Assignment 4
Categorization - The Alien game
Studygroup 5

The task
The alien game is a game where the participant is shown a sequence of aliens with differing appearances. The aliens can differ in appearance in 5 different ways: arms up/down, legs thick/thin, eyes on stalks/not on stalks, spots/no spots, and color blue/green. Some aliens are dangerous, some are nutritious, some are both and some are neither dangerous nor nutritious. The participants must choose whether to tap (to gather nutritious substance from the alien), kill, tap&kill, or ignore the alien, and they then receive feedback so they can learn what category the alien belongs to.
To win the game, the participant must correctly categorize the aliens (dangerous/harmless, nutritious/not nutritious), based on their appearance.
In this assignment we focus on the danger category and session 1: Aliens having spots AND slim legs are dangerous.

We simulate the experiment stimuli by making 5-dimensional vectors (for each possible difference in appearance), each vector value can be either 0 or 1 (i.e. 0 = no spots, 1 = spots), which creates 32 unique combinations of alien appearances.
The 32 combinations are presented in 3 cycles, thus we have 96 trials. In these simulated stimuli we define that if the agent has spots and eyes on stalks then the alien is dangerous, here called that feature 3 AND 4 are 1.

Model
We then simulate agents playing the game using the Generalized Context Model (GCM) which assumes that stimuli are stored as exemplars. The model then predicts that people make similarity comparisons between the stimuli and the exemplars, and base their decision on the overall similarities to each category. Having implemented these agents we plot their performance for different parameter values. The general idea with the GCM being that each agent has a weight for each type of stimuli (i.e. 5) and a similarity constant that determines the exponential decay rate of how similarity decays with distance (i.e. higher values steeper decay).
Figure 1, shows how agents with different parameter values from the GCM performs. Here c is the similarity constant and w being an indicator for which weights the agent has following eq. 1. meaning that “right” means that the agent has weights that suit the experiment for performing well, “wrong” being weights that are unsuited for performing well and equal being “equal” weights for each weight. As can be seen from the figure there is an interaction between the similarity constant and the weights, such that having the “right” weights only matter for performance when the similarity constant is high enough i.e. above 1.5.

Eq1:

$$weight = \begin{cases} Dirichlet(100,100,100,100,100)\ if\ w = "equal" \\ Dirichlet(50,50,500,500,50)\ if\ \ w = "right" \\ Dirichlet(200,200,50,50,200)\ if\ \ w = "wrong" \end{cases}$$

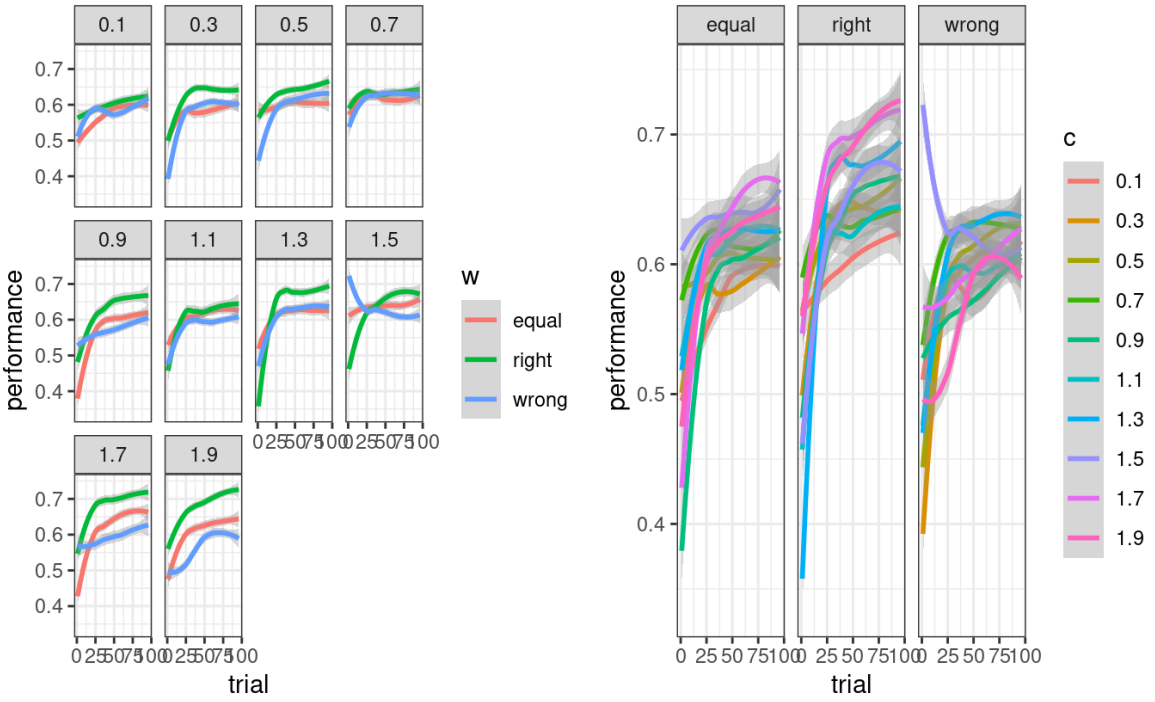
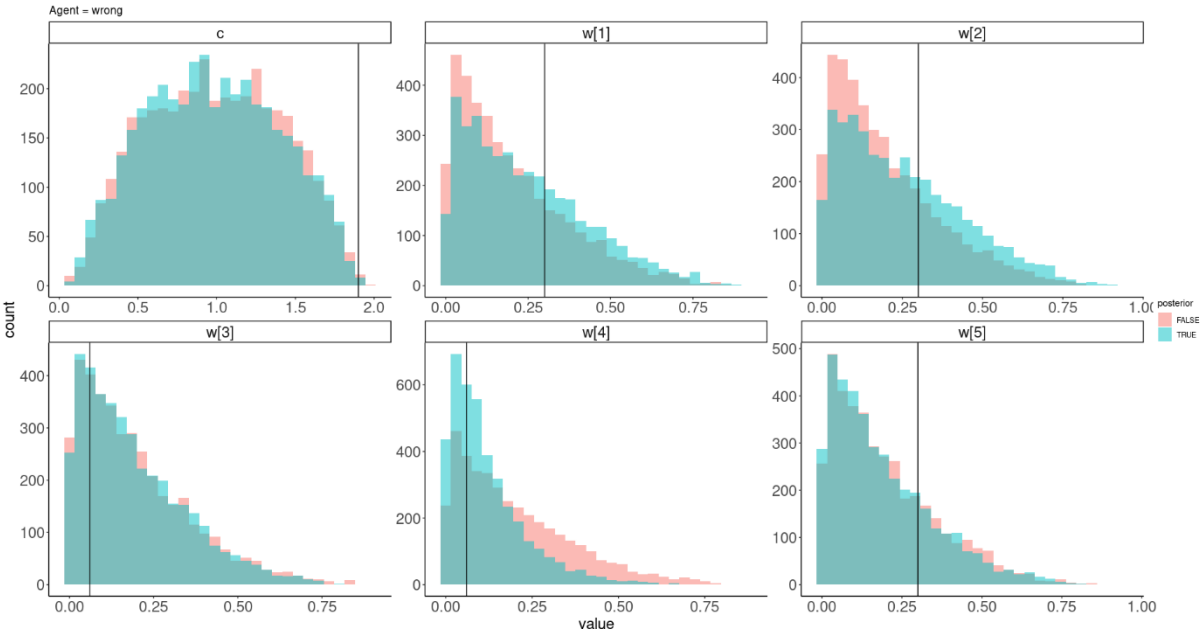
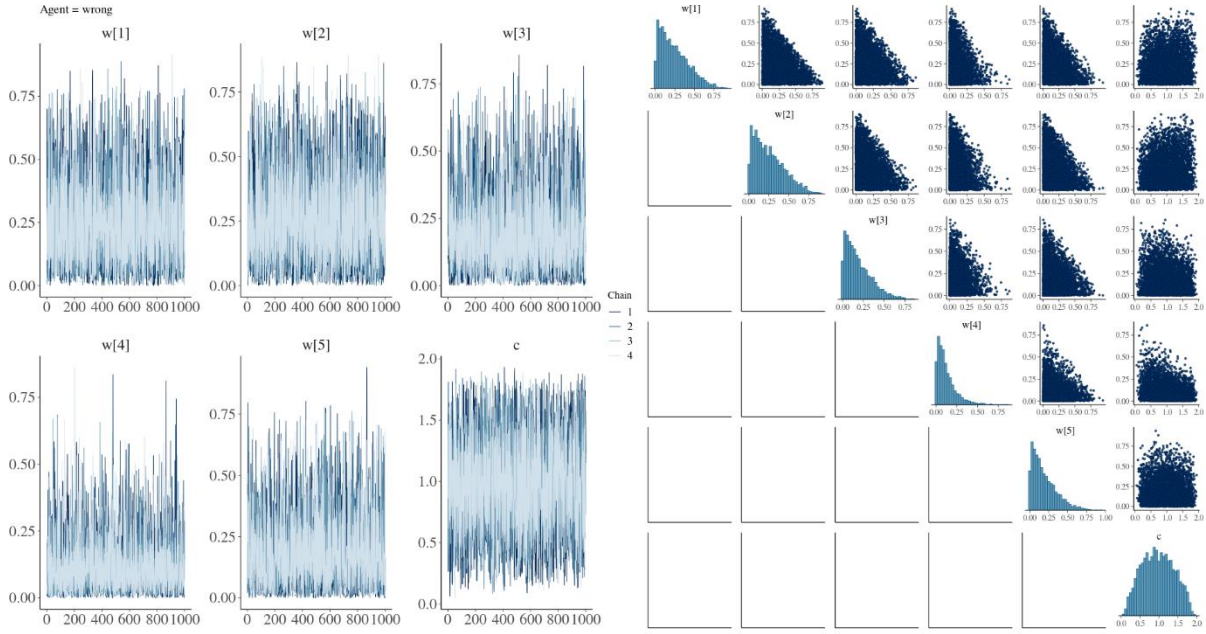


