Introduction à la modélisation statistique bayésienne

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

Langage de la modélisation

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

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.

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 :

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*).

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 (*updating*), afin de calculer la distribution postérieure.

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 (*updating*), afin de calculer la distribution postérieure.
- Interpréter les estimations du modèle, évaluer ses prédictions, éventuellement modifier le modèle.

Un premier modèle

Un premier modèle

```
library(rethinking)
library(tidyverse)

data(Howell1)
d <- Howell1
str(d)
```

Un premier modèle

```
library(rethinking)
library(tidyverse)

data(Howell1)
d <- Howell1
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 ...
```

Un premier modèle

```
library(rethinking)
library(tidyverse)

data(Howell1)
d <- Howell1
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 ...
```

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

Un premier modèle

```
library(rethinking)
library(tidyverse)

data(Howell1)
d <- Howell1
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 ...
```

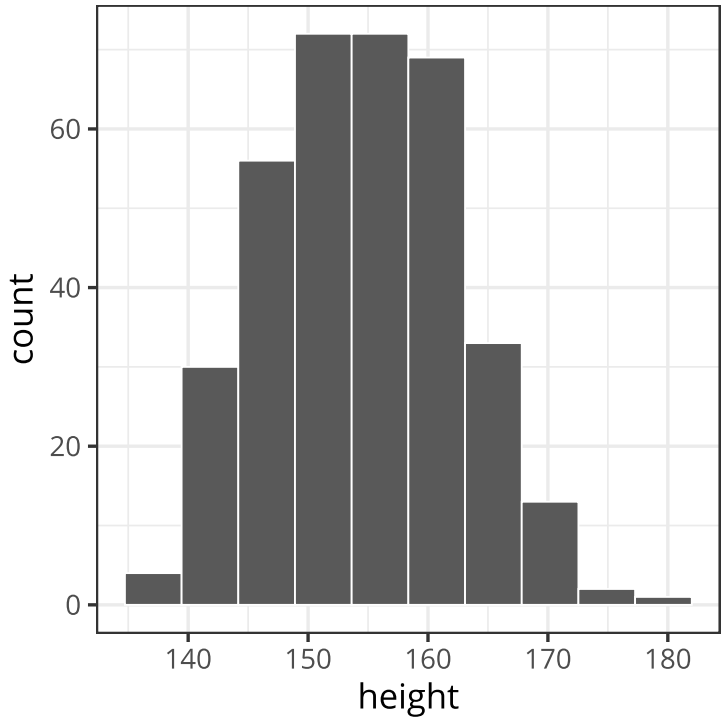
```
d2 <- d %>% filter(age >= 18)
head(d2)
```

```
   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

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

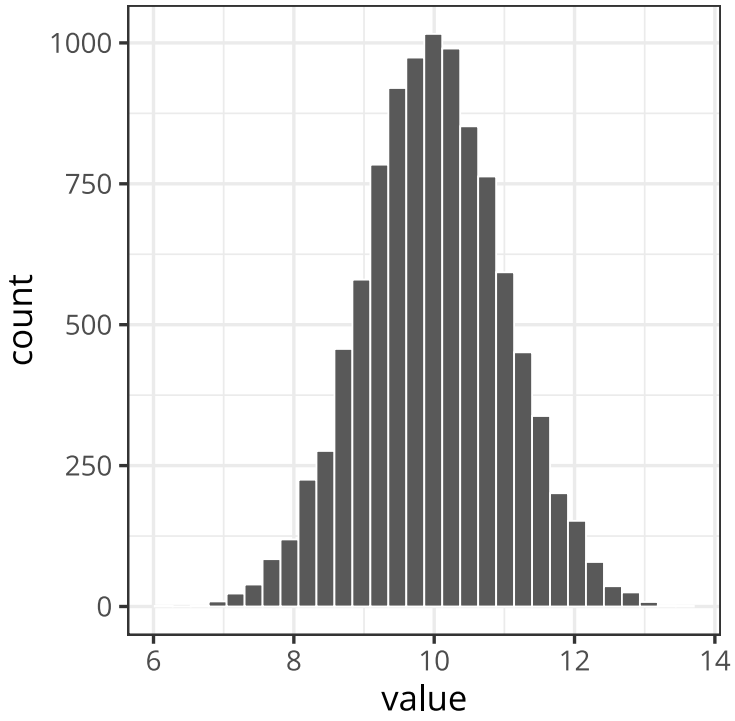
```
d2 %>%
