim(simTwoGroupsI.values{a}.d(b), simTwoGroups
        simTwoGroupsI.data{a}.nR_S1{b} = simTwoGroupsI.data{a}.sims{b}.nR_S1;
        simTwoGroupsI.data{a}.nR_S2{b} = simTwoGroupsI.data{a}.sims{b}.nR_S2;
    end
end
```

Fit each of the groups separately:

```
simTwoGroupsI.fit.group1 = fit_meta_d_mcmc_group(simTwoGroupsI.data{1}.nR_S1, simTwoGroupsI.dat
```

```
------------------------------------
Hierarchical meta-d' model
https://github.com/smfleming/HMeta-d
------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 226.654304 seconds.
```

```
simTwoGroupsI.fit.group2 = fit_meta_d_mcmc_group(simTwoGroupsI.data{2}.nR_S1, simTwoGroupsI.dat
```

```
------------------------------------
Hierarchical meta-d' model
https://github.com/smfleming/HMeta-d
------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 272.440363 seconds.
```

Compute the highest density interval (HDI) over the distribution of the sample differences in logMratio:

NOTE: For details on the appropriate statistics and HDI see Fleming, 2017: "HMeta-d: hierarchical Bayesian estimation of metacognitive efficiency from confidence ratings". If the HDI spans zero, then we cannot be confident that there is a true difference in metacognition between the groups.

```
sampleDiff = simTwoGroupsI.fit.group1.mcmc.samples.mu_logMratio - simTwoGroupsI.fit.group2.mcmc
hdi = calc_HDI(sampleDiff(:));
fprintf(('\n Mratio group values = %.2f and %.2f'), exp(simTwoGroupsI.fit.group1.mu_logMratio),
```

```
 Mratio group values = 1.04 and 0.68
```

```
fprintf(['\n Estimated difference in Mratio between groups: ', num2str(exp(simTwoGroupsI.fit.gr
```

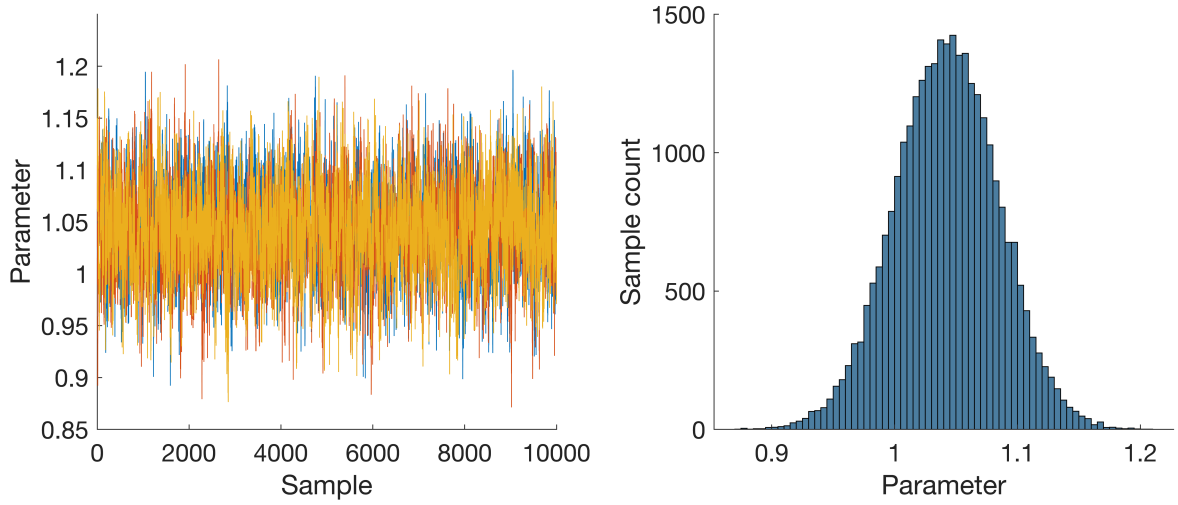
```
 Estimated difference in Mratio between groups: 0.36181
```