Figure 1; displays different agents' performance in playing the alien game with different parameter values from the GCM model.

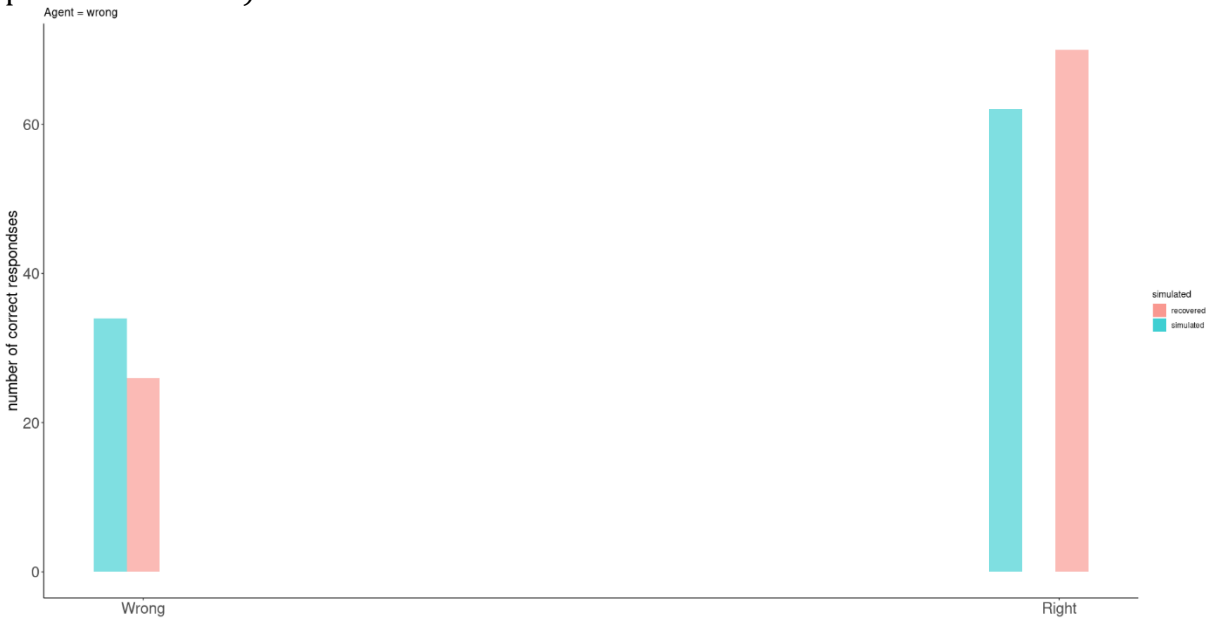
After having seen that our agents behave as we expect (i.e. higher weights for the right stimulus category performing better), we now want to invert the model and see if we can recover the parameters we simulated our agents with (i.e. parameter recovery). We first look at parameter recovery for our 3 different agents as demonstrated above (equal, right, wrong) with a high similarity constant value. We do this because it seems from looking at the performance plot above that the difference in performance is first really present after the similarity constant goes above 1.6. The following plots are therefore all for similarity constant of 1.9.
First looking at the agent with the "wrong weights".



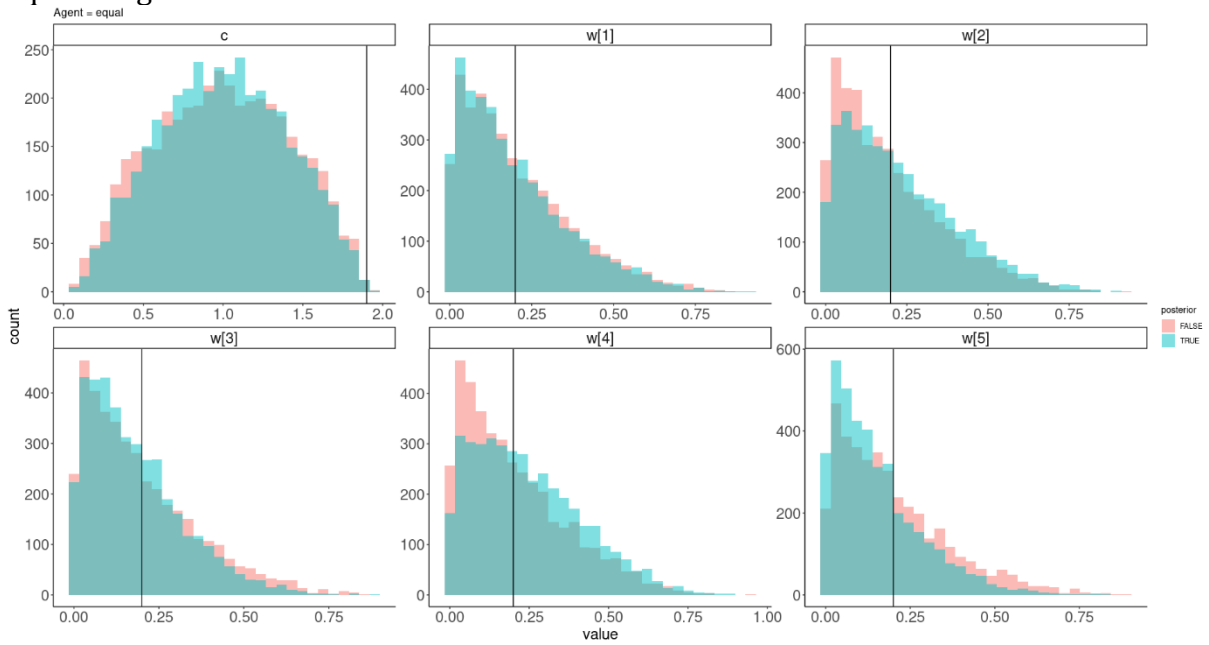
Convergence:

