

# Introduction à la modélisation statistique bayésienne

Un cours en R et Stan avec brms

Ladislas Nalborczyk (LPC, LNC, CNRS, 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

## **Cours n°09 : Modèles multi-niveaux généralisés**

Cours n°10 : Data Hackathon



# 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).

Le paquet **brms** permet également de fitter des modèles multivariés (plusieurs outcomes) en les combinant avec **c()**:

```
1 c(Reaction, Memory) ~ Days + (1 + Days | Subject)
```

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

```
1 c(Reaction, Memory) ~ Days + (1 + Days | Subject)
2 c(Reaction, Memory) ~ 1 + Days + (1 + Days | Subject)
```



# Rappels de syntaxe

Si l'on veut fitter un modèle sans intercept (why not), il faut le spécifier explicitement comme ci-dessous.

```
1 c(Reaction, Memory) ~ 0 + Days + (1 + Days | Subject)
```

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 variables (effets aléatoires).

```
1 c(Reaction, Memory) ~ Days + (1 | Subject)
2 c(Reaction, Memory) ~ Days + (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 variables (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 c(Reaction, Memory) ~ Days + (1 + Days || Subject)
```

Les modèles précédents postulaient une fonction de vraisemblance Gaussienne. Ce postulat peut être changé facilement en spécifiant la fonction de vraisemblance souhaitée via l'argument **family**.

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



# Mise en pratique - absentéisme expérimental

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](#)).

```

1 library(tidyverse)
2 library(imsb)
3
4 data <- open_data(absence_multilevel) %>%
5   mutate(reminder = ifelse(test == 1, yes = 0.5, -0.5) )
6 data %>% sample_frac() %>% head(10)

  reminder researcher presence absence total
1     -0.5           1      16     86    102
2      0.5           4      92      3    95
3     -0.5          10      23     58    81
4      0.5           6      82      6    88
5     -0.5           3      31     65    96
6      0.5           2     109      3   112
7     -0.5           6      34     54    88
8      0.5           7      64     16    80
9     -0.5           5      34     49    83
10     0.5           8      81     11    92

```



# Mise en pratique - absentéisme expérimental

$$y_i \sim \text{Binomial}(n_i, p_i)$$

$$\text{logit}(p_i) = \alpha_{\text{researcher}_{[i]}} + \beta_{\text{researcher}_{[i]}} \times \text{reminder}_i$$

$$\begin{bmatrix} \alpha_{\text{researcher}} \\ \beta_{\text{researcher}} \end{bmatrix} \sim \text{MVNormal} \left( \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \mathbf{S} \right)$$

$$\mathbf{S} = \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \mathbf{R} \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix}$$

$$\alpha \sim \text{Normal}(0, 1)$$

$$\beta \sim \text{Normal}(0, 1)$$

$$(\sigma_\alpha, \sigma_\beta) \sim \text{HalfCauchy}(0, 1)$$

$$\mathbf{R} \sim \text{LKJcorr}(2)$$

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

```
1 prior1 <- c(  
2     prior(normal(0, 1), class = Intercept),  
3     prior(normal(0, 1), class = b),  
4     prior(cauchy(0, 1), class = sd),  
5     prior(lkj(2), class = cor)  
6 )
```

```
1 mod1 <- brm(  
2     formula = presence | trials(total) ~ 1 + reminder + (1 + reminder | researcher),  
3     family = binomial(link = "logit"),  
4     prior = prior1,  
5     data = data,  
6     sample_prior = TRUE,  
7     warmup = 2000, iter = 10000,  
8     chains = 4, cores = parallel::detectCores(),  
9     control = list(adapt_delta = 0.95),  
10    backend = "cmdstanr"  
11 )
```

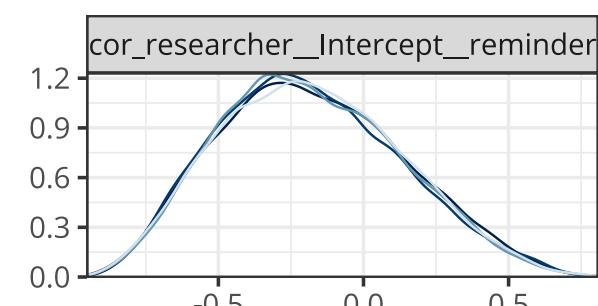
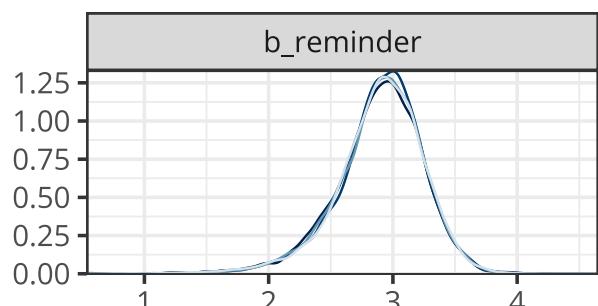
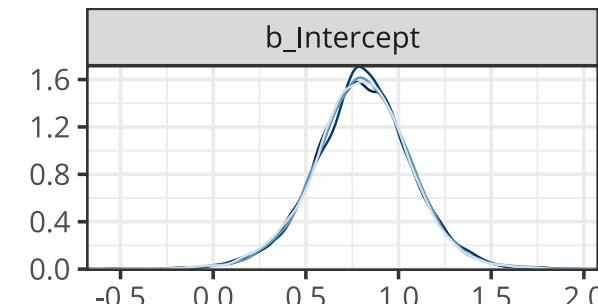


# Mise en pratique - absentéisme expérimental

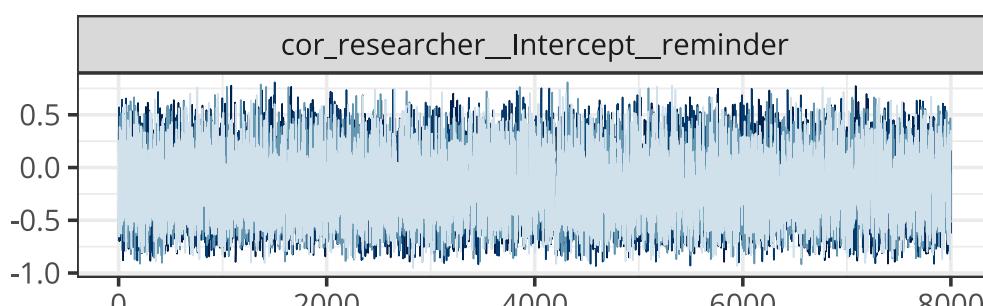
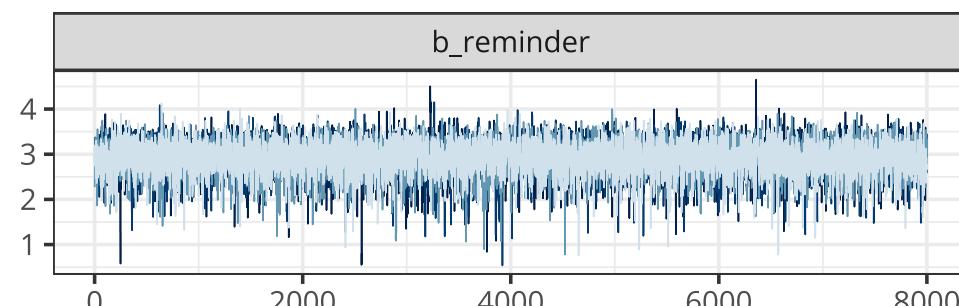
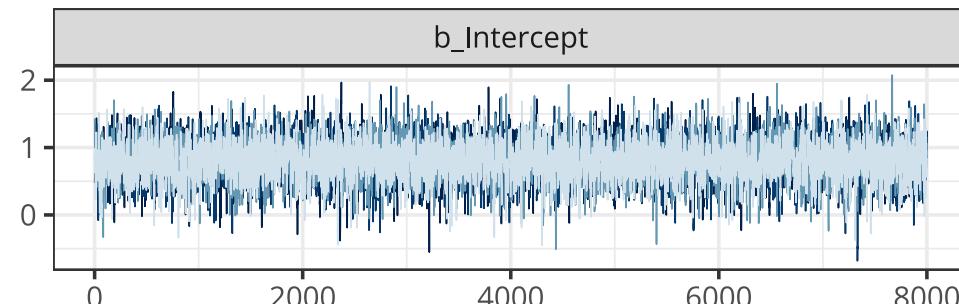
```

1 mod1 %>%
2   plot(
3     combo = c("dens_overlay", "trace"), pars = c("^b_",
4     "cor_"), widths = c(1, 1.5),
5     theme = theme_bw(base_size = 16, base_family = "Open Sans")
6   )

```



Chain  
— 1  
— 2  
— 3  
— 4



Chain  
— 1  
— 2  
— 3  
— 4



# Mise en pratique - absentéisme expérimental

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

```
1 posterior_summary(x = mod1, pars = c("^b_",
                                         "sd_"))
  Estimate   Est.Error    Q2.5    Q97.5
b_Intercept 0.8039263 0.2654262 0.2597796 1.323720
b_reminder  2.8958126 0.3533834 2.0886413 3.504831
sd_researcher_Intercept 0.7857415 0.2347242 0.4531115 1.350410
sd_researcher_reminder  0.9072398 0.3347502 0.4322093 1.741153
```

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  $p = \exp(\alpha)/(1 + \exp(\alpha))$ .

```
1 a <- fixef(mod1)[1] # on récupère la valeur de l'intercept
2 exp(a) / (1 + exp(a)) # équivalent à plogis(a)
[1] 0.6908137
```



# 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).

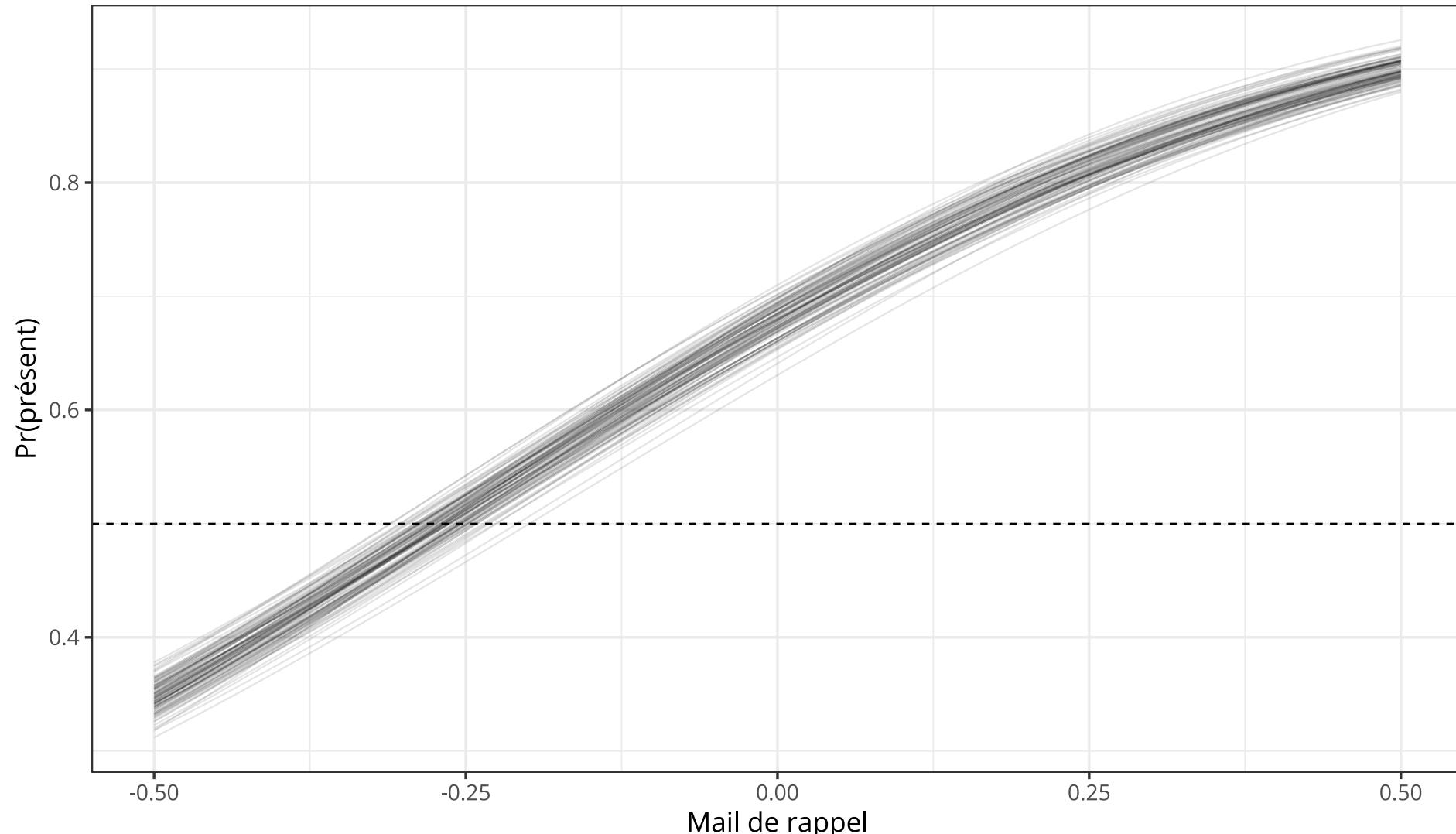
```
1 fixef(mod1)[2, c(1, 3, 4)] %>% exp()  
  
Estimate      Q2.5      Q97.5  
18.098203   8.073937  33.275828
```

Envoyer un mail de rappel multiplie par environ 18 la cote (i.e., le rapport  $\frac{\text{Pr(présent)}}{\text{Pr(absent)}}$ ).



