Introduction à la modélisation statistique bayésienne

Un cours en R et Stan avec brms

Ladislas Nalborczyk (CNRS, LPL, Aix-Marseille Univ)

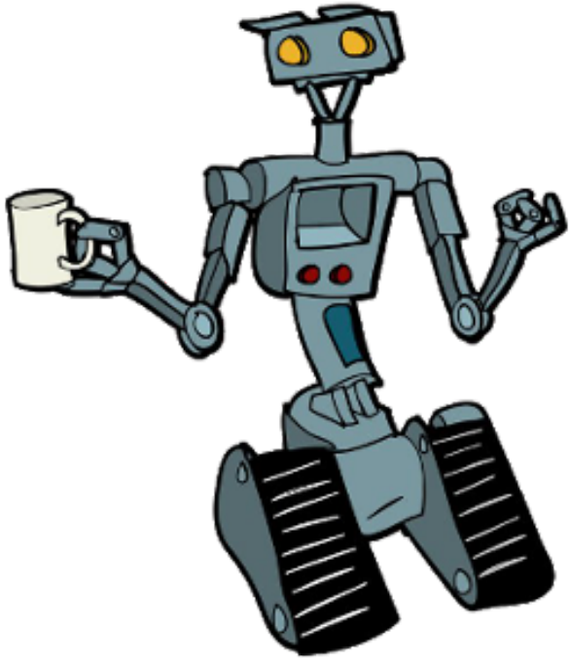
Planning

Cours n°01 : Introduction à l'inférence bayésienne
Cours n°02 : Modèle Beta-Binomial
Cours n°03 : Introduction à brms, modèle de régression linéaire
Cours n°04 : Modèle de régression linéaire (suite)
Cours n°05 : Markov Chain Monte Carlo
Cours n°06 : Modèle linéaire généralisé
Cours n°07 : Comparaison de modèles
Cours n°08 : Modèles multi-niveaux (généralisés)
Cours n°09 : Examen final

Modèles multi-niveaux

Le but est de construire un modèle qui puisse **apprendre à plusieurs niveaux**, qui puisse produire des estimations qui seront informées par les différents groupes présents dans les données. Nous allons suivre l'exemple suivant tout au long de ce cours.

Imaginons que nous ayons construit un robot visiteur de cafés, et que celui-ci s'amuse à mesurer le temps d'attente après avoir commandé. Ce robot visite 20 cafés différents, 5 fois le matin et 5 fois l'après-midi, et mesure le temps de service après avoir commandé un cappuccino.



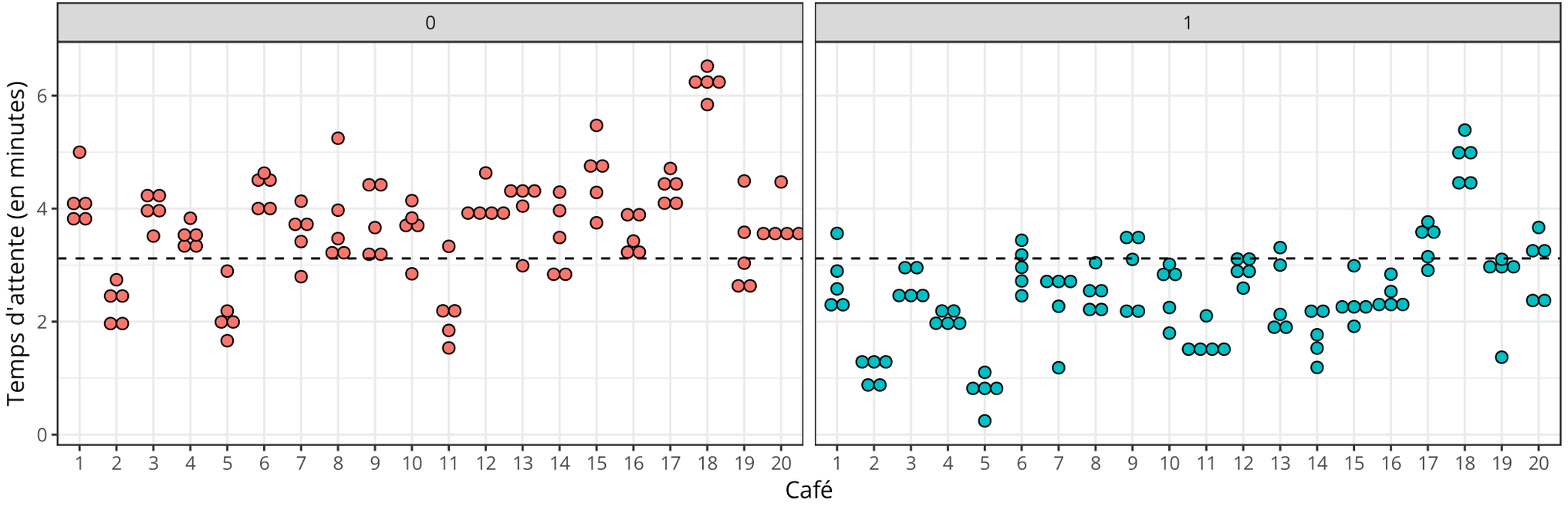
Robot et café

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

df <- open_data(robot)
slice_sample(df, n = 15) %>% arrange(cafe, afternoon)
```

	cafe	afternoon	wait
1	2	0	2.521509
2	2	0	2.387150
3	2	0	1.954398
4	3	1	3.044962
5	7	0	4.132367
6	8	0	5.246669
7	9	1	2.236393
8	11	1	2.103433
9	12	0	3.900817
10	13	0	4.044824
11	15	0	4.752380
12	16	1	2.533639
13	18	0	5.841681
14	18	0	6.156569
15	19	0	3.582722

Robot et café

```
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 = 2) +
    facet_wrap(~afternoon, ncol = 2) +
    labs(x = "Café", y = "Temps d'attente (en minutes)")
```



Robot et café, premier modèle

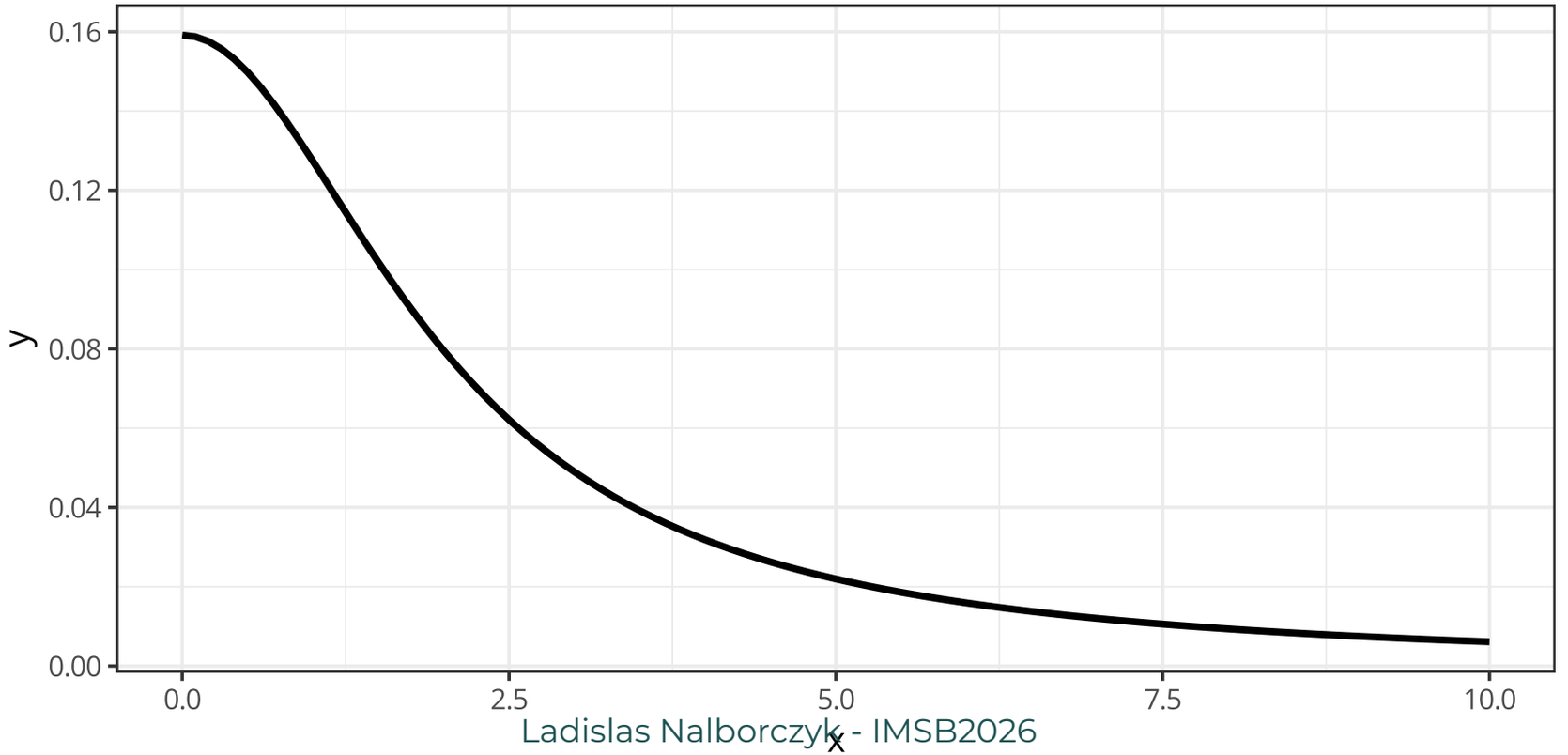
On peut construire un premier modèle, qui estime le temps moyen (sur tous les cafés confondus) pour être servi.

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

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
        )
```



Robot et café, premier modèle

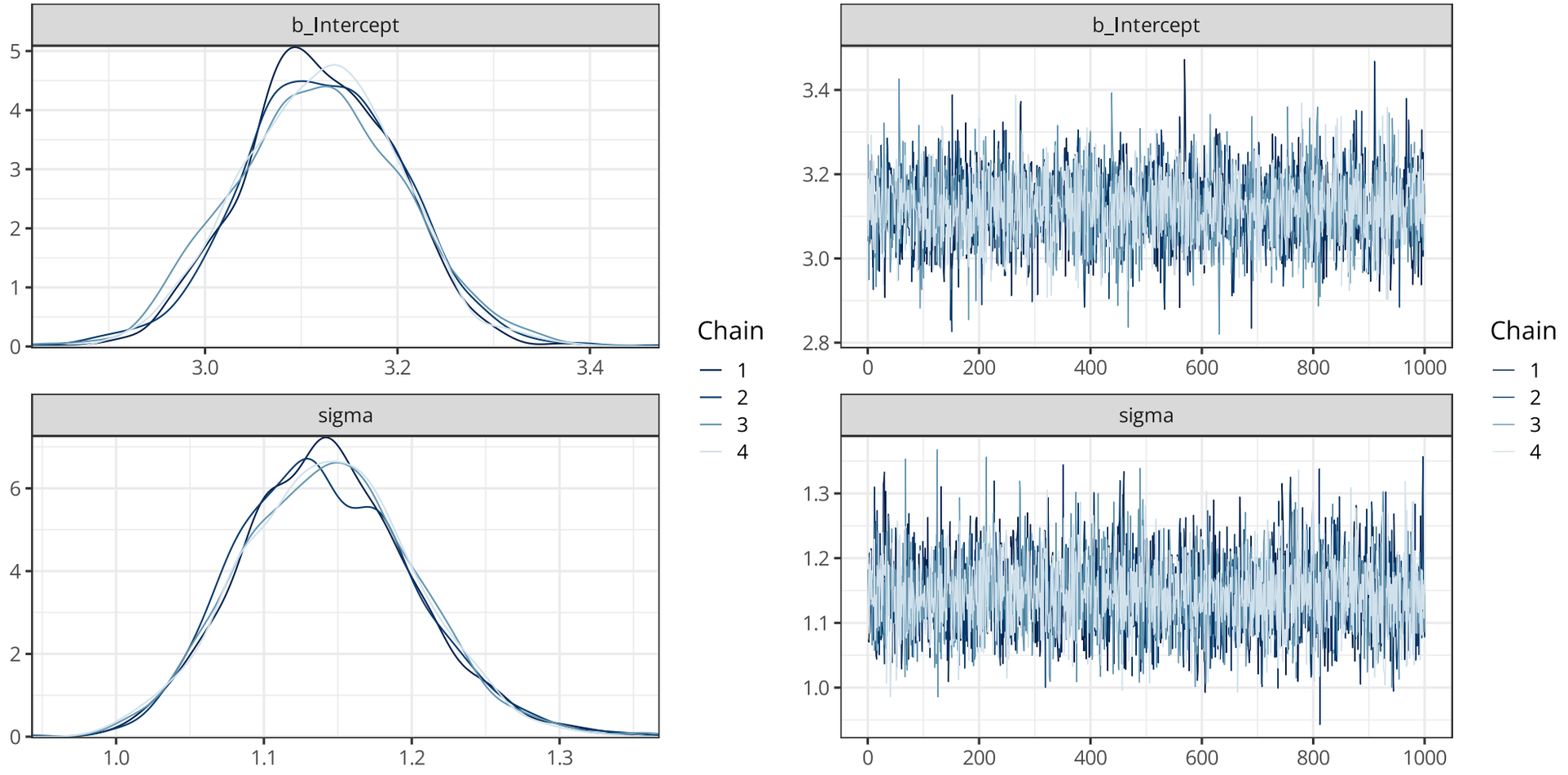
```r
library(brms)

mod1 <- brm(
  formula = wait ~ 1,
  prior = c(
    prior(normal(5, 10), class = Intercept),
    prior(cauchy(0, 2), class = sigma)
    ),
  data = df,
  chains = 4, cores = 4
  )
```

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

```
           Estimate  Est.Error     Q2.5    Q97.5
b_Intercept  3.12029 0.08305618 2.961966 3.278956
sigma        1.14329 0.05821889 1.035996 1.263067
```

Diagnostic plot

```
plot(
  x = mod1, combo = c("dens_overlay", "trace"),
  theme = theme_bw(base_size = 16, base_family = "Open Sans")
  )
```



Un intercept par café

Deuxième modèle qui estime un intercept par café. Équivalent à construire 20 dummy variables.

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

```r
mod2 <- brm(
  formula = wait ~ 0 + factor(cafe),
  prior = c(
    prior(normal(5, 10), class = b),
    prior(cauchy(0, 2), class = sigma)
    ),
  data = df,
  chains = 4, cores = 4
  )
```

Un intercept par café

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

```
              Estimate Est.Error      Q2.5    Q97.5
b_factorcafe1  3.446720 0.2581097 2.9570223 3.955792
b_factorcafe2  1.735698 0.2547330 1.2214308 2.255546
b_factorcafe3  3.321396 0.2538071 2.8150220 3.827093
b_factorcafe4  2.792376 0.2622491 2.2902496 3.304397
b_factorcafe5  1.463107 0.2495957 0.9666945 1.962709
b_factorcafe6  3.645492 0.2606125 3.1280079 4.164413
b_factorcafe7  2.944369 0.2537705 2.4381786 3.436924
b_factorcafe8  3.171952 0.2711110 2.6295728 3.690205
b_factorcafe9  3.333783 0.2657974 2.8171111 3.862563
b_factorcafe10 3.098585 0.2603172 2.5906690 3.619480
b_factorcafe11 1.920268 0.2672054 1.4035323 2.435402
b_factorcafe12 3.492849 0.2569514 2.9907473 3.989367
b_factorcafe13 3.220294 0.2559399 2.7253452 3.721055
b_factorcafe14 2.632583 0.2506545 2.1207012 3.127929
b_factorcafe15 3.479847 0.2600625 2.9548957 3.995019
b_factorcafe16 3.002896 0.2608633 2.5028342 3.505349
b_factorcafe17 3.878754 0.2685568 3.3476324 4.409492
b_factorcafe18 5.533190 0.2599808 5.0222679 6.055488
b_factorcafe19 2.975092 0.2587433 2.4756302 3.488851
b_factorcafe20 3.361872 0.2606892 2.8497260 3.878911
```

Modèle multi-niveaux

Est-ce qu'on ne pourrait pas faire en sorte que le temps mesuré au café 1 **informe** la mesure réalisée au café 2, et au café 3 ? Ainsi que le temps moyen pour être servi ? Nous allons apprendre le prior à partir des données...

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

Le prior de l'intercept pour chaque café ($\alpha_{\text{café}}$) est maintenant fonction de deux paramètres (α et $\sigma_{\text{café}}$). α et $\sigma_{\text{café}}$ sont appelés des **hyper-paramètres**, ce sont des paramètres pour des paramètres, et leurs priors sont appelés des **hyperpriors**. Il y a deux niveaux dans le modèle...

Équivalences (encore)

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

NB : α est ici défini dans le prior de $\alpha_{\text{café}}$ mais on pourrait, de la même manière, le définir dans le modèle linéaire :

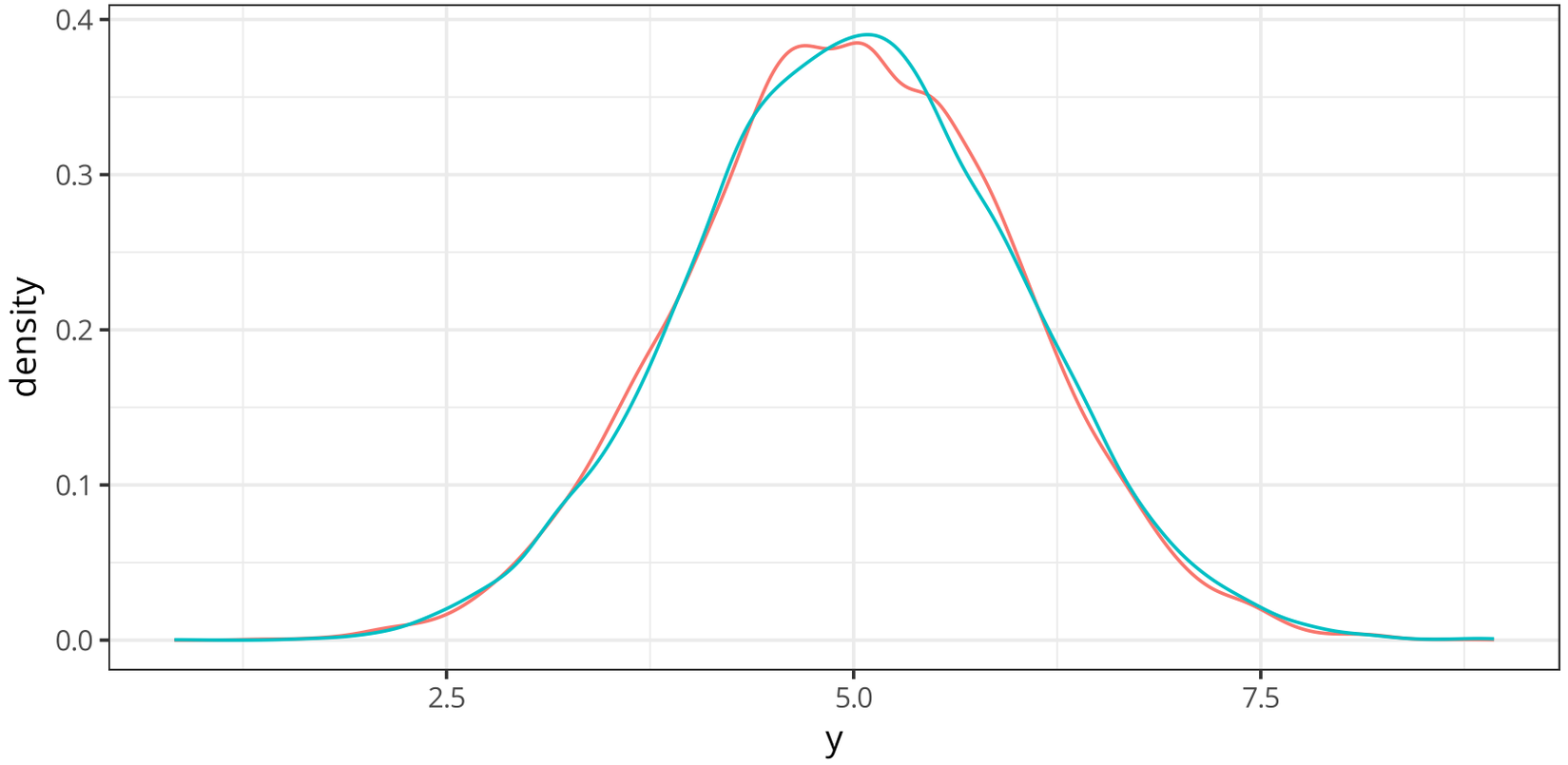
$$
\begin{aligned}
w_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \alpha_{\text{café}[i]} \\
\alpha_{\text{café}} &\sim \text{Normal}(0, \sigma_{\text{café}})
\end{aligned}
$$

On peut toujours “enlever” la moyenne d’une distribution gaussienne et la considérer comme une constante plus une gaussienne centrée sur zéro.