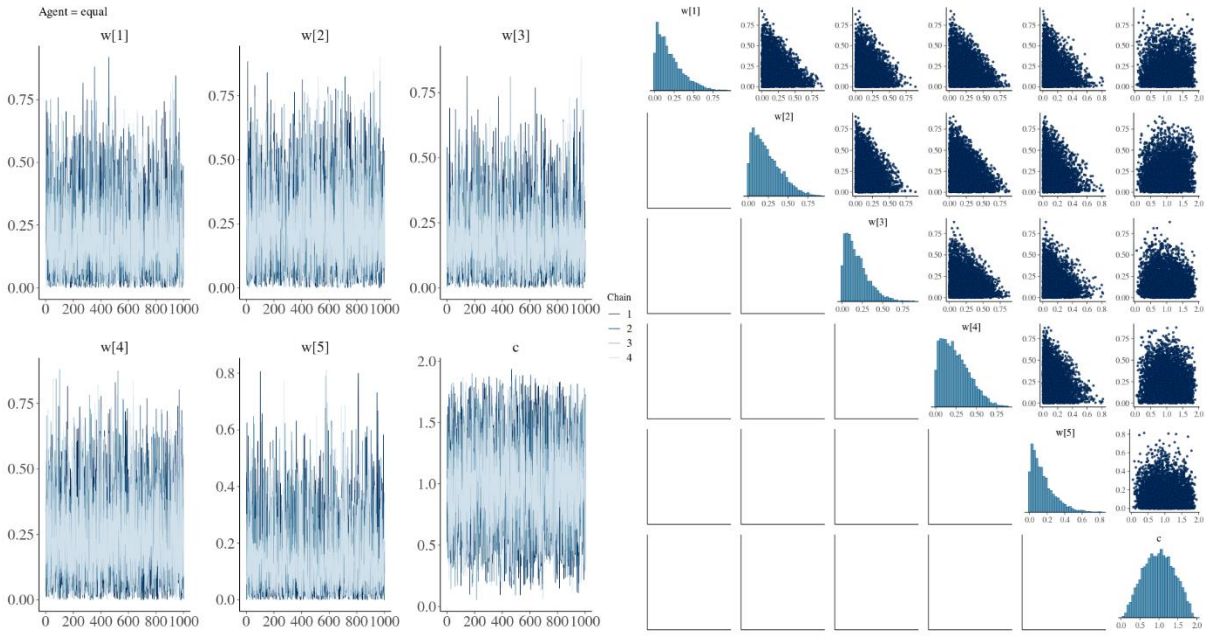


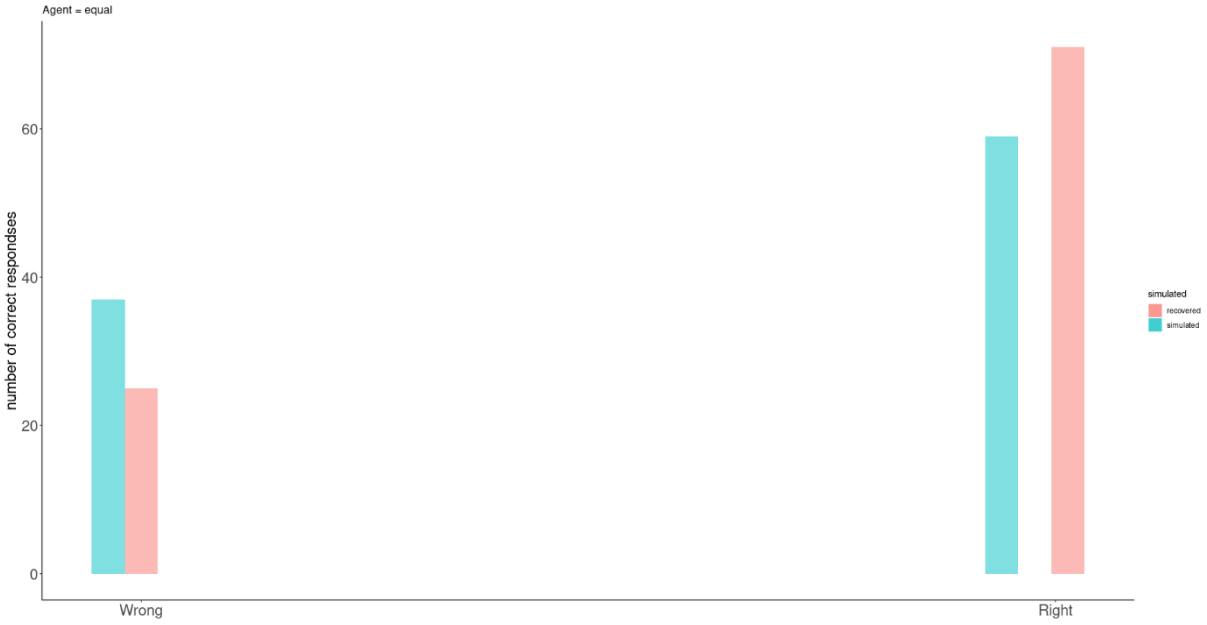
Lastly looking at the simulated responses vs the recovered responses (i.e. posterior predictive checks).



