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

Null Hypothesis Significance Testing (NHST)

On s'intéresse aux différences de taille entre hommes et femmes. On va mesurer 100 femmes et 100 hommes.

```r
set.seed(19) # pour reproduire les résultats
men <- rnorm(100, 175, 10) # 100 tailles d'hommes
women <- rnorm(100, 170, 10) # 100 tailles de femmes
```

```r
t.test(men, women) # test de student pour différence de moyenne
```

```
    Welch Two Sample t-test

data:  men and women
t = 2.4969, df = 197.98, p-value = 0.01335
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.7658541 6.5209105
sample estimates:
mean of x mean of y
 175.1258  171.4825
```

Null Hypothesis Significance Testing (NHST)

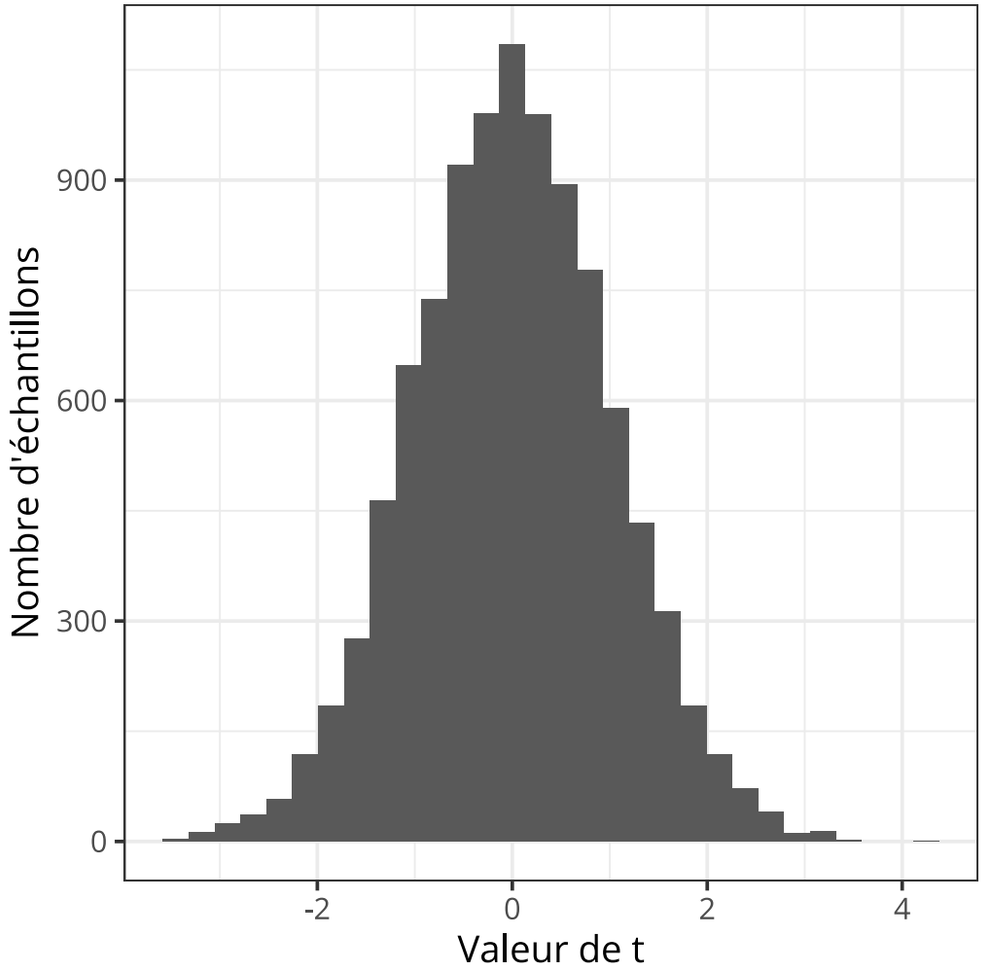
On va simuler des t-valeurs issues de données générées sous l'hypothèse d'une absence de différence entre hommes et femmes.

```r
nsims <- 1e4 # nombre de simulations
t <- rep(NA, nsims) # initialisation d'un vecteur vide

for (i in 1:nsims) {

    men2 <- rnorm(100, 170, 10) # 100 tailles d'hommes
    women2 <- rnorm(100, 170, 10) # 100 tailles de femmes (de même distribution)
    t[i] <- t.test(men2, women2)$statistic # on conserve la t-valeur

}
```

```r
# une autre manière de réaliser la même opération, sans boucle for
t <- replicate(nsims, t.test(rnorm(100, 170, 10), rnorm(100, 170, 10) )$statistic)
```

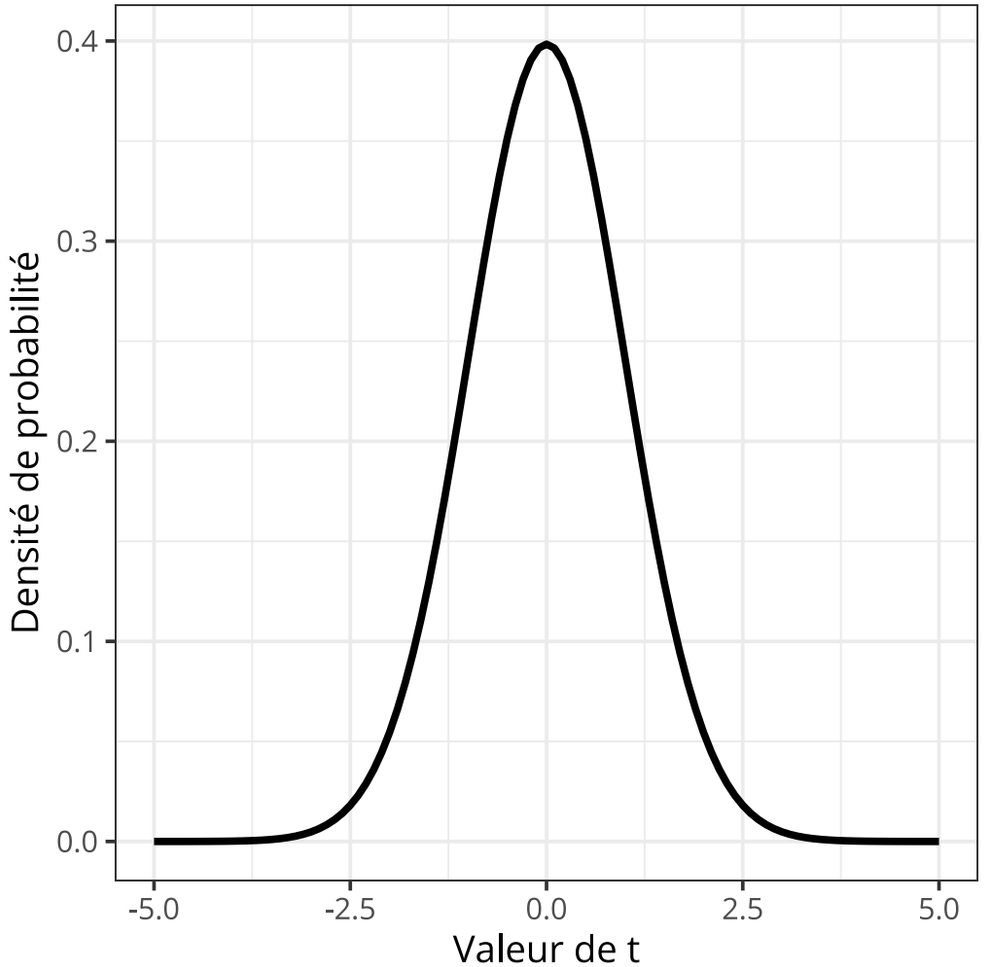
Null Hypothesis Significance Testing (NHST)

```r
data.frame(t = t) %>%
    ggplot(aes(x = t) ) +
    geom_histogram() +
    labs(x = "Valeur de t", y = "Nombre d'échantillons")
```



Null Hypothesis Significance Testing (NHST)

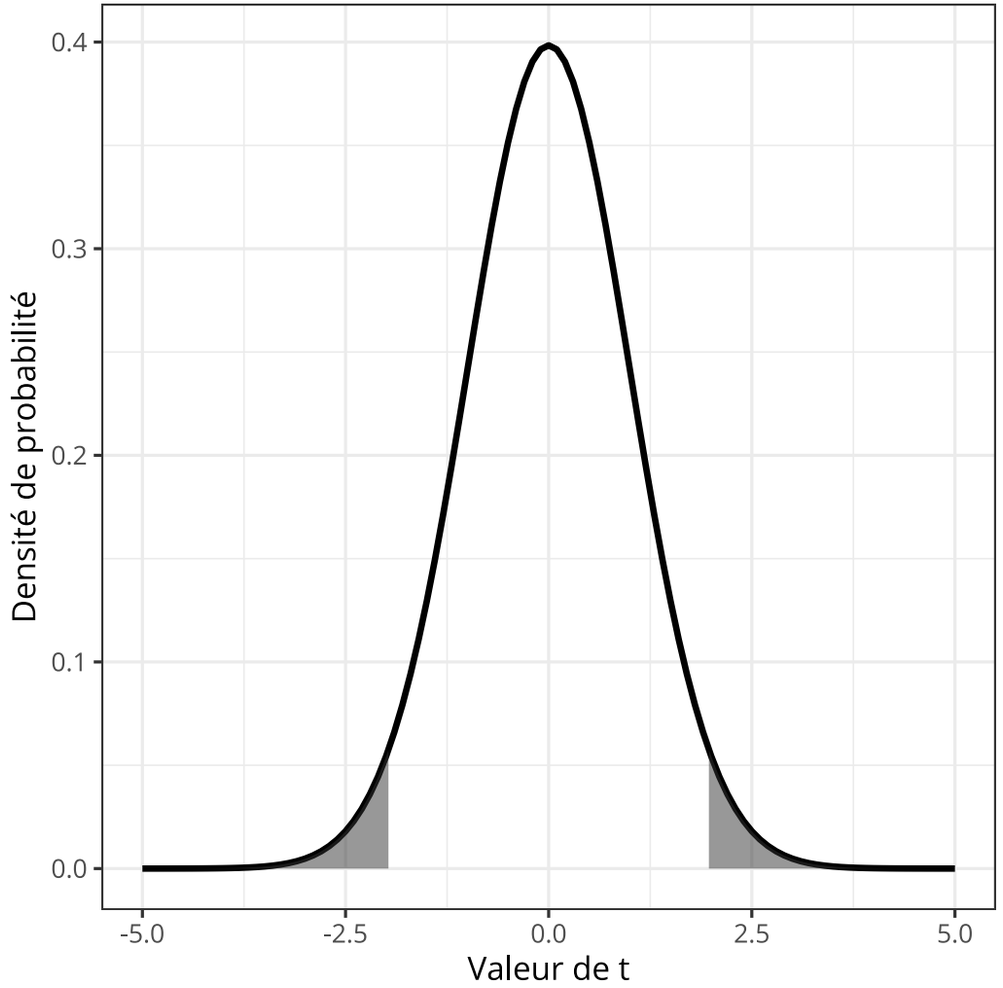
```r
data.frame(x = c(-5, 5) ) %>%
    ggplot(aes(x = x) ) +
    stat_function(fun = dt, args = list(df = t.test(men, women)$parameter), size = 1.5) +
    labs(x = "Valeur de t", y = "Densité de probabilité")
```



Null Hypothesis Significance Testing (NHST)

```r
alpha <- 0.05 # seuil de significativité
abs(qt(p = alpha / 2, df = t.test(men, women)$parameter) ) # valeur de t critique
```

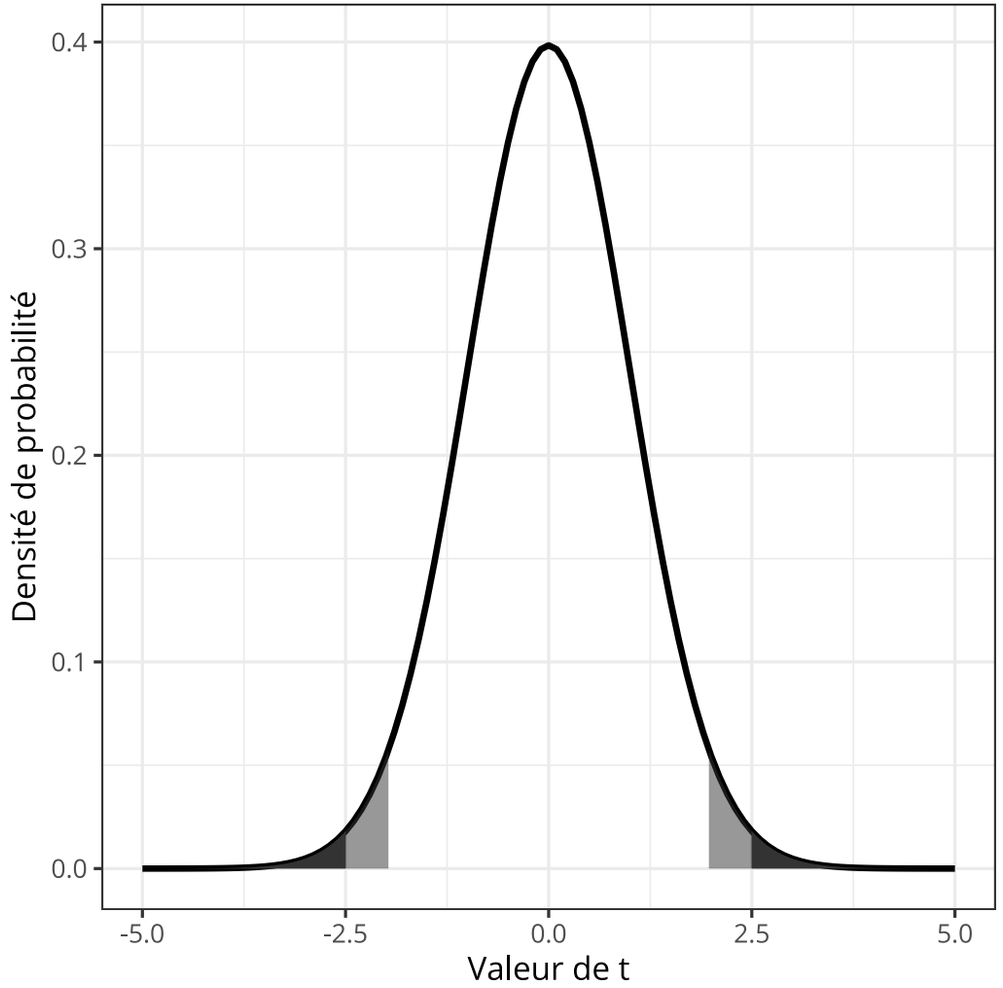
```
[1] 1.972019
```



Null Hypothesis Significance Testing (NHST)

```r
tobs <- t.test(men, women)$statistic # valeur de t observée
tobs %>% as.numeric
```

```
[1] 2.496871
```



P-valeur

Une p-valeur est une aire sous la courbe (une intégrale) sous la distribution de statistiques de test sous l'hypothèse nulle (i.e., étant admis que l'hypothèse nulle est vraie). La p-valeur indique la probabilité d'observer la valeur de la statistique de test (e.g., une valeur de t) observée, ou une valeur plus extrême, sous l'hypothèse nulle.

$$p[\mathbf{t}(\mathbf{x}^{\text{rep}}; \mathcal{H}_0) \geq t(x)]$$

```r
t.test(men, women)$p.value
```

```
[1] 0.01334509
```

```r
tvalue <- abs(t.test(men, women)$statistic)
df <- t.test(men, women)$parameter
2 * integrate(dt, tvalue, Inf, df = df)$value
```

```
[1] 0.01334509
```

```r
2 * (1 - pt(abs(t.test(men, women)$statistic), t.test(men, women)$parameter) )
```

```
          t
0.01334509
```


Intervalles de confiance

Les intervalles de confiance peuvent être interprétés comme des régions de significativité (ou des régions de **compatibilité** avec l'hypothèse nulle). Par conséquent, un intervalle de confiance contient la même information qu'une p-valeur et s'interprète de manière similaire.

On ne peut pas dire qu'un intervalle de confiance a une probabilité de 95% de contenir la vraie valeur (i.e., la valeur dans la population) de θ (cf. the inverse fallacy), contrairement à l'intervalle de crédibilité bayésien.

Un intervalle de confiance à 95% représente une degré de "recouvrement" (coverage). Le "95%" fait référence à une propriété fréquentiste (i.e., sur le long-terme) de la procédure, mais ne fait pas référence au paramètre θ. Autrement dit, sur le long-terme, 95% des intervalles de confiance à 95% que l'on pourrait calculer (dans une réplication exacte de notre expérience) contiendraient la valeur du paramètre dans la population (i.e., la "vraie" valeur de θ). Cependant, nous ne pouvons pas dire qu'un intervalle de confiance en particulier a une probabilité de 95% de contenir la "vraie" valeur de θ... soit ce dernier contient la vraie valeur de θ, soit il ne la contient pas.

