Introduction to Bayesian statistical modelling

A course with R, Stan, and brms

Ladislas Nalborczyk (UNICOG, NeuroSpin, CEA, Gif/Yvette, France)

Planning

Course n°01: Introduction to Bayesian inference, Beta-Binomial model

Course n°02: Introduction to brms, linear regression

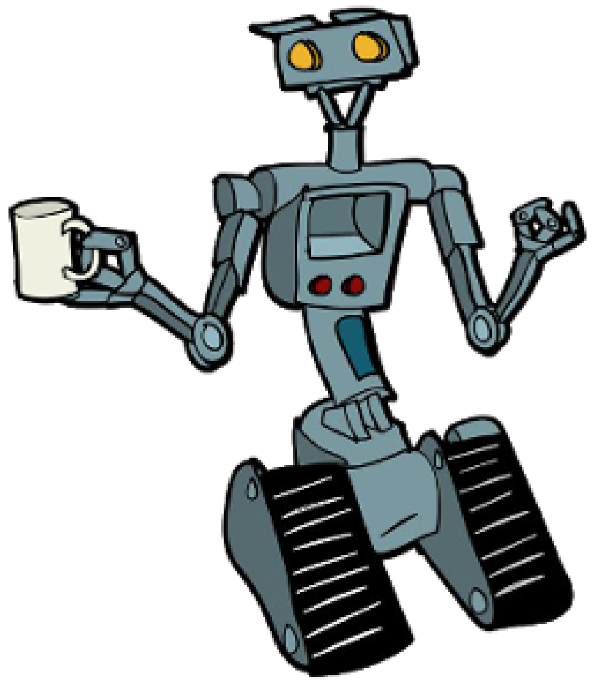
Course n°03: Markov Chain Monte Carlo, generalised linear model

Course n°04: Multilevel models, cognitive models

Multilevel models

The aim is to build a model that can **learn at several levels**, a model that can produce estimates that will be informed by the different groups present in the data. We will follow the following example throughout this course.

Let's assume that we've built a robot that visits cafés and measures the waiting time after ordering a coffee. This robot visits 20 different cafés, 5 times in the morning and 5 times in the afternoon, and measures the time (in minutes) it takes to get a coffee.



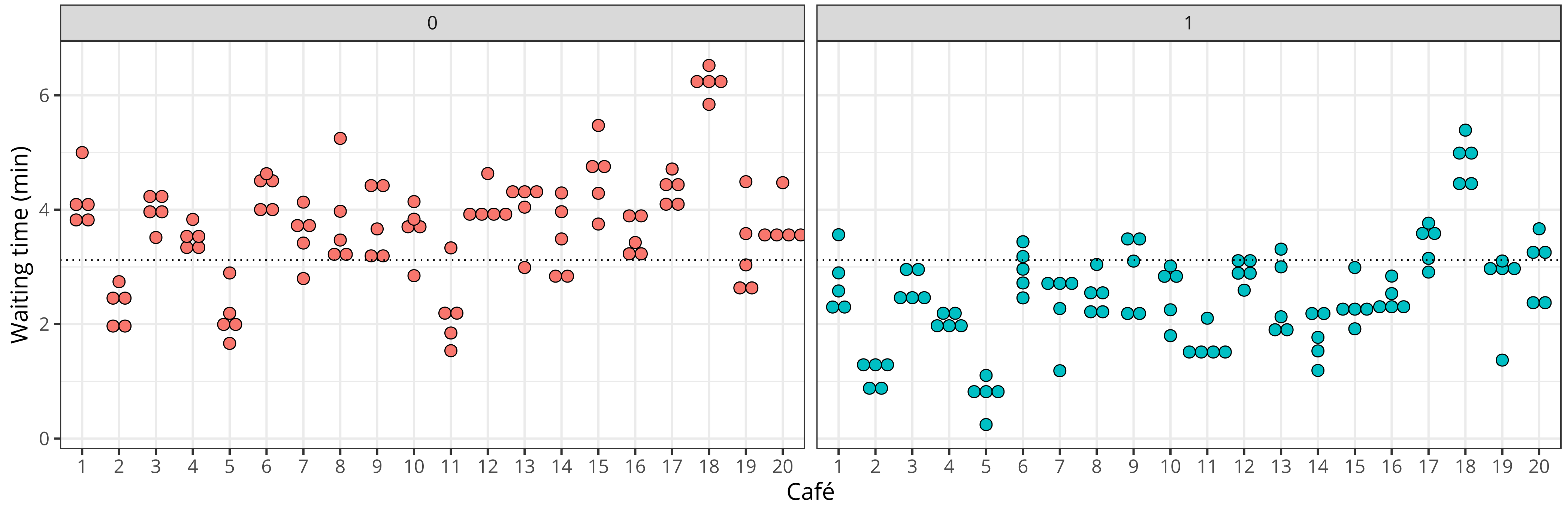
Coffee robot

```r
library(tidyverse)
library(imsb)

df <- open_data(robot)
head(x = df, n = 15)
```

```
   cafe afternoon      wait
1     1         0 4.9989926
2     1         1 2.2133944
3     1         0 4.1866730
4     1         1 3.5624399
5     1         0 3.9956779
6     1         1 2.8957176
7     1         0 3.7804582
8     1         1 2.3844837
9     1         0 3.8617982
10    1         1 2.5800004
11    2         0 2.7421223
12    2         1 1.3525907
13    2         0 2.5215095
14    2         1 0.9628102
15    2         0 1.9543977
```

Coffee robot

```r
df %>%
  ggplot(aes(x = factor(cafe), y = wait, fill = factor(afternoon) ) ) +
  geom_dotplot(
    stackdir = "center", binaxis = "y",
    dotsize = 1, show.legend = FALSE
    ) +
  geom_hline(yintercept = mean(df$wait), linetype = 3) +
  facet_wrap(~afternoon, ncol = 2) +
  labs(x = "Café", y = "Waiting time (min)")
```



Coffee robot, a first model

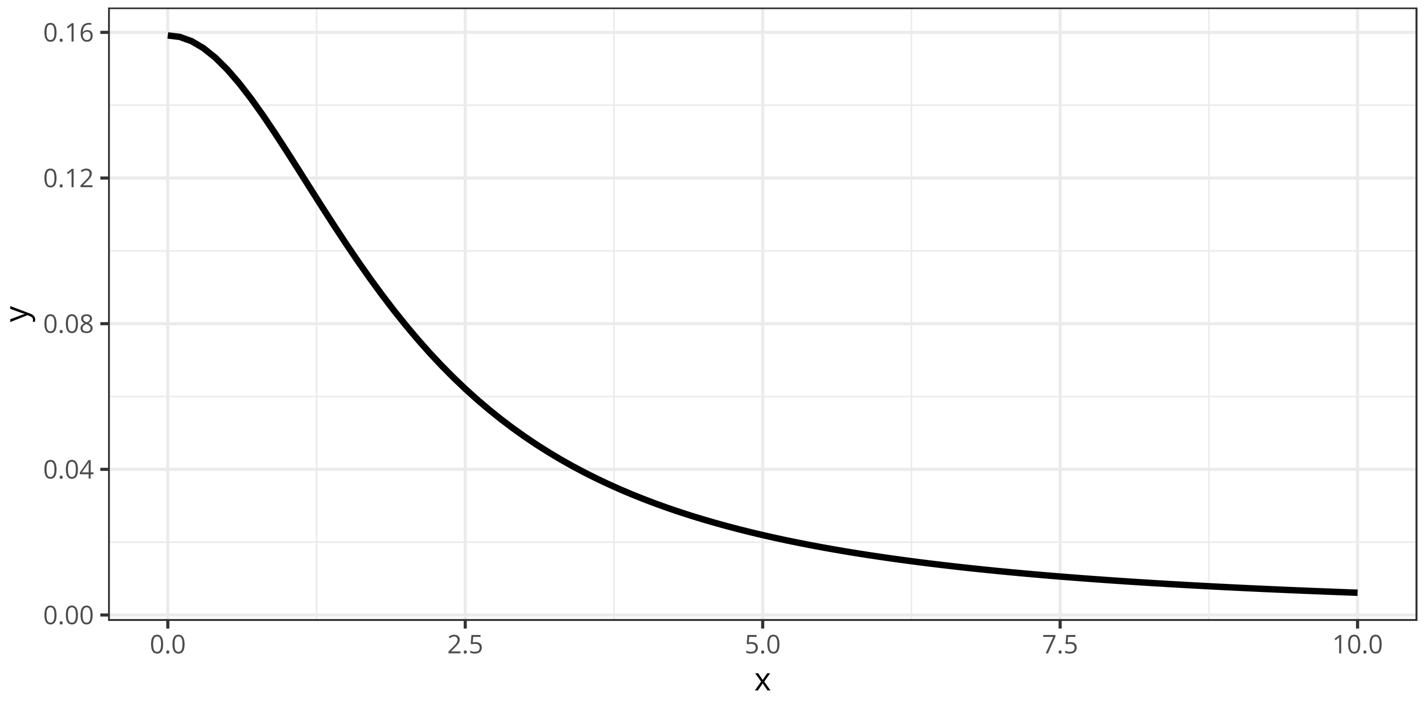
An initial model can be built, estimating the average time (across all cafés combined) to be served.

$$
\begin{aligned}
w_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha \\
\alpha &\sim \text{Normal}(5, 10) \\
\sigma &\sim \text{HalfCauchy}(0, 2)
\end{aligned}
$$

Half-Cauchy

$$p(x \mid x_0, \gamma) = \left(\pi\gamma\left[1 + \left(\frac{x - x_0}{\gamma}\right)^2\right]\right)^{-1}$$

```r
ggplot(data = data.frame(x = c(0, 10) ), aes(x = x) ) +
    stat_function(
        fun = dcauchy,
        args = list(location = 0, scale = 2), size = 1.5
        )
```



Coffee robot, a first model

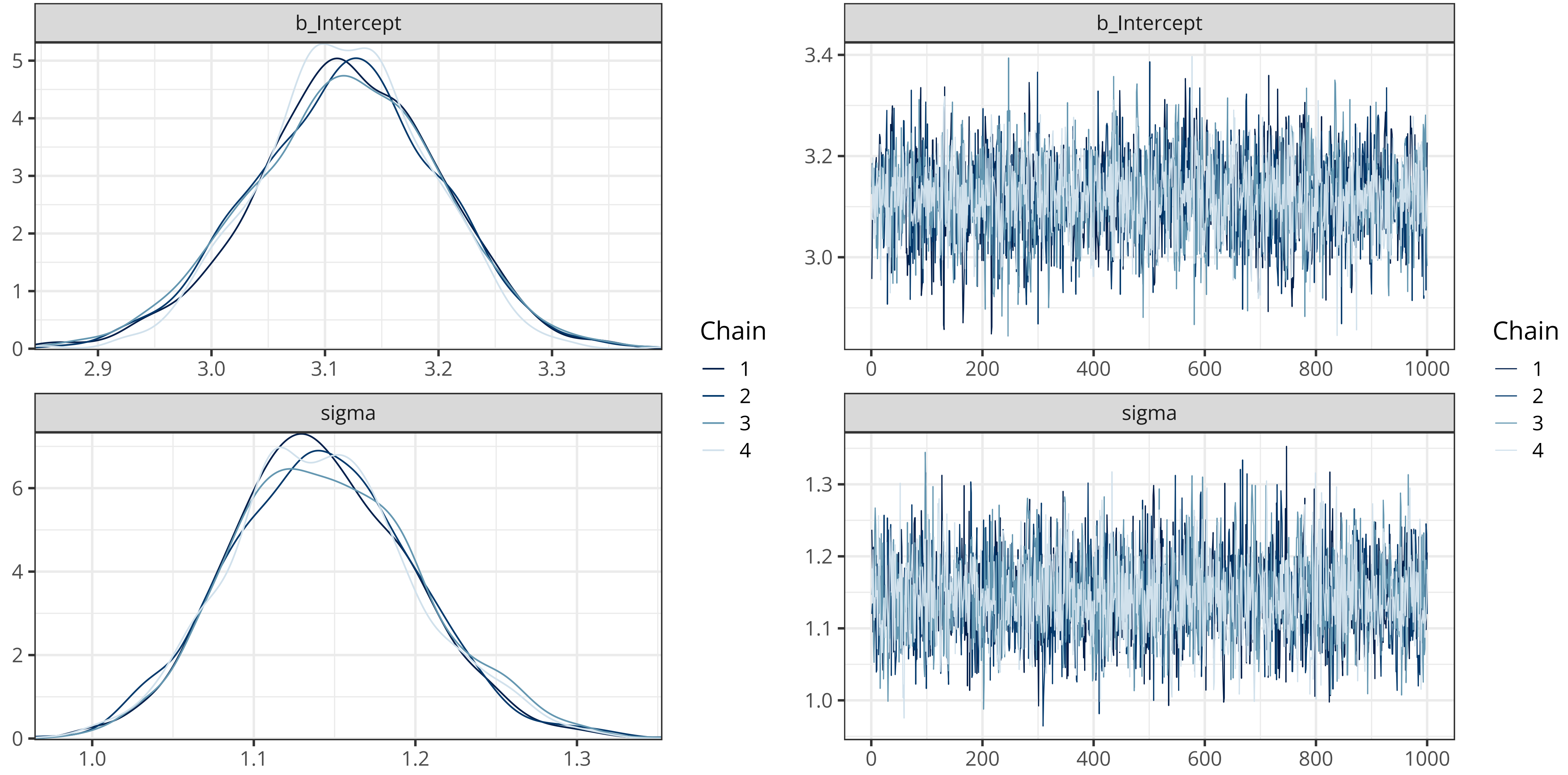
```r
library(brms)

mod1 <- brm(
  formula = wait ~ 1,
  prior = c(
    prior(normal(5, 10), class = Intercept),
    prior(cauchy(0, 2), class = sigma)
    ),
  data = df,
  cores = parallel::detectCores()
  )
```

```r
posterior_summary(x = mod1, probs = c(0.025, 0.975), pars = c("^b_", "sigma") )
```

```
            Estimate  Est.Error     Q2.5    Q97.5
b_Intercept 3.118251 0.08020407 2.956309 3.268246
sigma       1.143048 0.05675120 1.035605 1.259202
```

Diagnostic plot

```r
plot(x = mod1, combo = c("dens_overlay", "trace") )
```



One intercept per café

Second model which estimates one intercept per café. Equivalent to constructing 20 dummy variables.

$$
\begin{aligned}
w_i &\sim \mathrm{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha_{\text{café}[i]} \\
\alpha_{\text{café}[i]} &\sim \mathrm{Normal}(5, 10) \\
\sigma &\sim \mathrm{HalfCauchy}(0, 2)
\end{aligned}
$$

```r
mod2 <- brm(
  formula = wait ~ 0 + factor(cafe),
  prior = c(
    prior(normal(5, 10), class = b),
    prior(cauchy(0, 2), class = sigma)
    ),
  data = df,
  cores = parallel::detectCores()
  )
```

One intercept per café

```r
posterior_summary(x = mod2, pars = "^b_")
```

```
               Estimate Est.Error      Q2.5    Q97.5
b_factorcafe1  3.451085 0.2610853 2.9512768 3.974664
b_factorcafe2  1.724707 0.2618685 1.2119623 2.232001
b_factorcafe3  3.321328 0.2591069 2.8128314 3.818617
b_factorcafe4  2.795010 0.2491789 2.3113631 3.271451
b_factorcafe5  1.461022 0.2614662 0.9426324 1.981485
b_factorcafe6  3.637604 0.2580897 3.1300960 4.135650
b_factorcafe7  2.943423 0.2566098 2.4467521 3.447219
b_factorcafe8  3.167672 0.2568818 2.6596383 3.673230
b_factorcafe9  3.334906 0.2577492 2.8215052 3.834208
b_factorcafe10 3.096252 0.2613306 2.5830027 3.594380
b_factorcafe11 1.919792 0.2688887 1.4079115 2.432918
b_factorcafe12 3.487732 0.2518169 2.9864178 3.968451
b_factorcafe13 3.219461 0.2626960 2.7083647 3.729026
b_factorcafe14 2.630692 0.2653149 2.0847316 3.159034
b_factorcafe15 3.476996 0.2600783 2.9567007 3.996322
b_factorcafe16 3.005773 0.2583324 2.5112834 3.514806
b_factorcafe17 3.879522 0.2627024 3.3518362 4.399417
b_factorcafe18 5.527586 0.2586780 5.0194691 6.029890
b_factorcafe19 2.972815 0.2567997 2.4740427 3.487078
b_factorcafe20 3.364395 0.2704167 2.8278114 3.897341
```

Multilevel model

Couldn't we ensure that the time measured at café 1 **informs** the measurement taken at café 2 and café 3? As well as the average time taken to be served? We're going to learn the priors from the data...

$$
\begin{aligned}
\text{Level 1}: w_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha_{\text{café}[i]} \\
\text{Level 2}: \alpha_{\text{café}} &\sim \text{Normal}(\alpha, \sigma_{\text{café}}) \\
\alpha &\sim \text{Normal}(5, 10) \\
\sigma_{\text{café}} &\sim \text{HalfCauchy}(0, 2) \\
\sigma &\sim \text{HalfCauchy}(0, 2)
\end{aligned}
$$

The prior for the intercept of each coffee ($\alpha_{\text{café}}$) is now a function of two parameters (α and $\sigma_{\text{café}}$). α and $\sigma_{\text{café}}$ are called **hyper-parameters**, they are parameters for parameters, and their priors are called **hyperpriors**. There are two levels in the model...