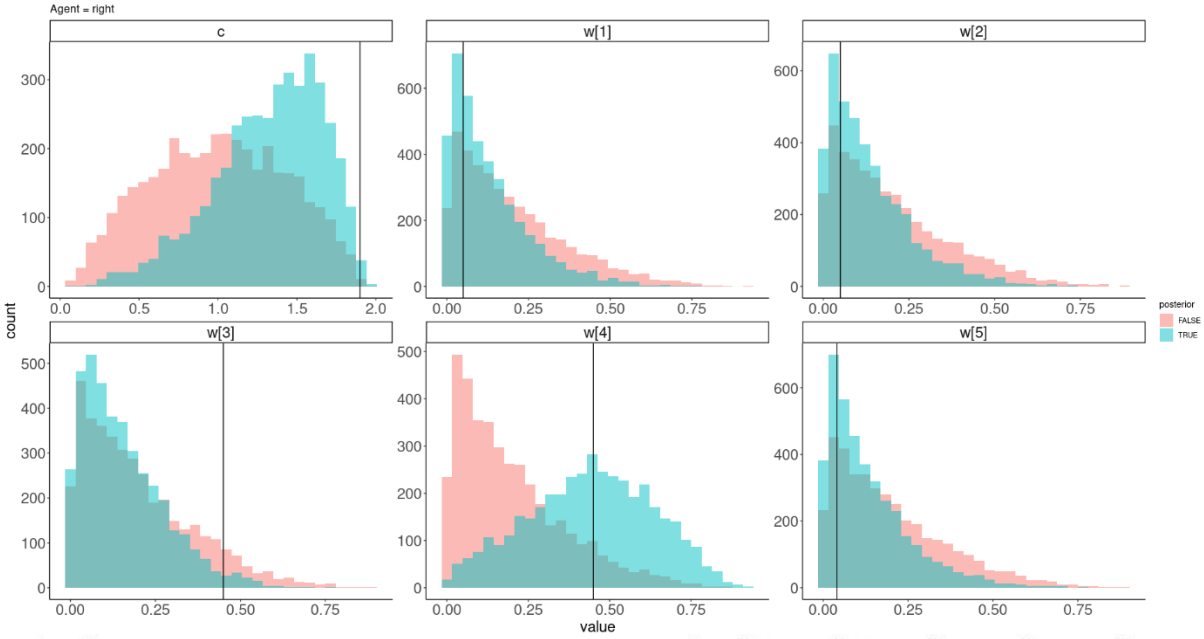
We then do this for the other 2 agents as well:
equal weights:

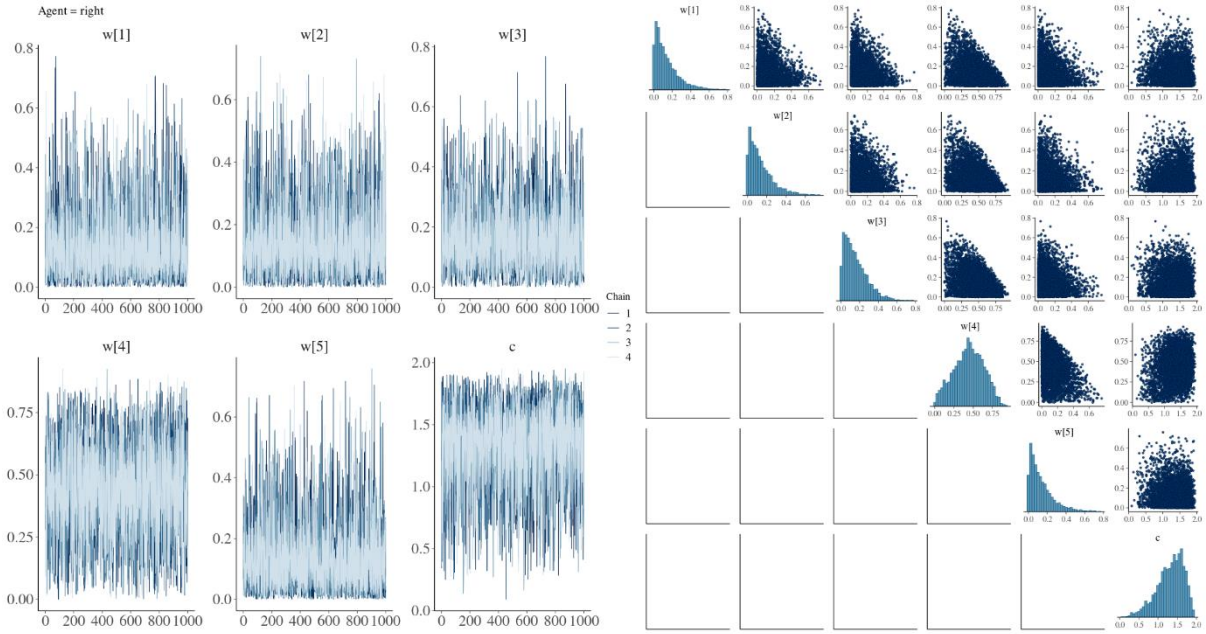


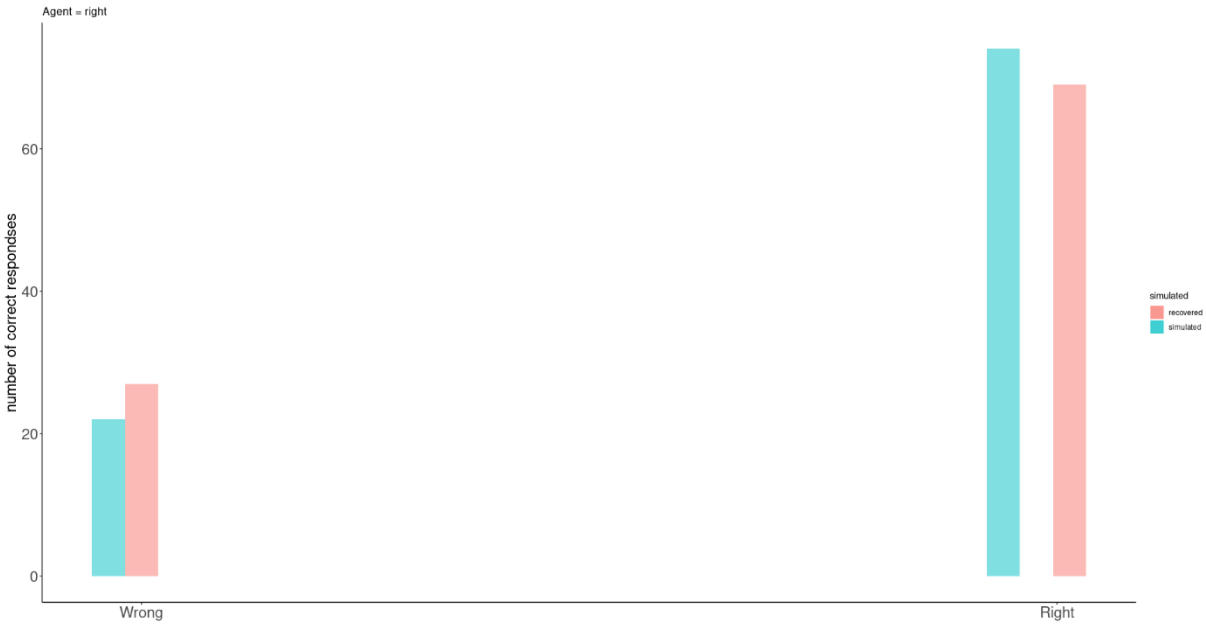




the right weights:

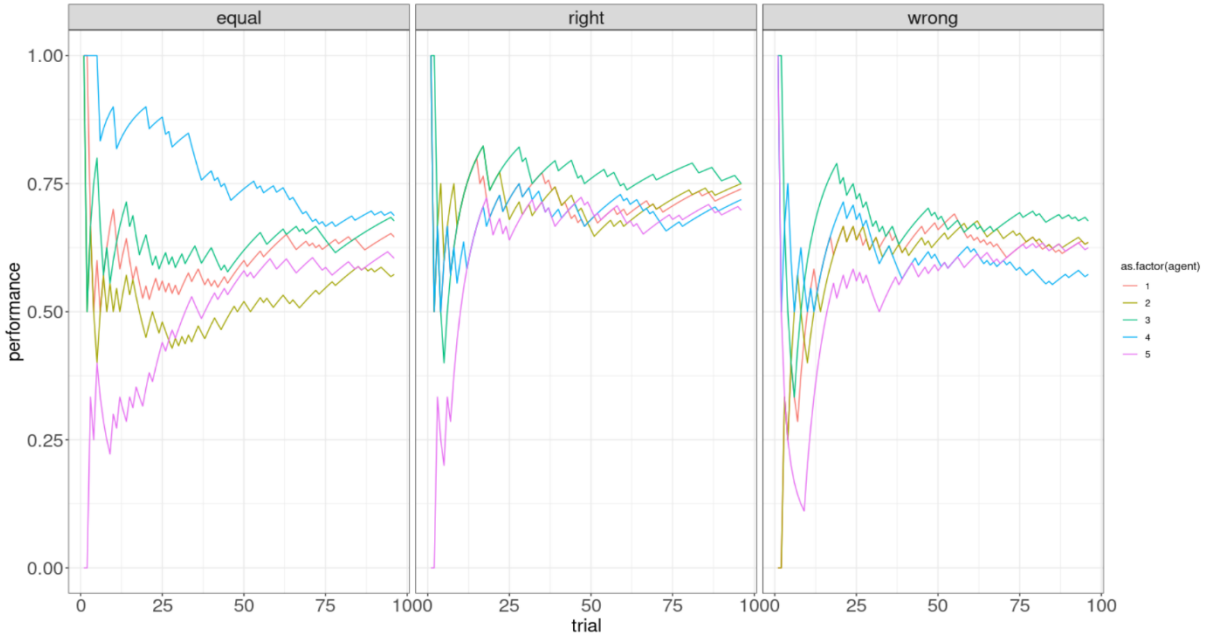

Agent = right
c
w[1]
w[2]
w[3]
w[4]
w[5]
count
value
posterior
FALSE
TRUE


Agent = right
w[1]
w[2]
w[3]
w[4]
w[5]
c
Chain
1
2
3
4



From these convergence, posterior predictive checks, and prior posterior updates we see a couple of things. There seems to be no convergence issues which is good, however the prior posterior updates indicate that the parameters really can't be recovered when the weights are wrong or equal, but when the weights have the "right" configuration the prior posterior updates look a lot better especially for weight 4 and the similarity constant. The posterior predictive checks are decent for all models, but best for the "right" weights which is reassuring.

We now implement a hierarchical version of the same model and simulate responses based on that. Here we simulate 5 agents coming from the 3 different types (i.e. "wrong", "right" and "equal".



Here the values for the hierarchical level parameters were:

$$c \sim \beta(\mu_c, \kappa_c) \cdot 2$$

$$u_c = 0.9$$

$$\kappa_c = 100$$

$$W_{1,2,\dots,S} \sim Dirichlet(S, W \cdot \kappa_W)$$

$$\kappa_W = \begin{matrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{matrix}$$

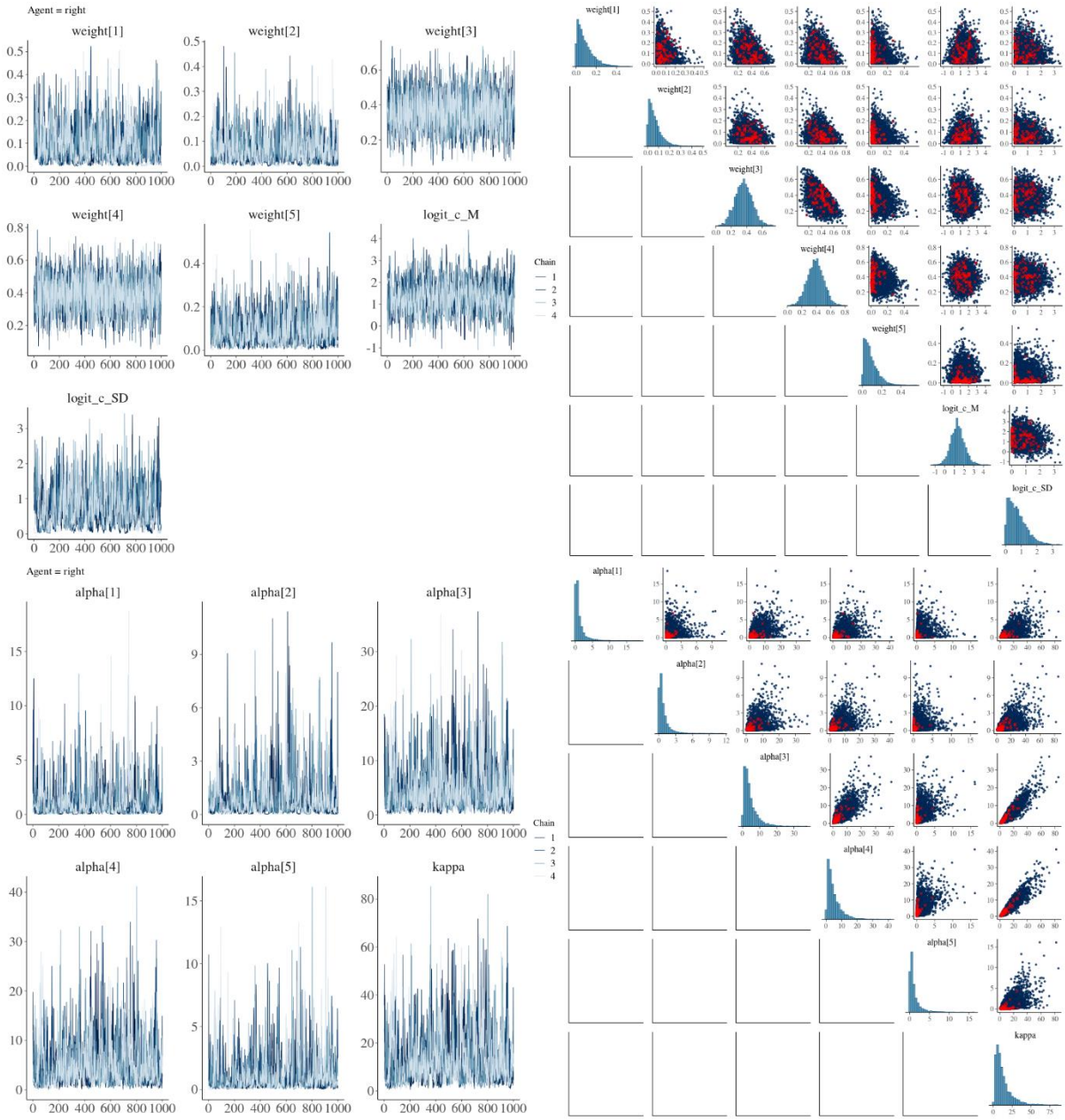
$$W = \begin{cases} [100,100,100,100,100] & if\ "equal \\ [50,50,500,500,50] & if\ "right" \\ [200,200,50,200,200] & if\ "wrong" \end{cases}$$

where () is the reparameterized beta distribution with mean and precision.