Facteur de Bayes

On compare deux modèles :

- $\mathcal{H}_0 : \mu_1 = \mu_2 \rightarrow \delta = 0$
- $\mathcal{H}_1 : \mu_1 \neq \mu_2 \rightarrow \delta \neq 0$

$$\underbrace{\frac{p(\mathcal{H}_0 \mid D)}{p(\mathcal{H}_1 \mid D)}}_{\text{posterior odds}} = \underbrace{\frac{p(D \mid \mathcal{H}_0)}{p(D \mid \mathcal{H}_1)}}_{\text{Bayes factor}} \times \underbrace{\frac{p(\mathcal{H}_0)}{p(\mathcal{H}_1)}}_{\text{prior odds}}$$

$$\text{evidence} = p(D \mid \mathcal{H}) = \int p(\theta \mid \mathcal{H}) p(D \mid \theta, \mathcal{H}) \mathrm{d}\theta$$

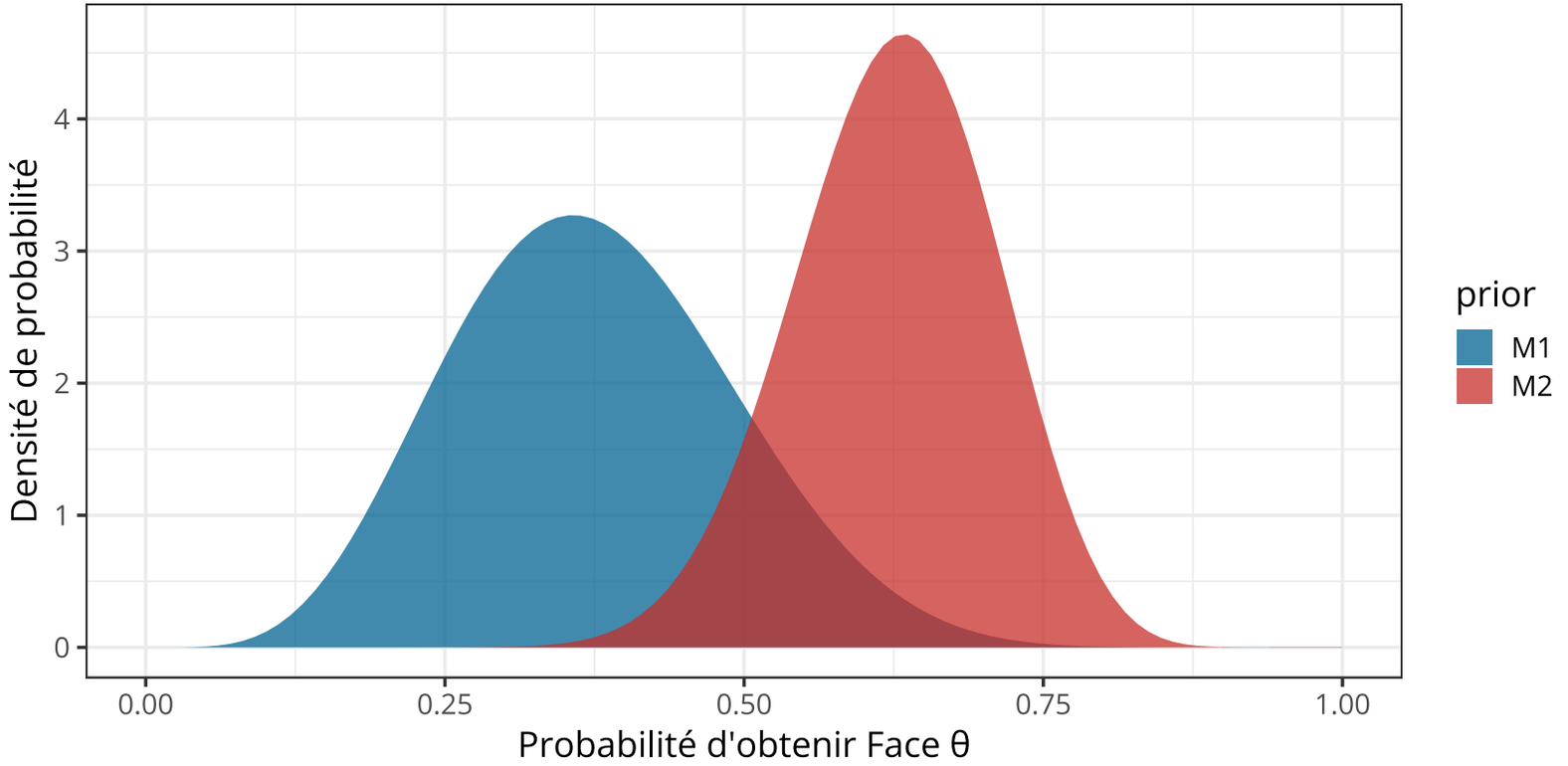
L'évidence en faveur d'un modèle correspond à la **vraisemblance marginale** d'un modèle (le dénominateur du théorème de Bayes), c'est à dire à la vraisemblance moyennée sur le prior... Ce qui fait du facteur de Bayes un ratio de vraisemblances, pondéré par (ou moyenné sur) le prior.

Facteur de Bayes, exemple d'application

On lance une pièce 100 fois et on essaye d'estimer la probabilité θ (le biais de la pièce) d'obtenir Face. On compare deux modèles qui diffèrent par leur prior sur θ.

$$\mathcal{M}_1 : y_i \sim \text{Binomial}(n, \theta)$$
$$\theta \sim \text{Beta}(6, 10)$$

$$\mathcal{M}_2 : y_i \sim \text{Binomial}(n, \theta)$$
$$\theta \sim \text{Beta}(20, 12)$$



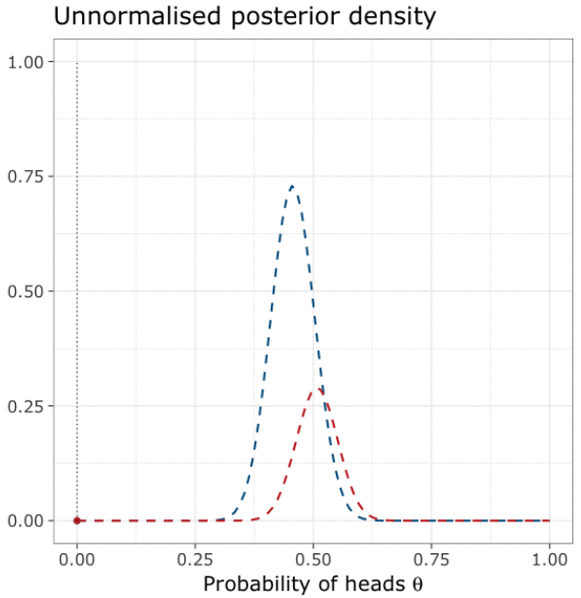
Facteur de Bayes, exemple d'application

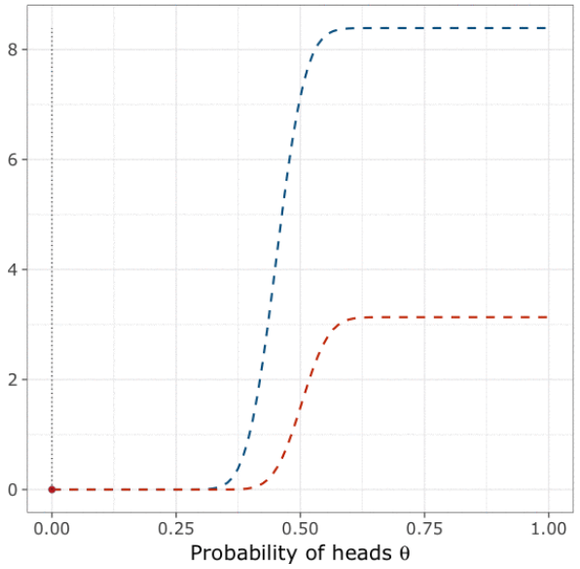
$$\begin{aligned}\mathcal{M}_1 : y_i &\sim \text{Binomial}(n, \theta) \\ \theta &\sim \text{Beta}(6, 10)\end{aligned}$$

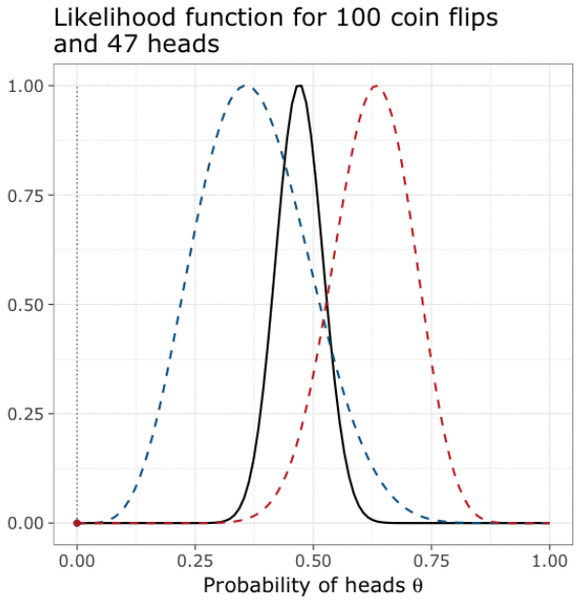
$$\begin{aligned}\mathcal{M}_2 : y_i &\sim \text{Binomial}(n, \theta) \\ \theta &\sim \text{Beta}(20, 12)\end{aligned}$$

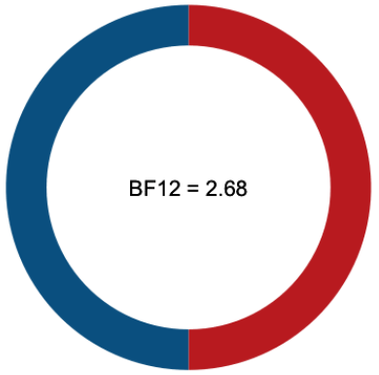
$$\text{BF}_{12} = \frac{p(D|\mathcal{M}_1)}{p(D|\mathcal{M}_2)} = \frac{\int p(\theta|\mathcal{M}_1)p(D|\theta, \mathcal{M}_1)\text{d}\theta}{\int p(\theta|\mathcal{M}_2)p(D|\theta, \mathcal{M}_2)\text{d}\theta} = \frac{\int \text{Binomial}(n, \theta)\text{Beta}(6, 10)\text{d}\theta}{\int \text{Binomial}(n, \theta)\text{Beta}(20, 12)\text{d}\theta}$$

Facteur de Bayes, exemple d'application









Le facteur de Bayes est la nouvelle p-valeur

Attention à ne pas interpréter le BF comme un rapport des chances a posteriori (posterior odds)...

Le BF est un facteur qui nous indique de combien notre rapport des chances a priori (prior odds) doit changer, au vu des données. Il ne nous dit pas quelle est l'hypothèse la plus probable, sachant les données (sauf si les prior odds sont de $1/1$). Par exemple :

- $\mathcal{H}_0$: la précognition n'existe pas.
- $\mathcal{H}_1$: la précognition existe.

On fait une expérience et on calcule un $\mathrm{BF}_{10} = 27$. Quelle est la probabilité a posteriori de $\mathcal{H}_1$?

$$\underbrace{\frac{p(\mathcal{H}_1 \mid D)}{p(\mathcal{H}_0 \mid D)}}_{\text{posterior odds}} = \underbrace{\frac{27}{1}}_{\text{Bayes factor}} \times \underbrace{\frac{1}{1000}}_{\text{prior odds}} = \frac{27}{1000} = 0.027$$

Double sens

Dans le cadre bayésien, le terme de **prior** peut faire référence soit :

- À la probabilité a priori ou a posteriori d'un modèle (par rapport à un autre modèle), c'est à dire $\Pr(\mathcal{M}_i)$. Voir ce blogpost.
- Aux priors qu'on définit sur les paramètres d'un modèle, par exemple $\beta \sim \text{Normal}(0, 1)$. Voir ce blogpost.

Comparaison de modèles

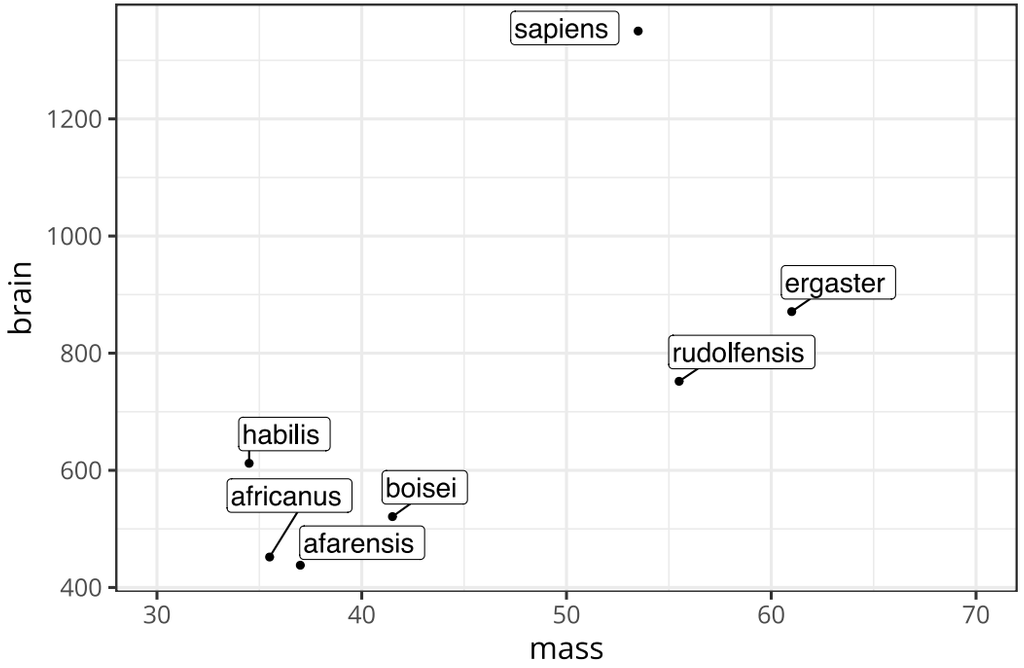
Deux problèmes récurrents à éviter en modélisation : le sous-apprentissage (underfitting) et sur-apprentissage (overfitting). Comment s'en sortir ?

- Utiliser des priors pour **régulariser**, pour contraindre le posterior (i.e., accorder moins de poids à la vraisemblance).
- Utiliser des critères d'information (e.g., AIC, WAIC).

$$R^2 = \frac{\text{var(outcome)} - \text{var(residuals)}}{\text{var(outcome)}} = 1 - \frac{\text{var(residuals)}}{\text{var(outcome)}}$$

Overfitting

```
ppnames <- c("afarensis", "africanus", "habilis", "boisei",
        "rudolfensis", "ergaster", "sapiens")
brainvolcc <- c(438, 452, 612, 521, 752, 871, 1350)
masskg <- c(37.0, 35.5, 34.5, 41.5, 55.5, 61.0, 53.5)

d <- data.frame(species = ppnames, brain = brainvolcc, mass = masskg)

d %>%
    ggplot(aes(x = mass, y = brain, label = species) ) +
    geom_point() +
    ggrepel::geom_label_repel(hjust = 0, nudge_y = 50, size = 5) +
    xlim(30, 70)
```



Overfitting

```r
mod1.1 <- lm(brain ~ mass, data = d)
(var(d$brain) - var(residuals(mod1.1) ) ) / var(d$brain)
```

```
[1] 0.490158
```

```r
mod1.2 <- lm(brain ~ mass + I(mass^2), data = d)
(var(d$brain) - var(residuals(mod1.2) ) ) / var(d$brain)
```

```
[1] 0.5359967
```

```r
mod1.3 <- lm(brain ~ mass + I(mass^2) + I(mass^3), data = d)
(var(d$brain) - var(residuals(mod1.3) ) ) / var(d$brain)
```

```
[1] 0.6797736
```


Overfitting

```r
mod1.4 <- lm(brain ~ mass + I(mass^2) + I(mass^3) + I(mass^4), data = d)
(var(d$brain) - var(residuals(mod1.4) ) ) / var(d$brain)
```

```
[1] 0.8144339
```

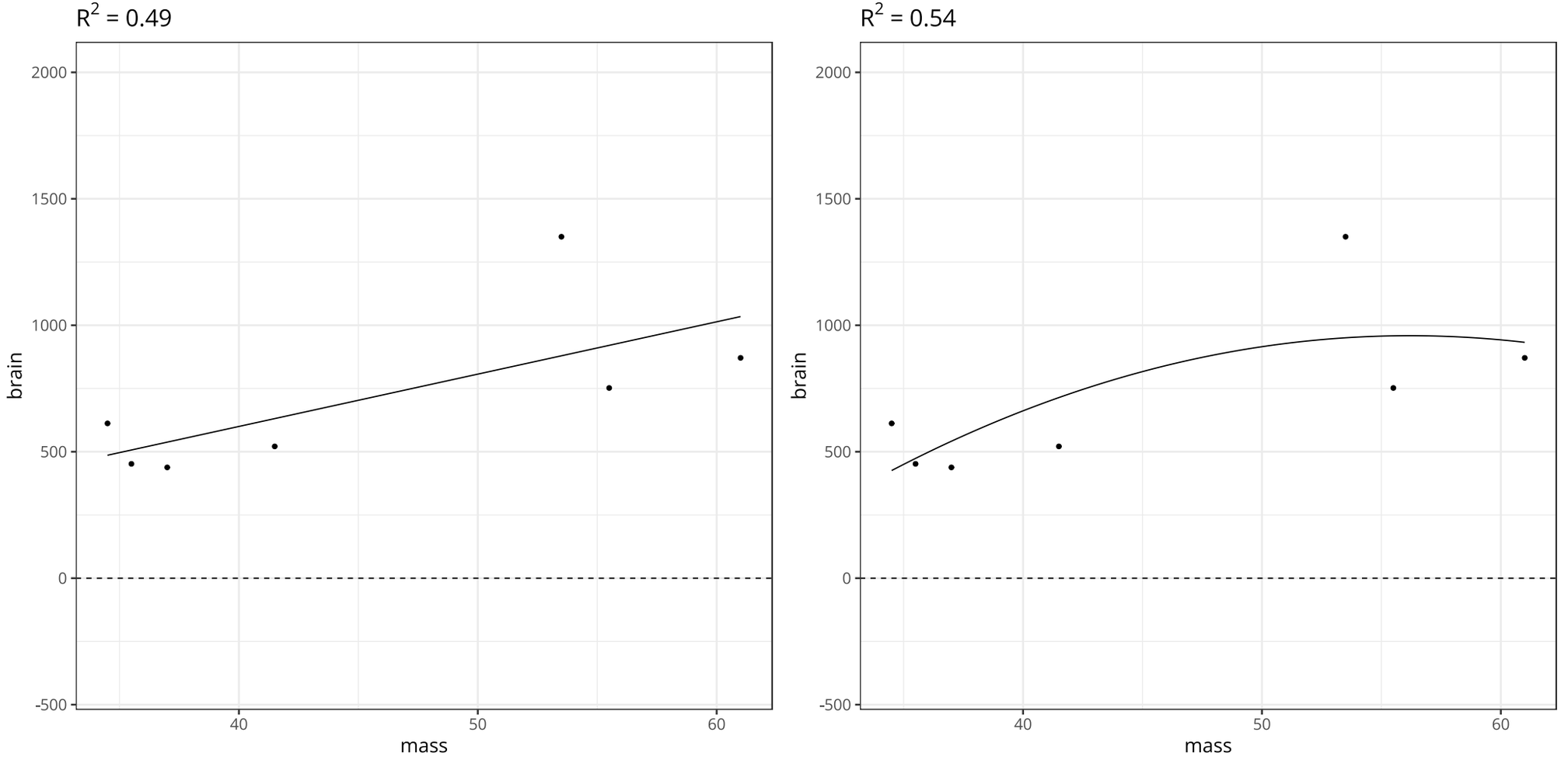
```r
mod1.5 <- lm(brain ~ mass + I(mass^2) + I(mass^3) + I(mass^4) +
    I(mass^5), data = d)
(var(d$brain) - var(residuals(mod1.5) ) ) / var(d$brain)
```

```
[1] 0.988854
```