NB : quand α est défini dans le modèle linéaire, les $\alpha_{\text{café}}$ représentent des déviations de l’intercept moyen. Il faut donc ajouter α et $\alpha_{\text{café}}$ pour obtenir le temps d’attente moyen par café…

Équivalences (encore)

```
1  y1 <- rnorm(n = 1e4, mean = 5, sd = 1)
2  y2 <- rnorm(n = 1e4, mean = 0, sd = 1) + 5
3
4  data.frame(y1 = y1, y2 = y2) %>%
5      pivot_longer(cols = 1:2, names_to = "x", values_to = "y") %>%
6      ggplot(aes(x = y, colour = x) ) +
7      geom_density(show.legend = FALSE)
```

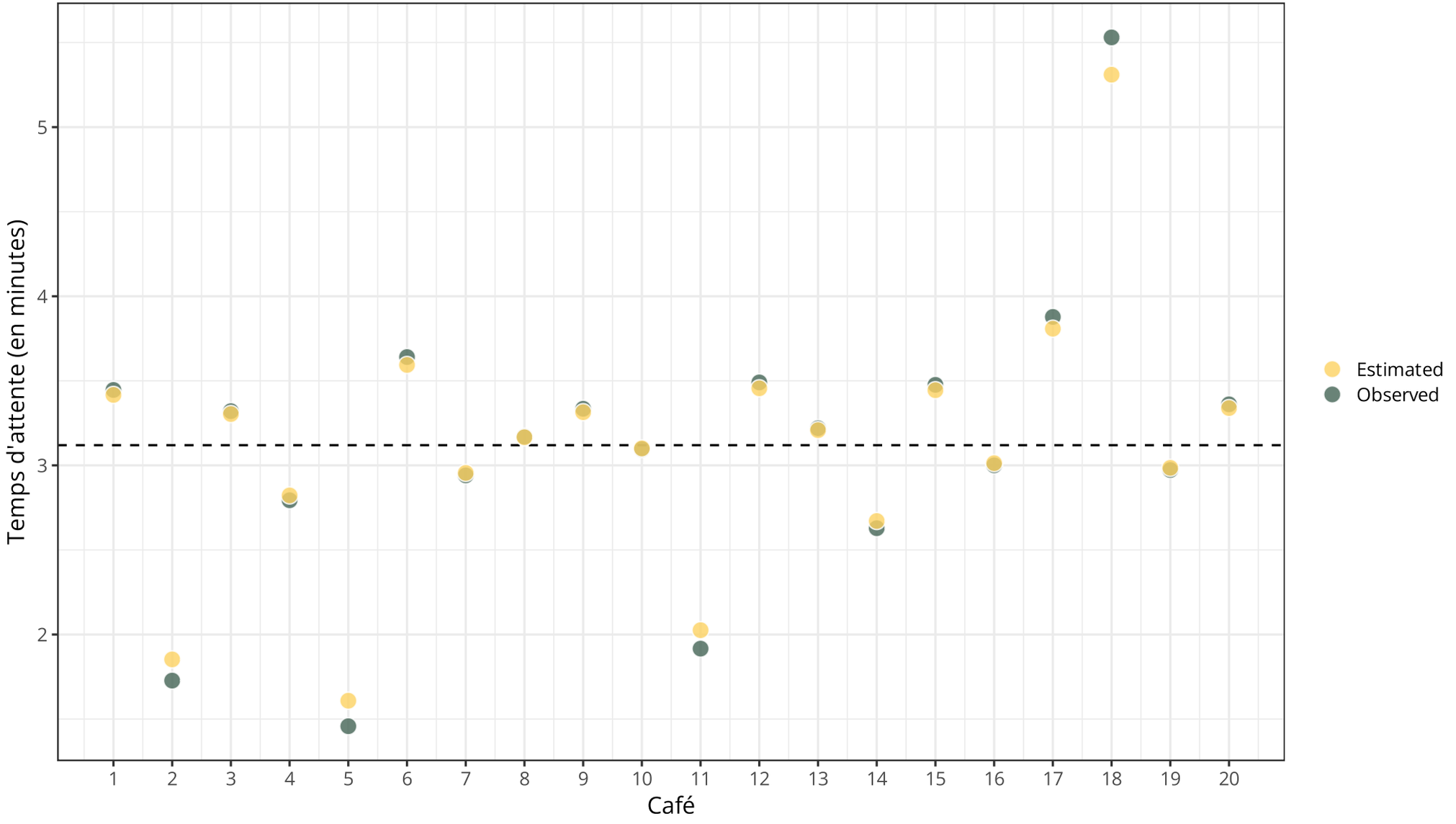


Modèle multi-niveaux

```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,
  chains = 4, cores = 4
  )
```

Ce modèle a 23 paramètres, l'intercept général α, la variabilité résiduelle σ, la variabilité entre les cafés $\sigma_{\text{café}}$, et un intercept par café.

Shrinkage


Temps d'attente (en minutes)
2
3
4
5
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

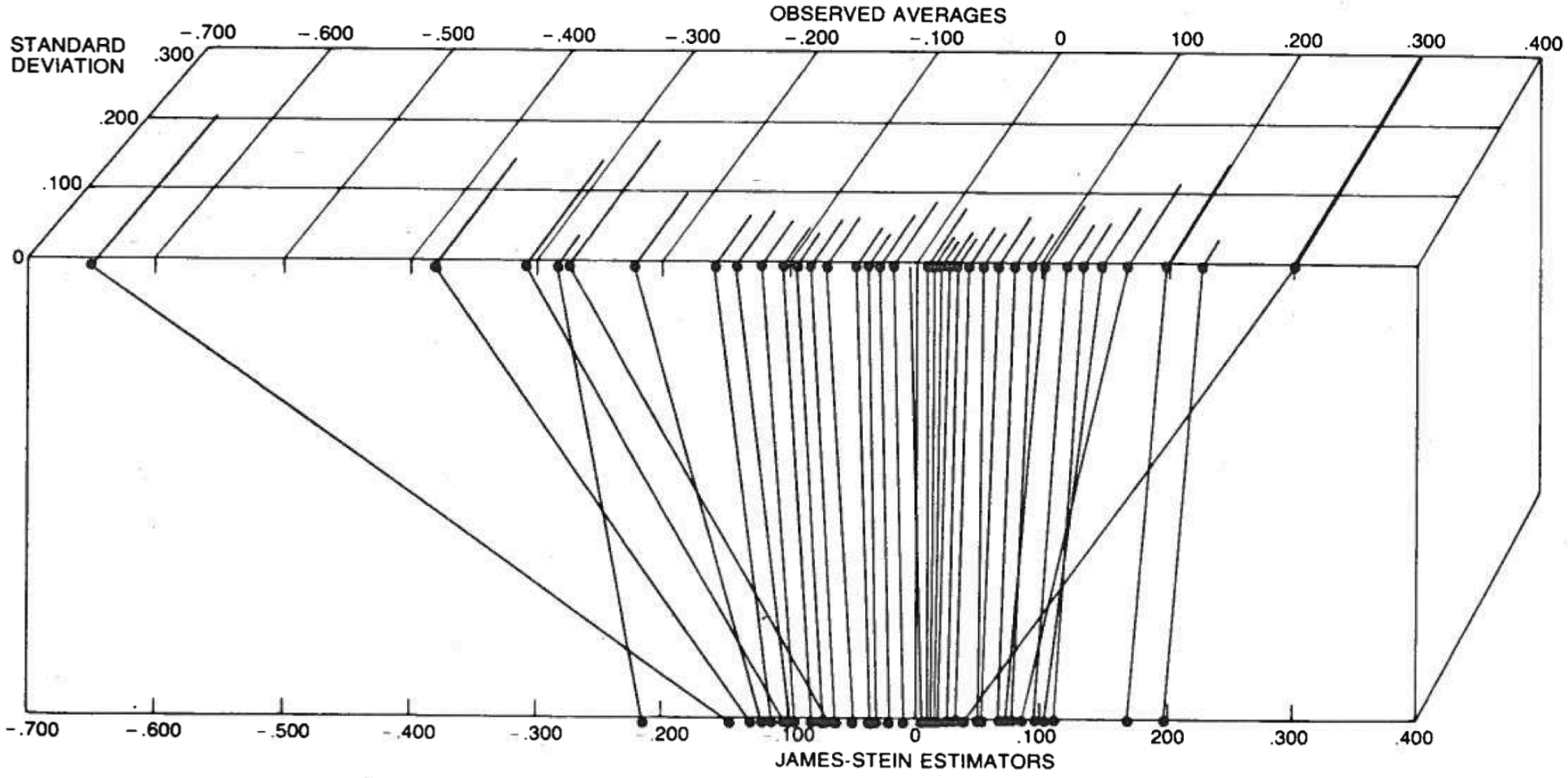
The best guess about the future is usually obtained by computing the average of past events. Stein's paradox defines circumstances in which there are estimators better than the arithmetic average

by Bradley Efron and Carl Morris

L'estimateur James-Stein est défini comme $z = \bar{y} + c(y - \bar{y})$, où $\bar{y}$ désigne la moyenne de l'échantillon, y une observation individuelle, et c une constante, le **shrinking factor** (Efron & Morris, 1977).

Shrinkage magic (Efron & Morris, 1977)

Le shrinking factor est déterminé à la fois par la variabilité (imprécision) de la mesure (e.g., son écart-type) et par la distance à l'estimation moyenne (i.e., $y - \bar{y}$). En d'autres termes, cet estimateur fait moins "confiance" (i.e., accorde moins de poids) aux observations imprécises et/ou extrêmes. En pratique, le shrinkage agit comme une protection contre le sur-apprentissage (overfitting).



Pooling

Le **shrinkage** observé slide précédente est dû à des phénomènes de partage (pooling) de l'information entre les cafés. L'estimation de l'intercept pour chaque café informe l'estimation de l'intercept des autres cafés, ainsi que l'estimation de l'intercept général (i.e., la moyenne générale).

On distingue en général trois perspectives (ou stratégies) :

- **Complete pooling** : on suppose que le temps d'attente est invariant, on estime un intercept commun (`mod1`).
- **No pooling** : on suppose que les temps d'attente de chaque café sont uniques et indépendants : on estime un intercept par café, mais sans informer le niveau supérieur (`mod2`).
- **Partial pooling** : on utilise un prior adaptatif, comme dans l'exemple précédent (`mod3`).

La stratégie **complete pooling** en général underfitte les données (faibles capacités de prédiction) tandis que le stratégie **no pooling** revient à overfitter les données (faibles capacités de prédiction ici aussi). La stratégie **partial pooling** (i.e., celle des modèles multi-niveaux) permet d''équilibrer underfitting et overfitting.

Comparaison de modèles

On peut comparer ces trois modèles en utilisant le WAIC (discuté au Cours n°07).

```r
# calcul du WAIC et ajout du WAIC à chaque modèle
mod1 <- add_criterion(mod1, "waic")
mod2 <- add_criterion(mod2, "waic")
mod3 <- add_criterion(mod3, "waic")

# comparaison des WAIC de chaque modèle
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  -253.9       8.3         18.2    1.5     507.7   16.6
mod2   -0.6       1.3  -254.5       8.4         19.5    1.6     509.0   16.8
mod1  -57.4      10.6  -311.2      10.5          2.1    0.4     622.4   21.1
```

On remarque que le modèle 3 a seulement 18 “effective parameters” (pWAIC) et moins de paramètres que le modèle 2, alors qu’il en a en réalité 2 de plus... `posterior_summary(mod3)[3, 1]` nous donne le sigma du prior adaptatif des $\alpha_{\text{café}}$ ($\sigma_{\text{café}} = 0.82$). On remarque que ce sigma est très faible et correspond à assigner un prior très contraignant, ou **régularisateur**.

Comparaison de modèles

On compare les estimations du premier modèle (complete pooling model) et du troisième modèle (partial pooling model).

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

```
           Estimate  Est.Error     Q2.5    Q97.5
b_Intercept  3.12029 0.08305618 2.961966 3.278956
sigma        1.14329 0.05821889 1.035996 1.263067
```

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

```
            Estimate  Est.Error      Q2.5     Q97.5
b_Intercept 3.1192094 0.20762244 2.7129150 3.5327098
sigma       0.8220335 0.04318892 0.7431579 0.9115307
```

Les deux modèles font la même prédiction (en moyenne) pour α, mais le modèle 3 est plus incertain de sa prédiction que le modèle 1 (voir l'erreur standard pour α)...

L'estimation de σ du modèle 3 est plus petite que celle du modèle 1 car le modèle 3 **décompose** la variabilité non expliquée en deux sources : la variabilité du temps d'attente entre les cafés $\sigma_{\text{café}}$ et la variabilité résiduelle σ.

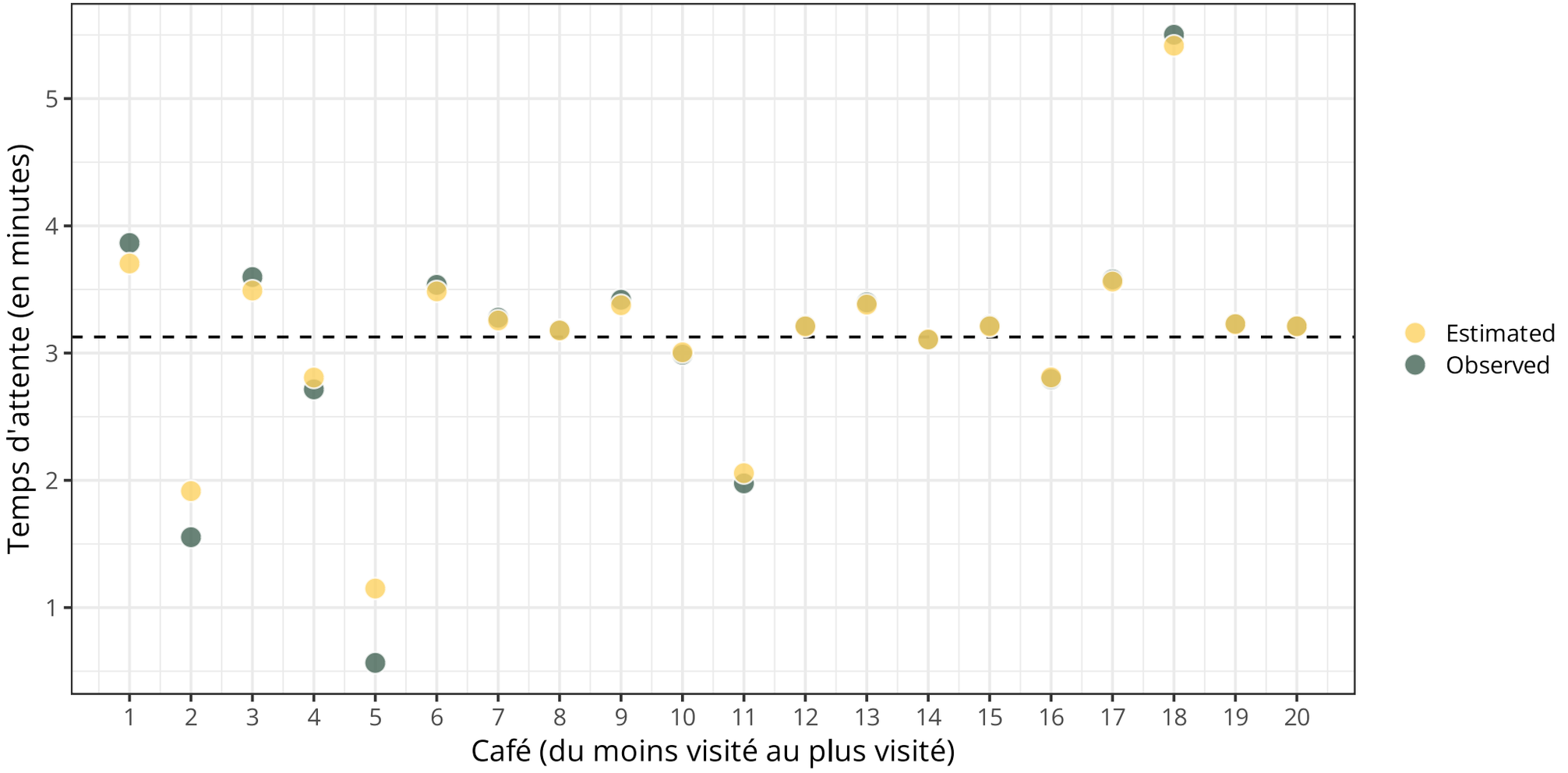
Robot et café

Imaginons que notre robot ne visite pas tous les cafés le même nombre de fois (comme dans le cas précédent) mais qu'il visite plus souvent les cafés proches de chez lui...

```
df2 <- open_data(robot_unequal) # nouveau jeu de données

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,
  chains = 4, cores = 4
  )
```

Shrinkage

On observe que les cafés qui sont souvent visités (à droite) subissent moins l'effet du **shrinkage**. Leur estimation est moins "tirée" vers la moyenne générale que les estimations des cafés les moins souvent visités (à gauche).



Aparté : effets fixes et effets aléatoires

Cinq définitions (contradictoires) relevées par 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) suggèrent plutôt l'utilisation des termes de **constant effects** et **varying effects**, et de toujours utiliser la modélisation multi-niveaux, en considérant que ce qu'on appelle **effet fixe** peut simplement être considéré comme un **effet aléatoire** dont la variance serait égale à 0.