Next we invert these models with stan as with the single subject level models. We do this first using a centered approach to the “right” agents, as one could expect this to recover most nicely, given the single subject model.
To help reduce divergent chains we here do not run with the usual adapt_delta and max_treedepth values but instead with adapt_delta = 0.99 and max_treedepth = 15.

looking at the convergence we run into issues:

```
Warning: 104 of 4000 (3.0%) transitions ended with a divergence.
See https://mc-stan.org/misc/warnings for details.

Warning: 2 of 4 chains had an E-BFMI less than 0.2.
See https://mc-stan.org/misc/warnings for details.
```



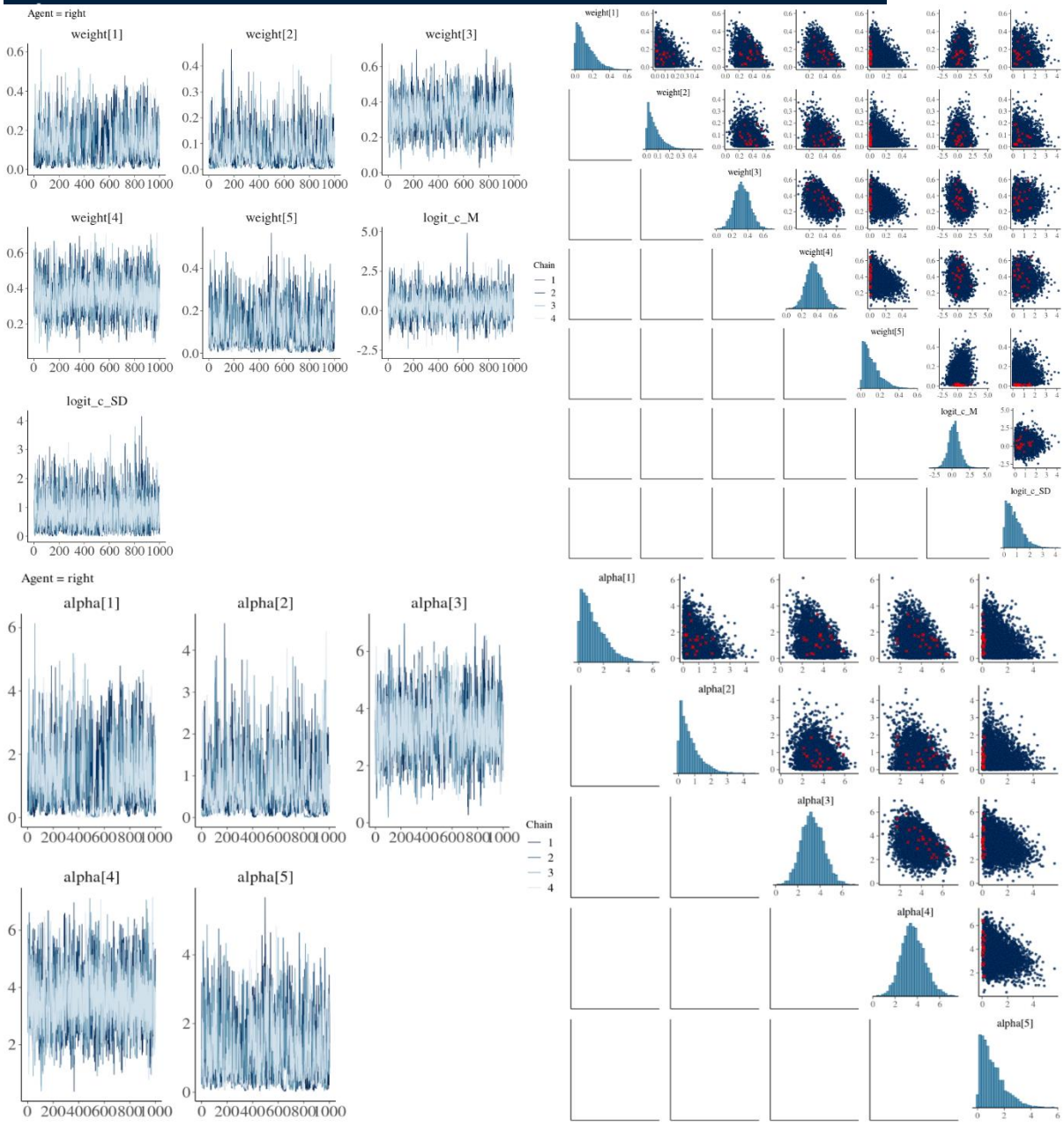
Given the amount of divergent transitions and that 2 of the 4 chains had an E-BFMI of less than 0.2 it seems obvious to make the model non-centered especially when looking at the draws from kappa and the alpha values. However it is not clear how to non-center the kappa and alpha values if even possible so we start with doing it for the similarity parameter and look at the differences: (there might be a way https://discourse.mc-stan.org/t/non-centred-parameterisation-for-dirichlet-distribution/1843)

which did not help:

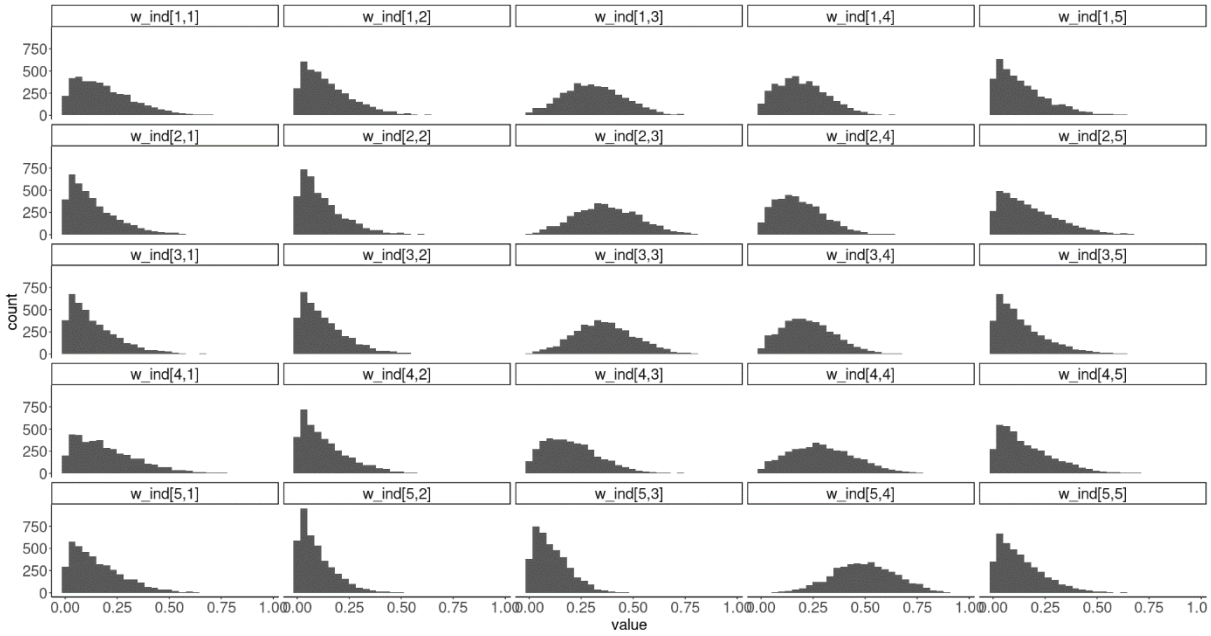
```
Registered S3 method overwritten by 'data.table':
  method           from
  print.data.table
Warning: 105 of 4000 (3.0%) transitions ended with a divergence.
See https://mc-stan.org/misc/warnings for details.

Warning: 4 of 4 chains had an E-BFMI less than 0.2.
See https://mc-stan.org/misc/warnings for details.
```

Next we therefore tried to fix the kappa parameter (here to 10) as it is this parameter with the alphas that seems to cause the problem.
this seemed to help a bit:

```
Warning: 21 of 4000 (1.0%) transitions ended with a divergence.
See https://mc-stan.org/misc/warnings for details.
```



The remaining problem could be that the alpha 5 parameter gets very close to 0 as that is where the divergent transitions are. Next we therefore fix the alpha parameters as well. This model fits very nice and fast, the problem with this is that the weights are not actually hierarchical anymore, they are just taken from the prior of (i.e. dirichlet(1,1,1,1,1)).
However, showing the individual posterior distributions for the weights for the subjects we see that something is definitely happening on feature 3 and 4. which we expect given the simulated data.