Equivalences

$$
\begin{aligned}
w_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha_{\text{café}[i]} \\
\alpha_{\text{café}} &\sim \text{Normal}(\alpha, \sigma_{\text{café}})
\end{aligned}
$$

NB: α is defined here in the prior for $\alpha_{\text{café}}$ but it could also be defined in the linear model:

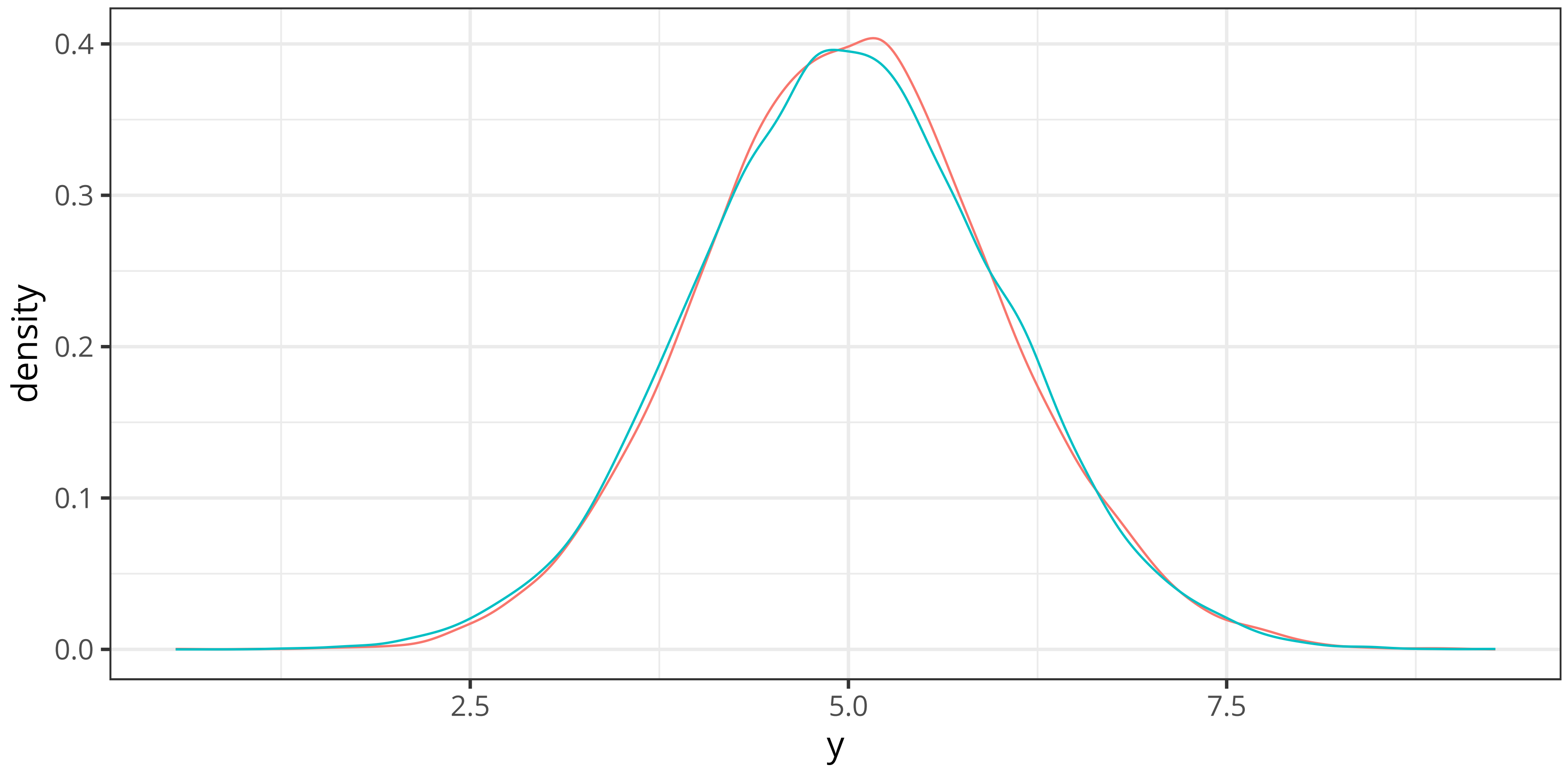
$$
\begin{aligned}
w_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \alpha_{\text{café}[i]} \\
\alpha_{\text{café}} &\sim \text{Normal}(0, \sigma_{\text{café}})
\end{aligned}
$$

We can always “remove” the mean from a Gaussian distribution and consider it as a constant plus a Gaussian centred on zero.

NB: when α is defined in the linear model, the $\alpha_{\text{café}}$ represent deviations from the mean intercept. It is therefore necessary to add α and $\alpha_{\text{café}}$ to obtain the average waiting time per café...

Equivalences

```r
y1 <- rnorm(n = 1e4, mean = 5, sd = 1)
y2 <- rnorm(n = 1e4, mean = 0, sd = 1) + 5

data.frame(y1 = y1, y2 = y2) %>%
    pivot_longer(cols = 1:2, names_to = "x", values_to = "y") %>%
    ggplot(aes(x = y, colour = x) ) +
    geom_density(show.legend = FALSE)
```

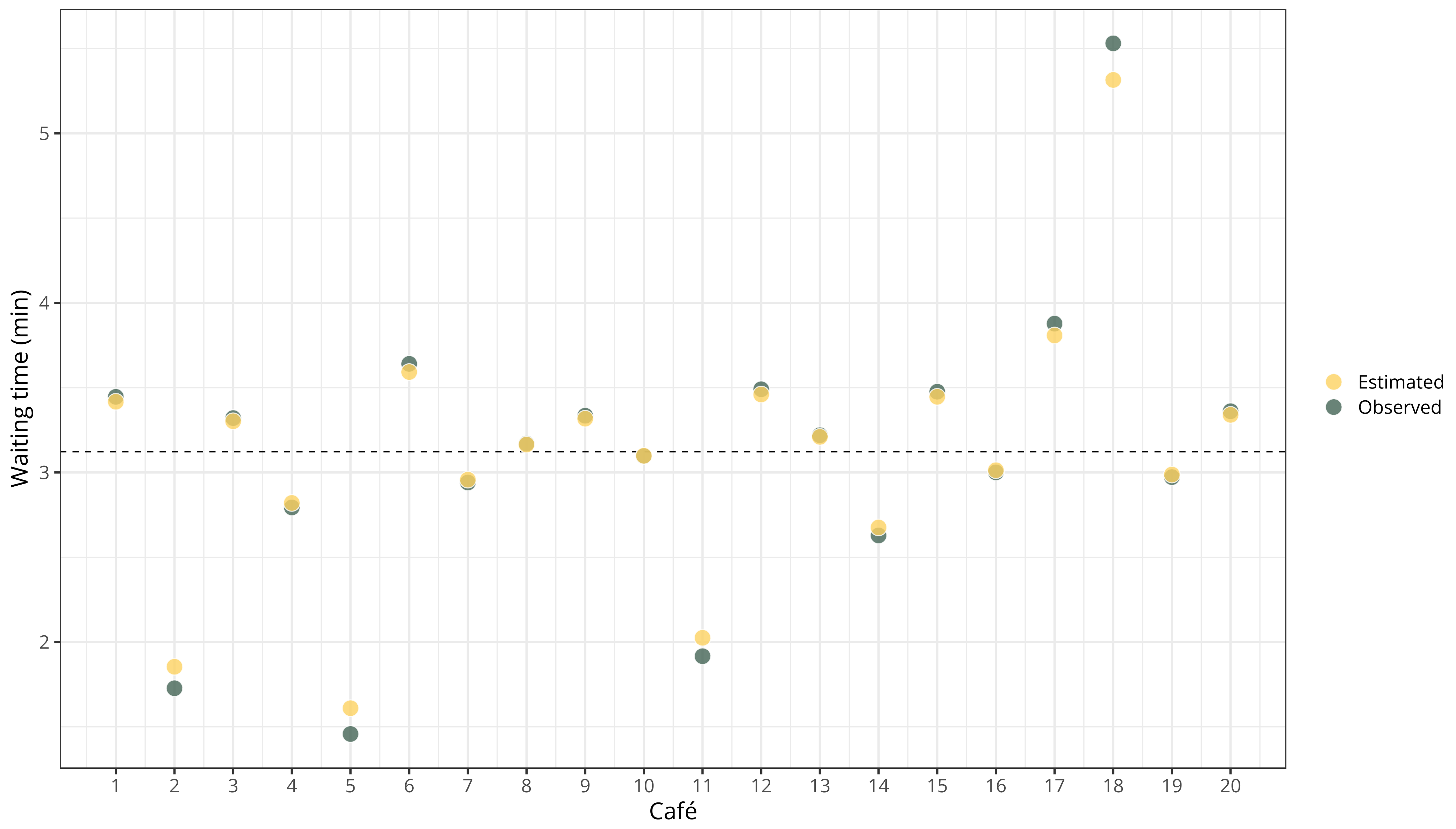


Multilevel model

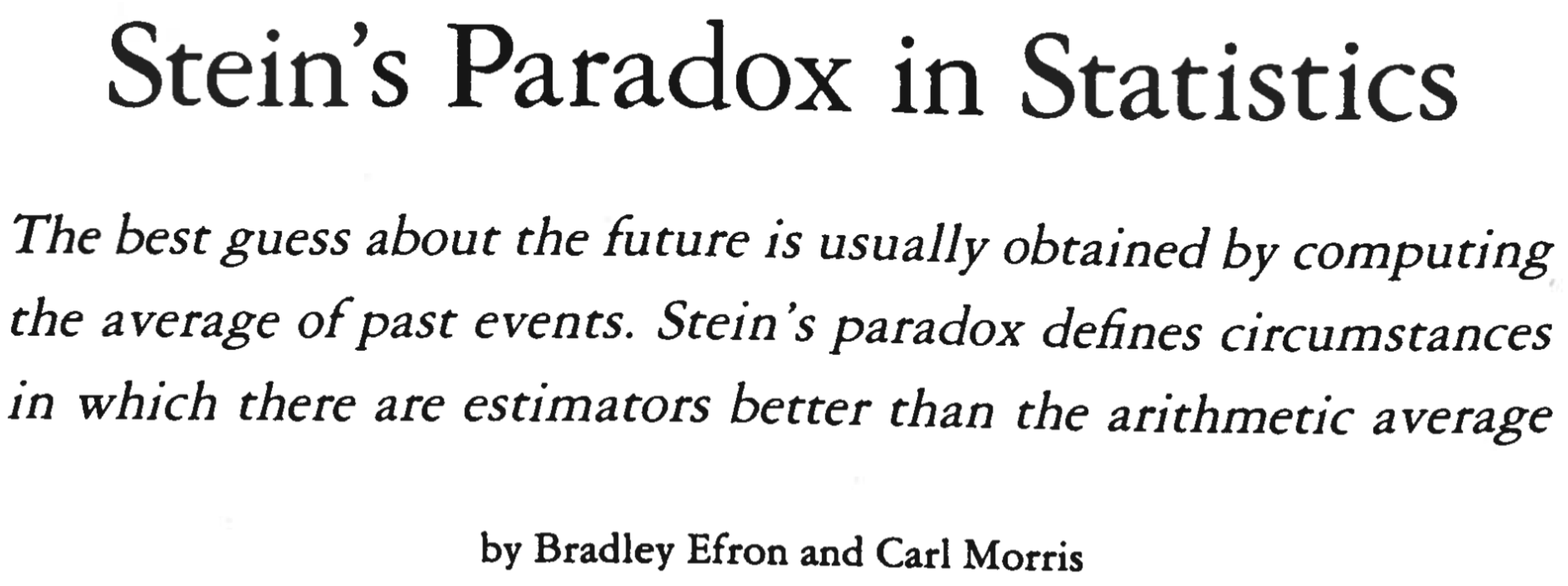
```r
mod3 <- brm(
  formula = wait ~ 1 + (1 | cafe),
  prior = c(
    prior(normal(5, 10), class = Intercept),
    prior(cauchy(0, 2), class = sigma),
    prior(cauchy(0, 2), class = sd)
    ),
  data = df,
  warmup = 1000, iter = 5000,
  cores = parallel::detectCores()
  )
```

This model has 23 parameters, the general intercept α, the residual variability σ, the variability between cafés and one intercept per café.

Shrinkage


Waiting time (min)
5
4
3
2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Café
Estimated
Observed

Shrinkage magic (Efron & Morris, 1977)


Stein's Paradox in Statistics

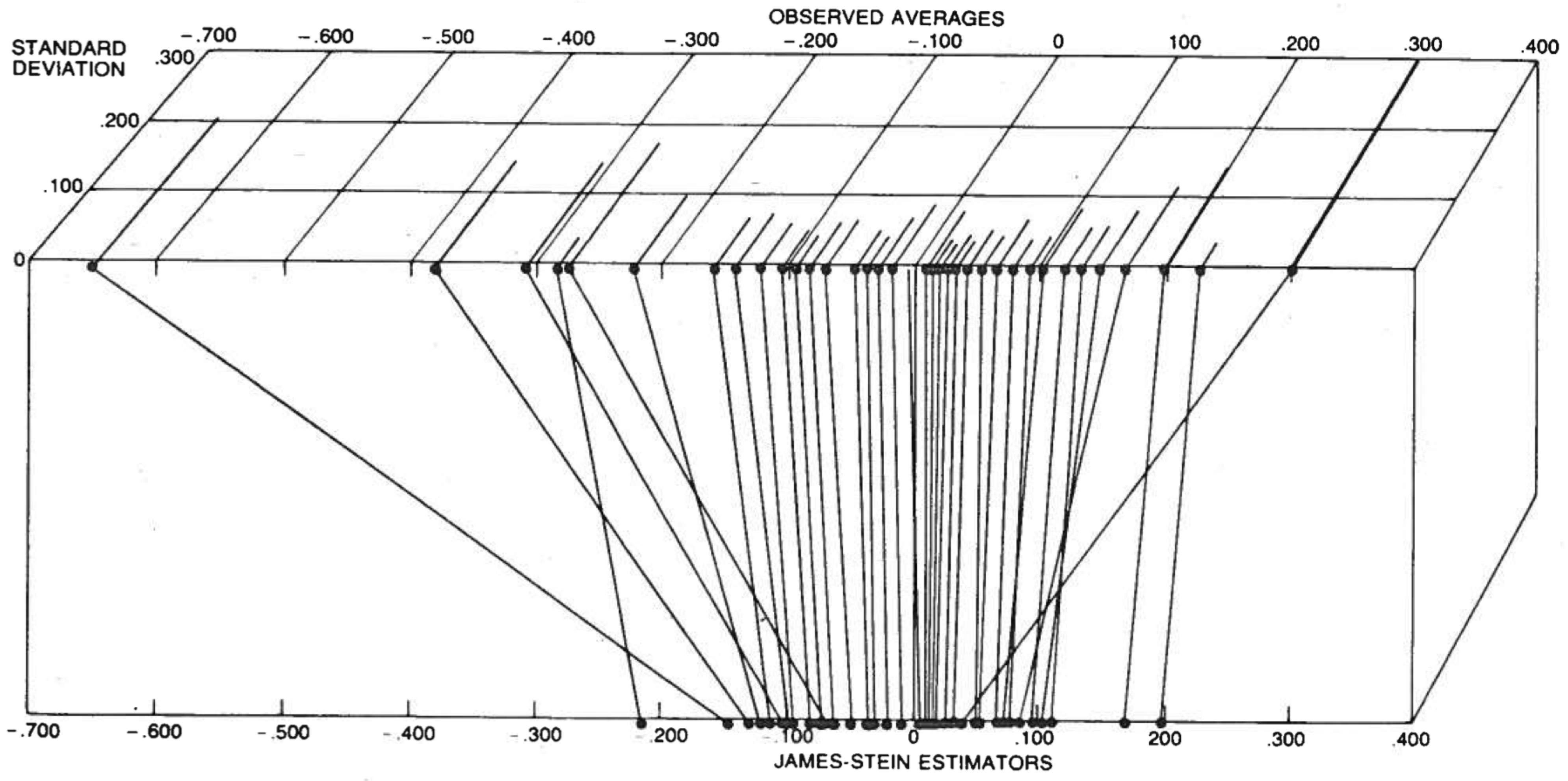
The best guess about the future is usually obtained by computing the average of past events. Stein's paradox defines circumstances in which there are estimators better than the arithmetic average

by Bradley Efron and Carl Morris

The James-Stein estimator is defined as $z = \bar{y} + c(y - \bar{y})$, where $\bar{y}$ is the sample mean, y is an individual observation, and c is a constant, the **shrinking factor** (Efron & Morris, 1977).

Shrinkage magic (Efron & Morris, 1977)

The shrinking factor is determined both by the variability (imprecision) of the measurement (e.g., its standard deviation) and by the distance to the mean estimate (i.e., $y - \bar{y}$). In other words, this estimator is less "confident" about (i.e., gives less weight to) imprecise and/or extreme observations. In practice, shrinkage acts as a safeguard against overlearning (overfitting).



Pooling

The **shrinkage** observed on the previous slide is due to information pooling between cafés. The estimate of the intercept for each café informs the intercept estimates of the other cafés, as well as the estimate of the general intercept (i.e., the overall average waiting time).

There are generally three perspectives (or strategies):

- **Complete pooling**: the waiting time is assumed to be invariant, a common intercept (`mod1`) is estimated.
- **No pooling**: it is assumed that each café's waiting time is unique and independent: an intercept is estimated for each café, but without informing the higher level (`mod2`).
- **Partial pooling**: an adaptive prior is used, as in the previous example (`mod3`).

The **complete pooling** strategy generally underfits the data (low predictive capacity) whereas the **no pooling** strategy amounts to overfitting the data (low predictive capacity here too). The **partial pooling** strategy (i.e., that of multilevel models) balances underfitting and overfitting.

Model comparison

We can compare these models using indices derived from information theory (extensions of AIC), such as the WAIC (the lower the better).

```r
# computing the WAIC for each model and storing it
mod1 <- add_criterion(mod1, "waic")
mod2 <- add_criterion(mod2, "waic")
mod3 <- add_criterion(mod3, "waic")

# comparing these WAICs
w <- loo_compare(mod1, mod2, mod3, criterion = "waic")
print(w, simplify = FALSE)
```

```
     elpd_diff se_diff elpd_waic se_elpd_waic p_waic se_p_waic waic   se_waic
mod3    0.0       0.0  -253.8       8.3         18.2     1.5     507.6   16.6
mod2   -0.7       1.3  -254.5       8.4         19.6     1.6     509.1   16.8
mod1  -57.3      10.6  -311.1      10.5          2.0     0.3     622.2   21.1
```

We note that model 3 has only 18 effective parameters (pWAIC) and fewer parameters than model 2, whereas it actually has 2 more... `posterior_summary(mod3)[3, 1]` gives us the sigma of the adaptive prior on $\alpha_{\text{café}}$ ($\sigma_{\text{café}} = 0.82$). Note that this sigma is very low and corresponds to assigning a very restrictive or **regularising** prior.

Model comparaison

We compare the estimates from the first (complete pooling) and third (partial pooling) model.

```r
posterior_summary(mod1, pars = c("^b", "sigma") )
```

```
             Estimate  Est.Error     Q2.5    Q97.5
b_Intercept 3.118251 0.08020407 2.956309 3.268246
sigma       1.143048 0.05675120 1.035605 1.259202
```

```r
posterior_summary(mod3, pars = c("^b", "sigma") )
```

```
             Estimate Est.Error      Q2.5     Q97.5
b_Intercept 3.1226606 0.2075211 2.7209592 3.5347898
sigma       0.8221927 0.0440647 0.7409711 0.9137384
```

Both models make the same prediction (on average) for α, but model 3 is more uncertain of its prediction than model 1 (see the standard error for α)…

The σ estimate of model 3 is smaller than that of model 1 because model 3 **decomposes** the unexplained variability into two sources: variability in waiting time between cafés and the residual variability σ.