```
fprintf(['\n HDI on difference in log(Mratio): ', num2str(hdi) '\n\n'])
```

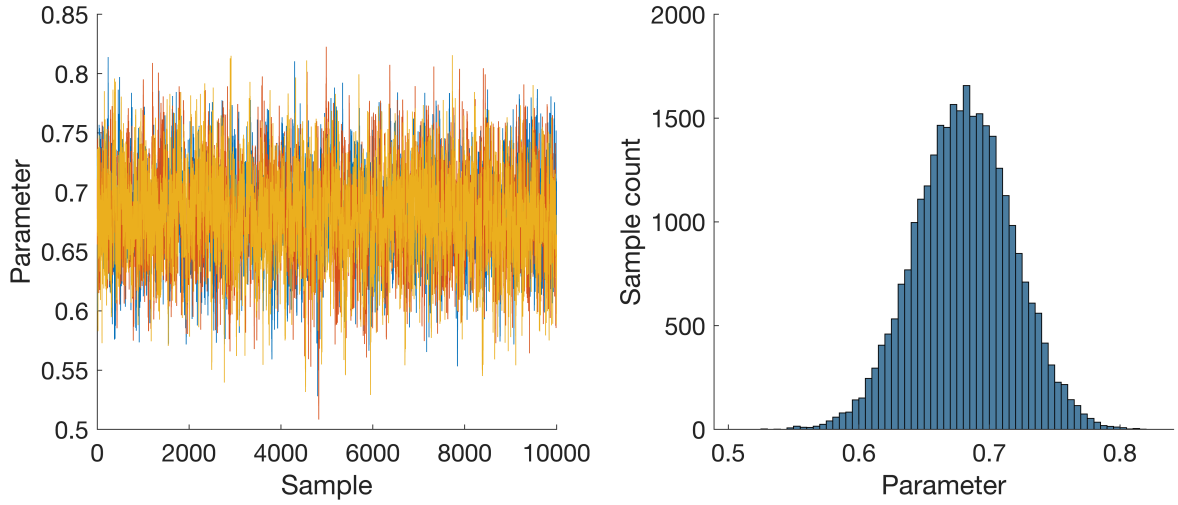
```
 HDI on difference in log(Mratio): 0.2962     0.56292
```

Plot the distributions for Mratio and the sample difference:

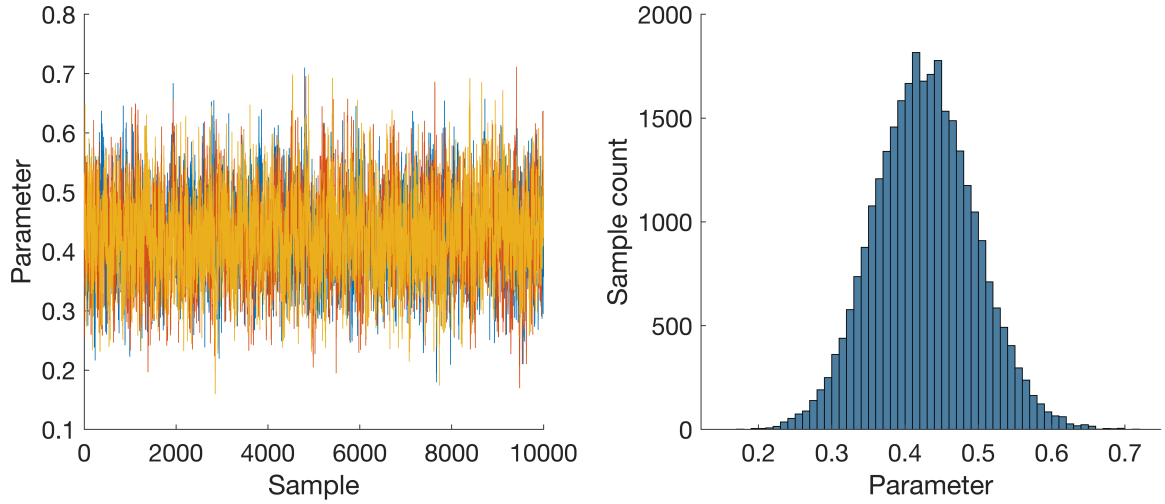
```
plotSamples(exp(simTwoGroupsI.fit.group1.mcmc.samples.mu_logMratio))
```




```
plotSamples(exp(simTwoGroupsI.fit.group2.mcmc.samples.mu_logMratio))
```



```
plotSamples(sampleDiff)
```



Check the convergence values - They should be close to one to ensure reliability of estimation:

```
fprintf(['\n Convergence Rhat group1: ', num2str(simTwoGroupsI.fit.group1.mcmc.Rhat.mu_logMrati
```

```
 Convergence Rhat group1: 1.001
```

```
fprintf(['\n Convergence Rhat group2: ', num2str(simTwoGroupsI.fit.group2.mcmc.Rhat.mu_logMrati
```