Régularisation et terminologie

Le fait de faire varier les intercepts par café est simplement une autre manière de régulariser (de manière adaptative), c'est à dire de diminuer le poids accordé aux données dans l'estimation. Le modèle devient à même d'estimer à quel point les groupes (ici, les cafés) sont différents, tout en estimant les caractéristiques de chaque café...

Différence entre les **cross-classified** (ou "crossed") multilevel models et **nested or hierarchical** multilevel models. Le premier type de modèle concerne des données structurées selon deux (ou plus) facteurs aléatoires non "nichés". Le deuxième type de modèles concerne des données structurées de manière hiérarchique (e.g., un élève dans une classe dans une école dans une ville...). Voir cette discussion pour plus de détails.

Les deux types de modèles s'écrivent cependant de manière similaire, sur plusieurs "niveaux". Le terme "multi-niveaux" (dans notre terminologie) fait donc référence à la structure du modèle, à sa spécification. À distinguer de la structure des données.

Exemple de modèle “cross-classified”

On pourrait se poser la question de savoir si la récence des cafés (leur âge) ne serait pas une source de variabilité non contrôlée ? Il suffit d’ajouter un intercept qui varie par âge, et de lui attribuer un prior adaptatif.

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

Robot et café : varying intercept + varying slope

On s'intéresse maintenant à l'effet du moment de la journée sur le temps d'attente. Attend-on plus le matin, ou l'après-midi ?

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

Où A_i est une dummy variable codée 0/1 pour le matin et l'après-midi et où $\beta_{\text{café}}$ est donc un paramètre de différence (i.e., une pente) entre le matin et l'après-midi.

Remarque : on sait que les cafés ont des intercepts et des pentes qui co-varient… Les cafés populaires seront surchargés le matin et beaucoup moins l'après-midi, résultant en une pente importante. Ces cafés auront aussi un temps d'attente moyen plus long (i.e., un intercept plus grand). Dans ces cafés, α est grand et β est loin de zéro. À l'inverse, dans un café peu populaire, le temps d'attente sera faible, ainsi que la différence entre matin et après-midi.

On pourrait donc utiliser la co-variation entre intercept et pente pour faire de meilleures inférences. Autrement dit, faire en sorte que l'estimation de l'intercept informe celle de la pente, et réciproquement.

Robot et café : varying intercept + varying slope

On s'intéresse maintenant à l'effet du moment de la journée sur le temps d'attente. Attend-on plus le matin, ou l'après-midi ?

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

La troisième ligne postule que chaque café a un intercept $\alpha_{\text{café}}$ et une pente $\beta_{\text{café}}$, définis par un prior Gaussien bivarié (i.e., à deux dimensions) ayant comme moyennes α et β et comme matrice de covariance $\mathbf{S}$.

Aparté : distribution gaussienne multivariée

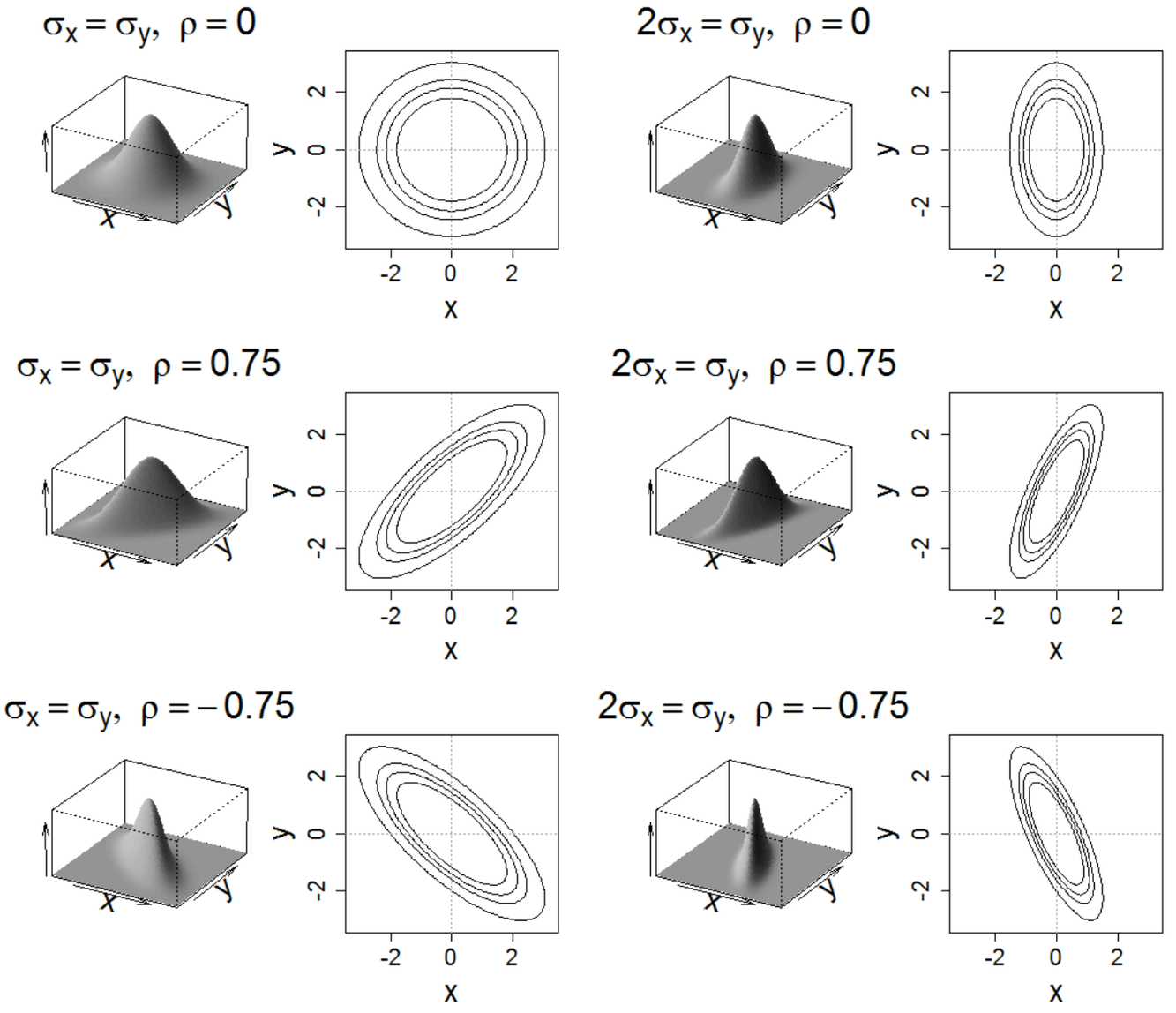
$$\mathbf{x} \sim \square(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Où $\boldsymbol{\mu}$ est un vecteur (à k dimensions) de moyennes, par exemple: `mu <- c(a, b)`.

$\boldsymbol{\Sigma}$ est une matrice de covariance de $k \times k$ dimensions, et qui correspond à la matrice donnée par la fonction vcov().

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

Aparté : distribution gaussienne multivariée



Aparté : distribution gaussienne multivariée

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

Cette matrice peut se construire de deux manières différentes, strictement équivalentes.

```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 : distribution gaussienne multivariée

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

La deuxième méthode est pratique car elle considère séparément les écart-types et les corrélations.

```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
```

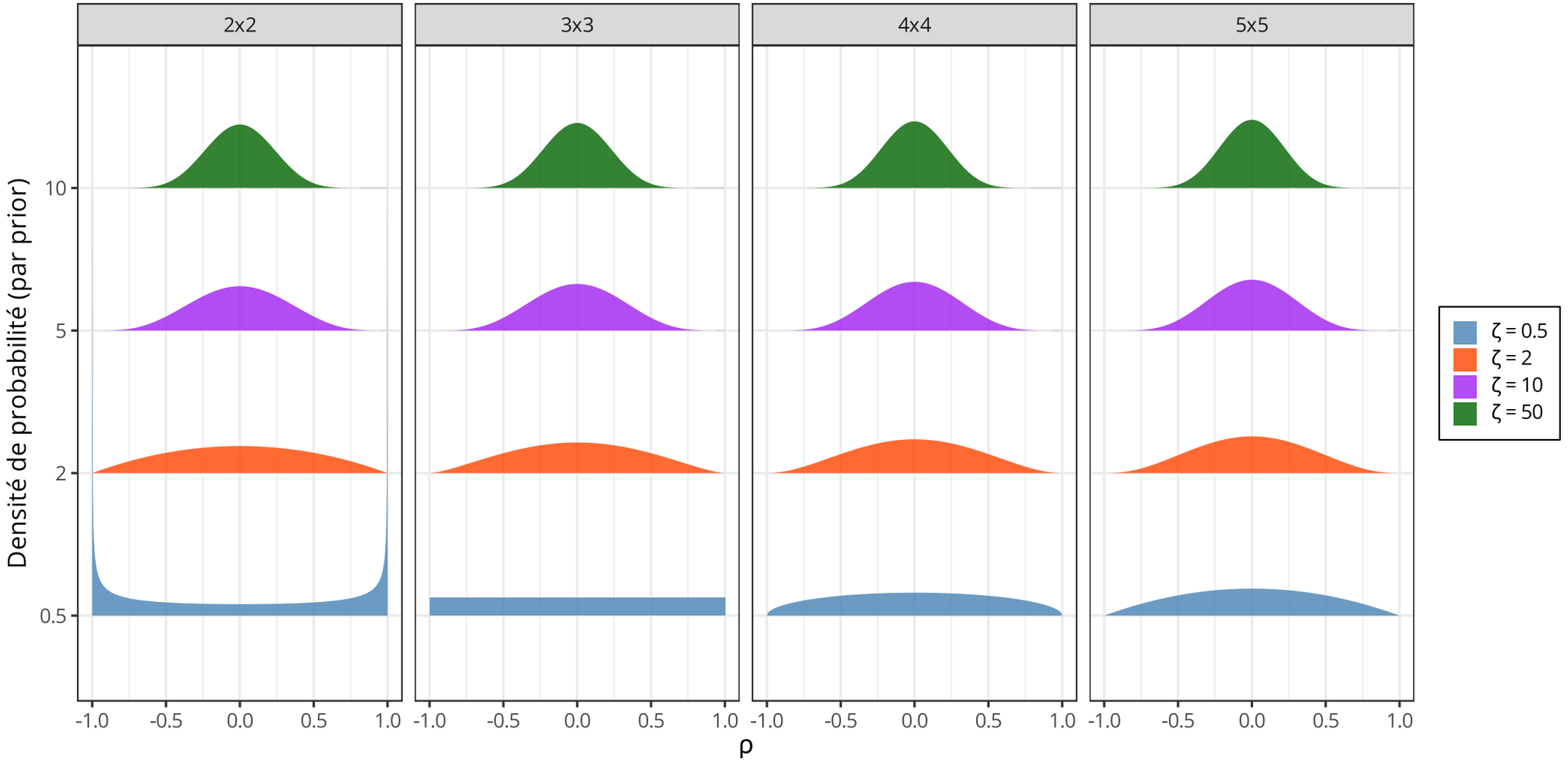
Robot et café : varying intercept + varying slope

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

$\mathbf{S}$ est définie en factorisant σ_α, σ_β, et la matrice de corrélation $\mathbf{R}$. La suite du modèle définit simplement les priors pour les effets constants. La dernière ligne spécifie le prior pour $\mathbf{R}$.

LKJ prior

Prior proposé par Lewandowski et al. (2009). Un seul paramètre ζ (zeta) spécifie la concentration de la distribution du coefficient de corrélation. Le prior $\mathrm{LKJ}(2)$ définit un prior peu informatif pour ρ (rho) qui est sceptique des corrélations extrêmes (i.e., des valeurs proches de -1 ou 1).



Rappels de syntaxe

Le paquet `brms` utilise la même syntaxe que les fonctions de base R (comme `lm`) ou que le paquet `lme4`.

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

La partie gauche représente notre variable dépendante (ou “outcome”, i.e., ce qu’on essaye de prédire).

La partie droite permet de définir les prédicteurs. L’intercept est généralement implicite, de sorte que les deux écritures ci-dessous sont équivalentes.

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

Rappels de syntaxe

La première partie de la partie droite de la formule représente les effets constants (effets fixes), tandis que la seconde partie (entre parenthèses) représente les effets “variants” ou “variables” (effets aléatoires).

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

Le premier modèle ci-dessus contient seulement un intercept variable, qui varie par `Subject`. Le deuxième modèle contient également un intercept variable, mais aussi une pente variable pour l’effet de `Days`.

Rappels de syntaxe

Lorsqu'on inclut plusieurs effets variants (e.g., un intercept et une pente variables), `brms` postule qu'on souhaite aussi estimer la corrélation entre ces deux effets. Dans le cas contraire, on peut supprimer cette corrélation (i.e., la fixer à 0) en utilisant `||`.

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

Les modèles précédents postulaient un modèle génératif Gaussien. Ce postulat peut être changé facilement en spécifiant la fonction souhaitée via l'argument `family`.

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

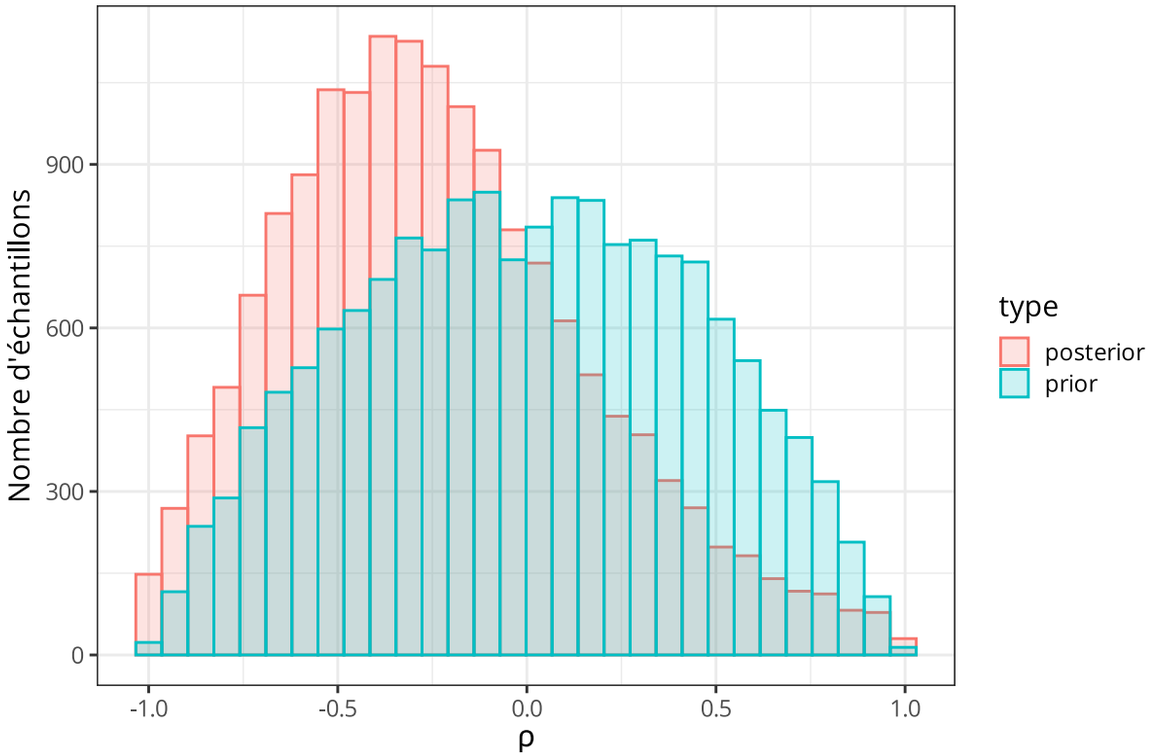
Modèle brms

On spécifie un intercept et une pente (pour l'effet d'`afternoon`) qui varient par `cafe`.

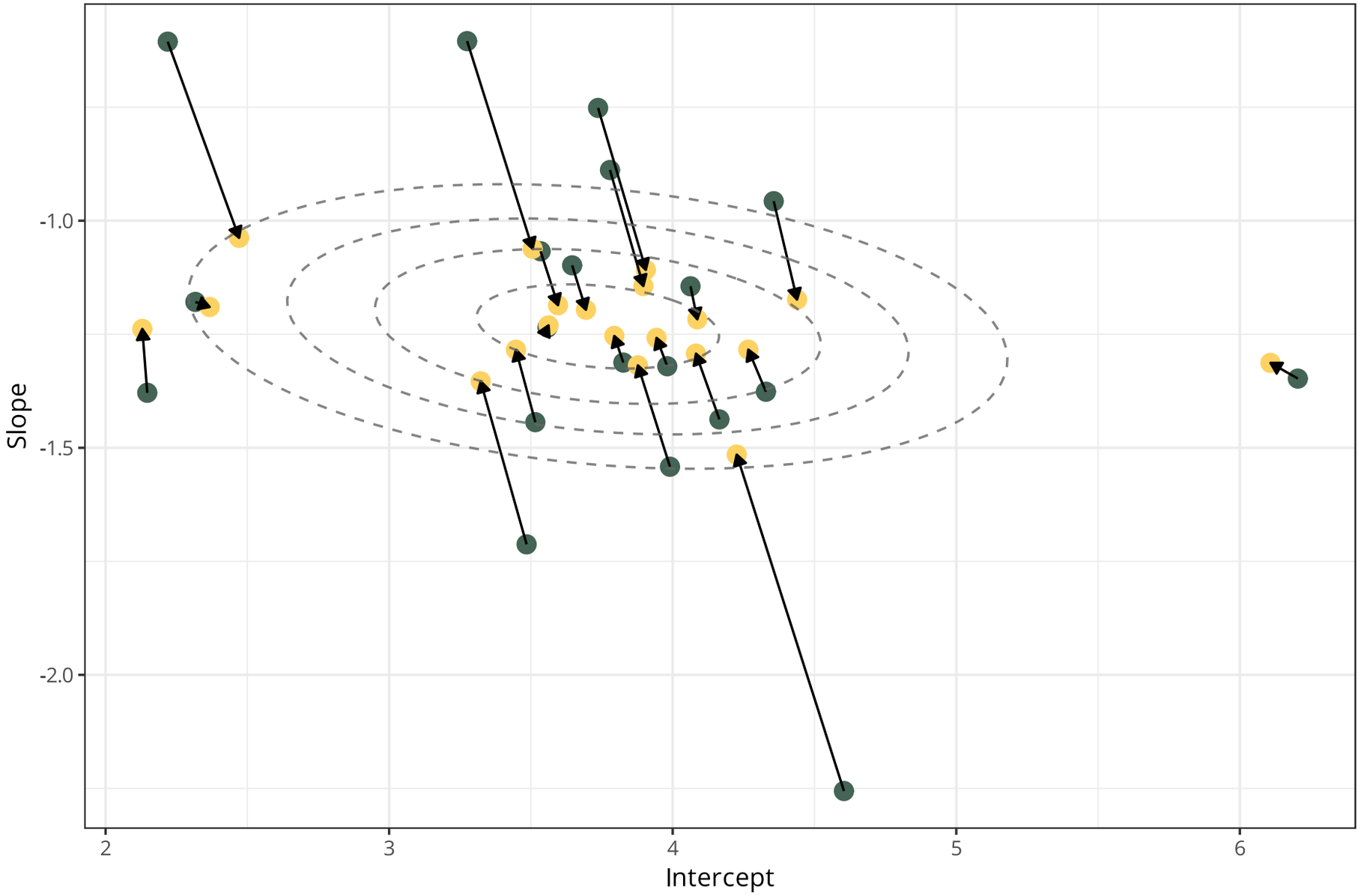
```
mod5 <- brm(
  formula = wait ~ 1 + afternoon + (1 + afternoon | cafe),
  prior = c(
    prior(normal(0, 10), class = Intercept),
    prior(normal(0, 10), class = b),
    prior(cauchy(0, 2), class = sigma),
    prior(cauchy(0, 2), class = sd)
    ),
  data = df,
  warmup = 1000, iter = 5000,
  chains = 4, cores = 4
  )
```

Distribution postérieure

```r
post <- as_draws_df(x = mod5) # extracts 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 = "Nombre d'échantillons")
```



Shrinkage en deux dimensions



Comparaison de modèles

On compare le premier modèle (complete pooling model), le troisième modèle (partial pooling model), et le dernier modèle (avec intercept et pente variable).

