Introduction à la modélisation statistique bayésienne

Un cours en R et Stan avec brms

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

Planning

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

Langage de la modélisation

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta x_i \\
\alpha &\sim \text{Normal}(60, 10) \\
\beta &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{HalfCauchy}(0, 1)
\end{aligned}
$$

Objectif de la séance : comprendre ce type de modèle.

Les constituants de nos modèles seront toujours les mêmes et nous suivrons les trois mêmes étapes :

- Construire le modèle (likelihood + priors).
- Mettre à jour grâce aux données, afin de calculer la distribution postérieure.
- Interpréter les estimations du modèle, évaluer ses prédictions, éventuellement modifier le modèle.

Un premier modèle

```
 1 library(tidyverse)
 2 library(imsb)
 3
 4 d <- open_data(howell)
 5 str(d)
```

```
'data.frame':   544 obs. of  4 variables:
 $ height: num  152 140 137 157 145 ...
 $ weight: num  47.8 36.5 31.9 53 41.3 ...
 $ age   : num  63 63 65 41 51 35 32 27 19 54 ...
 $ male  : int  1 0 0 1 0 1 0 1 0 1 ...
```

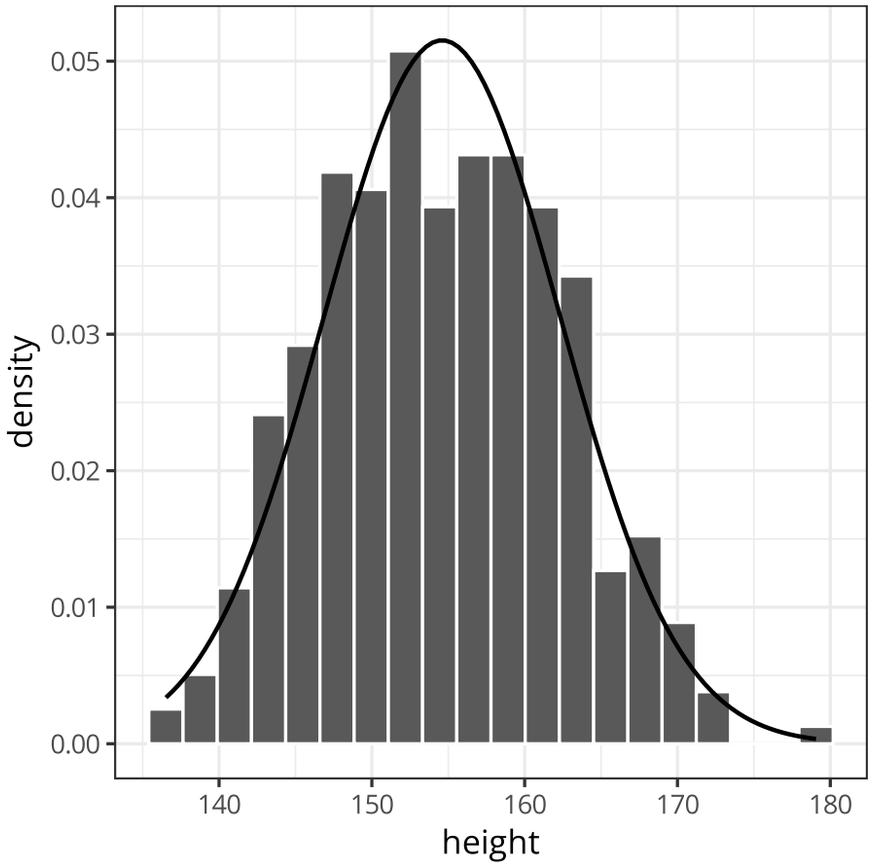
```
 1 d2 <- d %>% filter(age >= 18)
 2 head(d2)
```

```
   height   weight age male
1 151.765 47.82561  63    1
2 139.700 36.48581  63    0
3 136.525 31.86484  65    0
4 156.845 53.04191  41    1
5 145.415 41.27687  51    0
6 163.830 62.99259  35    1
```

Un premier modèle

$$h_i \sim \text{Normal}(\mu, \sigma)$$

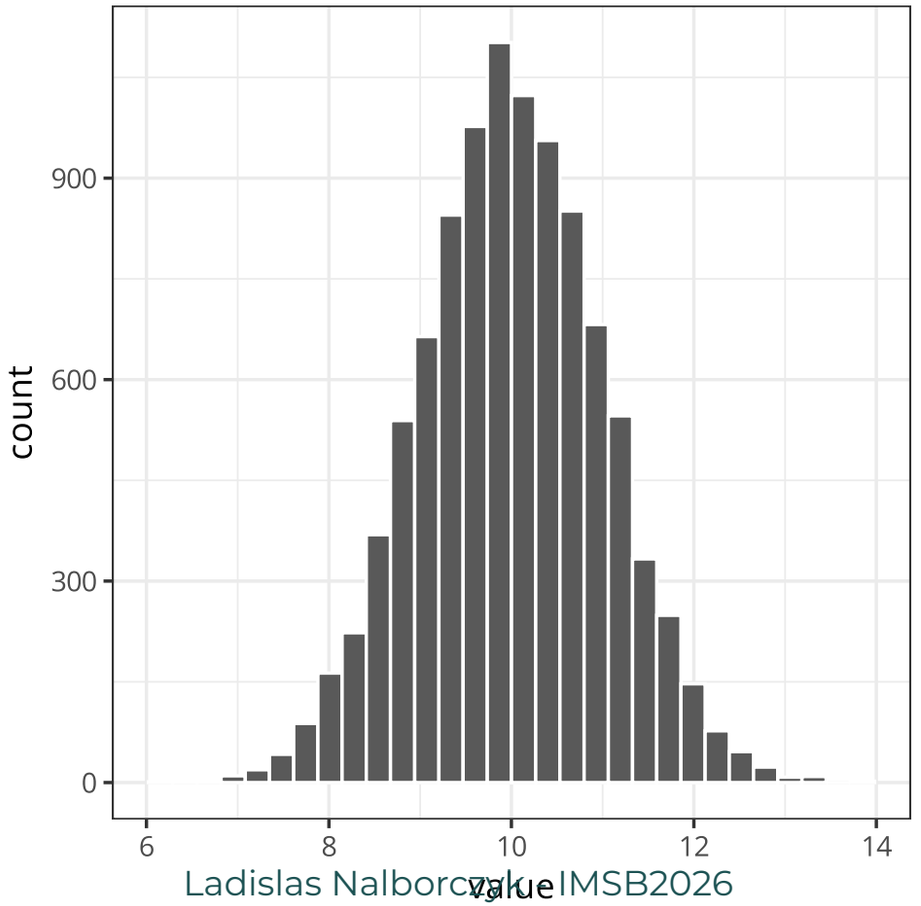
```
d2 %>%
    ggplot(aes(x = height) ) +
    geom_histogram(aes(y = ..density..), bins = 20, col = "white") +
    stat_function(fun = dnorm, args = list(mean(d2$height), sd(d2$height) ), size = 1)
```



Loi normale

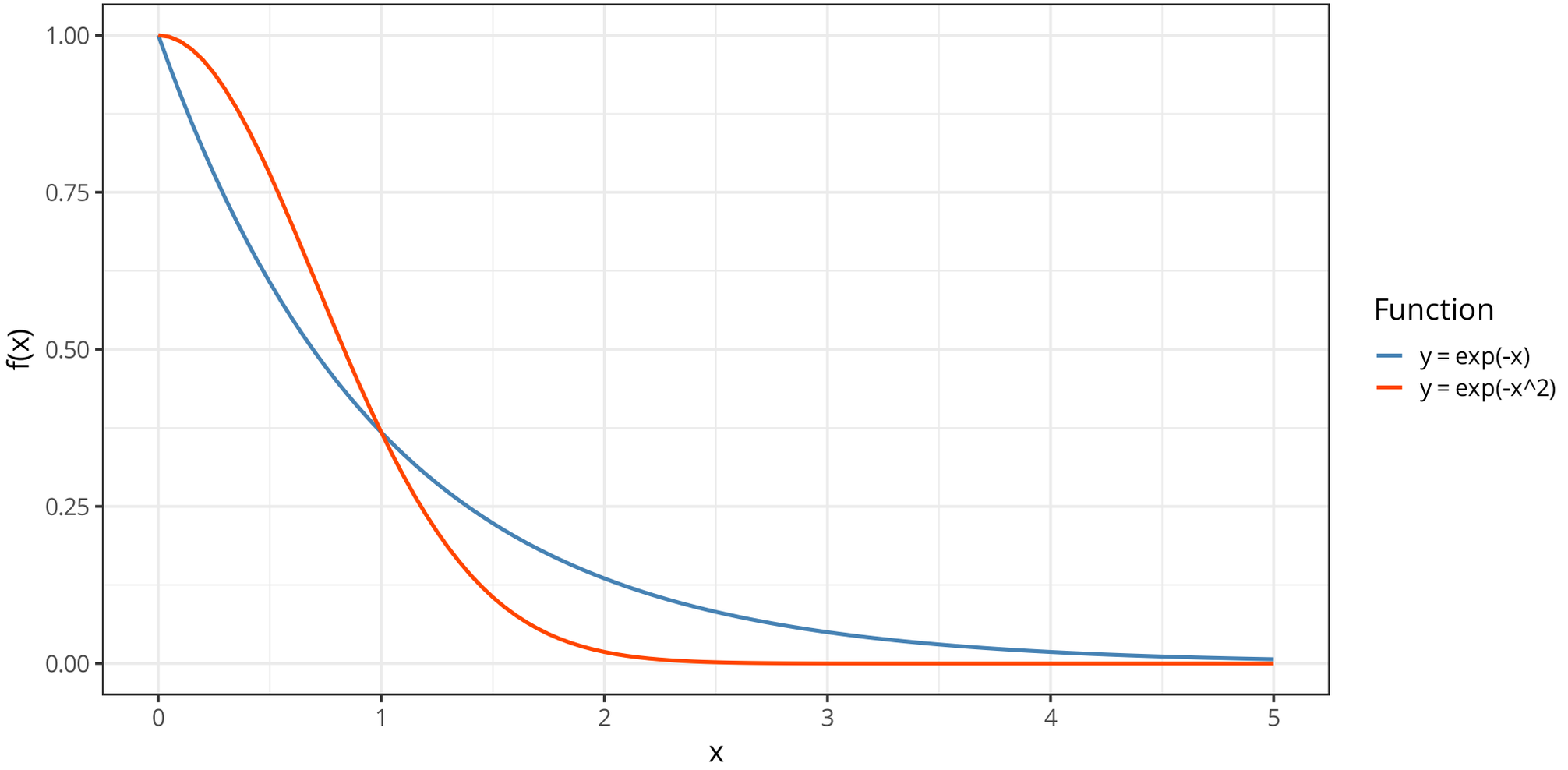
$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\mu - x)^2 \right]$$

```r
data.frame(value = rnorm(n = 1e4, mean = 10, sd = 1) ) %>% # 10.000 samples from Normal(10, 1)
    ggplot(aes(x = value) ) +
    geom_histogram(col = "white")
```



D'où vient la loi normale ?

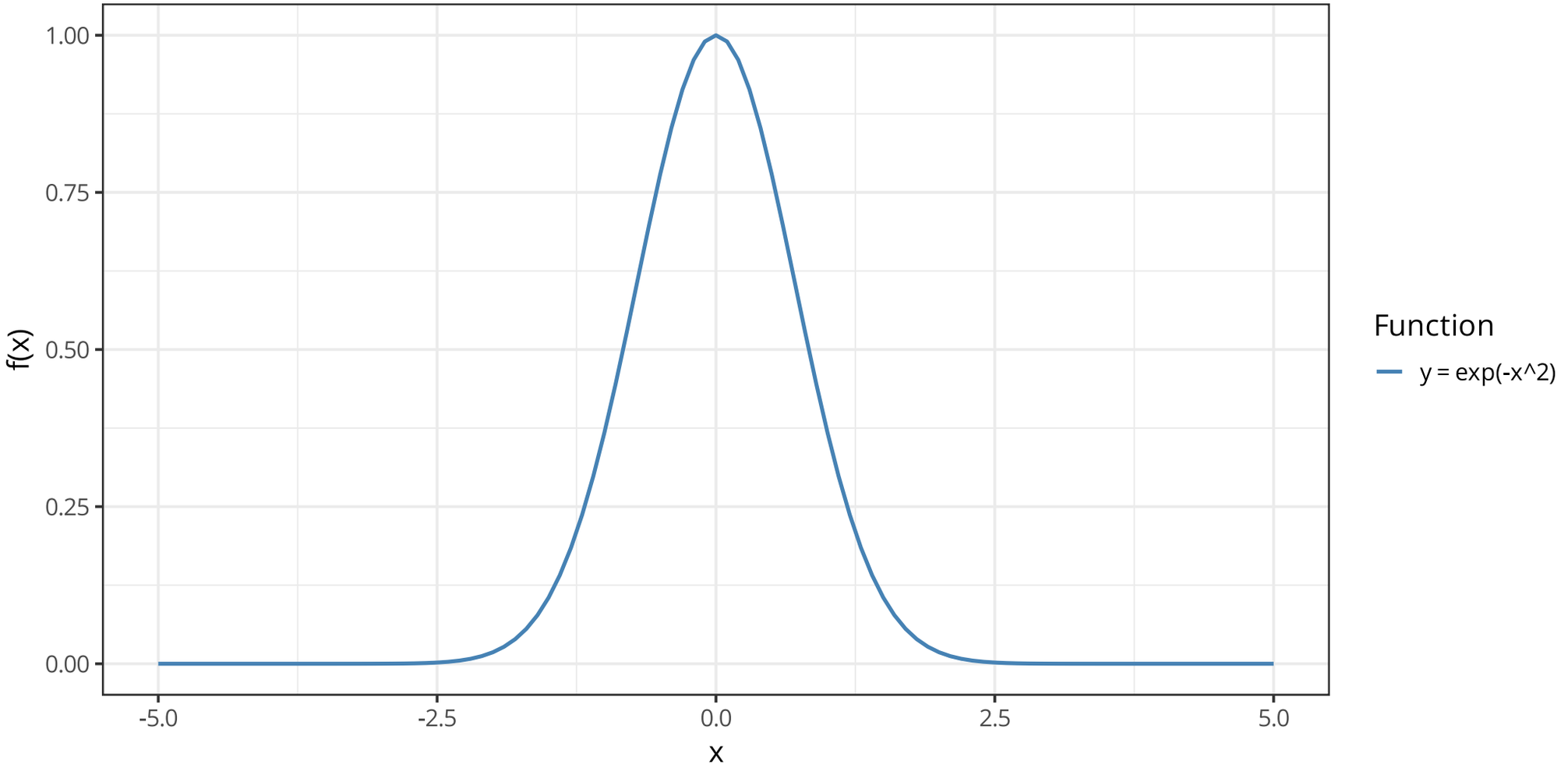
Contraintes : Certaines valeurs soient fortement probables (autour de la moyenne μ). Plus on s'éloigne, moins les valeurs sont probables (en suivant une décroissance exponentielle).



D'où vient la loi normale ?

$$y = \exp\left[-x^2\right]$$

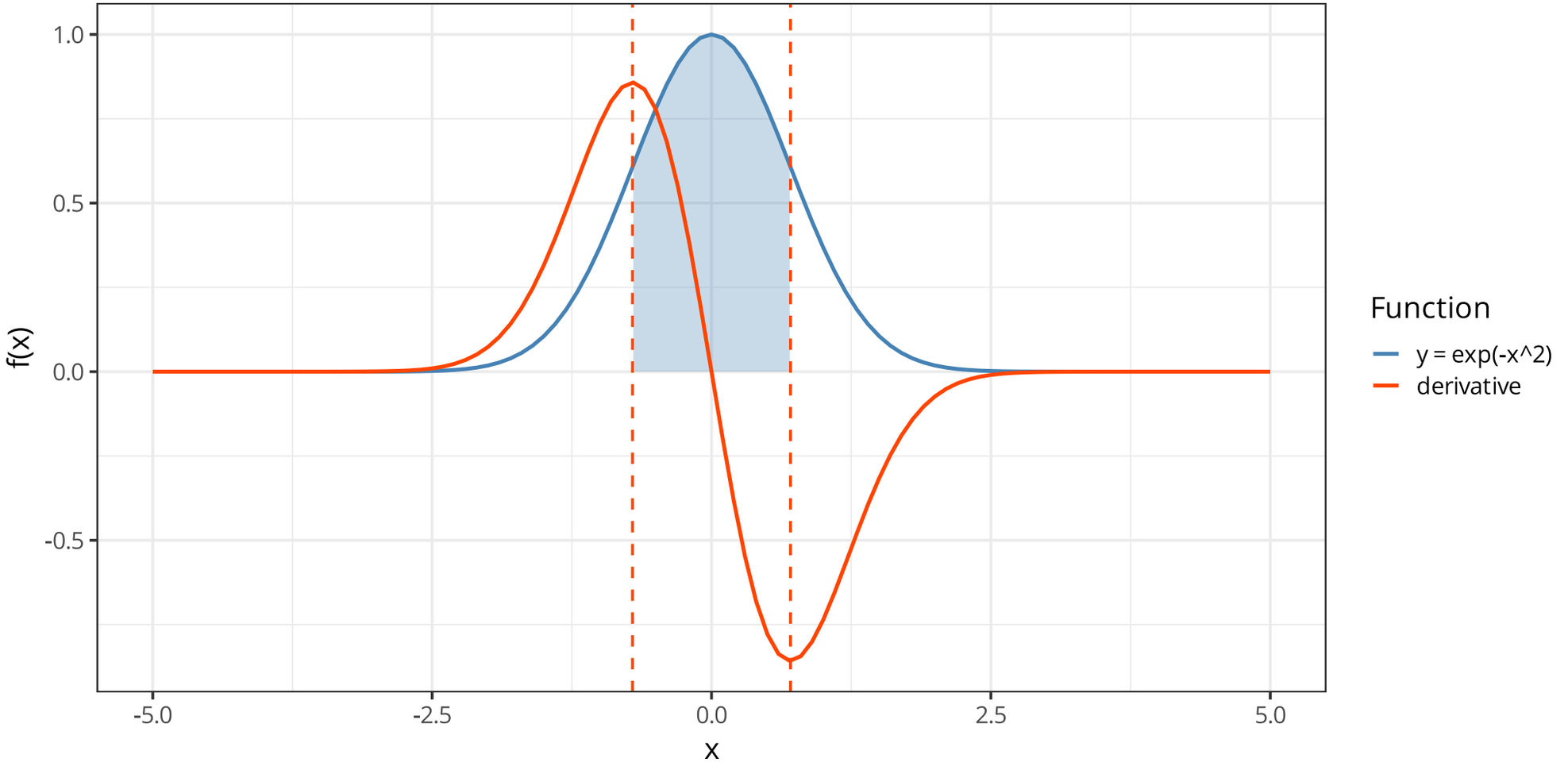
On étend notre fonction aux valeurs négatives.



