

Introduction à la modélisation statistique bayésienne

Ladislav Nalborczyk

GIPSA-lab, CNRS, Univ. Grenoble Alpes



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 Hackaton

Langage de la modélisation

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

$$\alpha \sim \text{Normal}(60, 10)$$

$$\beta \sim \text{Normal}(0, 10)$$

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

Langage de la modélisation

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

$$\alpha \sim \text{Normal}(60, 10)$$

$$\beta \sim \text{Normal}(0, 10)$$

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

Objectif de la séance : comprendre ce type de modèle.

Langage de la modélisation

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

$$\alpha \sim \text{Normal}(60, 10)$$

$$\beta \sim \text{Normal}(0, 10)$$

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

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

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

$$\alpha \sim \text{Normal}(60, 10)$$

$$\beta \sim \text{Normal}(0, 10)$$

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

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

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

$$\alpha \sim \text{Normal}(60, 10)$$

$$\beta \sim \text{Normal}(0, 10)$$

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

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

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

$$\alpha \sim \text{Normal}(60, 10)$$

$$\beta \sim \text{Normal}(0, 10)$$

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

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 ...
```

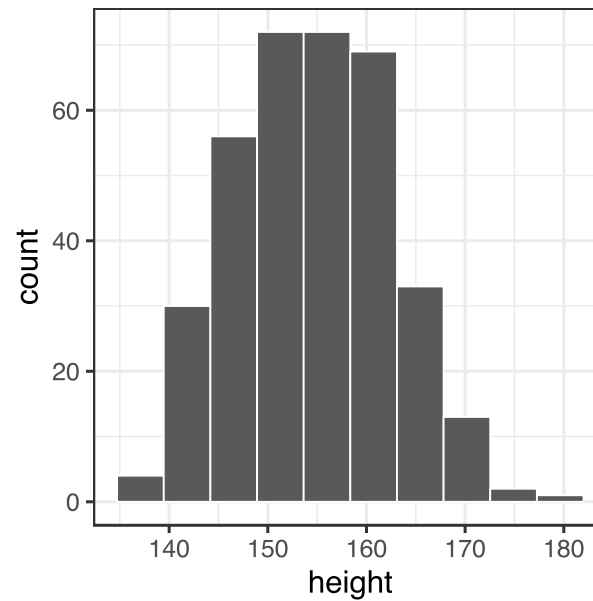
```
d2 <- d %>% filter(age >= 18)  
head(d2)
```

	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

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

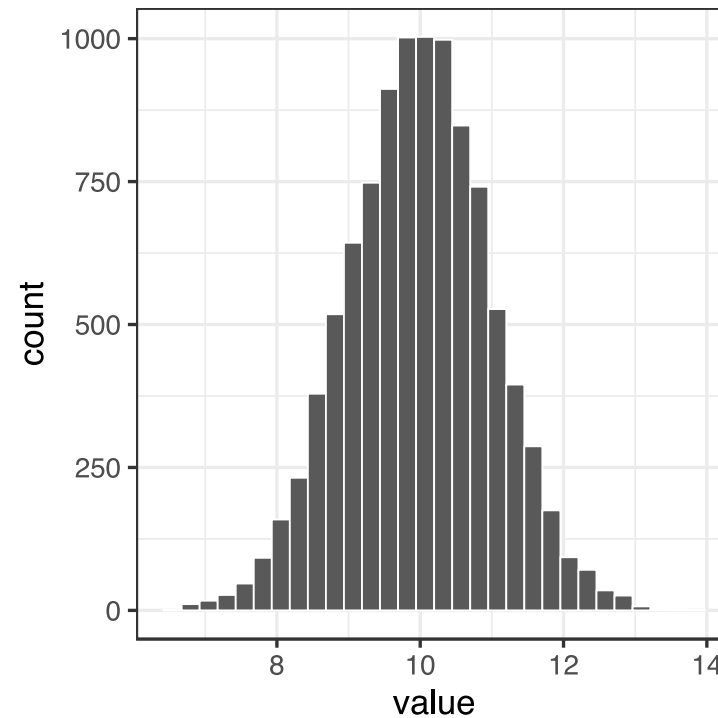
```
d2 %>%  
  ggplot(aes(x = height) ) +  
  geom_histogram(bins = 10, col = "white") +  
  theme_bw(base_size = 18)
```



Loi normale

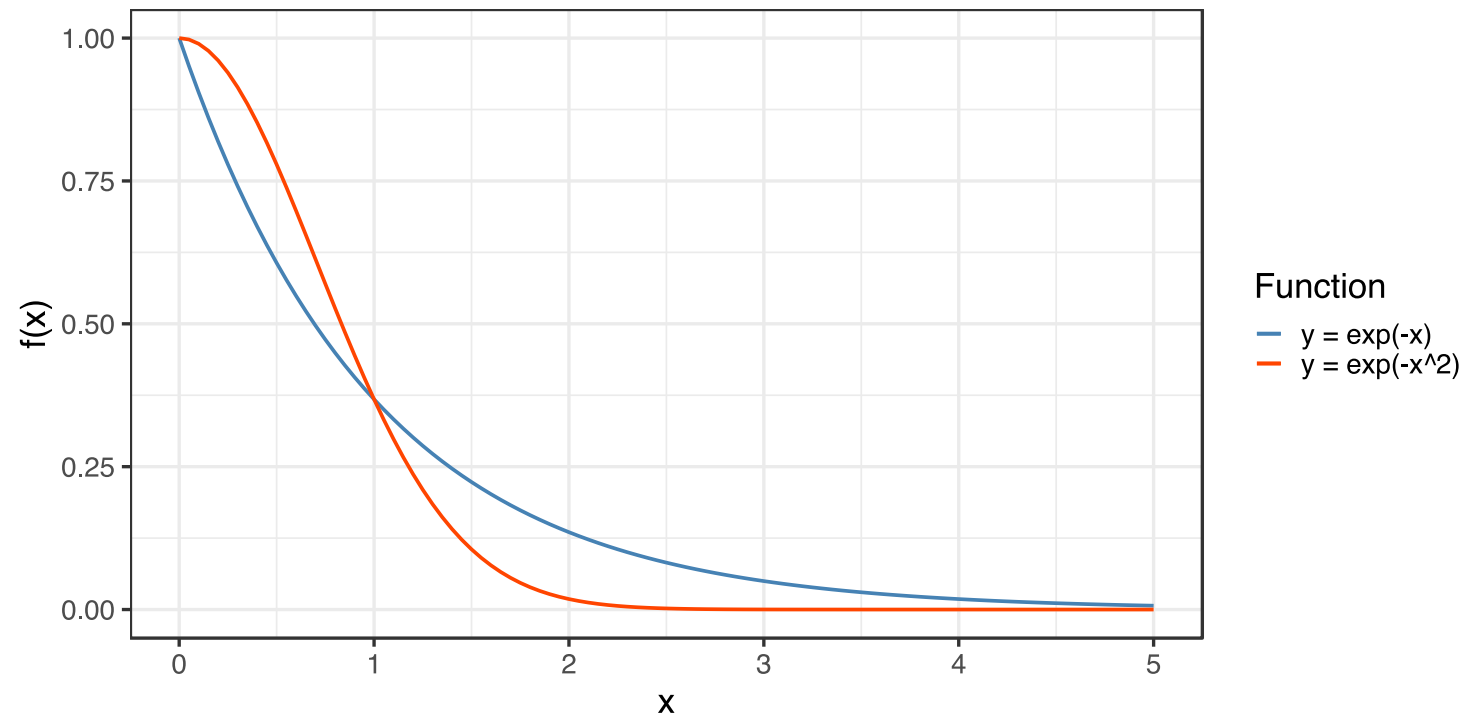
$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{1}{2\sigma^2} (\mu - x)^2 \right]$$