```
 Convergence Rhat group2: 1.0004
```

EXTENSION EXERCISE 1:

Run the exercise again, but this time create two groups a very small (e.g. 0.2) difference in Mratio, plus make the variance (sigma) larger (e.g. 0.4) within each of the groups. Do the model fits still give us a difference between the groups, or does it now accurately tell us that there is no difference? This is a sanity check to make sure that the model does not still produce a group difference when there is none, due to too much regularisation towards each group mean.

EXTENSION OPTION 2: DEPENDENT GROUP COMPARISONS WITH HMETA-D

We will now go through an example where we simulate and fit from two dependent groups, for example if we would like to run a longitudinal analysis on repeated measures or a comparison between two metacognitive

tasks from the same subjects. To do this, we use a version of the model that samples from a multivariate normal distribution, and thus does not assume independence between samples.

Set up the parameters:

```
simTwoGroupsD.params.d = 2; % Set task performance (d')
simTwoGroupsD.params.d_sigma = 0.1; % Include some between-subject variability
simTwoGroupsD.params.c = 0; % Set task bias (c)
simTwoGroupsD.params.c_sigma = 0.1; % Include some between-subject variability
simTwoGroupsD.params.Ntrials = 100; % Set the number of trials performed
simTwoGroupsD.params.Nratings = 4; % Choose the rating scale to use
simTwoGroupsD.params.Nsims = 60; % Specify the number of simulations for each session
simTwoGroupsD.params.Mratio(1) = 1; % Specify Mratio for session 1
simTwoGroupsD.params.Mratio(2) = 0.6; % Specify Mratio for session 2 = worse
simTwoGroupsD.params.Mratio_sigma = 0.1; % Include some variability in the Mratio scores
simTwoGroupsD.params.Mratio_rho = 0; % Specify the correlation between the two Mratios
```

Create a covariance matrix for the two Mratios:

```
simTwoGroupsD.params.covMatrix = [simTwoGroupsD.params.Mratio_sigma^2 simTwoGroupsD.params.Mrat
```

Simulate data for each of the groups:

```
for b = 1:simTwoGroupsD.params.Nsims
    % Generate the Mratio values from a multivariate normal distribution
    simTwoGroupsD.MVvalues.Mratio(b,:) = mvnrnd([simTwoGroupsD.params.Mratio(1) simTwoGroupsD.p
    for a = 1:2
        % Generate dprime values
        simTwoGroupsD.values{a}.d(b) = normrnd(simTwoGroupsD.params.d, simTwoGroupsD.params.d_s
        % Generate bias values
        simTwoGroupsD.values{a}.c(b) = normrnd(simTwoGroupsD.params.c, simTwoGroupsD.params.c_s
        % Generate meta-d values
        simTwoGroupsD.values{a}.metad(b) = simTwoGroupsD.MVvalues.Mratio(b,a) .* simTwoGroupsD.
        % Simulate data
        simTwoGroupsD.data{a}.sims{b} = ss_metad_sim(simTwoGroupsD.values{a}.d(b), simTwoGroups
        simTwoGroupsD.responses.nR_S1(a).counts{b} = simTwoGroupsD.data{a}.sims{b}.nR_S1;
        simTwoGroupsD.responses.nR_S2(a).counts{b} = simTwoGroupsD.data{a}.sims{b}.nR_S2;
    end
end
```

Fit the two sessions all together:

```
simTwoGroupsD.fit = fit_meta_d_mcmc_groupCorr(simTwoGroupsD.responses.nR_S1, simTwoGroupsD.resp
```

```
----------------------------------------
Hierarchical meta-d' model
https://github.com/smfleming/HMeta-d
```

```
----------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 663.117041 seconds.
```

Compute the HDI of the difference in logMratio:

```
sampleDiff = simTwoGroupsD.fit.mcmc.samples.mu_logMratio(:,:,1) - simTwoGroupsD.fit.mcmc.sample
hdi = calc_HDI(sampleDiff(:));
fprintf(('\n Mratio session values = %.2f and %.2f'), exp(simTwoGroupsD.fit.mu_logMratio(1)), e
```