# 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”.

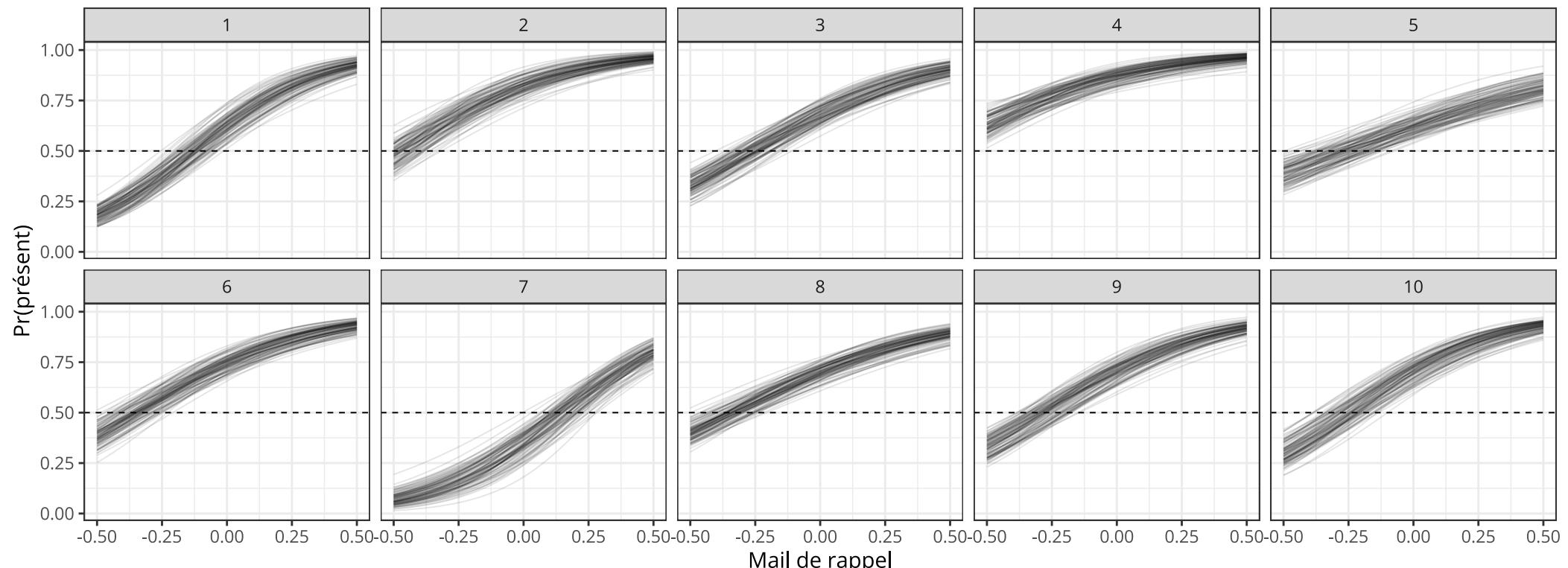


# Représenter les prédictions du modèle

```

1 data %>%
2   group_by(researcher, total) %>%
3   data_grid(reminder = seq_range(reminder, n = 1e2) ) %>%
4   add_fitted_samples(mod1, newdata = ., n = 100, scale = "linear") %>%
5   mutate(estimate = plogis(estimate) ) %>%
6   ggplot(aes(x = reminder, y = estimate, group = .iteration)) +
7   geom_hline(yintercept = 0.5, lty = 2) +
8   geom_line(aes(y = estimate, group = .iteration), size = 0.5, alpha = 0.1) +
9   facet_wrap(~researcher, nrow = 2) +
10  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.

```
1 (hyp1 <- hypothesis(x = mod1, 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     2.9      0.35    2.09      3.5          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.
```

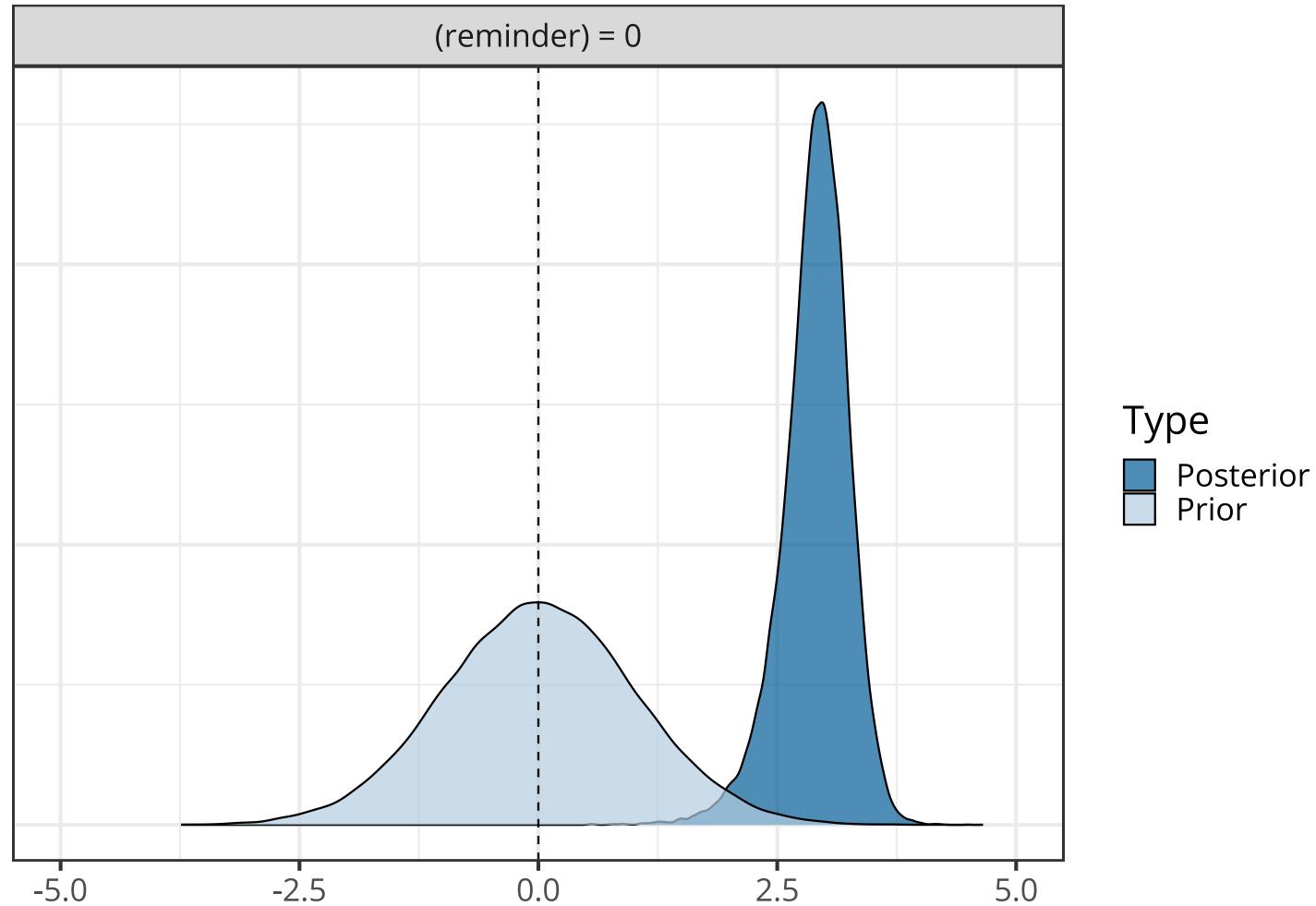
```
1 1 / hyp1$hypothesis$Evid.Ratio # BF10 = 1 / BF01 (and BF01 = 1 / BF10)
```

```
[1] 1.358937e+16
```



# Test d'hypothèse - 1

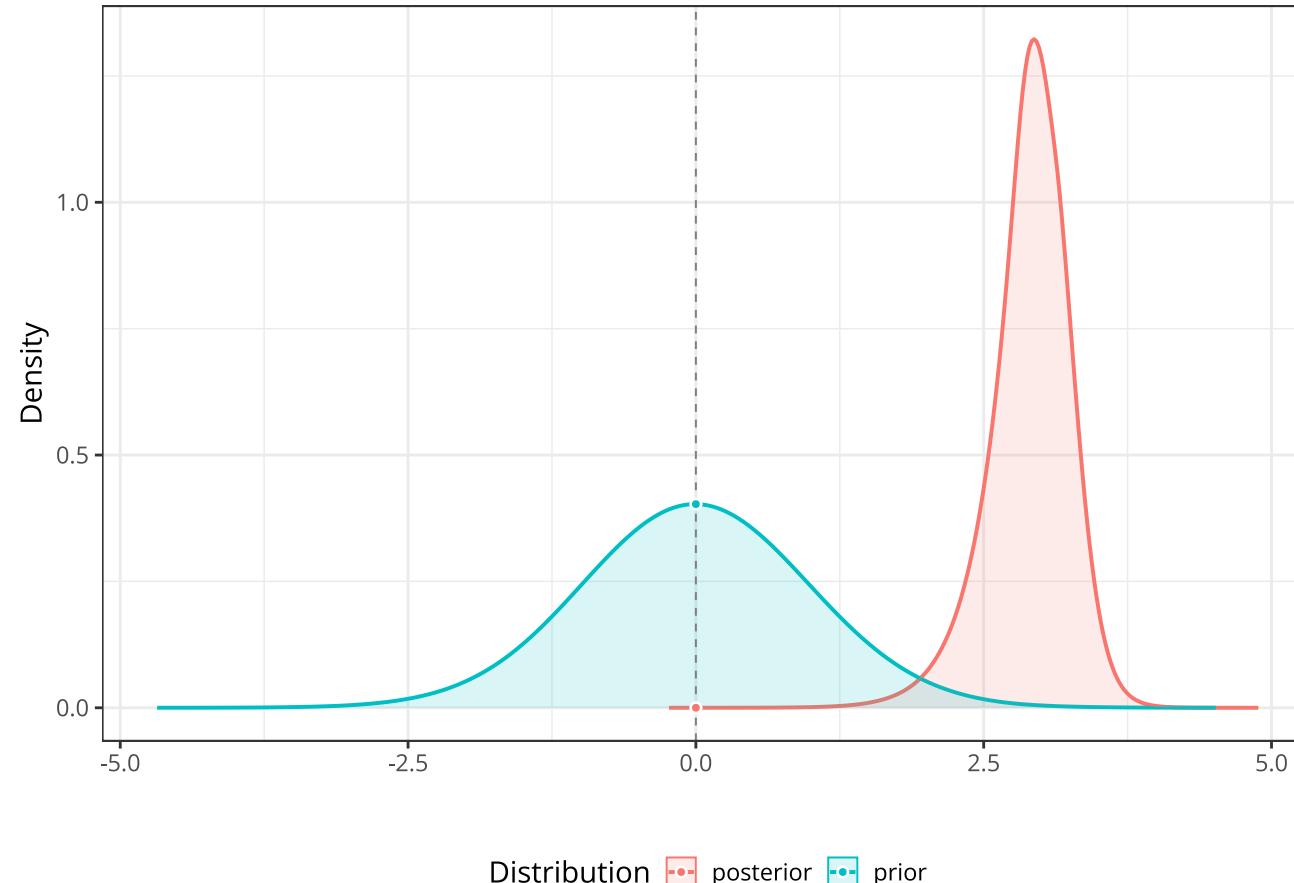
```
1 plot(hyp1, plot = FALSE, theme = theme_bw(base_size = 20, base_family = "Open Sans") )[[1]] +  
2   geom_vline(xintercept = 0, linetype = 2) +  
3   coord_cartesian(xlim = c(-5, 5))
```



# 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](https://easystats.github.io/bayestestR/articles/bayes_factors.html).