D'où vient la loi normale ?

$$y = \exp\left[-x^2\right]$$

Les points d'inflection nous donnent une bonne indication de là où la plupart des valeurs se trouvent (i.e., entre les points d'inflection). Les pics de la dérivée nous montrent les points d'inflection.



D'où vient la loi normale ?

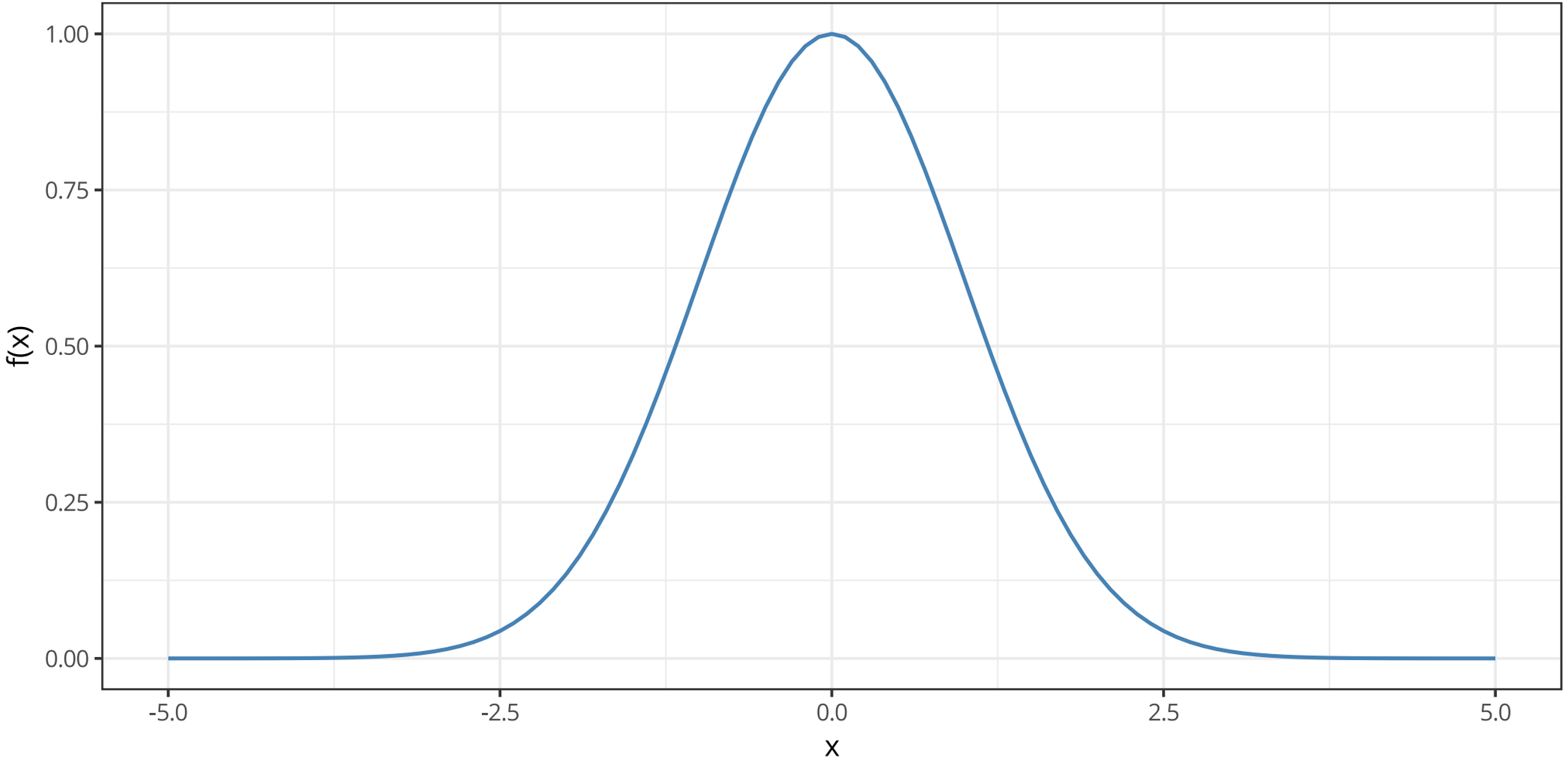
$$y = \exp\left[-\frac{1}{2}x^2\right]$$

Ensuite on standardise la distribution de manière à ce que les deux points d'inflection se trouvent à $x = -1$ et $x = 1$.

D'où vient la loi normale ?

$$y = \exp\left[-\frac{1}{2\sigma^2}x^2\right]$$

On insère un paramètre σ^2 pour contrôler la distance entre les points d'inflection.



D'où vient la loi normale ?

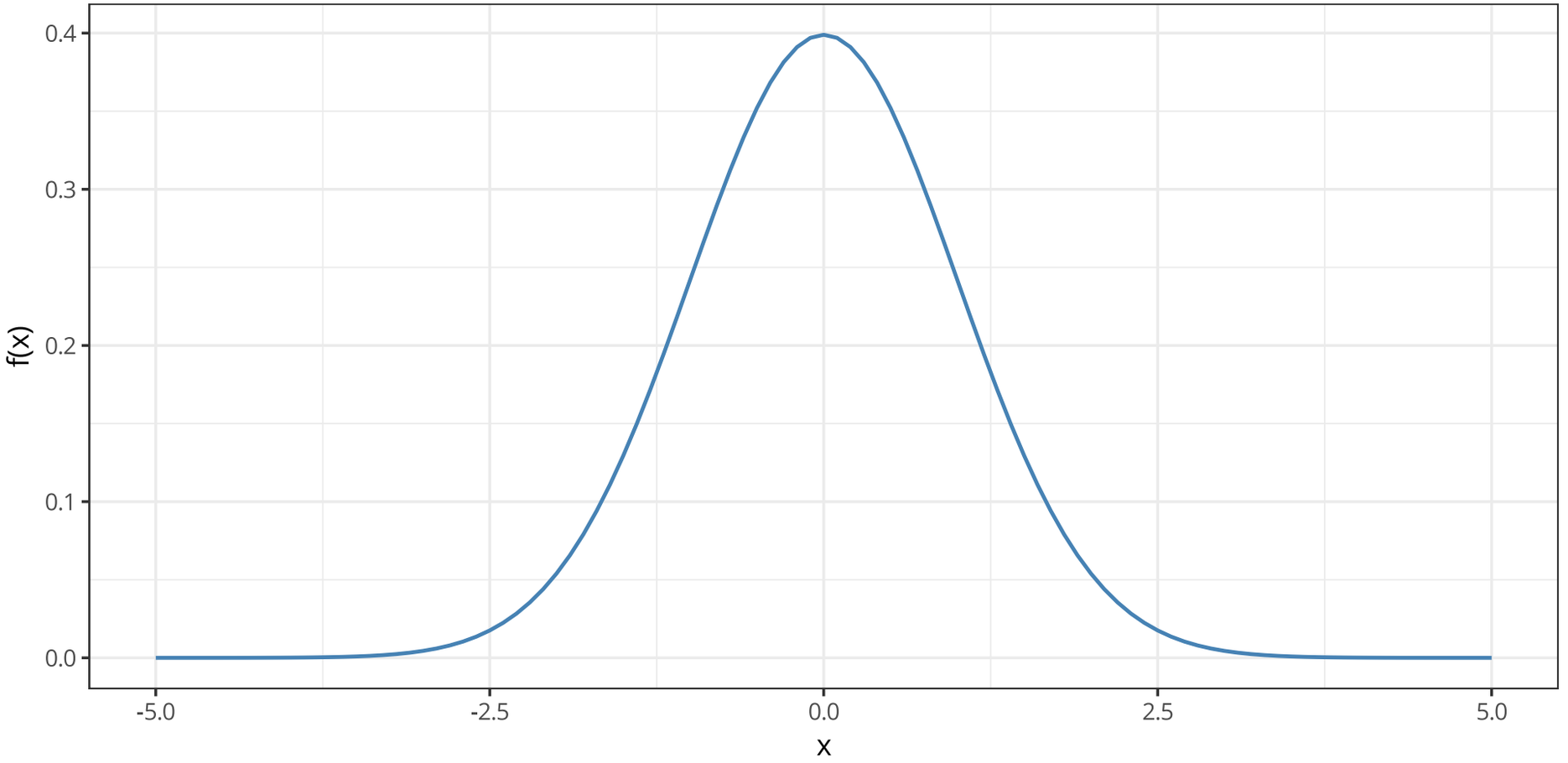
$$y = \exp\left[-\frac{1}{2\sigma^2}(\mu - x)^2\right]$$

On insère ensuite un paramètre μ afin de pouvoir contrôler la position (la tendance centrale) de la distribution.

D'où vient la loi normale ?

$$y = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[- \frac{1}{2\sigma^2} (\mu - x)^2 \right]$$

Mais... cette distribution n'intègre pas à 1. On divise donc par une constante de normalisation (la partie gauche), afin d'obtenir une distribution de probabilité.



Modèle gaussien

Nous allons construire un modèle de régression, mais avant d'ajouter un prédicteur, essayons de modéliser la distribution des tailles.

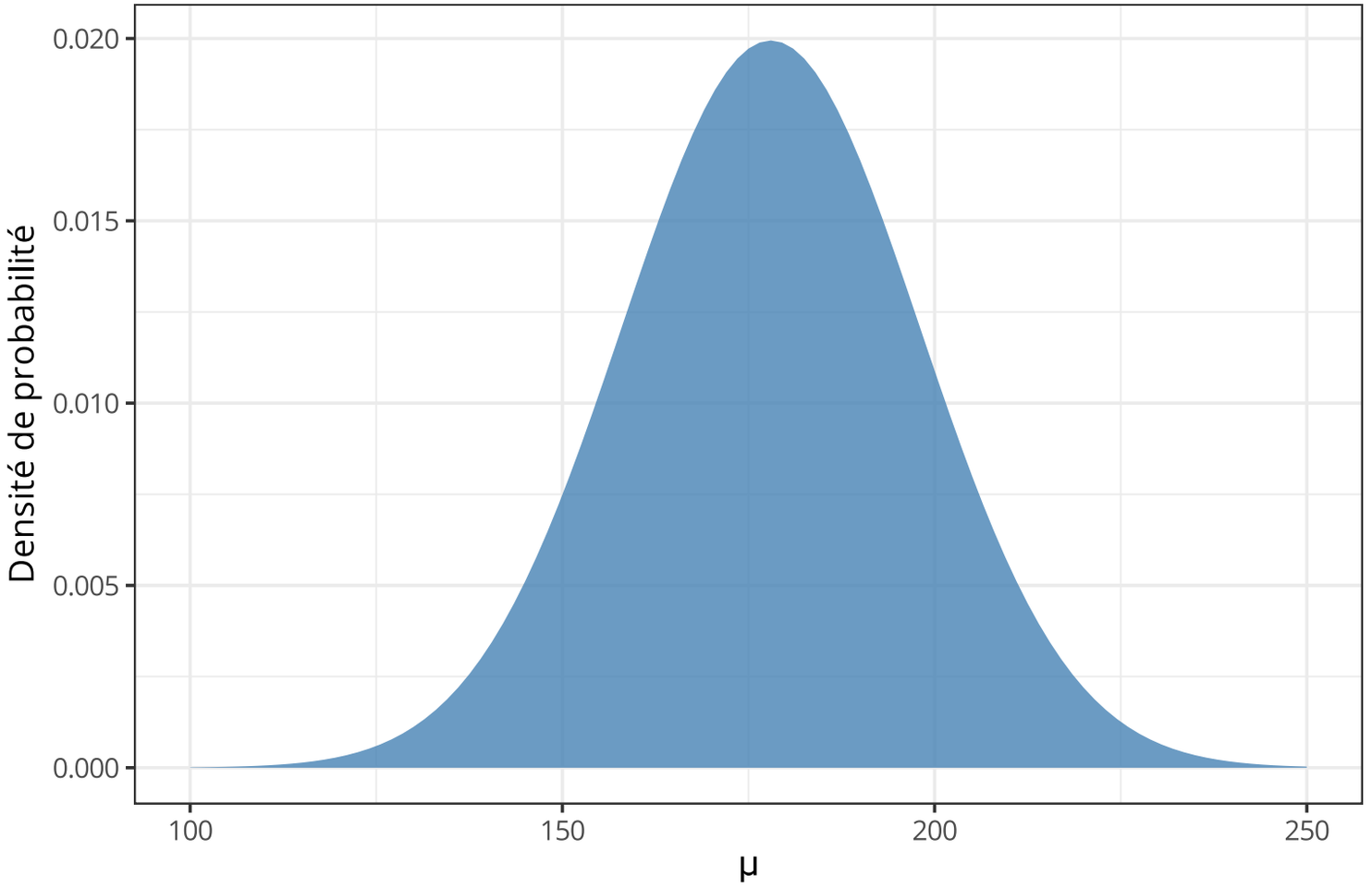
On cherche à savoir quel est le modèle (la distribution) qui décrit le mieux la répartition des tailles. On va donc explorer toutes les combinaisons possibles de μ et σ et les classer par leurs probabilités respectives.

Notre but, une fois encore, est de décrire **la distribution postérieure**, qui sera donc d'une certaine manière **une distribution de distributions**.

Modèle gaussien

On définit $p(\mu, \sigma)$, la distribution a priori conjointe de tous les paramètres du modèle. On peut spécifier ces priors indépendamment pour chaque paramètre, sachant que $p(\mu, \sigma) = p(\mu)p(\sigma)$.

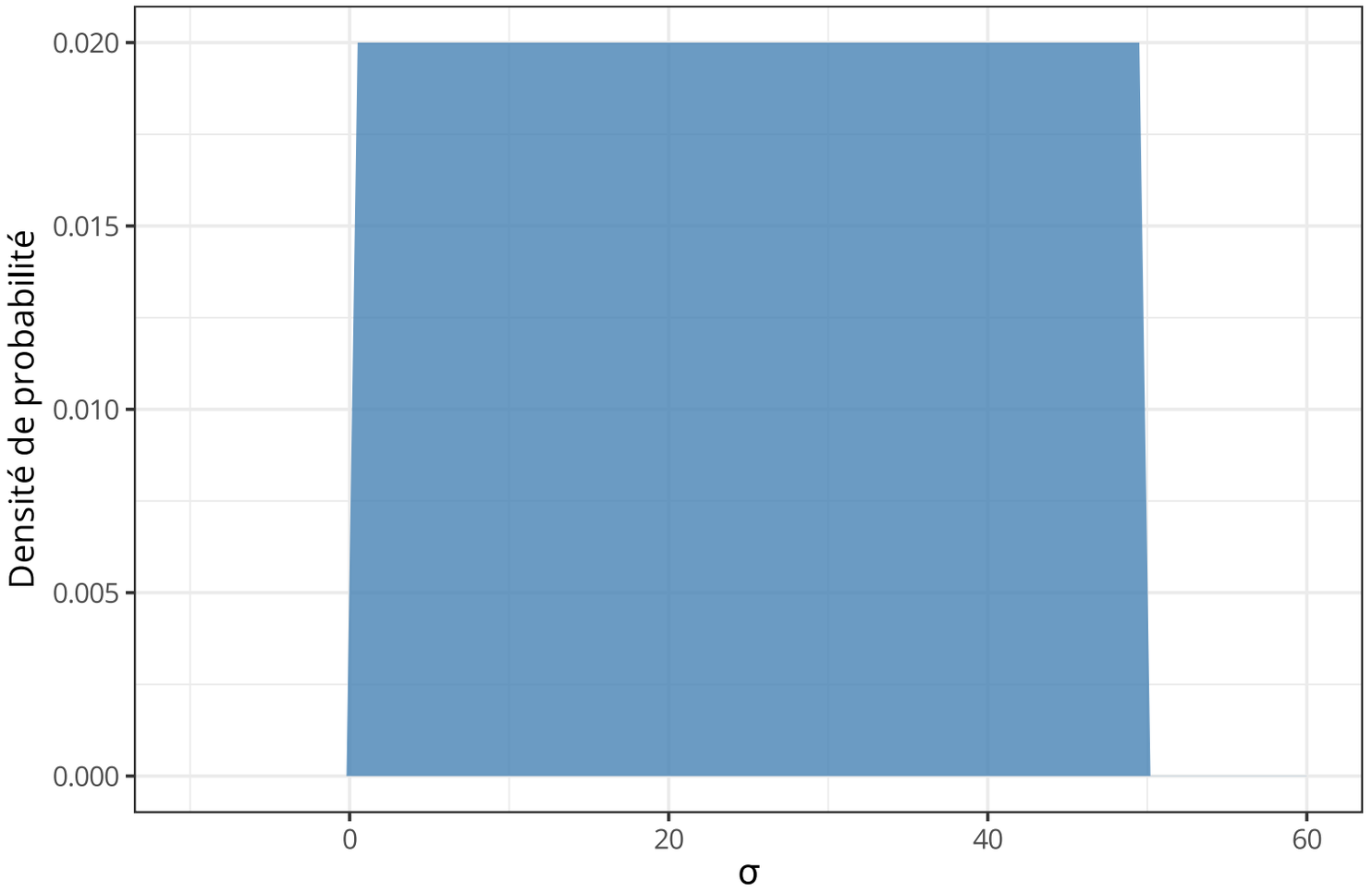
$$\mu \sim \text{Normal}(178, 20)$$



Modèle gaussien

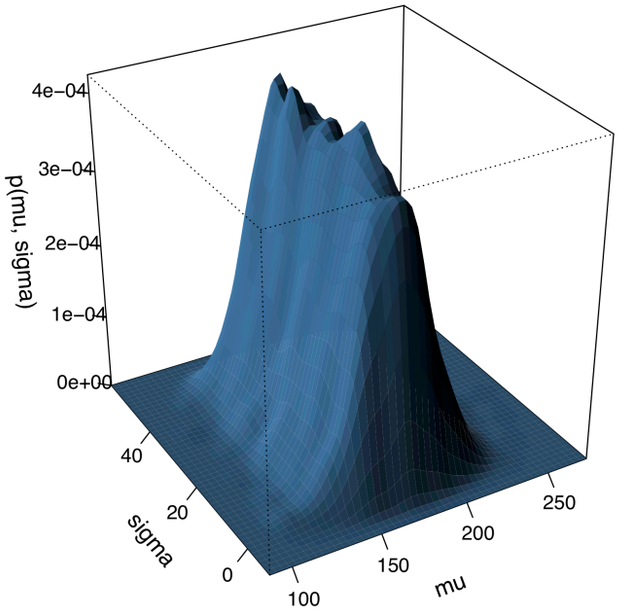
On définit $p(\mu, \sigma)$, la distribution a priori conjointe de tous les paramètres du modèle. On peut spécifier ces priors indépendamment pour chaque paramètre, sachant que $p(\mu, \sigma) = p(\mu)p(\sigma)$.