```
 Mratio session values = 1.06 and 0.64
```

```
fprintf(['\n Estimated difference in Mratio between sessions: ', num2str(exp(simTwoGroupsD.fit.
```

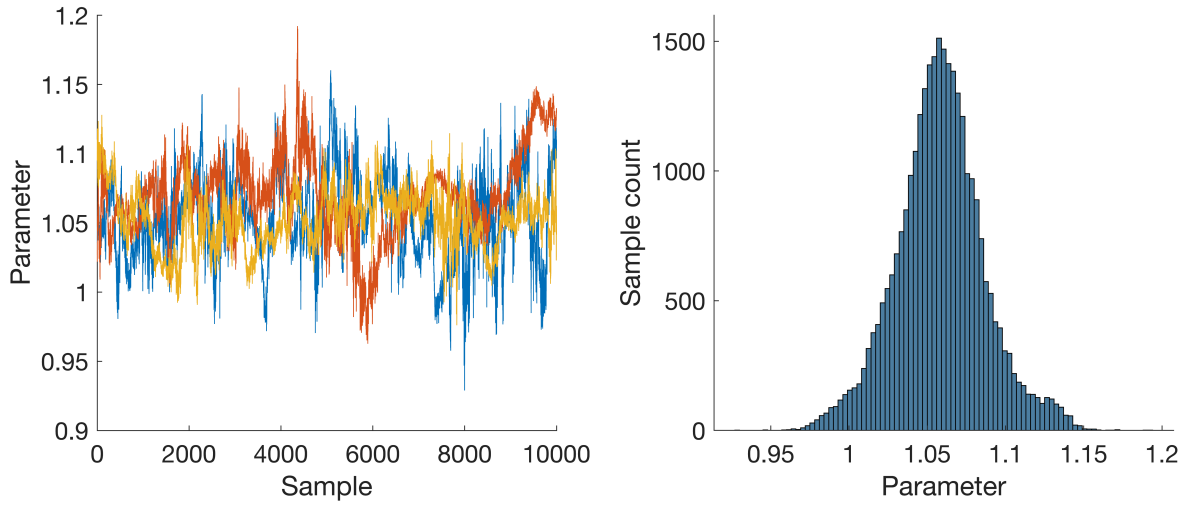
```
 Estimated difference in Mratio between sessions: 0.41745
```

```
fprintf(['\n HDI on difference in log(Mratio): ', num2str(hdi) '\n\n'])
```

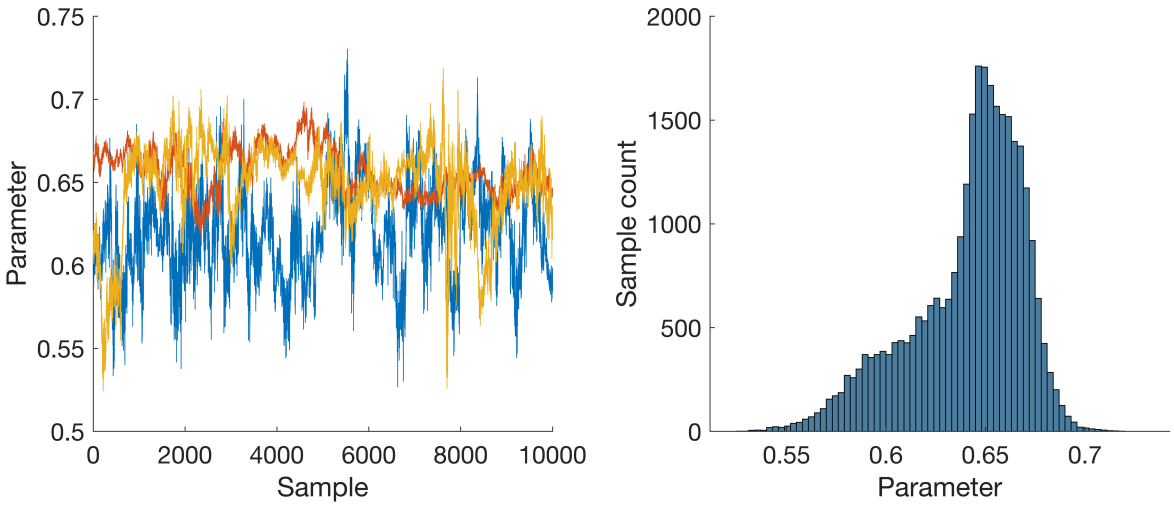
```
 HDI on difference in log(Mratio): 0.41641      0.6145
```

Plot each of the group distributions for Mratio:

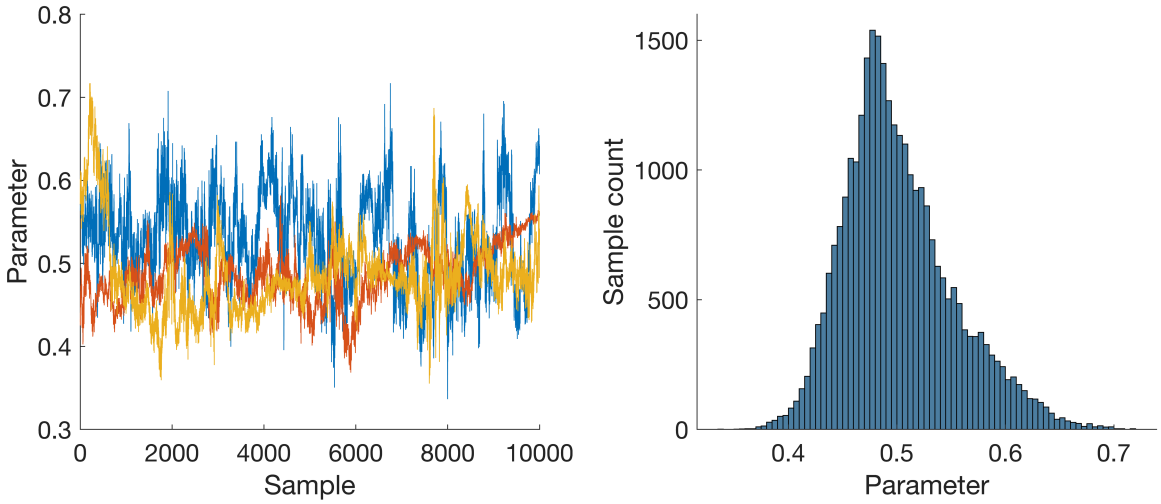
```
plotSamples(exp(simTwoGroupsD.fit.mcmc.samples.mu_logMratio(:,:,1)))
```