    ggplot(aes(x = height) ) +
    geom_histogram(bins = 10, col = "white")
```



Loi normale

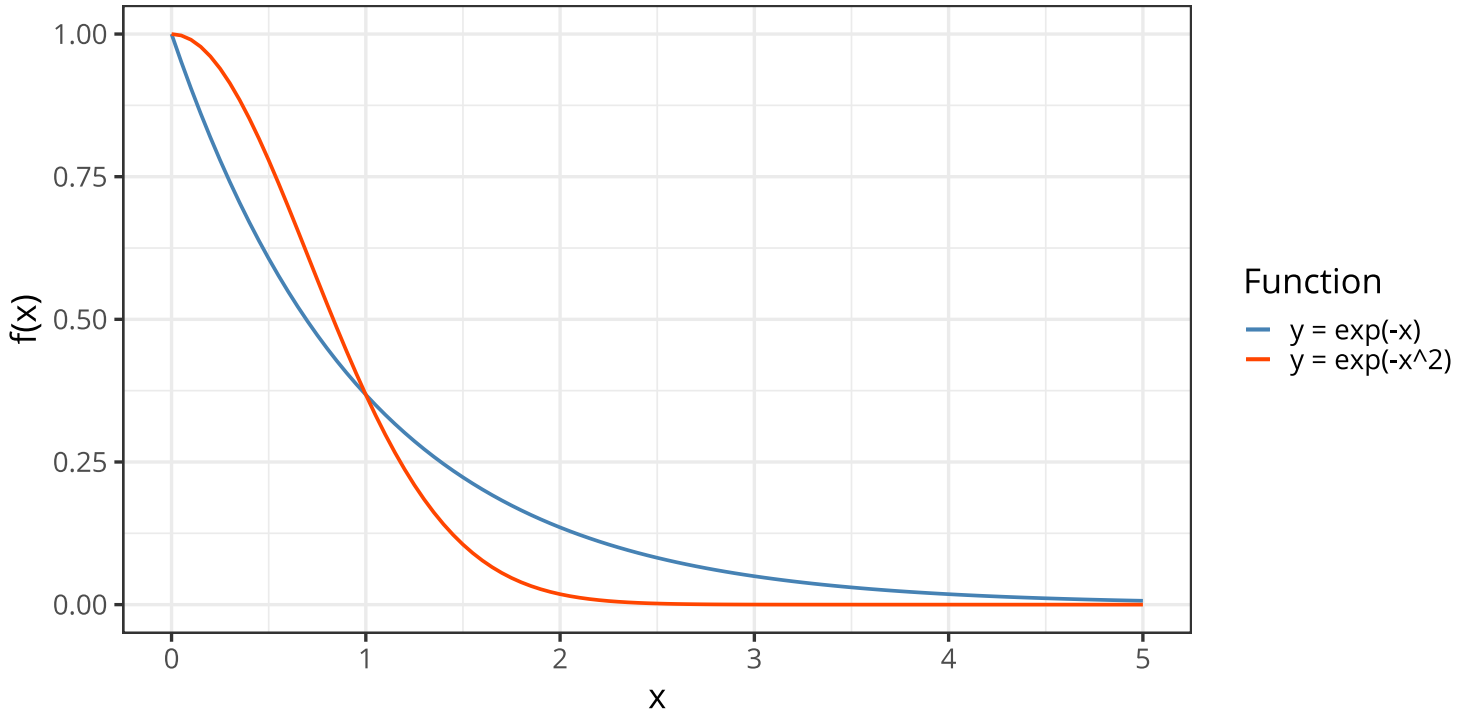
$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\mu - x)^2 \right]$$

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



D’où vient la loi normale ?

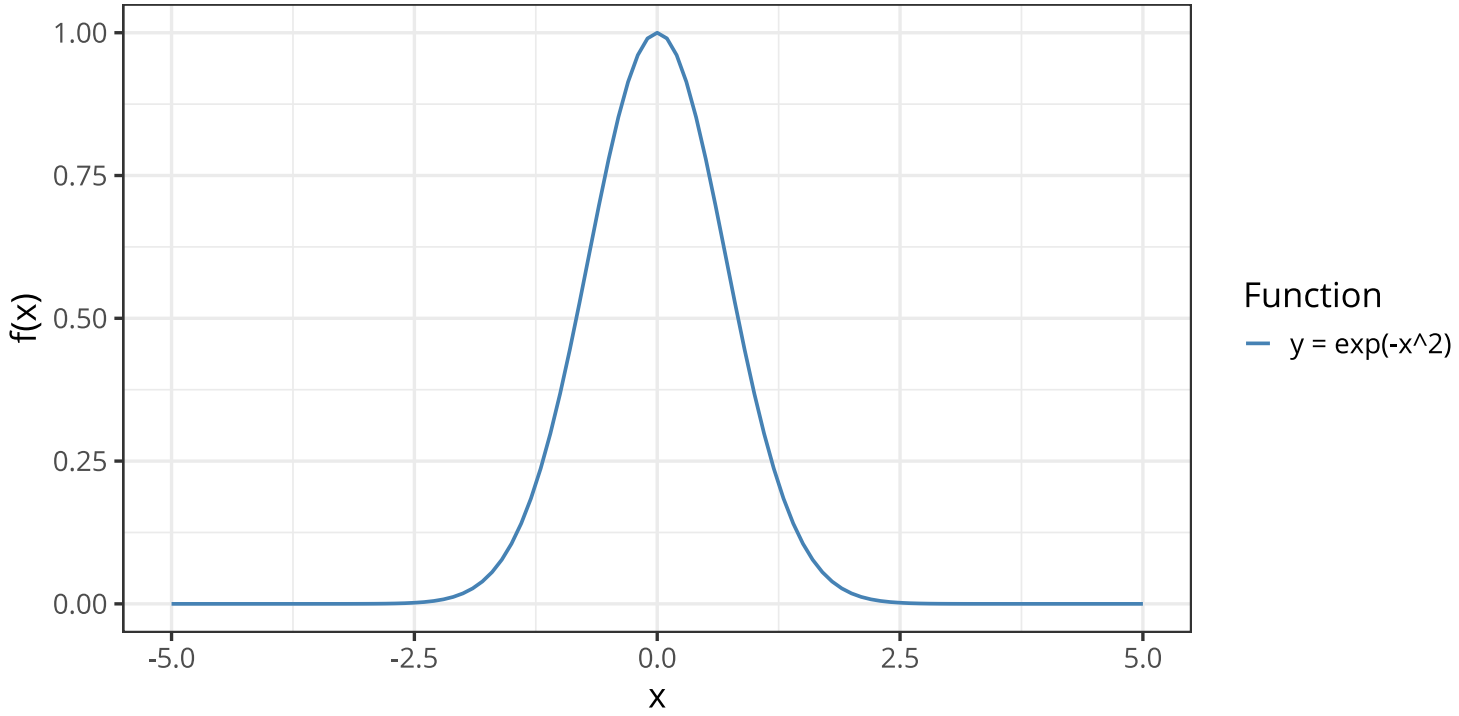
Certaines valeurs sont fortement probables (autour de la moyenne μ). Plus on s'éloigne, moins les valeurs sont probables (en suivant une décroissance exponentielle).



D'où vient la loi normale ?

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

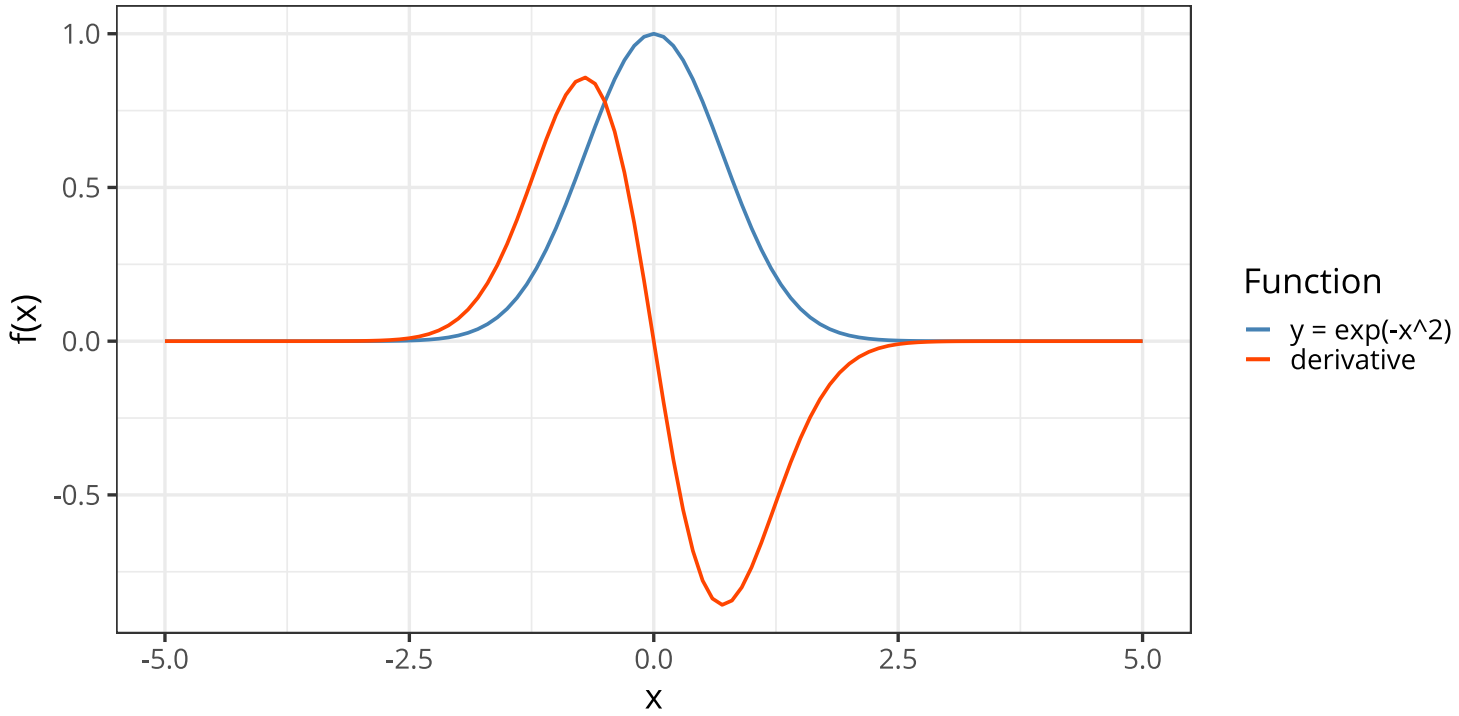
On étend notre fonction aux valeurs négatives.



D’où vient la loi normale ?

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

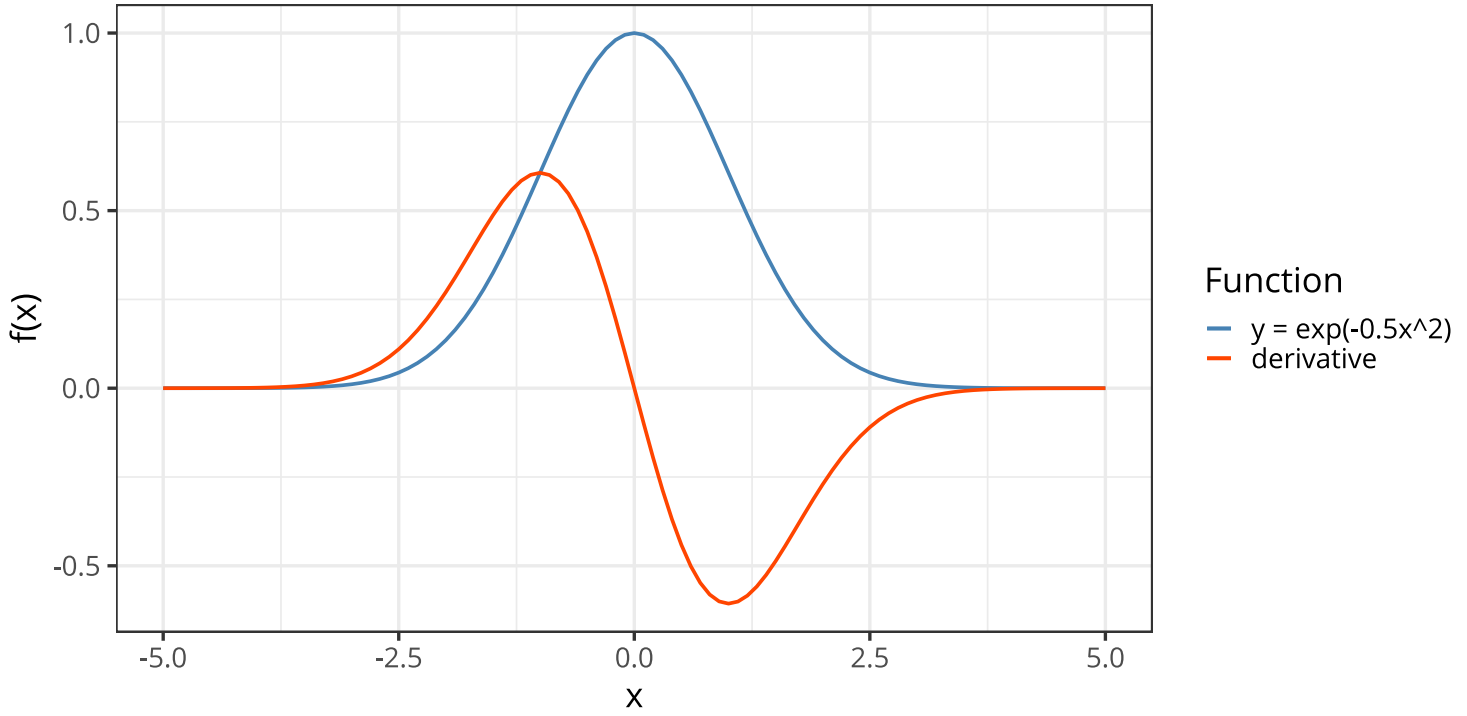
Les points d'inflection nous donnent une bonne indication de là où la plupart des valeurs se trouvent (i.e., entre les points d'inflection). Les pics de la dérivée nous montrent les points d'inflection.



D'où vient la loi normale ?

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

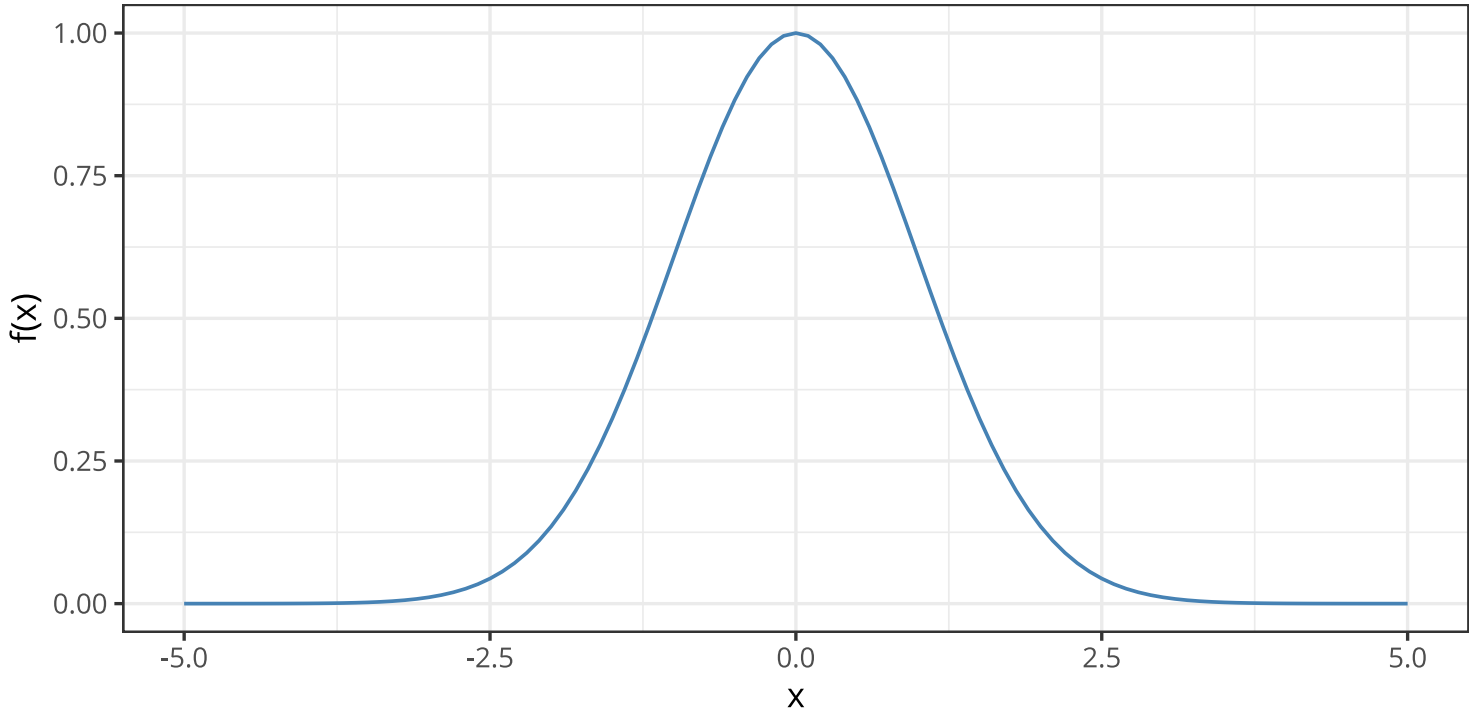
Ensuite on standardise la distribution de manière à ce que les deux points d'inflection se trouvent à $x = -1$ et $x = 1$.



D'où vient la loi normale ?

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

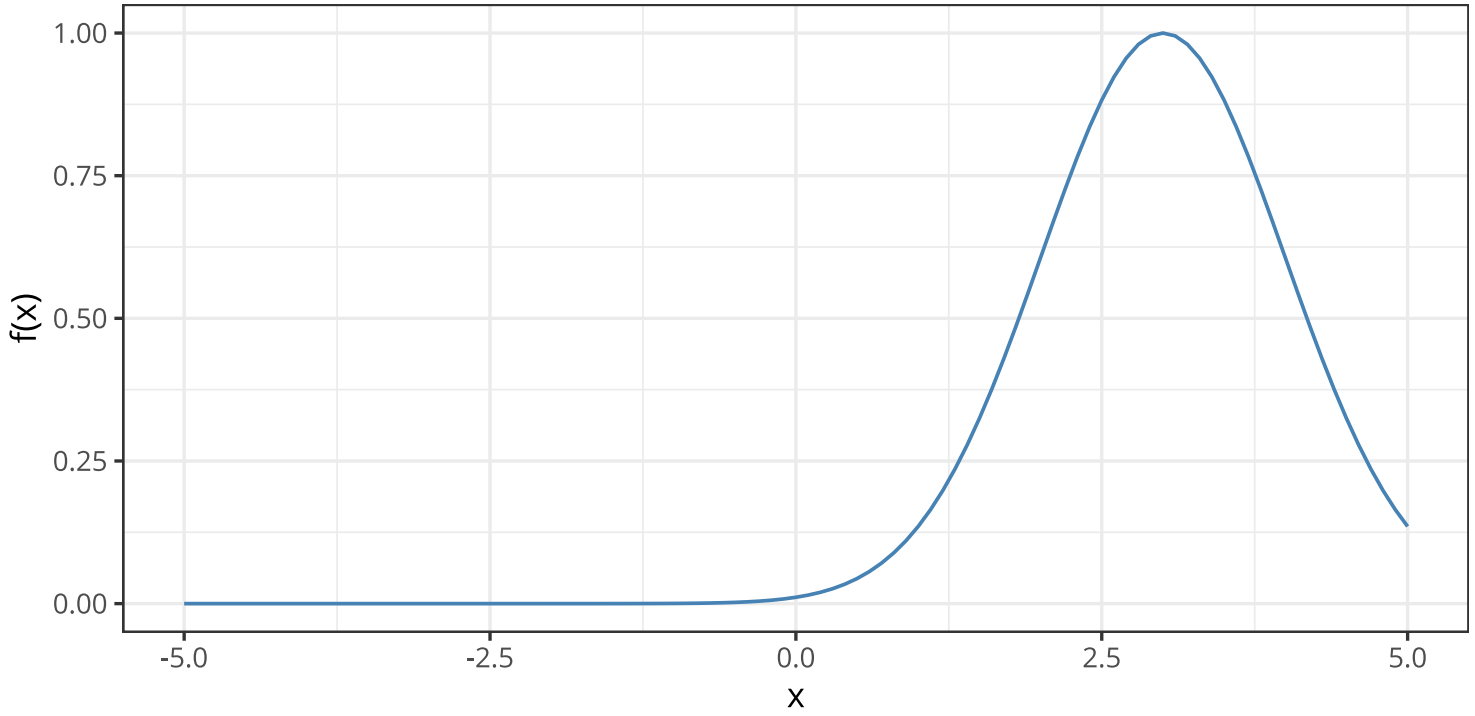
On insère un paramètre σ^2 pour contrôler la distance entre les points d'inflection.



D'où vient la loi normale ?

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

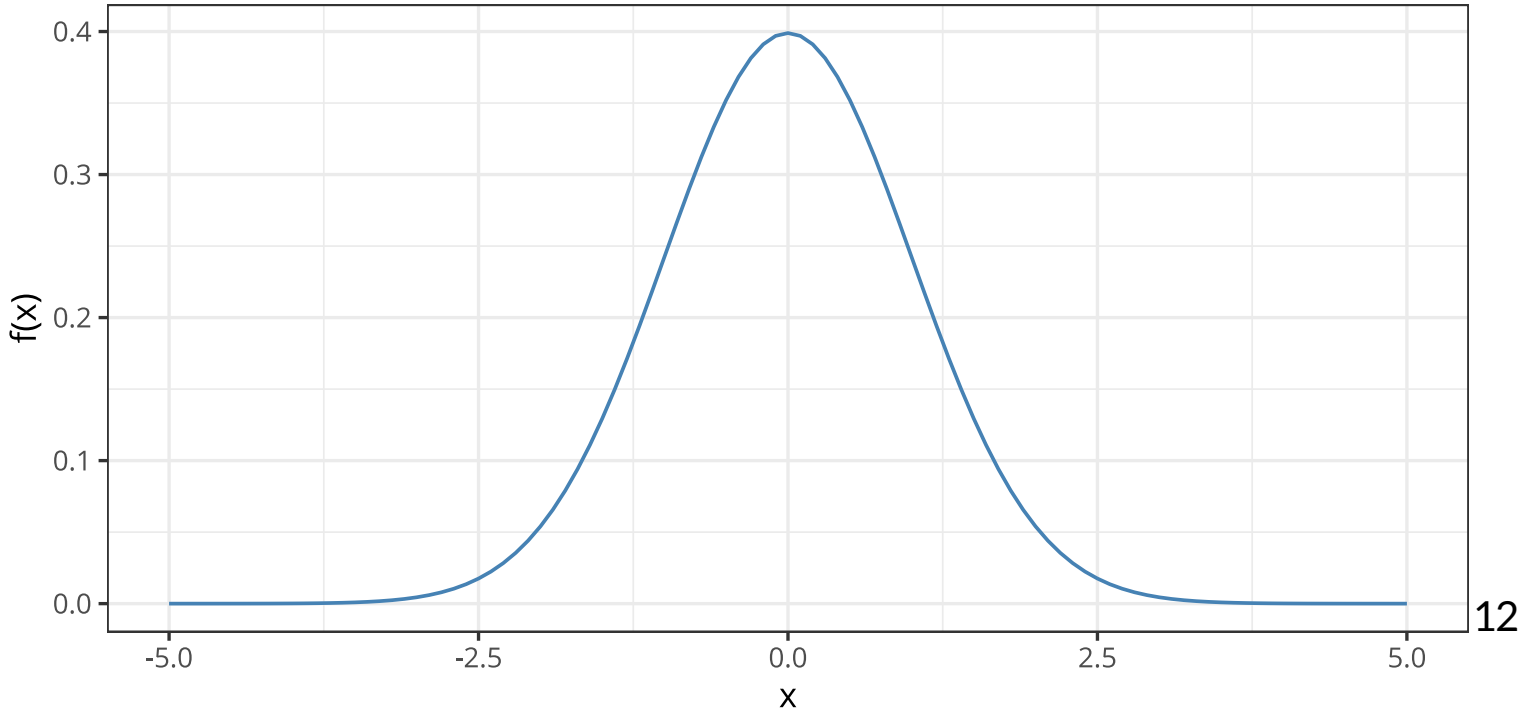