```
1 library(bayestestR)
2 bf <- bayesfactor_parameters(posterior = mod1, null = 0)
3 plot(bf)
```

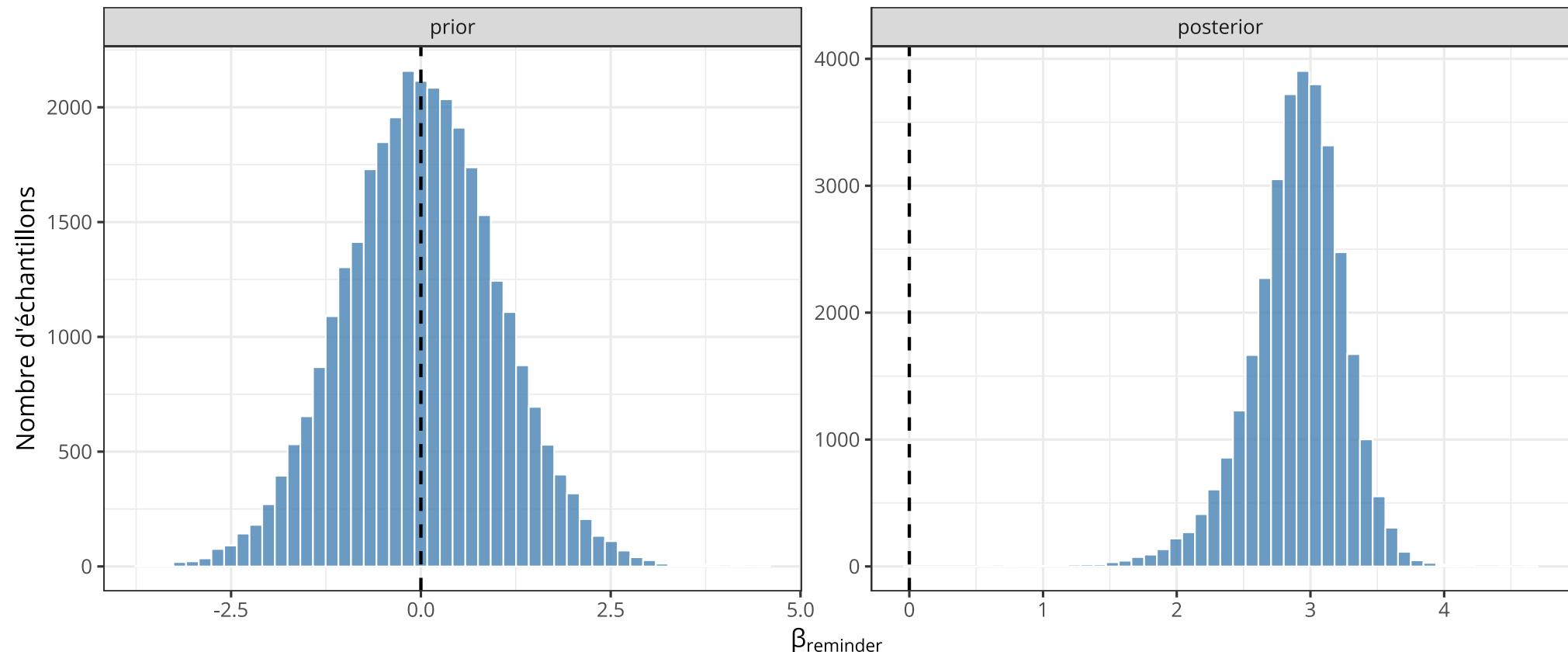


# Comparer le prior et le posterior

```

1 data.frame(prior = hyp1$prior_samples$H1, posterior = hyp1$samples$H1) %>%
2   gather(type, value) %>%
3   mutate(type = factor(type, levels = c("prior", "posterior")) ) %>%
4   ggplot(aes(x = value)) +
5   geom_histogram(bins = 50, alpha = 0.8, col = "white", fill = "steelblue") +
6   geom_vline(xintercept = 0, lty = 2, size = 1) +
7   facet_wrap(~type, scales = "free") +
8   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.

```

1 prior2 <- c(
2   prior(normal(0, 10), class = Intercept, coef = ""),
3   prior(cauchy(0, 10), class = sd),
4   prior(lkj(2), class = cor) )
5
6 mod2 <- brm(presence | trials(total) ~ 1 + reminder + (1 + reminder | researcher),
7   family = binomial(link = "logit"),
8   prior = prior1,
9   data = data,
10  # this line is important for bridgesampling
11  save_all_pars = TRUE,
12  warmup = 2000, iter = 1e4,
13  cores = parallel::detectCores(), backend = "cmdstanr",
14  control = list(adapt_delta = 0.95) )
15
16 mod3 <- brm(presence | trials(total) ~ 1 + (1 + reminder | researcher),
17   family = binomial(link = "logit"),
18   prior = prior2,
19   data = data,
20   save_all_pars = TRUE,
21   warmup = 2000, iter = 1e4,
22   cores = parallel::detectCores(), backend = "cmdstanr",
23   control = list(adapt_delta = 0.95) )

```



## 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](#)).

```
1 bayes_factor(mod2, mod3)
```

```
Estimated Bayes factor in favor of mod2 over mod3: 142403.46738
```



# 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(mod2, mod3, compare = FALSE)
```

Output of model 'mod2':

Computed from 32000 by 20 log-likelihood matrix

	Estimate	SE
elpd_waic	-59.8	2.1
p_waic	10.7	1.3
waic	119.7	4.2

15 (75.0%) p\_waic estimates greater than 0.4. We recommend trying loo instead.

Output of model 'mod3':

Computed from 32000 by 20 log-likelihood matrix

	Estimate	SE
elpd_waic	-59.0	1.6
p_waic	10.1	0.5
waic	118.0	3.3

19 (95.0%) 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  $\theta$ , on peut simuler des données à partir de la **posterior predictive distribution** :

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

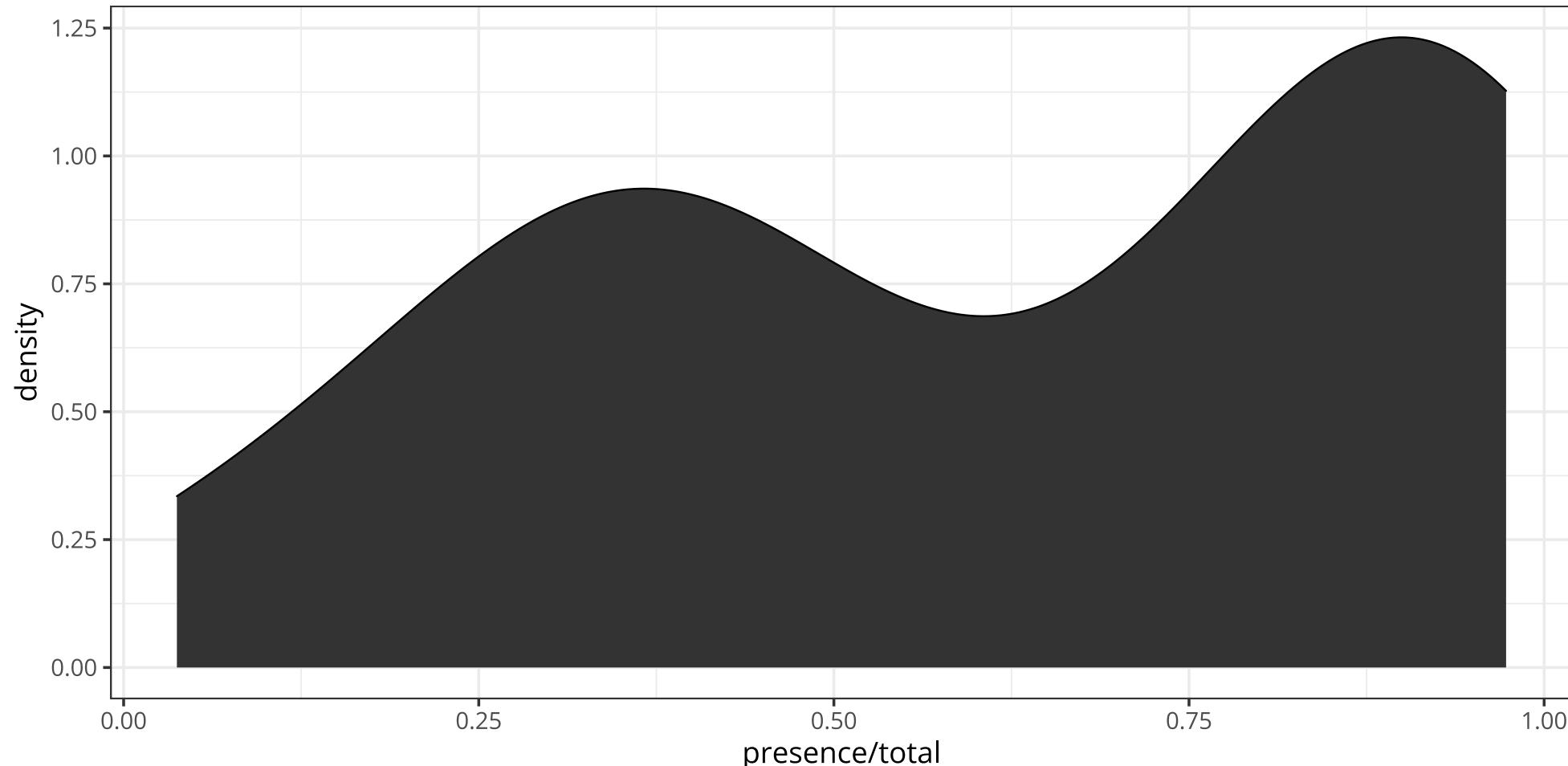
Si le modèle est un bon modèle, il devrait pouvoir générer des données qui ressemblent aux données qu'on a observées (e.g., [Gabry et al., 2019](#)).



# Posterior predictive checking

On représente ci-dessous la distribution de nos données.

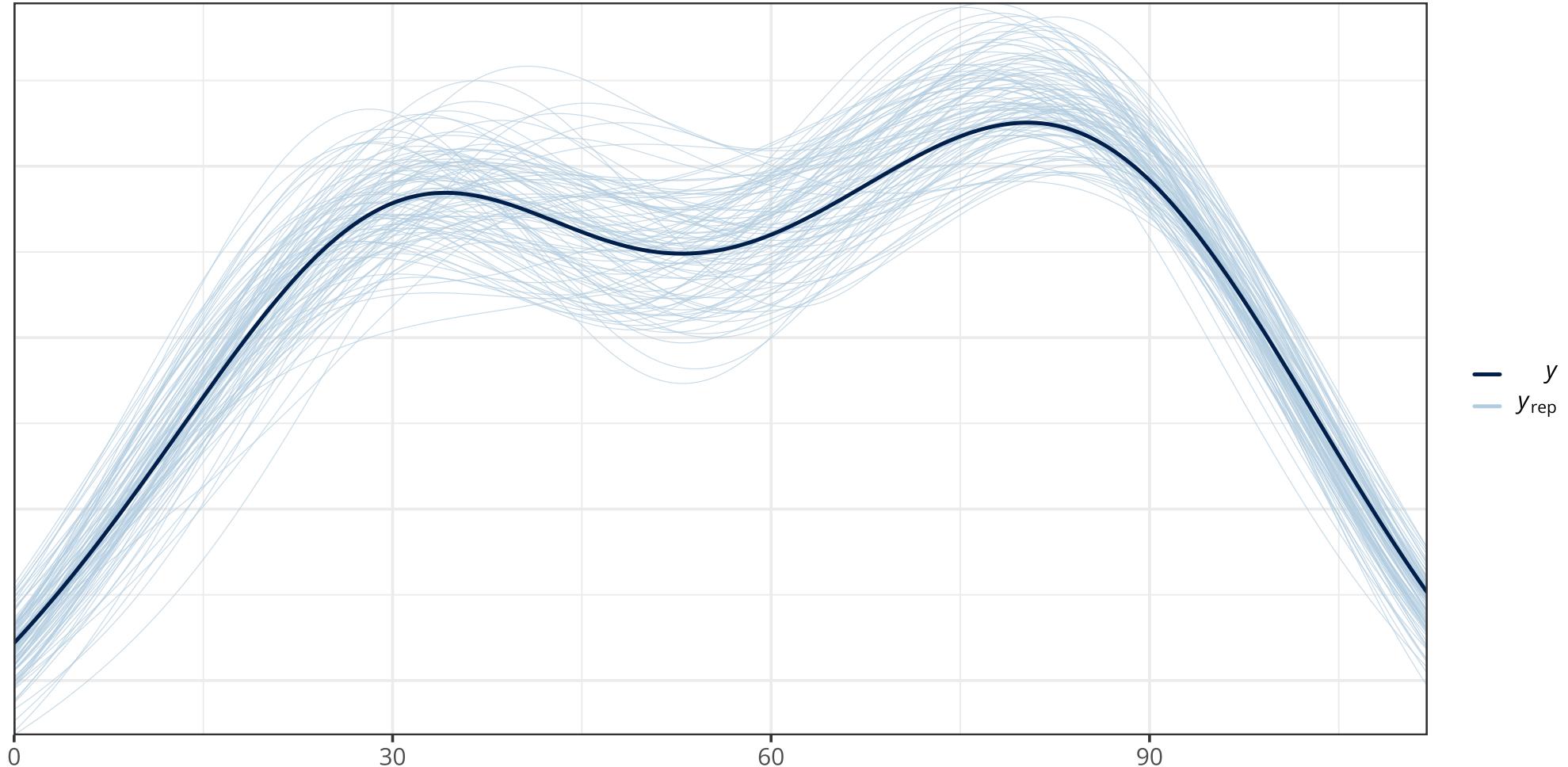
```
1 data %>%
2   ggplot(aes(x = presence / total)) +
3   geom_density(fill = "grey20")
```



# Posterior predictive checking

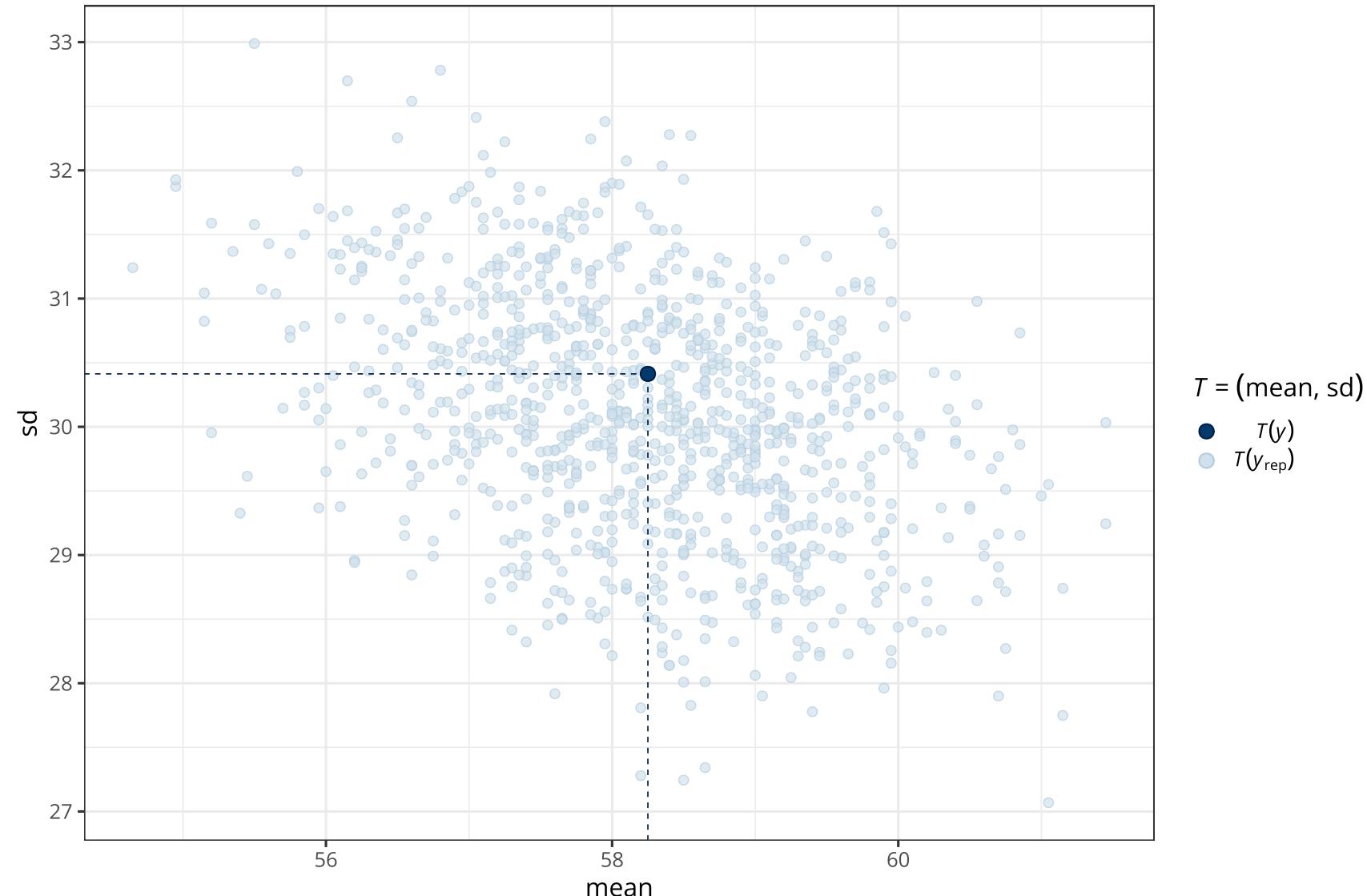
Cette procédure est implémentée dans **brms** via la méthode `pp_check()` qui permet de réaliser de nombreux checks. Par exemple, ci-dessous on compare les prédictions a posteriori ( $n = 100$ ) aux données observées.