$$\sigma \sim \text{Uniform}(0, 50)$$



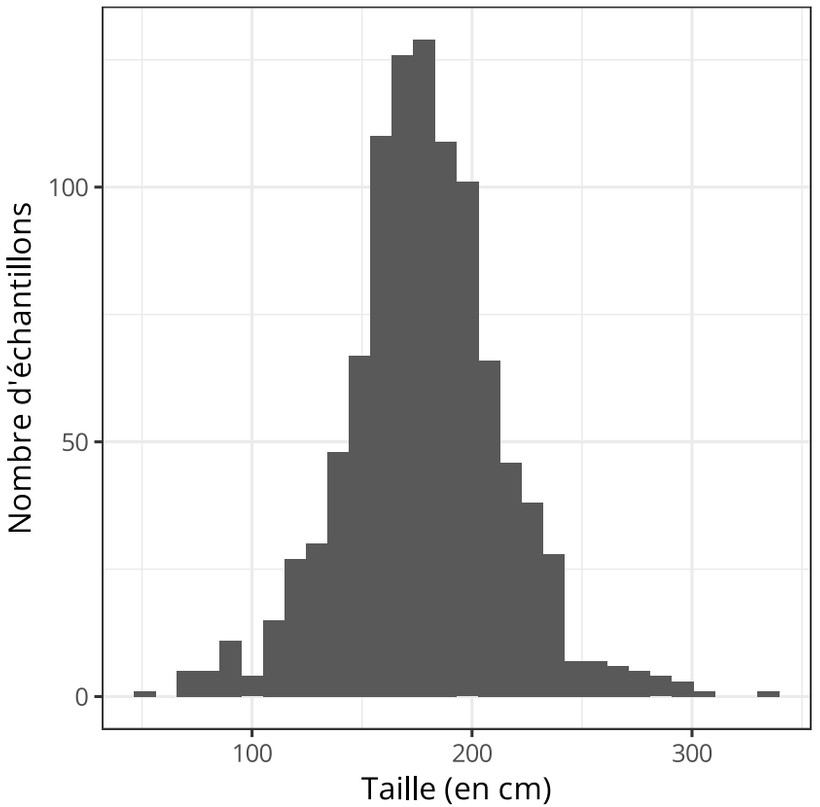
Visualiser le prior

```r
library(ks)
sample_mu <- rnorm(1e4, 178, 20) # prior on mu
sample_sigma <- runif(1e4, 0, 50) # prior on sigma
prior <- data.frame(cbind(sample_mu, sample_sigma) ) # multivariate prior
H.scv <- Hscv(x = prior, verbose = TRUE)
fhat_prior <- kde(x = prior, H = H.scv, compute.cont = TRUE)
plot(
    fhat_prior, display = "persp", col = "steelblue", border = NA,
    xlab = "\nmu", ylab = "\nsigma", zlab = "\n\np(mu, sigma)",
    shade = 0.8, phi = 30, ticktype = "detailed",
    cex.lab = 1.2, family = "Helvetica")
```



Prior predictive checking

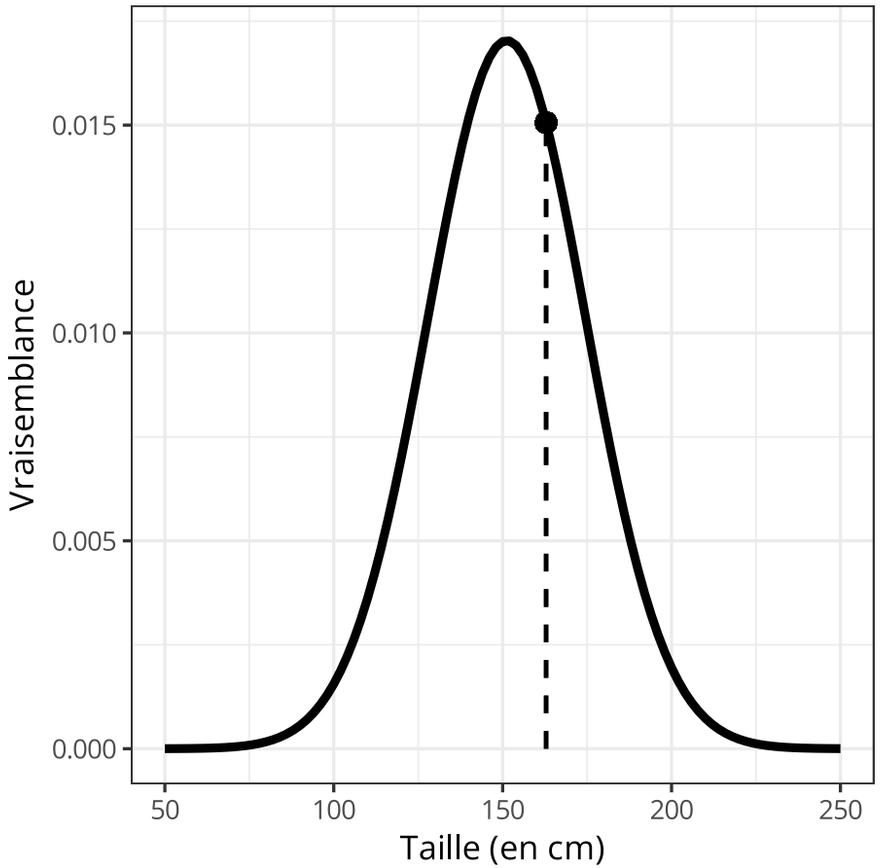
```r
sample_mu <- rnorm(1000, 178, 20)
sample_sigma <- runif(1000, 0, 50)

data.frame(x = rnorm(1000, sample_mu, sample_sigma) ) %>%
    ggplot(aes(x) ) +
    geom_histogram() +
    labs(x = "Taille (en cm)", y = "Nombre d'échantillons")
```



Fonction de vraisemblance

```r
mu_exemple <- 151.23
sigma_exemple <- 23.42

d2$height[34] # une observation de taille (pour l'exemple)
```

```
[1] 162.8648
```



Fonction de vraisemblance

On veut calculer la probabilité d'observer une certaine valeur de taille, sachant certaines valeurs de μ et σ, c'est à dire :

$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\mu - x)^2 \right]$$

On peut calculer cette **densité de probabilité** à l'aide des fonctions `dnorm`, `dbeta`, `dt`, `dexp`, `dgamma`, etc.

```
1 dnorm(d2$height[34], mu_exemple, sigma_exemple)
```

```
[1] 0.01505675
```

Fonction de vraisemblance

$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\mu - x)^2 \right]$$

Ou avec une fonction maison...

```r
normal_likelihood <- function (x, mu, sigma) {

  bell <- exp( (- 1 / (2 * sigma^2) ) * (mu - x)^2 )
  norm <- sqrt(2 * pi * sigma^2)

  return (bell / norm)

}
```

```r
normal_likelihood(d2$height[34], mu_exemple, sigma_exemple)
```

```
[1] 0.01505675
```

Distribution postérieure

$$p(\mu, \sigma \mid h) = \frac{\prod_i \text{Normal}(h_i \mid \mu, \sigma)\text{Normal}(\mu \mid 178, 20)\text{Uniform}(\sigma \mid 0, 50)}{\int\int \prod_i \text{Normal}(h_i \mid \mu, \sigma)\text{Normal}(\mu \mid 178, 20)\text{Uniform}(\sigma \mid 0, 50)\text{d}\mu\text{d}\sigma}$$

$$p(\mu, \sigma \mid h) \propto \prod_i \text{Normal}(h_i \mid \mu, \sigma)\text{Normal}(\mu \mid 178, 20)\text{Uniform}(\sigma \mid 0, 50)$$

Il s'agit de la même formule vue lors des cours 1 et 2, mais cette fois en considérant qu'il existe plusieurs observations de taille (h_i), et deux paramètres à estimer : μ et σ.

Pour calculer la **vraisemblance marginale** (en vert), il faut donc intégrer sur deux paramètres : μ et σ. On réalise ici encore que la probabilité a posteriori est proportionnelle au produit de la vraisemblance et du prior.

Distribution postérieure - Grid approximation

```r
# on définit une grille de valeurs possibles pour mu et sigma
mu.list <- seq(from = 140, to = 160, length.out = 200)
sigma.list <- seq(from = 4, to = 9, length.out = 200)

# on étend la grille à toutes les combinaisons possibles de mu et sigma
post <- crossing(mu = mu.list, sigma = sigma.list) %>% data.frame()

# calcul de la log-vraisemblance (pour chaque combinaison de mu et sigma)
post$LL <-
  sapply(
    1:nrow(post),
    function (i) sum(dnorm(
      d2$height,
      mean = post$mu[i],
      sd = post$sigma[i],
      log = TRUE) )
    )
```

```r
# calcul de la probabilité a posteriori (non normalisée)
post$prod <-
  post$LL +
  dnorm(x = post$mu, mean = 178, sd = 20, log = TRUE) +
  dunif(x = post$sigma, min = 0, max = 50, log = TRUE)

# on "annule" le log (après avoir normalisé par la valeur maximale, pour éviter les erreurs d'arrondi)
post$prob <- exp(post$prod - max(post$prod) )
```

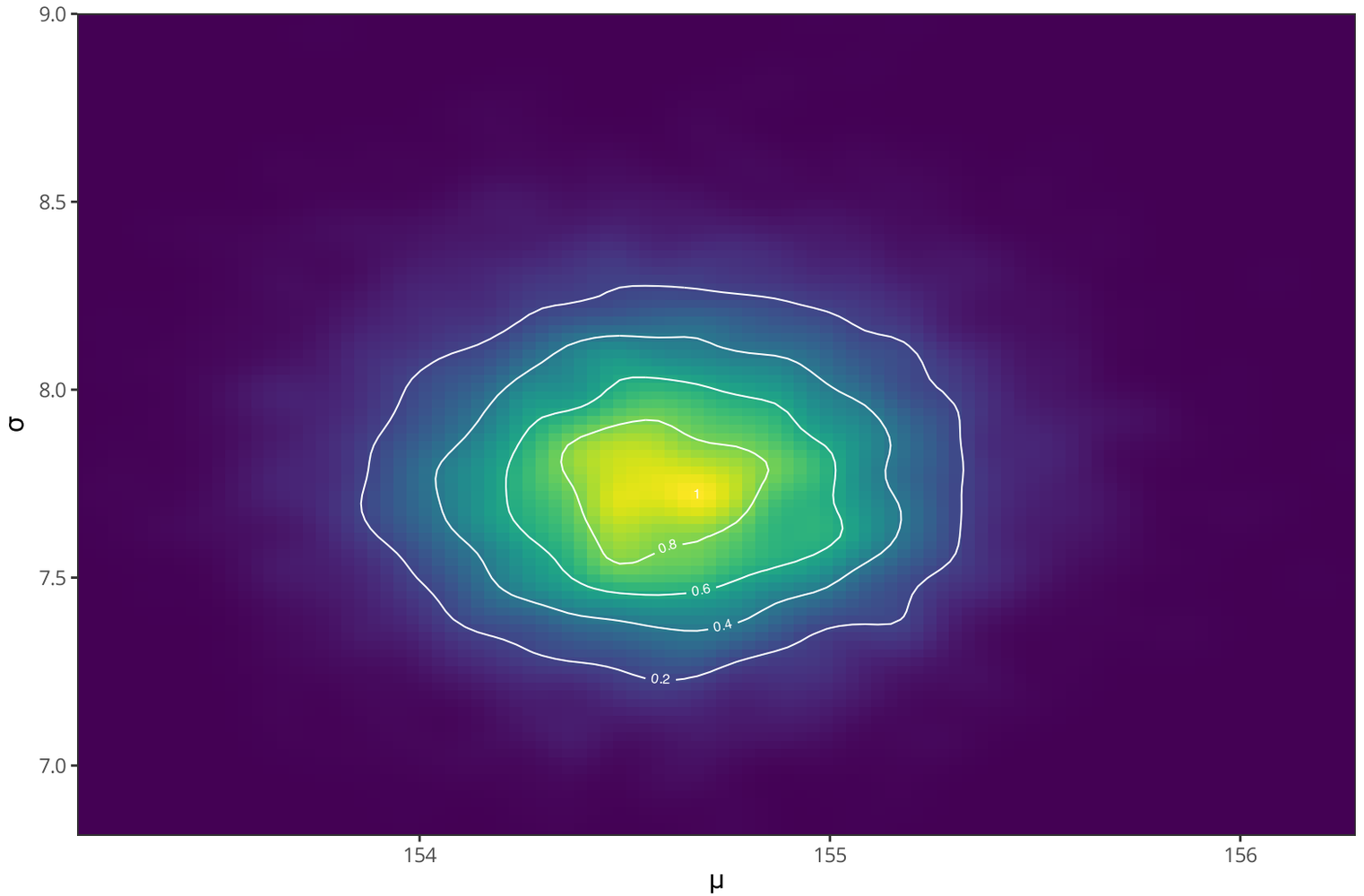
Distribution postérieure - Grid approximation

```r
# on affiche 10 lignes au hasard
post %>% slice_sample(n = 10, replace = FALSE)
```

```
         mu    sigma        LL      prod          prob
1  150.7538 7.065327 -1274.520 -1283.275  9.120349e-25
2  148.1407 7.442211 -1352.391 -1361.332  1.148621e-58
3  140.8040 5.407035 -2422.663 -2432.220  0.000000e+00
4  153.8693 8.974874 -1227.669 -1236.223  2.477911e-04
5  146.3317 4.879397 -1828.281 -1837.362 2.103866e-265
6  140.9045 8.547739 -1674.354 -1683.900 9.341393e-199
7  151.7588 8.798995 -1243.132 -1251.819  4.178022e-11
8  140.5025 6.688442 -2009.133 -2018.717  0.000000e+00
9  153.6683 7.618090 -1222.098 -1230.665  6.427073e-02
10 159.3970 4.577889 -1554.410 -1562.670 4.170956e-146
```