On insère ensuite un paramètre μ afin de pouvoir contrôler la position (la tendance centrale) de la distribution.



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.

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.

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 $p(\mu, \sigma) = p(\mu)p(\sigma)$.

Modèle gaussien

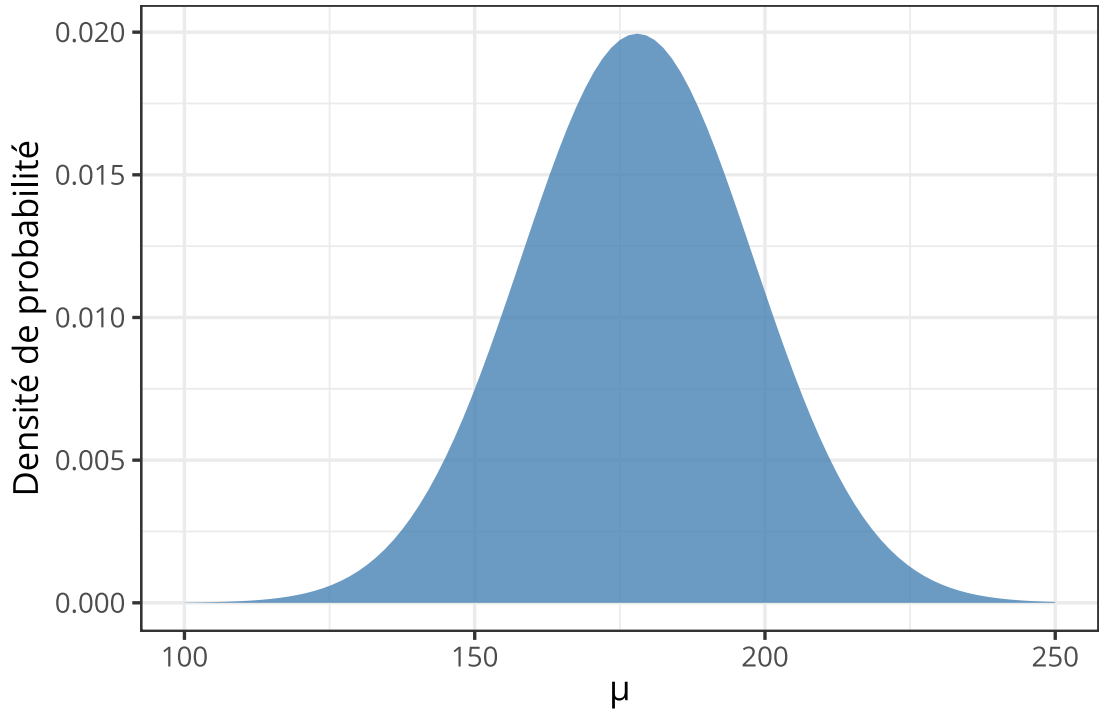
On définit $p(\mu, \sigma)$, la distribution a priori conjointe de tous les paramètres du modèle. On peut spécifier ces priors indépendamment pour chaque paramètre, sachant que $p(\mu, \sigma) = p(\mu)p(\sigma)$.

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

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 $p(\mu, \sigma) = p(\mu)p(\sigma)$.

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



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 $p(\mu, \sigma) = p(\mu)p(\sigma)$.

Modèle gaussien

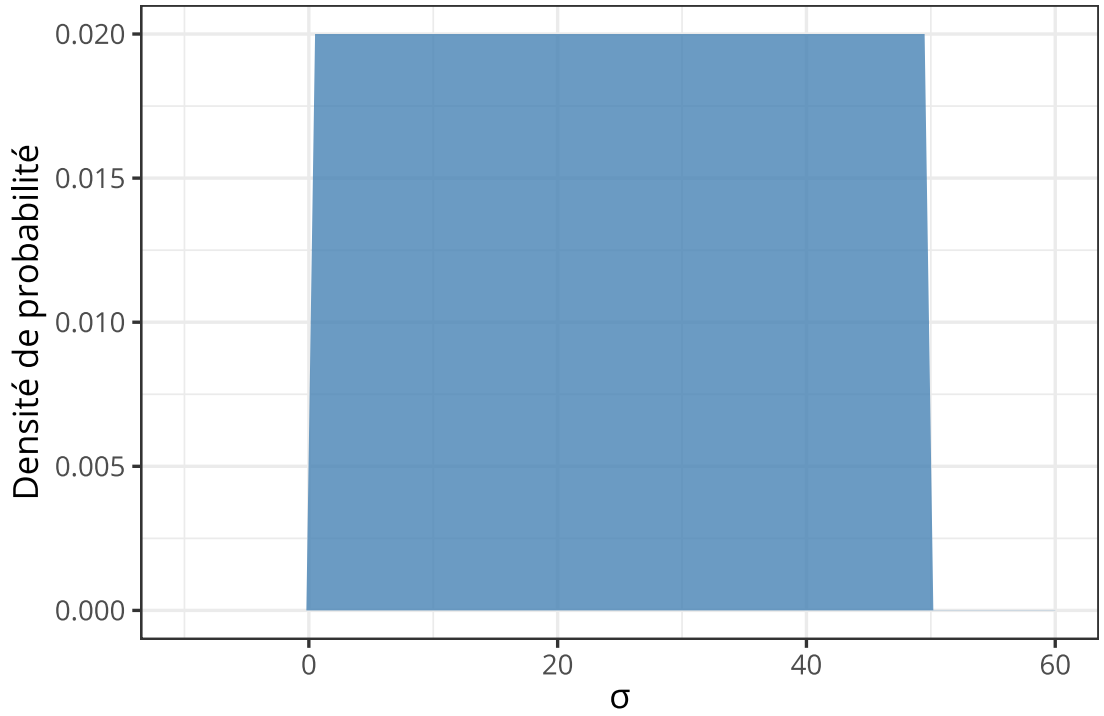
On définit $p(\mu, \sigma)$, la distribution a priori conjointe de tous les paramètres du modèle. On peut spécifier ces priors indépendamment pour chaque paramètre, sachant que $p(\mu, \sigma) = p(\mu)p(\sigma)$.

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

Modèle gaussien

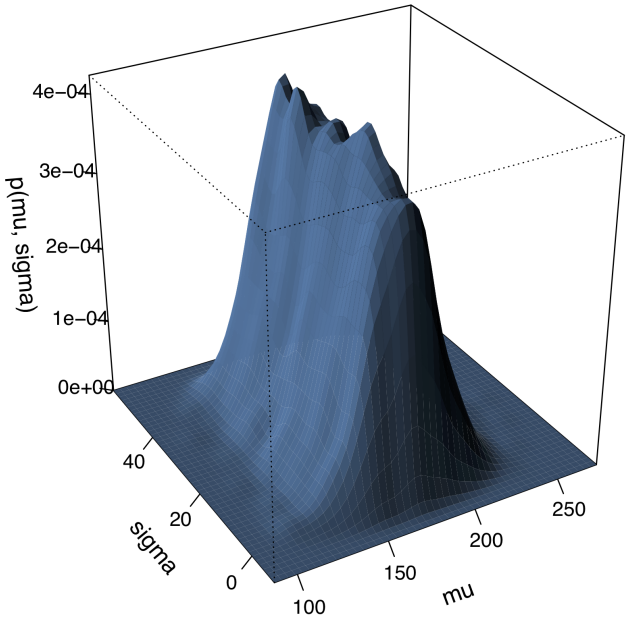
On définit $p(\mu, \sigma)$, la distribution a priori conjointe de tous les paramètres du modèle. On peut spécifier ces priors indépendamment pour chaque paramètre, sachant que $p(\mu, \sigma) = p(\mu)p(\sigma)$.

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



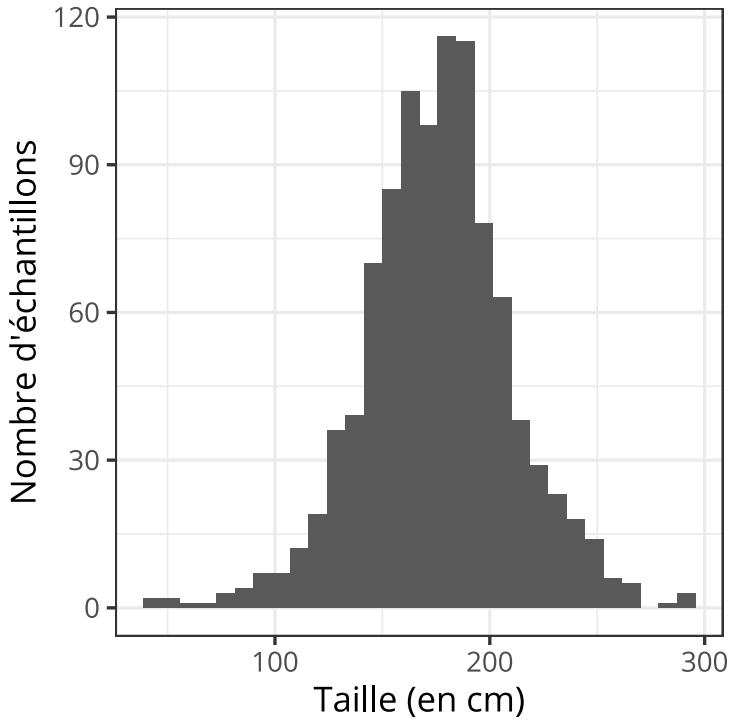
Visualiser le prior