We therefore went with this option as it produced no divergences and was 10x faster:

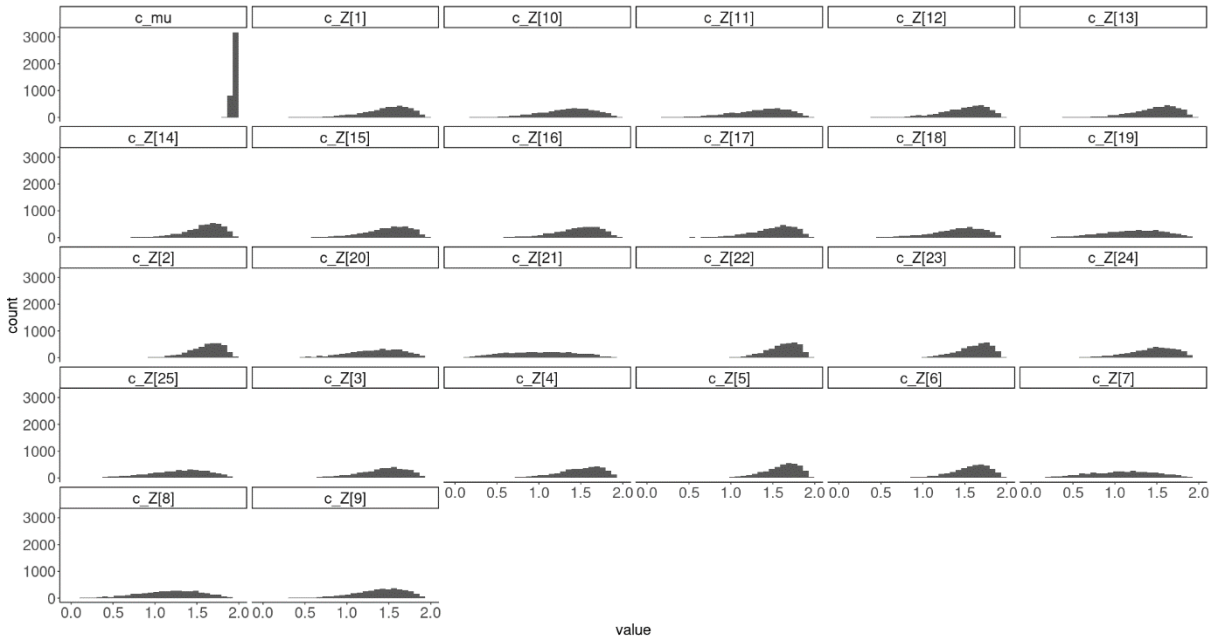
fitting to the empirical data. An important note seems to be that all participants do not get the same sequence of aliens and this has to be accounted for when modeling it.

The model converged nicely here are some of the trace plots:


Agent =
logit_c_M
logit_c_SD
logit_c_Z[1]
logit_c_Z[2]
logit_c_Z[3]
logit_c_Z[4]
logit_c_Z[5]
logit_c_Z[6]
logit_c_Z[7]
logit_c_Z[8]
logit_c_Z[9]
logit_c_Z[10]
logit_c_Z[11]
logit_c_Z[12]
logit_c_Z[13]
logit_c_Z[14]
logit_c_Z[15]
logit_c_Z[16]
logit_c_Z[17]
logit_c_Z[18]
logit_c_Z[19]
logit_c_Z[20]
logit_c_Z[21]
logit_c_Z[22]
logit_c_Z[23]
logit_c_Z[24]
logit_c_Z[25]
Chain
1
2
3
4
Agent =
w_ind[1,1]
w_ind[2,1]
w_ind[3,1]
w_ind[4,1]
w_ind[5,1]
w_ind[6,1]
w_ind[7,1]
w_ind[8,1]
w_ind[9,1]
w_ind[10,1]
w_ind[11,1]
w_ind[12,1]
w_ind[13,1]
w_ind[14,1]
w_ind[15,1]
w_ind[16,1]
w_ind[17,1]
w_ind[18,1]
w_ind[19,1]
w_ind[20,1]
w_ind[21,1]
w_ind[22,1]
w_ind[23,1]
w_ind[24,1]
w_ind[25,1]
Chain
1
2
3
4

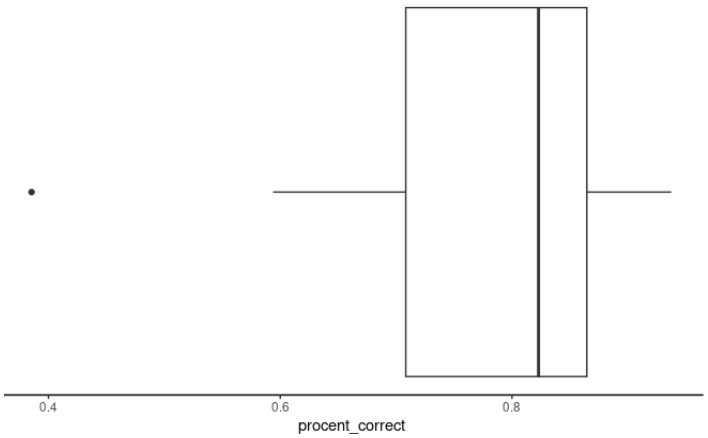

Agent =
w_ind[1,2]
w_ind[2,2]
w_ind[3,2]
w_ind[4,2]
w_ind[5,2]
w_ind[6,2]
w_ind[7,2]
w_ind[8,2]
w_ind[9,2]
w_ind[10,2]
w_ind[11,2]
w_ind[12,2]
w_ind[13,2]
w_ind[14,2]
w_ind[15,2]
w_ind[16,2]
w_ind[17,2]
w_ind[18,2]
w_ind[19,2]
w_ind[20,2]
w_ind[21,2]
w_ind[22,2]
w_ind[23,2]
w_ind[24,2]
w_ind[25,2]
w_ind[1,3]
w_ind[2,3]
w_ind[3,3]
w_ind[4,3]
w_ind[5,3]
w_ind[6,3]
w_ind[7,3]
w_ind[8,3]
w_ind[9,3]
w_ind[10,3]
w_ind[11,3]
w_ind[12,3]
w_ind[13,3]
w_ind[14,3]
w_ind[15,3]
w_ind[16,3]
w_ind[17,3]
w_ind[18,3]
w_ind[19,3]
w_ind[20,3]
w_ind[21,3]
w_ind[22,3]
w_ind[23,3]
w_ind[24,3]
w_ind[25,3]
Chain
1
2
3
4

Next we look at some of the posterior distributions for the similarity constant parameters.