```
1 pp_check(object = mod2, nsamples = 1e2)
```



# Posterior predictive checking

```
1 pp_check(object = mod2, nsamples = 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**.

```
1 mod2 <- brm(
2   presence | trials(total) ~ 1 + reminder + (1 + reminder | researcher),
3   family = binomial(link = "logit"),
4   prior = prior2,
5   data = data,
6   warmup = 2000, iter = 1e4,
7   cores = parallel::detectCores(), # using all available cores
8   control = list(adapt_delta = 0.95) # adjusting the delta step size
9 )
```

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 ordinaire ([Bürkner & Vuorre, 2019](#)).

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



# Modèles de méta-analyse

# Méta-analyse

Une méta-analyse est simplement une analyse d'analyses. Il s'agit d'un modèle linéaire (presque) comme les autres, sauf que nos observations sont (généralement) des données déjà résumées par une taille d'effet (ou pas). On peut traiter ces tailles d'effets comme des observations, avec une variance connue.

On distingue deux classes de modèles :

- Méta-analyse à effets fixes (constant effects) : on considère que la taille d'effet estimée par toutes les études est la même.
- Méta-analyse à effets aléatoires (varying effects) : on modélise la variabilité et les sources de dépendance dans les données.

On ne considère que le deuxième type de modèle, en notant (cf. Cours n°08) qu'un modèle à effet fixe peut être considéré comme un modèle à effet aléatoire dont on a fixé  $\tau = 0$ .



# Méta-analyse

Le jeu de données ci-dessous recense les résultats de 32 expériences visant à évaluer l'effet des contraintes biomécaniques sur l'évaluation des distances ([Molto et al., 2020](#)).

```
1 d <- open_data(meta)
2 head(d, 15)
```

	study	experiment	yi	vi
1	Costello (2015)		1 0.735209366	0.29846545
2	Costello (2015)		2 0.925134052	0.04796738
3	Durgin & Russell (2008)		1 -0.124624069	0.17928177
4	Hutchison & Loomis (2006)		1 -0.128328394	0.10001576
5	Hutchison & Loomis (2006)		2 0.350487089	0.05963093
6	Kirsch & Kunde (2012)		1 0.609278390	0.31866670
7	Kirsch & Kunde (2013a)		1 0.511022255	0.16231426
8	Kirsch & Kunde (2013a)		2 0.407867633	0.04216585
9	Kirsch & Kunde (2013b)		1 0.229159882	0.32082216
10	Kirsch & Kunde (2013b)		2 -0.223592130	0.08876431
11	Kirsch & Kunde (2013b)		3 0.145865089	0.10038189
12	Lessard, Linkenauger & P. (2009)		1 0.172318408	0.09078810
13	Morgado & al. (2013)		1 0.002778791	0.13837508
14	Osiurak & Morgado (2012)		1 0.460865217	0.22233808
15	Proffitt, S, B & Epstein (2003)		1 0.351381281	0.10386001



# Méta-analyse

On peut écrire un premier modèle de la manière suivante.

$$y_j \sim \text{Normal}(\mu_j, \sigma_j)$$

$$\mu_j = \alpha + \alpha_{\text{study}[j]}$$

$$\alpha_{\text{study}[j]} \sim \text{Normal}(0, \tau_s)$$

$$\alpha \sim \text{Normal}(0, 1)$$

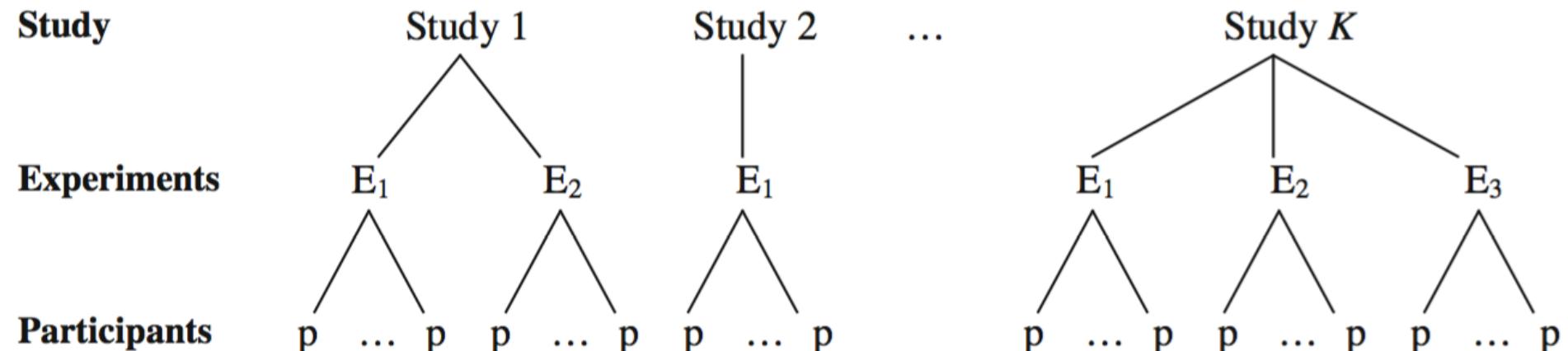
$$\tau_s \sim \text{HalfCauchy}(0, 1)$$

Où  $\sigma_j^2 = \nu_j$  est la variance de l'effet dans l'étude  $j$  et  $\alpha$  la taille d'effet dans la population. L'index  $\alpha_{\text{study}[j]}$  indique la taille d'effet moyenne dans l'étude  $j$ . En plus de la variance dans l'échantillon  $\sigma_j^2$  (qui est connue), on estime la variabilité des effets entre les études  $\text{Var}(\alpha_{\text{study}}) = \tau_s^2$  (niveau 2).



# Méta-analyse

Le modèle précédent peut être étendu à plus de deux niveaux pour prendre en compte les structures de dépendance dans le jeu de données. En effet, chaque étude (chaque papier publié) contenait plusieurs expériences. On pourrait s'attendre à ce que les expériences d'une même étude se ressemblent plus entre elles...



# Méta-analyse

On peut écrire ce modèle sur trois niveaux, comme ci-dessous.

$$\begin{aligned}
 y_{ij} &\sim \text{Normal}(\mu_{ij}, \sigma_{ij}) \\
 \mu_{ij} &= \alpha + \alpha_{\text{study}[j]} + \alpha_{\text{experiment}[ij]} \\
 \alpha_{\text{study}[j]} &\sim \text{Normal}(0, \tau_s) \\
 \alpha_{\text{experiment}[ij]} &\sim \text{Normal}(0, \tau_e) \\
 \alpha &\sim \text{Normal}(0, 1) \\
 \tau_e, \tau_s &\sim \text{HalfCauchy}(0, 1)
 \end{aligned}$$

On estime maintenant, en plus de la variabilité  $\sigma_{ij}$ , deux autres sources de variation : la variation des effets entre différentes expériences issues d'une même étude  $\text{Var}(\alpha_{\text{experiment}}) = \tau_e^2$  (niveau 2) et la variation entre différentes études  $\text{Var}(\alpha_{\text{study}}) = \tau_s^2$  (niveau 3).