```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

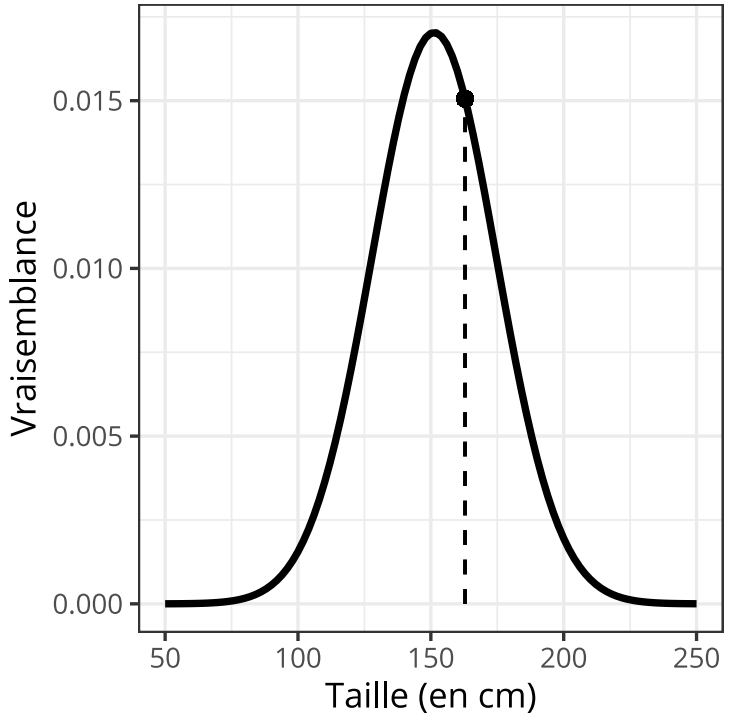
```r
sample_mu <- rnorm(1000, 178, 20)
sample_sigma <- runif(1000, 0, 50)

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

```
mu_exemple <- 151.23
sigma_exemple <- 23.42

d2$height[34] # une observation de taille (pour 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 :

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 :

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

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.

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.

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

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.

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

```
[1] 0.01505675
```

Fonction de vraisemblance

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

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 à la main…

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 à la main…

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

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

  return(bell / norm)

}
```

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 à la main…

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

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

  return(bell / norm)

}
```

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

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 à la main…

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

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

  return(bell / norm)

}
```

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

```
[1] 0.01505675
```

Distribution postérieure

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

Distribution postérieure

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

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

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.

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.

Distribution postérieure

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

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

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
# 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)

# étend la grille en deux dimensions (chaque combinaison de mu et sigma)
post <- expand.grid(mu = mu.list, sigma = sigma.list)

# calcul de la log-vraisemblance (pour chaque couple 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)
      )
    )

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

# on "annule" le log en avec exp() et on standardise par la valeur maximale
post$prob <- exp(post$prod - max(post$prod) )
```

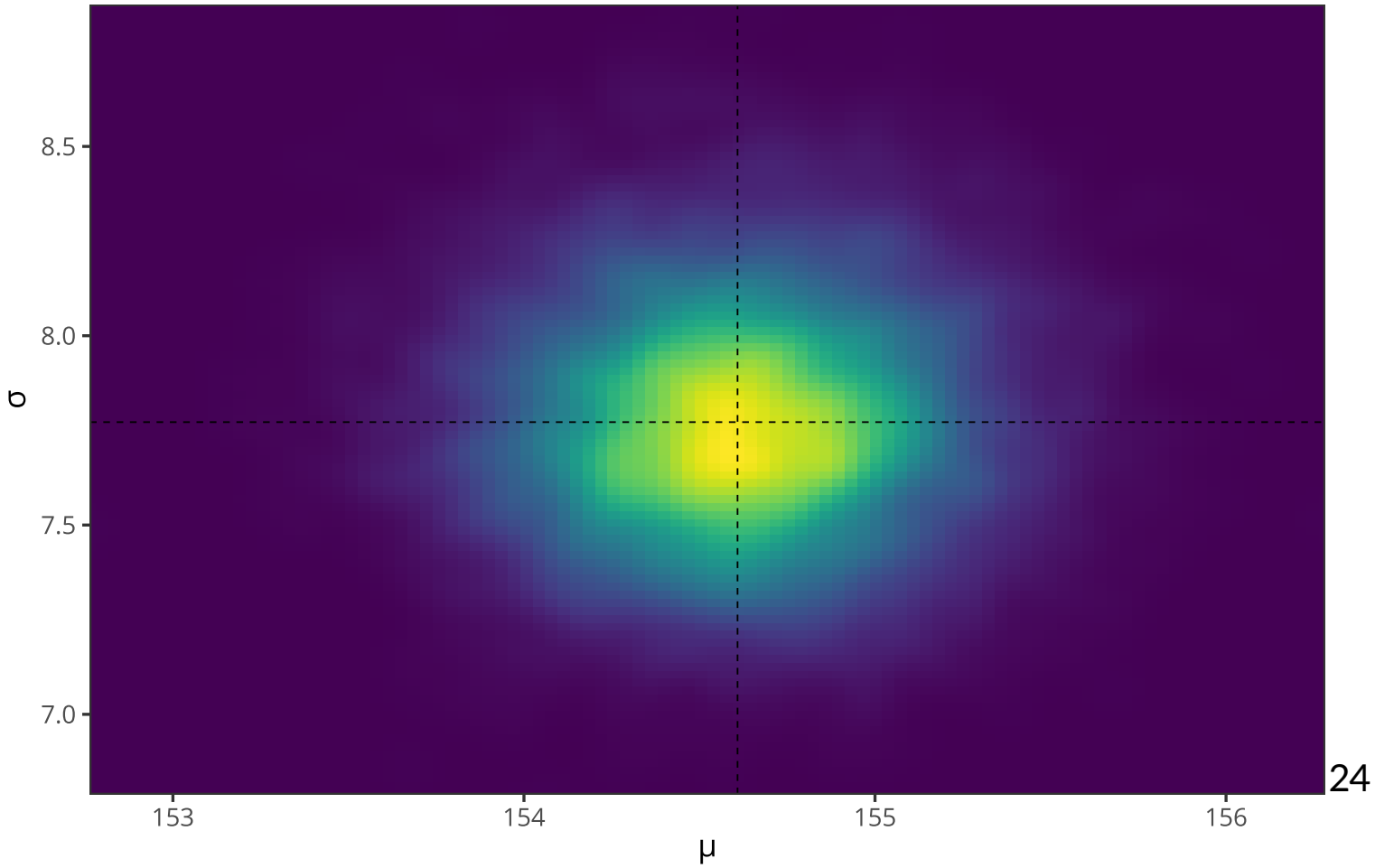

Distribution postérieure - Grid approximation

```
# select random 20 rows of the dataframe
post %>% slice_sample(n = 20, replace = FALSE)
```

```
          mu    sigma        LL      prod          prob
1  157.3869 6.713568 -1257.520 -1265.878  3.276633e-17
2  153.0653 4.452261 -1400.692 -1409.296  1.696364e-79
3  141.6080 7.316583 -1775.207 -1784.689 1.579913e-242
4  140.9045 7.266332 -1845.777 -1855.324 3.327535e-273
5  156.7839 4.201005 -1472.485 -1480.874 1.392174e-110
6  146.9347 8.145729 -1376.067 -1385.100  5.469654e-69
7  143.1156 5.582915 -2010.694 -2020.042  0.000000e+00
8  154.3719 4.452261 -1380.310 -1388.835  1.305211e-70
9  158.3920 5.432161 -1361.576 -1369.883  2.220653e-62
10 156.8844 7.944724 -1234.250 -1242.634  4.073824e-07
11 151.0553 6.738693 -1275.325 -1284.059  4.162371e-25
12 147.9397 6.788945 -1395.149 -1404.105  3.048800e-77
13 150.6533 5.030151 -1416.070 -1424.831  3.038659e-86
14 140.9045 4.000000 -3531.301 -3540.848  0.000000e+00
15 142.8141 5.105528 -2238.372 -2247.747  0.000000e+00
16 148.4422 6.663317 -1378.182 -1387.101  7.391499e-70
17 153.4673 7.668342 -1223.249 -1231.828  2.009124e-02
18 155.4774 6.814070 -1228.461 -1236.921  1.232804e-04
19 149.3467 4.226131 -1691.473 -1700.326 6.870178e-206
20 146.6332 7.668342 -1409.259 -1418.315  2.054325e-83
```

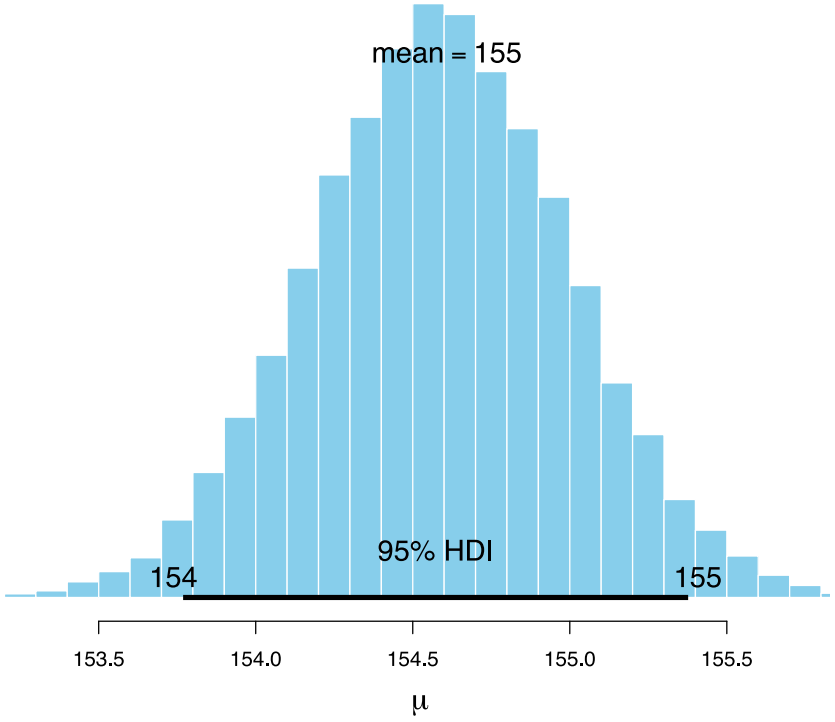

Distribution postérieure - Grid approximation

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

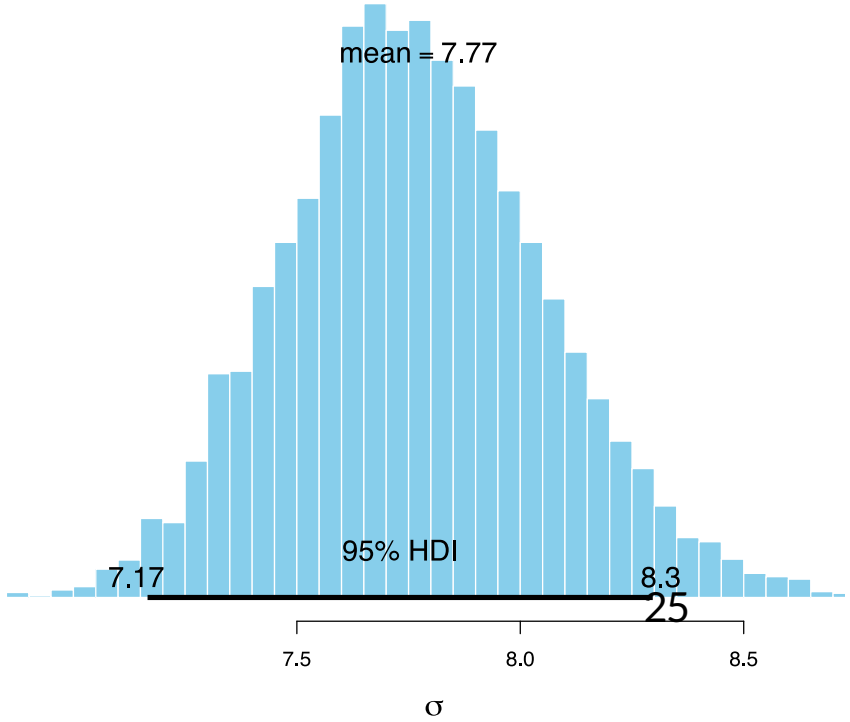


Distribution postérieure - Distributions marginales

```
BEST::plotPost(
  sample.mu, breaks = 40, xlab =
expression(mu)
  )
```




```
BEST::plotPost(
  sample.sigma, breaks = 40, xlab =
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`.

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 :

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 :

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

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 :

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

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

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 :

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

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`.

Rappels de syntaxe

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

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

Rappels de syntaxe

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

```
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()`.