```
plotSamples(exp(simTwoGroupsD.fit.mcmc.samples.mu_logMratio(:,:,2)))
```




```
plotSamples(sampleDiff)
```



EXTENSION EXERCISE 2:

This version of the model also allows you to simultaneously estimate the correlation coefficient ('rho') between the two sessions. To explore this, change the correlation coefficient to 0.4 in the parameters above, and re-run the simulations and fit in this section. How well does the model recover this value?

HINT: Rho can be found in simTwoGroupsD.fit.rho

EXTENSION OPTION 3: BETWEEN-SUBJECT COMPARISONS WITH HMETA-D

One potential issue with using an hierarchical group model is that inter-individual variability within a group can be severely reduced by the hierarchical regularisation. This may become a problem if you are interested in comparing single subject values, for example against an external variable in a regression analysis. In this exercise, we will compare the original HMeta-d group fit + a post-hoc regression on the fitted log(Mratio) values, and an extended Regression-HMeta-d model, whereby a group regression coefficient ('beta') is simultaneously hierarchically estimated alongside the group Mratio.

Set up the parameters:

```
simRegression.params.d = 2; % Set task performance (d')
simRegression.params.d_sigma = 0.1; % Include some between-subject variability
simRegression.params.c = 0; % Set task bias (c)
simRegression.params.c_sigma = 0.1; % Include some between-subject variability
simRegression.params.Ntrials = 60; % Set the number of trials performed - let's go low...
simRegression.params.Nratings = 4; % Choose the rating scale to use
simRegression.params.Nsims = 60; % Specify the number of simulations for each session
simRegression.params.Mratio = 0.8; % Specify Mratio for the group
simRegression.params.Mratio_sigma = 0.1; % Include some variability in the Mratio scores
simRegression.params.beta = 0.2; % Specify the group beta value
simRegression.params.beta_sigma = 0.1; % Include some variability in the beta value
simRegression.params.cov = rand(simRegression.params.Nsims, 1)'; % Include a covariate
```

Simulate the data for the group:

```
% Z-score the covariate
simRegression.values.covZscored = zscore(simRegression.params.cov);
% Set up array of d' values normally distributed around group value
simRegression.values.d = normrnd(simRegression.params.d, simRegression.params.d_sigma, [1,simRe
% Set up distribution of criterion values normally distributed around 0
simRegression.values.c = normrnd(simRegression.params.c, simRegression.params.c_sigma, [1,simRe
% Set up initial array of Mratio values
simRegression.values.Mratio = normrnd(simRegression.params.Mratio, simRegression.params.Mratio_
% Set up and array of beta values
simRegression.values.beta = normrnd(simRegression.params.beta, simRegression.params.beta_sigma,
% Set up adjusted log_mratio and meta-d' values according to group beta value
simRegression.values.log_mratio_adjusted = log(simRegression.values.Mratio) + simRegression.val
simRegression.values.Mratio_adjusted = exp(simRegression.values.log_mratio_adjusted);
simRegression.values.meta_d_adjusted = simRegression.values.Mratio_adjusted .* simRegression.va
% Simulate responses
for a = 1:simRegression.params.Nsims
    simRegression.data{a} = ss_metad_sim(simRegression.values.d(a), simRegression.values.meta_d
    simRegression.responses.nR_S1{a} = simRegression.data{a}.nR_S1;
    simRegression.responses.nR_S2{a} = simRegression.data{a}.nR_S2;
```

```
end
```

Run fit for for Hierarchical Bayesian model, followed by regression (first checking if toolbox is installed):

```
any(any(contains(struct2cell(ver), 'Statistics and Machine Learning Toolbox')))
```

```
ans = logical
   1
```

```
simRegression.fit.bayesGroupMean = fit_meta_d_mcmc_group(simRegression.responses.nR_S1, simRegr
```