# Méta-analyse

Ce modèle se fit facilement avec `brms`.

```
1 prior4 <- c(  
2   prior(normal(0, 1), coef = intercept),  
3   prior(cauchy(0, 1), class = sd)  
4 )  
5  
6 mod4 <- brm(  
7   formula = yi | se(sqrt(vi)) ~ 0 + intercept + (1 | study) + (1 | experiment),  
8   data = d,  
9   prior = prior4,  
10  save_all_pars = TRUE,  
11  warmup = 2000, iter = 1e4,  
12  cores = parallel::detectCores(),  
13  control = list(adapt_delta = 0.99),  
14  backend = "cmdstanr"  
15 )
```



# Méta-analyse

```
1 summary(mod4)
```

Family: gaussian  
 Links: mu = identity; sigma = identity  
 Formula: yi | se(sqrt(vi)) ~ 0 + intercept + (1 | study) + (1 | experiment)  
 Data: d (Number of observations: 32)  
 Draws: 4 chains, each with iter = 10000; warmup = 2000; thin = 1;  
 total post-warmup draws = 32000

**Group-Level Effects:**

	Estimate	Est.Error	l-95%	CI	u-95%	CI	Rhat	Bulk_ESS	Tail_ESS
~experiment (Number of levels: 4)									
sd(Intercept)	0.35	0.26	0.06	1.01	1.00	9388	9594		

~study (Number of levels: 19)

	Estimate	Est.Error	l-95%	CI	u-95%	CI	Rhat	Bulk_ESS	Tail_ESS
sd(Intercept)	0.22	0.10	0.04	0.43	1.00	9846	9619		

**Population-Level Effects:**

	Estimate	Est.Error	l-95%	CI	u-95%	CI	Rhat	Bulk_ESS	Tail_ESS
intercept	0.42	0.22	-0.02	0.88	1.00	11052	13321		

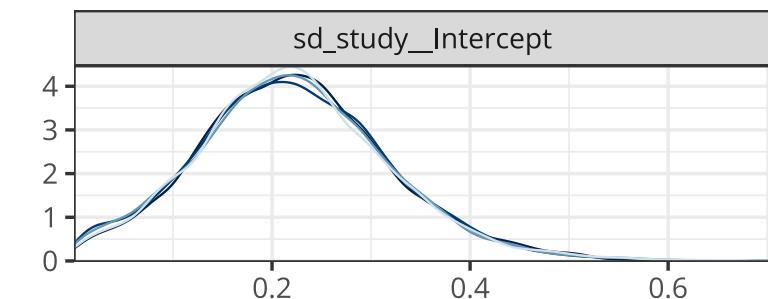
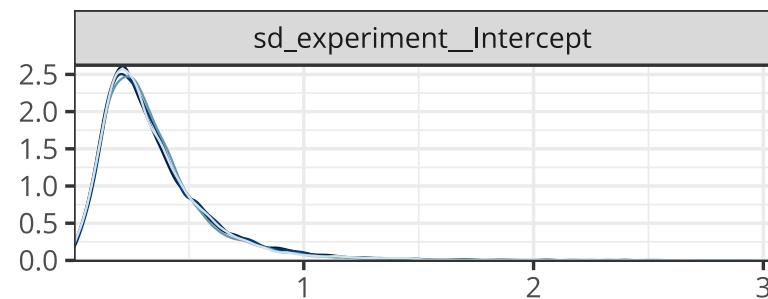
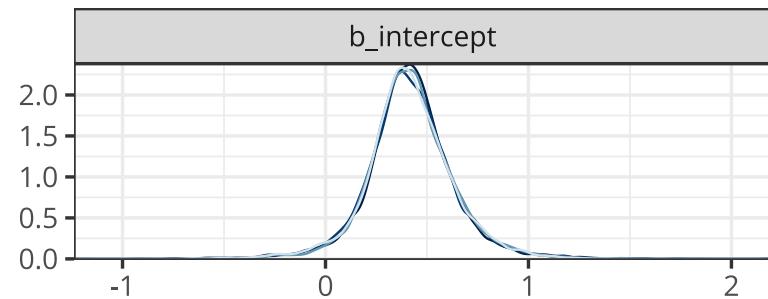
**Family Specific Parameters:**



# Méta-analyse

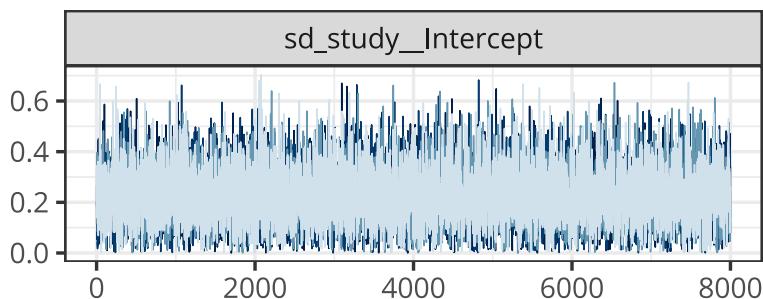
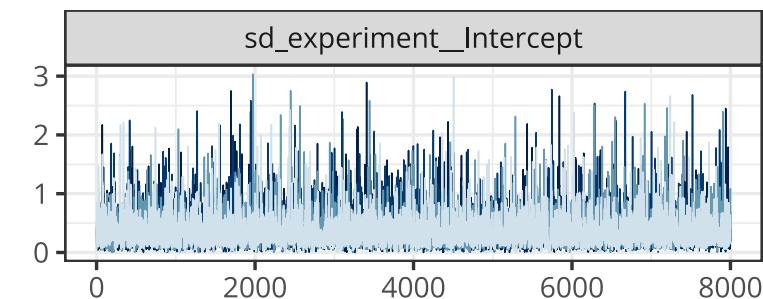
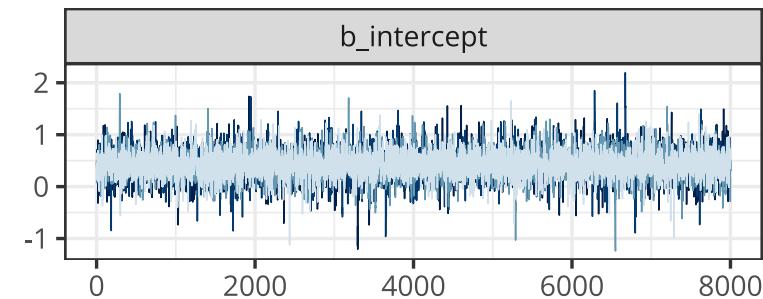
```

1 mod4 %>%
2   plot(
3     pars = c("b_",
4     "sd_"),
5     combo = c("dens_overlay", "trace"),
6     theme = theme_bw(base_size = 16, base_family = "Open Sans")
7   )
  
```



Chain

- 1
- 2
- 3
- 4

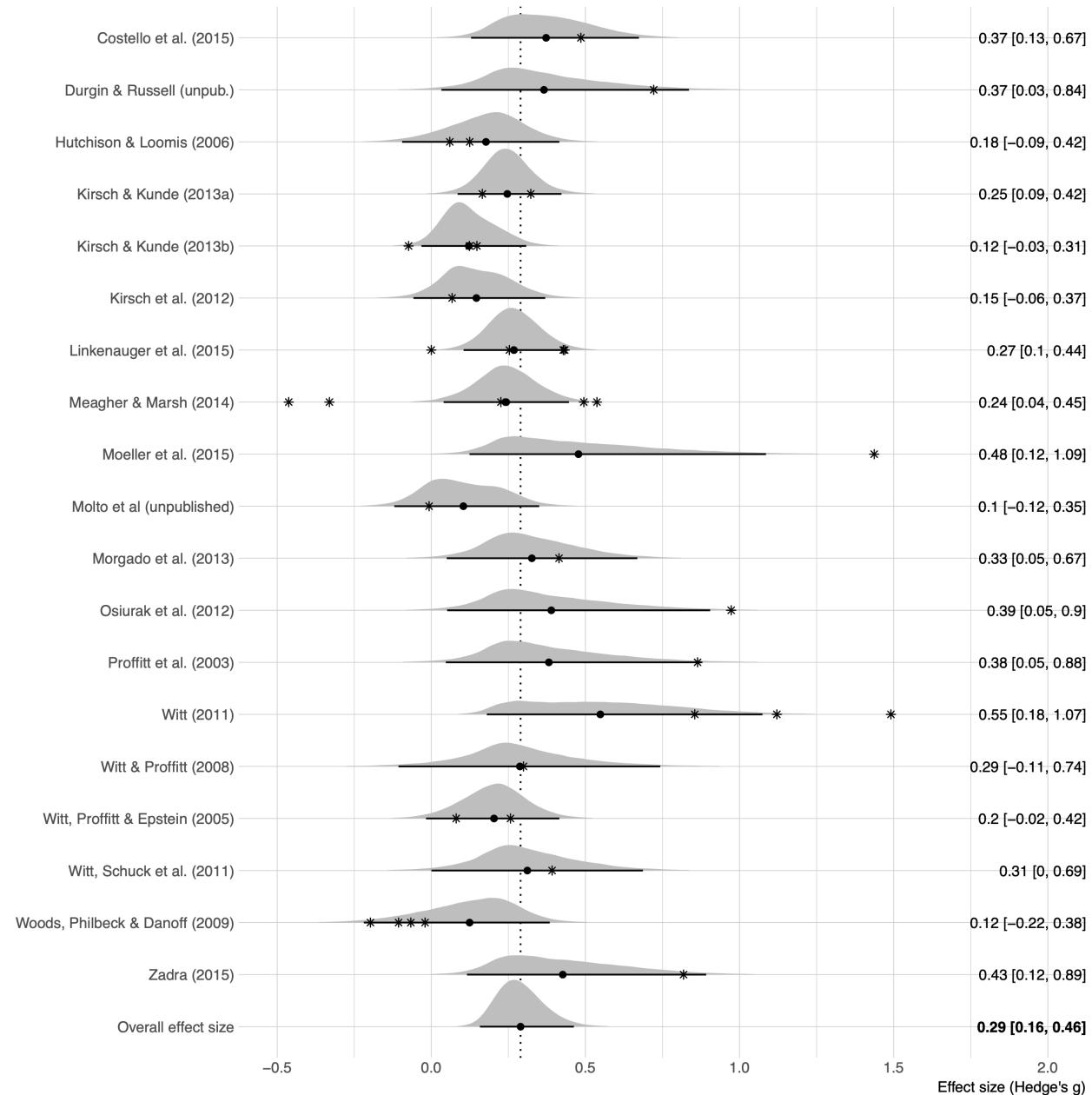


Chain

- 1
- 2
- 3
- 4



# Méta-analyse



# 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 prédictrice 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 outcomes), des modèles “distributionnels” (e.g., pour prédire des différence de variance), des generalised additive models, des processus Gaussiens (Gaussian processes), des modèles issus de la théorie de détection du signal, des 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.