Rappels de syntaxe

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

```
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()`.

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

Rappels de syntaxe

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

```
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()`.

```
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.

Rappels de syntaxe

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

```
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()`.

```
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.

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

Rappels de syntaxe

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

```
mvbind(Reaction, Memory) ~ 0 + 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.

```
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`.

Rappels de syntaxe

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

```
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`.

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

Rappels de syntaxe

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

```
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`.

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

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

Quelques fonctions utiles

Quelques fonctions utiles

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

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

# récupérer les prédiction 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, ...)

# test d'hypothèse
hypothesis(fit, hypothesis, ...)
```

Un premier exemple

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

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

```
             Estimate Est.Error       Q2.5      Q97.5
b_Intercept 154.59572 0.4177722 153.784575 155.442233
sigma         7.75961 0.2893936   7.224017   8.335747
```

Ces données représentent les distributions marginales de chaque paramètre. En d'autres termes, la *probabilité* de chaque valeur de μ, après avoir *moyenné* 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.

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.

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.

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

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.

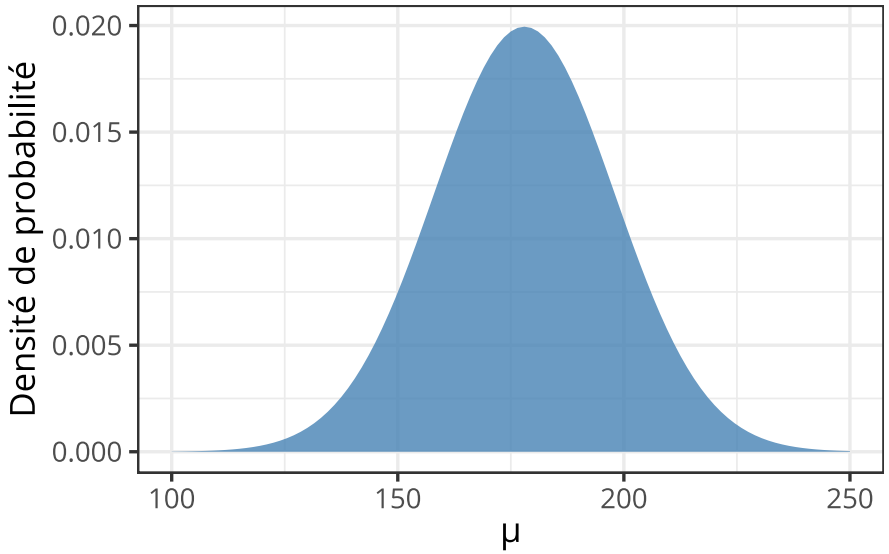
La fonction `get_prior()` permet de visualiser une liste des priors par défaut ainsi que de tous les priors qu'on peut spécifier, sachant une certaine formule (i.e., une manière d'écrire notre modèle) et un jeu de données.

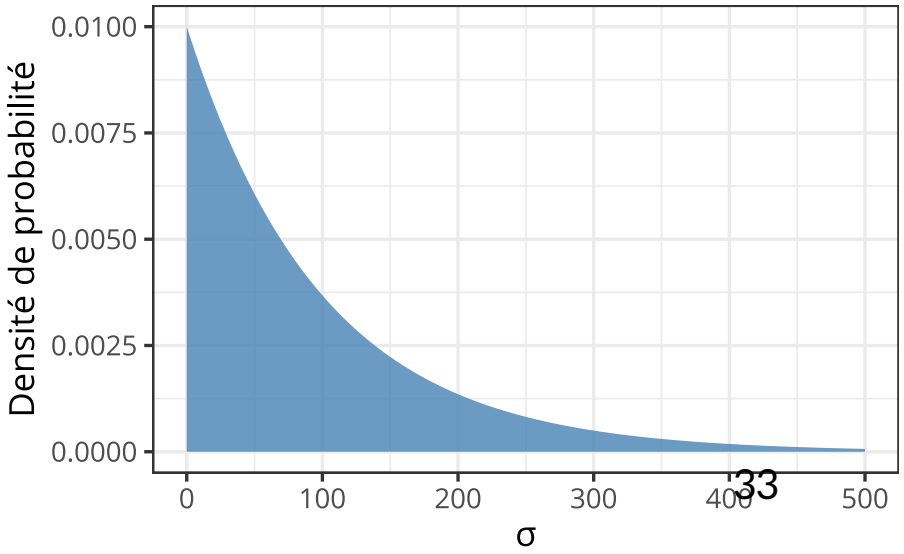
```
get_prior(height ~ 1, data = d2)
```

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





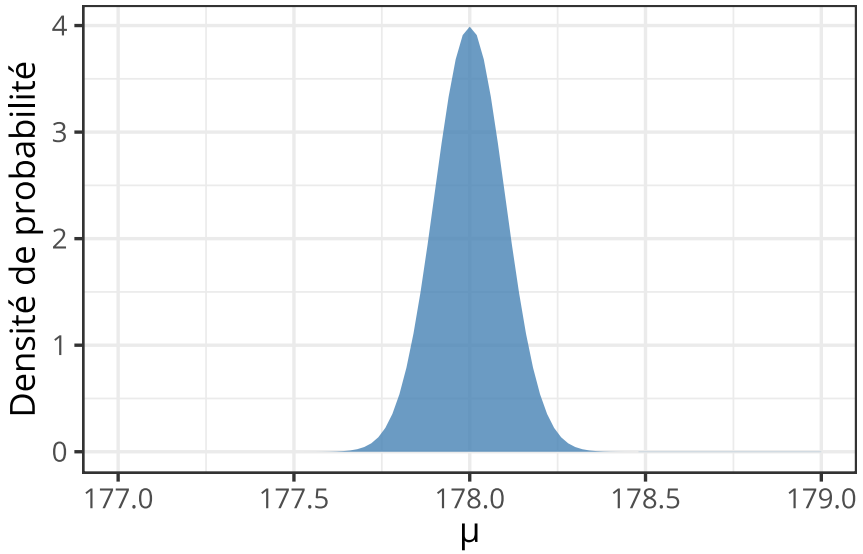
En utilisant notre prior

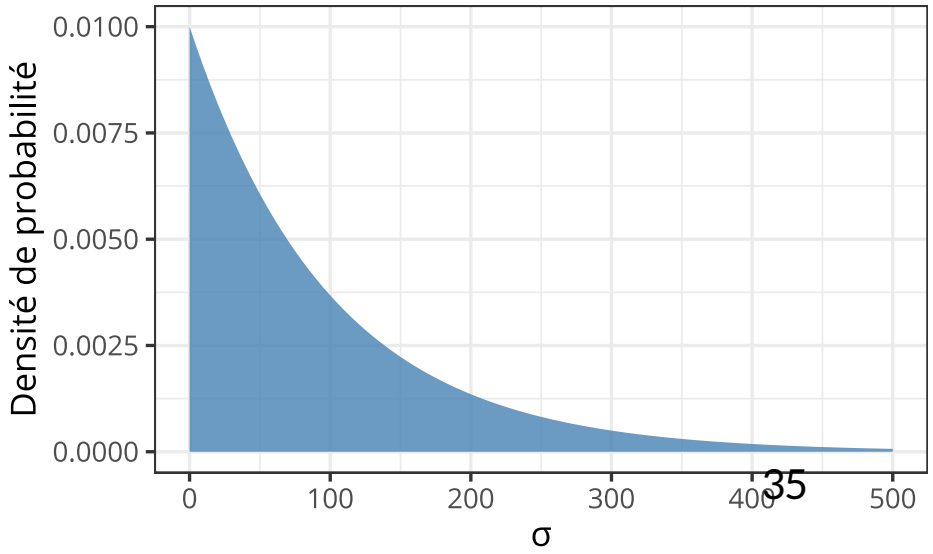
```
summary(mod2)
```

```
 Family: gaussian
  Links: mu = identity; sigma = 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

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   154.60      0.42   153.77   155.43 1.00     3124     2383

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     7.78      0.29     7.25     8.37 1.00     3519     2613

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; sigma = 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

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   177.86      0.10   177.67   178.05 1.00     3521     2776

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma    24.59      0.94    22.84    26.48 1.00     4164     3035

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…

Précision du prior (heuristique)

Le prior peut généralement être considéré comme un posterior obtenu sur des données antérieures.

Précision du prior (heuristique)

Le prior peut généralement être considéré comme un posterior obtenu sur des données antérieures.

On sait que le σ d'un posterior gaussien nous est donné par la formule :

Précision du prior (heuristique)

Le prior peut généralement être considéré comme un posterior obtenu sur des données antérieures.

On sait que le σ d'un posterior gaussien nous est donné par la formule :

$$\sigma_{post} = 1/\sqrt{n}$$

Précision du prior (heuristique)

Le prior peut généralement être considéré comme un posterior obtenu sur des données antérieures.

On sait que le σ d'un posterior gaussien nous est donné par la formule :

$$\sigma_{post} = 1/\sqrt{n}$$

Qui implique une *quantité de données* $n = 1/\sigma_{post}^2$. Notre prior avait un $\sigma = 0.1$, ce qui donne $n = 1/0.1^2 = 100.$

Précision du prior (heuristique)

Le prior peut généralement être considéré comme un posterior obtenu sur des données antérieures.

On sait que le σ d'un posterior gaussien nous est donné par la formule :

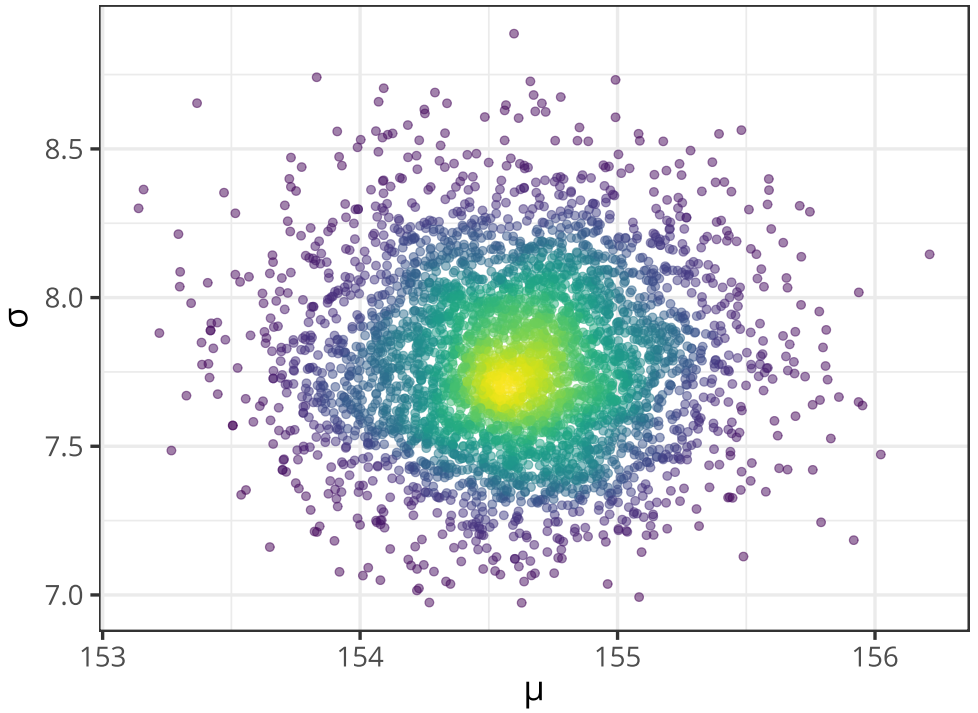
$$\sigma_{post} = 1/\sqrt{n}$$

Qui implique une *quantité de données* $n = 1/\sigma_{post}^2$. Notre prior avait un $\sigma = 0.1$, ce qui donne $n = 1/0.1^2 = 100$.

Donc, on peut considérer que le prior $\mu \sim \text{Normal}(178, 0.1)$ est équivalent au cas dans lequel nous aurions observé 100 tailles de moyenne 178.

Récupérer et visualiser les échantillons de la distribution postérieure