```
----------------------------------------
Hierarchical meta-d' model
https://github.com/smfleming/HMeta-d
----------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 359.944682 seconds.
```

```
simRegression.fit.bayesGroupMeanReg = fitlm(simRegression.values.covZscored, log(simRegression.
fprintf('\nFIT FOR GROUP MEAN HMETA-D + REGRESSION COMPLETE: Beta = %.2f\n', simRegression.fit.
```

```
FIT FOR GROUP MEAN HMETA-D + REGRESSION COMPLETE: Beta = 0.03
```

Run fit for for Hierarchical Bayesian model with simultaneous regression:

```
simRegression.fit.bayesGroupRegression = fit_meta_d_mcmc_regression(simRegression.responses.nR_
```

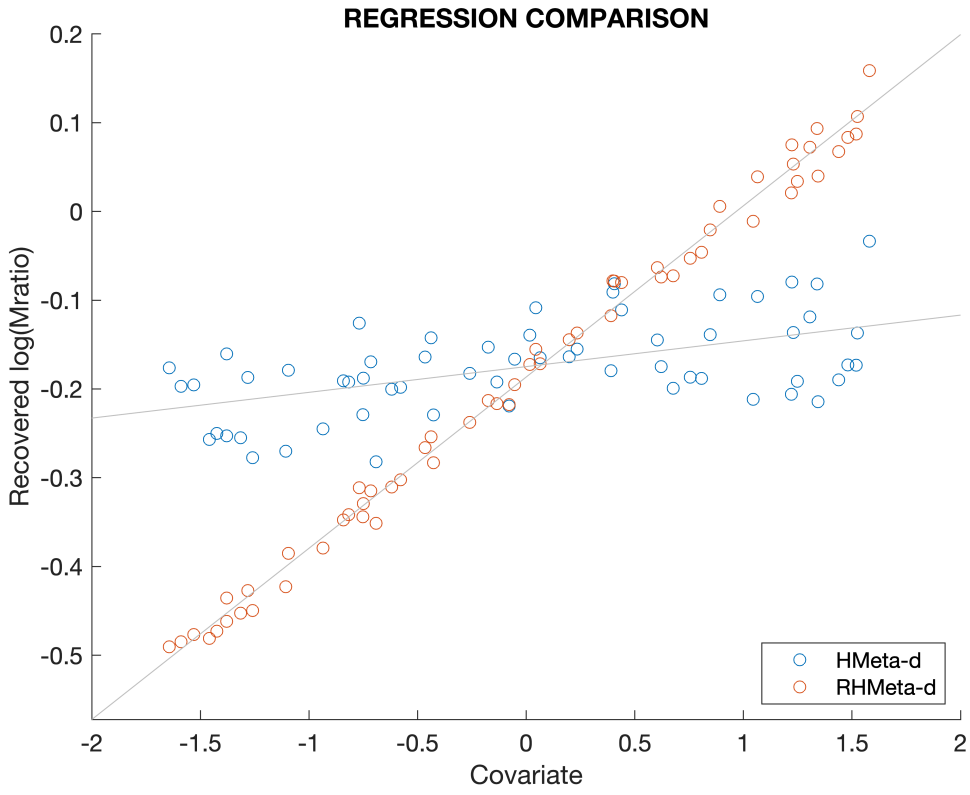
```
----------------------------------------
Hierarchical meta-d' regression model
https://github.com/smfleming/HMeta-d
----------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 588.011361 seconds.
```

```
fprintf('\nFIT FOR REGRESSION HMETA-D COMPLETE: Beta = %.2f\n', simRegression.fit.bayesGroupReg
```

```
FIT FOR REGRESSION HMETA-D COMPLETE: Beta = 0.19
```

Plot the log(Mratio) values against the covariate for each of the models:

```
ss_plot_simVfit(simRegression, simRegression.fit, 'log(Mratio)', 'regression');
```



EXTENSION EXERCISE 3:

Re-run this simulation and fit with a greater number of trials per subject (i.e. > 100 trials). What happens to the difference in the beta estimates between the models? Why do you think this may happen?

EXTENSION OPTION 4: RESPONSE-CONDITIONAL FITS WITH HMETA-D

One further option we can explore is the possibility that the Mratio may be different for each of the two stimulus options within the two-alternative task. Here, we will test out the HMeta-d model with a response-conditional fit, assuming different Mratios within the data, and compare this to the original model.

Set up the parameters:

```
simRC.params.d = 2; % Set task performance (d')
simRC.params.d_sigma = 0.1; % Include some between-subject variability
simRC.params.c = 0; % Set task bias (c)
```