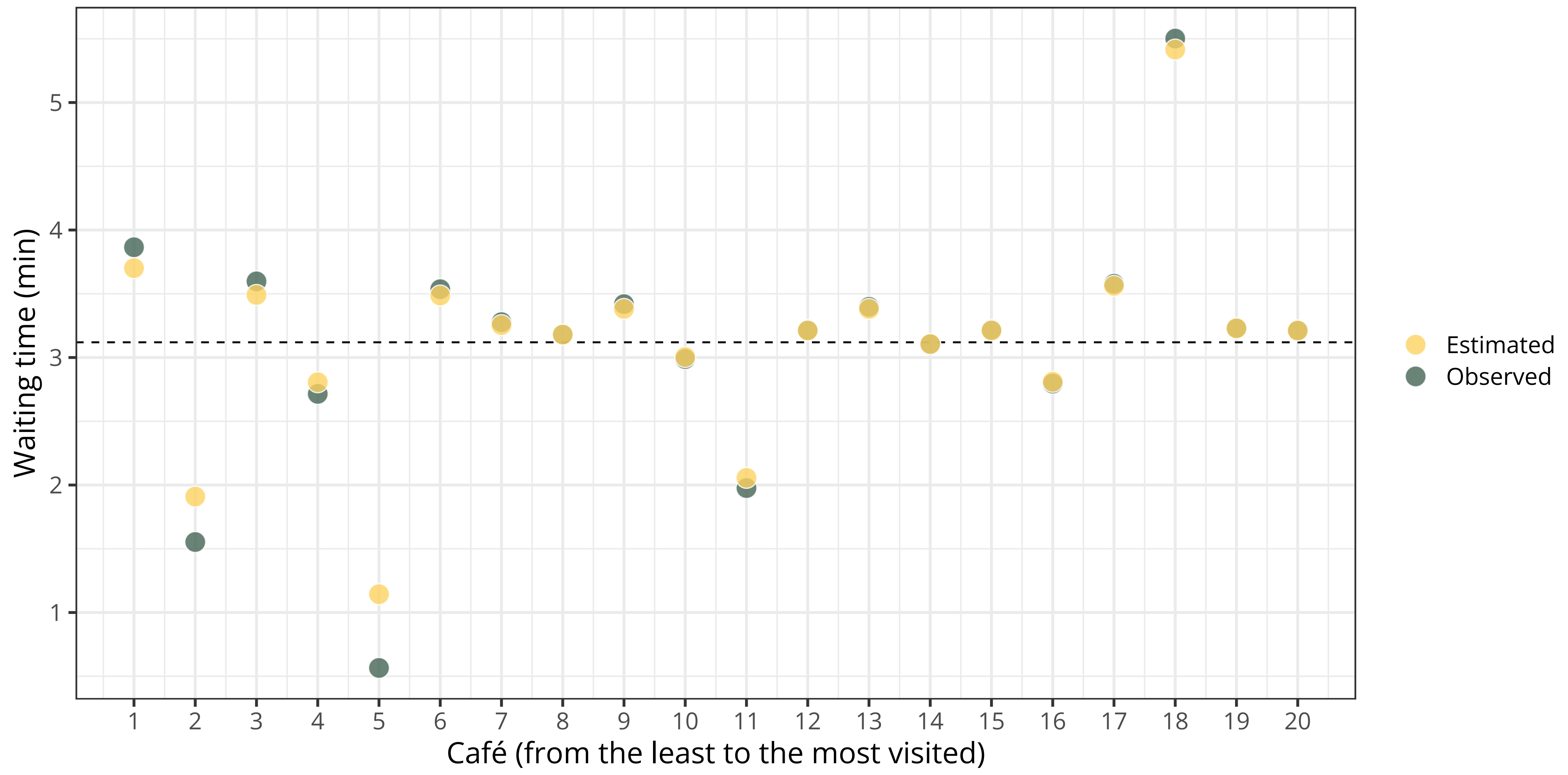
Coffee robot

Let's assume that our robot doesn't visit all the cafés the same number of times (as in the previous case) but that it visits more often the cafés close to home...

```r
df2 <- open_data(robot_unequal) # new dataset

mod4 <- brm(
  formula = wait ~ 1 + (1 | cafe),
  prior = c(
    prior(normal(5, 10), class = Intercept),
    prior(cauchy(0, 2), class = sigma),
    prior(cauchy(0, 2), class = sd)
    ),
  data = df2,
  warmup = 1000, iter = 5000,
  cores = parallel::detectCores()
  )
```

Shrinkage

We can see that cafés that are visited frequently (right) are less affected by the effect of **shrinkage**. Their estimates are less “pulled” towards the average than the estimates of the least frequently visited cafés (left).



Aparté: fixed and random effects

Five (contradictory) definitions identified by Gelman (2005).

- Fixed effects are constant across individuals, and random effects vary.
- Effects are fixed if they are interesting in themselves or random if there is interest in the underlying population.
- When a sample exhausts the population, the corresponding variable is fixed; when the sample is a small (i.e., negligible) part of the population the corresponding variable is random.
- If an effect is assumed to be a realized value of a random variable, it is called a random effect.
- Fixed effects are estimated using least squares (or, more generally, maximum likelihood) and random effects are estimated with shrinkage.

Gelman & Hill (2006) suggest instead the use of the terms **constant effcts** and **varying effects**, and to always use multilevel modelling, considering that the so-called **fixed effect** can simply be considered as a **random effect** whose variance would be equal to 0 (see also Nalborczyk et al., 2019).

Regularisation and terminology

Varying the intercepts for each café is simply another way of (adaptively) regularising, that is, reducing the weight given to the data in the estimation. The model becomes able to estimate the extent to which the groups (in this case the cafés) are different, while estimating the characteristics of each café…

Difference between **cross-classified** (or "crossed") multilevel models and **nested or hierarchical** multilevel models. Cross-classified models refer to data structured according to two (or more) non-nested random factors. Hierarchical models usually refers to hierarchically structured data (e.g., a student in a class in a school in a city…). See this discussion for more details.

However, the two types of models are written in a similar way, on several "levels". The term "multilevel" (in our terminology) therefore refers to the structure of the model, to its specification. It is distinct from the structure of the data.

Coffee robot: varying intercept + varying slope

We are now interested in the effect of the time of day on the waiting time. Do we wait more in the morning, or in the afternoon?

$$
\begin{aligned}
w_i &\sim \mathrm{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha_{\text{café}[i]} + \beta_{\text{café}[i]} \times A_i
\end{aligned}
$$

Where A_i is a dummy variable coded 0/1 for morning/afternoon and where $\beta_{\text{café}}$ is therefore a difference parameter (i.e., a slope) between morning and afternoon.

Note: we know that cafés have intercepts and slopes that co-vary... Popular cafés will be overcrowded in the morning and much less in the afternoon, resulting in a negative slope. These cafés will also have a longer average waiting time (i.e., a larger intercept). In these cafés, α is large and β is far from zero. Conversely, in an unpopular café, the waiting time will be short, as well as the difference between the morning and afternoon's waiting time.

We could therefore use the co-variation between the intercept and slope to make better inferences. In other words, ensure that the estimate of the intercept informs the estimate of the slope, and reciprocally.

Coffee robot: varying intercept + varying slope

We are now interested in the effect of the time of day on the waiting time. Do we wait more in the morning, or in the afternoon?

$$
\begin{aligned}
w_i &\sim \mathrm{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha_{\text{café}[i]} + \beta_{\text{café}[i]} \times A_i \\
\begin{bmatrix} \alpha_{\text{café}} \\ \beta_{\text{café}} \end{bmatrix} &\sim \mathrm{MVNormal}\left(\begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \mathbf{S} \right)
\end{aligned}
$$

The third line posits that every café has an intercept $\alpha_{\text{café}}$ and a slope $\beta_{\text{café}}$, defined by a bivariate (i.e., two-dimensional) Gaussian prior having as means α and β and as covariance matrix $\mathbf{S}$.

Aparté: multivariate Normal distribution

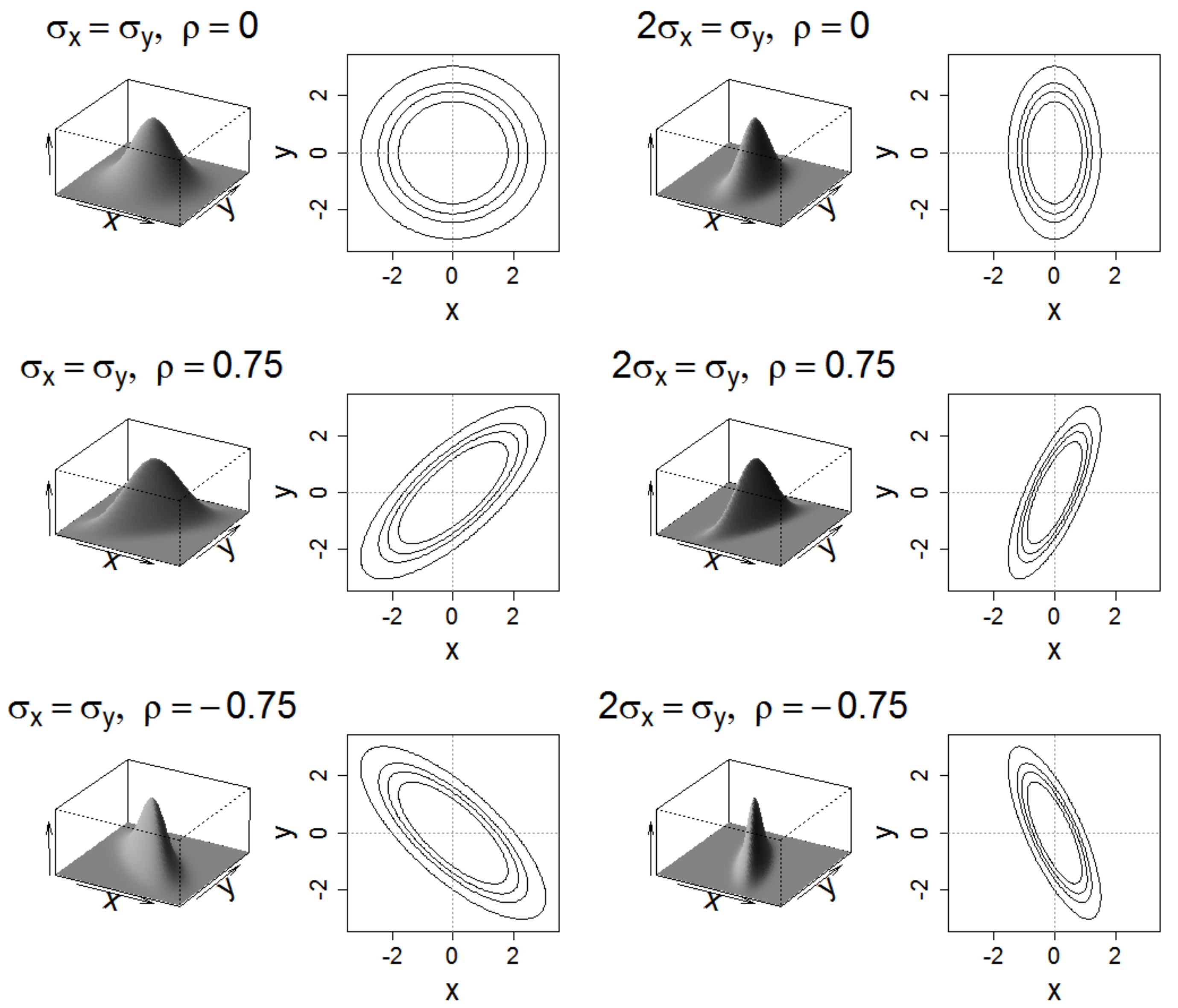
$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Where $\boldsymbol{\mu}$ is a (k-dimensional) vector of means, for instance: `mu <- c(a, b)`.

$\boldsymbol{\Sigma}$ is a covariance matrix of $k \times k$ dimensions, and which corresponds to the matrix given by the function `vcov()`.

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{\alpha}^{2} & \sigma_{\alpha}\sigma_{\beta}\rho \\ \sigma_{\alpha}\sigma_{\beta}\rho & \sigma_{\beta}^{2} \end{pmatrix}$$

Aparté: multivariate Normal distribution



Aparté: multivariate Normal distribution

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{\alpha}^{2} & \sigma_{\alpha}\sigma_{\beta}\rho \\ \sigma_{\alpha}\sigma_{\beta}\rho & \sigma_{\beta}^{2} \end{pmatrix}$$

This matrix can be constructed in two different ways, strictly equivalent.

```r
sigma_a <- 1
sigma_b <- 0.75
rho <- 0.7
cov_ab <- sigma_a * sigma_b * rho
(Sigma1 <- matrix(c(sigma_a^2, cov_ab, cov_ab, sigma_b^2), ncol = 2) )
```

```
      [,1]   [,2]
[1,] 1.000 0.5250
[2,] 0.525 0.5625
```

Aparté: multivariate Normal distribution

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_\alpha^2 & \sigma_\alpha \sigma_\beta \rho \\ \sigma_\alpha \sigma_\beta \rho & \sigma_\beta^2 \end{pmatrix}$$

The second method is convenient because it considers separately the standard deviations and correlations.

```r
(sigmas <- c(sigma_a, sigma_b) ) # standard deviations
```

```
[1] 1.00 0.75
```

```r
(Rho <- matrix(c(1, rho, rho, 1), nrow = 2) ) # correlation matrix
```

```
     [,1] [,2]
[1,]  1.0  0.7
[2,]  0.7  1.0
```

```r
(Sigma2 <- diag(sigmas) %*% Rho %*% diag(sigmas) )
```

```
      [,1]   [,2]
[1,] 1.000 0.5250
[2,] 0.525 0.5625
```

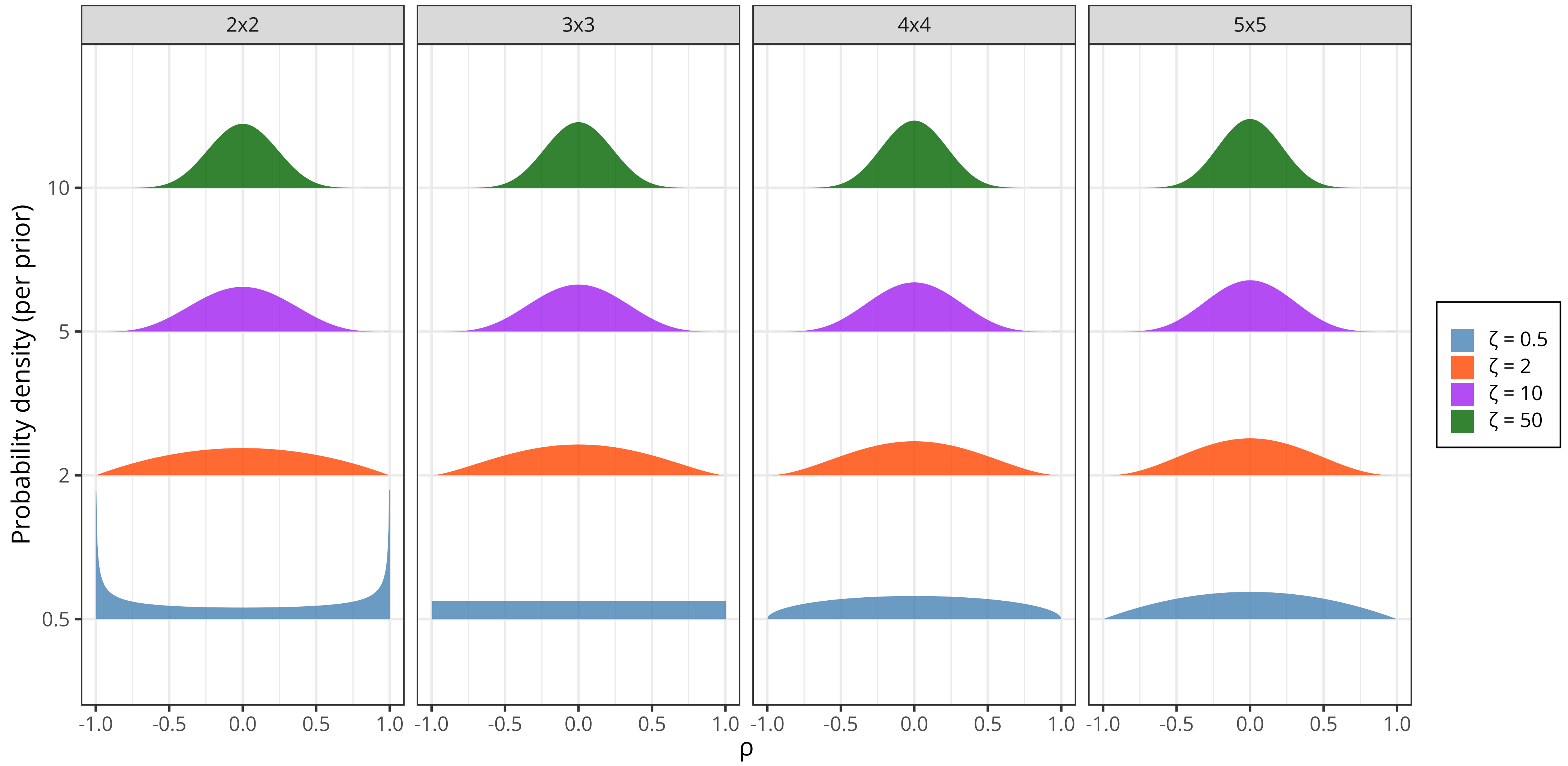
Coffee robot: varying intercept + varying slope

$$
\begin{aligned}
w_i &\sim \mathrm{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha_{\text{café}[i]} + \beta_{\text{café}[i]} \times A_i \\
\begin{bmatrix} \alpha_{\text{café}} \\ \beta_{\text{café}} \end{bmatrix} &\sim \mathrm{MVNormal}\left(\begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \mathbf{S} \right) \\
\mathbf{S} &= \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \mathbf{R} \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \\
\alpha &\sim \mathrm{Normal}(0, 10) \\
\beta &\sim \mathrm{Normal}(0, 10) \\
\sigma_\alpha &\sim \mathrm{HalfCauchy}(0, 2) \\
\sigma_\beta &\sim \mathrm{HalfCauchy}(0, 2) \\
\sigma &\sim \mathrm{HalfCauchy}(0, 2) \\
\mathbf{R} &\sim \mathrm{LKJ}(2)
\end{aligned}
$$

$\mathbf{S}$ is defined by factoring σ_α, σ_β, and the correlation matrix $\mathbf{R}$. The next lines of the model simply define priors for constant effects. The last line specifies the prior for $\mathbf{R}$.

LKJ prior

Prior proposed by Lewandowski et al. (2009). A single parameter ζ (zeta) specifies the concentration of the distribution of the correlation coefficient. The $\mathrm{LKJ}(2)$ prior defines an weakly informative prior for ρ (rho) which is sceptical of extreme correlations (i.e., values close to -1 or 1).



Syntax reminders

The `brms` package uses the same syntax as R base functions (like `lm`) or the `lme4` package.

```r
Reaction ~ Days + (1 + Days | Subject)
```

The left-hand side defines the dependent variable (or "outcome", i.e., what we are trying to predict).

The right-hand side defines the predictors. The intercept is usually implied, so the two formulations below are equivalent.

```r
Reaction ~ Days + (1 + Days | Subject)
Reaction ~ 1 + Days + (1 + Days | Subject)
```

Syntax reminders

The first part of the right-hand side of the formula represents the constant effects (fixed effects), whereas the second part (between parentheses) represents varying effects (random effects).

```
Reaction ~ 1 + Days + (1 | Subject)
Reaction ~ 1 + Days + (1 + Days | Subject)
```

The first model above contains only a varying intercept, which varies by `Subject`. The second model contains a varying intercept, but also a varying slope for the effect of `Days`.