```
post <- posterior_samples(mod2) %>%
    mutate(density = get_density(b_Intercept, sigma, n = 1e2) )

ggplot(post, 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

Récupérer les échantillons de la distribution postérieure

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

Récupérer les échantillons de la distribution postérieure

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

```
  b_Intercept    sigma      lp__    density
1    153.8300 7.530977 -1228.784 0.1737273
2    155.3849 8.195800 -1229.321 0.1099155
3    155.2592 8.230882 -1229.009 0.1648804
4    155.1455 8.211893 -1228.569 0.2236953
5    154.6705 7.674547 -1226.679 1.3277269
6    154.6709 7.664548 -1226.688 1.3043082
```

Récupérer les échantillons de la distribution postérieure

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

```
  b_Intercept    sigma      lp__    density
1    153.8300 7.530977 -1228.784 0.1737273
2    155.3849 8.195800 -1229.321 0.1099155
3    155.2592 8.230882 -1229.009 0.1648804
4    155.1455 8.211893 -1228.569 0.2236953
5    154.6705 7.674547 -1226.679 1.3277269
6    154.6709 7.664548 -1226.688 1.3043082
```

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

Récupérer les échantillons de la distribution postérieure

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

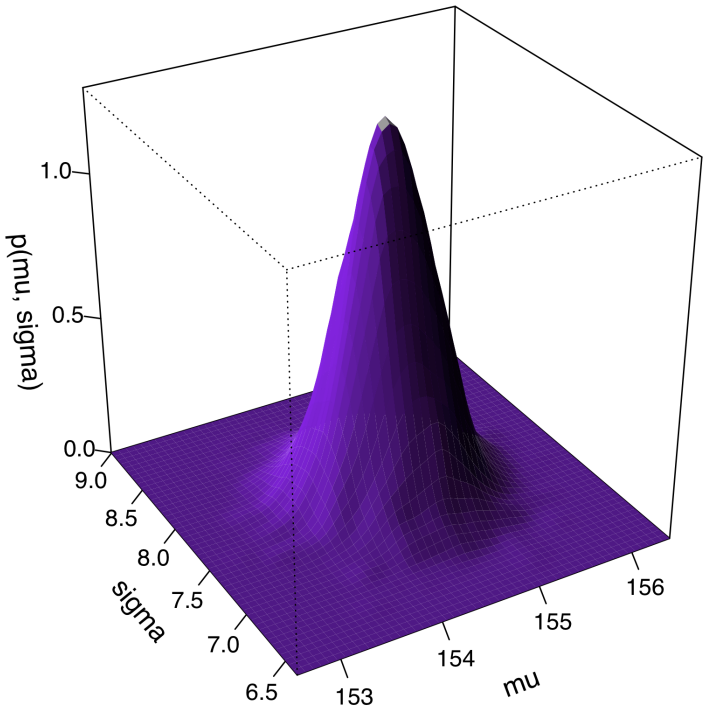
```
  b_Intercept    sigma      lp__    density
1    153.8300 7.530977 -1228.784 0.1737273
2    155.3849 8.195800 -1229.321 0.1099155
3    155.2592 8.230882 -1229.009 0.1648804
4    155.1455 8.211893 -1228.569 0.2236953
5    154.6705 7.674547 -1226.679 1.3277269
6    154.6709 7.664548 -1226.688 1.3043082
```

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

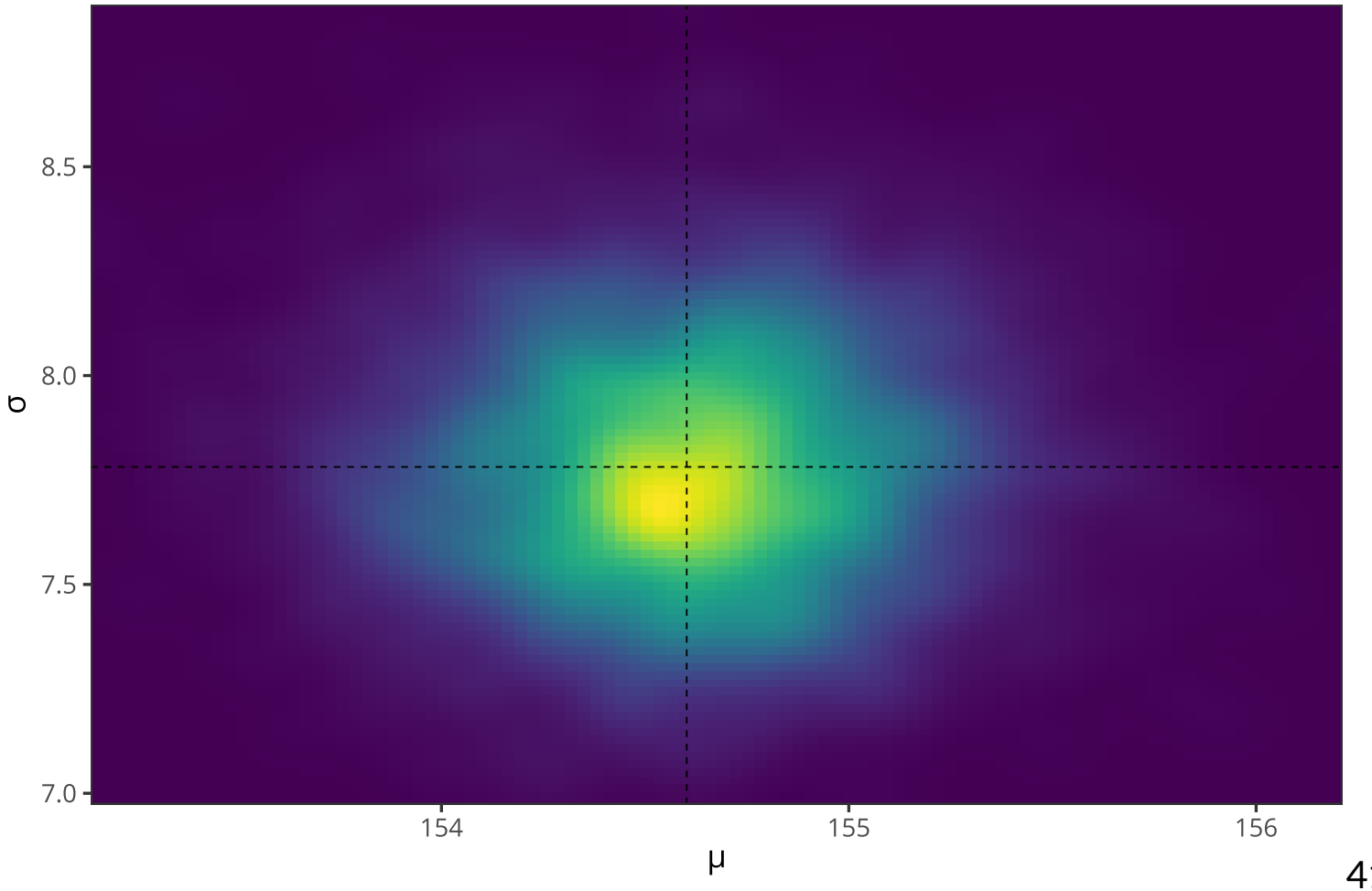
```
                2.5%       50%      97.5%
b_Intercept 153.772950 154.60629 155.429238
sigma         7.245842   7.76739   8.372069
```

Visualiser la distribution postérieure

```
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")
```



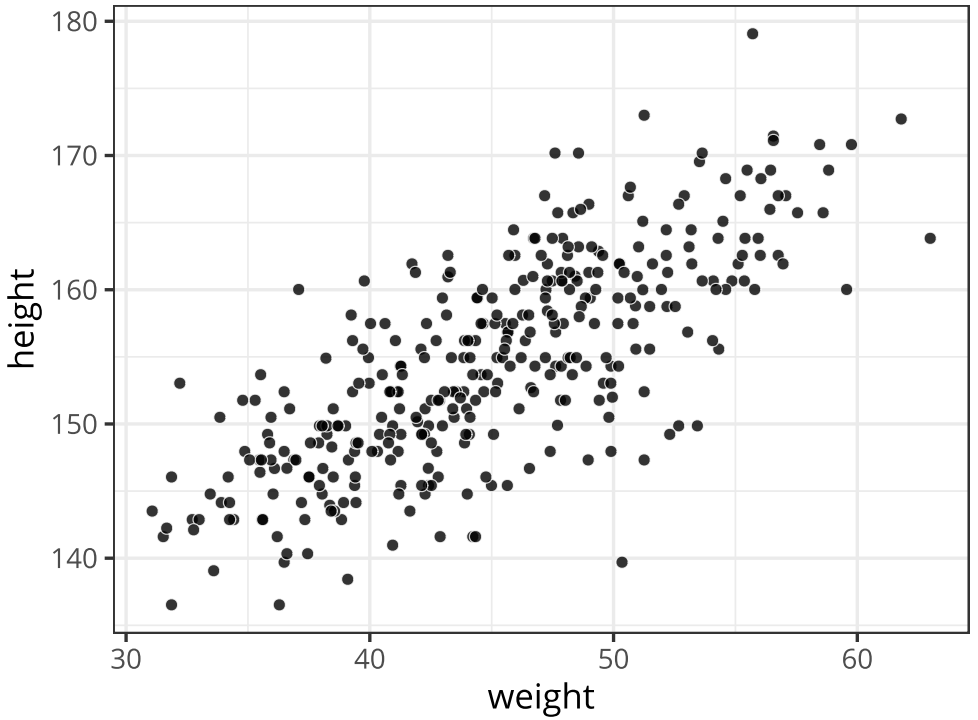
Visualiser la distribution postérieure



Ajouter un prédicteur

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

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



Régression linéaire à un prédicteur continu

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

Régression linéaire à un prédicteur continu

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

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

Régression linéaire à un prédicteur continu

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

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

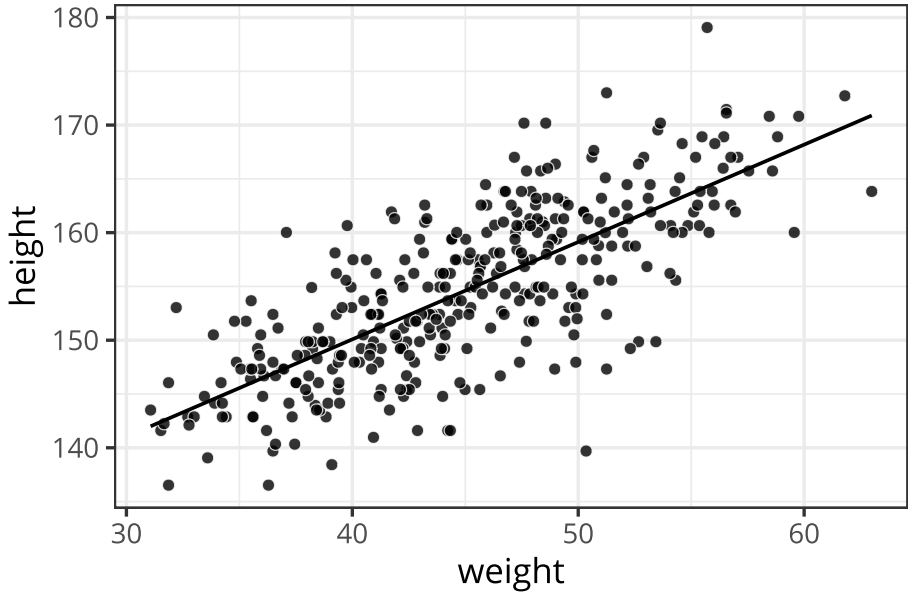
```
             mean   sd   2.5%  97.5%
(Intercept) 113.88 1.91 110.13 117.63
weight        0.91 0.04   0.82   0.99
```

Régression linéaire à un prédicteur continu

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

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

```
              mean   sd   2.5%  97.5%
(Intercept) 113.88 1.91 110.13 117.63
weight        0.91 0.04   0.82   0.99
```



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 :

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 :

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

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 :

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 :

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

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 :

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 :

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

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

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

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.