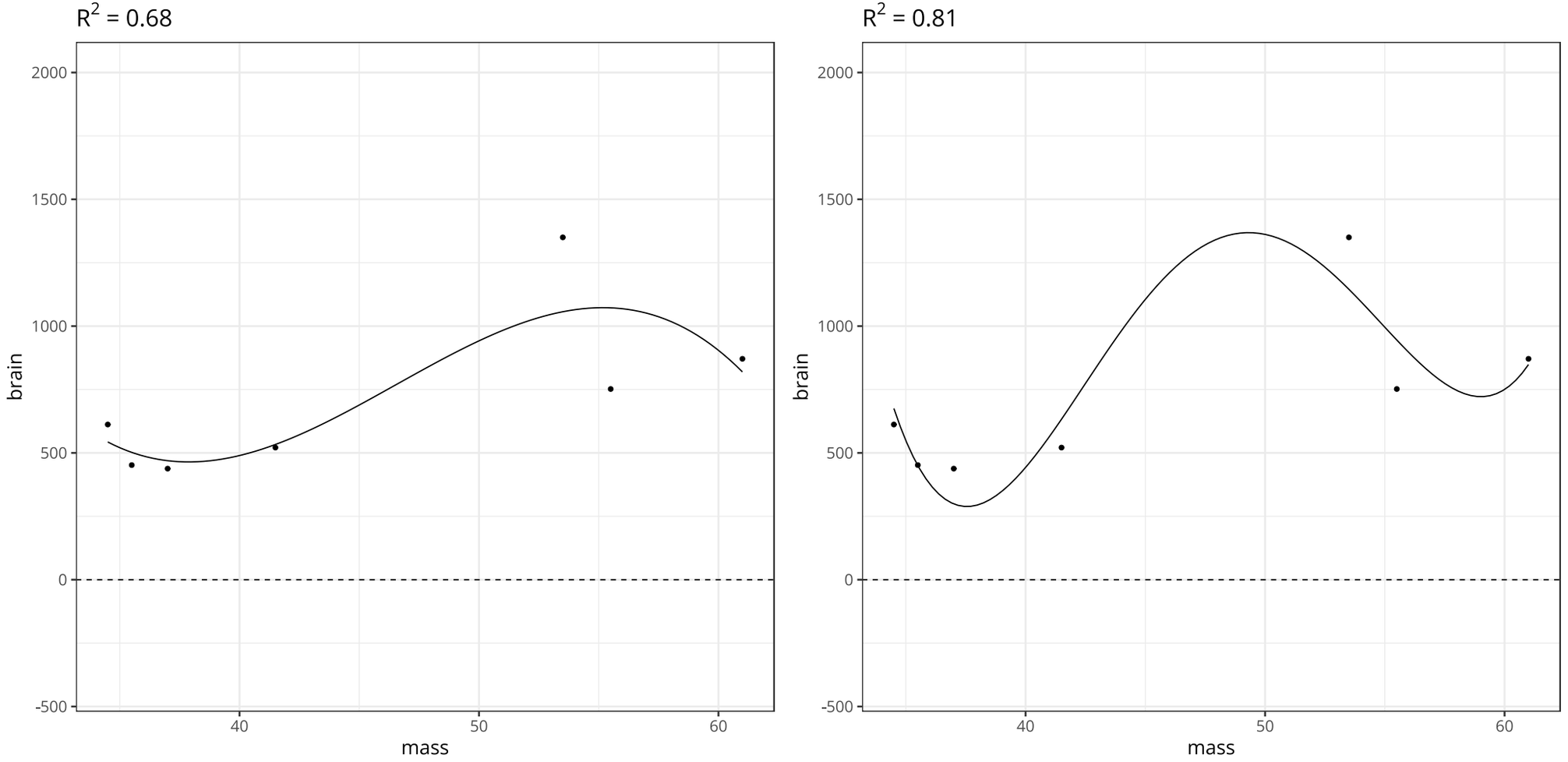
```r
mod1.6 <- lm( brain ~ mass + I(mass^2) + I(mass^3) + I(mass^4) +
    I(mass^5) + I(mass^6), data = d)
(var(d$brain) - var(residuals(mod1.6) ) ) / var(d$brain)
```

```
[1] 1
```

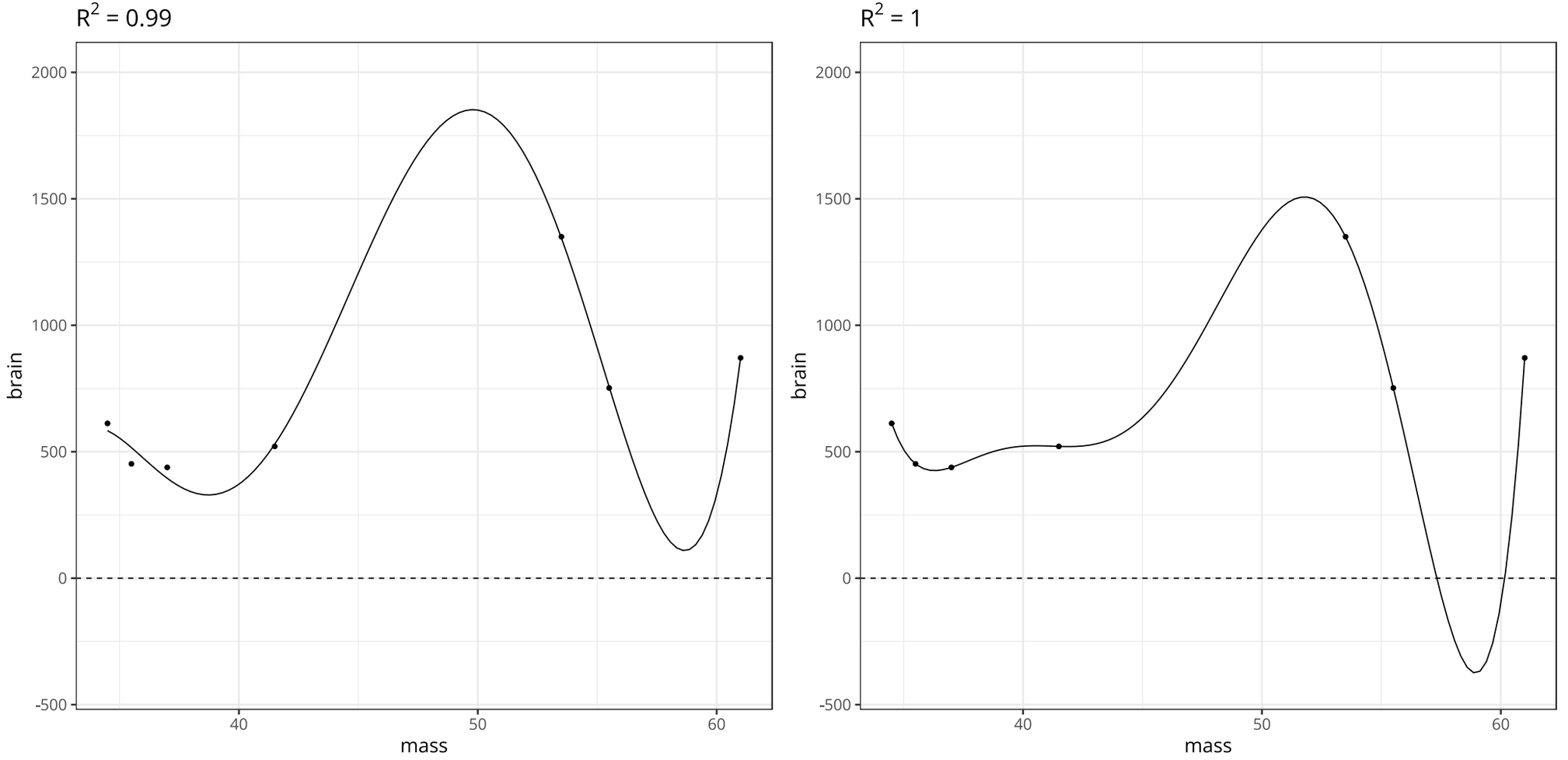

Overfitting



Overfitting



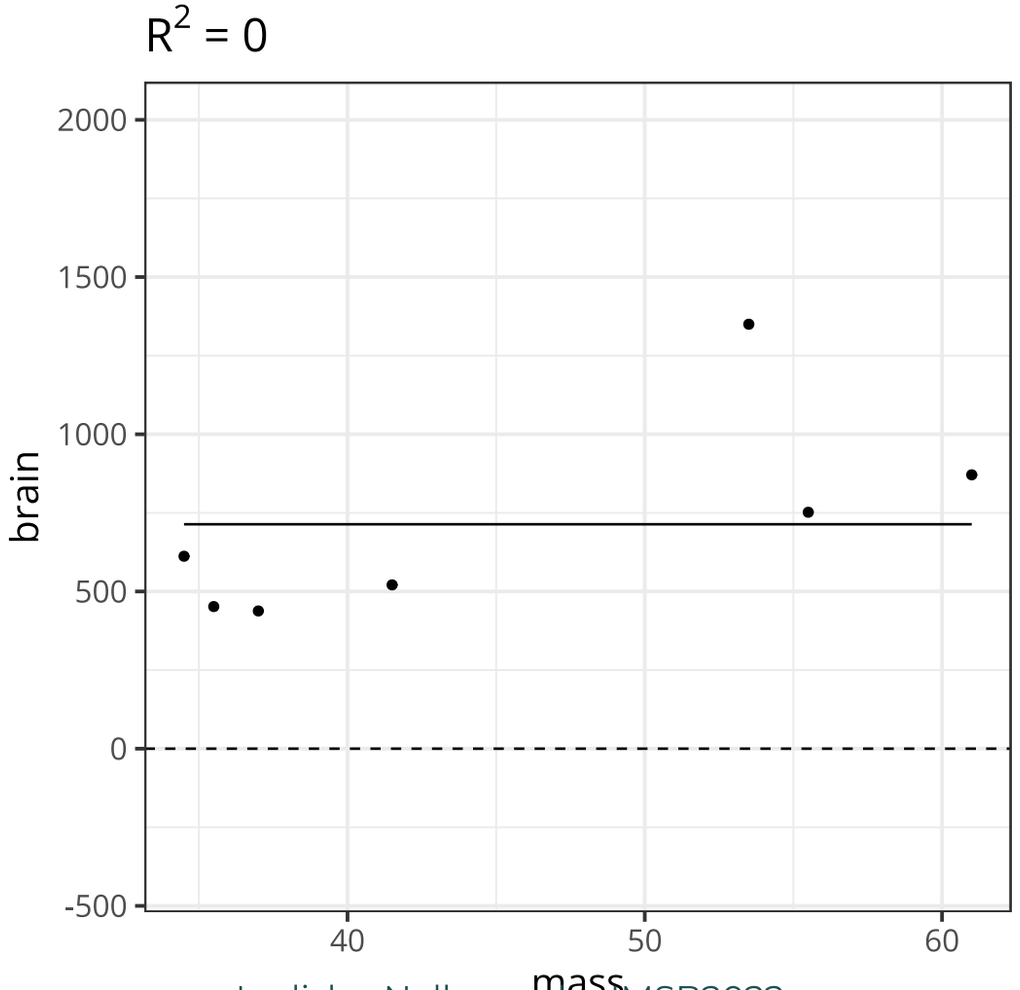
Overfitting



Underfitting

$$v_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha$$

```
1  mod1.7 <- lm(brain ~ 1, data = d)
```



Théorie de l'information

Il nous faut une autre mesure des capacités de prédiction pour évaluer nos modèles. Idéalement, on voudrait pouvoir mesurer la "distance" entre notre modèle et le "full model" (i.e., la "réalité", la nature)... mais on ne sait (encore) pas faire cela.

Par contre, on peut mesurer de combien notre incertitude est réduite en découvrant un outcome (une observation) supplémentaire. Cette réduction est la définition de l'**information**.

Mais il nous faut tout d'abord une mesure de l'incertitude (pour savoir si on l'a réduite, ou pas)... S'il existe n évènements possibles, et que chaque évènement i a pour probabilité p_i, alors une mesure de l'incertitude est donnée par l'entropie (de Shannon) H :

$$H(p) = -\mathbb{E}[\log(p_i)] = \sum_{i=1}^{n} p_i \log\left(\frac{1}{p_i}\right) = -\sum_{i=1}^{n} p_i \log(p_i)$$

> "
>
> En d'autres termes : l'incertitude contenue dans une distribution de probabilités est la log-probabilité moyenne d'un évènement.

Incertitude (exemple)

Exemple de prédiction météorologique. Si on imagine que la probabilité qu'il pleuve ou qu'il fasse beau (à Grenoble) est de $p_1 = 0.3$ et $p_2 = 0.7$.

Alors, $H(p) = -(p_1 \times \log(p_1) + p_2 \times \log(p_2)) \approx 0.61$.

```r
p <- c(0.3, 0.7) # distribution des probabilités de pluie et soleil à Grenoble
- sum(p * log(p) ) # équivalent à sum(p * log(1 / p) )
```

```
[1] 0.6108643
```

Imaginons que nous habitions à Abu Dhabi et que la probabilité qu'il y pleuve ou qu'il y fasse beau soit de $p_1 = 0.01$ et $p_2 = 0.99$.

```r
p <- c(0.01, 0.99) # distribution des probabilités de pluie et soleil à Abu Dhabi
- sum(p * log(p) ) # toujours équivalent à sum(p * log(1 / p) )
```

```
[1] 0.05600153
```


Divergence

On a donc un moyen de quantifier l'incertitude. Comment utiliser cette mesure pour quantifier la distance entre notre modèle et la réalité ?

Divergence : incertitude ajoutée par l'utilisation d'une distribution de probabilités pour décrire… une autre distribution de probabilités (Kullback-Leibler divergence, ou "entropie relative").

$$D_{\mathrm{KL}}(p, q) = \sum_i p_i \Big(\log(p_i) - \log(q_i) \Big) = \sum_i p_i \log \left(\frac{p_i}{q_i} \right)$$

La divergence est la différence moyenne en log-probabilités entre la distribution cible (p) et le modèle (q).

Divergence

$$D_{KL}(p,q) = \sum_i p_i \Big(\log(p_i) - \log(q_i) \Big) = \sum_i p_i \log \left(\frac{p_i}{q_i} \right)$$

Par exemple, supposons que la “véritable” distribution de nos évènements (soleil vs pluie) soit $p_1 = 0.3$ et $p_2 = 0.7$. Si nous pensons plutôt que ces évènements arrivent avec une probabilité $q_1 = 0.25$ et $q_2 = 0.75$, quelle quantité d’incertitude avons-nous ajoutée ?

```r
p <- c(0.3, 0.7)
q <- c(0.25, 0.75)

sum(p * log(p / q) )
```

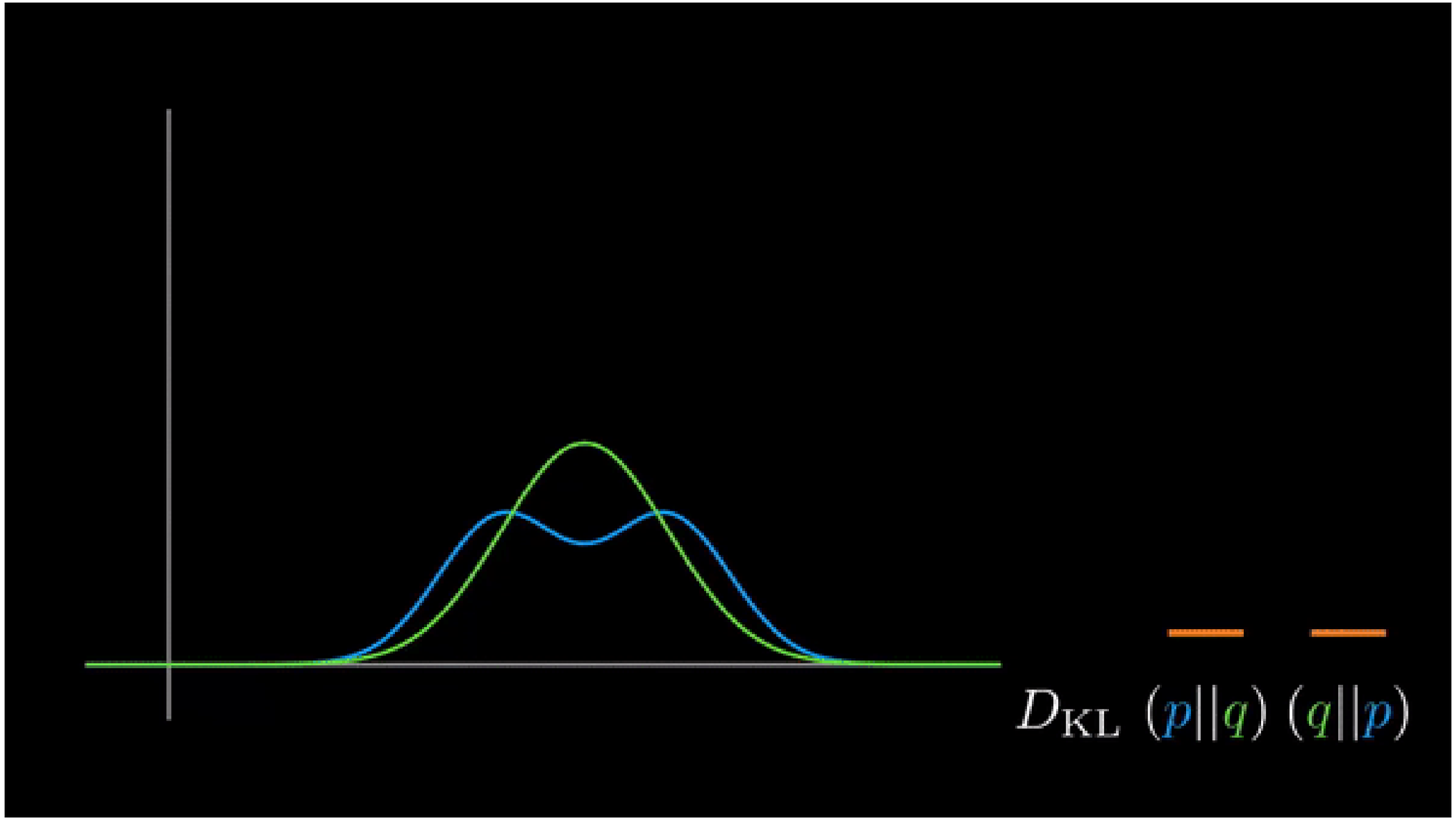
```
[1] 0.006401457
```

```r
# NB : La divergence n'est pas symétrique...
sum(q * log(q / p) )
```

```
[1] 0.006164264
```