```
# comparaison des WAIC de chaque modèle
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  -154.9      10.1         26.4    2.6     309.9   20.1
mod3  -98.9       8.3  -253.9       8.3         18.2    1.5     507.7   16.6
mod2  -99.5       8.3  -254.5       8.4         19.5    1.6     509.0   16.8
mod1 -156.3      13.7  -311.2      10.5          2.1    0.4     622.4   21.1
```

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

```
        mod1         mod2         mod3         mod5
1.339634e-68 5.838915e-44 1.087725e-43 1.000000e+00
```

Comparaison de modèles

L'estimation du temps d'attente moyen est plus incertaine lorsqu'on prend en compte de nouvelles sources d'erreur. Cependant, l'erreur du modèle (i.e., ce qui n'est pas expliqué), la variation résiduelle σ, diminue...

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

```
           Estimate  Est.Error     Q2.5    Q97.5
b_Intercept  3.12029 0.08305618 2.961966 3.278956
sigma        1.14329 0.05821889 1.035996 1.263067
```

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

```
            Estimate  Est.Error      Q2.5     Q97.5
b_Intercept 3.1192094 0.20762244 2.7129150 3.5327098
sigma       0.8220335 0.04318892 0.7431579 0.9115307
```

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

```
             Estimate  Est.Error       Q2.5      Q97.5
b_Intercept  3.7362476 0.21873485  3.3013415  4.1752168
b_afternoon -1.2328637 0.08678994 -1.4036481 -1.0601241
sigma        0.4895081 0.02684615  0.4394686  0.5450169
```

Conclusions

Les modèles multi-niveaux (ou “modèles mixtes”) sont des extensions naturelles des modèles de régression classiques, où les paramètres de ces derniers se voient eux-même attribués des “modèles”, gouvernés par des hyper-paramètres.

Cette extension permet de faire des prédictions plus précises en prenant en compte la variabilité liée aux groupes ou structures (clusters) présent(e)s dans les données. Autrement dit, en modélisant les populations d’où sont tirés les effets aléatoires (e.g., la population de participants ou de stimuli).

Un modèle de régression classique est équivalent à un modèle multi-niveaux où la variabilité des effets aléatoires serait fixée à 0.

La cadre bayésien permet une interprétation naturelle des distributions desquelles proviennent les effets aléatoires (varying effects). En effet, ces distributions peuvent être interprétées comme des distributions a priori, dont les paramètres sont estimés à partir des données.

Mise en pratique - Modèles multi-niveaux généralisés

Travailler avec des sujets humains implique un minimum de coopération réciproque. Mais ce n'est pas toujours le cas. Une partie non-négligeable des étudiants qui s'inscrivent pour passer des expériences de psychologie ne se présentent pas le jour prévu... On a voulu estimer la **probabilité de présence d'un étudiant inscrit** en fonction de l'envoi (ou non) d'un mail de rappel (cet exemple est présenté en détails dans deux blogposts, accessibles ici, et ici).

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

# import des données
absence_data <- open_data(absence_multilevel)

# on affiche 12 lignes "au hasard" dans ces données
absence_data %>% slice_sample(prop = 1) %>% head()
```

```
  reminder researcher presence absence total
1     -0.5          8       38      54    92
2      0.5          7       64      16    80
3      0.5          8       81      11    92
4     -0.5          4       61      34    95
5      0.5          6       82       6    88
6     -0.5         10       23      58    81
```

Mise en pratique - absentéisme expérimental

$$
\begin{aligned}
y_i &\sim \text{Binomial}(n_i, p_i) \\
\text{logit}(p_i) &= \alpha_{\text{researcher}_{[i]}} + \beta_{\text{researcher}_{[i]}} \times \text{reminder}_i \\
\begin{bmatrix} \alpha_{\text{researcher}} \\ \beta_{\text{researcher}} \end{bmatrix} &\sim \text{MVNormal}\left(\begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \mathbf{S} \right) \\
\mathbf{S} &= \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \mathbf{R} \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \\
\alpha &\sim \text{Normal}(0, 1) \\
\beta &\sim \text{Normal}(0, 1) \\
(\sigma_\alpha, \sigma_\beta) &\sim \text{HalfCauchy}(0, 1) \\
\mathbf{R} &\sim \text{LKJcorr}(2)
\end{aligned}
$$

Il s'agit du même modèle de régression logistique vu au Cours n°06, avec une fonction de lien logit, mais cette fois-ci sur plusieurs niveaux.

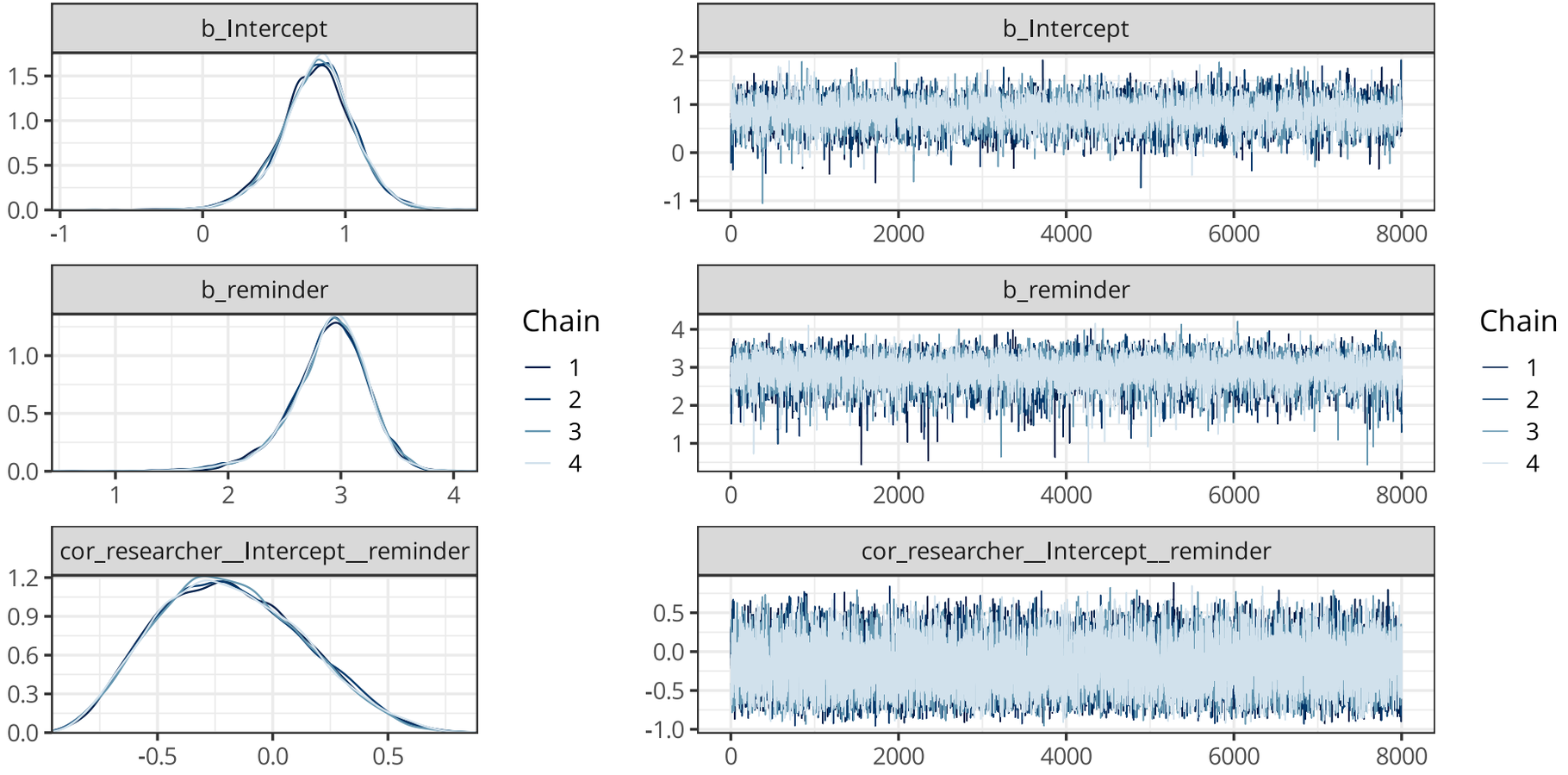
Mise en pratique - absentéisme expérimental

```r
prior6 <- c(
    prior(normal(0, 1), class = Intercept),
    prior(normal(0, 1), class = b),
    prior(cauchy(0, 1), class = sd),
    prior(lkj(2), class = cor)
    )
```

```r
mod6 <- brm(
    formula = presence | trials(total) ~ 1 + reminder + (1 + reminder | researcher),
    family = binomial(link = "logit"),
    prior = prior6,
    data = absence_data,
    sample_prior = TRUE,
    warmup = 2000, iter = 10000,
    chains = 4, cores = 4,
    control = list(adapt_delta = 0.95),
    backend = "cmdstanr"
    )
```

Mise en pratique - absentéisme expérimental

```
mod6 %>%
    plot(
        combo = c("dens_overlay", "trace"), pars = c("^b_", "^cor_"), widths = c(1, 1.5),
        theme = theme_bw(base_size = 16, base_family = "Open Sans")
        )
```



Mise en pratique - absentéisme expérimental

Attention, les estimations ci-dessous sont dans l'espace log-odds...

```
posterior_summary(x = mod6, pars = c("^b_", "^sd_") )
```

```
                         Estimate Est.Error      Q2.5    Q97.5
b_Intercept             0.8048514 0.2601641 0.2620946 1.304932
b_reminder              2.8921913 0.3472082 2.1004525 3.487141
sd_researcher__Intercept 0.7821731 0.2323250 0.4528986 1.344695
sd_researcher__reminder  0.9024132 0.3337226 0.4323950 1.710281
```

Afin de pouvoir les interpréter il faut appliquer la transformation logit-inverse. Par exemple, la probabilité de présence en moyenne (i.e., quel que soit le chercheur et pour toutes conditions confondues) est égale à $p = \exp(\alpha)/(1 + \exp(\alpha))$.

```
a <- fixef(mod6)[1] # on récupère la valeur de l'intercept
exp(a) / (1 + exp(a) ) # on "convertit" l'intercept en probabilité (équivalent à plogis(a))
```

```
[1] 0.6910113
```

Mise en pratique - absentéisme expérimental

On s'est ensuite interrogé sur l'effet du mail de rappel. Ici encore, on ne peut pas interpréter la pente directement... mais on sait que $\exp(\beta)$ nous donne un odds ratio (i.e., un rapport de cotes).

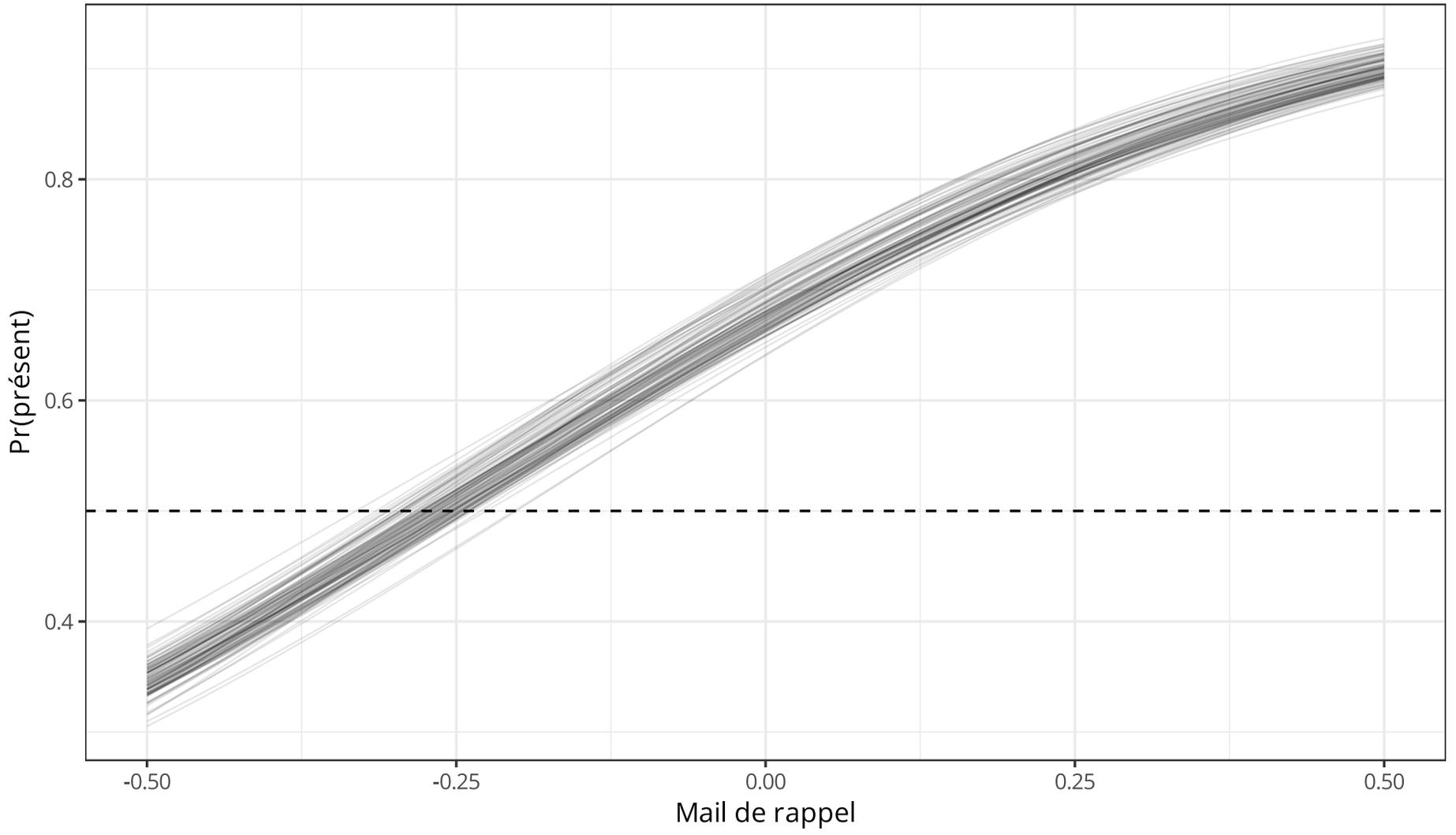
```
1 fixef(mod6)[2, c(1, 3, 4)] %>% exp()
```

```
 Estimate      Q2.5     Q97.5
18.032782  8.169866 32.692338
```

Envoyer un mail de rappel multiplie par environ 18 le rapport des cotes.

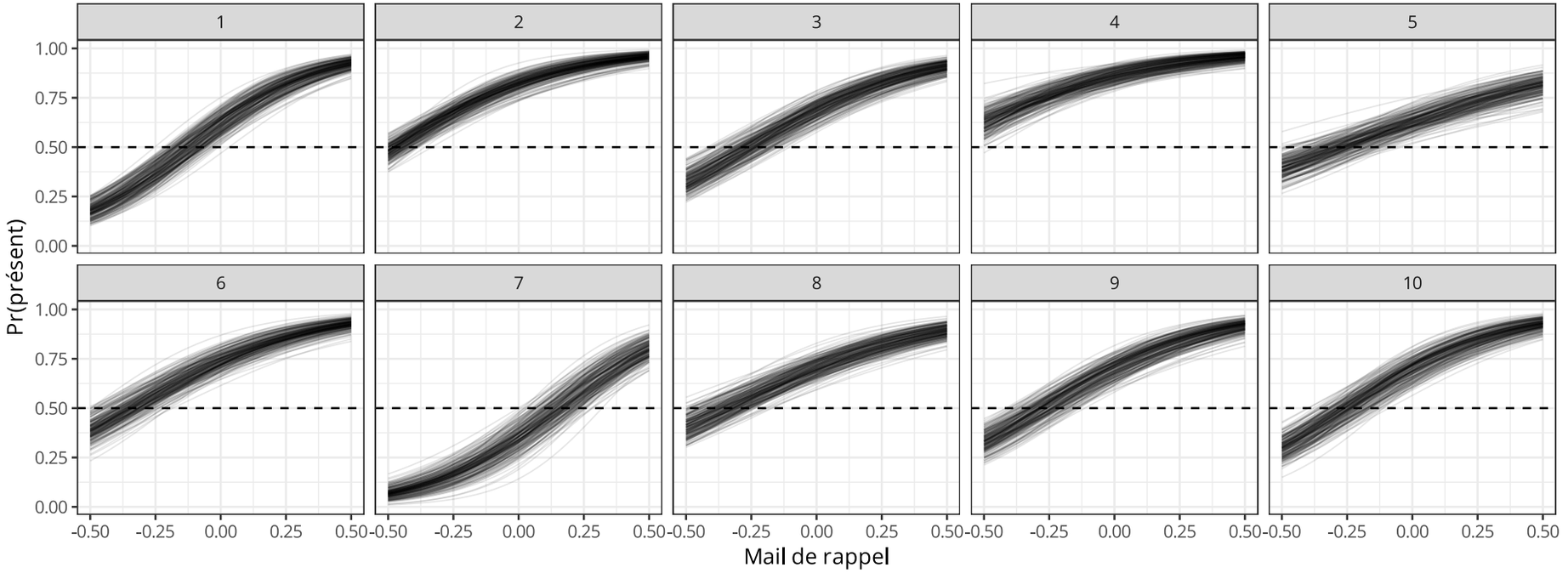
Représenter les prédictions du modèle

Une manière de représenter les prédictions du modèle est de plotter directement quelques échantillons issus de la distribution a posteriori. On appelle ce genre de plot un “spaghetti plot”.