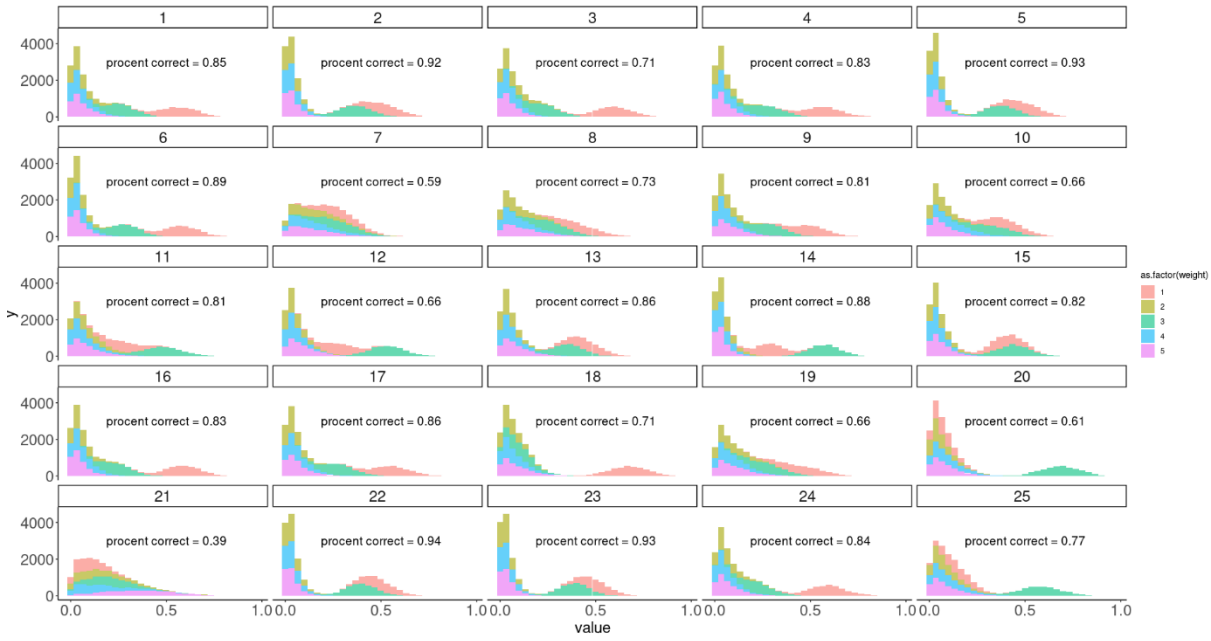


Here we see that the hierarchical mean parameter is very close to 2 and that most participants are scattered around values close to 2. Perhaps increasing the bound of this parameter could be meaningful i.e. making it go up to 5 or 10.

Next we look at the weights, before doing so we also look at the actual performance of the participants. As can be seen in the plot below people generally had good accuracy in the first session of the task mean = .78 sd = 0.13.



This was also reflected in the weights for the features as the features that actually code for whether an alien was dangerous can be seen below. Here one has to remember that feature 1 AND 3 coded for the alien being dangerous.

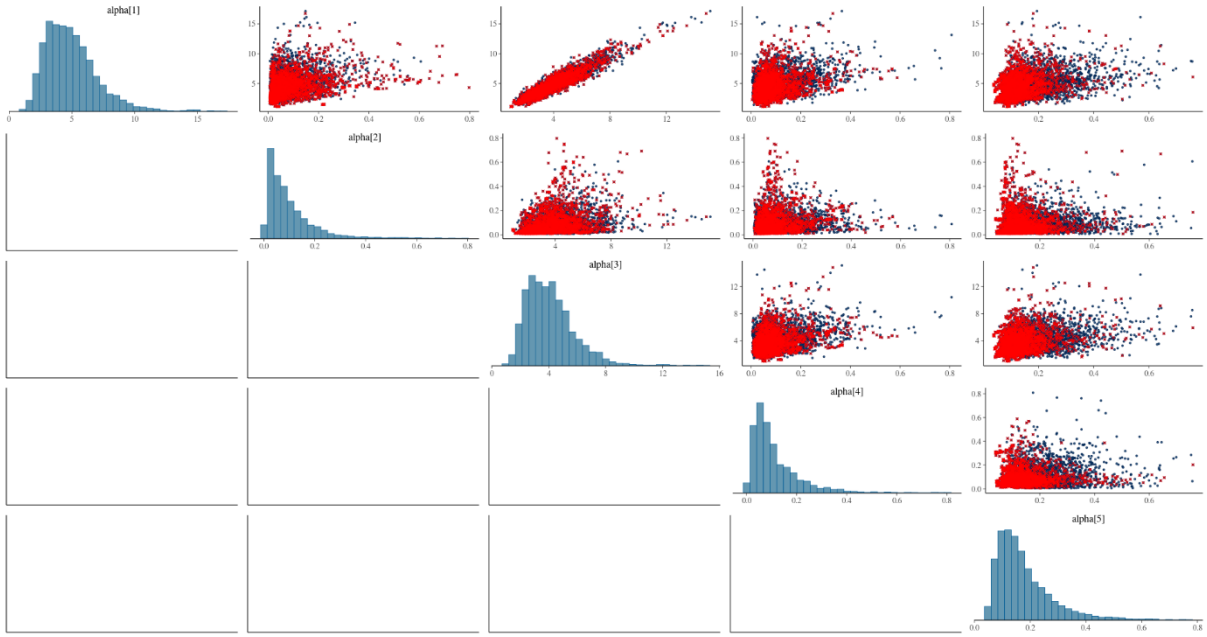


From the plot above it's pretty obvious that the participants with a high accuracy are picked up by the model by the fact that their posterior distributions for weight 1 and 3 are very different from the other 3 weights. An interesting observation is also that participants that score very poorly seem to have all weights grouped together (participant 21). Lastly and interestingly participants with a medium accuracy seem to have that one of the two weights are far away from 0 i.e. participant 25, 20 and 18.

We also tried to fit the model where the alpha parameter wasn't fixed i.e having hierarchical weights (and a fixed kappa), this resulted in many divergent chains:

```
Warning: 2283 of 4000 (57.0%) transitions ended with a divergence.
See https://mc-stan.org/misc/warnings for details.

Warning: 4 of 4 chains had an E-BFMI less than 0.2.
See https://mc-stan.org/misc/warnings for details.

$num_divergent
[1] 528 629 535 591

$num_max_treedepth
[1] 0 0 0 0

$ebfmi
[1] 0.14362364 0.15823557 0.06303812 0.13539517
```

again the problems might be due to the funnels and correlations in alpha values:



Conclusions:

As what reflected from the simulation, the GCM model provides a potential strategy on categorization tasks, but it only works with proper accuracy when the highest weights are allocated on the "right" features, i.e. that are associated with the category, and this also apply when it comes to parameter recovery and estimation, where the model performs the best when the weights are allocated "right". This reveals the model's limitation, as in the real life people would usually have a random or biased weight on each feature at the beginning, then adjust the weight on each feature based on the feedback along trails, in another word, the weight parameters are not fixed. Therefore, even when the parameters in the GCM model can be properly recovered or estimated, they still have limitations with representing the underlying cognitive mechanisms. When it comes to the hierarchical GCM, divergence is another critical issue. A potential solution to this issue is setting all parameter kappa and alpha with fixed values, however, in this case, it's not hierarchical anymore strictly speaking, which will harm the generalization and application of the model.