Divergence

La divergence n’est pas symétrique (ce n’est pas une distance)…



Entropie croisée et divergence

Entropie croisée : $H(p, q) = \sum_i p_i \log(q_i)$

```
1  sum(p * (log(q) ) )
```

La **Divergence** est définie comme l'entropie additionnelle ajoutée en utilisant q pour décrire p.

$$
\begin{aligned}
D_{\mathrm{KL}}(p, q) &= H(p, q) - H(p) \\
&= -\sum_i p_i \log(q_i) - \left(-\sum_i p_i \log(p_i) \right) \\
&= -\sum_i p_i \left(\log(q_i) - \log(p_i) \right)
\end{aligned}
$$

```
1  - sum(p * (log(q) - log(p) ) )
```

```
[1] 0.006401457
```

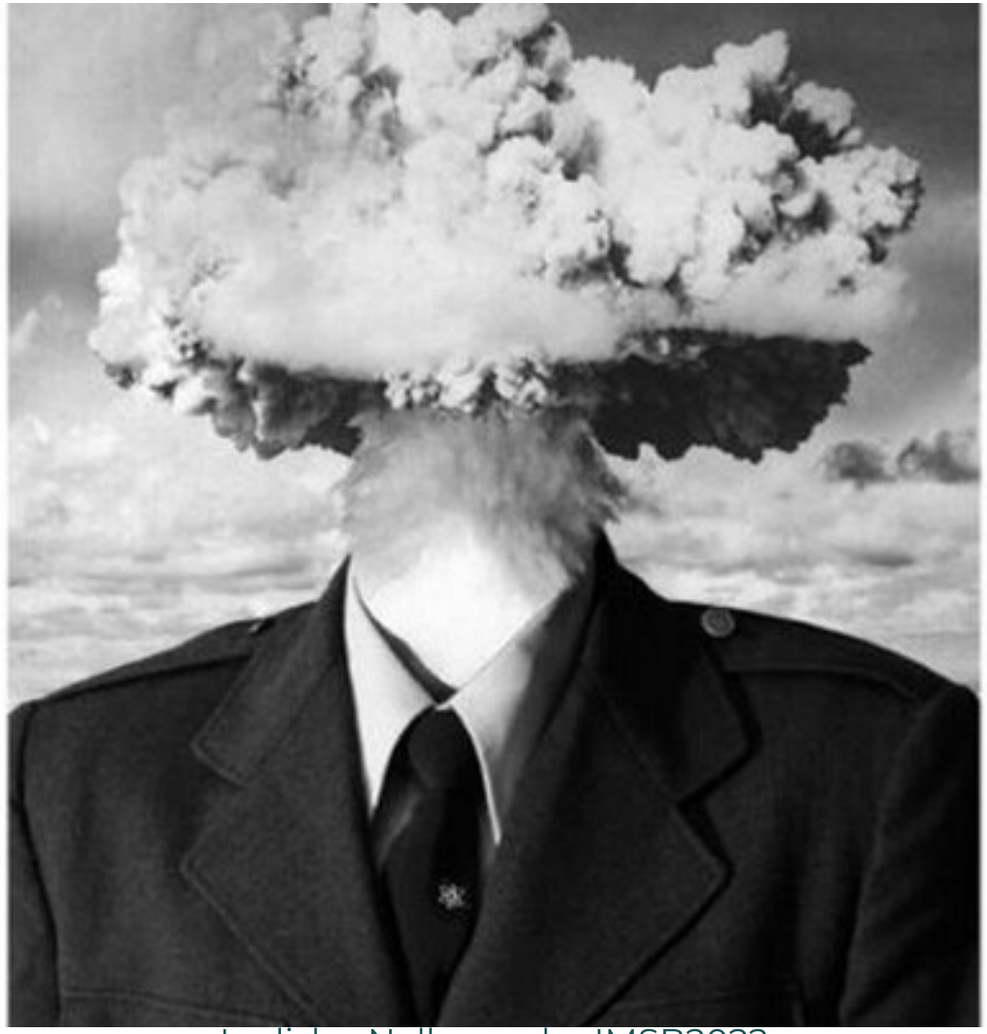
```
1  sum(p * log(p / q) )
```

```
[1] 0.006401457
```

Vers la déviance...

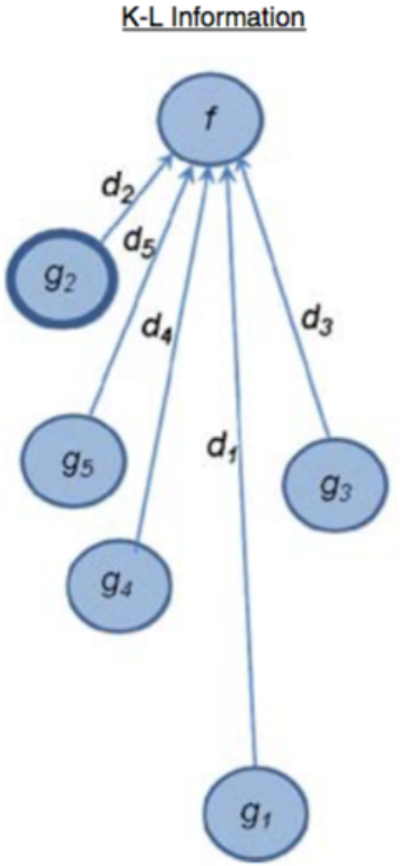
OK, mais nous ne connaissons pas la distribution “cible” (la réalité), à quoi cela peut donc nous servir ?

Astuce : si nous comparons deux modèles, q et r, pour approximer p, nous allons comparer leurs divergences... Et donc $\mathbb{E}[\log(p_i)]$ sera la même quantité pour les deux modèles !



Vers la déviance...

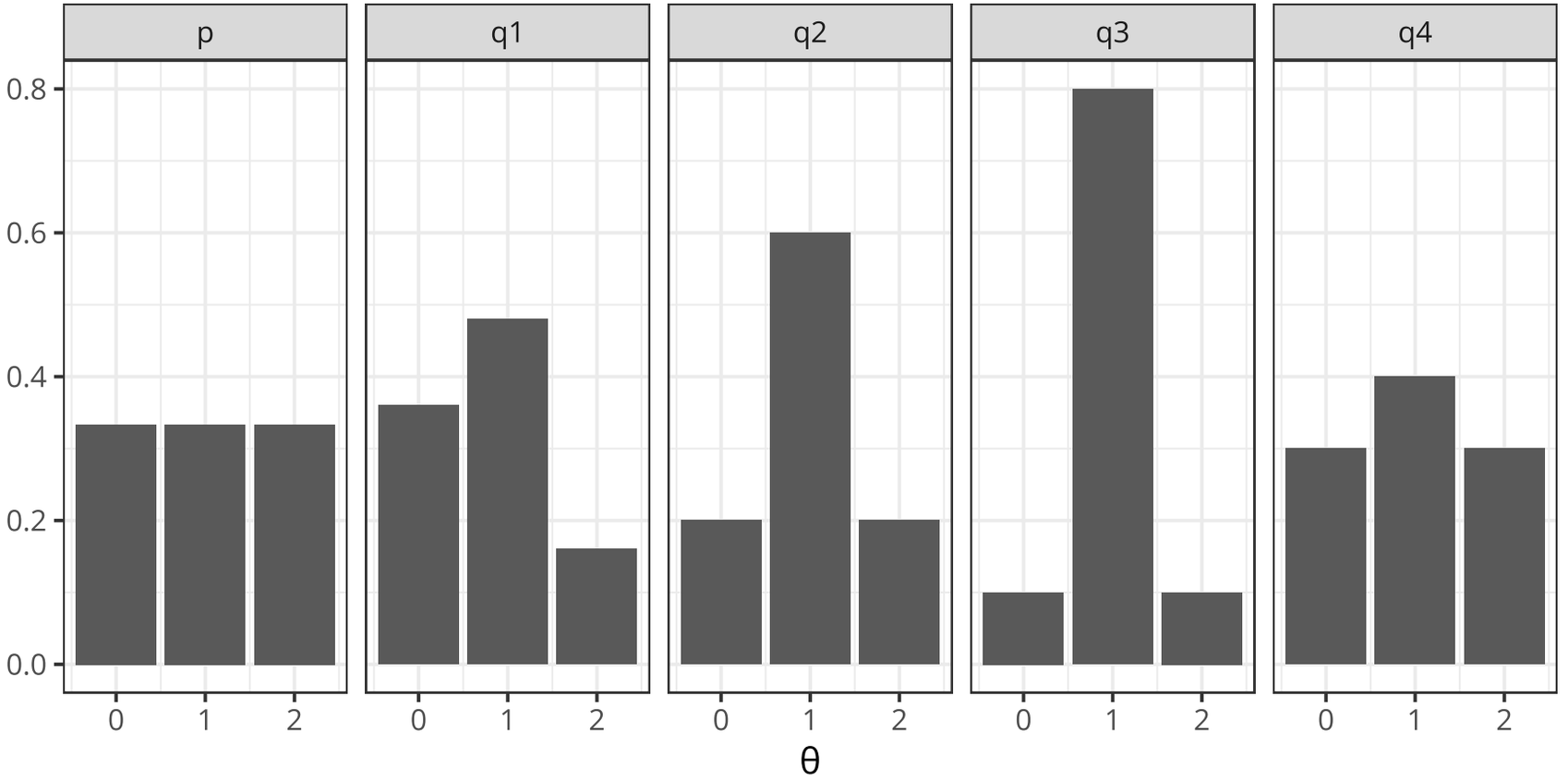
On peut donc utiliser $\mathbb{E}[\log(q_i)]$ et $\mathbb{E}[\log(r_i)]$ comme estimateurs de la distance **relative** entre chaque modèle et notre distribution cible. On a seulement besoin de la **log-probabilité moyenne des modèles**. Comme on ne connaît pas la distribution cible, cela veut dire qu'on ne peut pas interpréter cette quantité en termes absolus mais seulement en termes relatifs. Ce qui nous intéresse c'est $\mathbb{E}[\log(q_i)] - \mathbb{E}[\log(r_i)]$ (i.e., des différences de divergences).



Vers la déviance (exemple d'application)

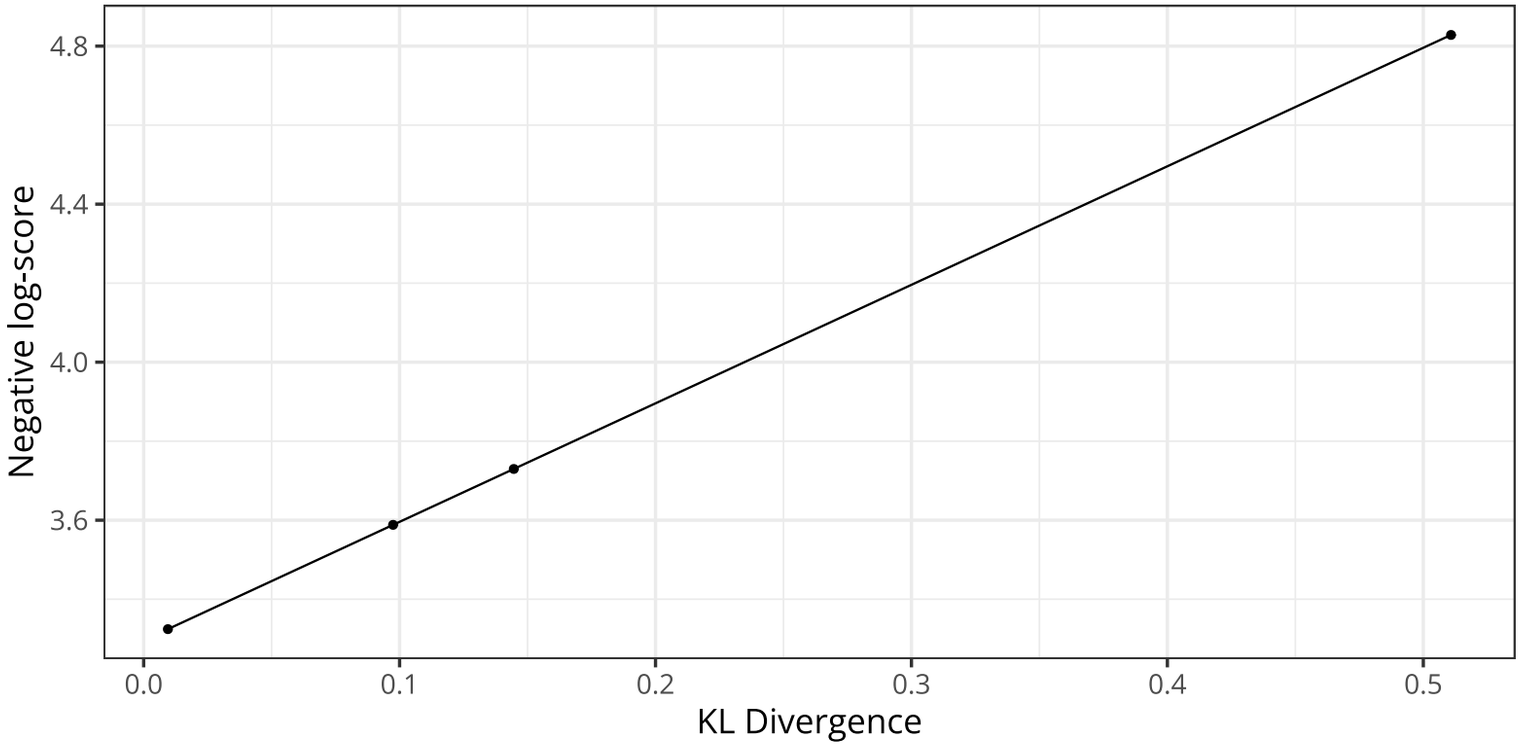
Exemple d'une distribution cible uniforme p, et 4 modèles $q1, q2, q3, q4$ (exemple tiré de cet article).

```r
p <- c(1/3, 1/3, 1/3) # distribution cible (distribution uniforme avec p = 1/3)
q1 <- c(0.36, 0.48, 0.16) # modèle q1
q2 <- c(0.2, 0.6, 0.2) # modèle q2
q3 <- c(0.1, 0.8, 0.1) # modèle q3
q4 <- c(0.3, 0.4, 0.3) # modèle q4
```



Vers la déviance (exemple d'application)

```r
divergence_q1 <- sum(p * log(p / q1) ) # divergence du modèle q1
divergence_q2 <- sum(p * log(p / q2) ) # divergence du modèle q2
divergence_q3 <- sum(p * log(p / q3) ) # divergence du modèle q3
divergence_q4 <- sum(p * log(p / q4) ) # divergence du modèle q4

# vecteur de divergences
divergences <- c(divergence_q1, divergence_q2, divergence_q3, divergence_q4)

# vecteur de negative log-scores
neg_log_scores <- c(-sum(log(q1) ), -sum(log(q2) ), -sum(log(q3) ), -sum(log(q4) ) )
```



Déviance

Pour approximer la valeur de $\mathbb{E}[\log(q_i)]$, on peut utiliser la déviance d'un modèle, qui est une mesure du fit **relatif** du modèle.

$$D(q) = -2 \sum_i \log(q_i)$$

où i indice chaque observation et q_i est la **vraisemblance** de chaque observation.

```r
# on standardise la variable "mass"
d$mass.s <- scale(d$mass)

# on "fit" un modèle de régression linéaire Gaussien
mod1.8 <- lm(formula = brain ~ mass.s, data = d)

# calcul de la déviance
-2 * logLik(mod1.8)
```

```
'log Lik.' 94.92499 (df=3)
```


Déviance

```r
# on récupère les paramètres estimés via MLE (intercept et pente)
alpha <- coef(mod1.8)[1]
beta <- coef(mod1.8)[2]

# calcul de la log-vraisemblance
ll <- sum(dnorm(
    x = d$brain,
    mean = alpha + beta * d$mass.s,
    sd = sigma(mod1.8),
    log = TRUE
    ) )

# calcul de la déviance
(-2) * ll
```

```
[1] 95.2803
```


Log-pointwise-predictive density

Dans le monde fréquentiste, on multiplie le log-score par -2 car la différence de deux déviances suit une loi de χ^2, ce qui est utile pour tester l'hypothèse nulle. Mais sans besoin de tester l'hypothèse nulle (avec une forme prédéfinie), on peut très bien travailler directement avec le log-score $S(q) = \sum_i \log(q_i)$, qu'on traite comme une estimation de $\mathbb{E}[\log(q_i)]$.

On peut calculer $S(q)$ sur toute la distribution postérieure, ce qui donne la version bayésienne du log-score, la **log-pointwise-predictive density** :

$$\text{lppd}(y, \Theta) = \sum_i \log \frac{1}{S} \sum_s p(y_i \mid \Theta_s)$$

Où S est le nombre d'échantillons et Θ_s est le s-ième ensemble de valeurs de paramètres échantillonnés de la distribution postérieure.