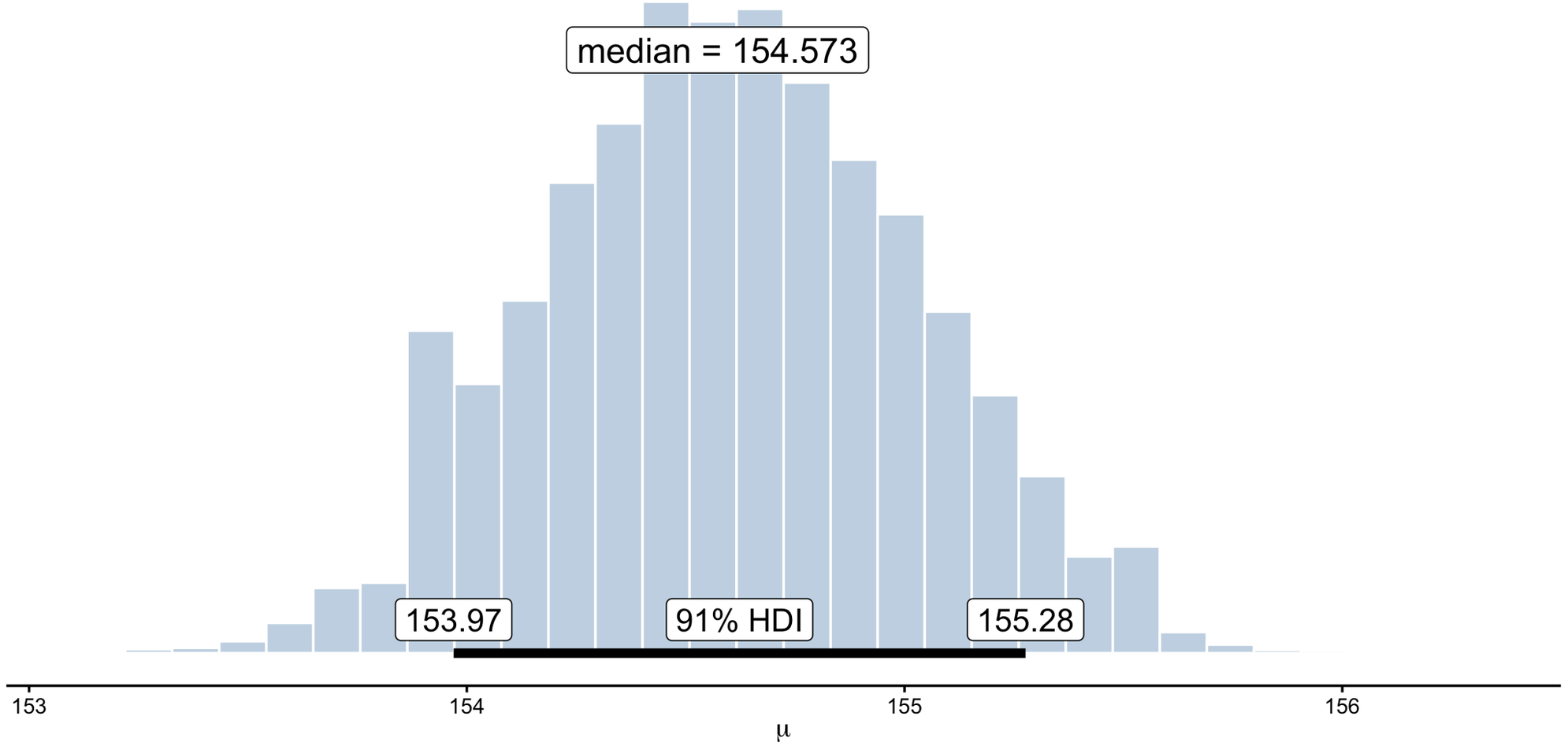
Distribution postérieure - Grid approximation

```
sample.rows <- sample(x = 1:nrow(post), size = 1e4, replace = TRUE, prob = post$prob)
sample.mu <- post$mu[sample.rows]
sample.sigma <- post$sigma[sample.rows]
```



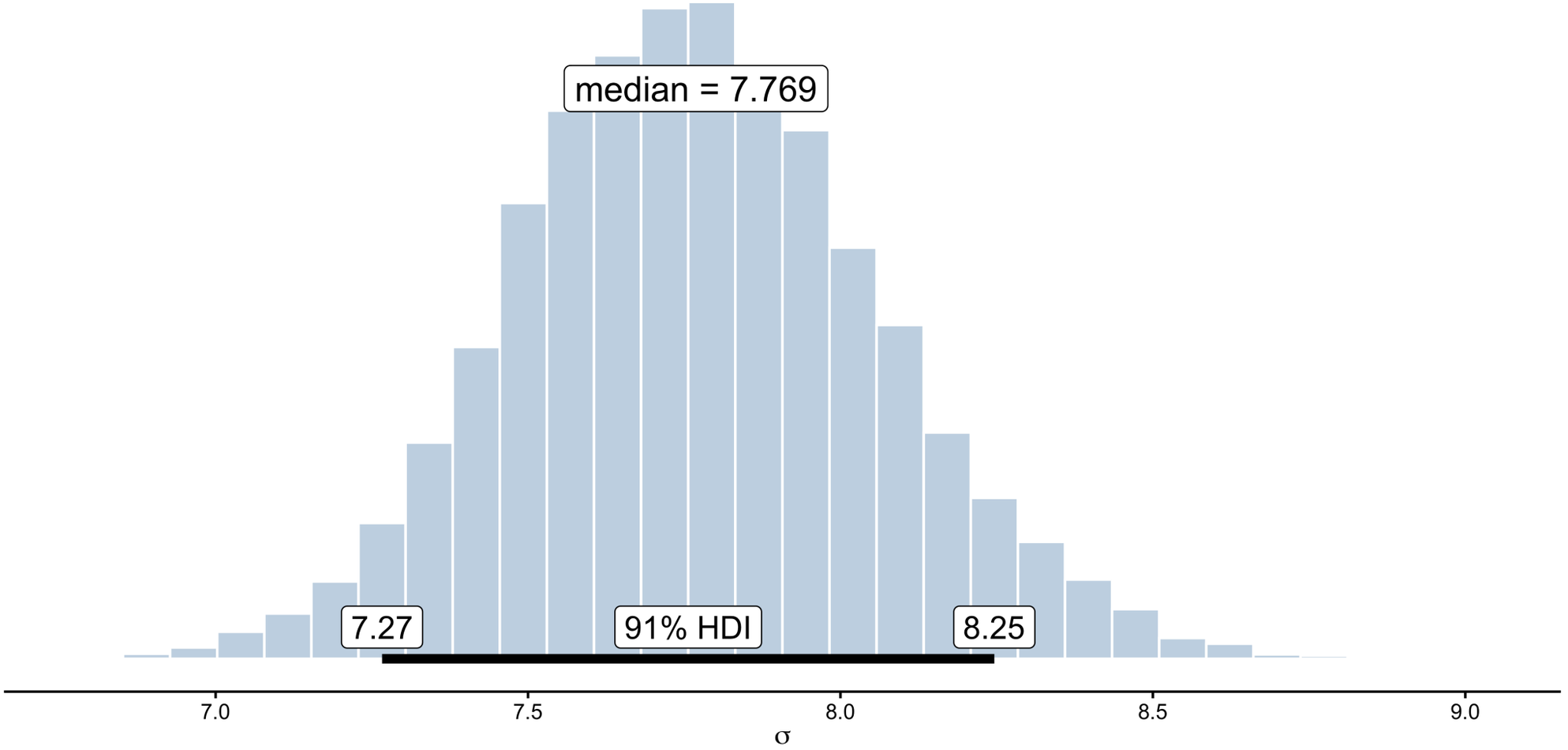
Distribution postérieure - Distributions marginales

```
1  posterior_plot(samples = sample.mu, nbins = 30) + labs(x = expression(mu) )
```



Distribution postérieure - Distributions marginales

```
1  posterior_plot(samples = sample.sigma, nbins = 30) + labs(x = expression(sigma) )
```



Introduction à brms

Under the hood : `Stan` est un langage de programmation probabiliste écrit en `C++`, et qui implémente plusieurs algorithmes de MCMC : HMC, NUTS, L-BFGS...

```
data {
  int<lower=0> J; // number of schools
  real y[J]; // estimated treatment effects
  real<lower=0> sigma[J]; // s.e. of effect estimates
  }

parameters {
  real mu;
  real<lower=0> tau;
  real eta[J];
  }

transformed parameters {
  real theta[J];
  for (j in 1:J)
    theta[j] = mu + tau * eta[j];
  }

model {
  target += normal_lpdf(eta | 0, 1);
  target += normal_lpdf(y | theta, sigma);
```

Bayesian regression models using Stan

Le package `brms` (Bürkner, 2017) permet de fitter des modèles multi-niveaux (ou pas) linéaires (ou pas) bayésiens en `Stan` mais en utilisant la syntaxe de `lme4`.

Par exemple, le modèle suivant :

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \alpha_{\text{subject}[i]} + \alpha_{\text{item}[i]} + \beta x_i
\end{aligned}
$$

se spécifie avec `brms` (comme avec `lme4`) de la manière suivante :

```
model <- brm(y ~ x + (1 | subject) + (1 | item), data = d, family = gaussian() )
```

Rappels de syntaxe

Le package `brms` utilise la même syntaxe que les fonctions de base R (comme `lm`) ou que le package `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 package `brms` permet également de fitter des modèles multivariés (plusieurs outcomes) en les combinant avec `mvbind()`.

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

Par défaut `brms` postule une vraisemblance gaussienne. Ce postulat peut être changé facilement en spécifiant la vraisemblance souhaitée via l'argument `family`.

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

Lisez la documentation (c'est très enthousiasmant à lire) accessible via `?brm`.

Quelques fonctions utiles

```r
# générer le code du modèle en Stan
make_stancode(formula, ...)
stancode(fit)

# définir les priors
get_prior(formula, ...)
set_prior(prior, ...)

# récupérer les prédictions du modèle
fitted(fit, ...)
predict(fit, ...)
conditional_effects(fit, ...)

# posterior predictive checking
pp_check(fit, ...)

# comparaison de modèles
loo(fit1, fit2, ...)
bayes_factor(fit1, fit2, ...)
model_weights(fit1, fit2, ...)

```

Un premier exemple

```
1 library(brms)
2 mod1 <- brm(height ~ 1, data = d2)
```

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

```
             Estimate Est.Error       Q2.5      Q97.5
b_Intercept 154.593518 0.4113323 153.790802 155.394615
sigma         7.752036 0.2961104   7.176074   8.332043
```

Ces données représentent les distributions marginales de chaque paramètre. En d'autres termes, la probabilité de μ, moyennée sur toutes les valeurs possible de σ, est décrite par une distribution gaussienne avec une moyenne de 154.59 et un écart type de 0.41. L'intervalle de crédibilité ($\neq$ intervalle de confiance) nous indique les 95% valeurs de μ ou σ les plus probables (sachant les données et les priors).

En utilisant notre prior

Par défaut `brms` utilise un prior très peu informatif centré sur la valeur moyenne de la variable mesurée. On peut donc affiner l'estimation réalisée par ce modèle en utilisant nos connaissances sur la distribution habituelle des tailles chez les humains.

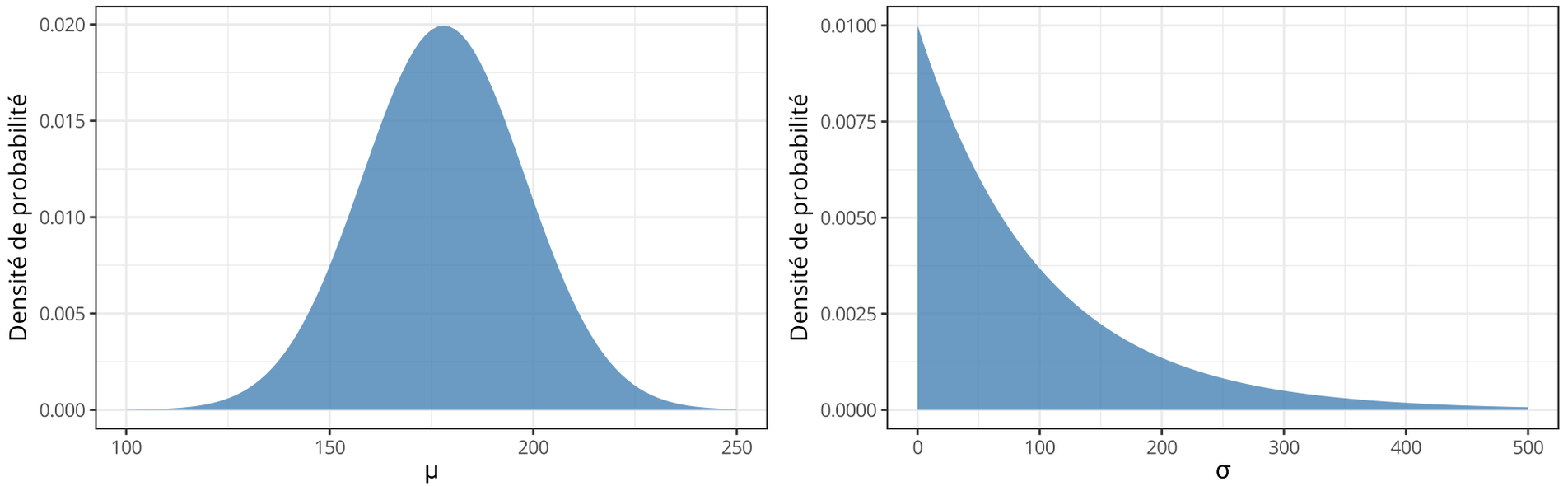
La fonction `get_prior()` permet de visualiser une liste des priors par défaut ainsi que de tous les priors qu'on peut spécifier, sachant une certaine formule (i.e., une manière d'écrire notre modèle) et un jeu de données.

```
1 get_prior(height ~ 1, data = d2)
```

```
                  prior     class coef group resp dpar nlpar lb ub tag
student_t(3, 154.3, 8.5) Intercept
    student_t(3, 0, 8.5)     sigma                         0
 source
default
default
```

En utilisant notre prior

```
priors <- c(
  prior(normal(178, 20), class = Intercept),
  prior(exponential(0.01), class = sigma)
  )

mod2 <- brm(
  height ~ 1,
  prior = priors,
  family = gaussian(),
  data = d2
  )
```



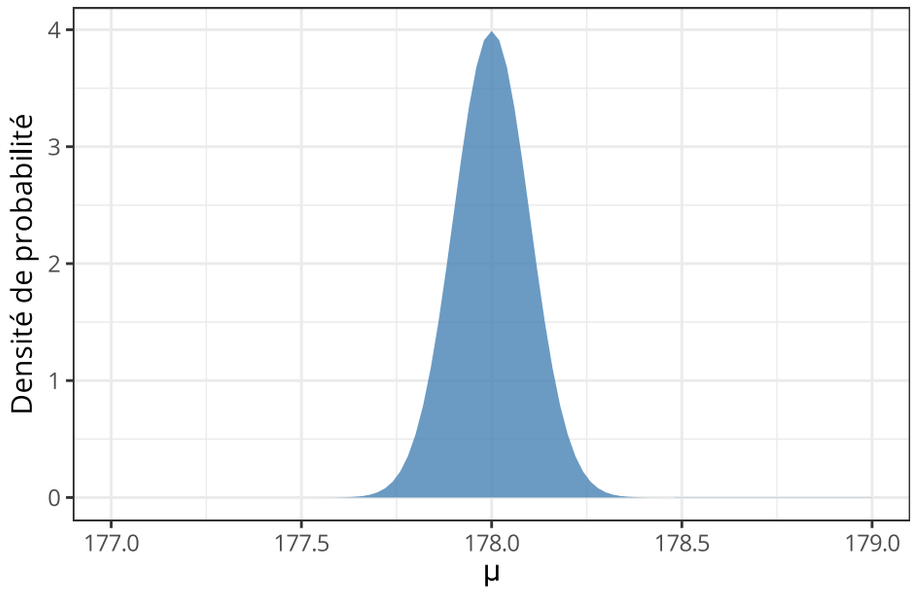
En utilisant notre prior

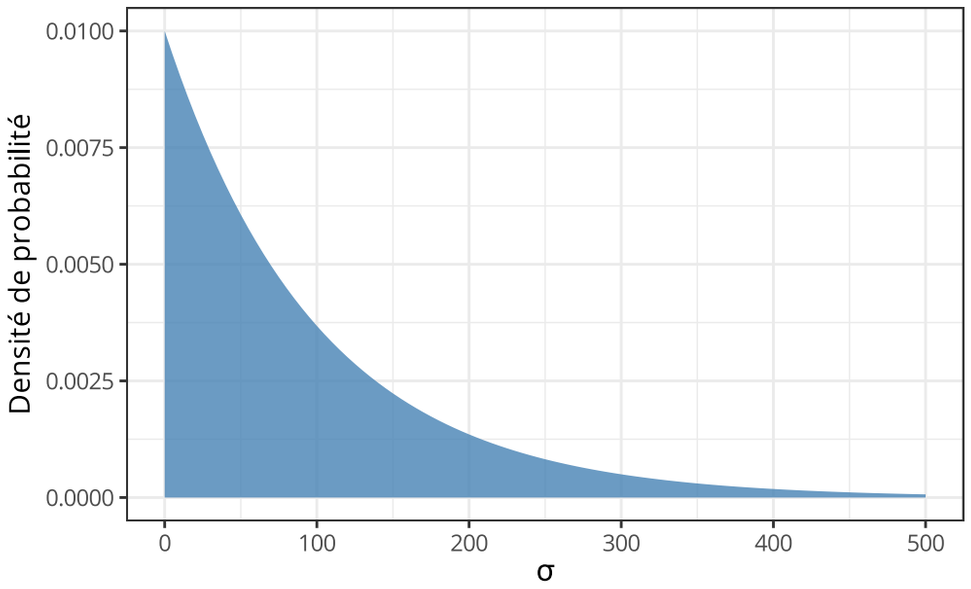
```
1 summary(mod2)
```

```
 Family: gaussian
  Links: mu = identity
Formula: height ~ 1
   Data: d2 (Number of observations: 352)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Regression Coefficients:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   154.60      0.42   153.78   155.44 1.00     3556     2566

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     7.77      0.29     7.21     8.36 1.00     3368     2592

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

En utilisant un prior plus informatif

