Your Turn!

```
# needed packages
library(curl)
```

```
## Using libcurl 7.68.0 with OpenSSL/1.1.1f
```

```
library(brms)
```

```
## Loading required package: Rcpp
```

```
## Loading 'brms' package (version 2.17.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
```

```
##
## Attaching package: 'brms'
```

```
## The following object is masked from 'package:stats':
##
##     ar
```

First, we need the data again, this time with the response variable:

```
# See https://github.com/mdnunez/encodingN200 for more information about the data
filename <- curl("https://raw.githubusercontent.com/jstbcs/ESCOP2022-WS/main/Data/pdmdata.csv")
pdm <- read.csv(filename)

pdm$spfn <- 1 - (as.numeric(as.factor(pdm$spf)) - 1)
pdm$response <- ifelse(pdm$accuracy==1, pdm$spfn, 1 - pdm$spfn)
```

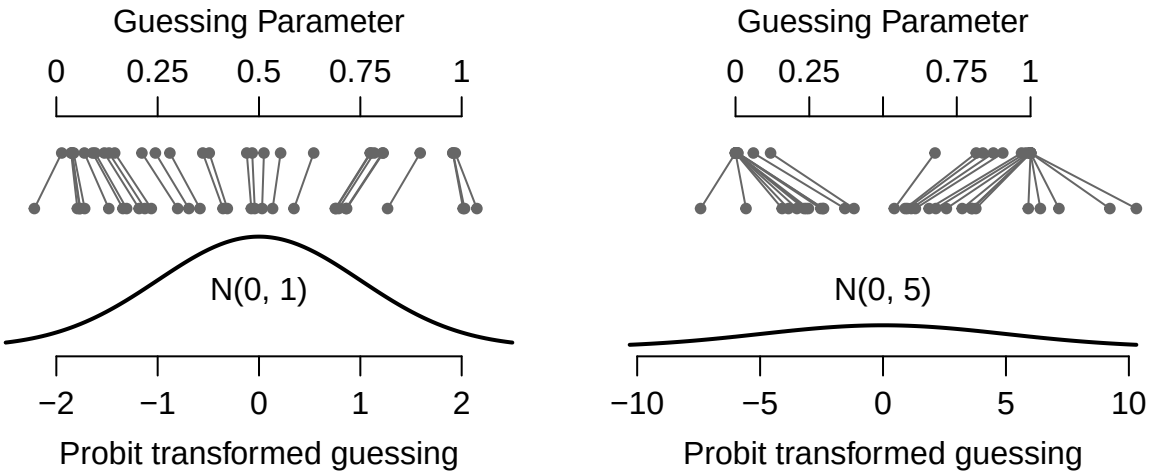
Non-hierarchical Signal Detection Model

$$Y_i \sim \text{Bernoulli}(p_i),$$
$$p_i = \Phi(\mu_i),$$
$$\mu_i = \beta_0 + \beta_1 \text{spf}_i.$$

```
fit1 <- brm(response ~ 1 + factor(spf),
            family = bernoulli(link="probit"),
            data = pdm[pdm$subject==1,])
```

Priors

Now let's think about priors. There is a subtle component of the model that should inform the priors. It's the probit-transformation from latent space to probability space. It is key to ensure a reasonable distribution on the actual parameter scale (here: probabilities). If the prior on the latent space is too wide, then the transformed prior on probabilities becomes weird. Here is an example of what that could look like:



Given that, we might want to use something like this:

```
Prior <- c(prior(normal(0, 1), class = "Intercept"),
           prior(normal(0, 1), class = "b")
           )
```

Now let's refit the model with these priors.

```
fit2 <- brm(response ~ 1 + factor(spf),
           family = bernoulli(link="probit"),
           data = pdm[pdm$subject==1,],
           prior = Prior)
```

```
summary(fit2)
```

Did anything change?

Non-hierarchical Signal Detection Model

Now we are ready to fit a model on all participants! For this, we need to adjust the model to account for nested data. That means the data get another subscript, Y_{ij} with i corresponding to people and j corresponding to trials.

$$
\begin{aligned}
Y_{ij} &\sim \text{Bernoulli}(p_{ij}),\\
p_{ij} &= \Phi(\mu_{ij}),\\
\mu_{ij} &= \nu_{0,i} + \nu_{1,i}\text{spf}_{ij}.
\end{aligned}
$$

This way, everyone receives their own criterion and sensitivity. These person-level parameters then come from dirstibutions themselves with overall criterion and sensitivity as means:

$$\beta_{0,i} \sim \text{Normal}(\beta_0, \sigma_0^2), \beta_{1,i} \sim \text{Normal}(\beta_1, \sigma_1^2).$$

This setup can be expressen in brms like this:

```
fit3 <- brm(response ~ 1 + factor(spf) + (1 + factor(spf) | subject),
           family = bernoulli(link="probit"),
           data = pdm)
```

Priors

Our previous priors for the βs can serve as priors again (though be careful that the priors on the individual parameters still make sense, they are variable depending on the priors on the σs). Now we only need to add priors for the variances. We again use priors on the square root, the standard deviation. We can stay simple and use a half-normal distribution on these standard deviations.

```
Prior <- c(prior(normal(0, 1), class = "Intercept"),
           prior(normal(0, 1), class = "b"),
           prior(normal(0, 1), class = "sd", lb = 0)
           )
```

Other Extensions

Notice that I added a few settings here. First, I reduce the adaptation algorithm so that the warmup is faster. This can lead to convergence issues under some circumstances, so make sure to assess both warnings in the output and convergence statistics carefully. I also save all parameters in an Rdata file. This can be useful if fitting the model takes a long time. This way you have access to all the posterior chains from all parameters. I run the model on four cores to reduce fitting time further.

```
fit4 <- brm(response ~ 1 + factor(spf) + (1 + factor(spf) | subject),
           family = bernoulli(link="probit"),
           data = pdm,
           prior = Prior,
           control = list(adapt_delta = .9),
           save_pars = save_pars("all"),
           cores = 4, iter = 2000,
           file = "data/sdtmodel")
```

```
summary(fit4)
```

```
##  Family: bernoulli
##   Links: mu = probit
## Formula: response ~ 1 + factor(spf) + (1 + factor(spf) | subject)
##    Data: pdm (Number of observations: 5532)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~subject (Number of levels: 12)
##                             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)                   0.10      0.05     0.02     0.21 1.00     1014
## sd(factorspflow)                0.20      0.07     0.08     0.35 1.00      909
## cor(Intercept,factorspflow)    -0.80      0.23    -0.99    -0.17 1.00      996
##                             Tail_ESS
## sd(Intercept)                   1137
## sd(factorspflow)                1083
## cor(Intercept,factorspflow)     1483
##
## Population-Level Effects:
##              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept       -0.12      0.04    -0.21    -0.04 1.00     2165     2253
## factorspflow    -0.94      0.07    -1.08    -0.80 1.00     1904     1886
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
```

```
## scale reduction factor on split chains (at convergence, Rhat = 1).
```