In-sample and out-of-sample

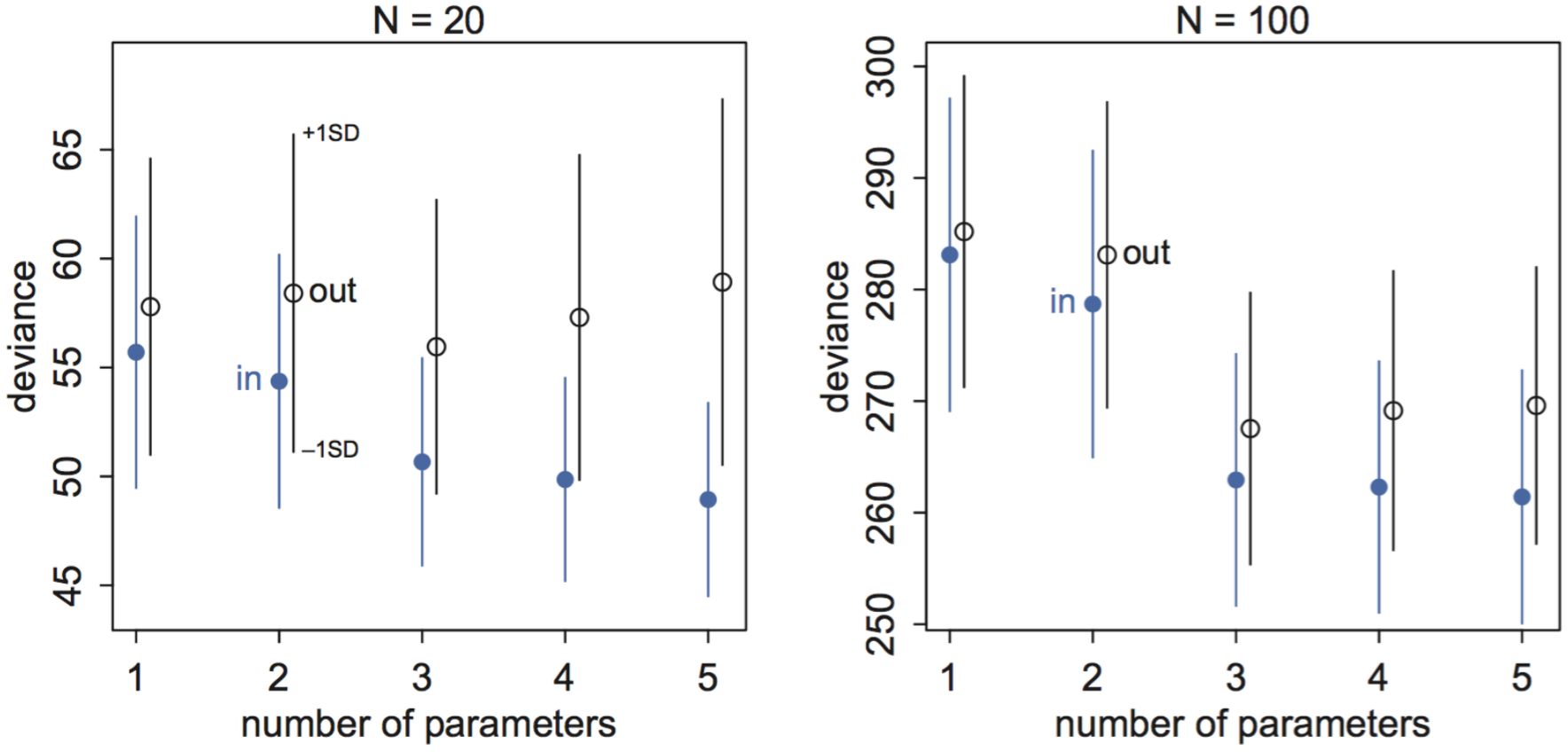
La déviance a le même problème que le R^2, lorsqu'elle est calculée sur l'échantillon observé. Dans ce cas, on l'appelle déviance **in-sample**.

Si on est intéressé par les capacités de prédiction de notre modèle, nous pouvons calculer la déviance du modèle sur de nouvelles données... qu'on appellera dans ce cas déviance **out-of-sample**. Cela revient à se demander si notre modèle est performant pour prédire de nouvelles données (non observées).

Imaginons que nous disposions d'un échantillon de taille N, que nous appellerons échantillon d'apprentissage (training). Nous pouvons calculer la déviance du modèle sur cet échantillon (D_{train} ou D_{in}). Si nous acquérons ensuite un nouvel échantillon de taille N issu du même processus de génération de données (que nous appellerons échantillon de test), nous pouvons calculer une déviance sur ce nouvel échantillon, en utilisant les paramètres estimés avec l'échantillon d'entraînement (que nous appellerons D_{test} ou D_{out}).

In sample and out of sample deviance

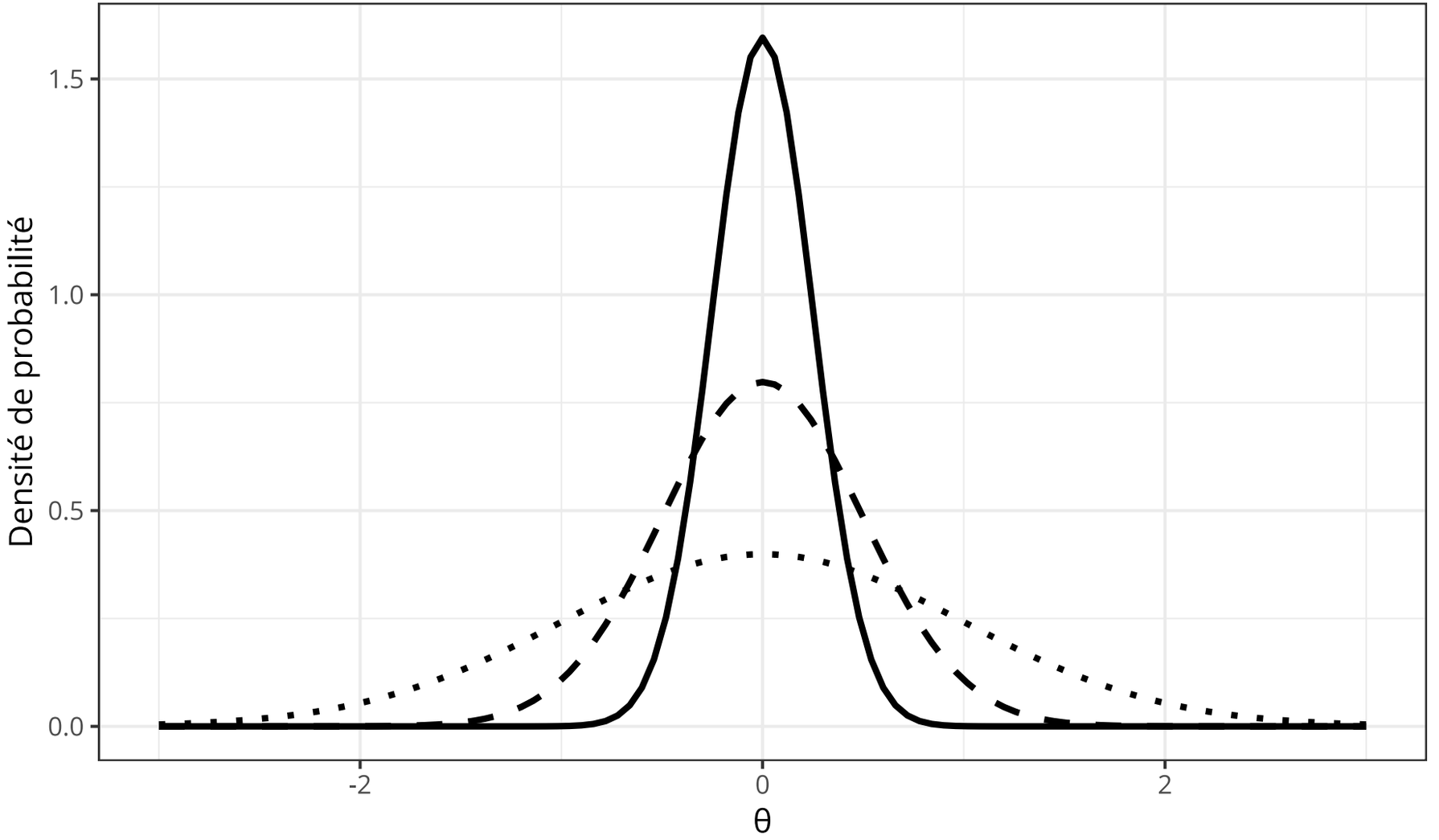
$$y_i \sim \text{Normal}(\mu_i, 1)$$
$$\mu_i = (0.15)x_{1,i} - (0.4)x_{2,i}$$



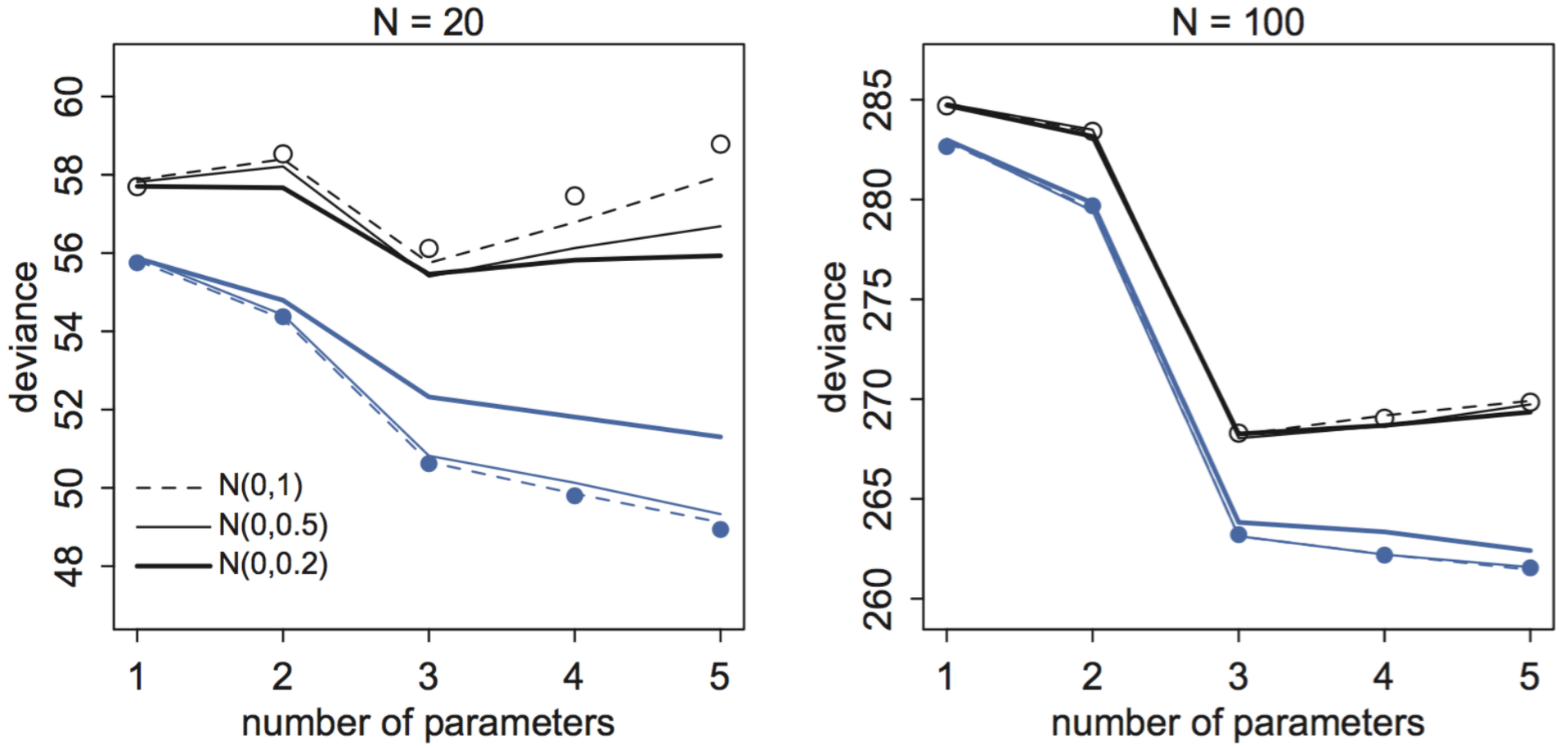
On a réalisé ce processus 10.000 fois pour cinq modèles de régression linéaire de complexité croissante. Les points bleus représentent la déviance calculée sur l'échantillon d'apprentissage et les points noirs la déviance calculée sur l'échantillon de test.

Régularisation

Une autre manière de lutter contre le sur-apprentissage (overfitting) est d'utiliser des priors "sceptiques" qui vont venir ralentir l'apprentissage réalisé sur les données (i.e., accorder plus de poids au prior).

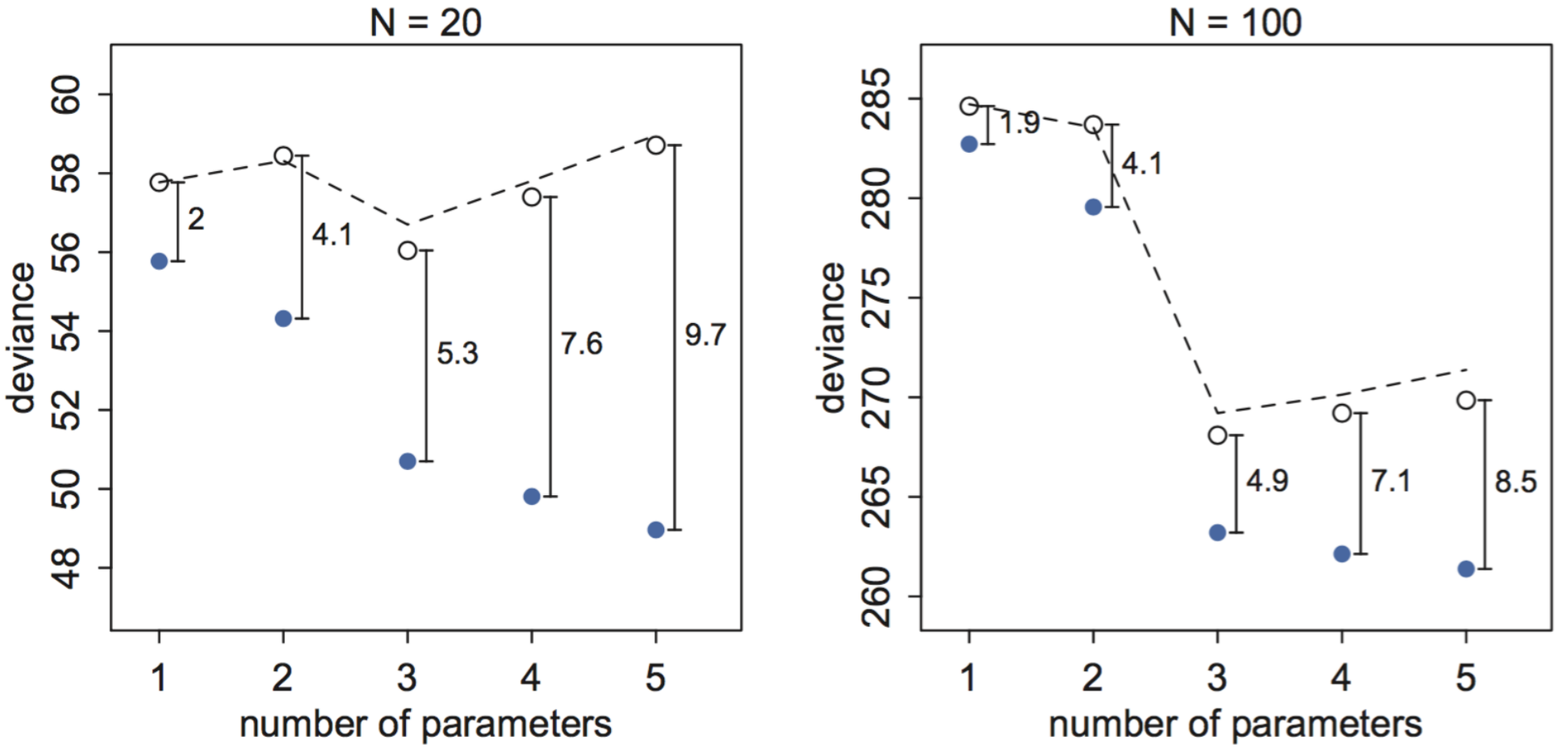


Régularisation



Comment décider de la précision du prior ? Est-ce que le prior est “assez” régularisateur ou pas ? On peut diviser le jeu de données en deux parties (training et testing) afin de choisir le prior qui produit la déviance **out-of-sample** la plus faible. On appelle cette stratégie la **validation croisée** (cross-validation).

Critères d'information



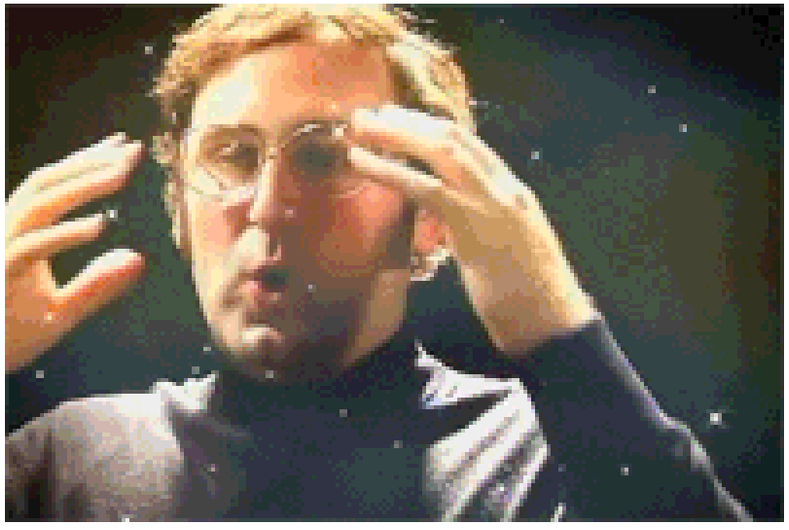
On mesure ici la différence entre la déviance **in-sample** (en bleu) et la déviance **out-of-sample** (en noir). On remarque que la déviance **out-of-sample** est presque exactement égale à la déviance **in-sample**, plus deux fois le nombre de paramètres du modèle...