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(mod4)
```

```
                Estimate Est.Error          Q2.5         Q97.5
b_Intercept   113.8691224 1.9101041   110.1773362   117.6849700
b_weight        0.9053111 0.0421729     0.8214331     0.9854805
sigma           5.1073937 0.1956593     4.7391734     5.5150977
lp__        -1083.3922606 1.2804405 -1086.8299632 -1081.9813013
```

- $\alpha = 113.87, 95\%$ CrI $[110.18, 117.68]$ représente la taille moyenne quand le poids est égal à 0kg…
- $\beta = 0.91, 95\%$ CrI $[0.82, 0.99]$ nous indique qu'une augmentation de 1kg entraîne une augmentation de 0.90cm.

Régression linéaire à un prédicteur continu

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

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

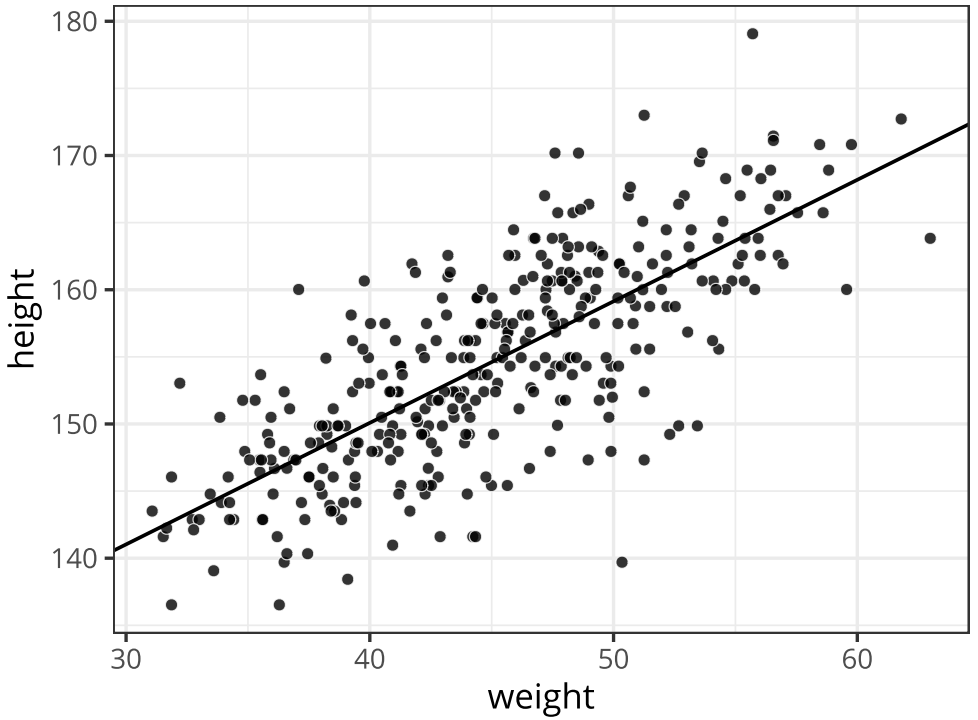
```
fixef(mod5) # retrieves the fixed effects estimates
```

```
             Estimate  Est.Error        Q2.5       Q97.5
Intercept 154.6043352 0.27319971 154.0763921 155.1465171
weight.c    0.9053015 0.04213109   0.8210985   0.9874467
```

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

Représenter les prédictions du modèle

```
d2 %>%
    ggplot(aes(x = weight, y = height) ) +
    geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
    geom_abline(intercept = fixef(mod4)[1], slope = fixef(mod4)[2], lwd = 1)
```



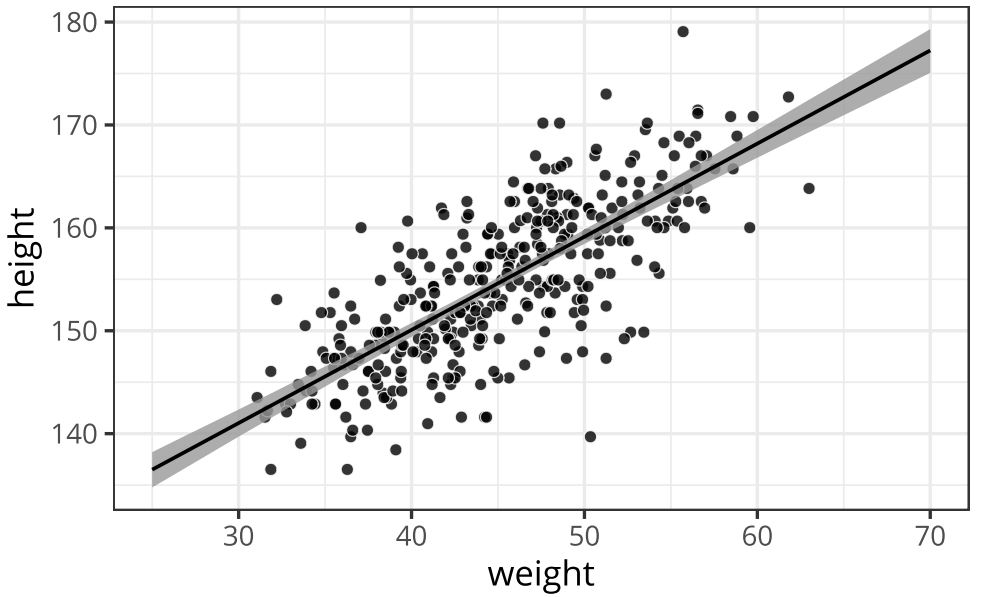
Représenter l'incertitude sur μ via fitted()

```
# 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  136.5019 0.8800198 134.7740 138.2039     25
2  137.4072 0.8400683 135.7518 139.0363     26
3  138.3125 0.8003448 136.7422 139.8680     27
4  139.2178 0.7608849 137.7266 140.6895     28
5  140.1231 0.7217319 138.7219 141.5187     29
6  141.0285 0.6829387 139.7092 142.3436     30
7  141.9338 0.6445702 140.6770 143.1818     31
8  142.8391 0.6067069 141.6502 144.0146     32
9  143.7444 0.5694496 142.6190 144.8633     33
10 144.6497 0.5329255 143.5997 145.7018     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 ?

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 ?

```
# 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)
```

Intervalles de prédiction (incorporer σ)

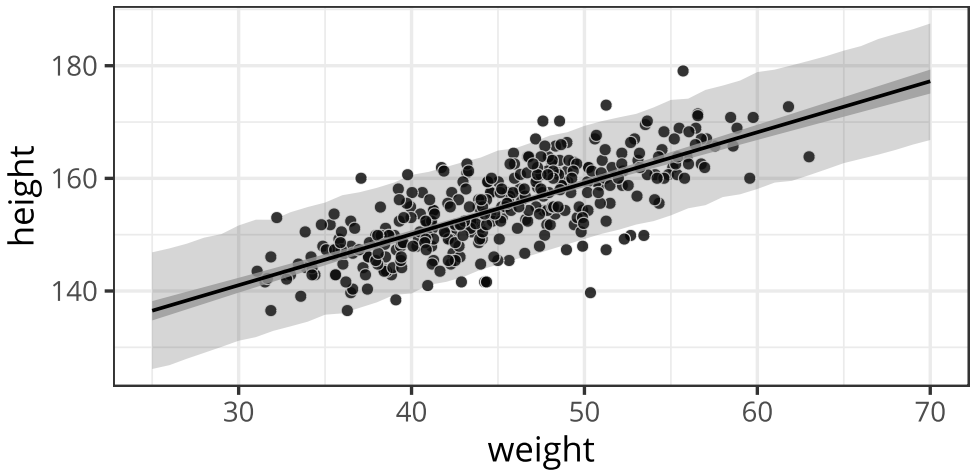
Pour rappel, voici notre modèle : $h_i \sim \text{Normal}(\alpha + \beta x_i, \sigma)$. Pour l'instant, on a seulement représenté les prédictions pour μ. Comment incorporer σ dans nos prédictions ?

```
# 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  136.4695  5.248905 126.1536 146.8657     25
2  137.3757  5.239130 126.8338 147.5680     26
3  138.3122  5.194219 127.9703 148.3992     27
4  139.3006  5.173988 129.0124 149.4677     28
5  140.1013  5.211367 130.0674 150.0982     29
6  141.1060  5.206877 131.1675 151.6373     30
7  141.9390  5.195607 131.8138 152.6651     31
8  142.7406  5.068944 132.9189 152.6539     32
9  143.8971  5.191560 133.7217 153.9880     33
10 144.6535  5.133855 134.5595 154.9149     34
```

Intervalles de prédiction (incorporer σ)

```
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
    )
```



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.

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.

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).

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.

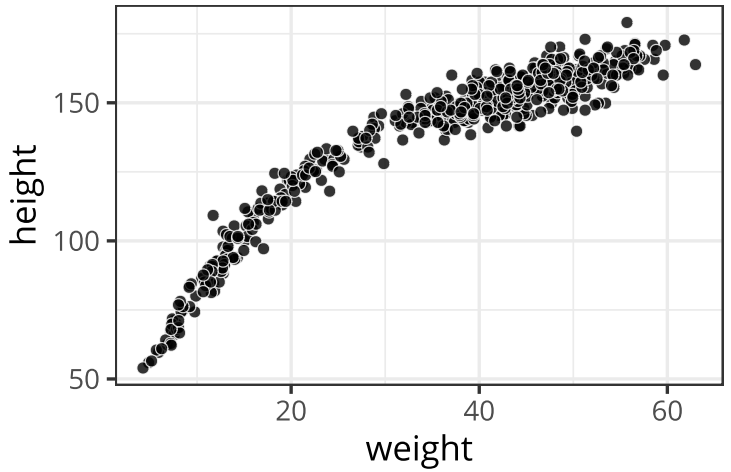
Incertitude épistémique : La distribution a posteriori ordonne toutes les combinaisons possibles des valeurs des paramètres selon leurs plausibilités relatives.

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 (2012).