```
priors <- c(
  prior(normal(178, 0.1), class = Intercept),
  prior(exponential(0.01), class = sigma)
  )

mod3 <- brm(
  height ~ 1,
  prior = priors,
  family = gaussian(),
  data = d2
  )
```





En utilisant un prior plus informatif

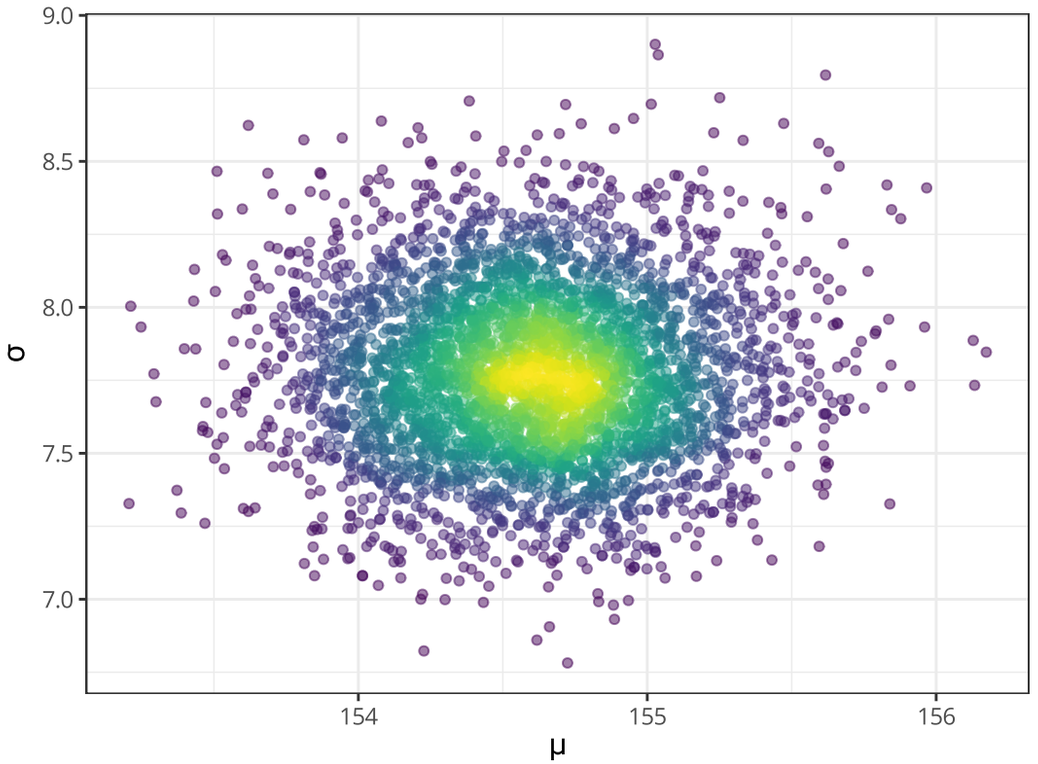
```
summary(mod3)
```

```
 Family: gaussian
  Links: mu = identity
Formula: height ~ 1
   Data: d2 (Number of observations: 352)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Regression Coefficients:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   177.86      0.10   177.67   178.06 1.00     3052     2762

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma    24.60      0.96    22.83    26.57 1.00     3469     2267

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

On remarque que la valeur estimée pour μ n'a presque pas "bougée" du prior...mais on remarque également que la valeur estimée pour σ a largement augmentée. Nous avons dit au modèle que nous étions assez certain de notre valeur de μ, le modèle s'est ensuite "adapté", ce qui explique la valeur de σ...

Visualiser les échantillons de la distribution postérieure

```r
post <- as_draws_df(x = mod2) %>%
    mutate(density = get_density(x = b_Intercept, y = sigma, n = 1e2) )

post %>%
    ggplot(aes(x = b_Intercept, y = sigma, color = density) ) +
    geom_point(size = 2, alpha = 0.5, show.legend = FALSE) +
    labs(x = expression(mu), y = expression(sigma) ) +
    viridis::scale_color_viridis()
```



Récupérer les échantillons de la distribution postérieure

```r
# gets the first 6 samples
head(post)
```

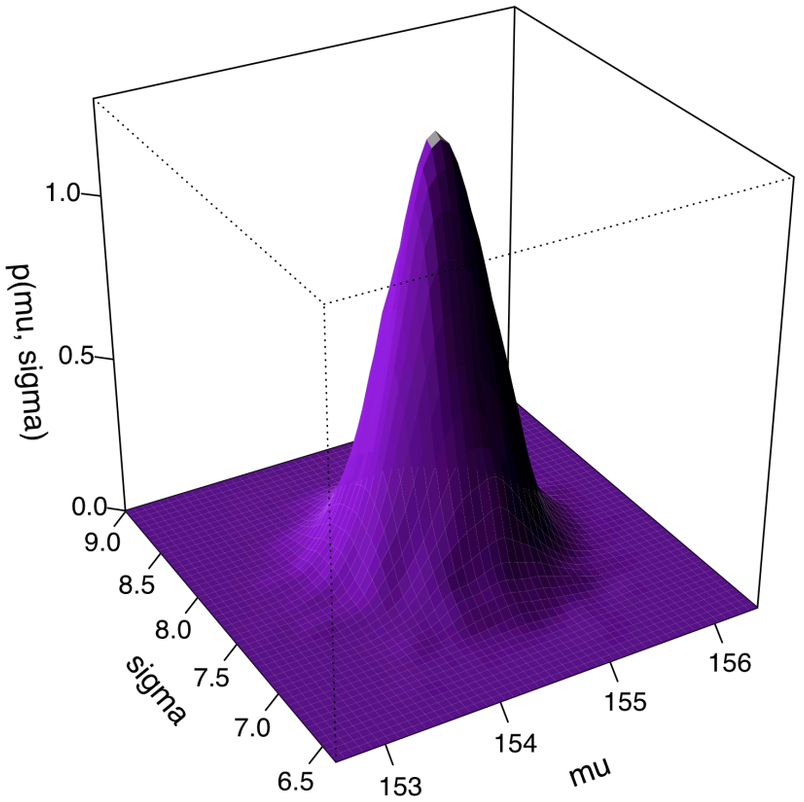
```
# A draws_df: 6 iterations, 1 chains, and 6 variables
  b_Intercept sigma Intercept lprior  lp__ density
1         155   8.1       155   -9.3 -1228    0.38
2         154   7.4       154   -9.3 -1228    0.44
3         155   7.4       155   -9.3 -1228    0.41
4         155   7.8       155   -9.3 -1227    1.35
5         155   8.0       155   -9.3 -1227    0.84
6         154   8.1       154   -9.3 -1227    0.65
# ... hidden reserved variables {'.chain', '.iteration', '.draw'}
```

```r
# gets the median and the 95% credible interval
t(sapply(post[, 1:2], quantile, probs = c(0.025, 0.5, 0.975) ) )
```

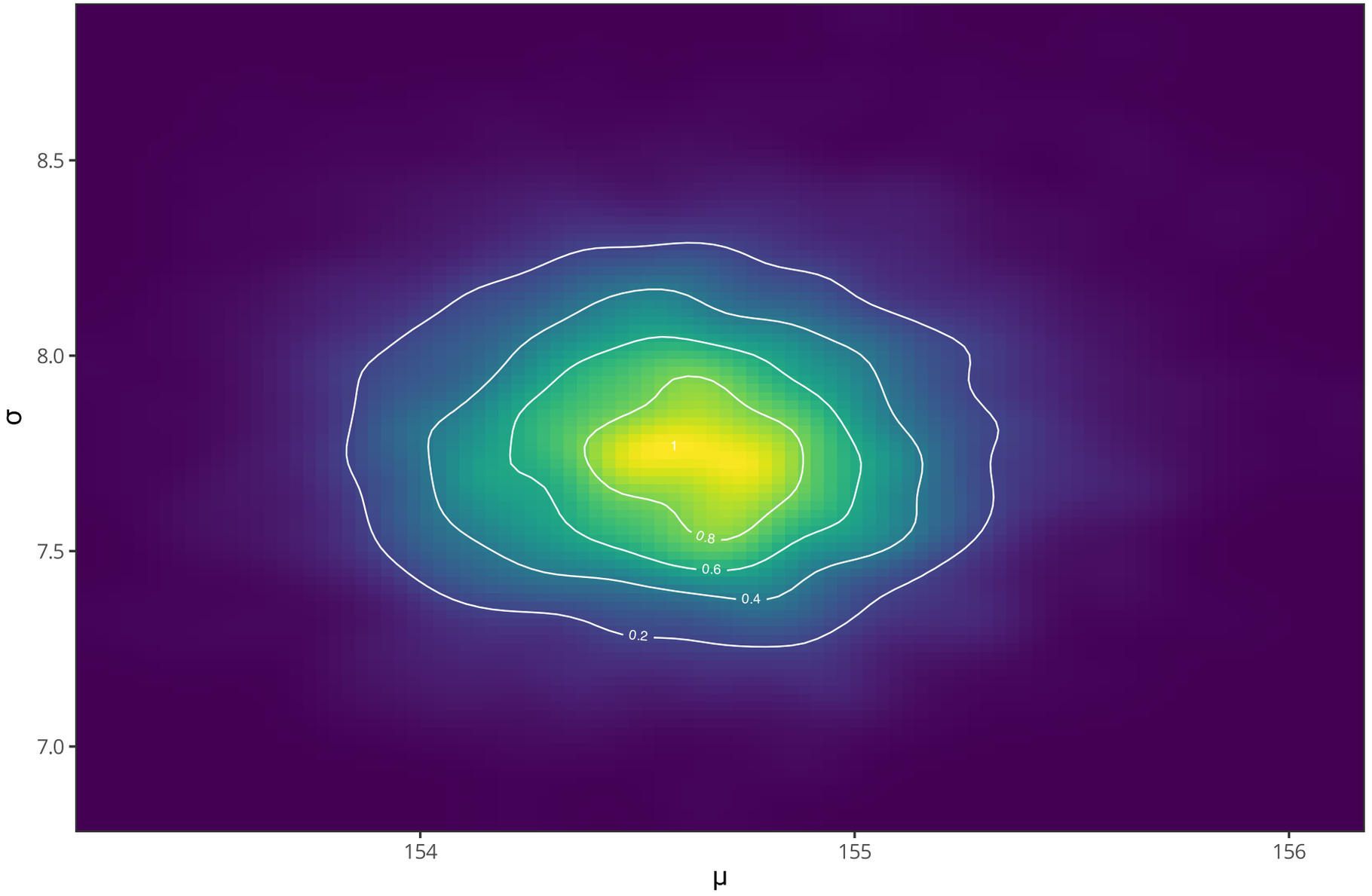
```
                2.5%        50%      97.5%
b_Intercept 153.784678 154.606848 155.443010
sigma         7.208937   7.756117   8.363607
```

Visualiser la distribution postérieure

```r
H.scv <- Hscv(post[, 1:2])
fhat_post <- kde(x = post[, 1:2], H = H.scv, compute.cont = TRUE)

plot(fhat_post, display = "persp", col = "purple", border = NA,
  xlab = "\nmu", ylab = "\nsigma", zlab = "\np(mu, sigma)",
  shade = 0.8, phi = 30, ticktype = "detailed",
  cex.lab = 1.2, family = "Helvetica")
```



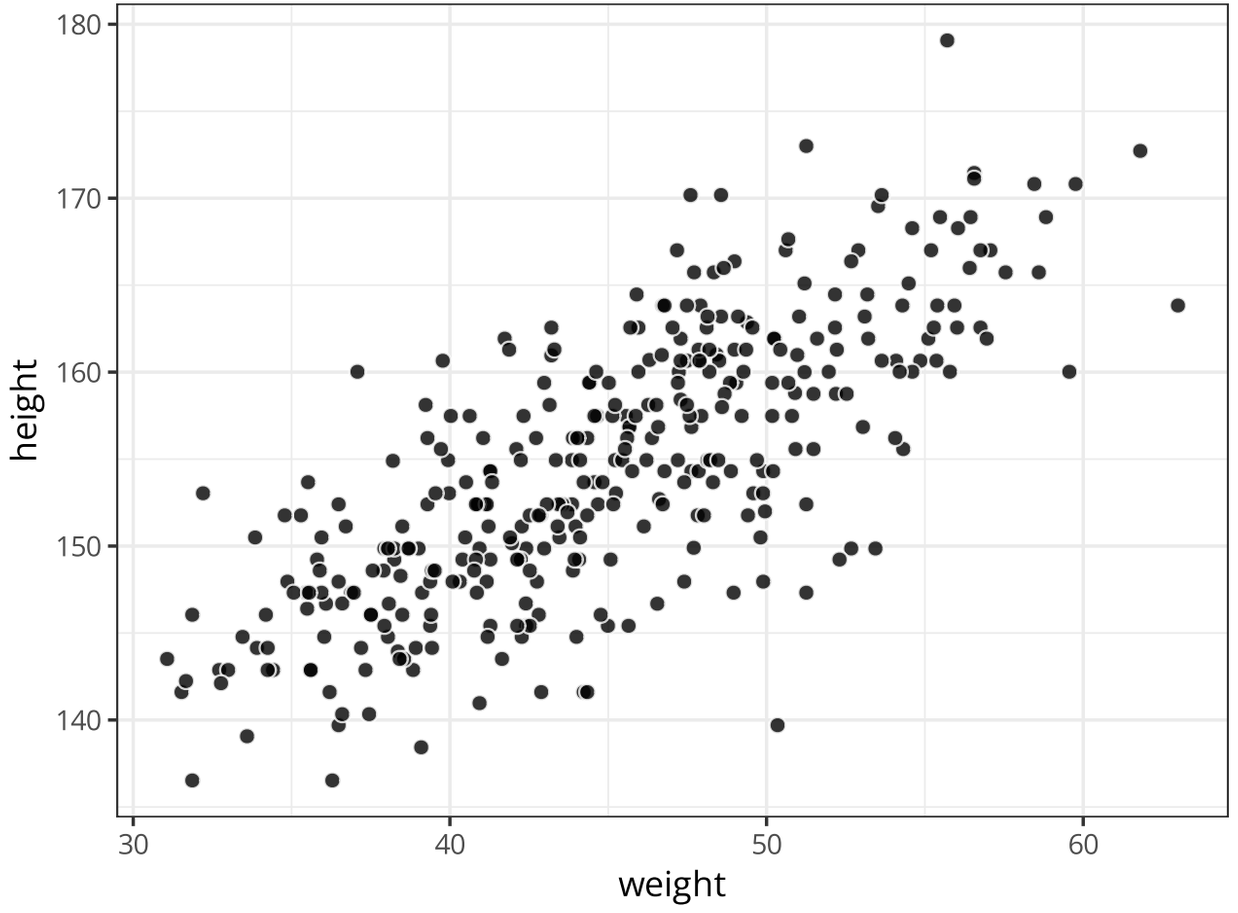
Visualiser la distribution postérieure



Ajouter un prédicteur

Comment est-ce que la taille co-varie avec le poids ?

```
1  d2 %>%
2    ggplot(aes(x = weight, y = height) ) +
3    geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8)
```

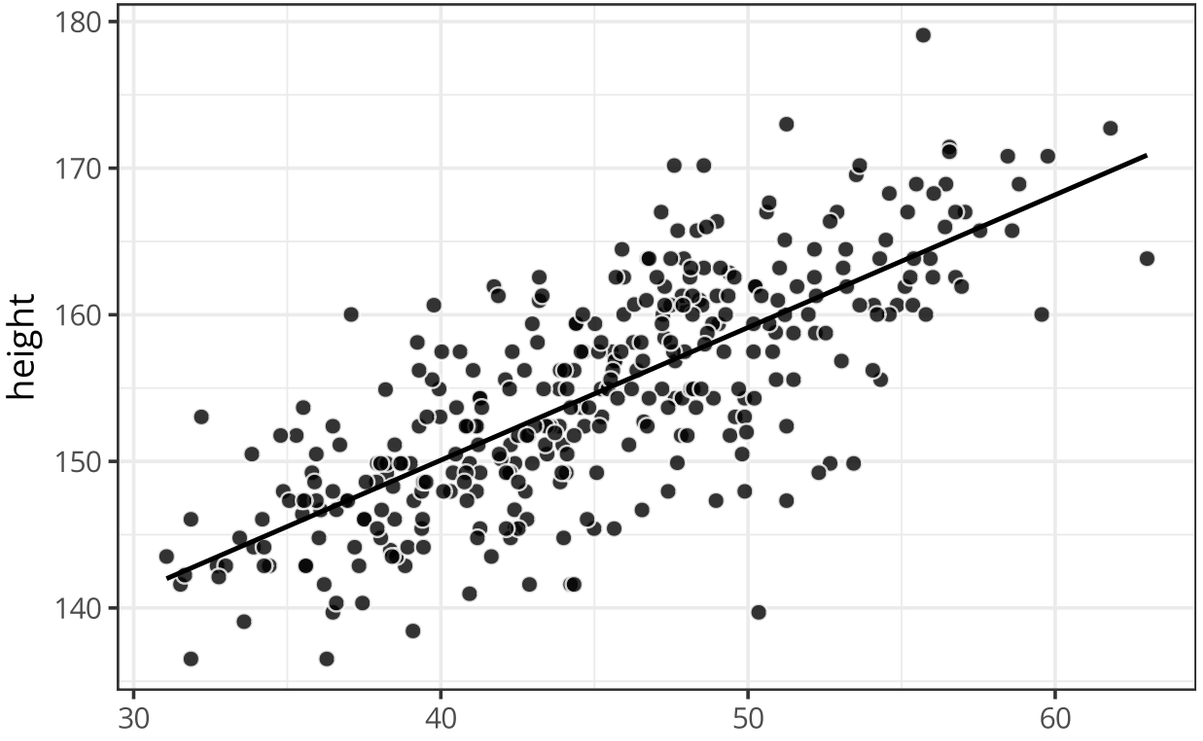


Régression linéaire à un prédicteur continu

$$h_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta x_i$$

```
1 linear_model <- lm(height ~ weight, data = d2)
2 rethinking::precis(linear_model, prob = 0.95)
```

```
                  mean         sd        2.5%       97.5%
(Intercept) 113.8793936 1.91106523 110.1337746 117.6250126
weight        0.9050291 0.04204752   0.8226175   0.9874407
```



Différentes notations

On considère un modèle de régression linéaire avec un seul prédicteur, une pente, un intercept, et des résidus distribués selon une loi normale. La notation :

$$h_i = \alpha + \beta x_i + \epsilon_i \quad \text{avec} \quad \epsilon_i \sim \text{Normal}(0, \sigma)$$

est équivalente à :

$$h_i - (\alpha + \beta x_i) \sim \text{Normal}(0, \sigma)$$

et si on réduit encore un peu :

$$h_i \sim \text{Normal}(\alpha + \beta x_i, \sigma).$$

Les notations ci-dessus sont équivalentes, mais la dernière est plus flexible, et nous permettra par la suite de l'étendre plus simplement aux modèles multi-niveaux.