Akaike information criterion

L'AIC fournit une approximation de la déviance **out-of-sample** :

$$\text{AIC} = D_{\text{train}} + 2p = -2\text{lppd} + 2p \approx D_{\text{test}}$$

où p est le nombre de paramètres libres (i.e., à estimer) dans le modèle. L'AIC donne donc une approximation des capacités de prédiction (out-of-sample) du modèle.[1]



1. NB : l'AIC fonctionne bien uniquement quand le nombre d'observations N est largement supérieur au nombre de paramètres p. Dans le cas contraire, on utilise plutôt l'AICc (Burnham & Anderson, 2004,

Deviance information criterion

Une condition d'application de l'AIC est que les priors soient plats ou dépassés par la vraisemblance (e.g., lorsqu'on a beaucoup de données). Le DIC est un indice qui ne requiert pas cette condition, en s'accommodant de priors informatifs.

Le DIC est calculé à partir de la distribution a posteriori de la déviance D calculée sur l'échantillon d'apprentissage (i.e., D_{train}).

$$\text{DIC} = \bar{D} + (\bar{D} - \hat{D}) = \bar{D} + p_D$$

où $\bar{D}$ est la moyenne de la distribution a posteriori D calculée pour chaque valeur de paramètre échantillonnée, et $\hat{D}$ la déviance calculée à la moyenne de la distribution a posteriori. La différence $\bar{D} - \hat{D} = p_D$ est analogue au nombre de paramètres utilisé dans le calcul de l'AIC (en cas de prior plat, cette différence revient à compter le nombre de paramètres).

Widely applicable information criterion

Une condition d'application de l'AIC et du DIC est que la distribution a posteriori soit une distribution gaussienne multivariée. Le WAIC relâche cette condition et il est souvent plus précis que le DIC.

Un aspect important du WAIC est qu'il est dit "pointwise", c'est à dire qu'il considère l'imprécision de prédiction point par point (donnée par donnée), indépendamment pour chaque observation.

On va commencer par calculer la **log-pointwise-predictive-density** (lppd), définie de la manière suivante :

$$\text{lppd}(y, \Theta) = \sum_i \log \frac{1}{S} \sum_s p(y_i \mid \Theta_s)$$

En Français : la log-densité prédictive point par point est **la somme du log de la vraisemblance moyenne de chaque observation**. Il s'agit de l'analogue point par point de la déviance, moyennée sur toute la distribution postérieure.

Widely applicable information criterion

```
library(brms)
data(cars)

priors <- c(
  prior(normal(0, 100), class = Intercept),
  prior(normal(0, 10), class = b),
  prior(exponential(0.1), class = sigma)
  )

mod1 <- brm(
  formula = dist ~ 1 + speed,
  prior = priors,
  data = cars
  )
```


Widely applicable information criterion

$$\text{lppd}(y, \Theta) = \sum_i \log \frac{1}{S} \sum_s p(y_i \mid \Theta_s)$$

```r
# pointwise log-likelihood (S samples * N observations)
ll <- log_lik(mod1) %>% data.frame()

(lppd <-
  ll %>%
  pivot_longer(
    cols = everything(),
    names_to = "i",
    values_to = "loglikelihood"
    ) %>%
  # log-likelihood to likelihood
  mutate(likelihood = exp(loglikelihood) ) %>%
  # pour chaque observation
  group_by(i) %>%
  # logarithme de la vraisemblance moyenne
  summarise(log_mean_likelihood = log(mean(likelihood) ) ) %>%
  # on prend la somme de ces valeurs
  summarise(lppd = sum(log_mean_likelihood) ) %>%
  ungroup() %>%
  pull(lppd) )
```

```
[1] -206.6047
```


Widely applicable information criterion

La deuxième partie du calcul du WAIC est le nombre de paramètres effectif, p_{WAIC} . On définit $\text{var}_\theta \log[p(y_i|\theta)]$ comme la variance de la log-vraisemblance pour chaque observation i de l'échantillon d'entraînement.

$$p_{\text{WAIC}} = \sum_i \text{var}_\theta \log[p(y_i \mid \theta)]$$

```r
(pwaic <-
  ll %>%
  pivot_longer(
    everything(),
    names_to = "i",
    values_to = "loglikelihood"
    ) %>%
  # pour chaque observation
  group_by(i) %>%
  # variance de la log-vraisemblance
  summarise(var_loglikelihood = var(loglikelihood) ) %>%
  # somme de ces variances
  summarise(pwaic = sum(var_loglikelihood) ) %>%
  ungroup() %>%
  pull(pwaic) )
```

```
[1] 3.342853
```


Widely applicable information criterion

Ensuite, le WAIC est défini par :

$$\text{WAIC}(y, \Theta) = -2\Big(\text{lppd} - \underbrace{\sum_i \text{var}_\theta \log[p(y_i \mid \theta)]}_{\text{penalty term}}\Big)$$

```
(WAIC <- -2 * (lppd - pwaic) )
```

```
[1] 419.8952
```

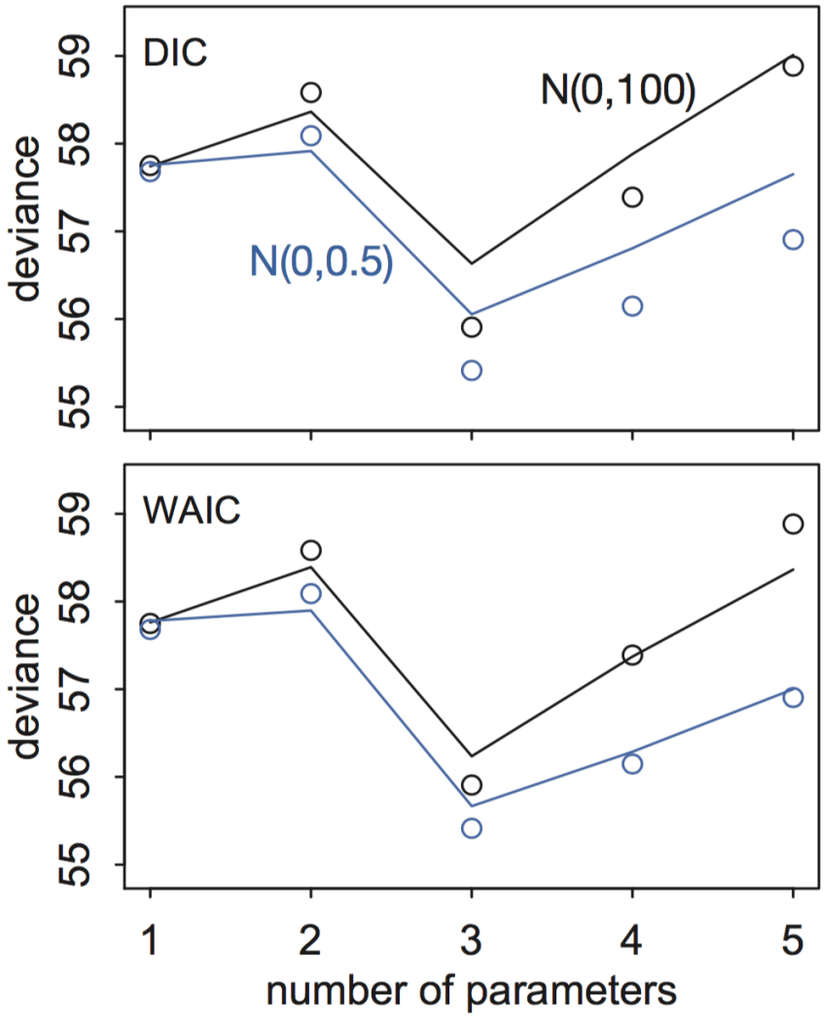
Widely applicable information criterion

Le WAIC est également un estimateur de la déviance **out-of-sample**. La fonction `brms::waic()` permet de le calculer directement.

```
1 waic(mod1)
```

```
Computed from 4000 by 50 log-likelihood matrix

          Estimate   SE
elpd_waic   -209.9  6.5
p_waic         3.3  1.2
waic         419.9 13.0

2 (4.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

Critères d'information et régularisation



Le DIC et le WAIC peuvent être conceptualisés (au même titre que l'AIC) comme des approximations de la déviance **out-of-sample**. On remarque que le WAIC produit des approximations plus précises que le DIC, et que l'utilisation de priors régularisateurs permet de réduire la déviance **out-of-sample**.

Et ensuite ?

Sélection de modèle : On choisit le meilleur modèle un utilisant un des outils présentés et on base nos conclusions sur les paramètres estimés par ce meilleur modèle.

Comparaison de modèles : On utilise la validation croisée ou des critères d'informations mais aussi les outils de **posterior predictive checking** discutés précédemment, pour chaque modèle, afin d'étudier leurs forces et faiblesses.

Moyennage de modèles : On va construire des posterior predictive checks qui exploitent ce qu'on sait des capacités de prédiction de chaque modèle (e.g., via le WAIC).

Comparaison de modèles

On essaye de prédire les kg par gramme de lait (`kcal.per.g`) avec les prédicteurs `neocortex` et le logarithme de `mass`. Nous allons ensuite fitter 4 modèles qui correspondent aux 4 combinaisons possibles de prédicteurs et les comparer en utilisant le WAIC.

```r
library(imsb)
d <- open_data(milk)
d <- milk[complete.cases(milk), ] # removing NAs
d$neocortex <- d$neocortex.perc / 100 # rescaling explanatory variable
head(d)
```

```
              clade            species kcal.per.g perc.fat perc.protein
1     Strepsirrhine     Eulemur fulvus       0.49    16.60        15.42
6  New World Monkey Alouatta seniculus       0.47    21.22        23.58
7  New World Monkey         A palliata       0.56    29.66        23.46
8  New World Monkey       Cebus apella       0.89    53.41        15.80
10 New World Monkey         S sciureus       0.92    50.58        22.33
11 New World Monkey   Cebuella pygmaea       0.80    41.35        20.85
   perc.lactose mass neocortex.perc neocortex
1         67.98 1.95          55.16    0.5516
6         55.20 5.25          64.54    0.6454
7         46.88 5.37          64.54    0.6454
8         30.79 2.51          67.64    0.6764
10        27.09 0.68          68.85    0.6885
11        37.80 0.12          58.85    0.5885
```


Comparaison de modèles

```
mod2.1 <- brm(
  formula = kcal.per.g ~ 1,
  family = gaussian,
  data = d,
  prior = c(
    prior(normal(0, 100), class = Intercept),
    prior(exponential(0.01), class = sigma)
    ),
  iter = 2000, warmup = 1000,
  backend = "cmdstanr" # on peut changer le backend de brms
  )

mod2.2 <- brm(
  formula = kcal.per.g ~ 1 + neocortex,
  family = gaussian,
  data = d,
  prior = c(
    prior(normal(0, 100), class = Intercept),
    prior(normal(0, 10), class = b),
    prior(exponential(0.01), class = sigma)
    ),
  iter = 2000, warmup = 1000,
  backend = "cmdstanr"
  )
```

Comparaison de modèles

```
mod2.3 <- brm(
  formula = kcal.per.g ~ 1 + log(mass),
  family = gaussian,
  data = d,
  prior = c(
    prior(normal(0, 100), class = Intercept),
    prior(exponential(0.01), class = sigma)
    ),
  iter = 2000, warmup = 1000,
  backend = "cmdstanr"
  )

mod2.4 <- brm(
  formula = kcal.per.g ~ 1 + neocortex + log(mass),
  family = gaussian,
  data = d,
  prior = c(
    prior(normal(0, 100), class = Intercept),
    prior(normal(0, 10), class = b),
    prior(exponential(0.01), class = sigma)
    ),
  iter = 2000, warmup = 1000,
  backend = "cmdstanr"
  )
```


Comparaison de modèles

On peut utiliser la méthode `update()` qui permet de fitter plus rapidement un nouveau modèle qui ressemble à un modèle déjà existant.

```
mod2.3 <- update(
  object = mod2.2,
  newdata = d,
  formula = kcal.per.g ~ 1 + log(mass)
  )

mod2.4 <- update(
  object = mod2.3,
  newdata = d,
  formula = kcal.per.g ~ 1 + neocortex + log(mass)
  )
```


Comparaison de modèles

```r
# calcul du WAIC et ajout du WAIC à chaque modèle

mod2.1 <- add_criterion(mod2.1, "waic")
mod2.2 <- add_criterion(mod2.2, "waic")
mod2.3 <- add_criterion(mod2.3, "waic")
mod2.4 <- add_criterion(mod2.4, "waic")

# comparaison des WAIC de chaque modèle

w <- loo_compare(mod2.1, mod2.2, mod2.3, mod2.4, criterion = "waic")
print(w, simplify = FALSE)
```

```
       elpd_diff se_diff elpd_waic se_elpd_waic p_waic se_p_waic waic  se_waic
mod2.4    0.0       0.0     8.4       2.6          3.1    0.8     -16.8   5.1
mod2.3   -3.9       1.7     4.5       2.1          2.0    0.4      -9.0   4.2
mod2.1   -4.0       2.4     4.4       1.9          1.3    0.3      -8.8   3.7
mod2.2   -4.8       2.5     3.6       1.6          1.9    0.3      -7.2   3.3
```

Akaike's weights

Le poids d'un modèle est une estimation de la probabilité que ce modèle fera les meilleures prédictions possibles sur un nouveau jeu de données, conditionnellement au set de modèles considéré.

$$w_i = \frac{\exp\left(-\frac{1}{2}\text{dWAIC}_i\right)}{\sum_{j=1}^{m}\exp\left(-\frac{1}{2}\text{dWAIC}_j\right)}$$

Cette fonction permet simplement de passer du WAIC à une probabilité (équivalent à une fonction softmax). Le modèle `mod2.4` a un poids de 0.96, qui le place en tête du jeu de modèles. N'oublions cependant pas que nous disposons seulement de 17 observations...

```
model_weights(mod2.1, mod2.2, mod2.3, mod2.4, weights = "waic") %>% round(digits = 3)
```

```
mod2.1 mod2.2 mod2.3 mod2.4
 0.017  0.008  0.019  0.955
```


Moyennage de modèles (model averaging)

Pourquoi ne conserver uniquement le premier modèle et oublier les autres ? Une autre stratégie consisterait à pondérer les prédictions des modèles par leurs poids respectifs. C'est ce qu'on appelle le moyennage de modèles (model averaging).

- Calculer le WAIC de chaque modèle.
- Calculer le poids de chaque modèle.
- Simuler des données à partir de chaque modèle.
- Combiner ces valeurs simulées dans un **ensemble** de prédictions pondérées par le poids du modèle.

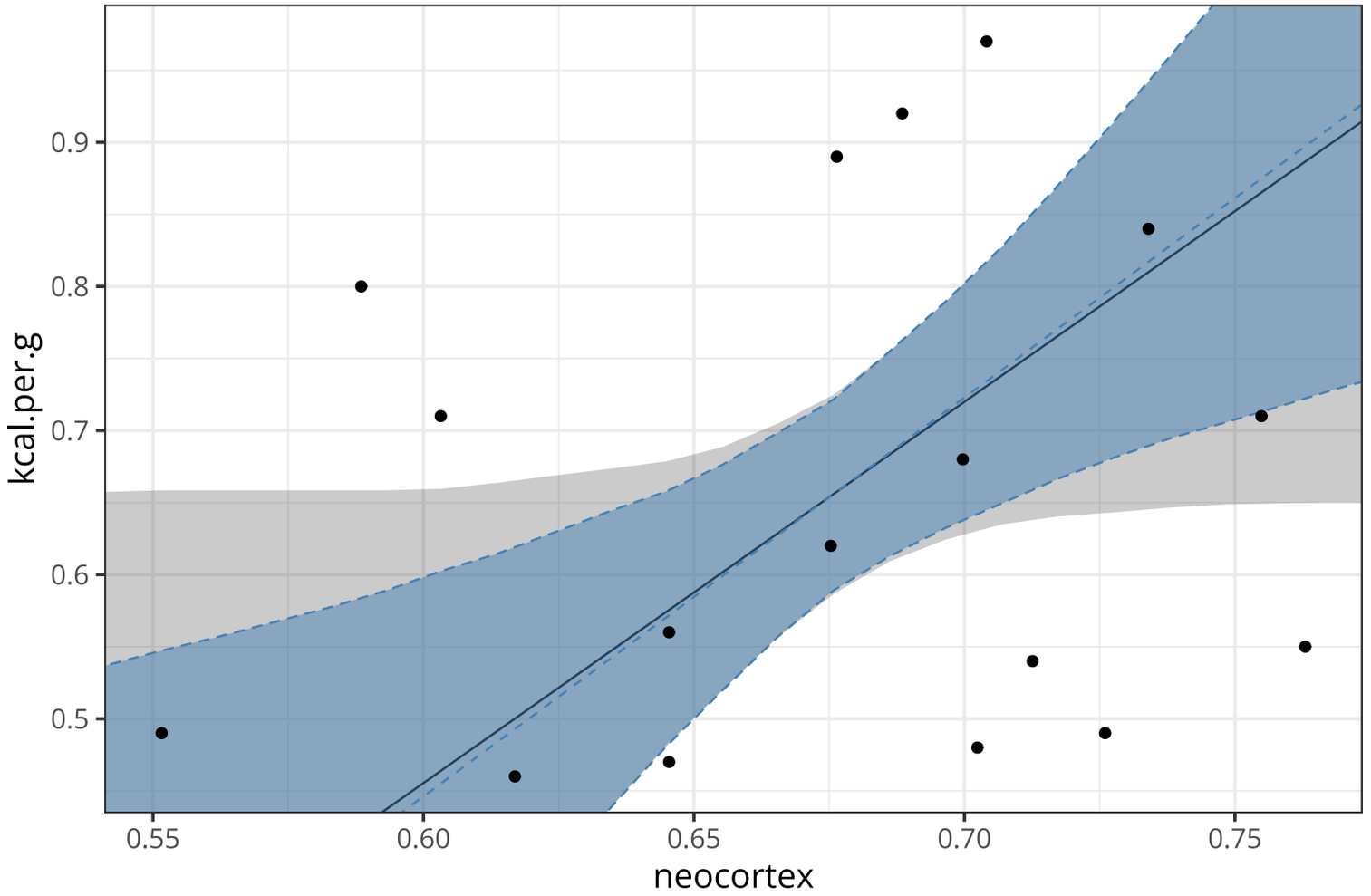
Moyennage de modèles (model averaging)

On peut utiliser la fonction `brms::pp_average()` qui pondère les prédictions de chaque modèle par leur poids.