Syntax reminders

When including several varying effects (e.g., an intercept and a slope), `brms` assumes that we also want to estimate the correlation between these effects. Otherwise, we can remove this correlation (i.e., set it to 0) using `||`.

```
Reaction ~ Days + (1 + Days || Subject)
```

Previous models assumed a Gaussian generative model. This assumption can be changed easily by specifying the desired function via the `family` argument.

```
brm(formula = Reaction ~ 1 + Days + (1 + Days | Subject), family = lognormal() )
```

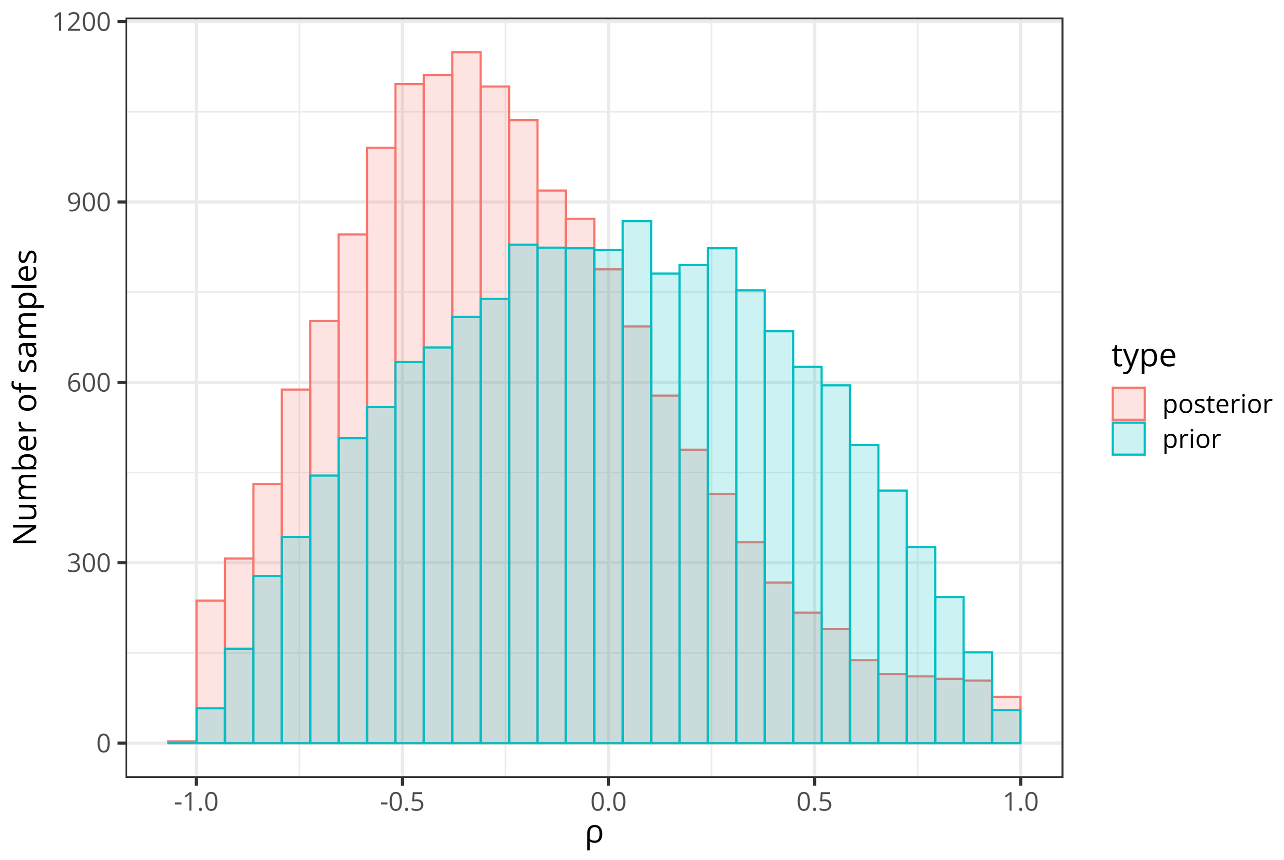
Implementation of our model via brms

We specify an intercept and a slope (for the `afternoon` effect) which vary by `cafe`.

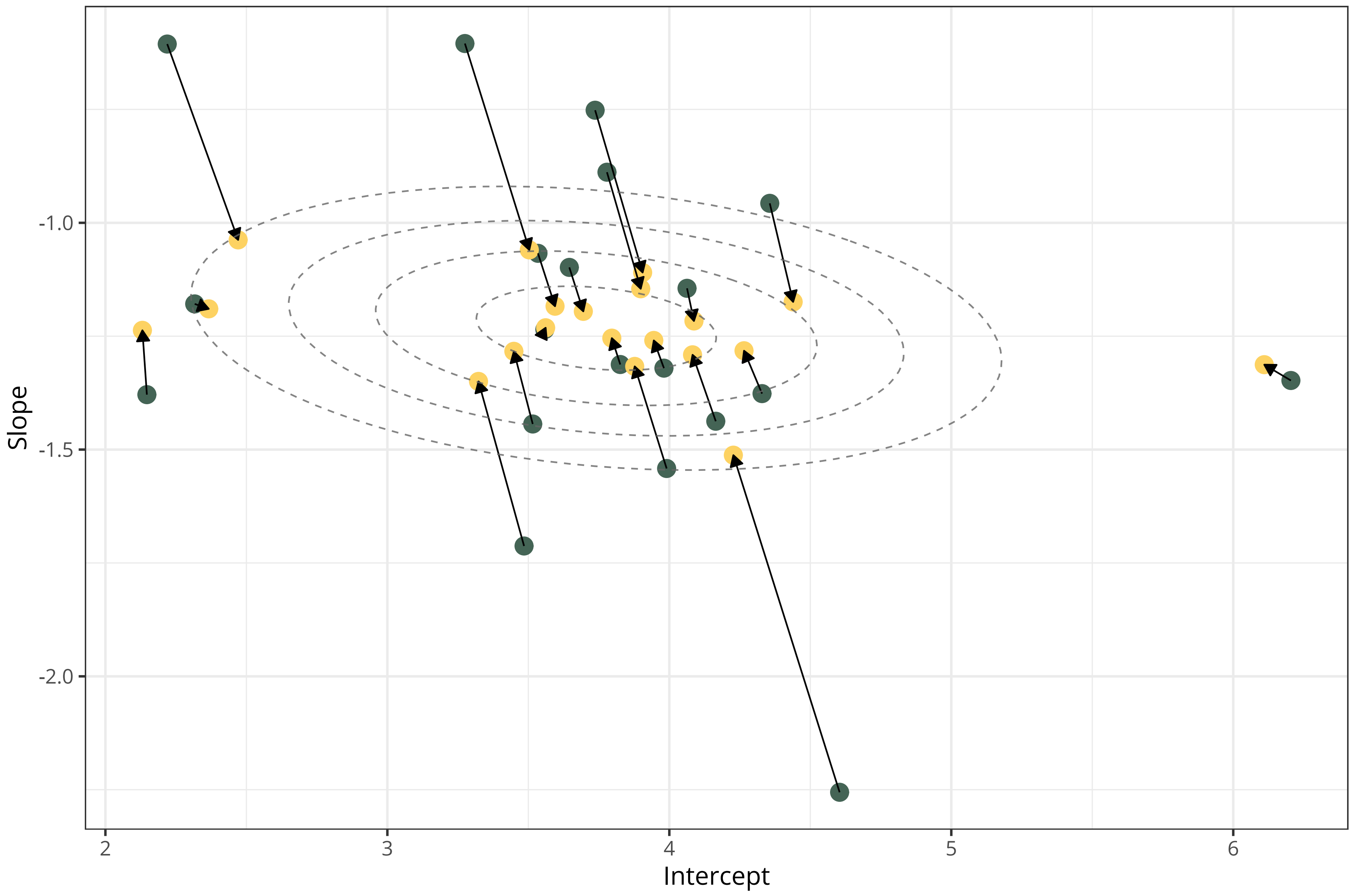
```r
mod5 <- brm(
  formula = wait ~ 1 + afternoon + (1 + afternoon | cafe),
  prior = c(
    prior(normal(0, 10), class = Intercept),
    prior(normal(0, 10), class = b),
    prior(cauchy(0, 2), class = sigma),
    prior(cauchy(0, 2), class = sd)
    ),
  data = df,
  warmup = 1000, iter = 5000,
  cores = parallel::detectCores()
  )
```

Posterior distribution

```r
post <- as_draws_df(x = mod5) # extracting posterior samples
R <- rethinking::rlkjcorr(n = 16000, K = 2, eta = 2) # samples from prior

data.frame(prior = R[, 1, 2], posterior = post$cor_cafe__Intercept__afternoon) %>%
    gather(type, value, prior:posterior) %>%
    ggplot(aes(x = value, color = type, fill = type) ) +
    geom_histogram(position = "identity", alpha = 0.2) +
    labs(x = expression(rho), y = "Number of samples")
```



Two-dimensional shrinkage



Model comparaison

We compare the first model (complete pooling model), the third model (partial pooling model), and the last model (with varying intercept and slope).

```r
mod5 <- add_criterion(mod5, "waic")
w <- loo_compare(mod1, mod2, mod3, mod5, criterion = "waic")
print(w, simplify = FALSE)
```

```
     elpd_diff se_diff elpd_waic se_elpd_waic p_waic se_p_waic waic   se_waic
mod5    0.0       0.0  -155.1      10.0         26.5     2.6     310.2   20.1
mod3  -98.7       8.3  -253.8       8.3         18.2     1.5     507.6   16.6
mod2  -99.5       8.3  -254.5       8.4         19.6     1.6     509.1   16.8
mod1 -156.0      13.7  -311.1      10.5          2.0     0.3     622.2   21.1
```

```r
model_weights(mod1, mod2, mod3, mod5, weights = "waic")
```

```
        mod1         mod2         mod3         mod5 
1.763502e-68 6.397539e-44 1.337590e-43 1.000000e+00 
```

Model comparaison

The estimate of the average waiting time is more uncertain when we takes into account new sources of error. However, the overall error of the model (i.e., what is not explained), the residual variation σ, decreases...

```r
posterior_summary(mod1, pars = c("^b", "sigma") )
```

```
             Estimate  Est.Error     Q2.5    Q97.5
b_Intercept 3.118251 0.08020407 2.956309 3.268246
sigma       1.143048 0.05675120 1.035605 1.259202
```

```r
posterior_summary(mod3, pars = c("^b", "sigma") )
```

```
             Estimate Est.Error      Q2.5     Q97.5
b_Intercept 3.1226606 0.2075211 2.7209592 3.5347898
sigma       0.8221927 0.0440647 0.7409711 0.9137384
```

```r
posterior_summary(mod5, pars = c("^b", "sigma") )
```

```
             Estimate  Est.Error       Q2.5      Q97.5
b_Intercept  3.740964 0.21779934  3.3144648  4.1799312
b_afternoon -1.232429 0.08622682 -1.4041576 -1.0647728
sigma        0.489634 0.02755458  0.4379875  0.5474008
```

Conclusions

Multilevel models (or “mixed-effects models”) are natural extensions of classical (single-level) regression models, where classical parameters are themselves assigned “models”, governed by hyper-parameters.

This extension makes it possible to make more precise predictions by taking into account the variability related to groups or structures (clusters) present in the data. In other words, by modelling the populations from which the varying effects are drawn (e.g., the population of participants or stimuli).

A single-level regression model is equivalent to a multilevel model where the variability of varying effects would be fixed at 0.

The Bayesian framework allows a natural interpretation of distributions from which the varying effects come. Indeed, these distributions can be interpreted as prior distributions, whose parameters are estimated from the data.

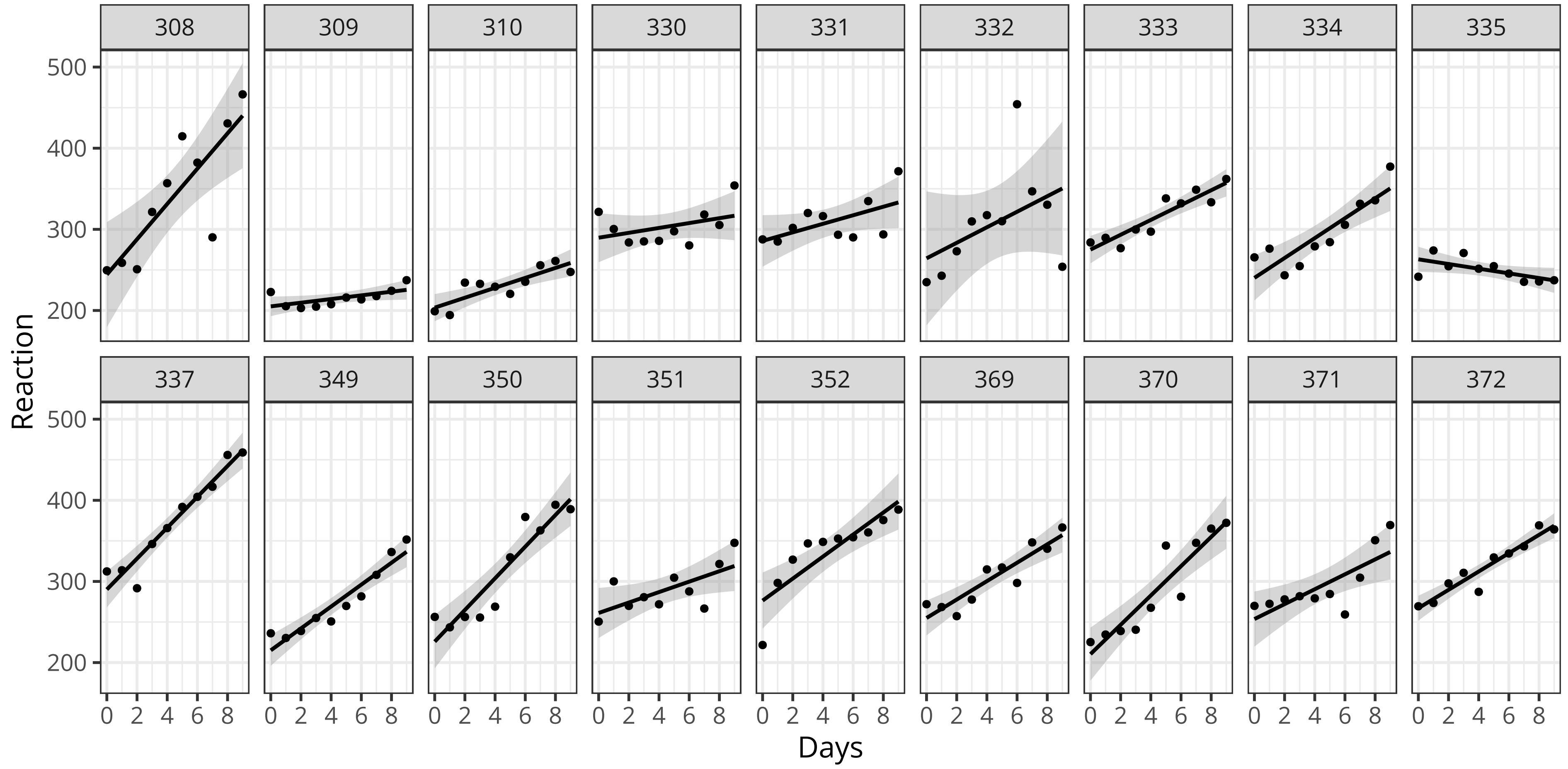
Practical work - sleepstudy

```r
library(lme4)
data(sleepstudy)
head(sleepstudy, 20)
```

```
   Reaction Days Subject
1  249.5600    0     308
2  258.7047    1     308
3  250.8006    2     308
4  321.4398    3     308
5  356.8519    4     308
6  414.6901    5     308
7  382.2038    6     308
8  290.1486    7     308
9  430.5853    8     308
10 466.3535    9     308
11 222.7339    0     309
12 205.2658    1     309
13 202.9778    2     309
14 204.7070    3     309
15 207.7161    4     309
16 215.9618    5     309
17 213.6303    6     309
18 217.7272    7     309
19 224.2957    8     309
20 237.3142    9     309
```

Practical work - sleepstudy

```
sleepstudy %>%
    ggplot(aes(x = Days, y = Reaction) ) +
    geom_smooth(method = "lm", colour = "black") +
    geom_point() +
    facet_wrap(~Subject, nrow = 2) +
    scale_x_continuous(breaks = c(0, 2, 4, 6, 8) )
```



Practical work - sleepstudy

It's up to you to build the mathematical models and `brms` models corresponding to the following models:

- A model with only the fixed effect of `Days`.
- A model with the fixed effect of `Days` + a random effect of `Subject` (varying intercept).
- A model with the fixed effect of `Days` + a random effect of `Subject` (varying intercept + varying slope for `Days`).

Then, compare these models using model comparison tools and conclude.

Proposed solution

```r
# frequentist (flat-priors) models
fmod0 <- lm(Reaction ~ Days, sleepstudy)
fmod1 <- lmer(Reaction ~ Days + (1 | Subject), sleepstudy)
fmod2 <- lmer(Reaction ~ Days + (1 + Days | Subject), sleepstudy)

# comparing fmod1 and fmod2
anova(fmod1, fmod2)
```

```
Data: sleepstudy
Models:
fmod1: Reaction ~ Days + (1 | Subject)
fmod2: Reaction ~ Days + (1 + Days | Subject)
      npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
fmod1    4 1802.1 1814.8 -897.04   1794.1
fmod2    6 1763.9 1783.1 -875.97   1751.9 42.139  2  7.072e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Proposed solution

```r
mod6 <- brm(
  Reaction ~ 1 + Days,
  prior = c(
    prior(normal(200, 100), class = Intercept),
    prior(normal(0, 10), class = b),
    prior(cauchy(0, 10), class = sigma)
    ),
  data = sleepstudy,
  warmup = 1000, iter = 5000,
  cores = parallel::detectCores()
  )
```

```r
posterior_summary(mod6)
```

```
              Estimate Est.Error         Q2.5       Q97.5
b_Intercept  251.88070  6.536260   239.019687   264.56435
b_Days        10.32575  1.222436     7.927364    12.71458
sigma         47.81060  2.545711    43.187711    53.16668
lprior       -15.69428  0.164818   -16.046503   -15.39603
lp__        -963.46880  1.225855  -966.580751  -962.07970
```

Proposed solution

```r
mod7 <- brm(
  Reaction ~ 1 + Days + (1 | Subject),
  prior = c(
    prior(normal(200, 100), class = Intercept),
    prior(normal(0, 10), class = b),
    prior(cauchy(0, 10), class = sigma),
    prior(cauchy(0, 10), class = sd)
    ),
  data = sleepstudy,
  warmup = 1000, iter = 5000,
  cores = parallel::detectCores()
  )
```

```r
posterior_summary(mod7, pars = c("^b", "sigma") )
```

```
             Estimate  Est.Error       Q2.5     Q97.5
b_Intercept 250.74661 10.0623899 230.812846 270.52849
b_Days       10.39303  0.8097478   8.790013  11.98756
sigma        31.07933  1.7373042  27.919034  34.68583
```

Proposed solution

```r
mod8 <- brm(
  Reaction ~ 1 + Days + (1 + Days | Subject),
  prior = c(
    prior(normal(200, 100), class = Intercept),
    prior(normal(0, 10), class = b),
    prior(cauchy(0, 10), class = sigma),
    prior(cauchy(0, 10), class = sd)
    ),
  data = sleepstudy,
  warmup = 1000, iter = 5000,
  cores = parallel::detectCores()
  )
```

```r
posterior_summary(mod8, pars = c("^b", "sigma") )
```

```
             Estimate Est.Error       Q2.5      Q97.5
b_Intercept 251.11727  7.042817 237.316651 265.01693
b_Days       10.06105  1.682028   6.654157  13.31108
sigma        25.85012  1.554566  22.983570  29.11508
```

Proposed solution

```r
# computing and storing the WAIC of each model
mod6 <- add_criterion(mod6, "waic")
mod7 <- add_criterion(mod7, "waic")
mod8 <- add_criterion(mod8, "waic")

# comparing the WAICs of these models
w <- loo_compare(mod6, mod7, mod8, criterion = "waic")
print(w, simplify = FALSE)
```

```
     elpd_diff se_diff elpd_waic se_elpd_waic p_waic se_p_waic waic   se_waic
mod8    0.0       0.0   -860.0      22.2         32.5     8.1    1720.0   44.4 
mod7  -24.7      11.5   -884.7      14.4         19.2     3.3    1769.4   28.8 
mod6  -93.3      20.8   -953.3      10.6          3.2     0.5    1906.6   21.1 
```

```r
# computing the relative weight of each model
model_weights(mod6, mod7, mod8, weights = "waic")
```