Représenter les prédictions du modèle

```
absence_data %>%
    group_by(researcher, total) %>%
    data_grid(reminder = seq_range(reminder, n = 1e2) ) %>%
    add_linpred_draws(object = mod6, newdata = ., ndraws = 200) %>%
    mutate(estimate = plogis(.linpred) ) %>%
    ggplot(aes(x = reminder, y = estimate, group = .draw) ) +
    geom_hline(yintercept = 0.5, lty = 2) +
    geom_line(aes(y = estimate, group = .draw), size = 0.5, alpha = 0.1) +
    facet_wrap(~researcher, nrow = 2) +
    labs(x = "Mail de rappel", y = "Pr(présent)")
```



Test d'hypothèse - 1

Plusieurs manières de tester des hypothèses avec `brms`. La fonction `hypothesis()` calcule un **evidence ratio** (équivalent au Bayes factor). Lorsque l'hypothèse testée est une hypothèse ponctuelle (on teste une valeur précise du paramètre, e.g., $\theta = 0$), cet **evidence ratio** est approximé via la méthode de **Savage-Dickey**. Cette méthode consiste simplement à comparer la densité du point testé accordée par le prior à la densité accordée par la distribution a posteriori.

```
(hyp1 <- hypothesis(x = mod6, hypothesis = "reminder = 0") ) # Savage-Dickey Bayes factor
```

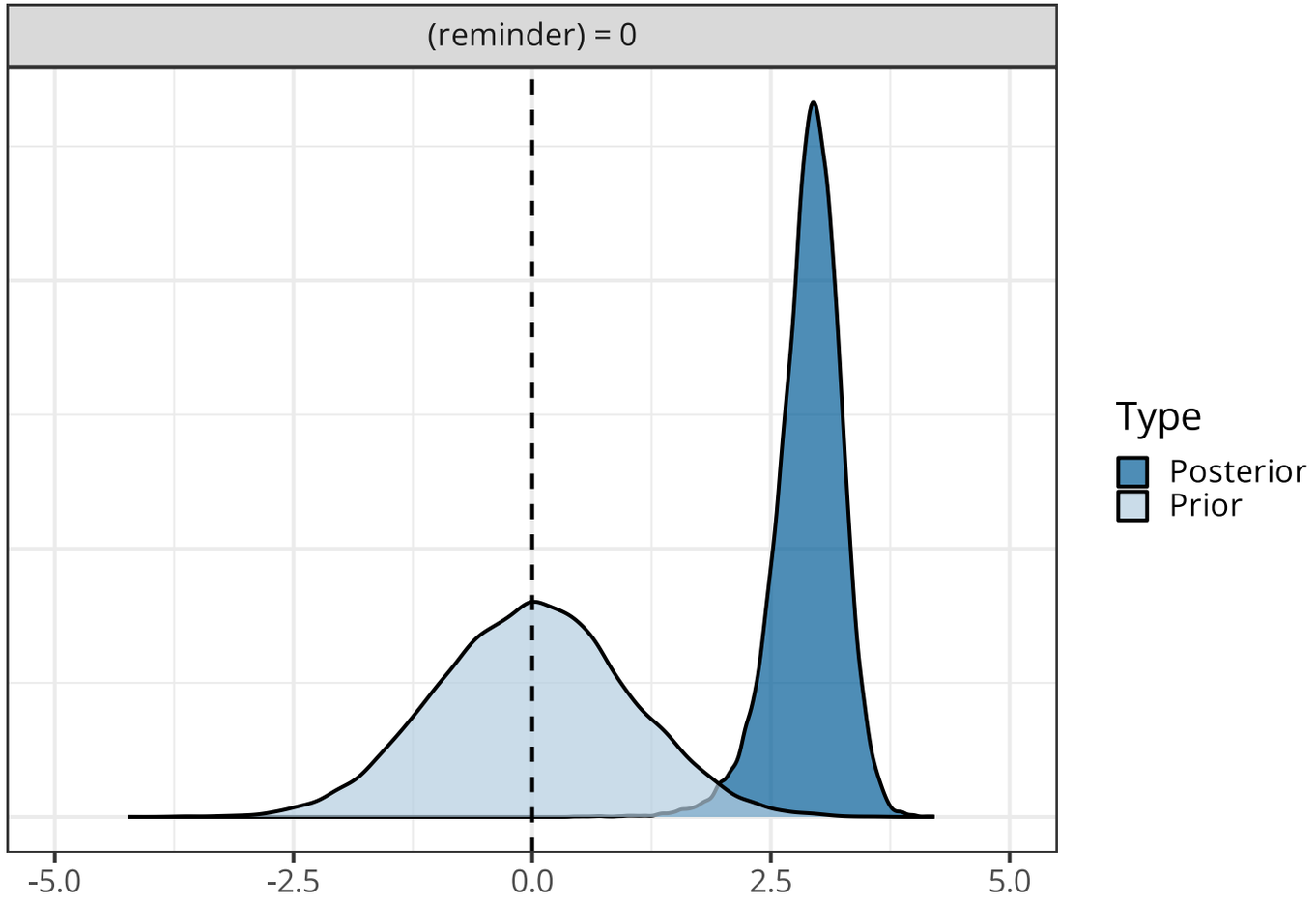
```
Hypothesis Tests for class b:
      Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
1 (reminder) = 0     2.89      0.35      2.1     3.49          0         0    *
---
'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
'*': For one-sided hypotheses, the posterior probability exceeds 95%;
for two-sided hypotheses, the value tested against lies outside the 95%-CI.
Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
1 / hyp1$hypothesis$Evid.Ratio # BF10 = 1 / BF01 (and BF01 = 1 / BF10)
```

```
[1] Inf
```

Test d'hypothèse - 1

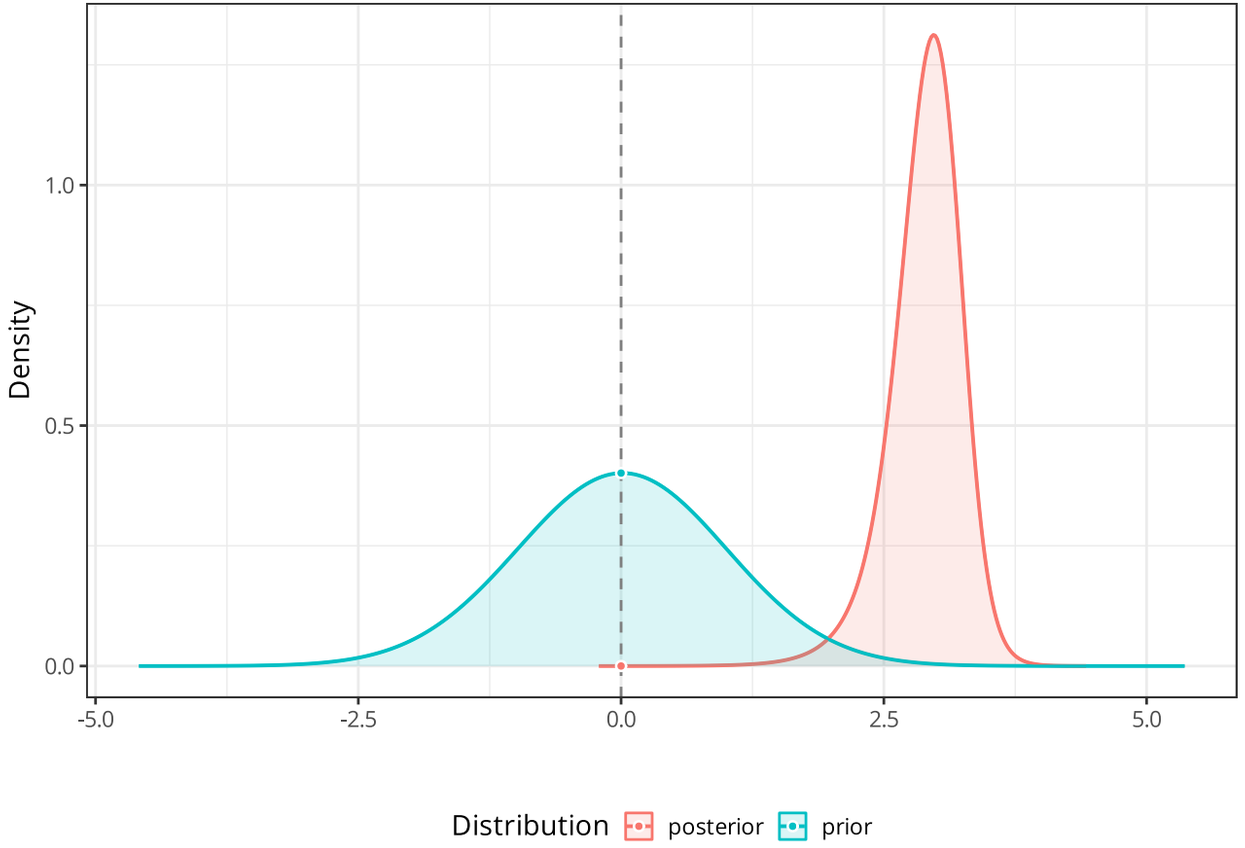
```
1 plot(hyp1, plot = FALSE, theme = theme_bw(base_size = 20, base_family = "Open Sans") )[[1]] +
2   geom_vline(xintercept = 0, linetype = 2) +
3   coord_cartesian(xlim = c(-5, 5) )
```



Test d'hypothèse - 1

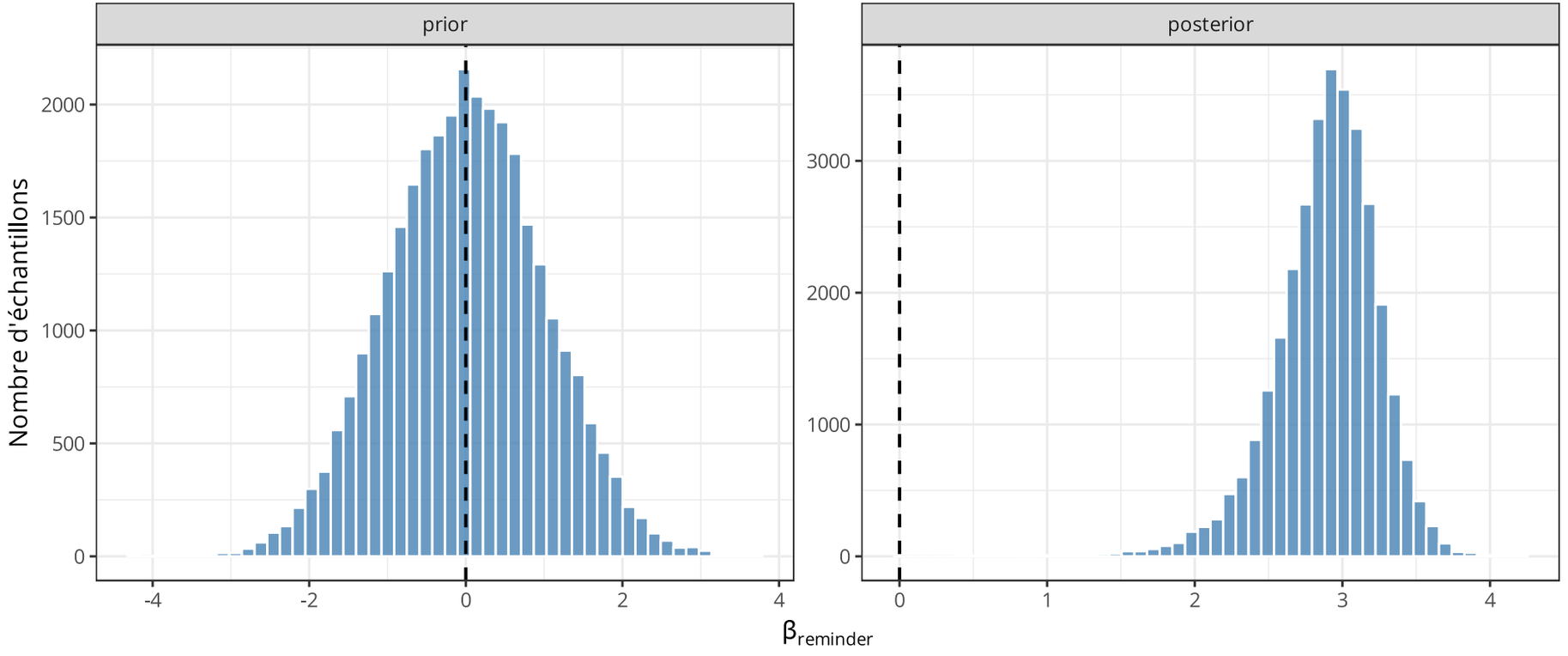
Voir la vignette détaillée du paquet `bayestestR` concernant les facteurs de Bayes : https://easystats.github.io/bayestestR/articles/bayes_factors.html.

```
1 library(bayestestR)
2 bf <- bayesfactor_parameters(posterior = mod6, null = 0)
3 plot(bf)
```



Comparer le prior et le posterior

```r
data.frame(prior = hyp1$prior_samples$H1, posterior = hyp1$samples$H1) %>%
    gather(type, value) %>%
    mutate(type = factor(type, levels = c("prior", "posterior") ) ) %>%
    ggplot(aes(x = value) ) +
    geom_histogram(bins = 50, alpha = 0.8, col = "white", fill = "steelblue") +
    geom_vline(xintercept = 0, lty = 2, size = 1) +
    facet_wrap(~type, scales = "free") +
    labs(x = expression(beta[reminder]), y = "Nombre d'échantillons")
```



Test d'hypothèse - 2

Une deuxième solution consiste à étendre l'approche par comparaison de modèles. Tester une hypothèse revient à comparer deux modèles : un modèle avec l'effet d'intérêt et un modèle sans l'effet d'intérêt.

```
prior7 <- c(
    prior(normal(0, 10), class = Intercept, coef = ""),
    prior(exponential(1), class = sd),
    prior(lkj(2), class = cor)
    )

mod7 <- brm(
    presence | trials(total) ~ 1 + reminder + (1 + reminder | researcher),
    family = binomial(link = "logit"),
    prior = prior7,
    data = absence_data,
    # this line is important for bridgesampling
    save_pars = save_pars(all = TRUE),
    warmup = 2000, iter = 1e4, cores = 4,
    control = list(adapt_delta = 0.95) )

mod8 <- brm(
    presence | trials(total) ~ 1 + (1 + reminder | researcher),
    family = binomial(link = "logit"),
    prior = prior7,
    data = absence_data,
```

Test d'hypothèse - 2

On peut ensuite comparer la vraisemblance marginale de ces modèles, c'est à dire calculer un Bayes factor. Le paquet `brms` propose la méthode `bayes_factor()` qui repose sur une approximation de la vraisemblance marginale via le paquet `bridgesampling` (Gronau et al., 2017).

```
# returns the median BF based on 10 repetitions of the algorithm
bayes_factor(mod7, mod8, repetitions = 10, cores = 10)
```

```
Estimated Bayes factor (based on medians of log marginal likelihood estimates)
 in favor of mod7 over mod8: 2156201.58988
Range of estimates: 1940819.53638 to 2486884.71185
Interquartile range: 114236.99734
```