```r
# grille de valeurs pour lesquelles on va générer des prédictions
new_data <- data.frame(
  neocortex = seq(from = 0.5, to = 0.8, length.out = 30),
  mass = 4.5
  )

# prédictions du modèle mod2.4
f <- fitted(mod2.4, newdata = new_data) %>%
  as.data.frame() %>%
  bind_cols(new_data)

# prédictions moyennées sur les 4 modèles
averaged_predictions <- pp_average(
  mod2.1, mod2.2, mod2.3, mod2.4,
  weights = "waic",
  method  = "fitted",
  newdata = new_data
  ) %>%
  as.data.frame() %>%
  bind_cols(new_data)
```


Moyennage de modèles (model averaging)

Voici les prédictions de tous les modèles considérés, pondérés par leur poids respectif. Comme le modèle `mod2.4` concentrait quasiment tout le poids, il fait sens que cette prédiction moyennée soit similaire aux prédictions du modèle `mod2.4`.

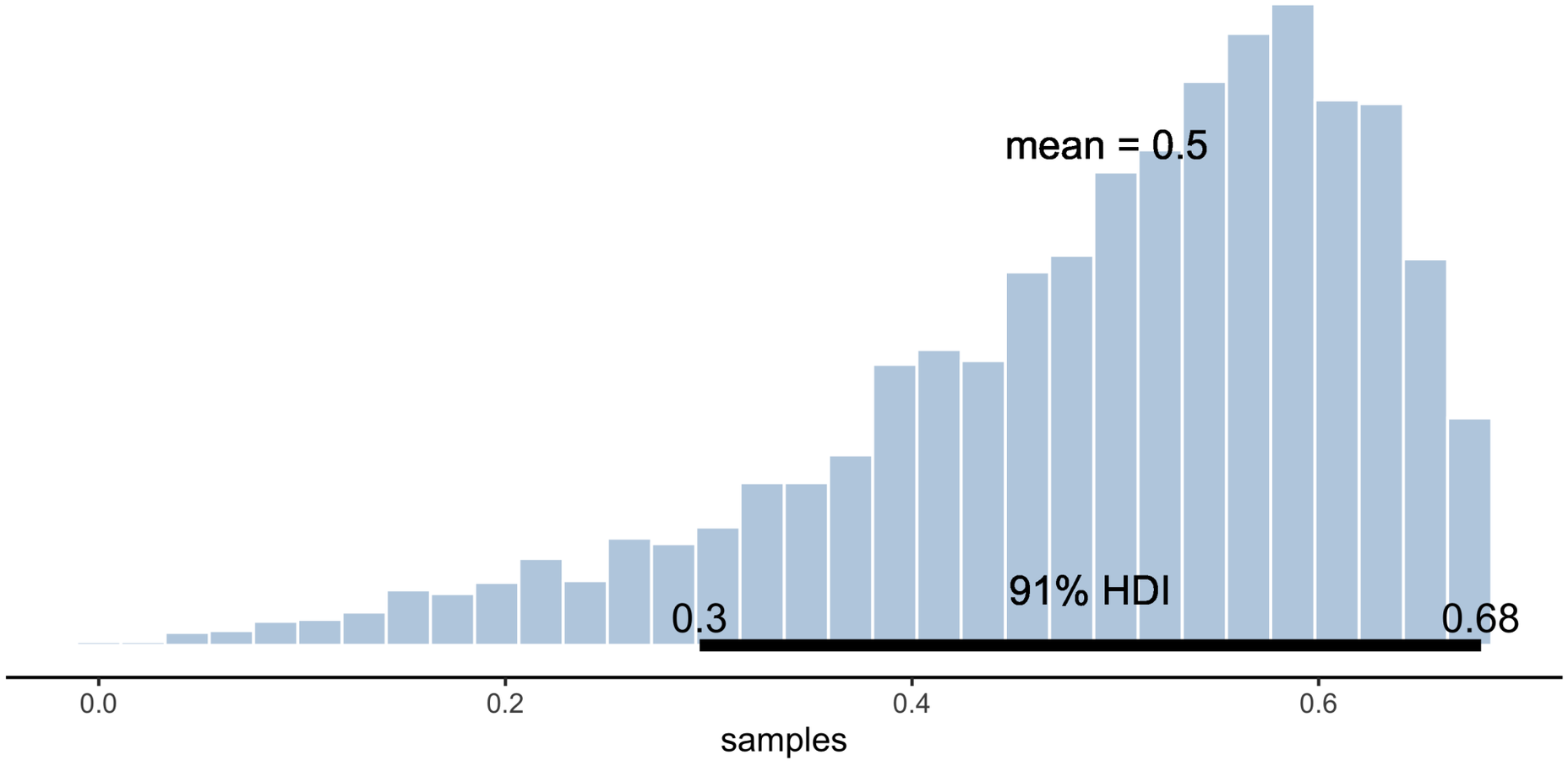


R-squared

```
1  bayes_R2(mod2.4) %>% round(digits = 3)
```

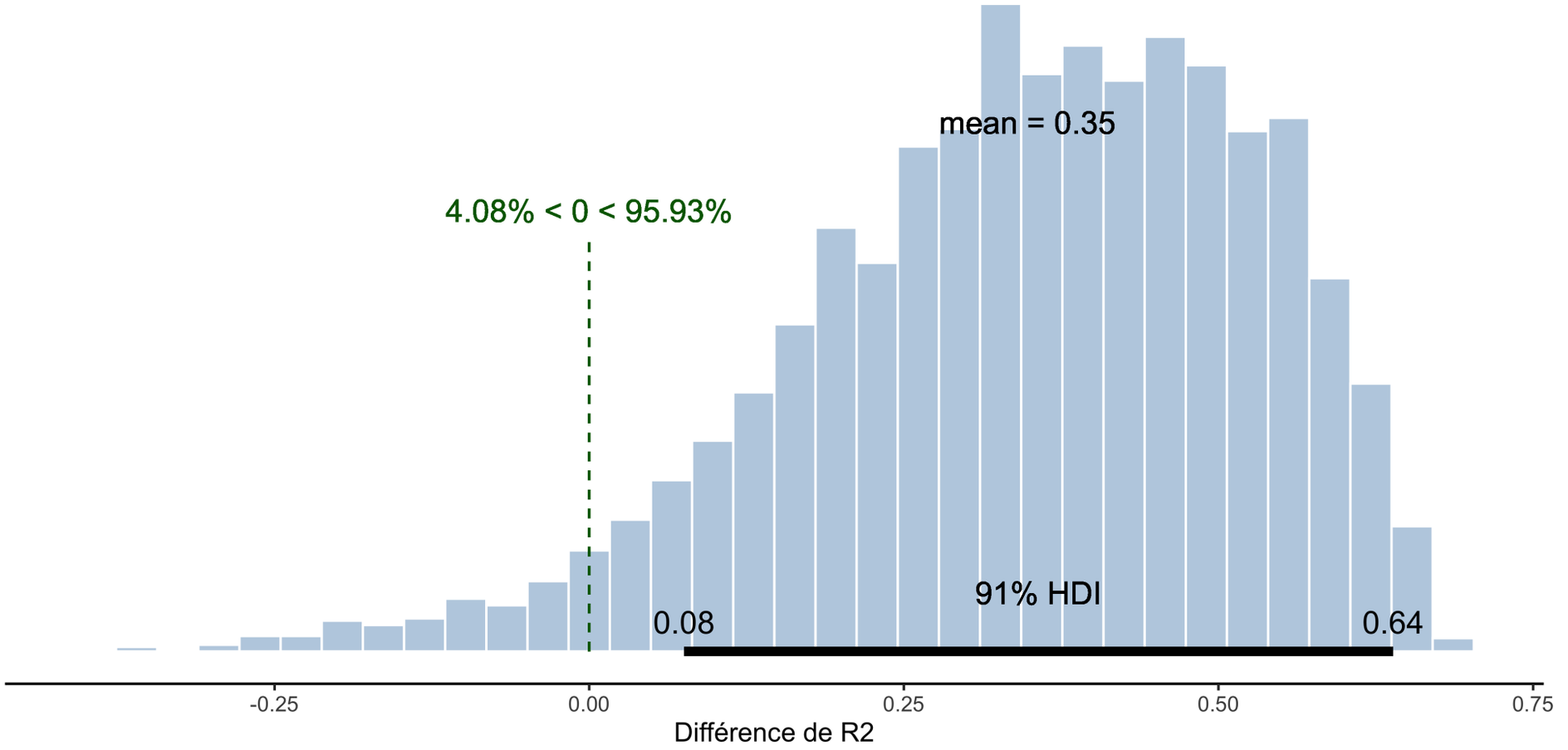
```
   Estimate Est.Error  Q2.5 Q97.5
R2    0.496      0.13 0.167 0.667
```

```
1  posterior_plot(samples = bayes_R2(mod2.4, summary = FALSE)[, 1])
```



R-squared

```
posterior_plot(
    samples = bayes_R2(mod2.4, summary = FALSE)[, 1] -
        bayes_R2(mod2.3, summary = FALSE)[, 1],
    compval = 0
    ) + labs(x = "Différence de R2")
```



Conclusions

Se méfier des interprétations intuitives (et souvent erronées) de la p-valeur, des intervalles de confiance, ou du facteur de Bayes.

Retenir que la difficulté en modélisation est de trouver un juste équilibre entre sous-apprentissage (underfitting) et sur-apprentissage (overfitting).

Pour contraindre l'apprentissage réalisé par le modèle sur les données observées (et ainsi éviter que le modèle accorde trop de poids à ces données), on peut utiliser des priors dits **régularisateurs** et/ou des outils comme la validation croisée ou les critères d'informations permettant d'estimer les capacités de prédiction du modèle sur de nouvelles données.

Travaux pratiques

```r
# import des données "howell"
d <- open_data(howell) %>% mutate(age = scale(age) )

# on définit une graine (afin de pouvoir reproduire les résultats)
set.seed(666)

# on échantillonne des lignes du jeu de données
i <- sample(1:nrow(d), size = nrow(d) / 2)

# on définit l'échantillon d'entraînement
d1 <- d[i, ]

# on définit l'échantillon de test
d2 <- d[-i, ]
```

Nous avons maintenant deux dataframes, de 272 lignes chacune. On va utiliser `d1` pour fitter nos modèles et `d2` pour les évaluer.

Travaux pratiques

Soit h_i les valeurs de taille et x_i les valeurs centrées d'âge, sur la ligne i. Construisez les modèles suivants avec `d1`, en utilisant `brms::brm()` et des priors faiblement régularisateurs.

$$
\begin{aligned}
\mathcal{M}_1 : h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i \\
\mathcal{M}_2 : h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 \\
\mathcal{M}_3 : h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 \\
\mathcal{M}_4 : h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 \\
\mathcal{M}_5 : h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \beta_5 x_i^5 \\
\mathcal{M}_6 : h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \beta_5 x_i^5 + \beta_6 x_i^6
\end{aligned}
$$

Travaux pratiques

- Comparer ces modèles en utilisant le WAIC. Comparer les rangs des modèles et leurs poids.
- Pour chaque modèle, produire un plot de la moyenne estimée et l'intervalle de confiance à 97% de la moyenne, surimposée aux données brutes. Comment ces prédictions diffèrent-elles selon les modèles ?
- Faire un plot des prédictions moyennées sur tous les modèles (sur les trois meilleurs). En quoi ces prédictions diffèrent-elles des prédictions du modèle avec le plus petit WAIC ?
- Calculer la déviance **out-of-sample** pour chaque modèle. Comparer les déviances obtenues à la question précédente aux valeurs de WAIC. Basé sur les déviances obtenues, quel modèle fait les meilleures prédictions ? Est-ce que le WAIC est un bon estimateur de la déviance **out-of-sample** ?

Solution - Question 1

```r
mod3.1 <- brm(
  formula = height ~ 1 + age,
  family = gaussian(),
  data = d1,
  prior = c(
    prior(normal(0, 100), class = Intercept),
    prior(exponential(0.01), class = sigma)
    ),
  backend = "cmdstanr"
  )

mod3.2 <- update(
  mod3.1,
  newdata = d1,
  formula = height ~ 1 + age + I(age^2)
  )

mod3.3 <- update(
  mod3.1,
  newdata = d1,
  formula = height ~ 1 + age + I(age^2) + I(age^3)
  )
```


Solution - Question 1

```r
mod3.4 <- update(
  mod3.1,
  newdata = d1,
  formula = height ~ 1 + age + I(age^2) + I(age^3) + I(age^4)
  )

mod3.5 <- update(
  mod3.1,
  newdata = d1,
  formula = height ~ 1 + age + I(age^2) + I(age^3) + I(age^4) + I(age^5)
  )

mod3.6 <- update(
  mod3.1,
  newdata = d1,
  formula = height ~ 1 + age + I(age^2) + I(age^3) + I(age^4) + I(age^5) + I(age^6)
  )
```


Solution - Question 1

```r
# calcul du WAIC et ajout du WAIC à chaque modèle

mod3.1 <- add_criterion(mod3.1, "waic")
mod3.2 <- add_criterion(mod3.2, "waic")
mod3.3 <- add_criterion(mod3.3, "waic")
mod3.4 <- add_criterion(mod3.4, "waic")
mod3.5 <- add_criterion(mod3.5, "waic")
mod3.6 <- add_criterion(mod3.6, "waic")

# comparaison des WAIC de chaque modèle

mod_comp <- loo_compare(mod3.1, mod3.2, mod3.3, mod3.4, mod3.5, mod3.6, criterion = "waic")
print(mod_comp, digits = 2, simplify = FALSE)
```

```
       elpd_diff se_diff  elpd_waic se_elpd_waic p_waic   se_p_waic waic
mod3.4     0.00      0.00  -954.40     12.41         5.74     0.79   1908.79
mod3.5    -0.70      0.47  -955.10     12.48         6.46     0.87   1910.20
mod3.6    -1.29      1.03  -955.69     12.27         7.44     0.93   1911.37
mod3.3   -21.90      6.37  -976.30     11.83         6.11     1.31   1952.60
mod3.2  -132.62     13.63 -1087.02     11.29         5.51     1.28   2174.03
mod3.1  -259.20     15.32 -1213.60     10.91         3.43     0.42   2427.20
       se_waic
mod3.4    24.83
mod3.5    24.95
mod3.6    24.53
mod3.3    23.66
mod3.2    22.58
mod3.1    21.82
```

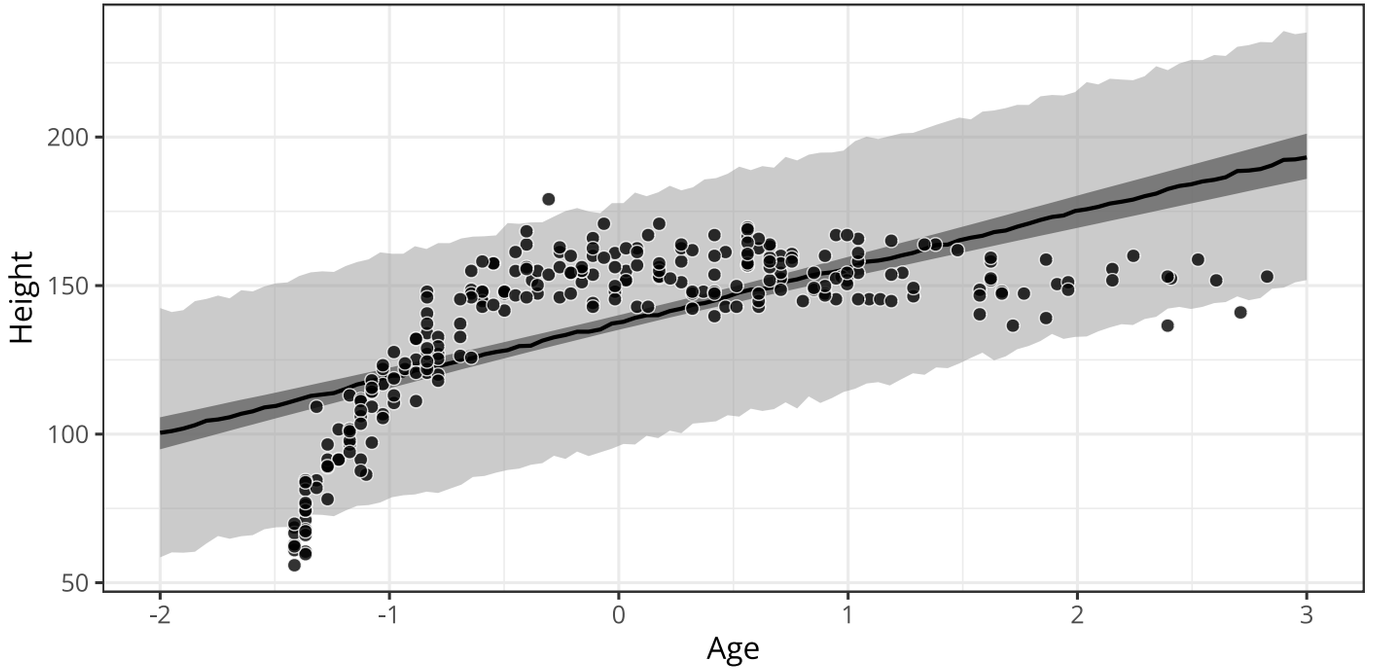

Solution - Question 2