```
        mod6         mod7         mod8 
3.025212e-41 1.838528e-11 1.000000e+00 
```

Scientific and cognitive modelling

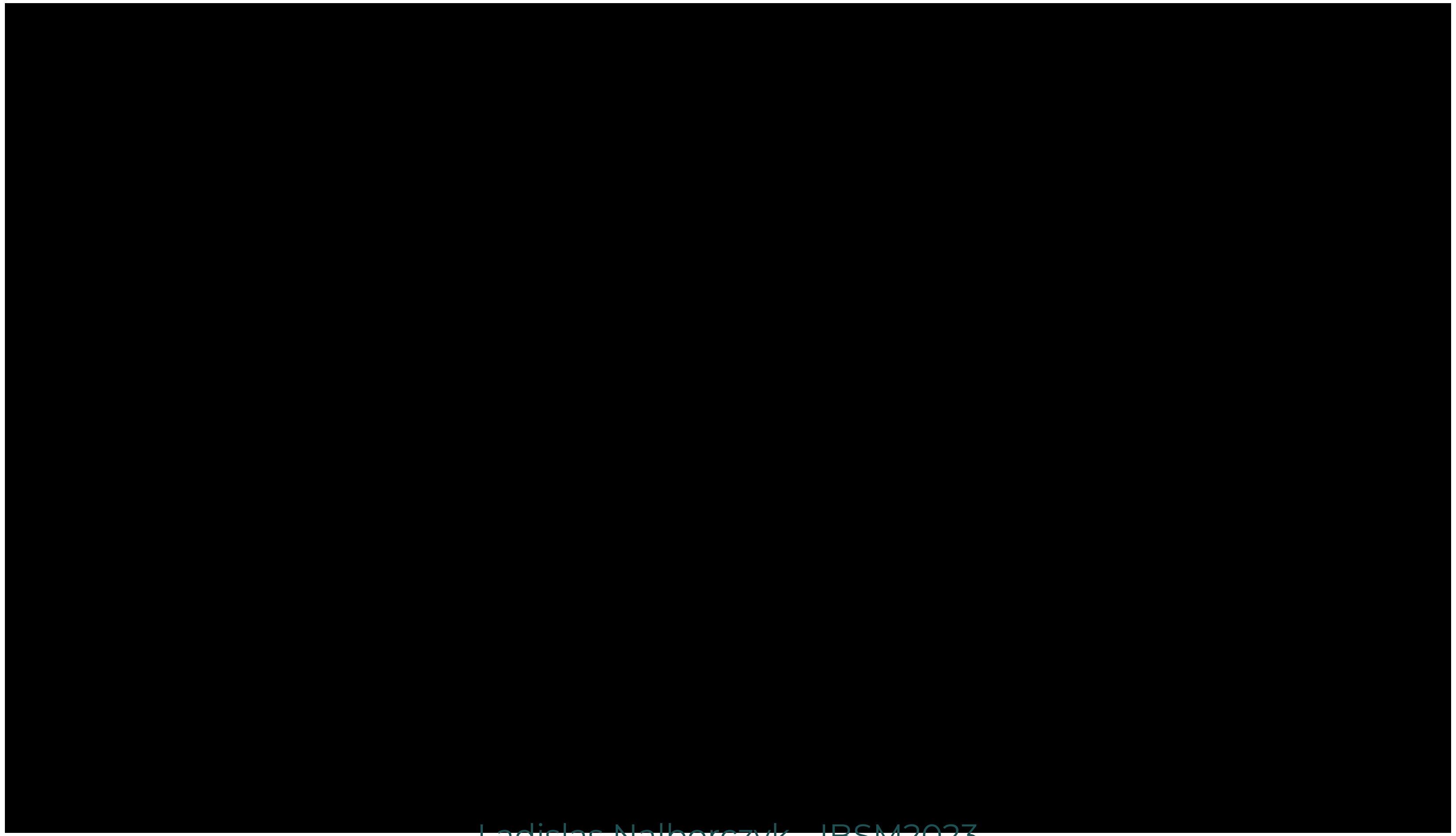
What is a model good for?

> One of the most basic problem in scientific inference is the so-called inverse problem: How to figure out causes from observations. It is a problem, because many different causes can produce the same evidence. So while it can be easy to go forward from a known cause to predicted observations, it can very hard to go backwards from observation to cause (McElreath, 2020).

So far, we have only considered **statistical models**. These models are useful devices to describe associations, but they tell us nothing about *how* these associations arise. In the last part of the course, we will focus on **process models**, aiming at describing the mechanisms generating the data (generative models).

Two-alternative forced choice

Two-alternative forced choice (2AFC) is a method for measuring the sensitivity of a person or animal to some particular sensory input, stimulus, through that observer's pattern of choices and response times to two versions of the sensory input. At each trial, the participant is forced to choose between two alternatives. For instance, in the random dot motion coherence task (below), the participant must make a choice response between two directions of motion (e.g., up or down or left or right), usually indicated by a motor response such as a saccade or pressing a button.



Reaction times

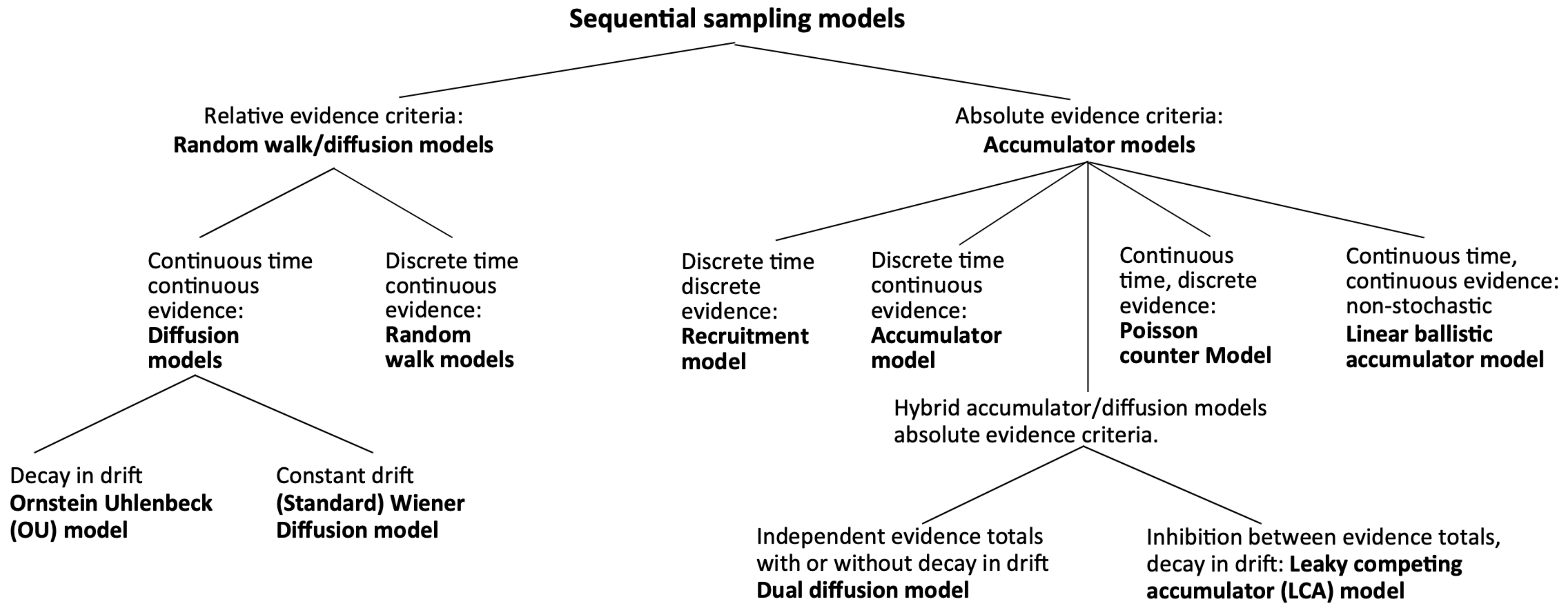
Reaction times (RTs) distributions are generally positively skewed, with the skewness increasing with task difficulty. We also know that the mean of the RTs is proportional to the standard deviation of the RTs. Increases in the difficulty usually lead to increased RTs and decreased accuracy. Moreover, changes in difficulty also produces regular changes in the distribution of RTs, most notably in its spread but not much in its shape (for a review, see Forstmann et al., 2016). Moreover, we often find a speed-accuracy trade-off in these tasks.

The use of simple statistical model (e.g., only analysing differences in group-level average RTs across conditions) is severely limited in such tasks. Therefore, several models have been proposed to account for the peculiarities of the data coming from these tasks as well as to relate it to the underlying cognitive processes.

Assumptions

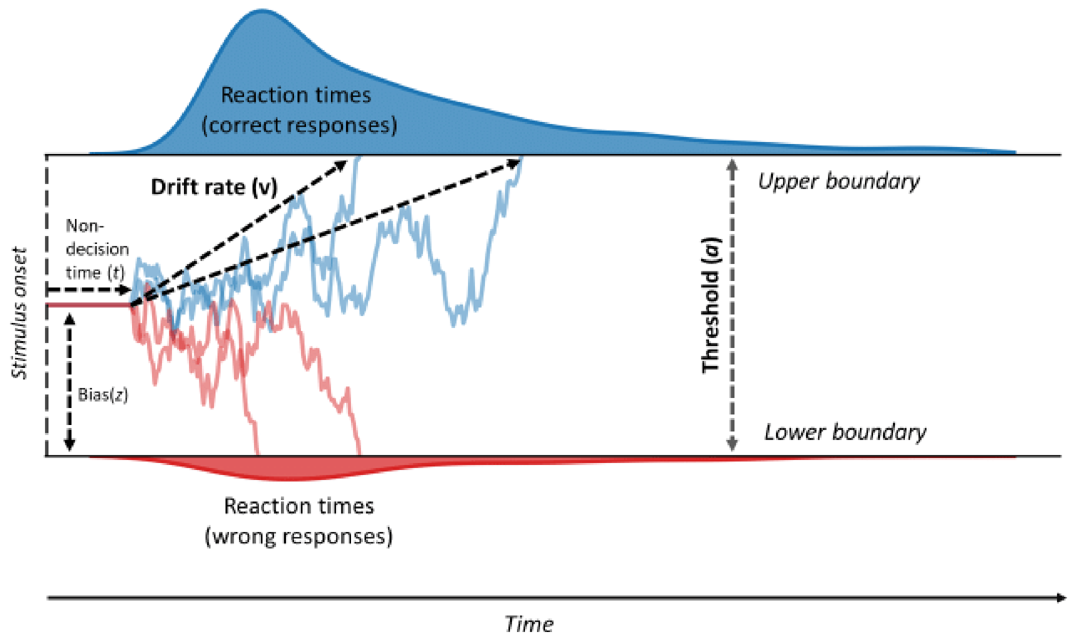
There are typically three assumptions made by **evidence accumulation models**:

- Evidence favouring each alternative is integrated over time
- The process is subject to random fluctuations
- The decision is made when sufficient evidence has accumulated favouring one alternative


Sequential sampling models
Relative evidence criteria: Random walk/diffusion models
Continuous time continuous evidence: Diffusion models
Discrete time continuous evidence: Random walk models
Decay in drift Ornstein Uhlenbeck (OU) model
Constant drift (Standard) Wiener Diffusion model
Absolute evidence criteria: Accumulator models
Discrete time discrete evidence: Recruitment model
Discrete time continuous evidence: Accumulator model
Continuous time, discrete evidence: Poisson counter Model
Continuous time, continuous evidence: non-stochastic Linear ballistic accumulator model
Hybrid accumulator/diffusion models absolute evidence criteria.
Independent evidence totals with or without decay in drift Dual diffusion model
Inhibition between evidence totals, decay in drift: Leaky competing accumulator (LCA) model

Drift-diffusion model

The drift-diffusion model (DDM) is a continuous-time evidence accumulation model for binary choice tasks (Ratcliff, 1978). It assumes that in each trial evidence is accumulated in a noisy (diffusion) process by a single accumulator. As shown below, evidence accumulation starts at some point (the starting point or "bias") and continues until the accumulator hits one of the two decision bounds in which case the corresponding response is given. The total response time is the sum of the decision time from the accumulation process plus non-decisional components (Vandekerckhove et al., 2010; Wabersich & Vandekerckhove, 2014; Wagenmakers, 2009). This kind of model provides a *decomposition* of RT data that isolates components (of processing) from stimulus encoding to decision so that they can be studied individually (Ratcliff & McKoon, 2008; Wagenmakers et al., 2007).



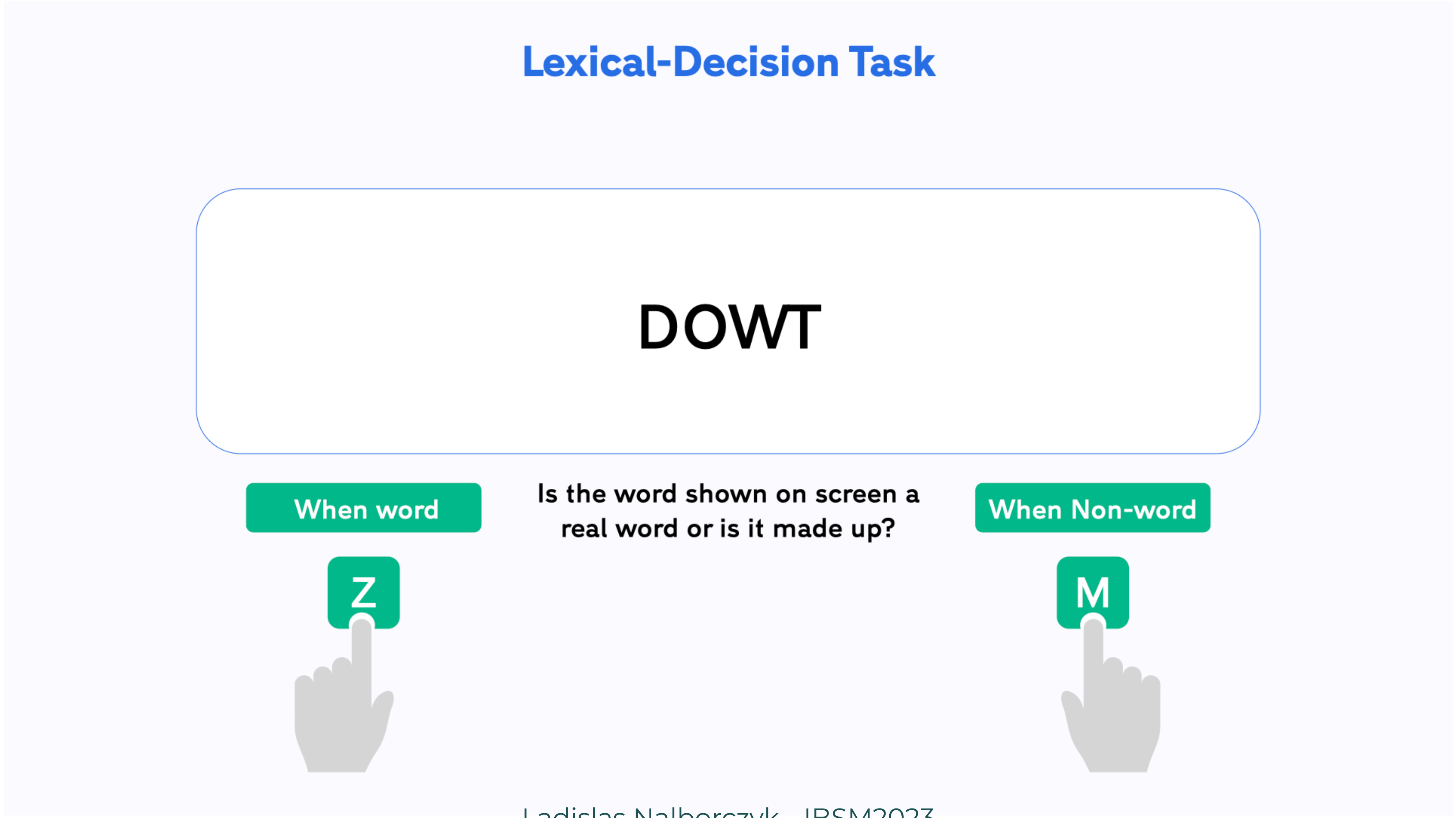
Drift-diffusion model

In sum, the original DDM allows decomposing responses to a binary choice tasks and corresponding response times into four latent processes (from Singmann, 2017):[1]

- The **drift rate** δ (delta) is the average slope of the accumulation process towards the boundaries (i.e., it represents the average amount of evidence accumulated per unit time). The larger the (absolute value of the) drift rate, the stronger the evidence for the corresponding response option (thus quantifying the “ease of processing”).
- The **boundary separation** α (alpha) is the distance between the two decision bounds and can be interpreted as a measure of response caution, with a high α corresponding to high caution.
- The **starting point** (or bias) β (beta) of the accumulation process is a measure of response bias towards one of the two response boundaries.
- The **non-decision time** τ (tau) captures all non-decisional processes such as stimulus encoding and (motor) response processes.

Application example: lexical decision task

The lexical decision task is a procedure used in many psychology and psycholinguistics experiments. The basic procedure involves measuring how quickly and accurately people classify stimuli as words or nonwords.



Application example: lexical decision task

We will adapt the example from Singmann (2017) and analyse part of the data from Experiment 1 of Wagenmakers et al. (2008). The data comes from 17 participants performing a lexical decision task. Participants made decisions under speed or accuracy emphasis instructions in different experimental blocks. After removing some extreme RTs, we restrict the analysis to high-frequency words (frequency = high) and the corresponding high-frequency non-words (frequency = nw_high) to reduce estimation time. To setup the model, we also need a numeric response variable in which 0 corresponds to responses at the lower response boundary and 1 corresponds to responses at the upper boundary.

```r
# loading the "speed_acc" data from the "rtdists" package
data(speed_acc, package = "rtdists")

# reshaping the data
df <- speed_acc %>%
    # removing extreme RTs
    filter(censor == FALSE) %>%
    # removing ppt with id=2 (less observations than others)
    filter(id != 2) %>%
    # focusing on high-frequency words and non-words
    filter(frequency %in% c("high", "nw_high") ) %>%
    # converting the response variable to a numeric 0/1 variable
    mutate(response2 = as.numeric(response == "word") ) %>%
    # keeping only some proportion of the data (for computational ease)
    filter(as.numeric(block) < 9) %>%
    mutate(id = factor(id), block = factor(block) )
```

Drift-diffusion model in brms

An important decision that has to be made before setting up a model is which parameters are allowed to differ between which conditions. One common constraint of the DDM is that parameters that are set before the evidence accumulation process starts (i.e., boundary separation, starting point, and non-decision time) cannot change based on stimulus characteristics that are not known to the participant before the start of the trial. Thus, the stimulus category, in the present case word versus non-word, is usually only allowed to affect the drift rate. We follow this constraint. Furthermore, all relevant variables are manipulated within-subject. Thus, the maximal varying-effects structure (Barr et al., 2013) can (and should) be implemented.

```r
# defining the model formula (one "linear model" per parameter)
formula <- brmsformula(
  # drift rate (delta)
  rt | dec(response2) ~ 1 + condition * stim_cat + (1 + condition * stim_cat | id),
  # boundary separation parameter (alpha)
  bs ~ 1 + condition + (1 + condition | id),
  # non-decision time (tau)
  ndt ~ 1 + condition + (1 + condition | id),
  # starting point or bias (beta)
  bias ~ 1 + condition + (1 + condition | id)
  )
```

Drift-diffusion model in brms

```r
# defining the contrasts
contrasts(df$condition) <- c(+0.5, -0.5)
contrasts(df$stim_cat) <- c(+0.5, -0.5)

# defining the priors
priors <- c(
  # priors for the intercepts
  prior(normal(0, 5), class = "Intercept"),
  prior(normal(0, 1), class = "Intercept", dpar = "bs"),
  prior(normal(0, 1), class = "Intercept", dpar = "ndt"),
  prior(normal(0, 1), class = "Intercept", dpar = "bias"),
  # priors for the slopes
  prior(normal(0, 1), class = "b"),
  # priors on the SD of the varying effects
  prior(exponential(1), class = "sd")
  )
```

Drift-diffusion model in brms

We then fit this model using the `brms::brm()` function. We run 8 chains for 5000 iterations and use the first 1000 iterations as warmup, resulting in a total of $8 \times (5000 - 1000) = 32000$ posterior samples.