Comparaison de modèles

On peut également s'intéresser aux capacités de prédiction de ces deux modèles et les comparer en utilisant des critères d'information. La fonction `waic()` calcule le **Widely Applicable Information Criterion** (cf. Cours n°07).

```
1 waic(mod7, mod8, compare = FALSE)
```

```
Output of model 'mod7':

Computed from 32000 by 20 log-likelihood matrix.

          Estimate  SE
elpd_waic    -59.7 2.4
p_waic        10.7 1.4
waic         119.5 4.8

15 (75.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.

Output of model 'mod8':

Computed from 32000 by 20 log-likelihood matrix.

          Estimate  SE
elpd_waic    -59.3 1.6
p_waic        10.3 0.6
waic         118.6 3.2

18 (90.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

Posterior predictive checking

Une autre manière d'examiner les capacités de prédiction d'un modèle est le **posterior predictive checking** (PPC). L'idée est simple : il s'agit de comparer les données observées à des données simulées à partir de la distribution **a posteriori**. Une fois qu'on a une distribution a posteriori sur θ, on peut simuler des données à partir de la **posterior predictive distribution** :

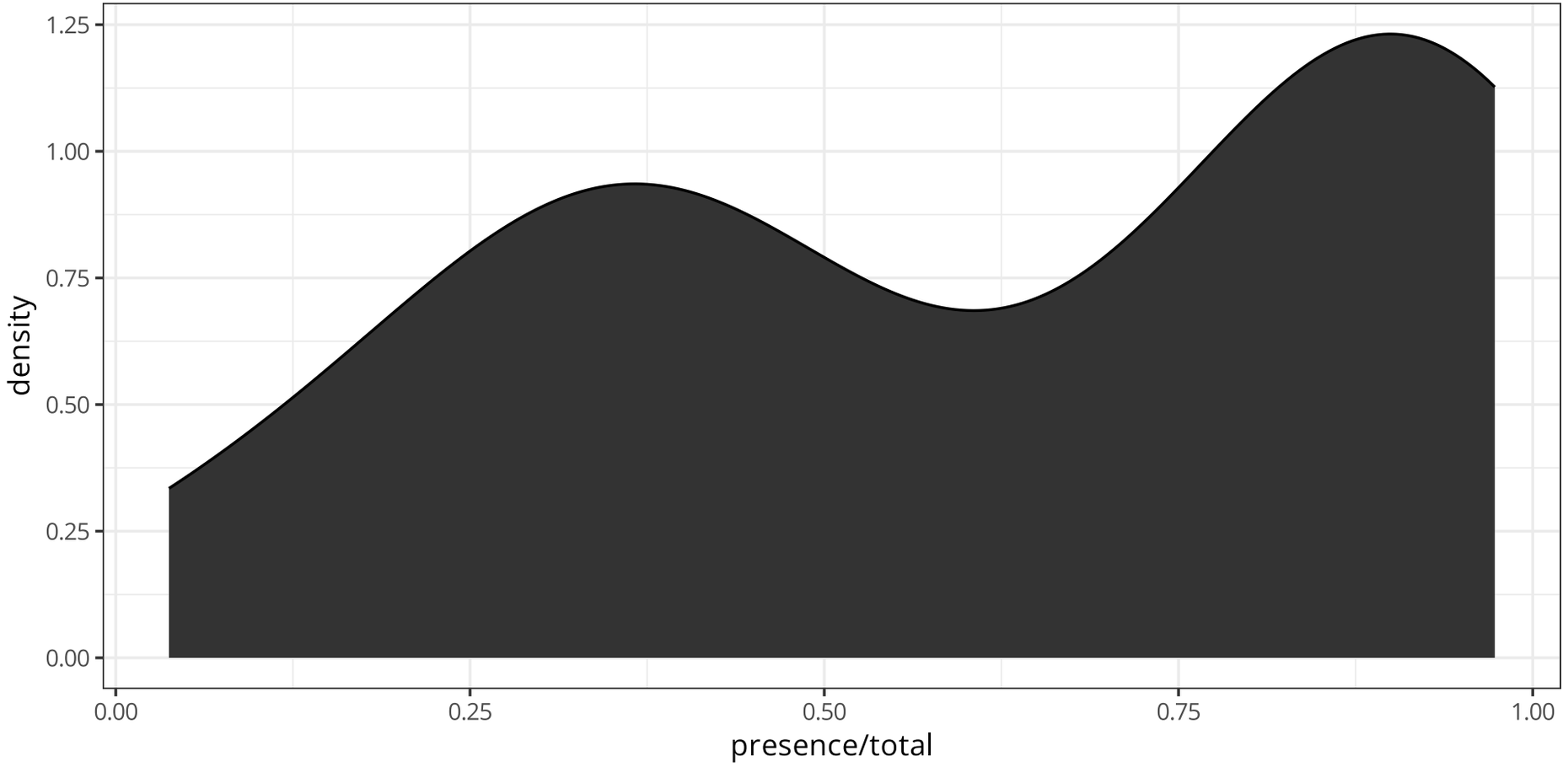
$$p(\tilde{y} \mid y) = \int p(\tilde{y} \mid \theta) p(\theta \mid y) d\theta$$

Si le modèle est un bon modèle, il devrait pouvoir générer des données qui ressemblent aux données qu'on a observées (e.g., Gabry et al., 2019).

Posterior predictive checking

On représente ci-dessous la distribution de nos données.

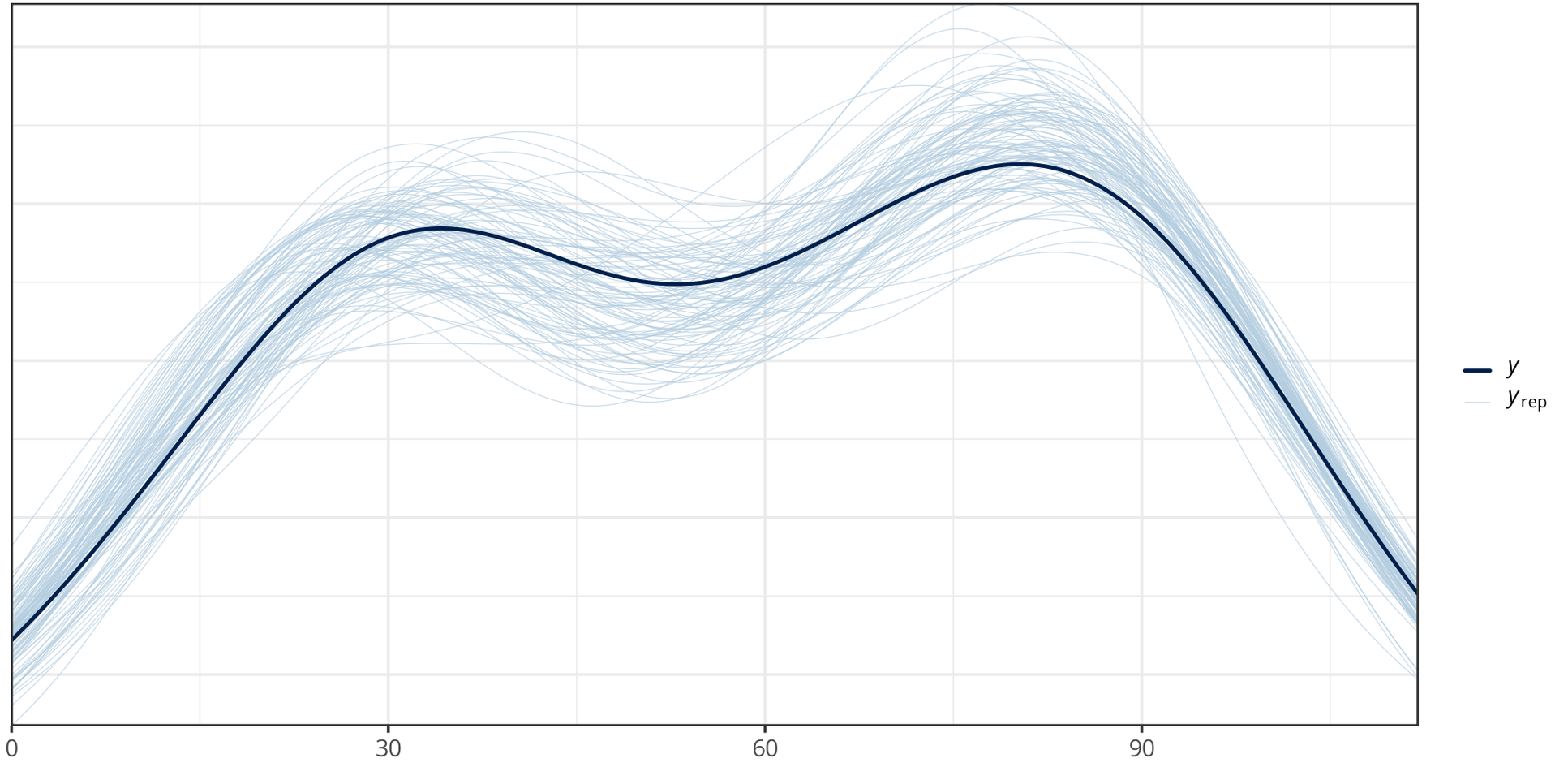
```
1  absence_data %>%
2      ggplot(aes(x = presence / total) ) +
3      geom_density(fill = "grey20")
```



Posterior predictive checking

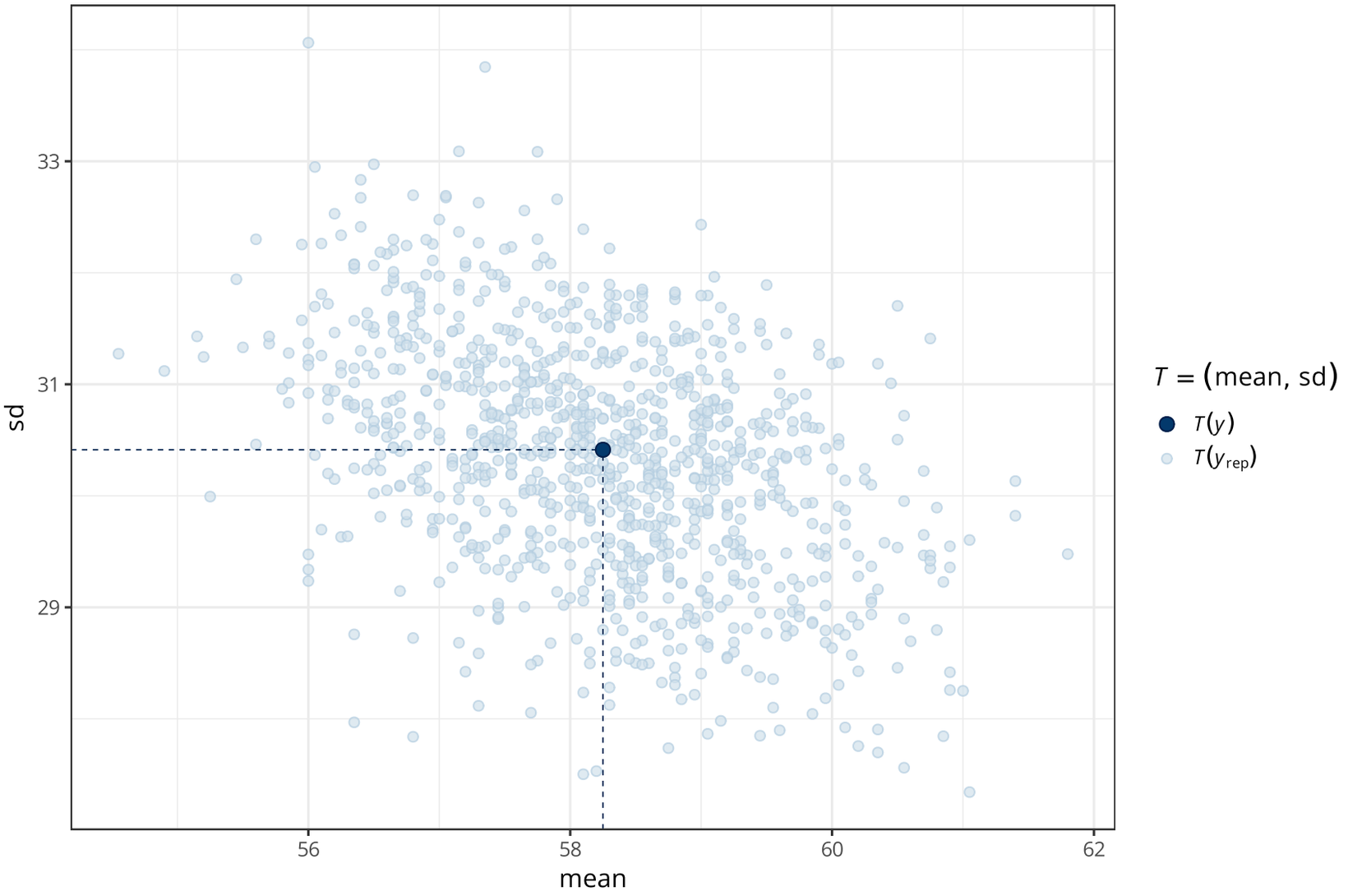
Cette procédure est implémentée dans `brms` via la méthode `pp_check()` qui permet de réaliser de nombreux checks. Par exemple, ci-dessous on compare les prédictions a posteriori (n = 100) aux données observées.

```
1  pp_check(object = mod7, ndraws = 1e2)
```



Posterior predictive checking

```
1  pp_check(object = mod7, ndraws = 1e3, type = "stat_2d")
```



Ajuster le comportement de Stan

En fittant des modèles un peu compliqués, il se peut que vous obteniez des messages d'avertissement du genre "`There were x divergent transitions after warmup`". Dans cette situation, on peut ajuster le comportement de `Stan` directement dans un appel de la fonction `brm()` en utilisant l'argument `control`.

```
mod9 <- brm(
    formula = presence | trials(total) ~ 1 + reminder + (1 + reminder | researcher),
    family = binomial(link = "logit"),
    data = absence_data,
    warmup = 2000, iter = 1e4,
    chains = 4, cores = 4,
    control = list(adapt_delta = 0.95) # adjusting the delta step size
    )
```

On peut par exemple augmenter le pas de l'algorithme, via `adapt_delta` (par défaut fixé à 0.8), ce qui ralentira probablement l'échantillonnage mais améliorera la validité des échantillons obtenus. Plus généralement, soyez attentifs aux messages d'erreur et d'avertissement générés par `brms`.

Tutoriels

Une liste d'articles de blog sur `brms` : https://paul-buerkner.github.io/blog/brms-blogposts/.

L'article d'introduction du paquet `brms` (Bürkner, 2017) et la version "advanced" (Bürkner, 2018).

Un tutoriel sur les modèles de régression logistique ordinale (Bürkner & Vuorre, 2019).

Notre article tutoriel d'introduction aux modèles multi-niveaux avec `brms` (Nalborczyk et al., 2019).

Application des modèles généralisés additifs multi-niveaux (GAMMs) aux séries temporelles (e.g., M/EEG, iEEG, pupillometry, finger- or mouse-tracking) (Nalborczyk & Bürkner, 2025), voir le package `neurogam` (interface à `brms`): https://lnalborczyk.github.io/neurogam/.

Bayesian workflow (Gelman et al., 2020)

Bayesian workflow*

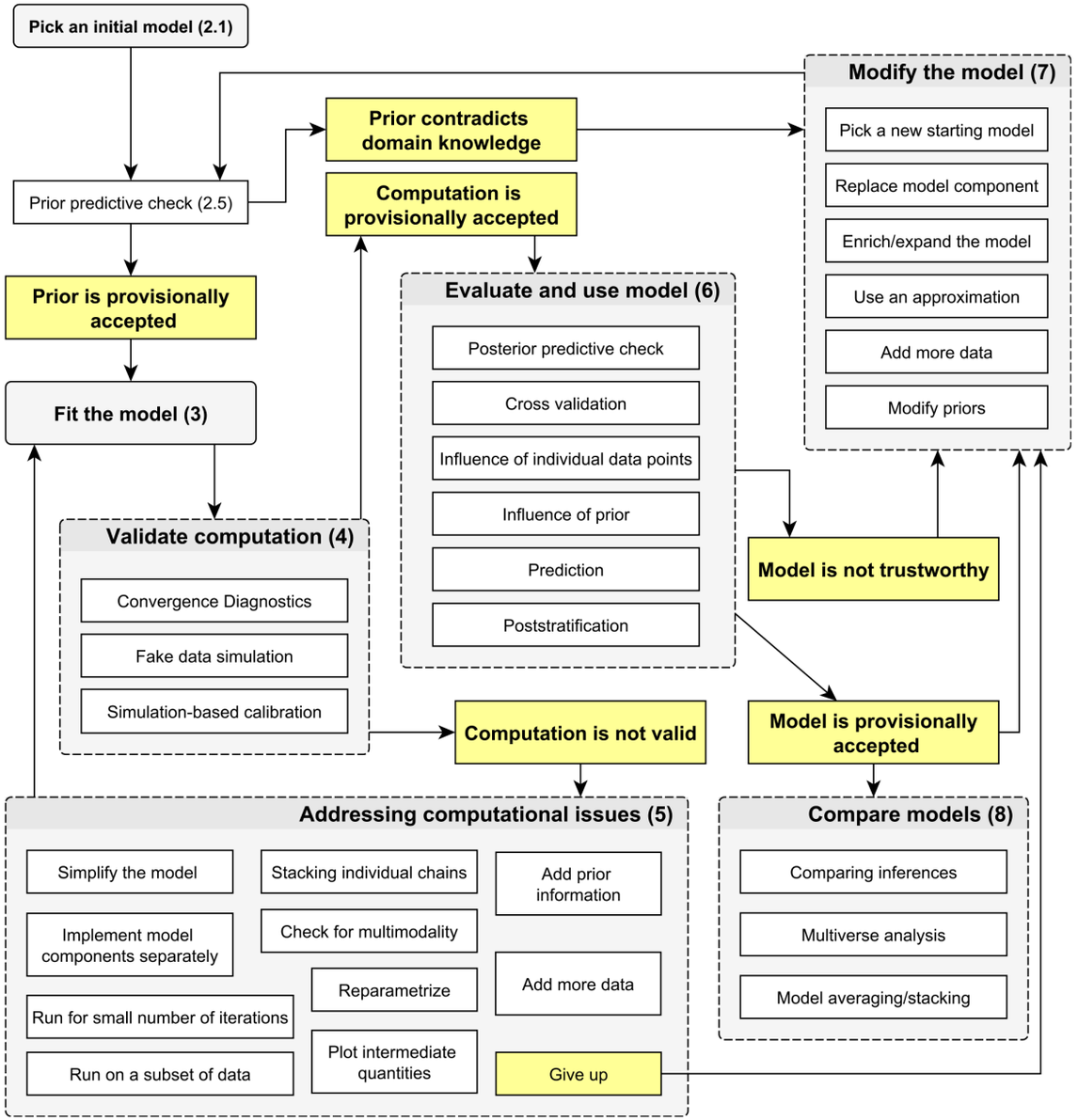
Andrew Gelman† Aki Vehtari‡ Daniel Simpson§ Charles C. Margossian†
Bob Carpenter¶ Yuling Yao† Lauren Kennedy‖ Jonah Gabry†
Paul-Christian Bürkner** Martin Modrák††

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

La statistique bayésienne est une approche générale de l'estimation de paramètres. Cette approche utilise la théorie des probabilités pour quantifier l'incertitude vis à vis de la valeur des paramètres de modèles statistiques.

Ces modèles sont composés de différents blocs (e.g., fonction de vraisemblance, priors, modèle linéaire ou non-linéaire) qui sont modifiables à souhait. Ce qu'on appelle classiquement "conditions d'application" sont simplement les conséquences des choix de modélisation réalisés par l'utilisateur. Autrement dit, c'est l'utilisateur qui choisit (et ne subit pas) les conditions d'application.

Nous avons vu que le modèle de régression linéaire est un modèle très flexible qui permet de décrire, via la modification de la fonction de vraisemblance et via l'introduction de fonctions de lien, des relations complexes (e.g., non-linéaires) entre variable prédite et variables prédictrices. Ces modèles peuvent gagner en précision par la prise en compte de la variabilité et des structures présentes dans les données (cf. modèles multi-niveaux).

Conclusions

Le paquet `brms` est un véritable couteau suisse de l'analyse statistique bayésienne en `R`. Il permet de fitter presque n'importe quel type de modèle de régression. Cela comprend tous les modèles que nous avons vu en cours, mais également bien d'autres. Entre autres, des modèles multivariés (i.e., avec plusieurs variables à prédire), des modèles "distributionnels" (e.g., pour prédire des différence de variance), des modèles additifs, des procesus Gaussiens (Gaussian processes), des modèles issus de la théorie de détection du signal, des modèles de mélange (mixture models), des modèles de diffusion, des modèles non-linéaires...

N'hésitez pas à me contacter pour plus d'informations sur ces modèles ou si vous avez des questions par rapport à vos propres données. Vous pouvez aussi contacter le créateur du paquet `brms`, très actif en ligne (voir son site). Voir aussi le forum Stan.

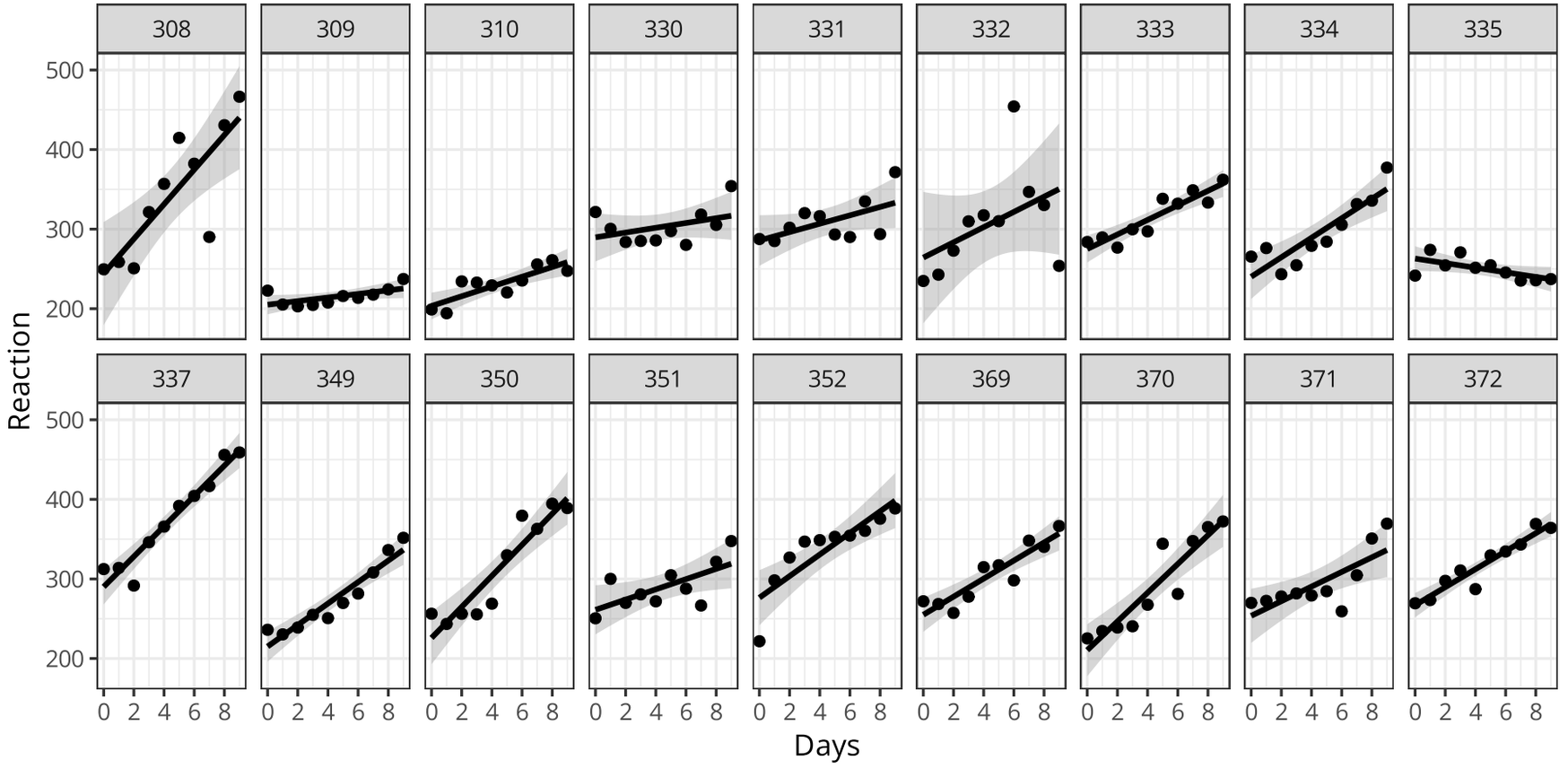
Travaux pratiques - sleepstudy

```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
```

Travaux pratiques - 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) )
```



Travaux pratiques - sleepstudy

À vous de construire les modèles mathématiques et les modèles `brms` correspondant aux modèles suivants :

- Modèle avec seulement l'effet fixe de `Days`.
- Modèle avec l'effet fixe de `Days` + un effet aléatoire de `Subject` (varying intercept).
- Modèle avec l'effet fixe de `Days` + un effet aléatoire de `Subject`. (varying intercept) + un effet aléatoire de `Days` (varying slope).