```r
# on crée un vecteur de valeurs possibles pour "age"
age_seq <- data.frame(age = seq(from = -2, to = 3, length.out = 1e2) )

# on récupère les prédictions du modèle pour ces valeurs
mu <- data.frame(fitted(mod3.1, newdata = age_seq) ) %>% bind_cols(age_seq)

# on récupère les prédictions du modèle pour ces valeurs
pred_age <- data.frame(predict(mod3.1, newdata = age_seq) ) %>% bind_cols(age_seq)

# on affiche les dix premières prédictions
head(pred_age, 10)
```

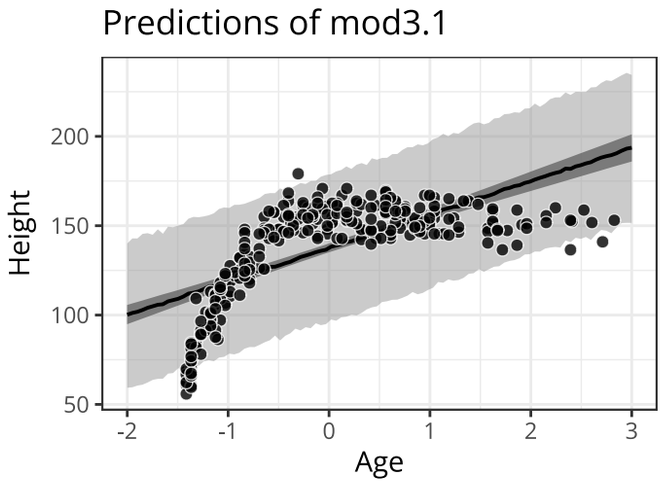
```
   Estimate Est.Error     Q2.5    Q97.5       age
1  100.4499  21.13775 58.51867 142.4313 -2.000000
2  101.0368  20.87114 60.20049 141.1053 -1.949495
3  101.8842  20.93198 60.09004 141.7364 -1.898990
4  103.0255  21.18667 60.41278 144.7774 -1.848485
5  104.5247  21.22701 63.20084 145.5188 -1.797980
6  104.9453  20.88182 65.70421 146.6455 -1.747475
7  105.6927  20.74311 64.78789 145.7145 -1.696970
8  106.8630  20.86651 65.64000 148.4519 -1.646465
9  107.6850  20.79897 66.01354 148.0521 -1.595960
10 108.9927  20.77924 67.99611 149.7180 -1.545455
```

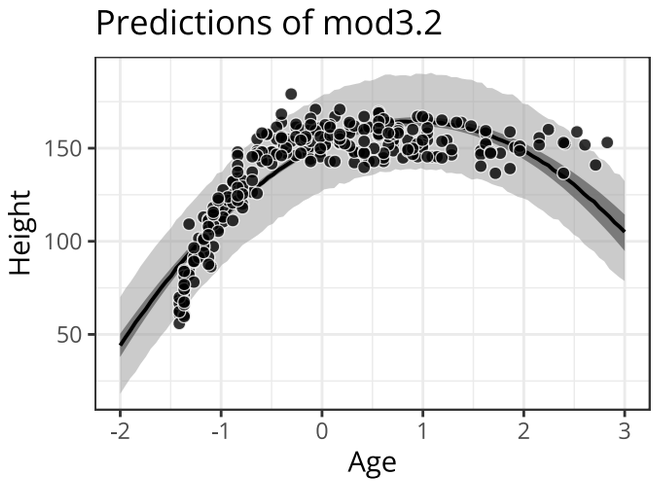
Solution - Question 2

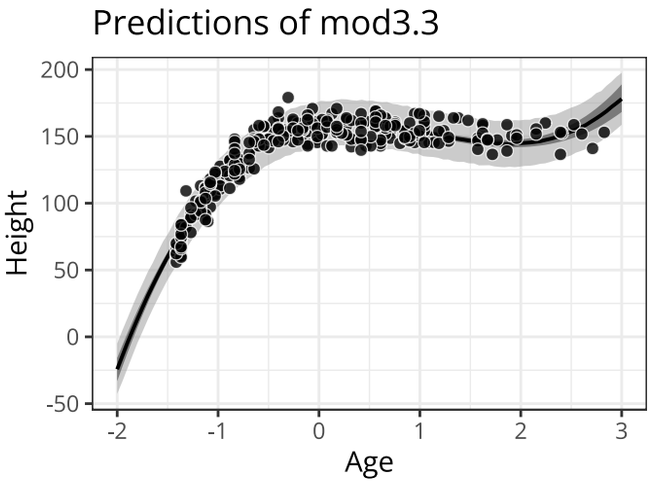
```r
d1 %>%
  ggplot(aes(x = age, y = height) ) +
  geom_ribbon(
    data = mu, aes(x = age, ymin = Q2.5, ymax = Q97.5),
    alpha = 0.8, inherit.aes = FALSE
    ) +
  geom_smooth(
    data = pred_age, aes(y = Estimate, ymin = Q2.5, ymax = Q97.5),
    stat = "identity", color = "black", alpha = 0.5, size = 1
    ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
  labs(x = "Age", y = "Height")
```

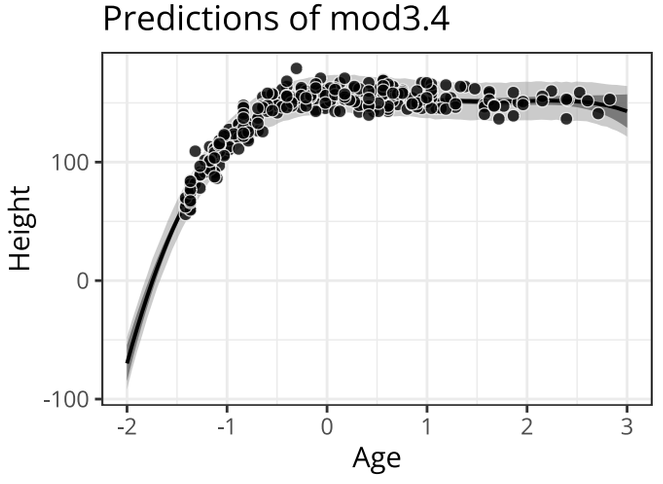


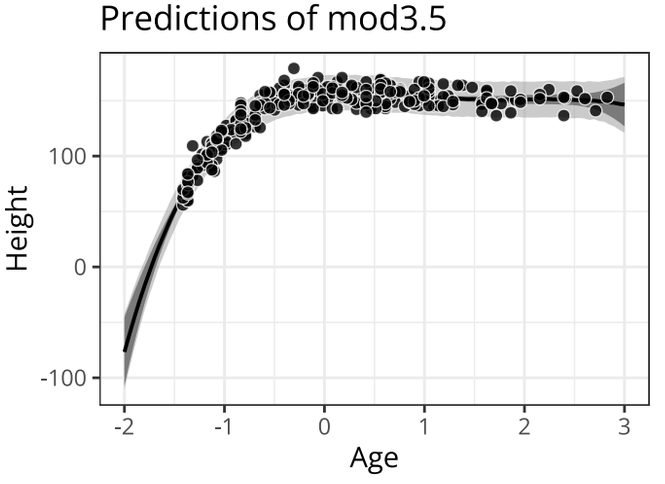
Solution - Question 2

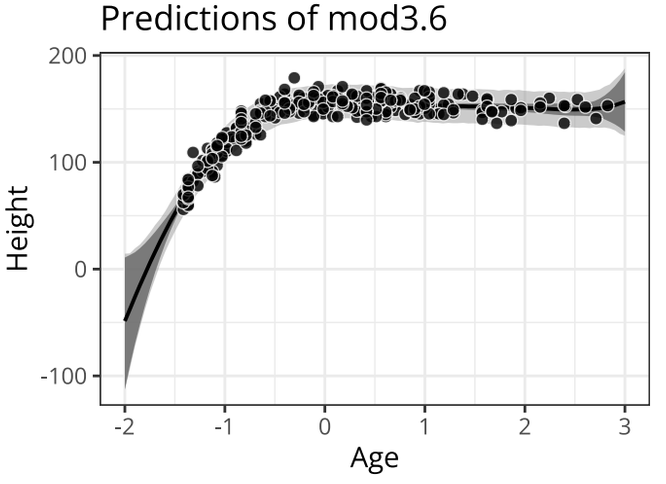










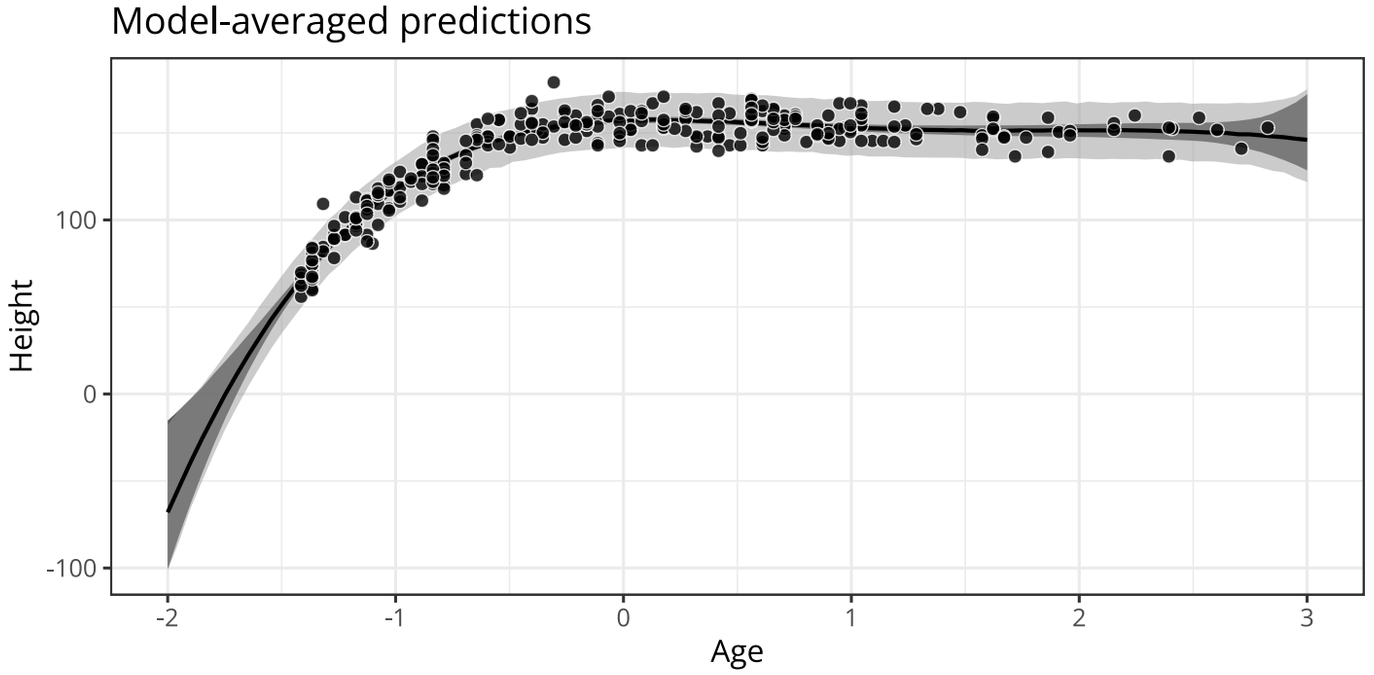


Solution - Question 3

```r
# prédictions moyennées sur les 4 modèles
averaged_predictions_mu <- pp_average(
  mod3.1, mod3.2, mod3.3, mod3.4, mod3.5, mod3.6,
  weights = "waic",
  method  = "fitted",
  newdata = age_seq
  ) %>%
  as.data.frame() %>%
  bind_cols(age_seq)

# prédictions moyennées sur les 4 modèles
averaged_predictions_age <- pp_average(
  mod3.1, mod3.2, mod3.3, mod3.4, mod3.5, mod3.6,
  weights = "waic",
  method  = "predict",
  newdata = age_seq
  ) %>%
  as.data.frame() %>%
  bind_cols(age_seq)
```


Solution - Question 3

```r
d1 %>%
  ggplot(aes(x = age, y = height) ) +
  geom_ribbon(
    data = averaged_predictions_mu, aes(x = age, ymin = Q2.5, ymax = Q97.5),
    alpha = 0.8, inherit.aes = FALSE
    ) +
  geom_smooth(
    data = averaged_predictions_age, aes(y = Estimate, ymin = Q2.5, ymax = Q97.5),
    stat = "identity", color = "black", alpha = 0.5, size = 1
    ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
  labs(x = "Age", y = "Height", title = "Model-averaged predictions")
```

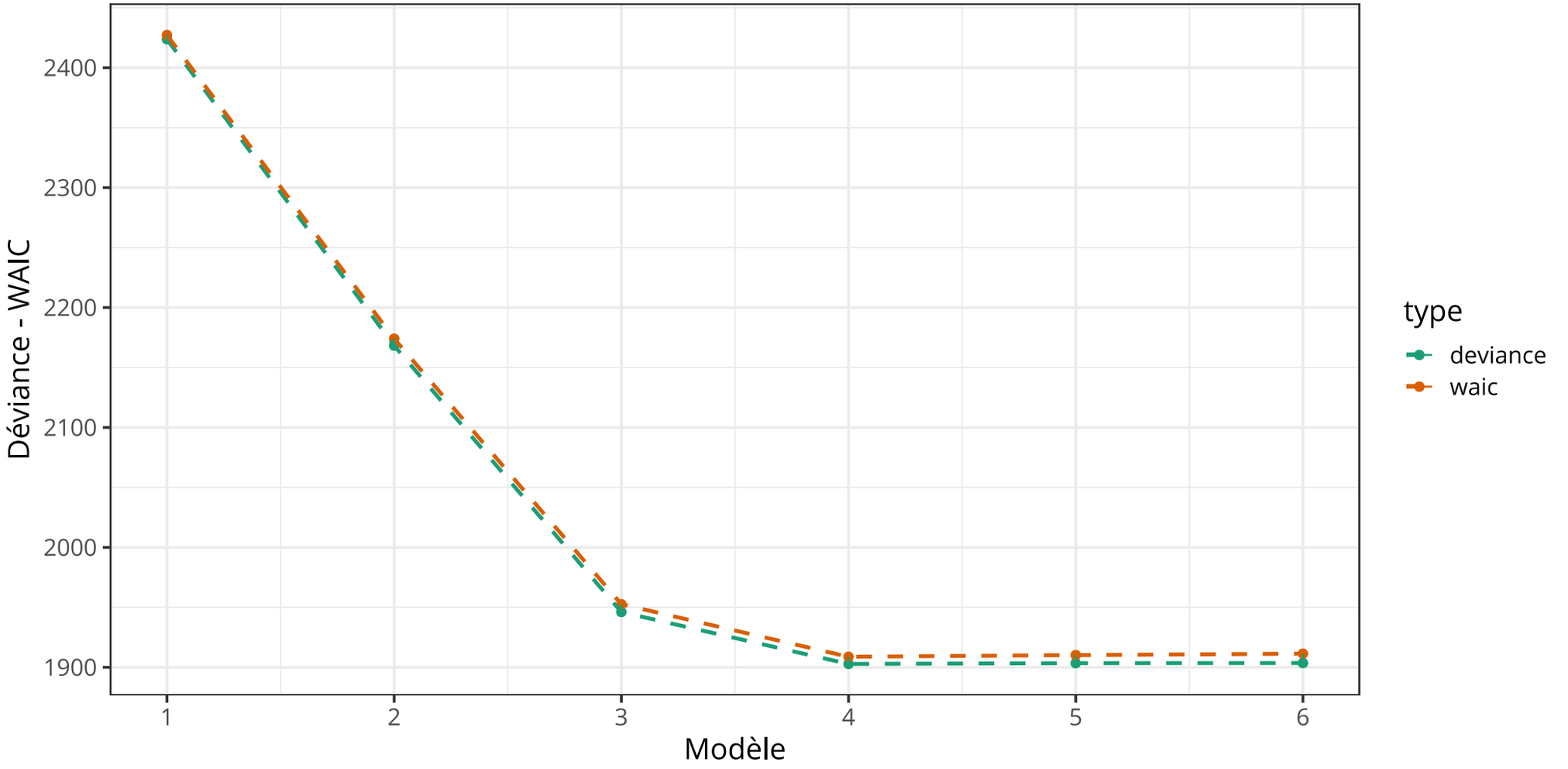


Solution - Question 4

```r
# calcul de la log-vraisemblance (log-likelihood) du modèle mod3.1
# chaque ligne est une itération et chaque colonne une observation
log_lik_mod3.1 <- log_lik(mod3.1)

# NB : la déviance possède également une distribution dans le monde bayésien...
dev.mod3.1 <- mean(-2 * rowSums(log_lik_mod3.1) )

# calcul de la log-vraisemblance (log-likelihood) du modèle mod3.2
dev.mod3.2 <- mean(-2 * rowSums(log_lik(mod3.2) ) )

# calcul de la log-vraisemblance (log-likelihood) du modèle mod3.3
dev.mod3.3 <- mean(-2 * rowSums(log_lik(mod3.3) ) )

# calcul de la log-vraisemblance (log-likelihood) du modèle mod3.4
dev.mod3.4 <- mean(-2 * rowSums(log_lik(mod3.4) ) )

# calcul de la log-vraisemblance (log-likelihood) du modèle mod3.5
dev.mod3.5 <- mean(-2 * rowSums(log_lik(mod3.5) ) )

# calcul de la log-vraisemblance (log-likelihood) du modèle mod3.6
dev.mod3.6 <- mean(-2 * rowSums(log_lik(mod3.6) ) )
```


Solution - Question 4

```
deviances <- c(dev.mod3.1, dev.mod3.2, dev.mod3.3, dev.mod3.4, dev.mod3.5, dev.mod3.6)
comparison <- mod_comp %>% data.frame %>% select(waic) %>% rownames_to_column()
waics <- comparison %>% arrange(rowname) %>% pull(waic)
```



Références

Burnham, K. P., & Anderson, D. R. (2004). Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods & Research*, *33*(2), 261–304. https://doi.org/10.1177/0049124104268644

Burnham, K. P., & Anderson, D. R. (2002). *Model selection and multimodel inference: A practical information-theoretic approach* (2nd ed). Springer.