Régression linéaire à un prédicteur continu

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta x_i \\
\alpha &\sim \text{Normal}(178, 20) \\
\beta &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{Exponential}(0.01)
\end{aligned}
$$

Dans ce modèle μ n'est plus un paramètre à estimer (car μ est **déterminé** par α et β). À la place, nous allons estimer α et β.

Rappels : α est l'intercept, c'est à dire la taille attendue, lorsque le poids est égal à 0. β est la pente, c'est à dire le changement de taille attendu quand le poids augmente d'une unité.

Régression linéaire à un prédicteur continu

```
priors <- c(
  prior(normal(178, 20), class = Intercept),
  prior(normal(0, 10), class = b),
  prior(exponential(0.01), class = sigma)
  )

mod4 <- brm(
  height ~ 1 + weight,
  prior = priors,
  family = gaussian(),
  data = d2
  )
```

Régression linéaire à un prédicteur continu

```
posterior_summary(x = mod4)
```

	Estimate	Est.Error	Q2.5	Q97.5
b_Intercept	113.8223425	1.94536342	110.1022497	117.5569179
b_weight	0.9063566	0.04274216	0.8234107	0.9895307
sigma	5.1065858	0.19260573	4.7465686	5.4968105
Intercept	154.5997640	0.27606882	154.0527761	155.1404293
lprior	-12.4811061	0.01619935	-12.5137830	-12.4493585
lp__	-1083.3937957	1.24317339	-1086.6590651	-1081.9799832

- α = 113.82, 95% CrI [110.1, 117.56] représente la taille moyenne quand le poids est égal à 0kg...
- β = 0.91, 95% CrI [0.82, 0.99] nous indique qu'une augmentation de 1kg entraîne une augmentation de 0.91cm.

Régression linéaire à un prédicteur continu

```r
d2$weight.c <- d2$weight - mean(d2$weight)

mod5 <- brm(
  height ~ 1 + weight.c,
  prior = priors,
  family = gaussian(),
  data = d2
  )
```

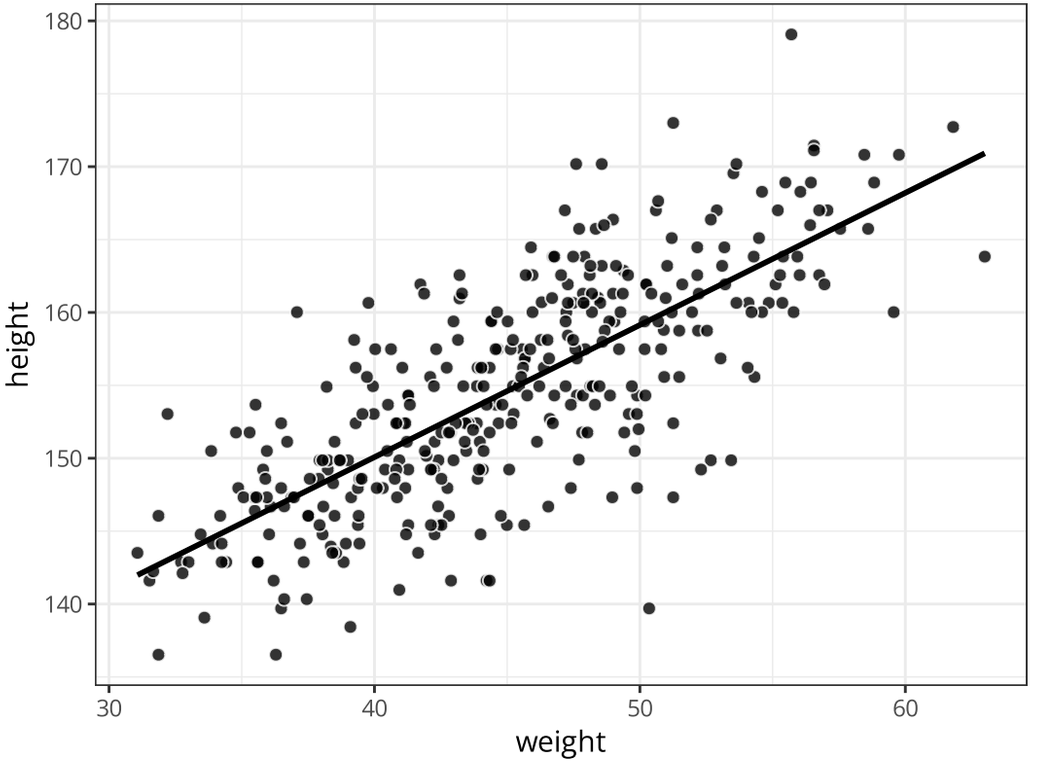
```r
fixef(mod5) # retrieves the fixed effects estimates
```

```
            Estimate Est.Error       Q2.5       Q97.5
Intercept 154.6011527 0.2749376 154.053646 155.1385527
weight.c    0.9056969 0.0418650   0.824197   0.9876061
```

Après avoir centré la réponse, l'intercept représente désormais la valeur attendue de taille (en cm) lorsque le poids est à sa valeur moyenne.

Représenter les prédictions du modèle

```r
d2 %>%
    ggplot(aes(x = weight, y = height) ) +
    geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
    geom_smooth(
        data = data.frame(fitted(object = mod4) ) %>% bind_cols(data.frame(weight = d2$weight) ),
        aes(y = Estimate), stat = "identity",
        se = FALSE, color = "black"
        )
```



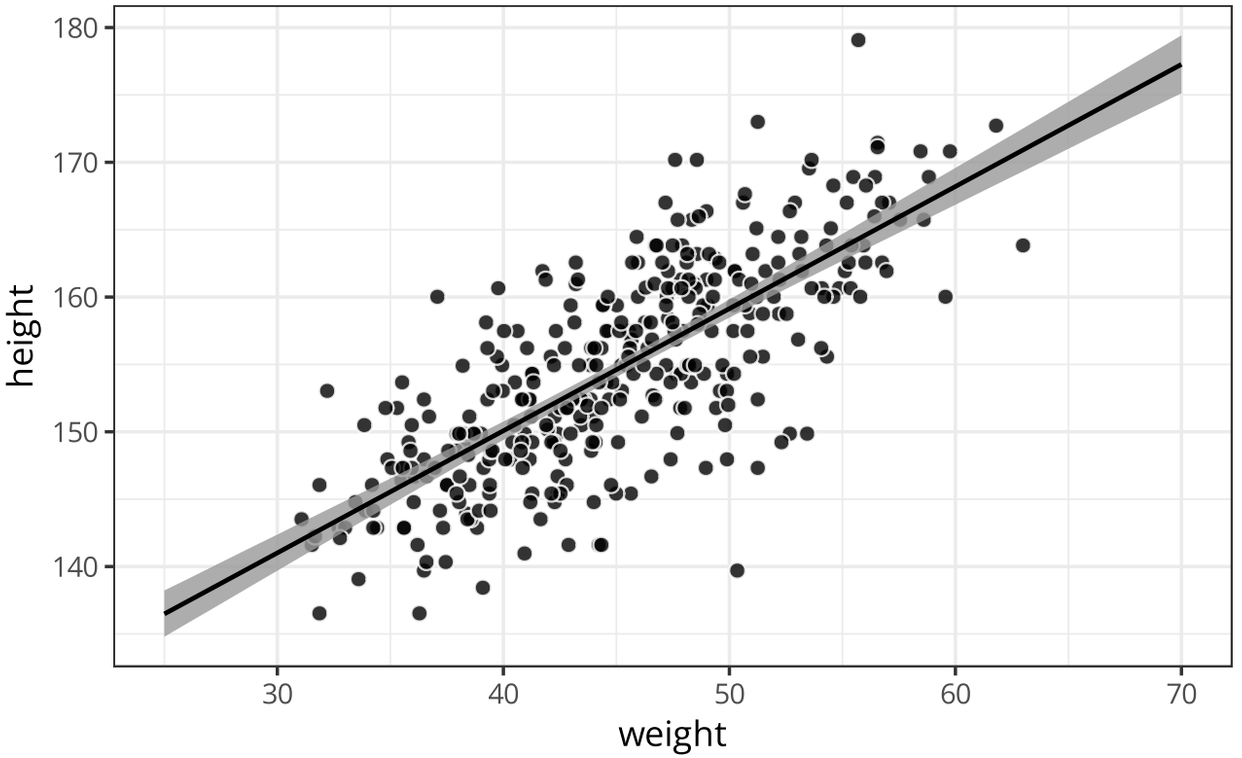
Représenter l'incertitude sur μ via fitted()

```r
# on crée un vecteur de valeurs possibles pour "weight"
weight.seq <- data.frame(weight = seq(from = 25, to = 70, by = 1) )

# on récupère les prédictions du modèle pour ces valeurs de poids
mu <- data.frame(fitted(mod4, newdata = weight.seq) ) %>% bind_cols(weight.seq)

# on affiche les 10 premières lignes de mu
head(mu, 10)
```

```
   Estimate Est.Error     Q2.5    Q97.5 weight
1  136.4813 0.9004813 134.7882 138.2262     25
2  137.3876 0.8598970 135.7685 139.0476     26
3  138.2940 0.8195322 136.7414 139.8819     27
4  139.2003 0.7794208 137.7247 140.7202     28
5  140.1067 0.7396043 138.7121 141.5522     29
6  141.0130 0.7001328 139.6897 142.3698     30
7  141.9194 0.6610682 140.6746 143.2128     31
8  142.8258 0.6224871 141.6457 144.0435     32
9  143.7321 0.5844853 142.6259 144.8726     33
10 144.6385 0.5471834 143.5975 145.6963     34
```

Représenter l'incertitude sur μ via fitted()

```
d2 %>%
  ggplot(aes(x = weight, y = height) ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
  geom_smooth(
    data = mu, aes(y = Estimate, ymin = Q2.5, ymax = Q97.5),
    stat = "identity",
    color = "black", alpha = 0.8, size = 1
    )
```



Intervalles de prédiction (incorporer σ)

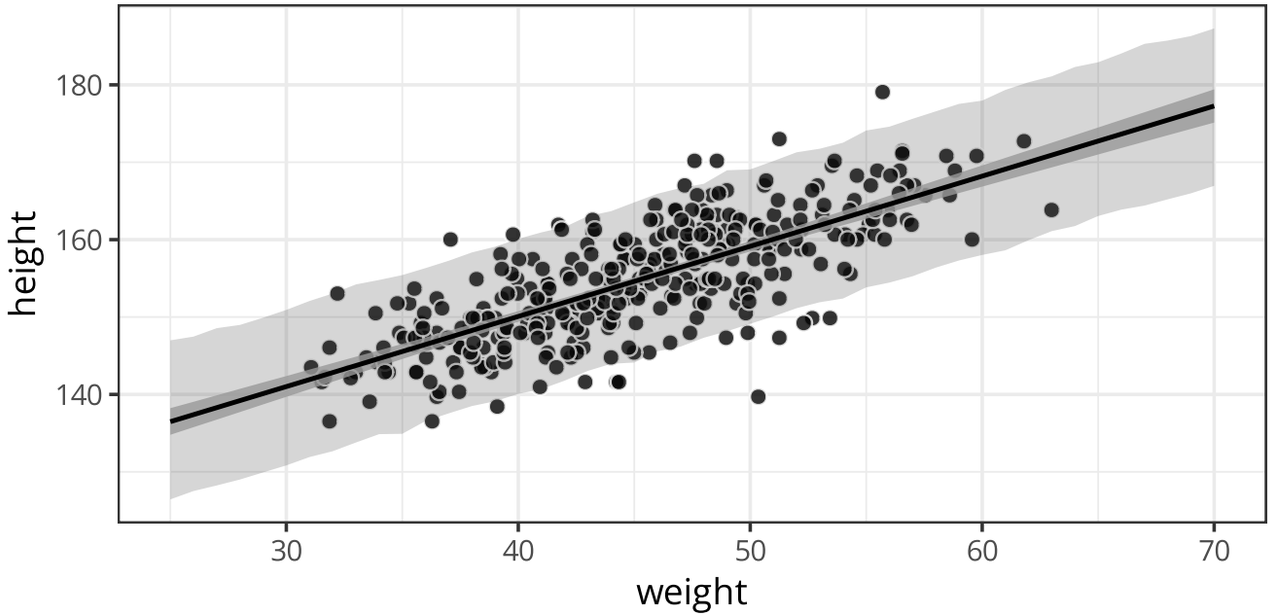
Pour rappel, voici notre modèle : $h_i \sim \text{Normal}(\alpha + \beta x_i, \sigma)$. Pour l'instant, on a seulement représenté les prédictions pour μ. Comment incorporer σ dans nos prédictions ?

```r
# on crée un vecteur de valeurs possibles pour "weight"
weight.seq <- data.frame(weight = seq(from = 25, to = 70, by = 1) )

# on récupère les prédictions du modèle pour ces valeurs de poids
pred_height <- data.frame(predict(mod4, newdata = weight.seq) ) %>% bind_cols(weight.seq)

# on affiche les 10 premières lignes de pred_height
head(pred_height, 10)
```

```
   Estimate Est.Error     Q2.5    Q97.5 weight
1  136.5780  5.240715 126.4420 146.9880     25
2  137.3608  5.225315 127.5202 147.4722     26
3  138.3976  5.228718 128.2392 148.5616     27
4  139.1763  5.168828 129.0059 148.9819     28
5  140.1587  5.079376 129.9234 149.9117     29
6  140.9944  5.153864 130.8329 150.9225     30
7  142.0068  5.155825 131.9084 152.0734     31
8  142.8773  5.194414 132.6620 153.0379     32
9  143.8315  5.217857 133.7898 154.2138     33
10 144.8128  5.065709 134.8662 154.7888     34
```

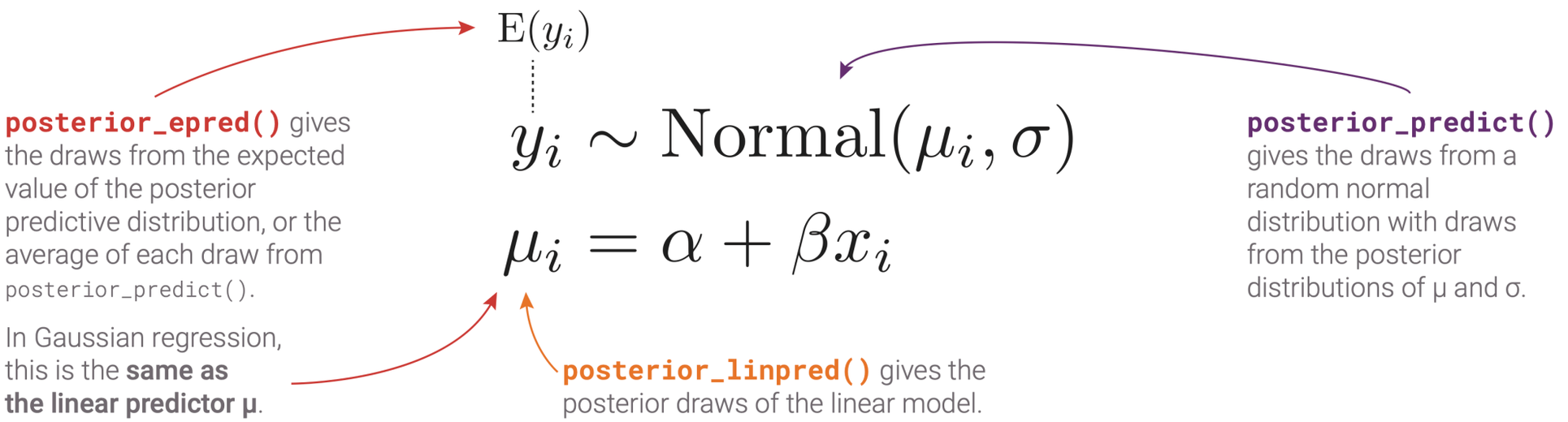
Intervalles de prédiction (incorporer σ)

```r
d2 %>%
  ggplot(aes(x = weight, y = height) ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
  geom_ribbon(
    data = pred_height, aes(x = weight, ymin = Q2.5, ymax = Q97.5),
    alpha = 0.2, inherit.aes = FALSE
    ) +
  geom_smooth(
    data = mu, aes(y = Estimate, ymin = Q2.5, ymax = Q97.5),
    stat = "identity", color = "black", alpha = 0.8, size = 1
    )
```



Built-in brms functions

Le paquet `brms` propose aussi les fonctions `posterior_epred()`, `posterior_linpred()`, et `posterior_predict()`, qui permettent de générer des prédictions à partir de modèles fittés avc `brms`. Andrew Heiss décrit de manière détaillée le fonctionnement de ces fonction dans cet article de blog.