Comparez ensuite ces modèles en utilisant les outils discutés aux cours précédents (e.g., WAIC) et concluez.

Proposition de solution

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

anova(fmod1, fmod2)
```

```
Data: sleepstudy
Models:
fmod1: Reaction ~ Days + (1 | Subject)
fmod2: Reaction ~ Days + (1 + Days | Subject)
      npar    AIC    BIC  logLik -2*log(L)  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
```

Proposition de solution

```
mod10 <- 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,
  chains = 4, cores = 4
  )
```

```
posterior_summary(mod10)
```

```
             Estimate Est.Error        Q2.5      Q97.5
b_Intercept  252.00386 6.5463872  239.155656  264.73560
b_Days        10.30080 1.2273023    7.895806   12.69874
sigma         47.79162 2.5566260   43.121417   53.16565
Intercept    298.35745 3.5275687  291.411834  305.27840
lprior       -15.69108 0.1655589  -16.034541  -15.38824
lp__        -963.46829 1.2346384 -966.689450 -962.07563
```

Proposition de solution

```
mod11 <- 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,
  chains = 4, cores = 4
  )
```

```
posterior_summary(mod11, pars = c("^b", "sigma") )
```

```
             Estimate  Est.Error       Q2.5    Q97.5
b_Intercept 251.12022 10.0378272 231.108730 270.7911
b_Days       10.39035  0.8006148   8.834878  11.9326
sigma        31.06778  1.7415064  27.834066  34.6227
```

Proposition de solution

```r
mod12 <- 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,
  chains = 4, cores = 4
  )
```

```r
posterior_summary(mod12, pars = c("^b", "sigma") )
```

```
             Estimate Est.Error       Q2.5     Q97.5
b_Intercept 251.08544  6.945013 237.325794 265.08006
b_Days       10.10976  1.639218   6.859139  13.28473
sigma        25.86807  1.552198  23.089295  29.12345
```

Proposition de solution

```r
# calcul du WAIC et ajout du WAIC à chaque modèle
mod10 <- add_criterion(mod6, "waic")
mod11 <- add_criterion(mod7, "waic")
mod12 <- add_criterion(mod8, "waic")

# comparaison des WAIC de chaque modèle
w <- loo_compare(mod10, mod11, mod12, criterion = "waic")
print(w, simplify = FALSE)
```

```
      elpd_diff se_diff elpd_waic se_elpd_waic p_waic se_p_waic waic  se_waic
mod12   0.0       0.0   -59.3       1.6         10.3    0.6     118.6   3.2
mod11  -0.4       1.5   -59.7       2.4         10.7    1.4     119.5   4.8
mod10  -0.7       1.3   -59.9       2.1         10.8    1.3     119.9   4.2
```

```r
# calcul du poids relatif de chaque modèle
model_weights(mod10, mod11, mod12, weights = "waic")
```

```
    mod10     mod11     mod12
0.2403211 0.2960379 0.4636410
```

Travaux pratiques

Ce jeu de données recense des données concernant 2000 élèves dans 100 écoles différentes. L'outcome principal est la popularité de l'élève, évaluée sur une échelle de 1 à 10, et estimée en utilisant une procédure sociométrique (i.e., on a demandé aux élèves de se noter mutuellement). Ces élèves étaient également notés par leurs professeurs (colonne `teachpop`), sur une échelle de 1 à 7. On dispose comme prédicteurs du genre de l'élève (boy = 0, girl = 1) et de l'expérience du professeur (`texp`, en années).

```r
d <- open_data(popular)
head(d, 10)
```

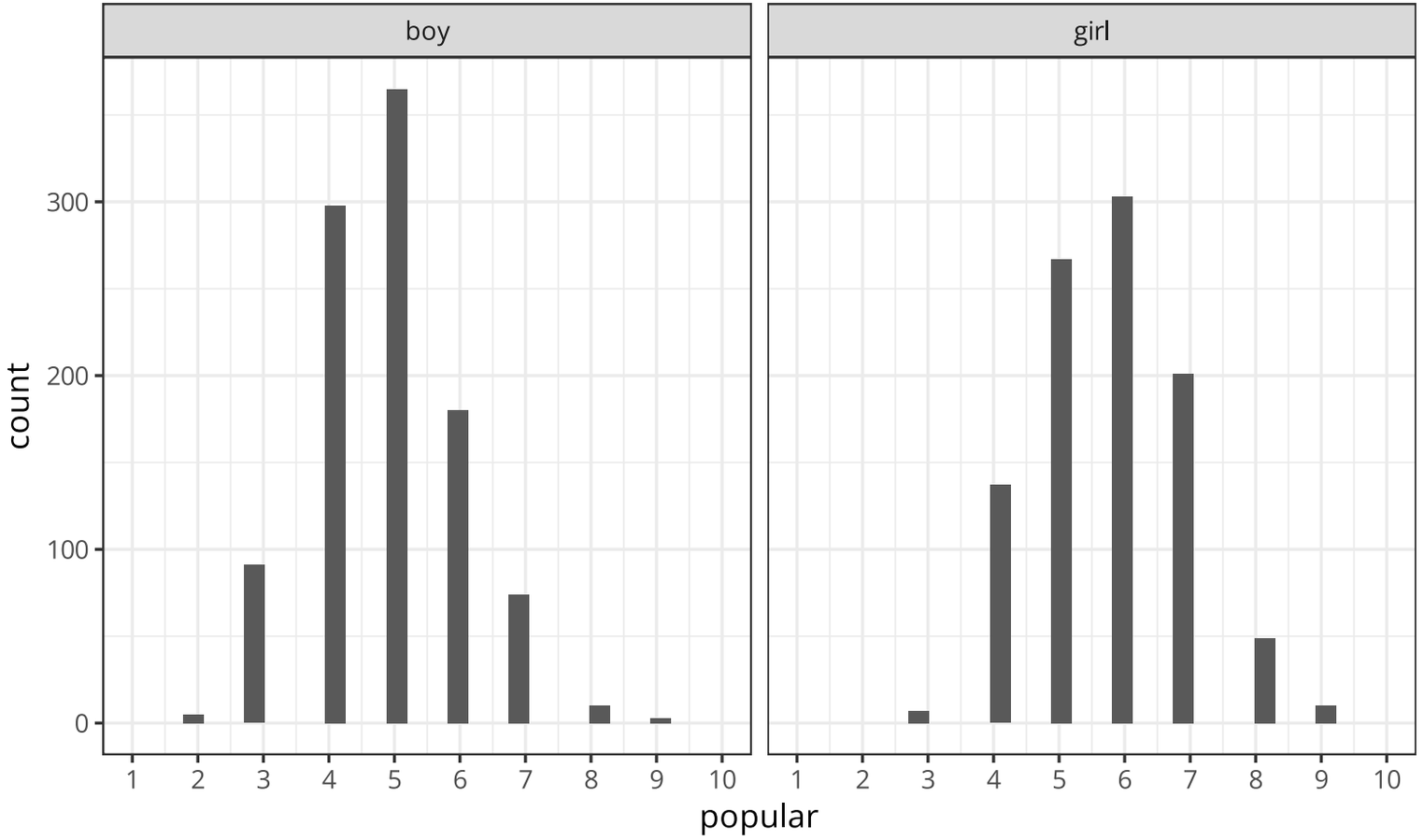
```
   pupil school popular  sex texp teachpop
1      1      1       8 girl   24        7
2      2      1       7  boy   24        7
3      3      1       7 girl   24        6
4      4      1       9 girl   24        6
5      5      1       8 girl   24        7
6      6      1       7  boy   24        7
7      7      1       7  boy   24        7
8      8      1       7  boy   24        7
9      9      1       7  boy   24        7
10    10      1       8  boy   24        6
```

Travaux pratiques

À vous d'explorer ce jeu de données, de fitter quelques modèles (avec `brms`) pour essayer de comprendre quels sont les facteurs qui expliquent (permettent de prédire) la popularité d'un élève...

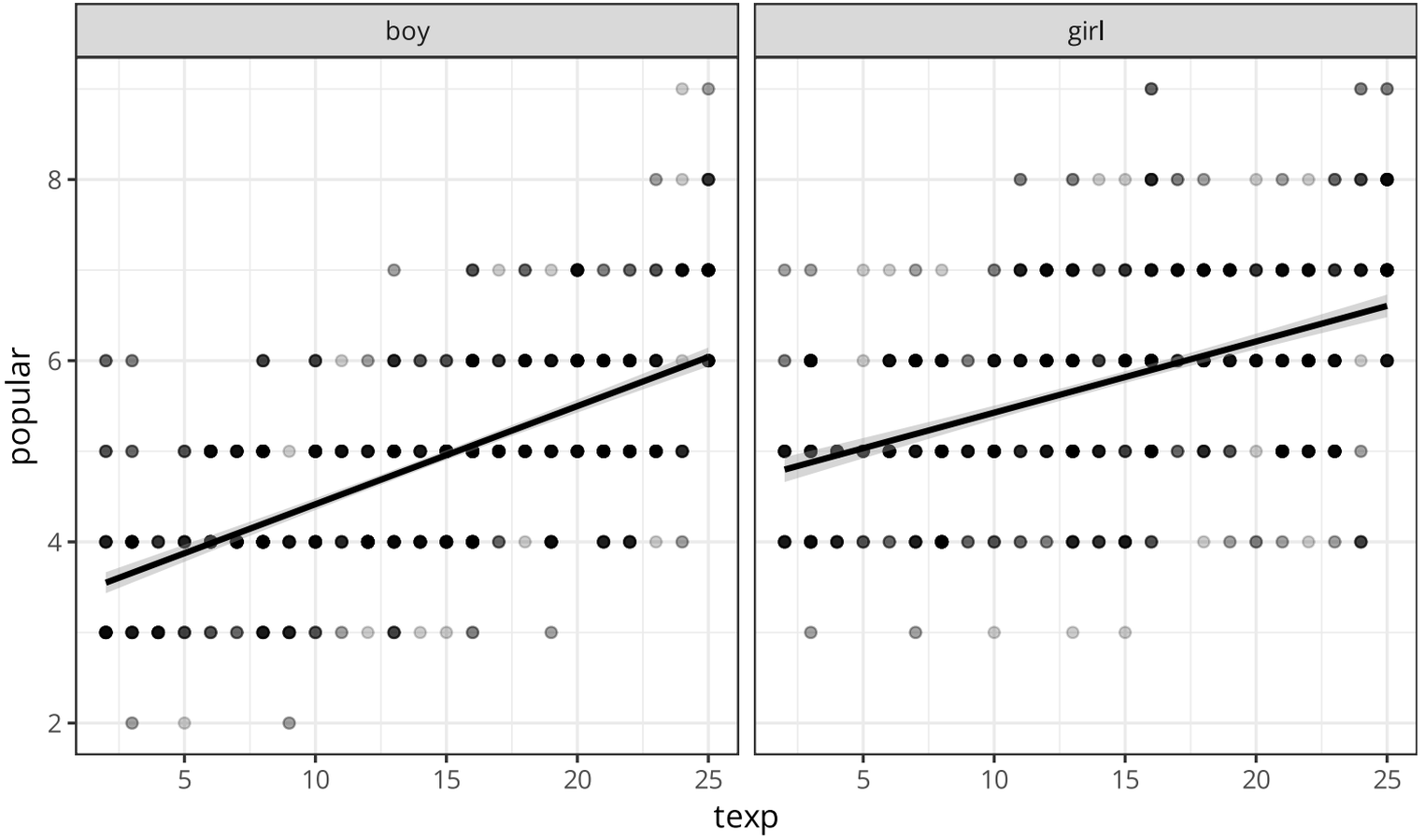
Proposition de solution - exploration graphique

```
d %>%
    ggplot(aes(x = popular) ) +
    geom_histogram() +
    facet_wrap(~sex) +
    scale_x_continuous(breaks = 1:10, limits = c(1, 10) )
```



Proposition de solution - exploration graphique

```
d %>%
    ggplot(aes(x = texp, y = popular) ) +
    geom_point(alpha = 0.2) +
    geom_smooth(method = "lm", colour = "black") +
    facet_wrap(~sex)
```



Proposition de solution

```r
d <- d %>%
    mutate(
        # using a sum contrast for gender
        sex = ifelse(sex == "boy", -0.5, 0.5),
        # centering and standardising teacher experience
        texp = scale(texp) %>% as.numeric
        )

prior13 <- c(
    prior(normal(5, 2.5), class = Intercept),
    prior(cauchy(0, 10), class = sd),
    prior(cauchy(0, 10), class = sigma)
    )

mod13 <- brm(
    formula = popular ~ 1 + (1 | school),
    data = d,
    prior = prior13,
    save_all_pars = TRUE,
    warmup = 2000, iter = 1e4, cores = 4
    )
```

Proposition de solution

```r
prior14 <- c(
    prior(normal(0, 1), class = Intercept),
    prior(normal(0, 1), class = b),
    prior(cauchy(0, 1), class = sd),
    prior(cauchy(0, 10), class = sigma)
    )

mod14 <- brm(
    formula = popular ~ 1 + texp + (1 | school),
    data = d,
    prior = prior14,
    save_all_pars = TRUE,
    warmup = 2000, iter = 1e4, cores = 4
    )
```

Proposition de solution

```
prior15 <- c(
    prior(normal(0, 1), class = Intercept),
    prior(normal(0, 1), class = b),
    prior(cauchy(0, 1), class = sd),
    prior(cauchy(0, 10), class = sigma),
    prior(lkj(2), class = cor)
    )

mod15 <- brm(
    formula = popular ~ 1 + sex + texp + (1 + sex | school),
    data = d,
    prior = prior15,
    save_all_pars = TRUE,
    warmup = 2000, iter = 1e4, cores = 4
    )
```

Proposition de solution

```r
mod16 <- brm(
    formula = popular ~ 1 + sex + texp + sex:texp + (1 + sex | school),
    data = d,
    prior = prior13,
    save_all_pars = TRUE,
    warmup = 2000, iter = 1e4, cores = 4
    )
```

```r
# calcul du WAIC et ajout du WAIC à chaque modèle
mod13 <- add_criterion(mod13, "waic")
mod14 <- add_criterion(mod14, "waic")
mod15 <- add_criterion(mod15, "waic")
mod16 <- add_criterion(mod16, "waic")
```

Proposition de solution

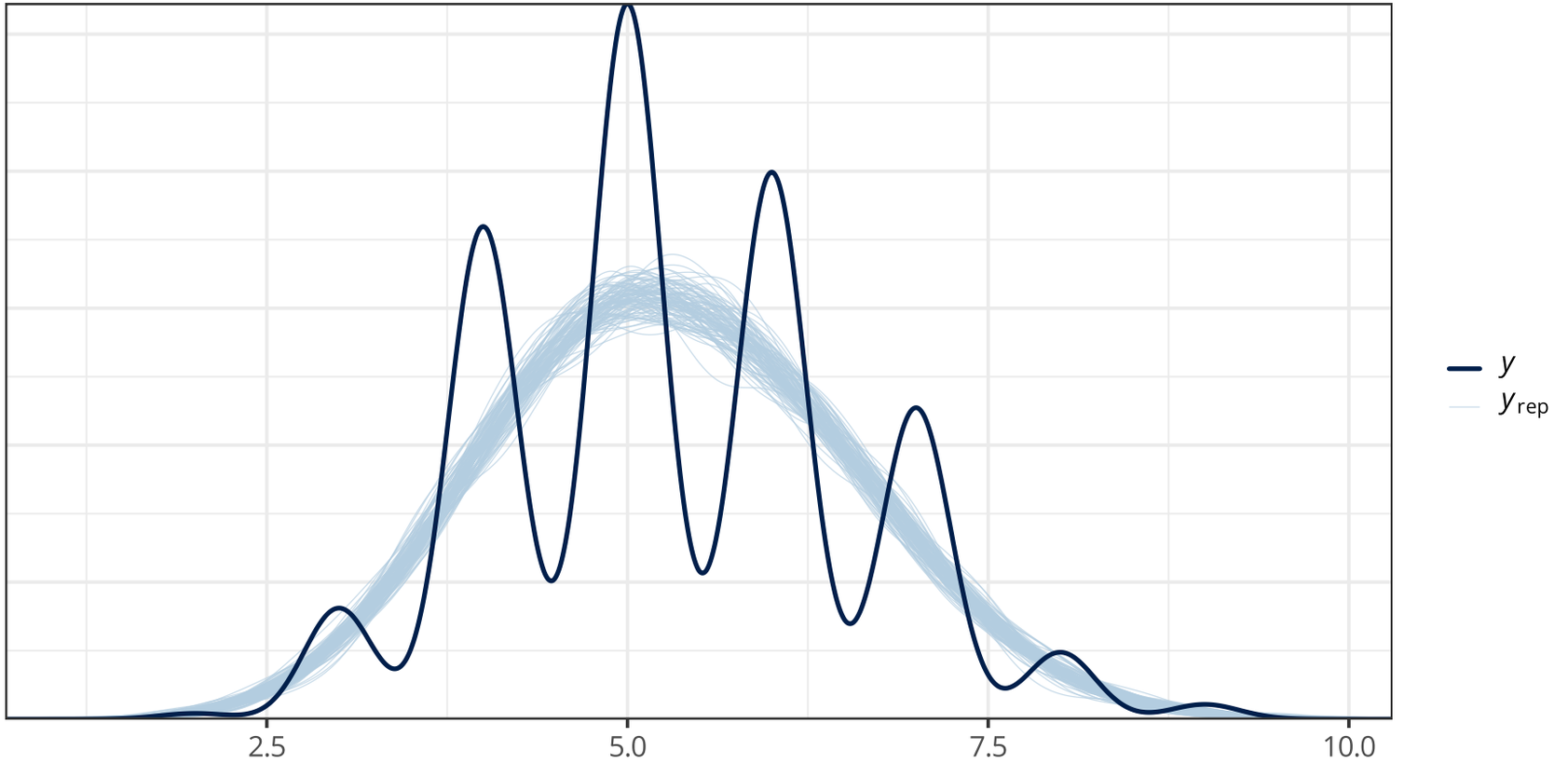
```r
# comparaison des WAIC de chaque modèle
model_comparison_table <- loo_compare(mod13, mod14, mod15, mod16, criterion = "waic") %>%
  data.frame() %>%
  rownames_to_column(var = "model")

weights <- data.frame(weight = model_weights(mod13, mod14, mod15, mod16, weights = "waic") ) %>%
  round(digits = 3) %>%
  rownames_to_column(var = "model")

left_join(model_comparison_table, weights, by = "model")
```

```
  model   elpd_diff   se_diff elpd_waic se_elpd_waic    p_waic se_p_waic
1 mod16    0.000000  0.000000 -1990.304     32.53308 161.05146  5.166530
2 mod15   -2.008898  2.065615 -1992.313     32.69868 164.00947  5.280188
3 mod14 -448.375590 25.860874 -2438.679     30.21622  92.88676  2.784893
4 mod13 -449.594992 25.922045 -2439.899     30.26226  95.22490  2.858814
      waic  se_waic weight
1 3980.608 65.06616  0.882
2 3984.625 65.39735  0.118
3 4877.359 60.43244  0.000
4 4879.798 60.52451  0.000
```

Proposition de solution

Les prédictions du modèle ne coïncident pas exactement avec les données car ces dernières sont discrètes. Les élèves étaient notés sur une échelle discrète allant de 1 à 10 (un élève ne pouvait pas avoir une note de 3.456). Ce type de données peut-être approximée par une distribution normale (comme nous l'avons fait) mais ce choix n'est pas optimal en termes de prédiction...