```r
# specify initial values to help the model start sampling
# (with small variation between chains)
chains <- 8 # number of chains
epsilon <- 0.1 # variability in starting value for the NDT intercept
get_init_value <- function (x) list(Intercept_ndt = rnorm(n = 1, mean = x, sd = epsilon) )
inits_drift <- replicate(chains, get_init_value(-3), simplify = FALSE)

# fitting the model
fit_wiener <- brm(
  formula = formula,
  data = df,
  # specifying the family and link functions for each parameter
  family = wiener(
    link = "identity", link_bs = "log",
    link_ndt = "log", link_bias = "logit"
    ),
  # comment this line to use default priors
  prior = priors,
  # list of initialisation values
  init = inits_drift,
  init_r = 0.05,
  warmup = 1000, iter = 5000,
  chains = chains, cores = chains,
  # control = list(adapt_delta = 0.99, max_treedepth = 15),
  # saves the model (as .rds) or loads it if it already exists
```

Aparté: Writing our model

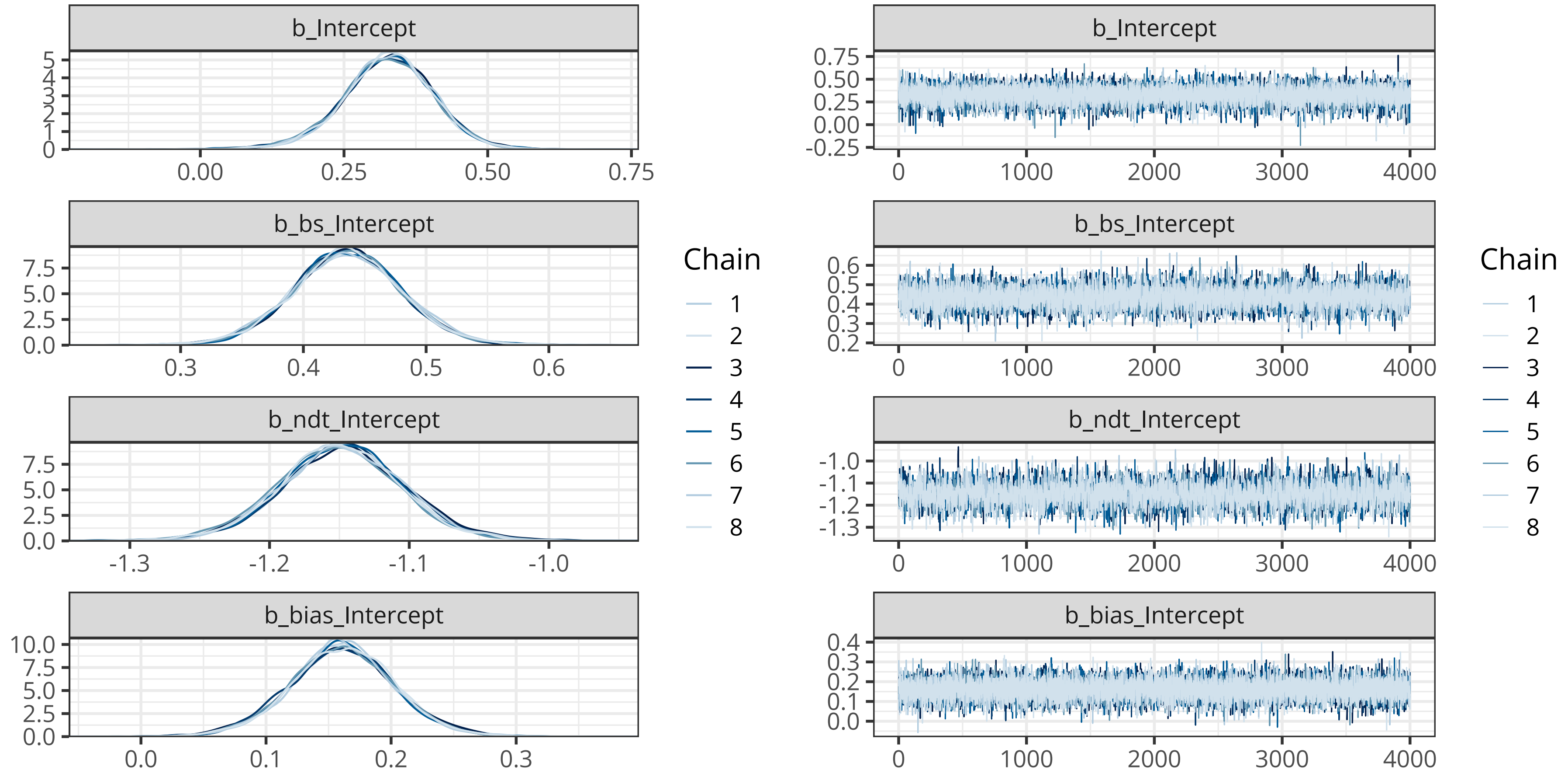
Our model can be written (in a simplified form, omitting the varying effects) as:

$$
\begin{aligned}
\text{RT}_i &\sim \text{DDM}(\alpha_i, \tau_i, \beta_i, \delta_i) && \text{Observation model for the RTs.} \\
\delta_i &= \beta_{0[\delta]} + \beta_{1[\delta]} \cdot \text{Condition}_i + \beta_{2[\delta]} \cdot \text{Stim\_cat}_i + && \text{Linear model for the drift rate.} \\
&\quad \beta_{3[\delta]} \cdot \text{Condition}_i \cdot \text{Stim\_cat}_i \\
\log(\alpha_i) &= \beta_{0[\alpha]} + \beta_{1[\alpha]} \cdot \text{Condition}_i && \text{Linear model for the (log) boundary s} \\
\log(\tau_i) &= \beta_{0[\tau]} + \beta_{1[\tau]} \cdot \text{Condition}_i && \text{Linear model for the (log) non-decisi} \\
\text{logit}(\beta_i) &= \beta_{0[\beta]} + \beta_{1[\beta]} \cdot \text{Condition}_i && \text{Linear model for the (logit) bias.} \\
\beta_{0[\delta]} &\sim \text{Normal}(0, 5) && \text{Prior on the intercept for the drift rate} \\
\beta_{1[\delta]}, \beta_{2[\delta]}, \beta_{3[\delta]} &\sim \text{Normal}(0, 1) && \text{Prior on the slopes for the drift rate.} \\
\beta_{0[\alpha]}, \beta_{0[\tau]}, \beta_{0[\beta]} &\sim \text{Normal}(0, 1) && \text{Prior on the intercept for the other pa} \\
\beta_{1[\alpha]}, \beta_{1[\tau]}, \beta_{1[\beta]} &\sim \text{Normal}(0, 1) && \text{Prior on the slopes for the other parar}
\end{aligned}
$$

where i denotes observations (i.e., lines in the dataframe).

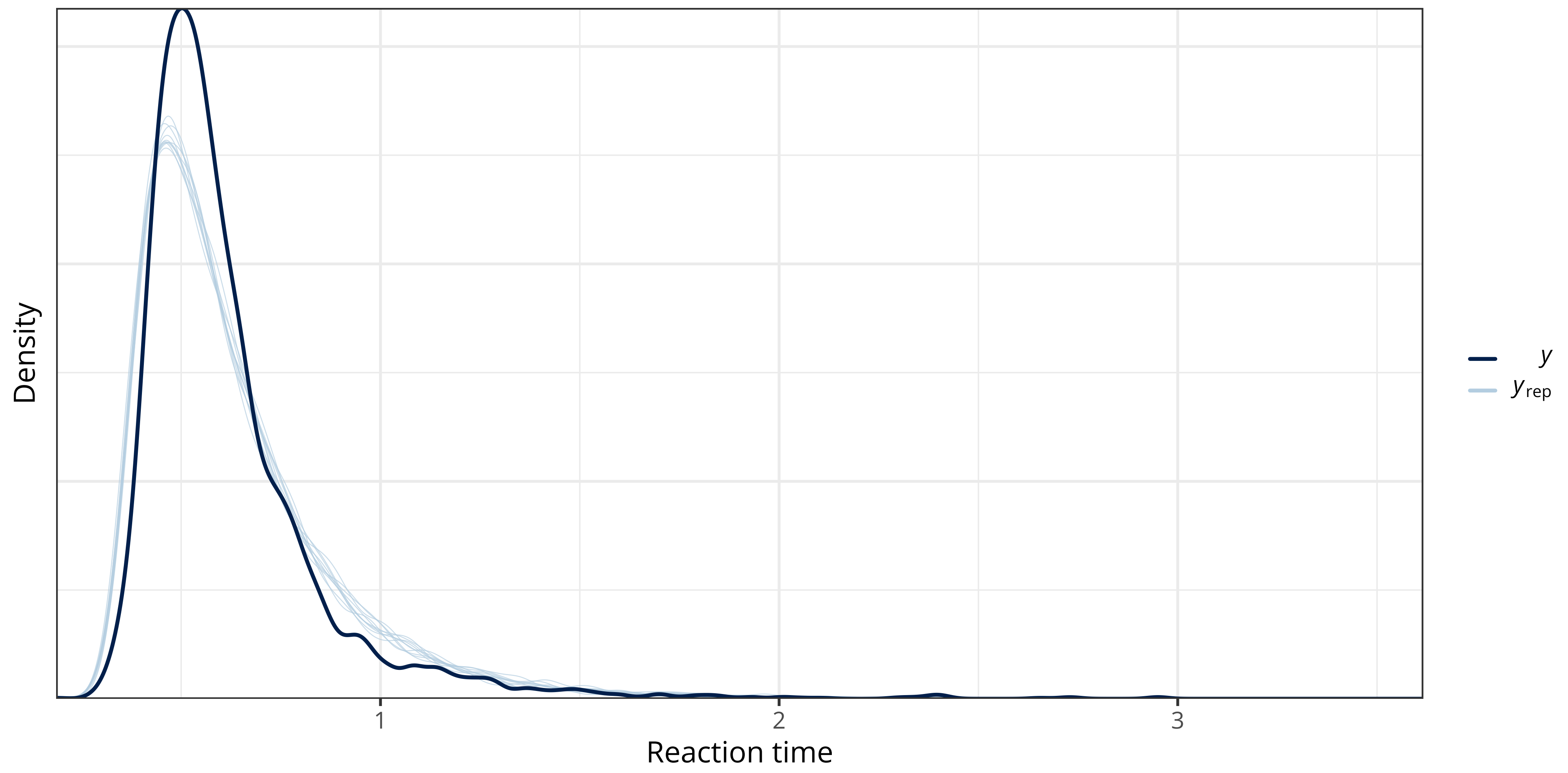
Assessing model convergence

```r
# combo can be hist, dens, dens_overlay, trace, trace_highlight...
# cf. https://mc-stan.org/bayesplot/reference/MCMC-overview.html
plot(
    x = fit_wiener, combo = c("dens_overlay", "trace"),
    variable = variables(fit_wiener)[1:4],
    ask = FALSE
    )
```



Assessing model fit 1/4

```r
pp_check(object = fit_wiener, ndraws = 10) +
  labs(x = "Reaction time", y = "Density")
```



Assessing model fit 2/4

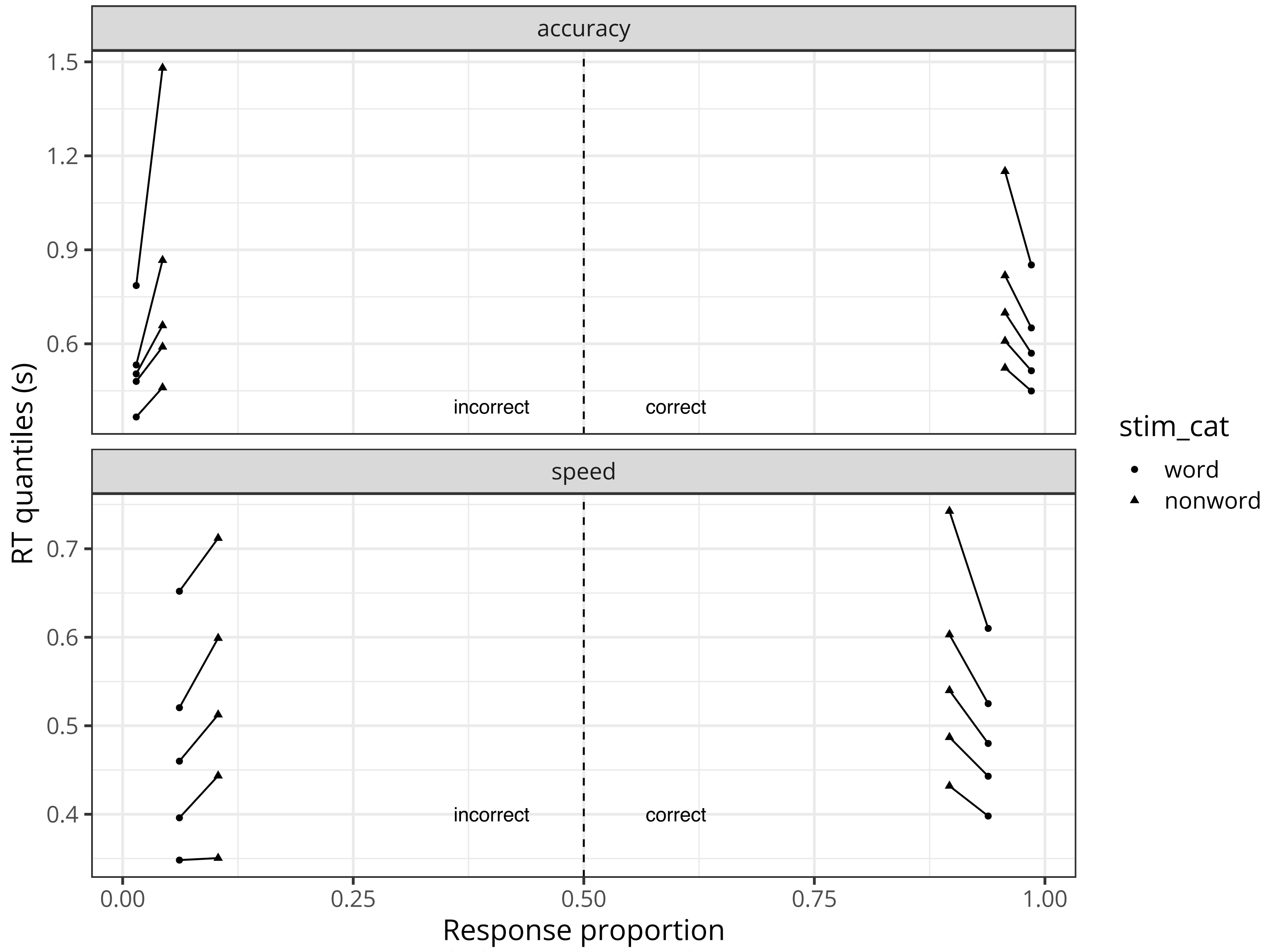
A powerful way to convey the relationship between response times and accuracy is using **quantile probability plots** (Ratcliff & Tuerlinckx, 2002) which show quantiles of the response times distribution (typically 0.1, 0.3, 0.5, 0.7, and 0.9) for correct and incorrect responses on the y-axis against probabilities of correct and incorrect responses for experimental conditions on the x-axis. The plot is built by first aggregating the data (cf. the detailed code online).

```r
# aggregating the data using the qpf() function from
# https://vasishth.github.io/bayescogsci/book/ch-lognormalrace.html#sec-acccoding
df_qpf <- df %>%
    mutate(acc = ifelse(as.character(stim_cat) == as.character(response), 1, 0) ) %>%
    group_by(stim_cat, condition) %>%
    qpf() %>%
    ungroup()