Régression polynomiale

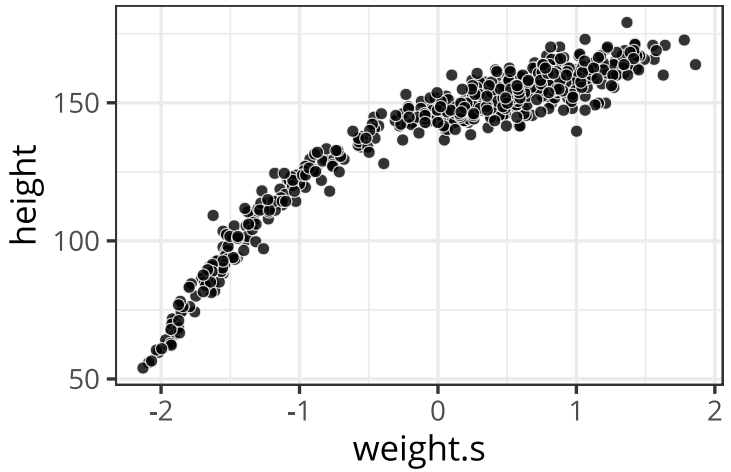
```
d %>% # on utilise d au lieu de d2
  ggplot(aes(x = weight, y = height) ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8)
```



Si on considère tout l'échantillon (pas seulement les adultes), la relation entre taille et poids semble incurvée…

Scores standardisés

```
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)
```




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

```
[1] -2.712698e-18  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, 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

```
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

```
summary(mod6)
```

```
 Family: gaussian
  Links: mu = identity; sigma = 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

Population-Level Effects:
            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept     146.66      0.37   145.95   147.38 1.00     3729     3002
weight.s       21.40      0.29    20.84    21.99 1.00     3642     2876
Iweight.sE2    -8.41      0.28    -8.95    -7.87 1.00     3184     2856

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     5.78      0.17     5.46     6.13 1.00     3358     2879

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).
```

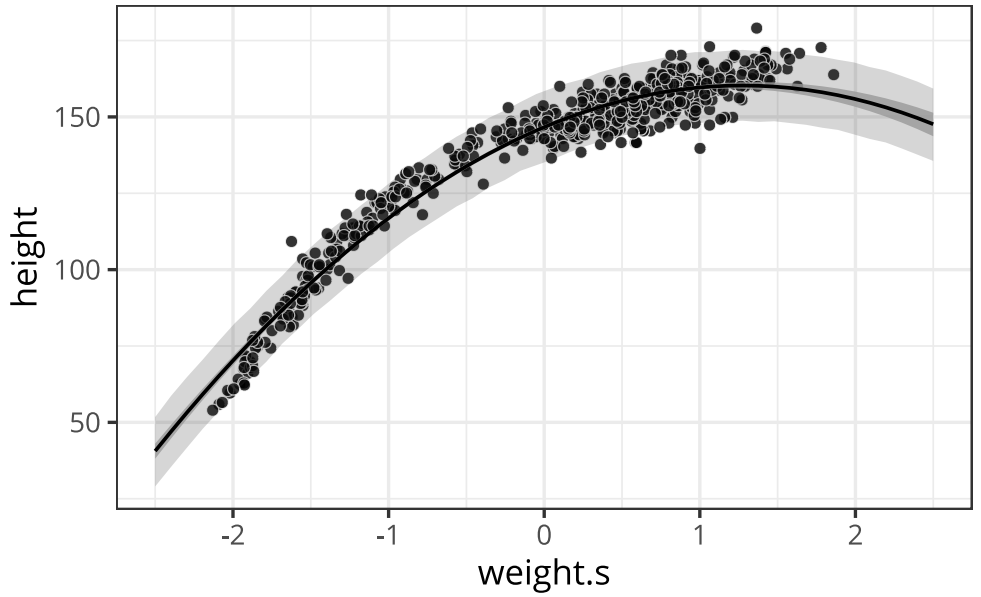

Représenter les prédictions du modèle

```
# 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  40.56051  5.935990 29.02612  51.73614 -2.500000
2  47.03215  5.929627 35.42704  58.67678 -2.397959
3  53.24911  5.807543 41.71367  64.82054 -2.295918
4  59.21773  5.773481 47.91146  70.57949 -2.193878
5  65.17064  5.785060 53.61187  76.60969 -2.091837
6  71.02563  5.826925 59.61307  82.38593 -1.989796
7  76.38210  5.833239 64.84921  87.33277 -1.887755
8  81.75520  5.890726 69.83837  93.04002 -1.785714
9  86.67901  5.703355 75.74684  98.00381 -1.683673
10 91.84259  5.846767 80.55267 103.54333 -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.

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 et al. (2017), 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 basé sur l'échantillon.

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 et al. (2017), 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 basé 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; Reynolds, 1977). 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

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

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

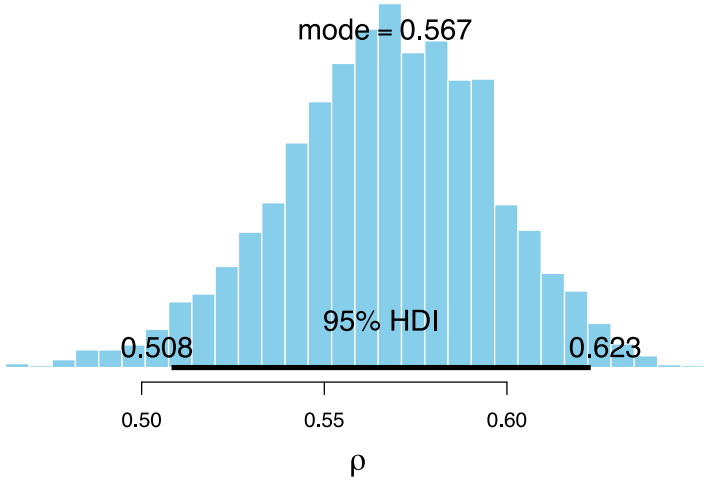
$$\rho^2 = \frac{\beta^2 \tau^2}{\sigma^2 + \beta^2 \tau^2}$$

```
post <- posterior_samples(mod4)
beta <- post$b_weight
sigma <- post$sigma

f1 <- beta^2 * var(d2$weight)
rho <- f1 / (f1 + sigma^2)
```

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

Modèle de régression, taille d'effet

```
BEST::plotPost(rho, showMode = TRUE, xlab = expression(rho) )
```




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

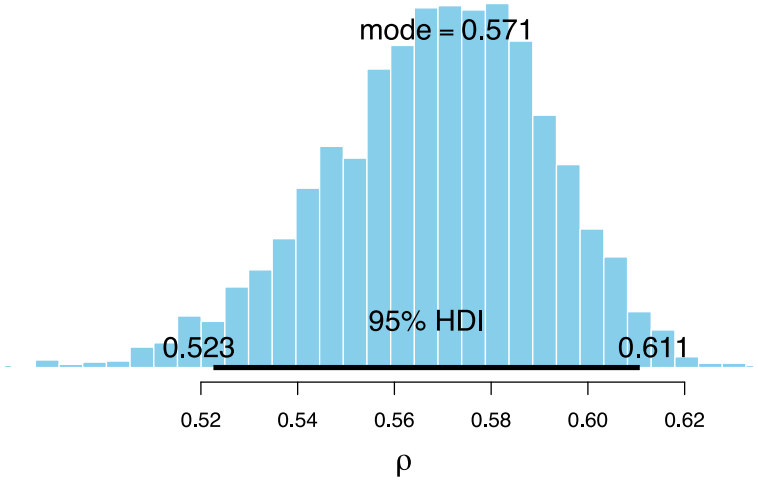
```
[1] 0.5696444
```

Modèle de régression, taille d'effet

```
bayes_R2(mod4)
```

```
    Estimate  Est.Error      Q2.5     Q97.5
R2 0.5684192 0.02304004 0.5188254 0.6083351
```

```
BEST::plotPost(bayes_R2(mod4, summary = FALSE), showMode = TRUE, xlab = 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.

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.

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()`.

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` (i.e., un modèle de classe `brmsfit`).

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` (i.e., un modèle de classe `brmsfit`).

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 `Howell1` 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 à la taille !

Modéliser alors la relation entre la taille (cm) et le log du poids (log-kg). Utiliser la dataframe `Howell1` 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

```
data(Howell1)

# on garde seulement les individus ayant moins de 18 ans
d <- Howell1 %>% 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; sigma = 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

Population-Level Effects:
          Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
Intercept    58.24      1.42    55.96    60.48 1.00     4035     2529
weight        2.72      0.07     2.61     2.83 1.00     3976     2953

Family Specific Parameters:
      Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
sigma     8.53      0.45     7.84     9.27 1.00     3563     2647

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).
```

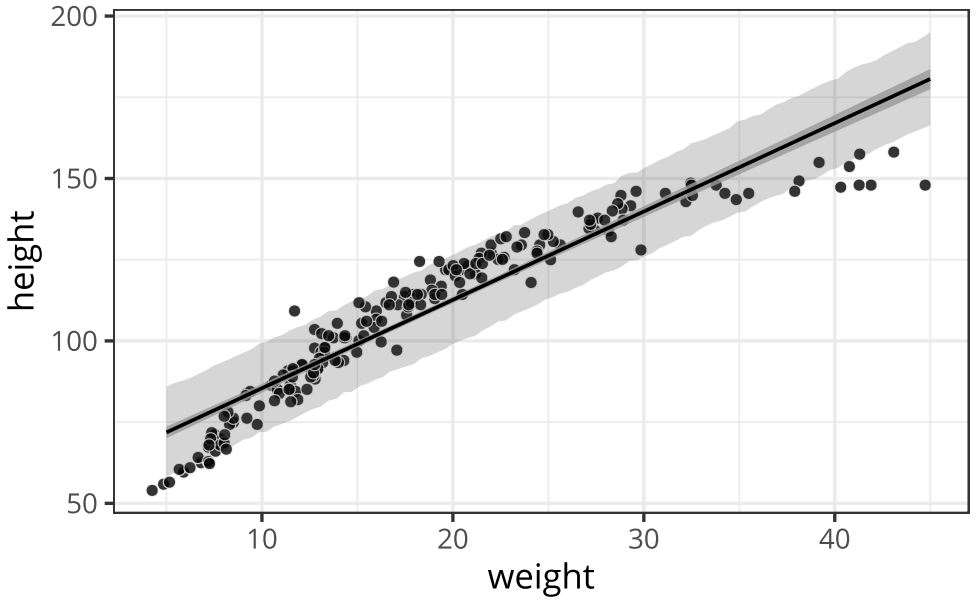

Représenter les prédictions du modèle

```
# 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 71.78587  8.699512 58.27737 85.92992 5.000000
2 72.91538  8.633055 59.36715 87.02888 5.404040
3 74.03239  8.512542 60.54960 87.69401 5.808081
4 75.04581  8.561011 61.12070 88.50054 6.212121
5 76.28063  8.562840 62.81625 89.68184 6.616162
6 77.01467  8.764254 62.86699 90.79444 7.020202
```

Représenter les prédictions du modèle

```
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

```
# on considère maintenant tous les individus
d <- Howell1

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


Proposition de solution

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

```
 Family: gaussian
  Links: mu = identity; sigma = 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

Population-Level Effects:
          Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
Intercept   -23.58      1.33   -25.68   -21.46 1.00     3969     2828
logweight    47.01      0.38    46.40    47.62 1.00     4013     3039

Family Specific Parameters:
      Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
sigma     5.15      0.15     4.91     5.40 1.00     4685     3041

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).
```

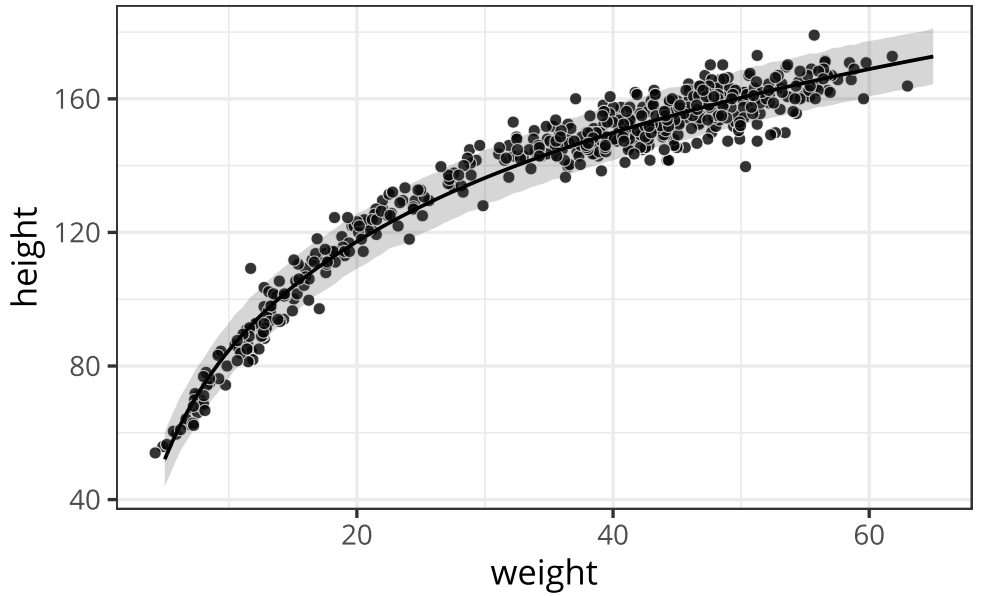

Représenter les prédictions du modèle

```
# 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 51.97258  5.173717 43.94201 60.32367 5.000000
2 57.41988  5.139568 49.18856 65.65924 5.606061
3 62.36721  5.122092 54.48203 70.83693 6.212121
4 66.66993  5.181096 58.28027 74.77304 6.818182
5 70.60074  5.216455 62.25285 79.03749 7.424242
6 74.36192  5.127477 65.92468 82.32891 8.030303
```


Représenter les prédictions du modèle

```
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
    )
```



Représenter les prédictions du modèle

```
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
    )
```