```
1  pp_check(object = mod16, ndraws = 1e2)
```



Proposition de solution

On pourrait choisir un modèle qui se rapproche du processus de génération des données. C'est le cas du modèle de régression logistique ordinale. Ce modèle est une sorte de généralisation à plus de 2 catégories du modèle de régression logistique vu au Cours n°06 (voir ce blogpost pour plus de détails, ou le chapitre 11 de Statistical Rethinking), sauf que les catégories sont ordonnées.

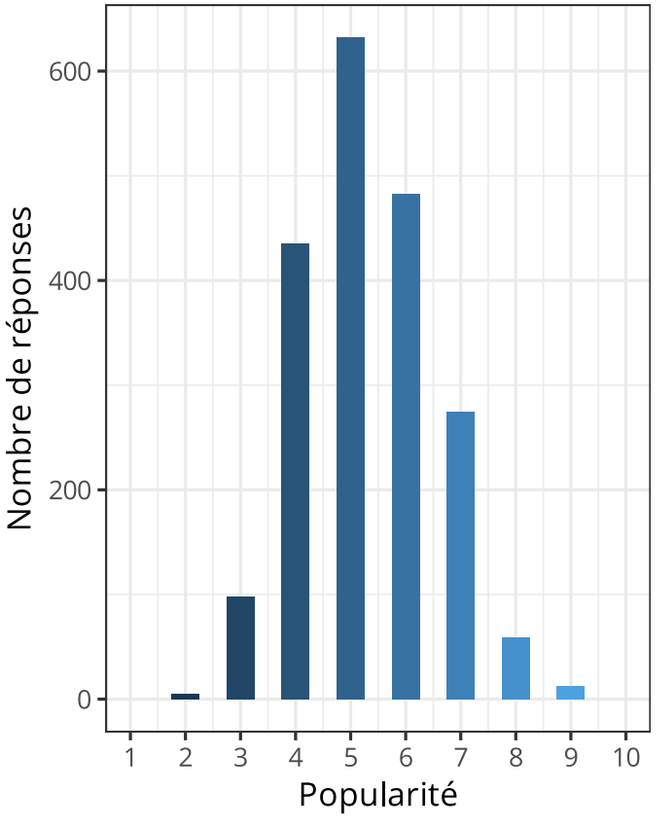
$$
\begin{aligned}
\text{pop}_i &\sim \text{Categorical}(\mathbf{p}) \\
\text{logit}(p_k) &= \alpha_k \\
\alpha_k &\sim \text{Normal}(0, 10)
\end{aligned}
$$

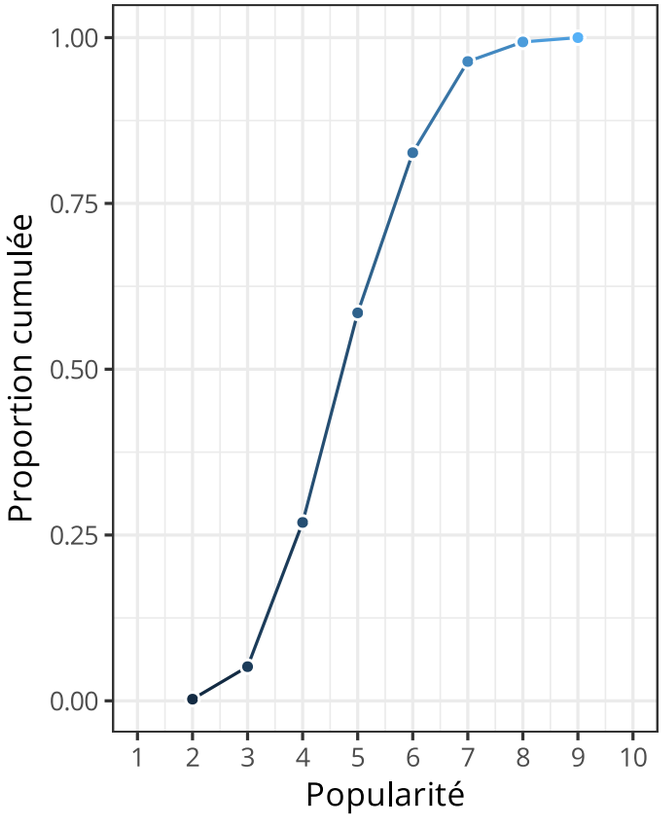
Où la distribution Categorical est une distribution discrète qui prend un vecteur de probabilités $\mathbf{p} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$ qui correspondent aux probabilités cumulées de chaque réponse (entre 1 et 10, 10 ayant une probabilité cumulée de 1).

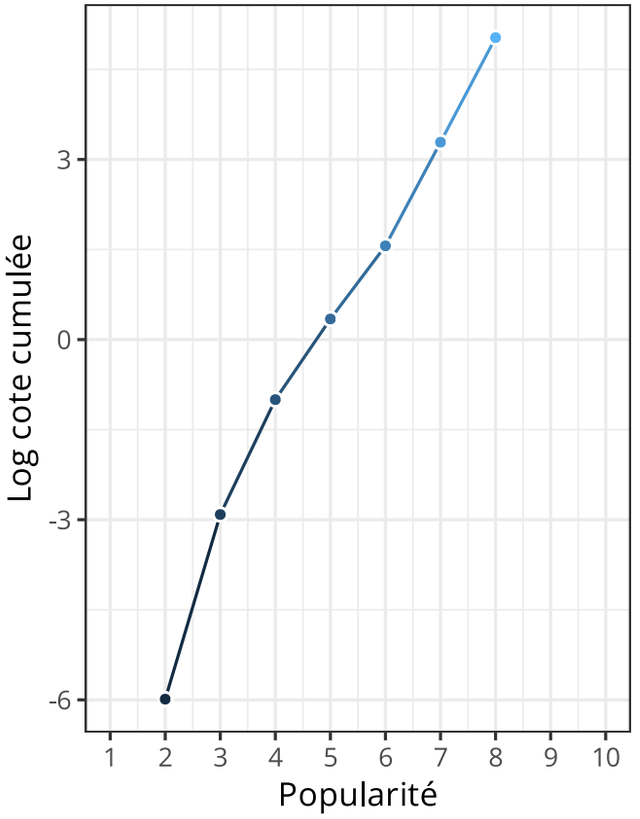
Proposition de solution

On définit une série de $N-1$ intercepts $\mathbf{p} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$ sur le logarithme de la cote cumulée (log-cumulative-odds).

$$\text{logit}(p_k) = \log \frac{\Pr(y_i \leq k)}{1 - \Pr(y_i \leq k)} = \alpha_k$$



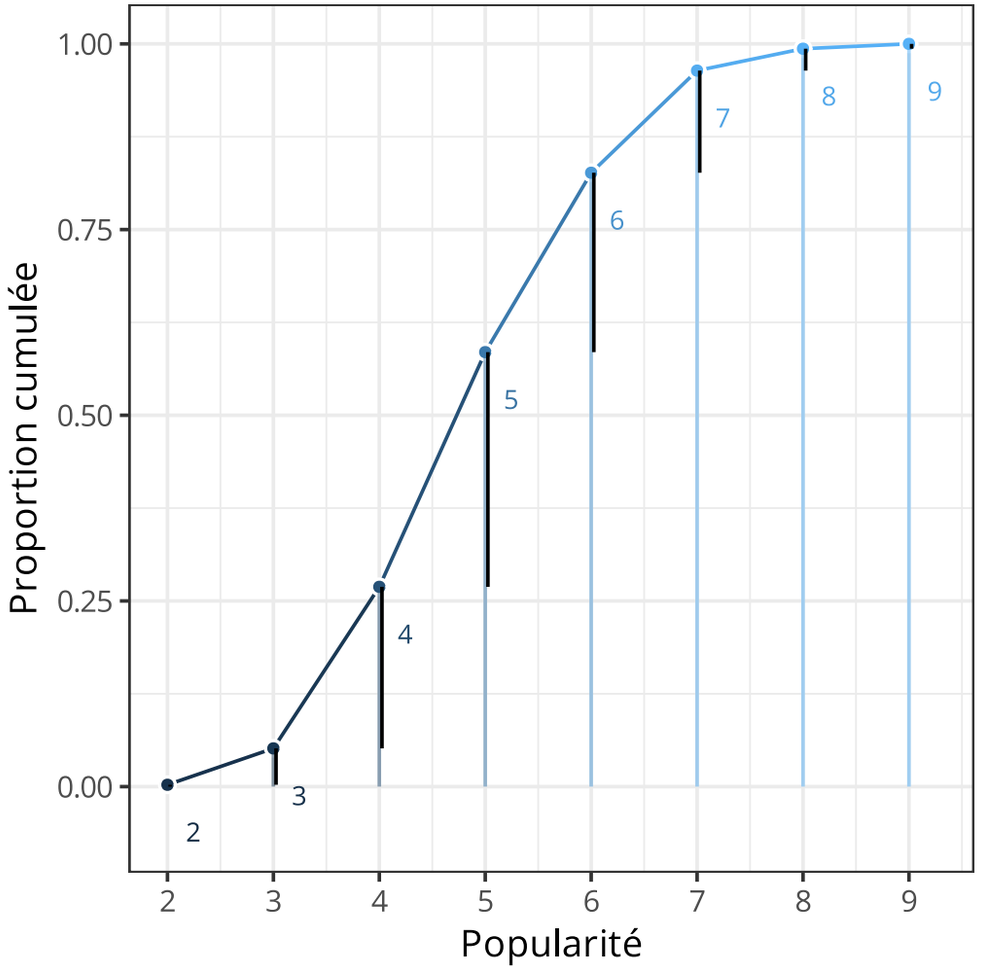




Proposition de solution

La vraisemblance de l'observation k (e.g., `pop = 3`) est donnée par soustraction des proportions cumulées. Cette vraisemblance est représentée par les barres verticales sur le graphique ci-dessous.

$$p_k = \Pr(y_i = k) = \Pr(y_i \leq k) - \Pr(y_i \leq k - 1)$$



Proposition de solution

NB : Ce modèle peut prendre plusieurs heures selon votre système...

```r
mod17 <- brm(
    popular ~ 1 + sex + texp + sex:texp + (1 | school),
    data = d,
    prior = prior14,
    warmup = 1000, iter = 5000,
    chains = 4, cores = 4,
    threads = threading(threads = 2),
    file = "models/mod17", backend = "cmdstanr"
    )
```

```r
prior18 <- c(
    brms::prior(normal(0, 10), class = Intercept),
    brms::prior(normal(0, 10), class = b),
    brms::prior(cauchy(0, 10), class = sd)
    )

mod18 <- brm(
    popular ~ 1 + sex + texp + sex:texp + (1 | school),
    data = d,
    family = cumulative(link = "logit"),
    prior = prior18,
    chains = 4, cores = 4,
    control = list(adapt_delta = 0.99, max_treedepth = 15),
    threads = threading(threads = 2),
    file = "models/mod18", backend = "cmdstanr"
    )
```

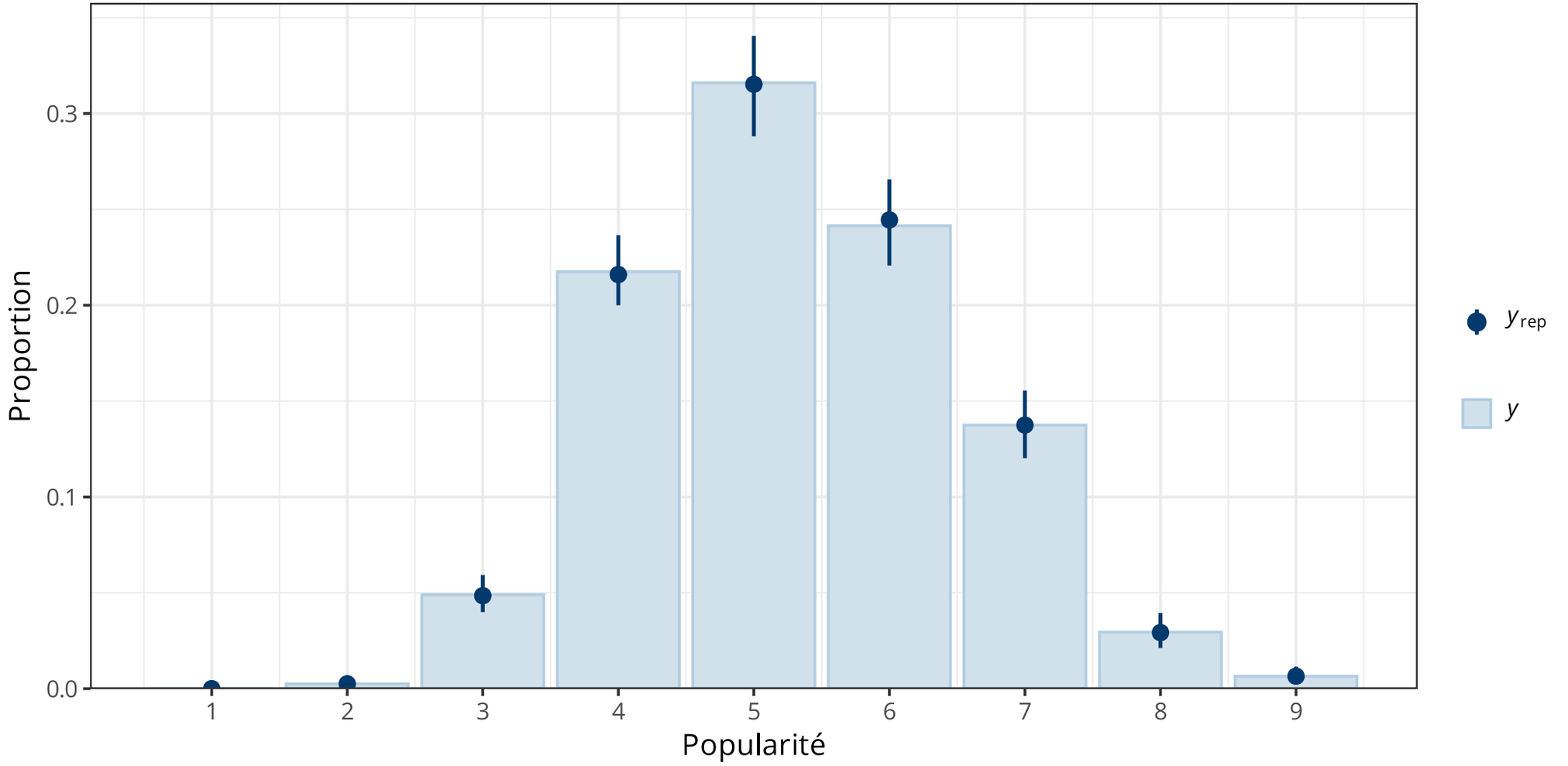
Proposition de solution

```
1 waic(mod17, mod18, compare = FALSE)
```

```
Output of model 'mod17':

Computed from 16000 by 2000 log-likelihood matrix.

          Estimate   SE
elpd_waic  -2086.4 31.3
p_waic        96.6  3.0
waic        4172.8 62.6

8 (0.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.

Output of model 'mod18':

Computed from 4000 by 2000 log-likelihood matrix.

          Estimate   SE
elpd_waic  -2074.2 32.4
p_waic       105.9  2.6
waic        4148.4 64.8

4 (0.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

Proposition de solution

```
1 pp_check(mod18, ndraws = 1e2, type = "bars", prob = 0.95, freq = FALSE) +
2   scale_x_continuous(breaks = 1:9) +
3   labs(x = "Popularité", y = "Proportion")
```



Références

Bürkner, P.-C. (2017). **brms** : An *R* Package for Bayesian Multilevel Models Using *Stan*. *Journal of Statistical Software*, *80*(1). https://doi.org/10.18637/jss.v080.i01

Bürkner, P.-C. (2018). Advanced Bayesian Multilevel Modeling with the R Package brms. *The R Journal*, *10*(1), 395–411. https://journal.r-project.org/archive/2018/RJ-2018-017/index.html

Bürkner, P.-C., & Vuorre, M. (2019). Ordinal Regression Models in Psychology: A Tutorial: *Advances in Methods and Practices in Psychological Science*. https://doi.org/10.1177/2515245918823199

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

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., & Gelman, A. (2019). Visualization in Bayesian workflow. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, *182*(2), 389–402. https://doi.org/10.1111/rssa.12378

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

Gronau, Q. F., Singmann, H., & Wagenmakers, E.-J. (2017). *Bridgesampling: An r package for estimating normalizing constants*. https://doi.org/10.48550/ARXIV.1710.08162

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

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

Nalborczyk, L., & Bürkner, P. (2025). *Precise temporal localisation of M/EEG effects with Bayesian generalised additive multilevel models*. https://doi.org/10.1101/2025.08.29.672336