# Bayesian workflow (Gelman et al., 2020)

## Bayesian workflow\*

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

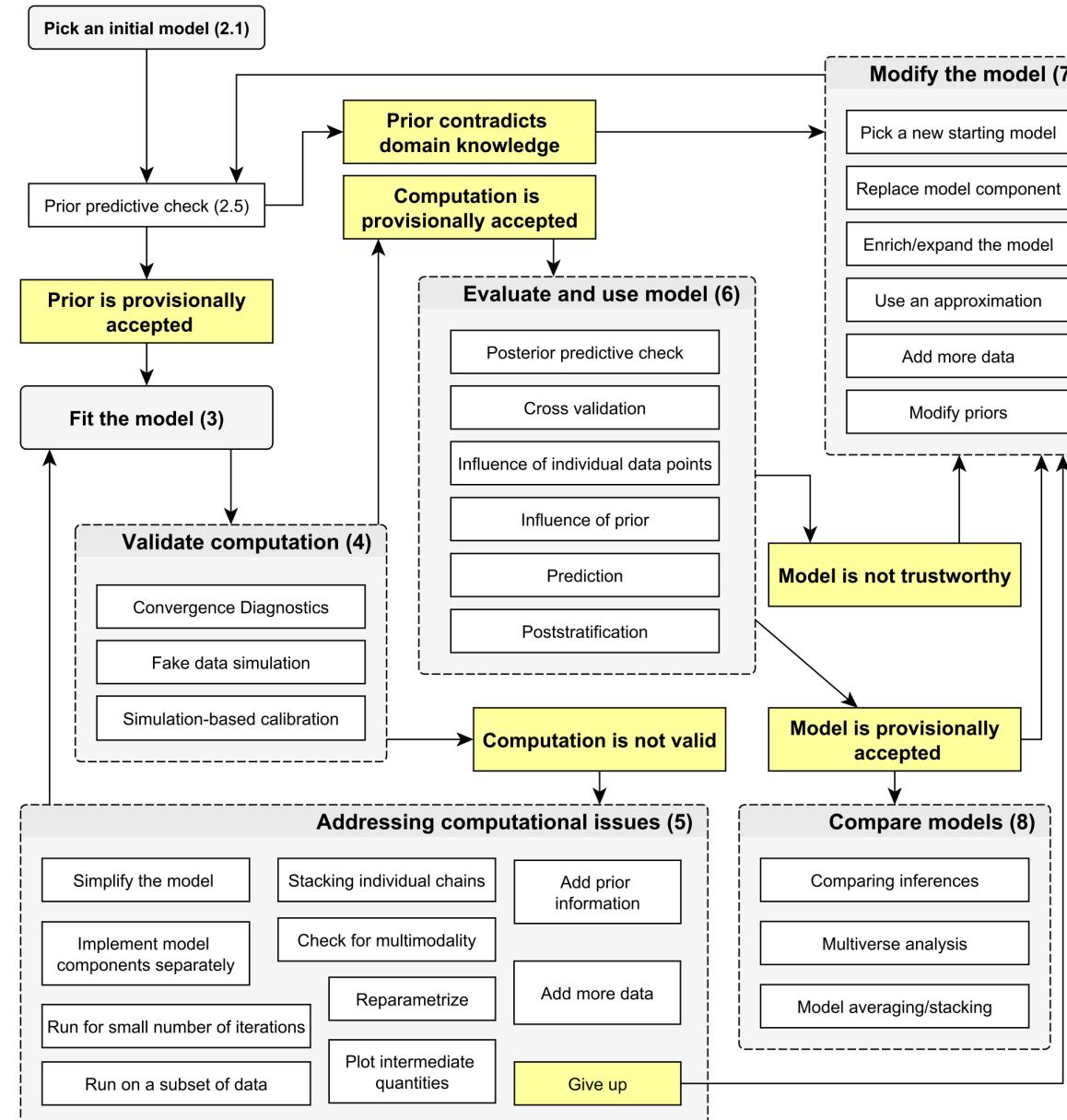
2 Nov 2020

### 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)



# 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).

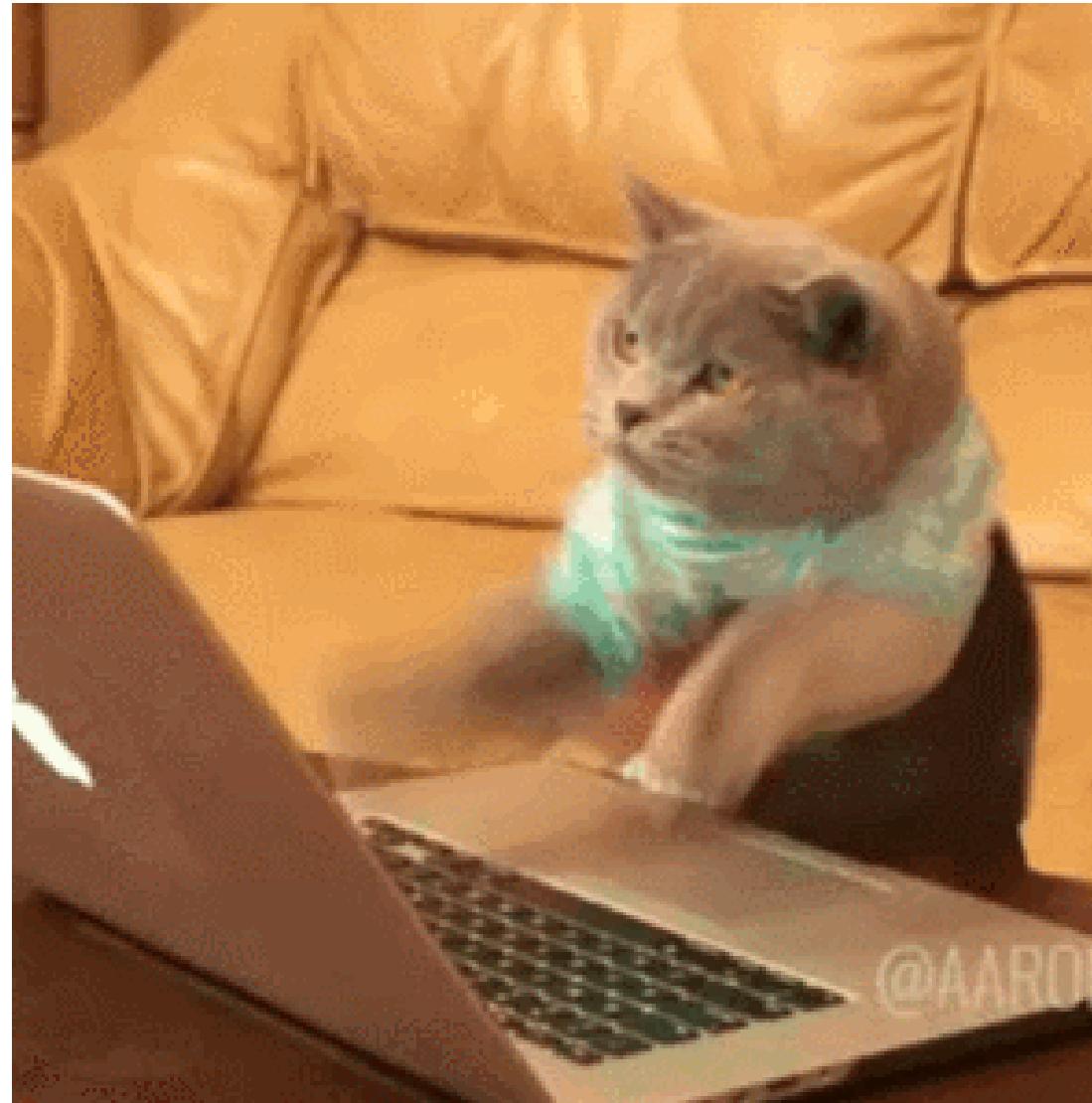
```
1 d <- open_data(popular)
2 head(d, 10)
```

	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



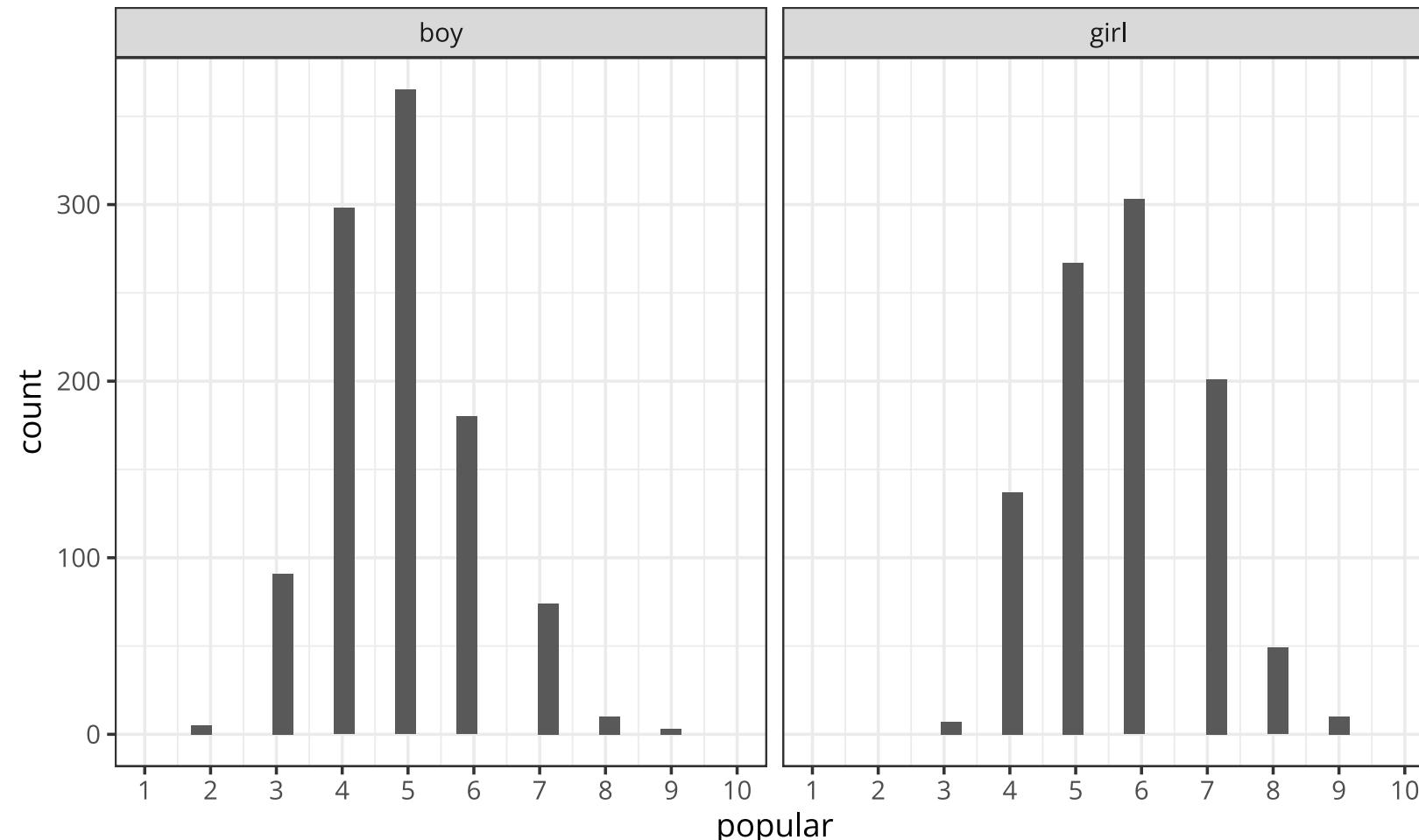
# Travaux pratiques

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...