```
simRC.params.c_sigma = 0.1; % Include some between-subject variability
simRC.params.Ntrials = 1000; % Set the number of trials performed
simRC.params.Nratings = 4; % Choose the rating scale to use
simRC.params.Nsims = 100; % Specify the number of simulations at each meta-d value
simRC.params.noise = [0.2 0.8]; % % Specify the internal noise to be different between the two
```

Get default parameters and change to response-conditional:

```
simRC.params.mcmc_params.standard = fit_meta_d_params;
simRC.params.mcmc_params.RC = fit_meta_d_params;
simRC.params.mcmc_params.RC.response_conditional = 1;
```

Simulate the data for the group using response-conditional parameters:

```
% Set up array of d' values normally distributed around group value
simRC.values.d = normrnd(simRC.params.d, simRC.params.d_sigma, [1,simRC.params.Nsims]);
% Set up distribution of criterion values normally distributed around 0
simRC.values.c = normrnd(simRC.params.c, simRC.params.c_sigma, [1,simRC.params.Nsims]);
% Simulate responses
for a = 1:simRC.params.Nsims
    simRC.data{a} = ss_type2_SDT_sim(simRC.values.d(a), simRC.params.noise, simRC.values.c(a),
    simRC.responses.nR_S1{a} = simRC.data{a}.nR_S1;
    simRC.responses.nR_S2{a} = simRC.data{a}.nR_S2;
end
```

Fit the data using a standard model (no response-conditional setting):

```
simRC.fit.standard = fit_meta_d_mcmc_group(simRC.responses.nR_S1, simRC.responses.nR_S2, simRC.
```

```
----------------------------------------
Hierarchical meta-d' model
https://github.com/smfleming/HMeta-d
----------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 890.993929 seconds.
```

```
fprintf('\nFIT FOR STANDARD HMETA-D COMPLETE: MRATIO = %.2f\n', exp(simRC.fit.standard.mu_logMr
```

```
FIT FOR STANDARD HMETA-D COMPLETE: MRATIO = 0.71
```

Fit the data using a response-conditional model:

```
simRC.fit.RC = fit_meta_d_mcmc_group(simRC.responses.nR_S1, simRC.responses.nR_S2, simRC.params
```

```
----------------------------------------
```

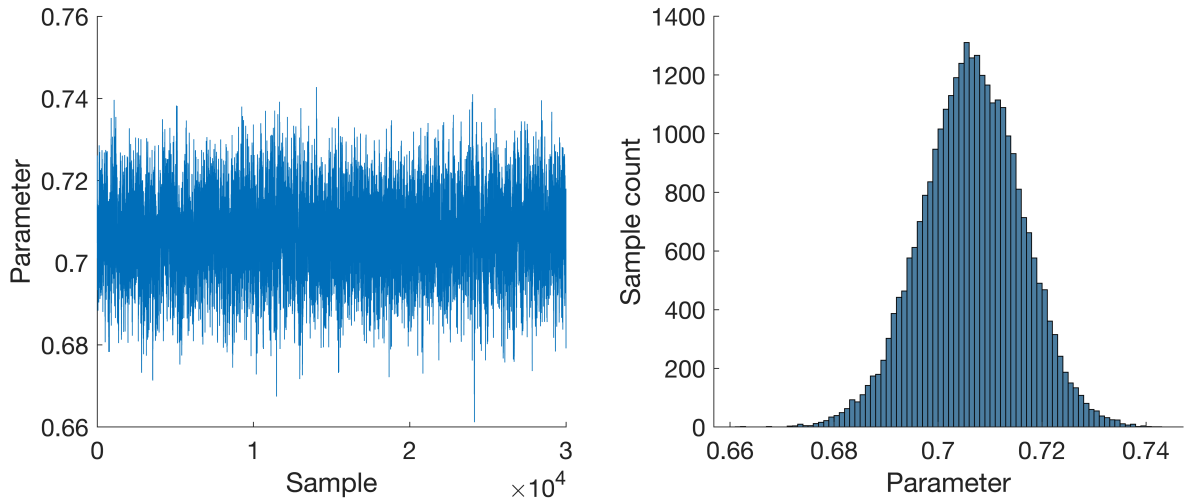
```
Hierarchical meta-d' model
https://github.com/smfleming/HMeta-d
---------------------------------------

Running JAGS ...
Running chain 1 (serial execution)
Running chain 2 (serial execution)
Running chain 3 (serial execution)
Elapsed time is 986.503349 seconds.
```