head(df_qpf)
```

```
# A tibble: 6 × 6
  stim_cat condition  rt_q      p     q response
  <fct>    <fct>     <dbl>  <dbl> <dbl> <chr>
1 word     accuracy  0.366 0.0147   0.1 incorrect
2 word     accuracy  0.48  0.0147   0.3 incorrect
3 word     accuracy  0.504 0.0147   0.5 incorrect
4 word     accuracy  0.533 0.0147   0.7 incorrect
5 word     accuracy  0.786 0.0147   0.9 incorrect
6 word     accuracy  0.449 0.985    0.1 correct
```

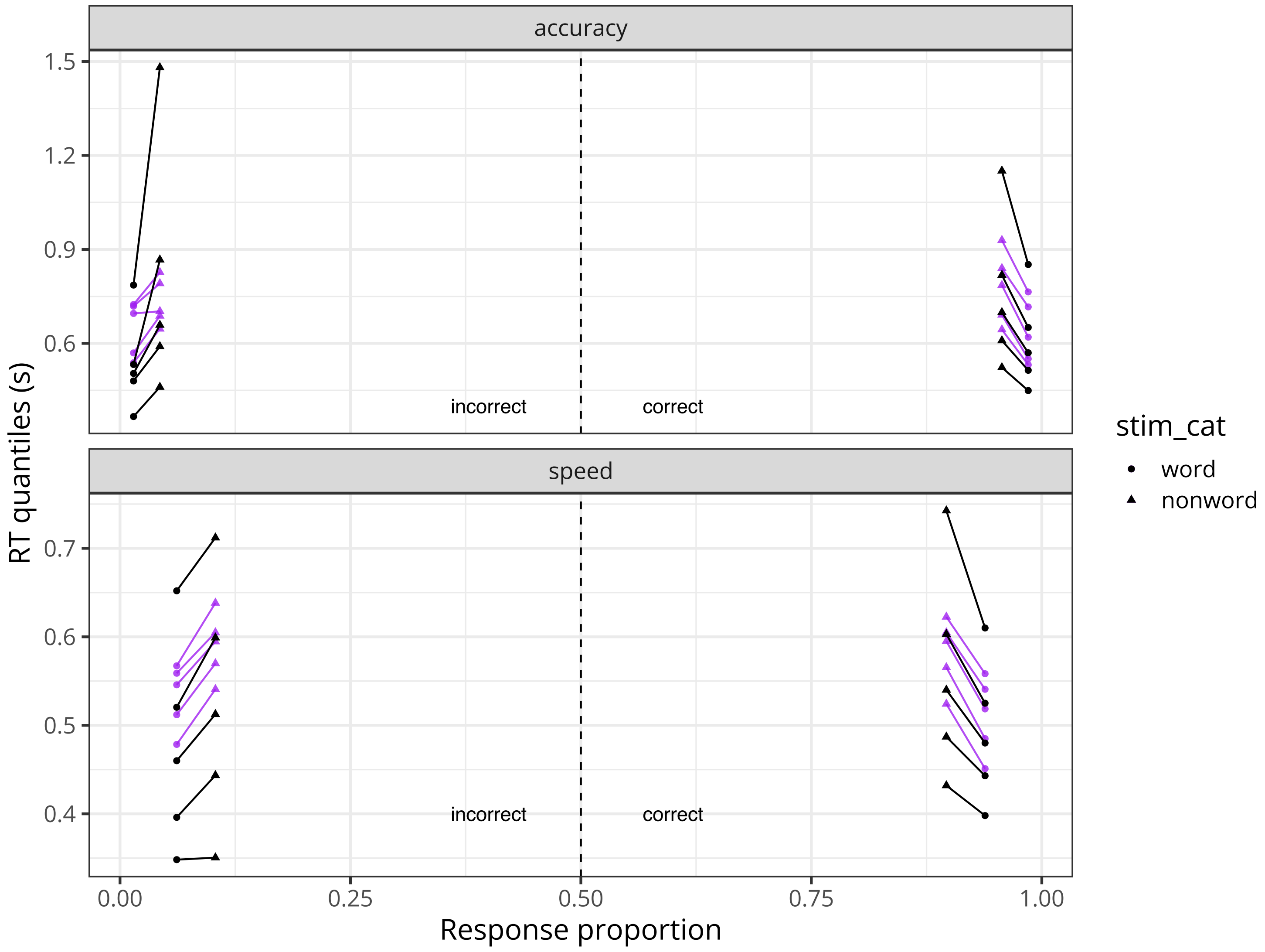
Assessing model fit 3/4

This plot shows that words are recognised faster than non-words, that responses are generally faster in the “speed” than in the “accuracy” condition, and that incorrect responses seem more variable than correct responses.



Assessing model fit 4/4

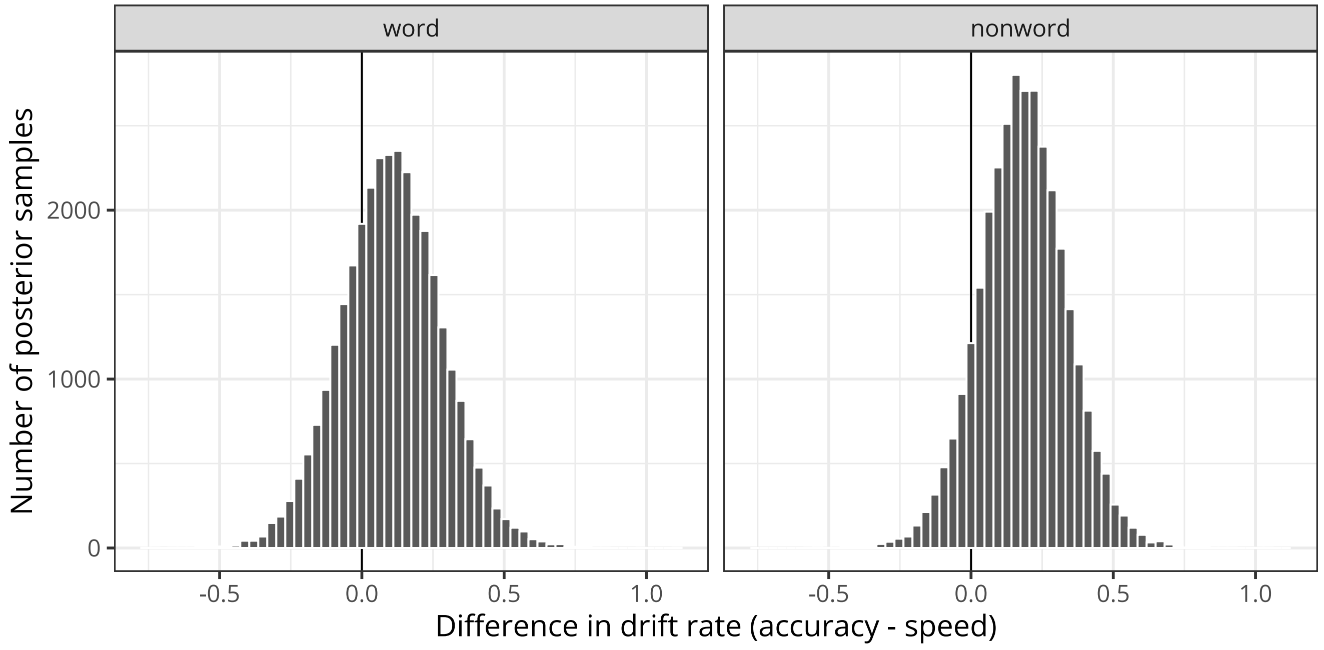
The model fit is not so bad, but the model is unable to capture fast errors (bottom left), and more generally, extreme quantiles...



Parameter estimates: differences in drift rate

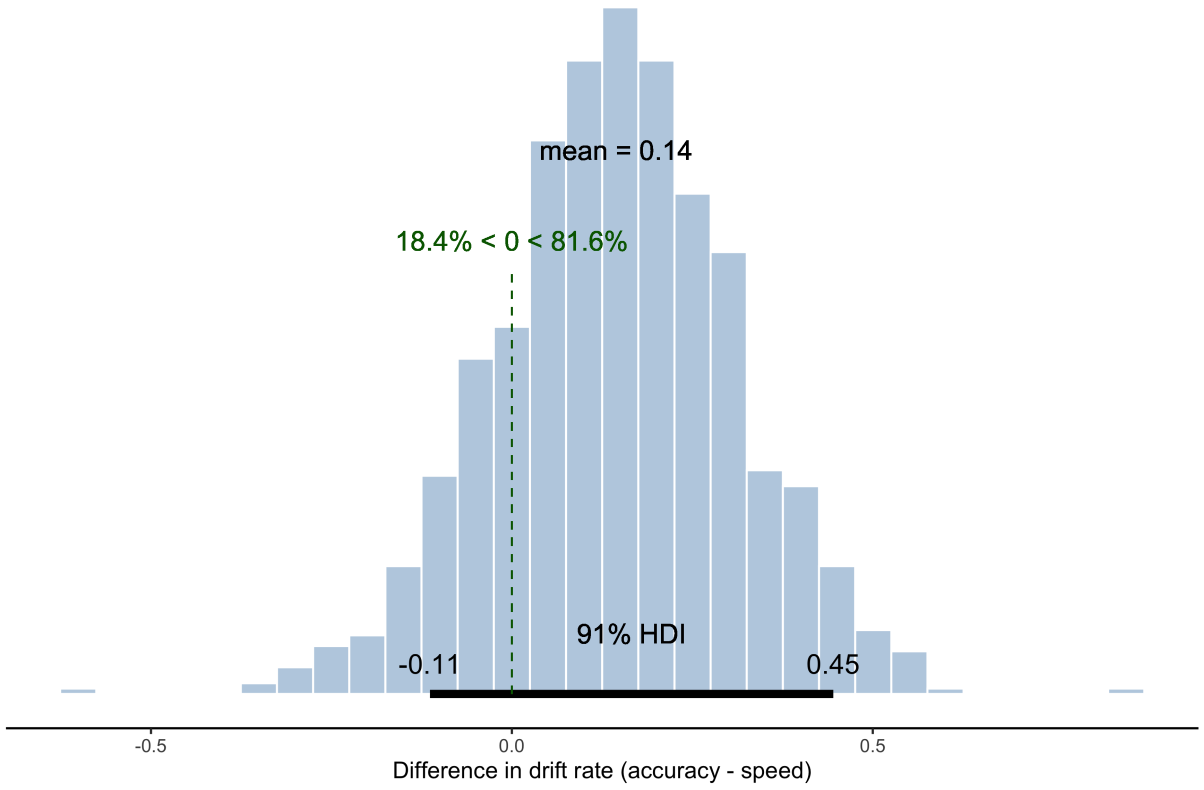
We first check whether there is a difference in drift rate between conditions for words and non-words. This shows that a non negligible part of the posterior mass is above zero, meaning there is some (weak) evidence that the drift rate is greater in the accuracy than in the speed condition.

```r
library(tidybayes)
library(emmeans)

drift_rate_samples_per_condition <- fit_wiener %>%
    # retrieving drift rate values per condition
    emmeans(~condition * stim_cat) %>%
    # retrieving posterior sample for each cell
    gather_emmeans_draws()
```



Parameter estimates: differences in drift rate

```r
samps <- drift_rate_samples_per_condition %>%
    mutate(.value = if_else(stim_cat == "nonword", (-1) * .value, .value) ) %>%
    pivot_wider(names_from = condition, values_from = .value) %>%
    mutate(accuracy_speed_diff = accuracy - speed)

posterior_plot(
    samples = sample(x = samps$accuracy_speed_diff, size = 1e3),
    compval = 0, nbins = 30
    ) + labs(x = "Difference in drift rate (accuracy - speed)")
```

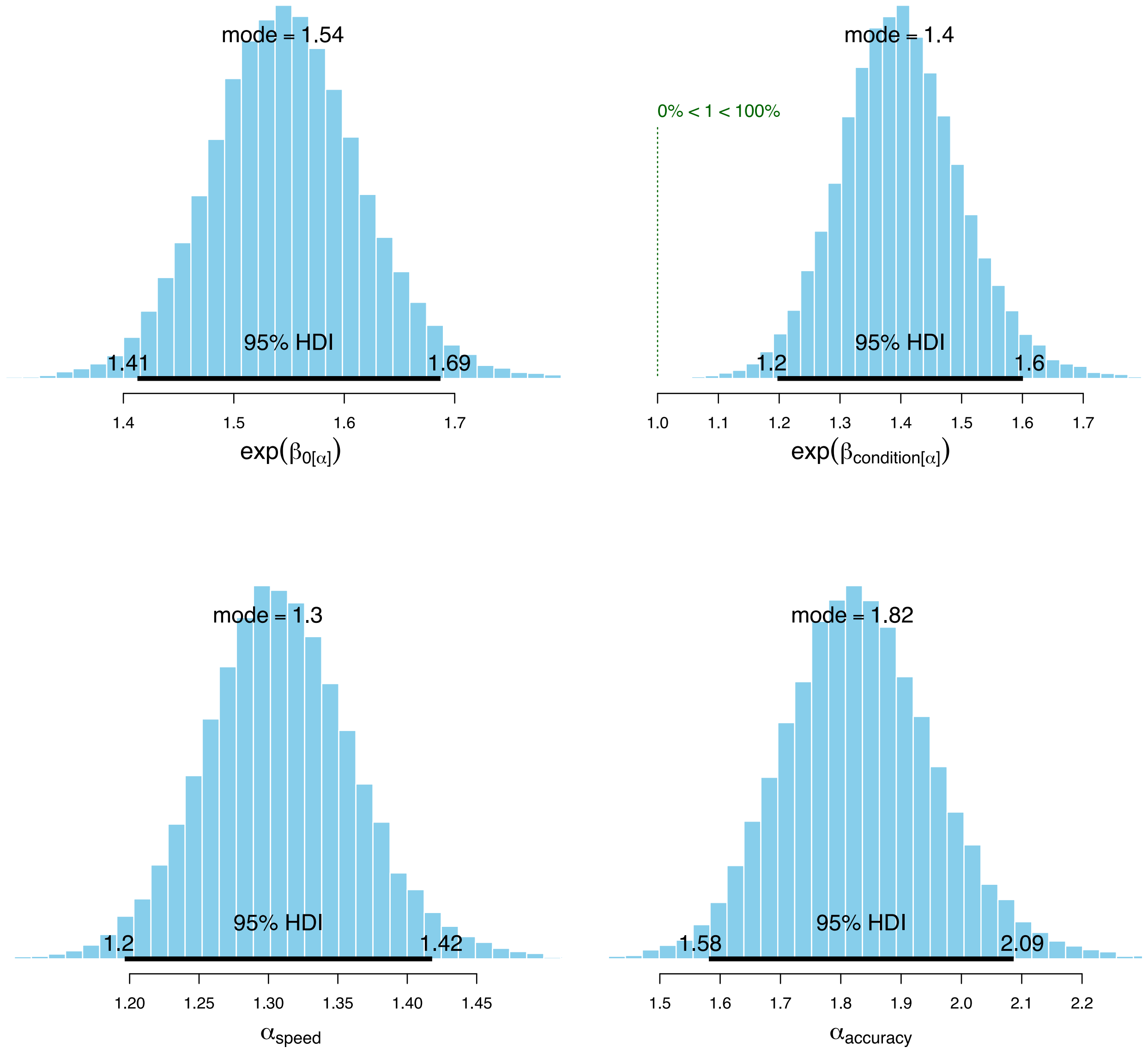


Parameter estimates: boundary separation

Recall that the boundary separation parameter can be interpreted as a measure of response caution (with high α corresponding to high response caution), and that the linear model for this parameter is on the log scale (i.e., we used a log link function): $\log(\alpha_i) = \beta_0 + \beta_1 \cdot \text{Condition}_i$. Therefore, we have to apply the inverse link function (i.e., $\exp(\cdot)$) to the parameter to be able to interpret it. Taking $\exp(\beta_1)$ gives the proportional change in the value of the boundary-separation parameter when we go from the speed to the accuracy condition (see upper right panel). In our case, $\exp(\beta_1) \approx 0.4$, which means that going from the speed to the accuracy condition leads to an increase of approximately 40% in the value of the boundary-separation parameter. In other words, response caution is higher in the accuracy (lower right panel) than in the speed (lower left panel) condition.

```r
# retrieving posterior samples
post <- as_draws_df(x = fit_wiener)
# retrieving the posterior samples for the boundary-separation
posterior_intercept_bs <- post$b_bs_Intercept
posterior_slope_bs <- post$b_bs_condition1
# computing the posterior distribution in the speed condition
posterior_bs_speed <- exp(posterior_intercept_bs - 0.5 * posterior_slope_bs)
# computing the posterior distribution in the accuracy condition
posterior_bs_accuracy <- exp(posterior_intercept_bs + 0.5 * posterior_slope_bs)
```

Parameter estimates: boundary separation

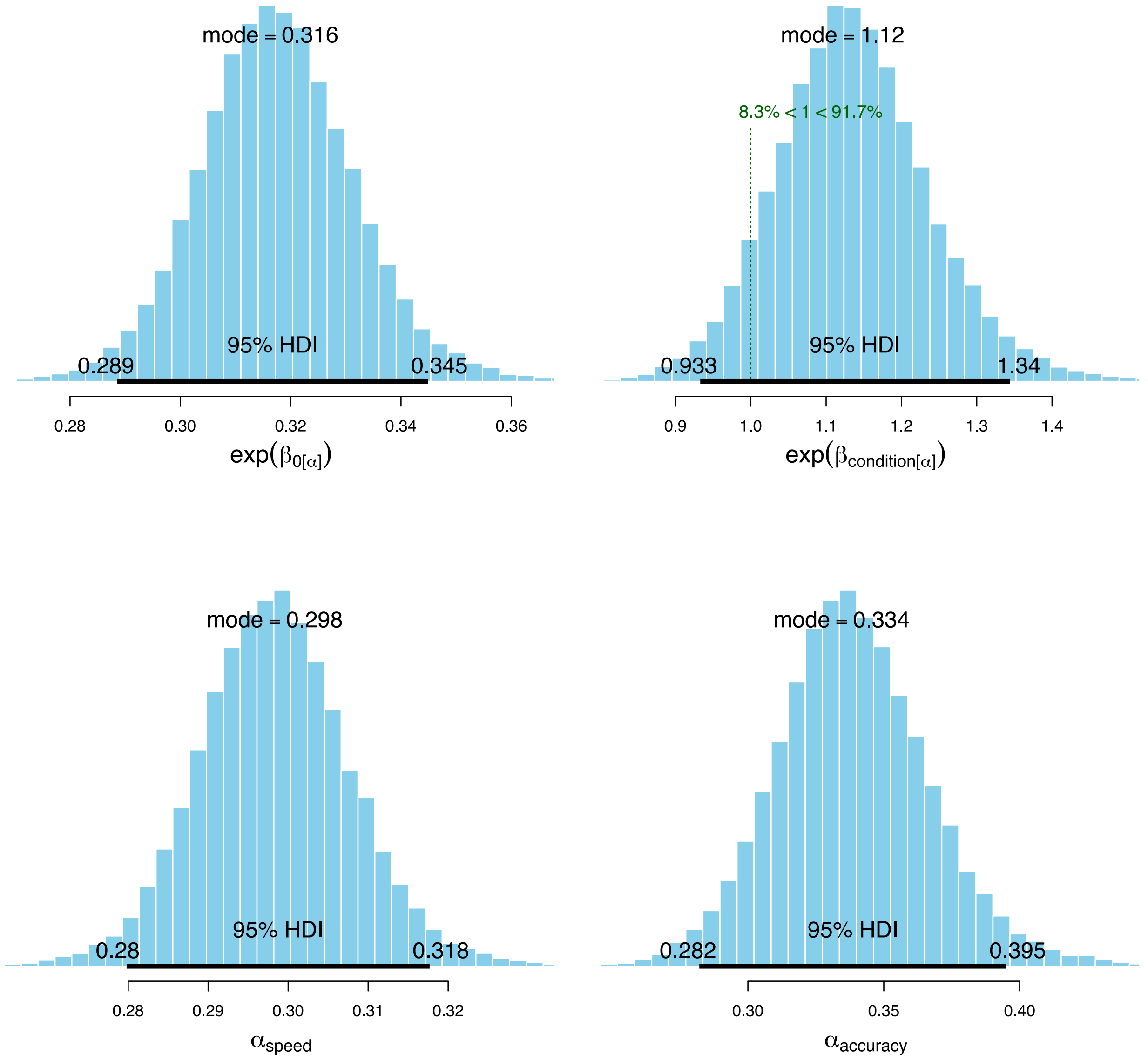


Parameter estimates: non-decision time

Recall that the non-decision time parameter can be interpreted as a measure of the time used by non-decisional processes such as stimulus encoding or motor response, and that the linear model for this parameter is on the log scale (i.e., we used a log link function): $\log(\tau_i) = \beta_0 + \beta_1 \cdot \text{Condition}_i$. Therefore, we have to apply the inverse link function (i.e., $\exp(\cdot)$) to the parameter to be able to interpret it. Taking $\exp(\beta_1)$ gives the proportional change in the value of the non-decision time parameter when we go from the speed to the accuracy condition. In our case, $\exp(\beta_1) \approx 1.12$ which means that going from the speed to the accuracy condition leads to an increase of approximately 12% of the non-decision time. In other words, non-decisional processes seem to take longer in the accuracy than in the speed condition.