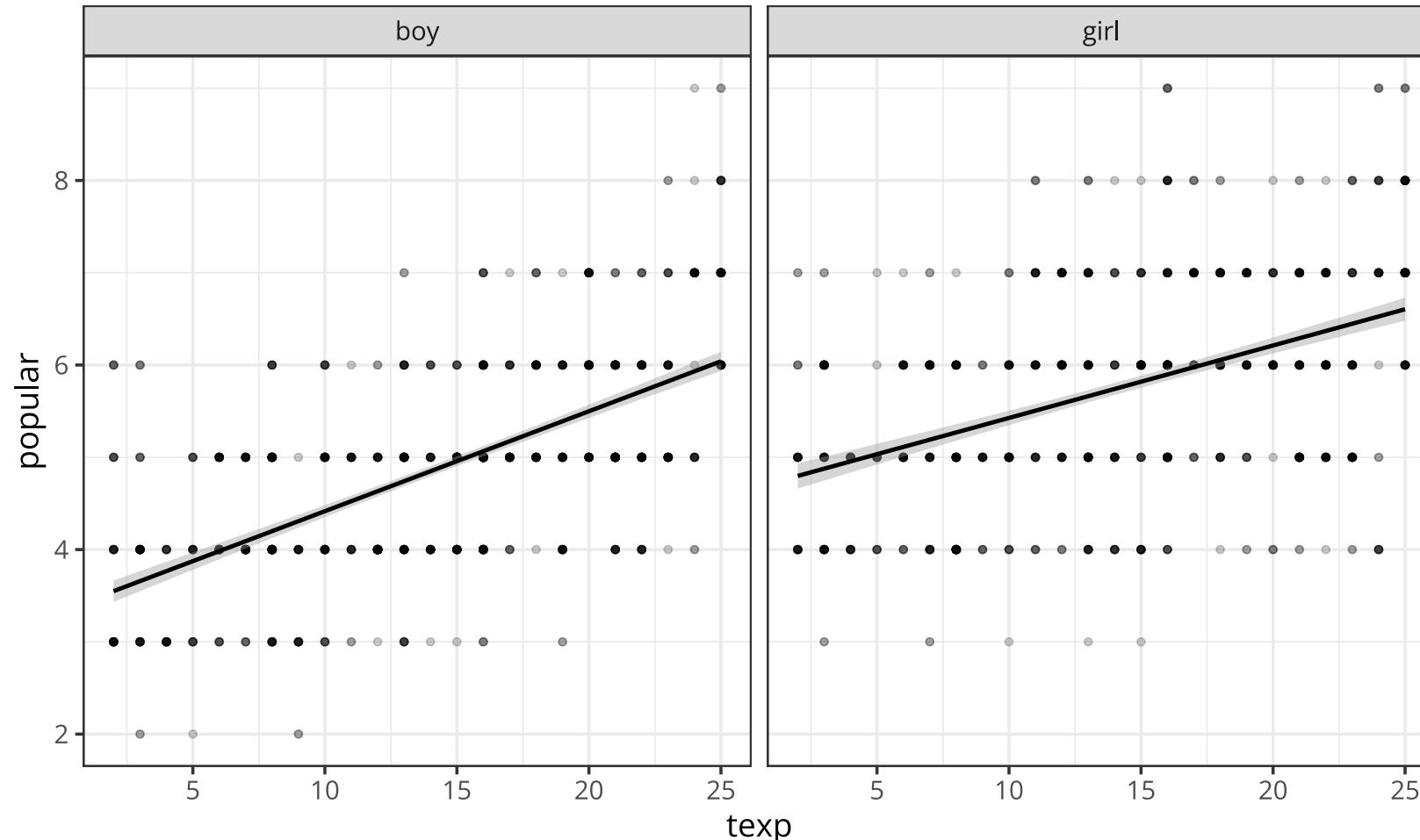
# Proposition de solution - exploration graphique

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



# Proposition de solution - exploration graphique

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



# Proposition de solution

```
1 d <- d %>%
2   mutate(
3     # using a sum contrast for gender
4     sex = ifelse(sex == "boy", -0.5, 0.5),
5     # centering and standardising teacher experience
6     texp = scale(texp) %>% as.numeric
7   )
8
9 prior5 <- c(
10   prior(normal(5, 2.5), class = Intercept),
11   prior(cauchy(0, 10), class = sd),
12   prior(cauchy(0, 10), class = sigma)
13 )
14
15 mod5 <- brm(
16   formula = popular ~ 1 + (1 | school),
17   data = d,
18   prior = prior5,
19   save_all_pars = TRUE,
20   warmup = 2000, iter = 1e4,
21   cores = parallel::detectCores()
22 )
```



# Proposition de solution

```
1 prior6 <- c(  
2     prior(normal(0, 1), class = Intercept),  
3     prior(normal(0, 1), class = b),  
4     prior(cauchy(0, 1), class = sd),  
5     prior(cauchy(0, 10), class = sigma)  
6 )  
7  
8 mod6 <- brm(  
9     formula = popular ~ 1 + texp + (1 | school),  
10    data = d,  
11    prior = prior6,  
12    save_all_pars = TRUE,  
13    warmup = 2000, iter = 1e4,  
14    cores = parallel::detectCores()  
15 )
```



# Proposition de solution

```
1 prior7 <- c(  
2     prior(normal(0, 1), class = Intercept),  
3     prior(normal(0, 1), class = b),  
4     prior(cauchy(0, 1), class = sd),  
5     prior(cauchy(0, 10), class = sigma),  
6     prior(lkj(2), class = cor)  
7 )  
8  
9 mod7 <- brm(  
10     formula = popular ~ 1 + sex + texp + (1 + sex | school),  
11     data = d,  
12     prior = prior7,  
13     save_all_pars = TRUE,  
14     warmup = 2000, iter = 1e4,  
15     cores = parallel::detectCores()  
16 )
```



# Proposition de solution

```
1 mod8 <- brm(  
2   formula = popular ~ 1 + sex + texp + sex:texp + (1 + sex | school),  
3   data = d,  
4   prior = prior7,  
5   save_all_pars = TRUE,  
6   warmup = 2000, iter = 1e4,  
7   cores = parallel::detectCores()  
8 )
```

```
1 # calcul du WAIC et ajout du WAIC à chaque modèle  
2 mod5 <- add_criterion(mod5, "waic")  
3 mod6 <- add_criterion(mod6, "waic")  
4 mod7 <- add_criterion(mod7, "waic")  
5 mod8 <- add_criterion(mod8, "waic")
```



# Proposition de solution

```

1 # comparaison des WAIC de chaque modèle
2 model_comparison_table <- loo_compare(mod5, mod6, mod7, mod8, criterion = "waic") %>%
3   data.frame() %>%
4   rownames_to_column(var = "model")
5
6 weights <- data.frame(weight = model_weights(mod5, mod6, mod7, mod8, weights = "waic")) %>%
7   round(digits = 3) %>%
8   rownames_to_column(var = "model")
9
10 left_join(model_comparison_table, weights, by = "model")

```

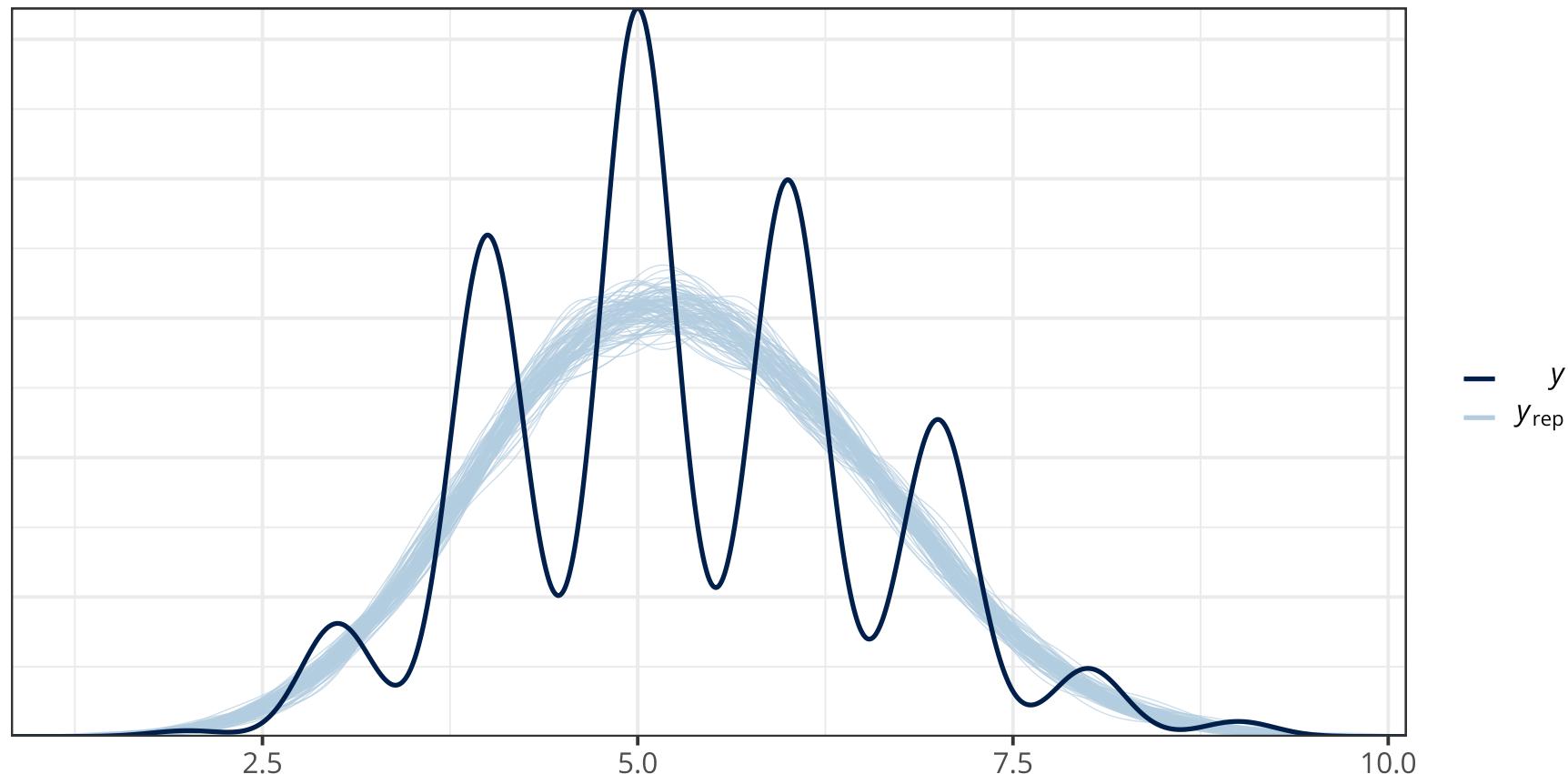
	model	elpd_diff	se_diff	elpd_waic	se_elpd_waic	p_waic	se_p_waic
1	mod8	0.000000	0.000000	-1990.388	32.53888	161.02615	5.163683
2	mod7	-1.914978	2.069366	-1992.303	32.70893	164.12668	5.283776
3	mod6	-448.540179	25.834425	-2438.928	30.21694	93.11361	2.790621
4	mod5	-449.486597	25.895504	-2439.874	30.25696	95.14274	2.852372
		waic	se_waic	weight			
1	3980.775	65.07776	0.872				
2	3984.605	65.41786	0.128				
3	4877.856	60.43388	0.000				
4	4879.748	60.51392	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 = mod8, nsamples = 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 ordinaire (ordered categorical model). 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.