Rappel : Deux types d'incertitude

Deux sources d'incertitude dans le modèle : incertitude concernant l'estimation de la valeur des paramètres mais également concernant le processus d'échantillonnage.

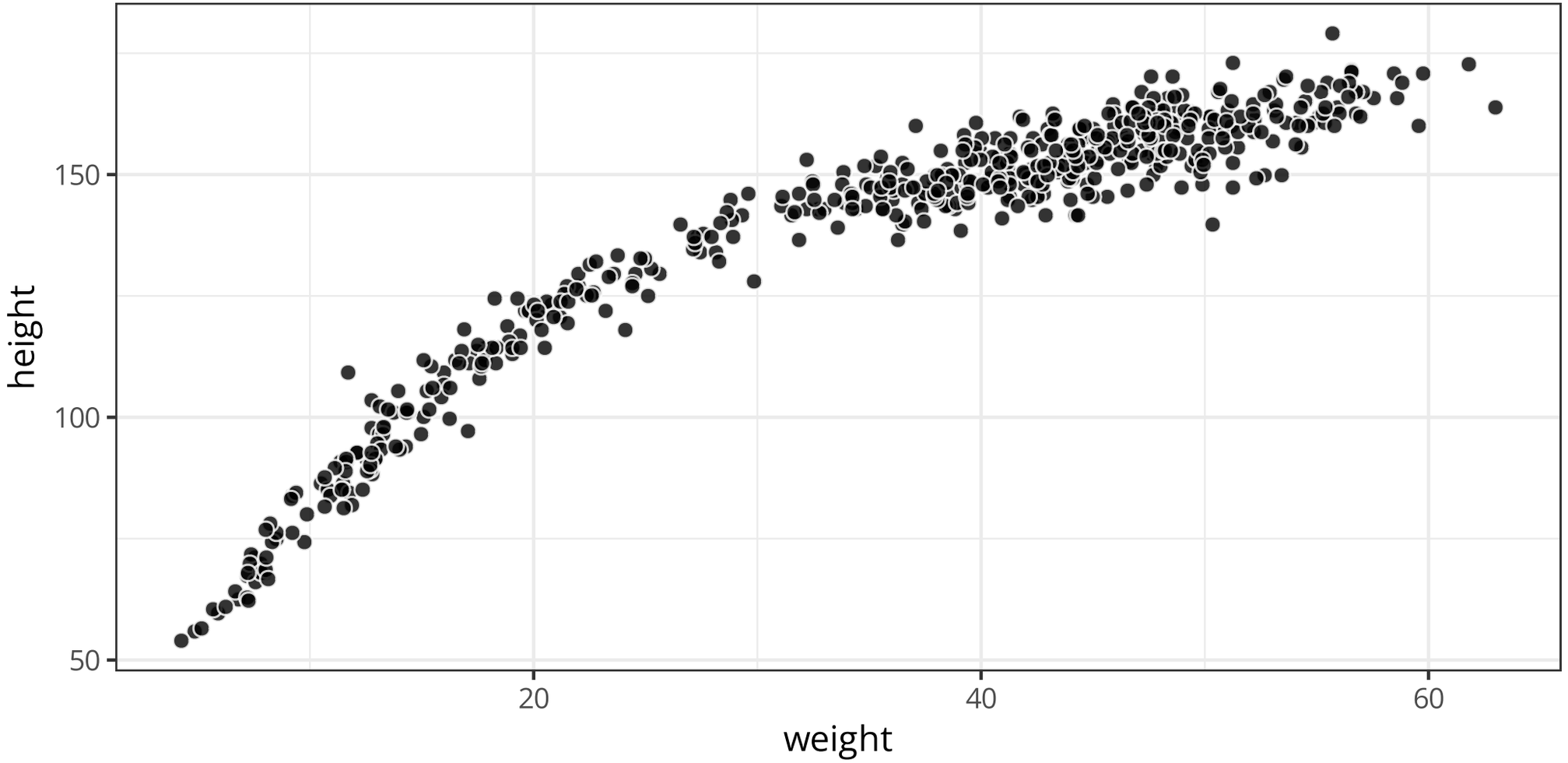
Incertitude épistémique : La distribution a posteriori ordonne toutes les combinaisons possibles des valeurs des paramètres selon leurs plausibilités relatives.

Incertitude aléatoire : La distribution des données simulées est elle, une distribution qui contient de l'incertitude liée à un processus d'échantillonnage (i.e., générer des données à partir d'une gaussienne).

Voir aussi ce court article par O'Hagan (2004).

Régression polynomiale

```r
d %>% # on utilise d au lieu de d2
  ggplot(aes(x = weight, y = height) ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8)
```

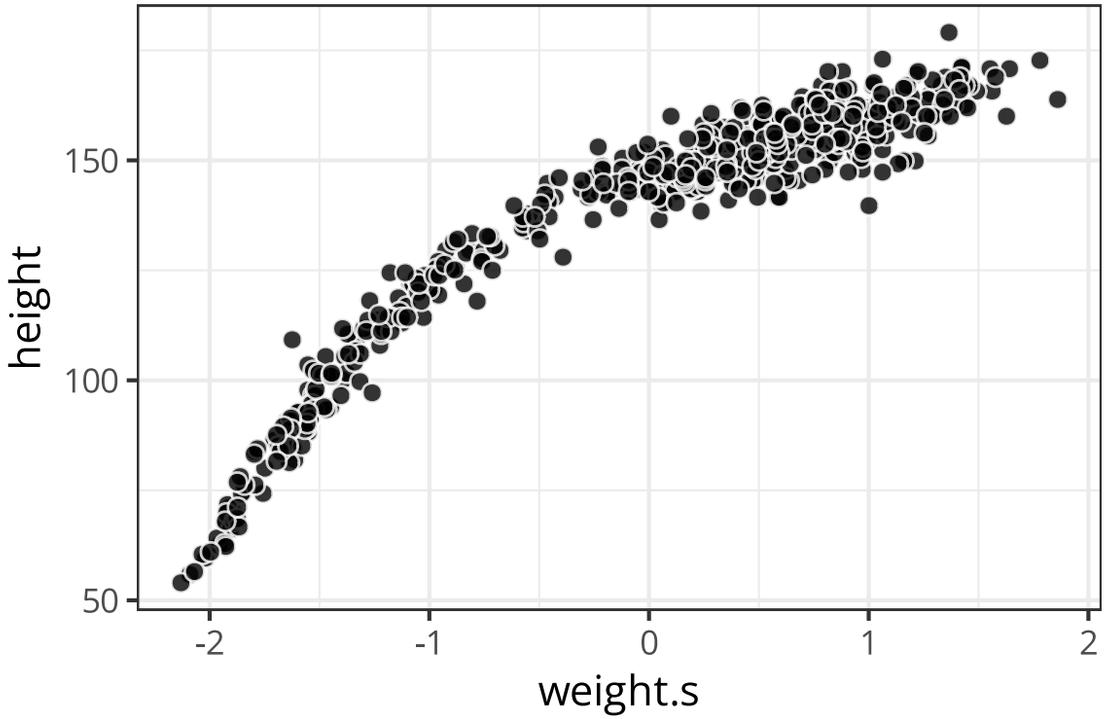


Scores standardisés

```r
d <- d %>% mutate(weight.s = (weight - mean(weight) ) / sd(weight) )

d %>%
    ggplot(aes(x = weight.s, y = height) ) +
    geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8)
```

```r
c(mean(d$weight.s), sd(d$weight.s) )
```

```
[1] 3.346996e-17 1.000000e+00
```



Scores standardisés

Pourquoi standardiser les prédicteurs ?

- **Interprétation**. Permet de comparer les coefficients de plusieurs prédicteurs. Un changement d'un écart-type du prédicteur correspond à un changement d'un écart-type sur la réponse (si la réponse est aussi standardisée).
- **Fitting**. Quand les prédicteurs contiennent de grandes valeurs (ou des valeurs trop différentes les unes des autres), cela peut poser des problèmes de convergence (cf. Cours n°05).

Modèle de régression polynomiale - Exercice

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 \\
\alpha &\sim \text{Normal}(156, 100) \\
\beta_1, \beta_2 &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{Exponential}(0.01)
\end{aligned}
$$

À vous de construire et fitter ce modèle en utilisant `brms::brm()`.

Modèle de régression polynomiale

```r
priors <- c(
  prior(normal(156, 100), class = Intercept),
  prior(normal(0, 10), class = b),
  prior(exponential(0.01), class = sigma)
  )

mod6 <- brm(
  # NB: polynomials should be written with the I() function...
  height ~ 1 + weight.s + I(weight.s^2),
  prior = priors,
  family = gaussian(),
  data = d
  )
```

Modèle de régression polynomiale

```
1 summary(mod6)
```

```
 Family: gaussian
  Links: mu = identity
Formula: height ~ 1 + weight.s + I(weight.s^2)
   Data: d (Number of observations: 544)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Regression Coefficients:
            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept     146.67      0.38   145.91   147.43 1.00     3641     2498
weight.s       21.40      0.29    20.81    21.98 1.00     3430     2938
Iweight.sE2    -8.41      0.29    -8.98    -7.82 1.00     3480     2941

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     5.77      0.18     5.45     6.13 1.00     3519     2764

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

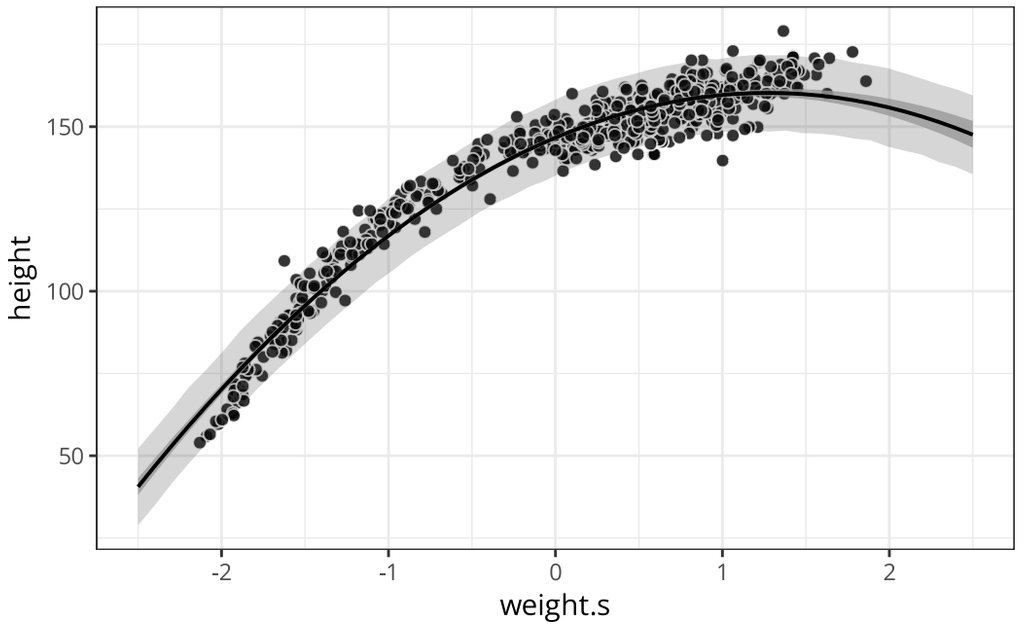
Représenter les prédictions du modèle

```r
# on crée un vecteur de valeurs possibles pour "weight"
weight.seq <- data.frame(weight.s = seq(from = -2.5, to = 2.5, length.out = 50) )

# on récupère les prédictions du modèle pour ces valeurs de poids
mu <- data.frame(fitted(mod6, newdata = weight.seq) ) %>% bind_cols(weight.seq)
pred_height <- data.frame(predict(mod6, newdata = weight.seq) ) %>% bind_cols(weight.seq)

# on affiche les 10 premières lignes de pred_height
head(pred_height, 10)
```

```
   Estimate Est.Error     Q2.5     Q97.5  weight.s
1  40.64504  5.869147 28.87838  52.18481 -2.500000
2  46.69845  5.990789 35.07197  58.26570 -2.397959
3  53.08186  5.764818 41.74931  64.38716 -2.295918
4  59.25501  5.824973 47.82025  70.90877 -2.193878
5  65.00461  5.771099 53.60479  76.11664 -2.091837
6  70.81054  5.780675 59.69612  81.83442 -1.989796
7  76.25468  5.845950 64.72576  87.98512 -1.887755
8  81.75858  5.784779 70.29188  93.17858 -1.785714
9  86.67803  5.780881 75.33656  98.02182 -1.683673
10 91.58160  5.790394 80.16872 102.80118 -1.581633
```

Représenter les prédictions du modèle

```
d %>%
  ggplot(aes(x = weight.s, y = height) ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
  geom_ribbon(
    data = pred_height, aes(x = weight.s, ymin = Q2.5, ymax = Q97.5),
    alpha = 0.2, inherit.aes = FALSE
    ) +
  geom_smooth(
    data = mu, aes(y = Estimate, ymin = Q2.5, ymax = Q97.5),
    stat = "identity", color = "black", alpha = 0.8, size = 1
    )
```



Modèle de régression, taille d'effet

Plusieurs méthodes pour calculer les tailles d'effet dans les modèles bayésiens. Gelman & Pardoe (2006) proposent une méthode pour calculer un R^2 basé sur l'échantillon.

Marsman & Wagenmakers (2017) et Marsman et al. (2019) généralisent des méthodes existantes pour calculer un ρ^2 pour les designs de type ANOVA (i.e., avec prédicteurs catégoriels), qui représente une estimation de la taille d'effet dans la population (et non basée sur l'échantillon).

> “
>
> Similar to most of the ES measures that have been proposed for the ANOVA model, the squared multiple correlation coefficient ρ^2 [...] is a so-called proportional reduction in error measure (PRE). In general, a PRE measure expresses the proportion of the variance in an outcome y that is attributed to the independent variables x (Marsman et al., 2019).

Modèle de régression, taille d'effet

$$\rho^2 = \frac{\sum_{i=1}^{n} \pi_i(\beta_i - \beta)^2}{\sigma^2 + \sum_{i=1}^{n} \pi_i(\beta_i - \beta)^2}$$

$$\rho^2 = \frac{\frac{1}{n}\sum_{i=1}^{n} \beta_i^2}{\sigma^2 + \frac{1}{n}\sum_{i=1}^{n} \beta_i^2}$$

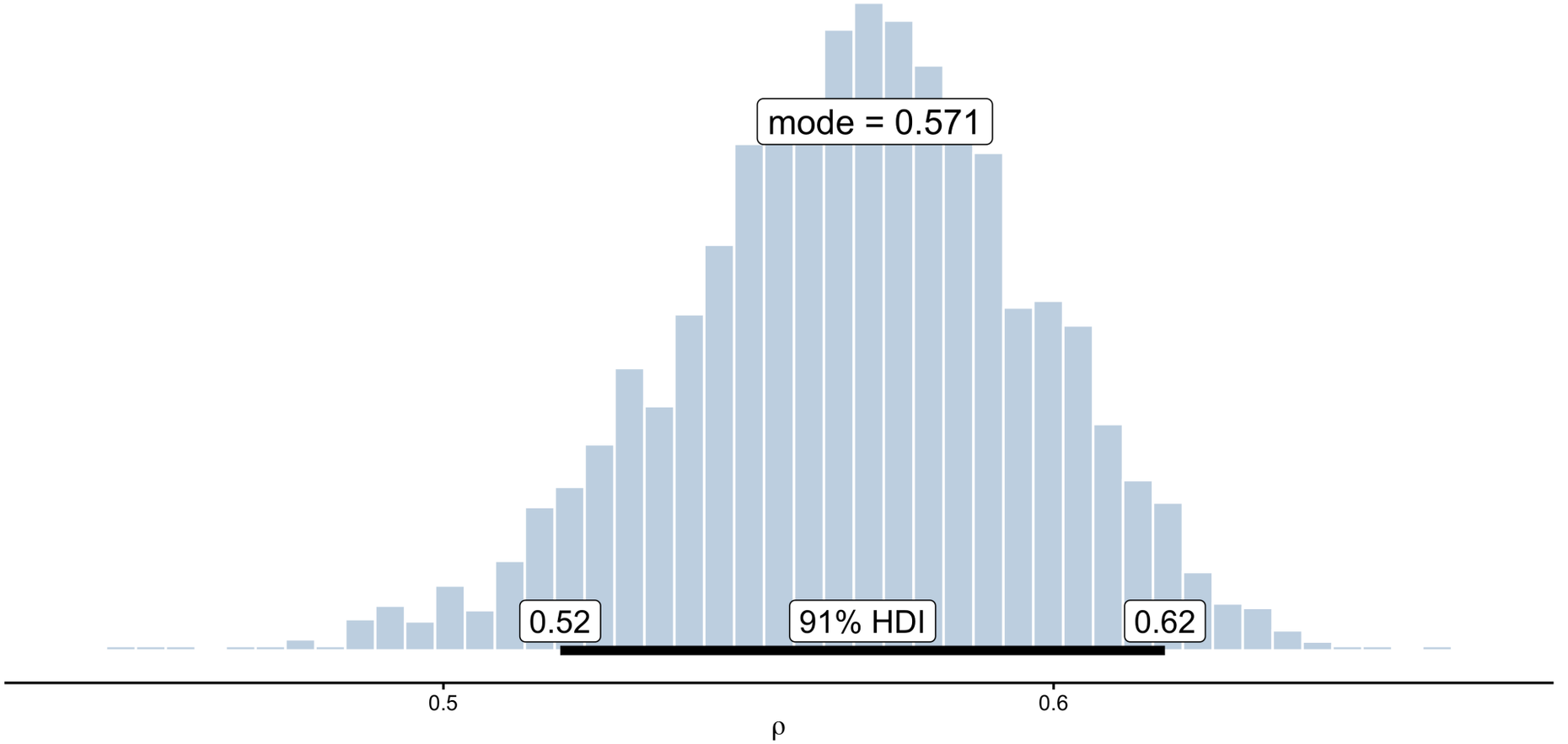
$$\rho^2 = \frac{\beta^2\tau^2}{\sigma^2 + \beta^2\tau^2}$$