```
fprintf('\nFIT FOR RESPONSE-CONDITIONAL HMETA-D COMPLETE: MRATIOS = %.2f and %.2f\n', exp(simRC
```

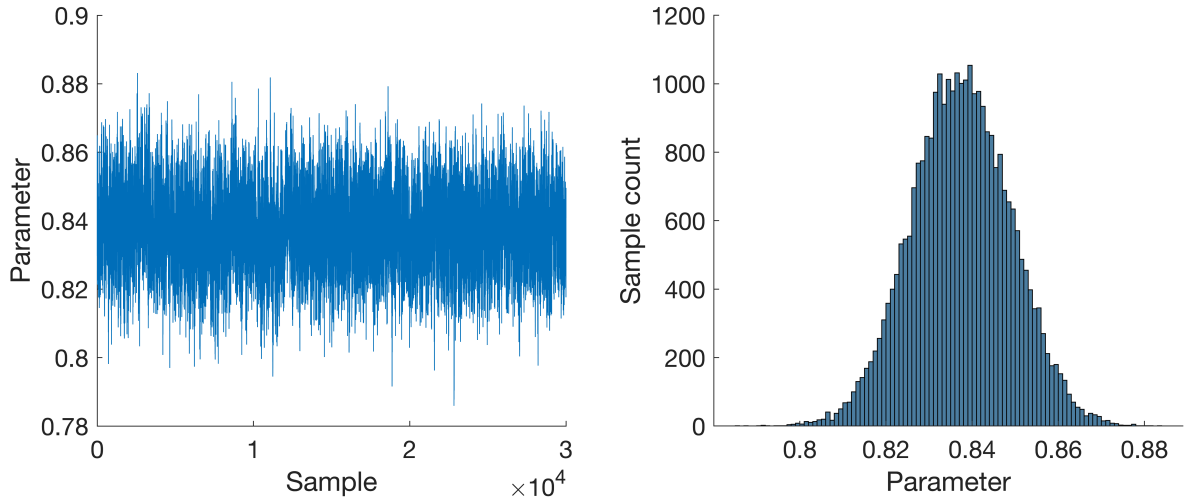
```
FIT FOR RESPONSE-CONDITIONAL HMETA-D COMPLETE: MRATIOS = 0.84 and 0.55
```

Plot the Mratio samples for each fit:

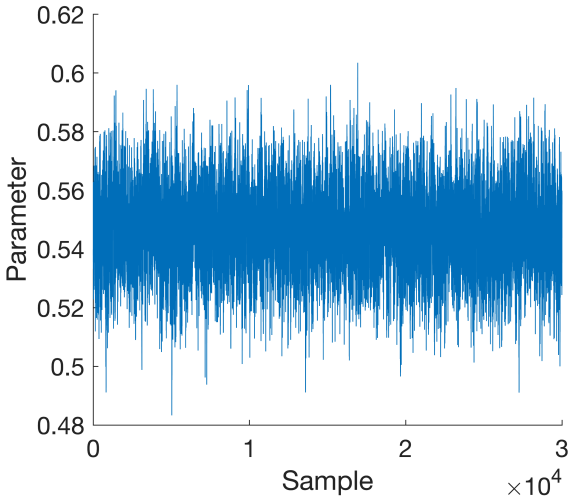
```
plotSamples(exp(simRC.fit.standard.mcmc.samples.mu_logMratio(:)))
```

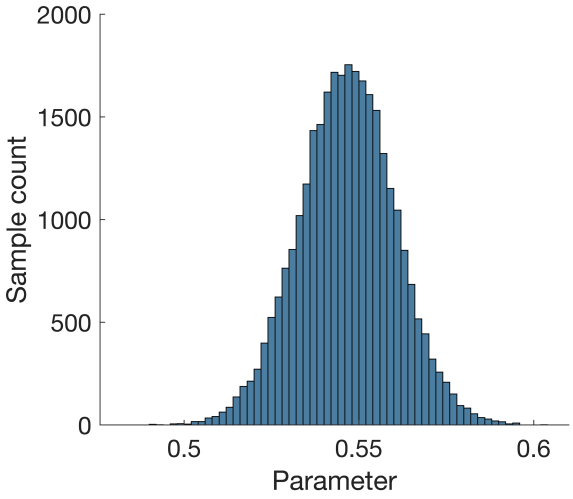



```
plotSamples(exp(simRC.fit.RC.mcmc.samples.mu_logMratio_rS1(:)))
```




```
plotSamples(exp(simRC.fit.RC.mcmc.samples.mu_logMratio_rS2(:)))
```





EXTENSION EXERCISE 4:

Try averaging the two Mratio scores from the response-conditional fit - does this equal the Mratio value from the standard fit? Why?

HINT: The group Mratio can be found by taking the exponential of the group mu_logMratio fit (i.e. exp(simRC.fit.standard.mu_logMratio))

THE END