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

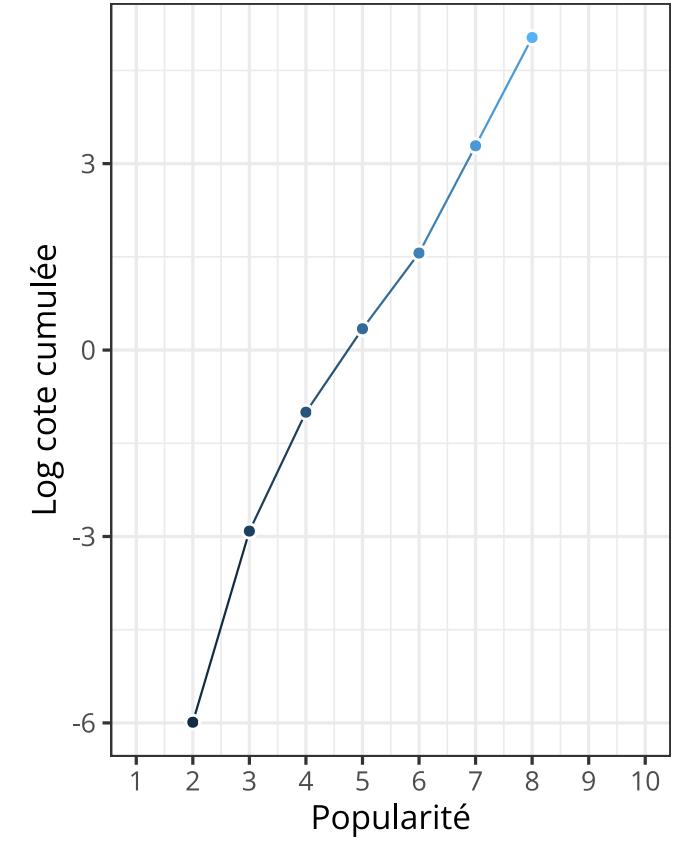
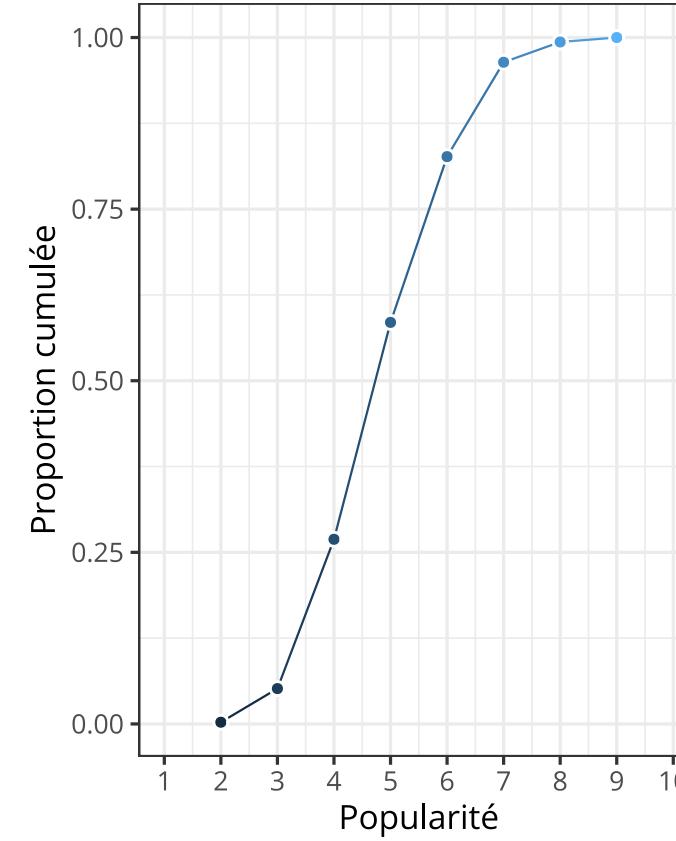
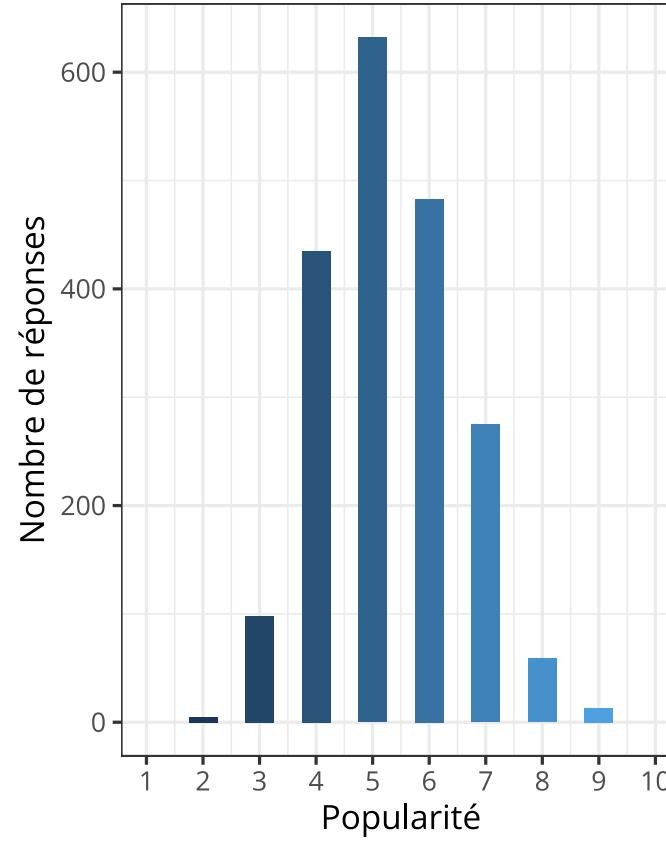
Où la distribution Categorical est une distribution discrète qui prend un vecteur de probabilités  $\mathbf{p} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$  qui correspondent aux probabilités cumulées de chaque réponse (entre 1 et 10, 10 ayant une probabilité cumulée de 1).



# 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).

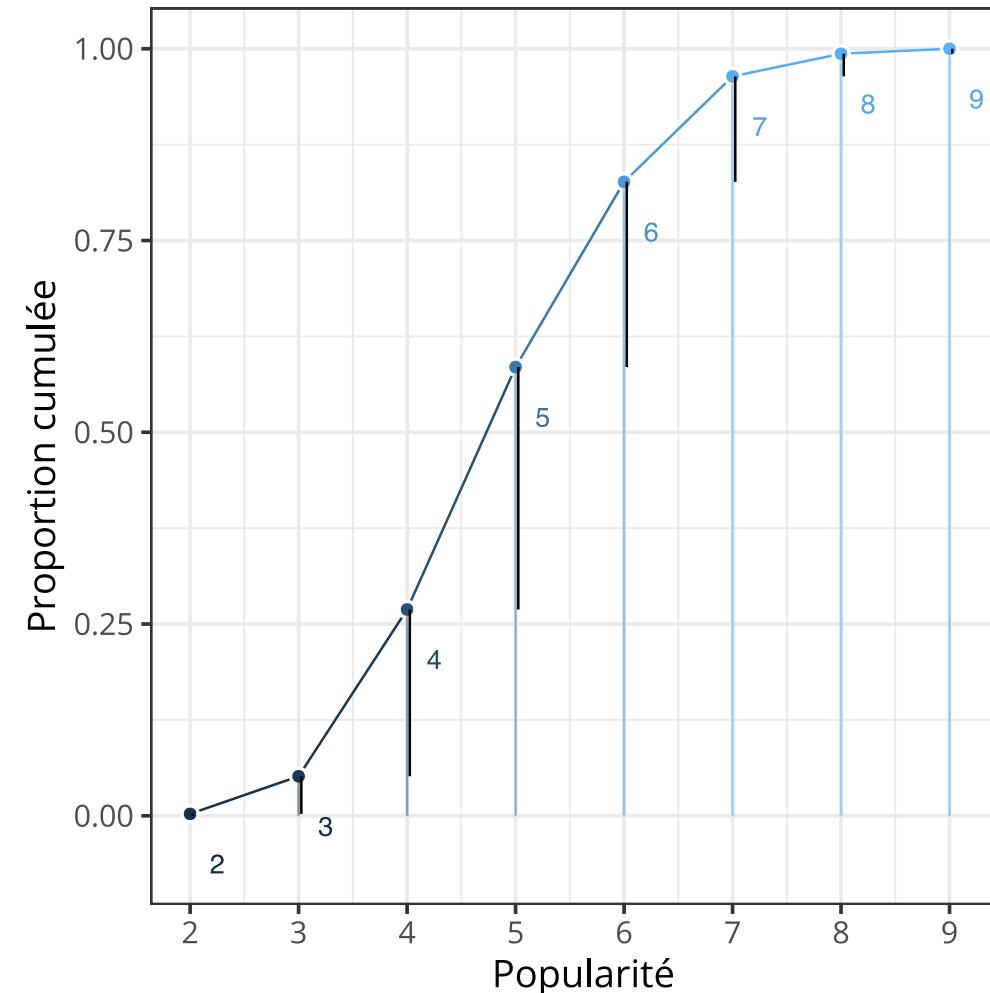
$$\text{logit}(p_k) = \log \frac{\Pr(y_i \leq k)}{1 - \Pr(y_i \leq k)} = \alpha_k$$



# 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...

```

1 mod9 <- brm(
2   popular ~ 1 + sex + texp + sex:texp + (1 | school),
3   data = d,
4   prior = prior6,
5   warmup = 2000, iter = 5000,
6   chains = 4, cores = parallel::detectCores(),
7   file = "models/mod9", backend = "cmdstanr"
8 )

```

```

1 prior10 <- c(
2   brms::prior(normal(0, 10), class = Intercept),
3   brms::prior(normal(0, 10), class = b),
4   brms::prior(cauchy(0, 10), class = sd)
5 )
6
7 mod10 <- brm(
8   popular ~ 1 + sex + texp + sex:texp + (1 | school),
9   data = d,
10  family = cumulative(link = "logit"),
11  prior = prior10,
12  chains = 4, cores = parallel::detectCores(),
13  control = list(adapt_delta = 0.99, max_treedepth = 15),
14  file = "models/mod10", backend = "cmdstanr"
15 )

```



# Proposition de solution

```
1 waic(mod9, mod10, compare = FALSE)
```



Output of model 'mod9':

Computed from 12000 by 2000 log-likelihood matrix

	Estimate	SE
elpd_waic	-2086.1	31.3
p_waic	96.3	3.0
waic	4172.2	62.6

7 (0.4%) p\_waic estimates greater than 0.4. We recommend trying loo instead.

Output of model 'mod10':

Computed from 4000 by 2000 log-likelihood matrix

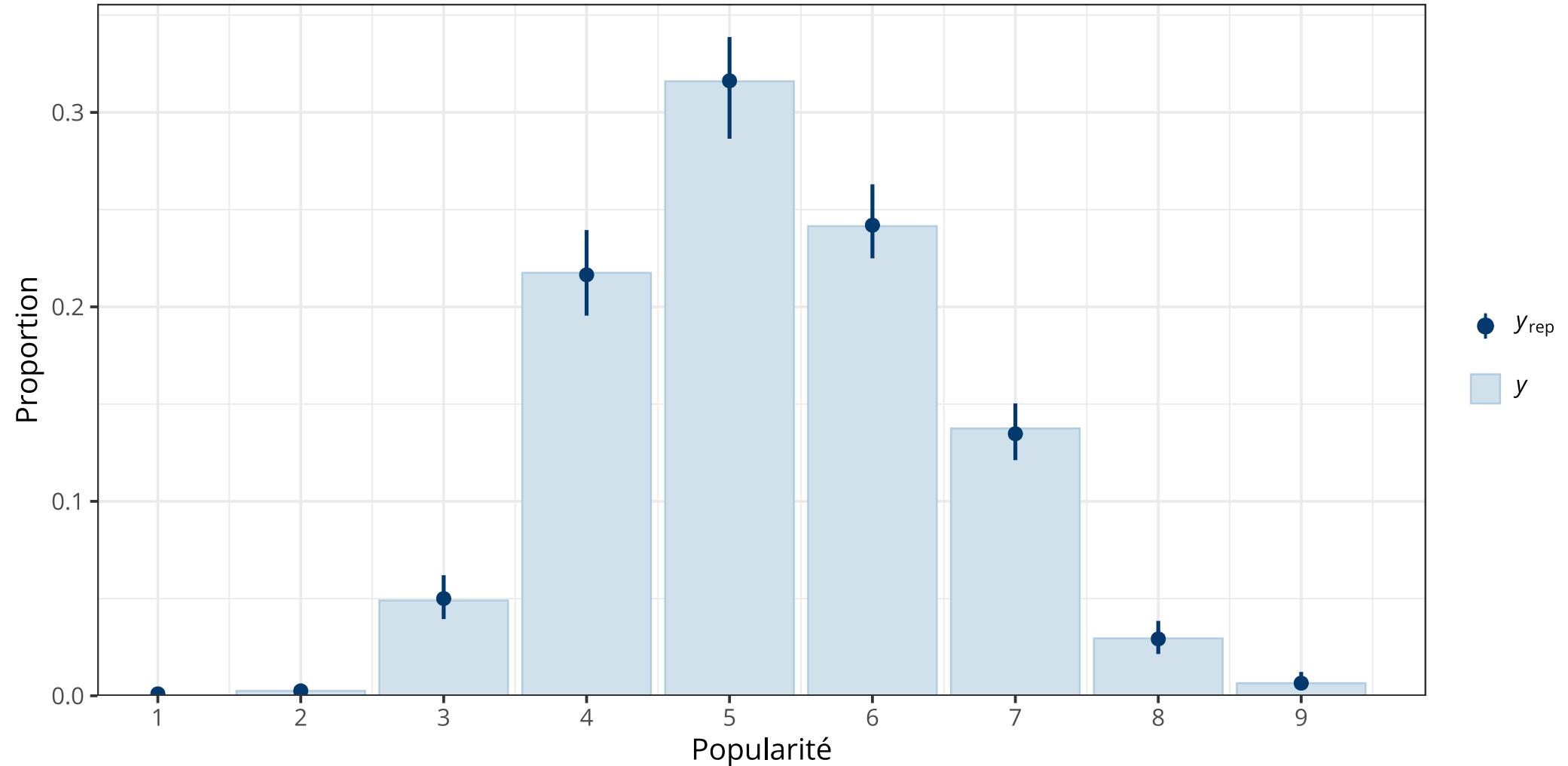
	Estimate	SE
elpd_waic	-2075.0	32.4
p_waic	106.7	2.6
waic	4150.1	64.9

3 (0.1%) p\_waic estimates greater than 0.4. We recommend trying loo instead.



# Proposition de solution

```
1 pp_check(mod10, nsamples = 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>
- 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., 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>
- Molto, L., Nalborczyk, L., Palluel-Germain, R., & Morgado, N. (2020). Action Effects on Visual Perception of Distances: A Multilevel Bayesian Meta-Analysis: *Psychological Science*. <https://doi.org/10.1177/0956797619900336>
- Nalborczyk, L., Batailler, C., Loevenbruck, 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](https://doi.org/10.1044/2018_jslhr-s-18-0006)