```
post <- as_draws_df(x = mod4)
beta <- post$b_weight
sigma <- post$sigma
rho <- beta^2 * var(d2$weight) / (sigma^2 + beta^2 * var(d2$weight) )
```

Attention, si plusieurs prédicteurs, dépend de la structure de covariance...

Modèle de régression, taille d'effet

```
1  posterior_plot(samples = rho, usemode = TRUE) + labs(x = expression(rho) )
```




```
1  summary(lm(height ~ weight, data = d2) )$r.squared
```

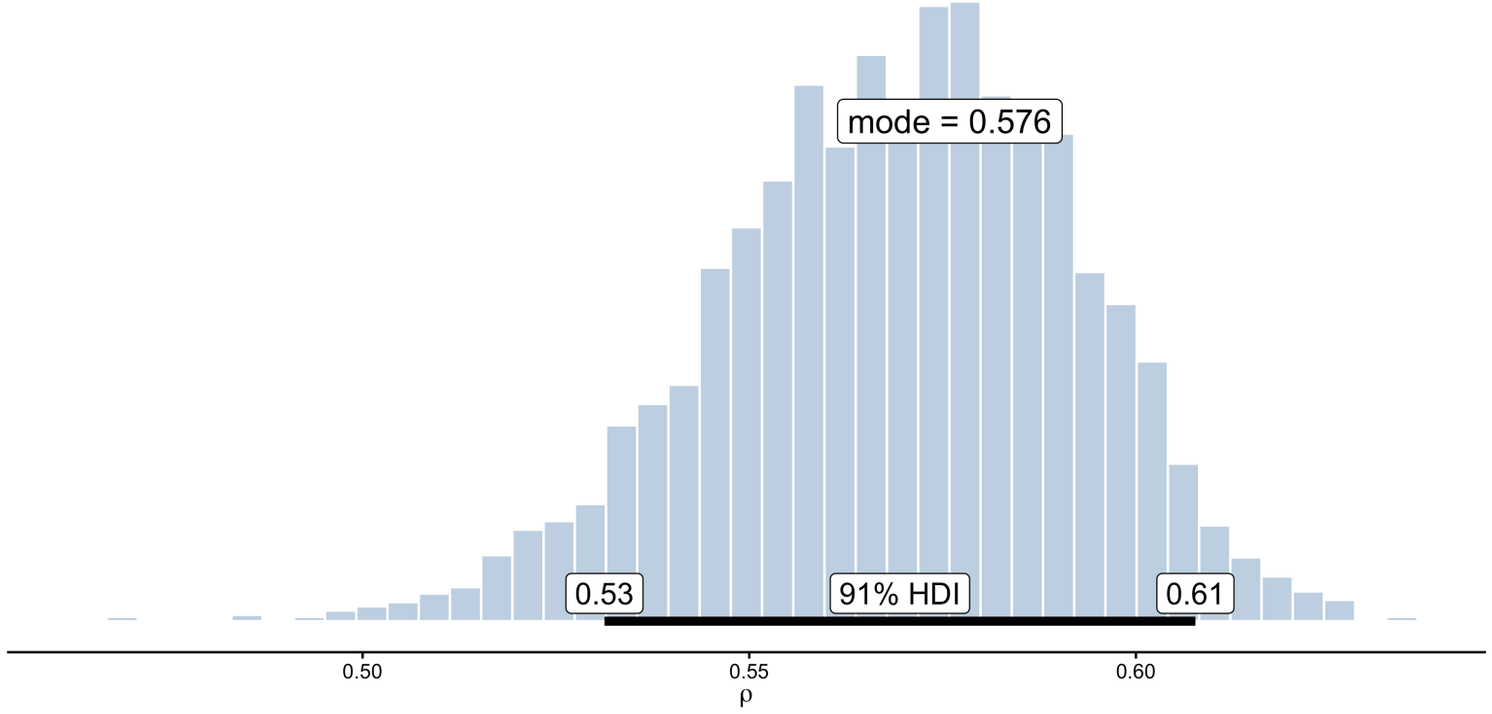
```
[1] 0.5696444
```

Modèle de régression, taille d'effet

```
1 bayes_R2(mod4)
```

```
   Estimate  Est.Error      Q2.5     Q97.5
R2 0.5689494 0.02318861 0.5201562 0.6100338
```

```
1 bayes_R2(mod4, summary = FALSE)[, 1] %>%
2     posterior_plot(usemode = TRUE) +
3     labs(x = expression(rho) )
```



Résumé du cours

On a présenté un nouveau modèle à deux puis trois paramètres : le modèle gaussien, puis la régression linéaire gaussienne, permettant de mettre en relation deux variables continues.

Comme précédemment, le théorème de Bayes est utilisé pour mettre à jour nos connaissances a priori quant à la valeur des paramètres en une connaissance a posteriori, synthèse entre nos priors et l'information contenue dans les données.

La package `brms` permet de fitter toutes sortes de modèles avec une syntaxe similaire à celle utilisée par `lm()`.

La fonction `fitted()` permet de récupérer les prédictions d'un modèle fitté avec `brms`.

La fonction `predict()` permet de simuler des données à partir d'un modèle fitté avec `brms`.

Travaux pratiques - 1/2

Sélectionner toutes les lignes du jeu de données `howell` correspondant à des individus mineurs (age < 18). Cela devrait résulter en une dataframe de 192 lignes.

Fitter un modèle de régression linéaire en utilisant la fonction `brms::brm()`. Reporter et interpréter les estimations de ce modèle. Pour une augmentation de 10 unités de `weight`, quelle augmentation de taille (`height`) le modèle prédit-il ?

Faire un plot des données brutes avec le poids sur l'axe des abscisses et la taille sur l'axe des ordonnées. Surimposer la droite de régression du modèle et un intervalle de crédibilité à 89% pour la moyenne. Ajouter un intervalle de crédibilité à 89% pour les tailles prédites.

Que pensez-vous du "fit" du modèle ? Quelles conditions d'application du modèle seriez-vous prêt.e.s à changer, afin d'améliorer le modèle ?

Travaux pratiques - 2/2

Imaginons que vous ayez consulté une collègue experte en allométrie (i.e., les phénomènes de croissance différentielle d'organes) et que cette dernière vous explique que ça ne fait aucun sens de modéliser la relation entre le poids et la taille... alors qu'on sait que c'est le logarithme du poids qui est relié (linéairement) à la taille !

Modéliser alors la relation entre la taille (cm) et le log du poids (log-kg). Utiliser la dataframe `howell` en entier (les 544 lignes). Fitter le modèle suivant en utilisant `brms::brm()`.

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta \cdot \log(w_i) \\
\alpha &\sim \text{Normal}(178, 100) \\
\beta &\sim \text{Normal}(0, 100) \\
\sigma &\sim \text{Exponential}(0.01)
\end{aligned}
$$

Où h_i est la taille de l'individu i et w_i le poids de l'individu i. La fonction pour calculer le log en `R` est simplement `log()`. Est-ce que vous savez interpréter les résultats ? Indice : faire un plot des données brutes et surimposer les prédictions du modèle...

Proposition de solution

```r
# on garde seulement les individus ayant moins de 18 ans
d <- open_data(howell) %>% filter(age < 18)

priors <- c(
  prior(normal(150, 100), class = Intercept),
  prior(normal(0, 10), class = b),
  prior(exponential(0.01), class = sigma)
  )

mod7 <- brm(
  height ~ 1 + weight,
  prior = priors,
  family = gaussian(),
  data = d
  )
```

Proposition de solution

```
summary(mod7, prob = 0.89)
```

```
 Family: gaussian
  Links: mu = identity
Formula: height ~ 1 + weight
   Data: d (Number of observations: 192)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Regression Coefficients:
          Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
Intercept    58.27      1.40    56.04    60.52 1.00     3893     3053
weight        2.72      0.07     2.61     2.83 1.00     3790     3134

Further Distributional Parameters:
      Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
sigma     8.53      0.45     7.83     9.27 1.00     3694     2812

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

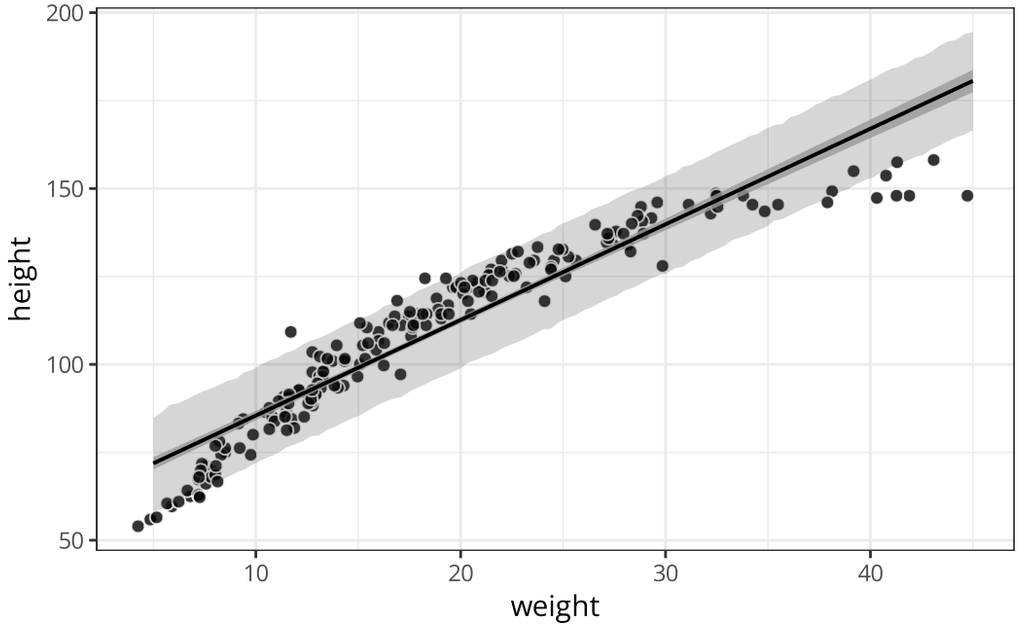
Représenter les prédictions du modèle

```r
# on crée un vecteur de valeurs possibles pour "weight"
weight.seq <- data.frame(weight = seq(from = 5, to = 45, length.out = 1e2) )

# on récupère les prédictions du modèle pour ces valeurs de poids
mu <- data.frame(
  fitted(mod7, newdata = weight.seq, probs = c(0.055, 0.945) )
  ) %>%
  bind_cols(weight.seq)

pred_height <- data.frame(
  predict(mod7, newdata = weight.seq, probs = c(0.055, 0.945) )
  ) %>%
  bind_cols(weight.seq)

# on affiche les 6 premières lignes de pred_height
head(pred_height)
```

```
  Estimate Est.Error     Q5.5    Q94.5   weight
1 71.73187  8.364933 58.23537 84.73636 5.000000
2 72.98623  8.705605 58.87403 86.45288 5.404040
3 74.11597  8.757565 60.18820 88.46508 5.808081
4 75.21958  8.678842 61.57353 88.99893 6.212121
5 76.24704  8.578545 62.57671 90.01098 6.616162
6 77.21991  8.831868 62.91383 91.25346 7.020202
```

Représenter les prédictions du modèle

```r
d %>%
  ggplot(aes(x = weight, y = height) ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
  geom_ribbon(
    data = pred_height, aes(x = weight, ymin = Q5.5, ymax = Q94.5),
    alpha = 0.2, inherit.aes = FALSE
    ) +
  geom_smooth(
    data = mu, aes(y = Estimate, ymin = Q5.5, ymax = Q94.5),
    stat = "identity", color = "black", alpha = 0.8, size = 1
    )
```



Proposition de solution

```r
# on considère maintenant tous les individus
d <- open_data(howell)

mod8 <- brm(
  # on prédit la taille par le logarithme du poids
  height ~ 1 + log(weight),
  prior = priors,
  family = gaussian(),
  data = d
  )
```

Proposition de solution

```
1 summary(mod8, prob = 0.89)
```

```
 Family: gaussian
  Links: mu = identity
Formula: height ~ 1 + log(weight)
   Data: d (Number of observations: 544)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Regression Coefficients:
          Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
Intercept   -23.59      1.34   -25.74   -21.48 1.00     3882     3256
logweight    47.02      0.38    46.42    47.64 1.00     3922     2878

Further Distributional Parameters:
      Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
sigma     5.16      0.16     4.91     5.41 1.00     3531     2943

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

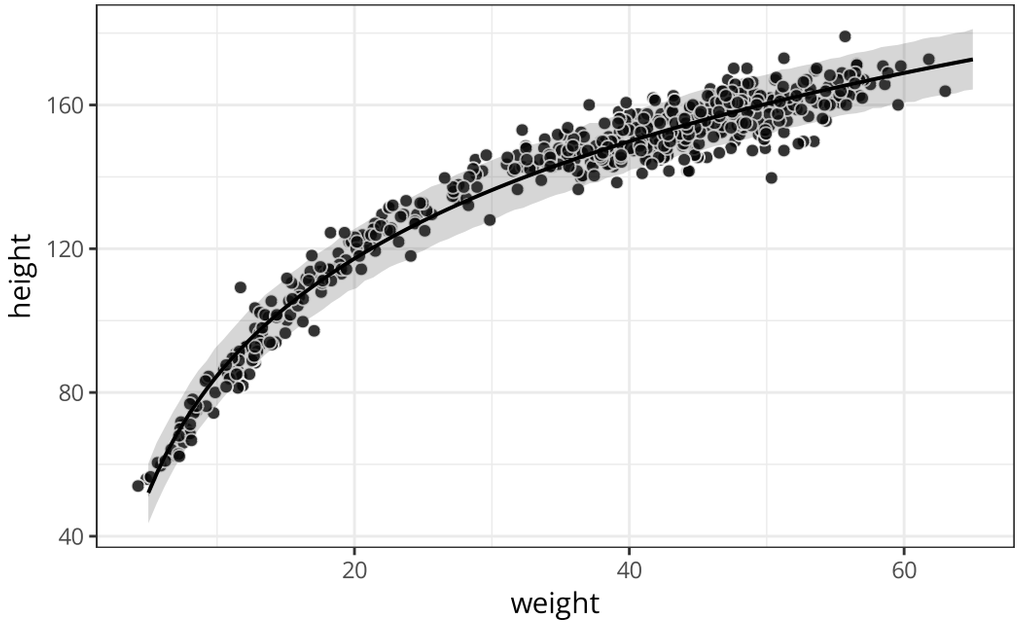
Représenter les prédictions du modèle

```r
# on crée un vecteur de valeurs possibles pour "weight"
weight.seq <- data.frame(weight = seq(from = 5, to = 65, length.out = 1e2) )

# on récupère les prédictions du modèle pour ces valeurs de poids
mu <- data.frame(
  fitted(mod8, newdata = weight.seq, probs = c(0.055, 0.945) )
  ) %>%
  bind_cols(weight.seq)

pred_height <- data.frame(
  predict(mod8, newdata = weight.seq, probs = c(0.055, 0.945) )
  ) %>%
  bind_cols(weight.seq)

# on affiche les 6 premières lignes de pred_height
head(pred_height)
```

```
  Estimate Est.Error     Q5.5    Q94.5   weight
1 52.04258  5.194183 43.61611 60.26030 5.000000
2 57.44268  5.304001 49.08526 66.05024 5.606061
3 62.24185  5.137826 54.03032 70.41580 6.212121
4 66.62461  5.080195 58.43134 74.71273 6.818182
5 70.60362  5.066627 62.61819 78.59577 7.424242
6 74.20932  5.213011 65.97021 82.64386 8.030303
```

Représenter les prédictions du modèle

```r
d %>%
  ggplot(aes(x = weight, y = height) ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
  geom_ribbon(
    data = pred_height, aes(x = weight, ymin = Q5.5, ymax = Q94.5),
    alpha = 0.2, inherit.aes = FALSE
    ) +
  geom_smooth(
    data = mu, aes(y = Estimate, ymin = Q5.5, ymax = Q94.5),
    stat = "identity", color = "black", alpha = 0.8, size = 1
    )
```



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

Gelman, A., & Pardoe, I. (2006). Bayesian measures of explained variance and pooling in multilevel (hierarchical) models. *Technometrics*, *48*(2), 241–251. https://doi.org/10.1198/004017005000000517

Marsman, M., & Wagenmakers, E.-J. (2017). Three Insights from a Bayesian Interpretation of the One-Sided *P* Value. *Educational and Psychological Measurement*, *77*(3), 529–539. https://doi.org/10.1177/0013164416669201

Marsman, M., Waldorp, L., Dablander, F., & Wagenmakers, E.-J. (2019). Bayesian estimation of explained variance in ANOVA designs. *Statistica Neerlandica*, *0*(0), 1–22. https://doi.org/10.1111/stan.12173

O'Hagan, T. (2004). Dicing with the unknown. *Significance*, *1*(3), 132–133. https://doi.org/10.1111/j.1740-9713.2004.00050.x