```r
# retrieves the posterior samples for the non-decision time
posterior_intercept_ndt <- post$b_ndt_Intercept
posterior_slope_ndt <- post$b_ndt_condition1
# computes the posterior distribution in the speed condition
posterior_ndt_speed <- exp(posterior_intercept_ndt - 0.5 * posterior_slope_ndt)
# computes the posterior distribution in the accuracy condition
posterior_ndt_accuracy <- exp(posterior_intercept_ndt + 0.5 * posterior_slope_ndt)
```

Parameter estimates: non-decision time

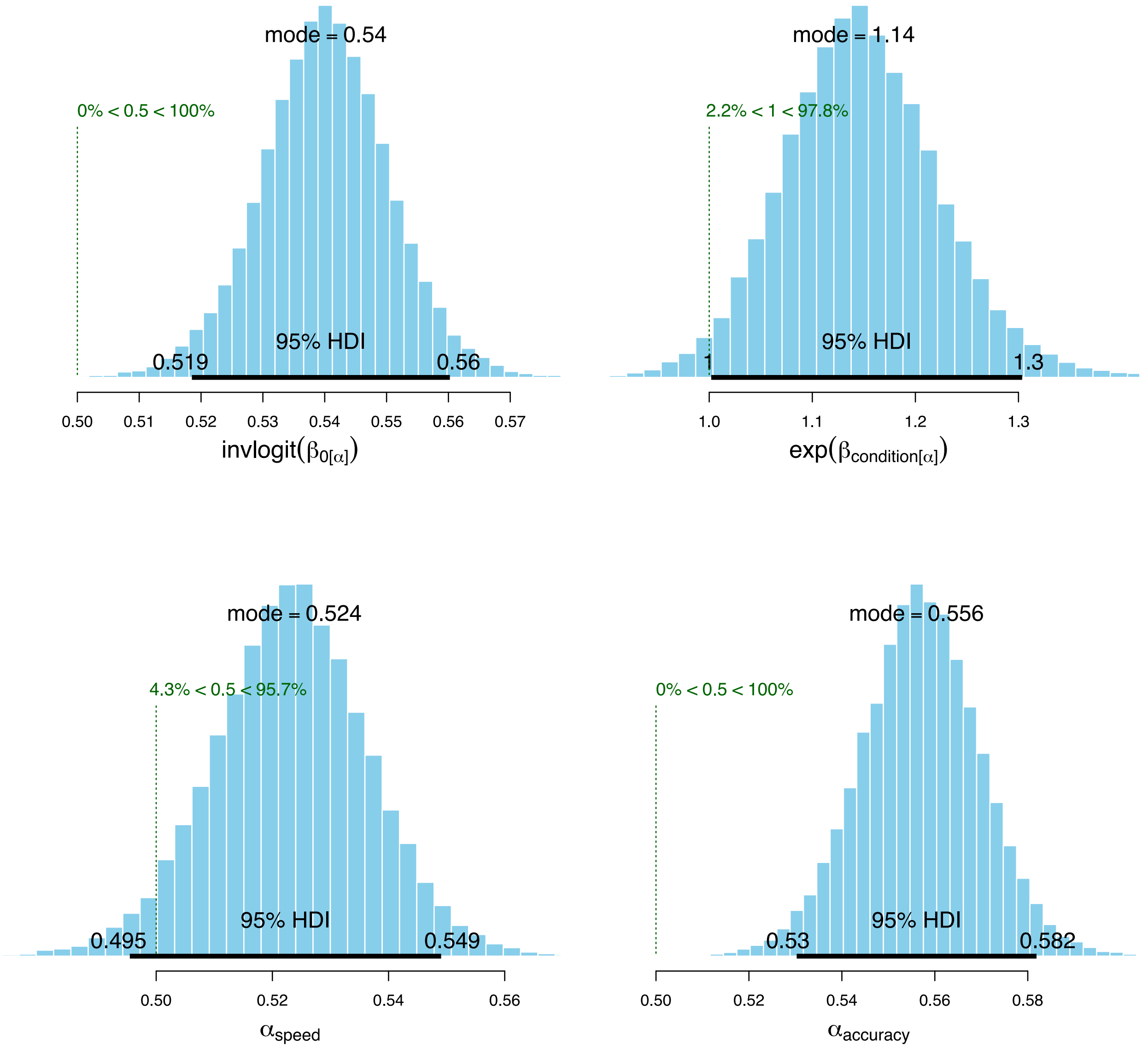


Parameter estimates: starting point (bias)

The starting point is a measure of response bias towards one of the two response boundaries and is bounded between 0 and 1. The linear model for this parameter is on the logit (log-odds) scale: $\log(\frac{\beta_i}{1-\beta_i}) = \beta_0 + \beta_1 \cdot \text{Condition}_i$. Therefore, we have to apply the inverse link function (i.e., $\text{logit}^{-1}(\beta_i) = \text{logistic}(\beta_i) = \frac{1}{1+\exp(-\beta_i)} = \frac{\exp(\beta_i)}{\exp(\beta_i)+1}$) to the parameter to be able to interpret it on its natural scale (i.e., between 0 and 1). There seems to be a bias toward the “word” responses in the accuracy condition, but not (or less) in the speed condition.

```r
# retrieves the posterior samples for the bias
posterior_intercept_bias <- post$b_bias_Intercept
posterior_slope_bias <- post$b_bias_condition1
# computes the posterior distribution in the speed condition
posterior_bias_speed <- plogis(posterior_intercept_bias - 0.5 * posterior_slope_bias)
# computes the posterior distribution in the accuracy condition
posterior_bias_accuracy <- plogis(posterior_intercept_bias + 0.5 * posterior_slope_bias)
```

Parameter estimates: starting point (bias)



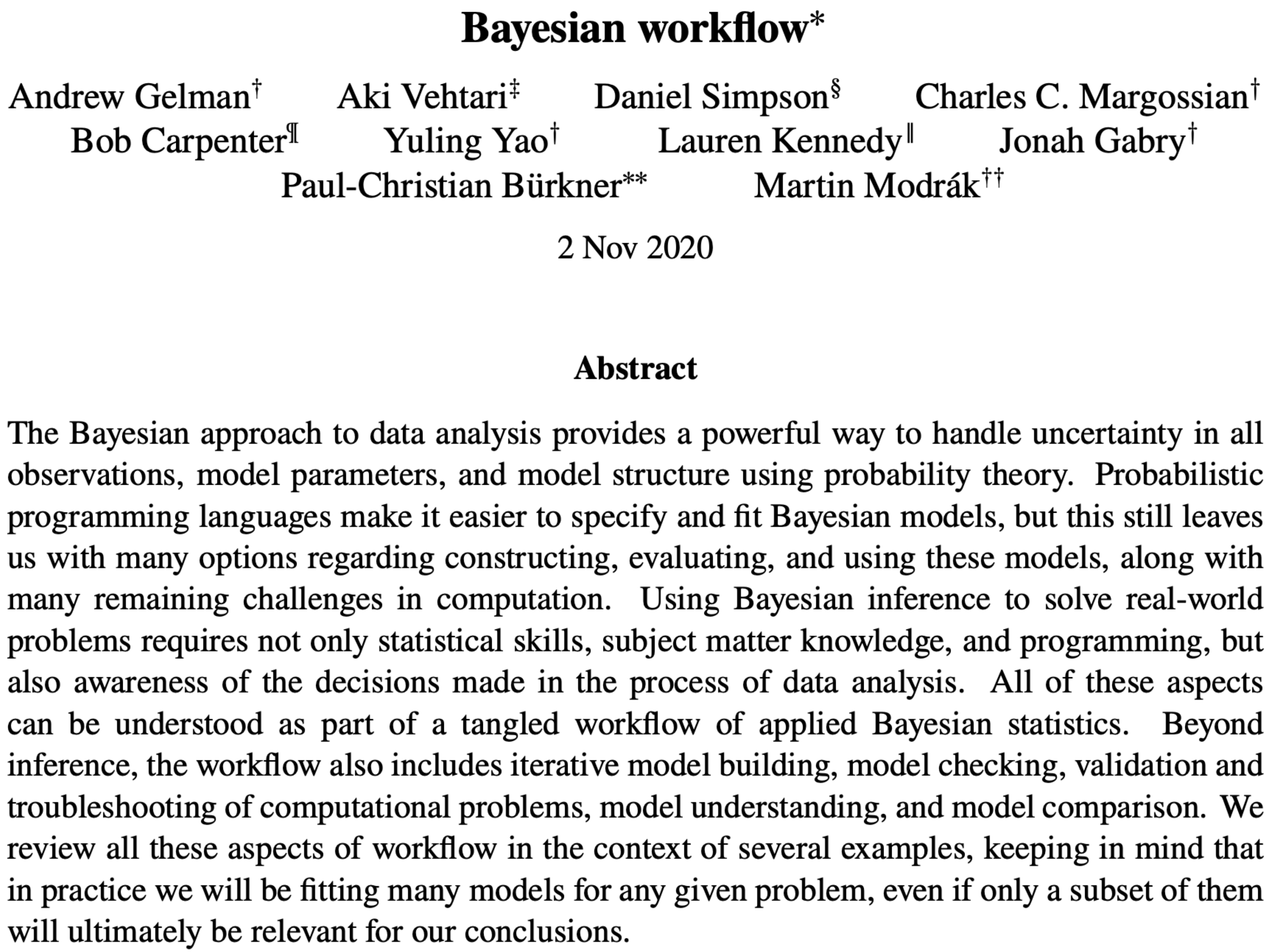
Summary

Somehow unsurprisingly, we find that response caution is much higher in the accuracy than in the speed condition, but the same goes for the drift rate and the non-decision time (to a lesser extent).

How do we know that these parameters actually refer to the processes we think they refer to? We check that experimental manipulations that are supposed to only affect some component (rate of information uptake, setting of response criteria, duration of the motor response and bias) effectively do (e.g., Ratcliff, 2002; Ratcliff & Rouder, 1998; Voss et al., 2004)

We can also check parameter values in different groups with known specificities (e.g., age-related slowing in Ratcliff et al., 2000, 2001) or we can try validating the interpretation of these parameters by using additional measures such as electrophysiogical (e.g., EMG, EEG) measures (e.g., Servant et al., 2021; Weindel et al., 2021).

Bayesian workflow (Gelman et al., 2020)



Bayesian workflow*

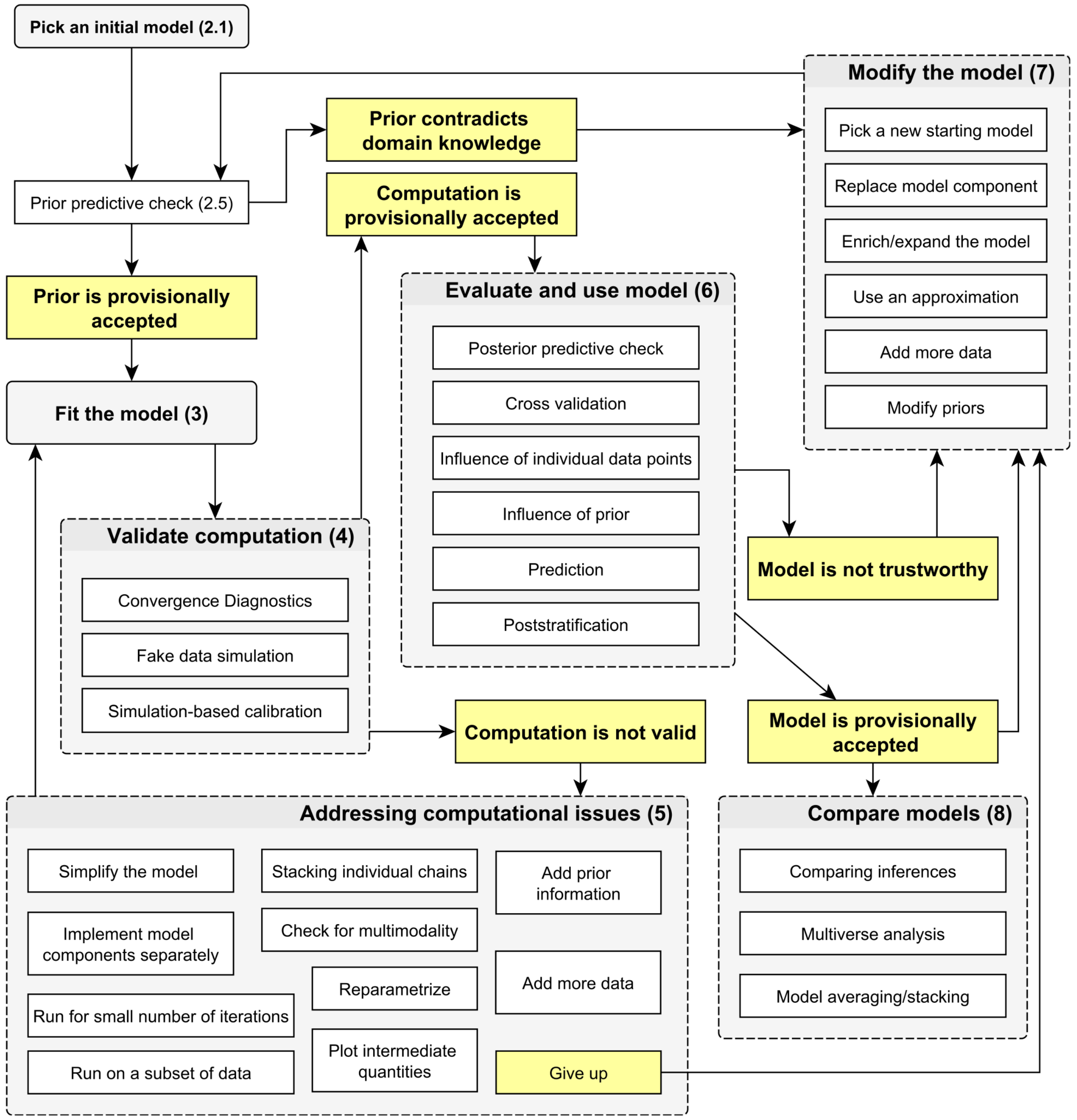
Andrew Gelman† Aki Vehtari‡ Daniel Simpson§ Charles C. Margossian†
Bob Carpenter¶ Yuling Yao† Lauren Kennedy‖ Jonah Gabry†
Paul-Christian Bürkner** Martin Modrák††

2 Nov 2020

Abstract

The Bayesian approach to data analysis provides a powerful way to handle uncertainty in all observations, model parameters, and model structure using probability theory. Probabilistic programming languages make it easier to specify and fit Bayesian models, but this still leaves us with many options regarding constructing, evaluating, and using these models, along with many remaining challenges in computation. Using Bayesian inference to solve real-world problems requires not only statistical skills, subject matter knowledge, and programming, but also awareness of the decisions made in the process of data analysis. All of these aspects can be understood as part of a tangled workflow of applied Bayesian statistics. Beyond inference, the workflow also includes iterative model building, model checking, validation and troubleshooting of computational problems, model understanding, and model comparison. We review all these aspects of workflow in the context of several examples, keeping in mind that in practice we will be fitting many models for any given problem, even if only a subset of them will ultimately be relevant for our conclusions.

Bayesian workflow (Gelman et al., 2020)



Conclusions

Bayesian inference is a general approach to parameter estimation. This approach uses probability theory to quantify the uncertainty with respect to the value of parameters from statistical models.

These models are composed of different blocks (e.g., likelihood function, priors, linear or non-linear model), which are modifiable as desired. What we usually refer to as “model assumptions” are simply the consequences of modelling choices. In other words, the user defines (and does not suffer) the model’s assumptions.

We have seen that the linear regression model provides a very flexible architecture which makes possible to describe, via the modification of the likelihood function and via the introduction of link functions, complex (e.g., non-linear) relationships between outcomes and predictors. These models can gain in precision by taking into account the variability and structures present in the data (cf. multilevel models).

Conclusions

The `brms` package is a real Swiss army knife of Bayesian statistics in `R`. It allows you to fit almost any type of regression model. This includes all models that we have seen, but also many others. Among others, multivariate models (i.e., models with several outcomes), "distributional" models (e.g., to predict variance differences), generalized additive models, Gaussian processes (Gaussian processes), models from signal detection theory, mixture models, drift-diffusion models, non-linear models...

Do not hesitate to contact me for more information on these models or if you have questions about your own data. You can also contact the creator of the `brms` package, who is very active online (see his site). See also the Stan forum.

References

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, *68*(3), 255–278. https://doi.org/10.1016/j.jml.2012.11.001

Efron, B., & Morris, C. (1977). Stein's paradox in statistics. *Scientific American*, *236*(5), 119–127. https://doi.org/10.1038/scientificamerican0577-119

Forstmann, B. U., Ratcliff, R., & Wagenmakers, E.-J. (2016). Sequential Sampling Models in Cognitive Neuroscience: Advantages, Applications, and Extensions. *Annual Review of Psychology*, *67*, 641–666. https://doi.org/10.1146/annurev-psych-122414-033645

Gelman, A. (2005). Analysis of variance? Why it is more important than ever. *The Annals of Statistics*, *33*(1), 1–53. https://doi.org/10.1214/009053604000001048

Gelman, A., & Hill, J. (2006). *Data analysis using regression and multilevel/hierarchical models*. https://doi.org/10.1017/cbo9780511790942

Gelman, A., Vehtari, A., Simpson, D., Margossian, C. C., Carpenter, B., Yao, Y., Kennedy, L., Gabry, J., Bürkner, P.-C., & Modrák, M. (2020). Bayesian workflow. *arXiv:2011.01808 [Stat]*. http://arxiv.org/abs/2011.01808

Lewandowski, D., Kurowicka, D., & Joe, H. (2009). Generating random correlation matrices based on vines and extended onion method. *Journal of Multivariate Analysis*, *100*(9), 1989–2001. https://doi.org/10.1016/j.jmva.2009.04.008

McElreath, R. (2020). *Statistical rethinking: A bayesian course with examples in r and stan* (2nd ed.). Taylor; Francis, CRC Press.

Nalborczyk, L., Batailler, C., Lœvenbruck, H., Vilain, A., & Bürkner, P.-C. (2019). An Introduction to Bayesian Multilevel Models Using brms: A Case Study of Gender Effects on Vowel Variability in

Standard Indonesian. *Journal of Speech, Language, and Hearing Research*, *62*(5), 1225–1242. https://doi.org/10.1044/2018_jslhr-s-18-0006

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, *85*(2), 59–108. https://doi.org/10.1037/0033-295X.85.2.59

Ratcliff, R. (2002). A diffusion model account of response time and accuracy in a brightness discrimination task: Fitting real data and failing to fit fake but plausible data. *Psychonomic Bulletin & Review*, *9*(2), 278–291. https://doi.org/10.3758/bf03196283

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation*, *20*(4), 873–922. https://doi.org/10.1162/neco.2008.12-06-420

Ratcliff, R., & Rouder, J. N. (1998). Modeling Response Times for Two-Choice Decisions. *Psychological Science*, *9*(5), 347–356. https://doi.org/10.1111/1467-9280.00067

Ratcliff, R., Spieler, D., & Mckoon, G. (2000). Explicitly modeling the effects of aging on response time. *Psychonomic Bulletin & Review*, *7*(1), 1–25. https://doi.org/10.3758/bf03210723

Ratcliff, R., Thapar, A., & McKoon, G. (2001). The effects of aging on reaction time in a signal detection task. *Psychology and Aging*, *16*(2), 323–341. https://doi.org/10.1037/0882-7974.16.2.323

Ratcliff, R., & Tuerlinckx, F. (2002). Estimating parameters of the diffusion model: Approaches to dealing with contaminant reaction times and parameter variability. *Psychonomic Bulletin & Review*, *9*(3), 438–481. https://doi.org/10.3758/BF03196302

Servant, M., Logan, G. D., Gajdos, T., & Evans, N. J. (2021). An integrated theory of deciding and acting. *Journal of Experimental Psychology: General*, *150*(12), 2435–2454. https://doi.org/10.1037/xge0001063

Singmann, H. (2017). Diffusion/Wiener Model Analysis with brms – Part I: Introduction and Estimation. In *Henrik Singmann - Computational Psychology*. http://singmann.org/wiener-model-analysis-with-brms-part-i/

Vandekerckhove, J., Verheyen, S., & Tuerlinckx, F. (2010). A crossed random effects diffusion model for speeded semantic categorization decisions. *Acta Psychologica*, *133*(3), 269–282. https://doi.org/10.1016/j.actpsy.2009.10.009

Voss, A., Rothermund, K., & Voss, J. (2004). Interpreting the parameters of the diffusion model: An empirical validation. *Memory & Cognition*, *32*(7), 1206–1220. https://doi.org/10.3758/BF03196893

Wabersich, D., & Vandekerckhove, J. (2014). The RWiener Package: An R Package Providing Distribution Functions for the Wiener Diffusion Model. *The R Journal*, *6*(1), 49–56. https://journal.r-project.org/archive/2014/RJ-2014-005/index.html

Wagenmakers, E.-J. (2009). Methodological and empirical developments for the Ratcliff diffusion model of response times and accuracy. *European Journal of Cognitive Psychology*, *21*(5), 641–671. https://doi.org/10.1080/09541440802205067

Wagenmakers, E.-J., Ratcliff, R., Gomez, P., & McKoon, G. (2008). A diffusion model account of criterion shifts in the lexical decision task. *Journal of Memory and Language*, *58*(1), 140–159. https://doi.org/10.1016/j.jml.2007.04.006

Wagenmakers, E.-J., Van Der Maas, H. L. J., & Grasman, R. P. P. P. (2007). An EZ-diffusion model for response time and accuracy. *Psychonomic Bulletin & Review*, *14*(1), 3–22. https://doi.org/10.3758/BF03194023

Weindel, G., Anders, R., Alario, F.-X., & Burle, B. (2021). Assessing model-based inferences in decision making with single-trial response time decomposition. *Journal of Experimental Psychology: General*, *150*(8), 1528–1555. https://doi.org/10.1037/xge0001010