```
data.frame(value = rnorm(1e4, 10, 1) ) %>% # 10.000 samples from Normal(10, 1)
  ggplot(aes(x = value) ) +
  geom_histogram(col = "white") +
  theme_bw(base_size = 20)
```



D'où vient la loi normale ?

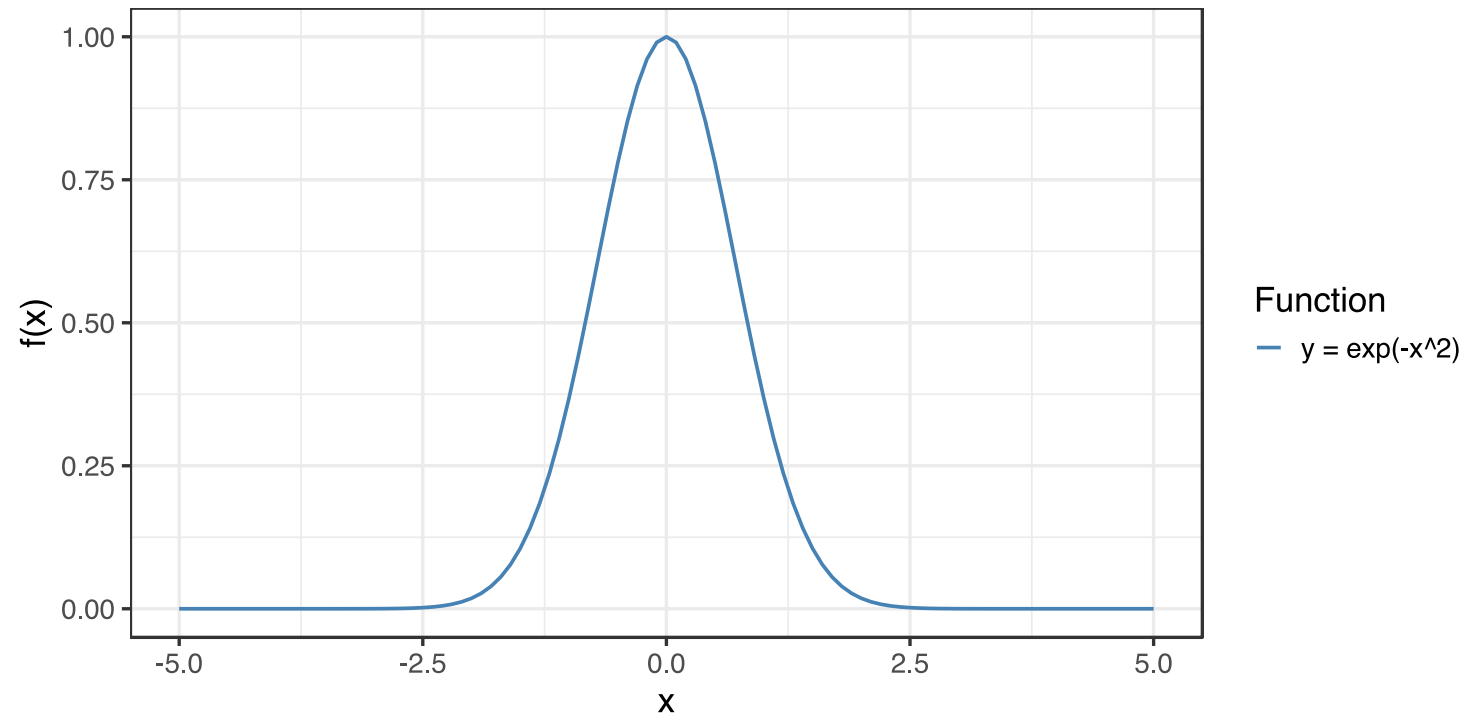
Certaines valeurs sont fortement probables (autour de la moyenne μ). Plus on s'éloigne, moins les valeurs sont probables (en suivant une décroissance exponentielle).



D'où vient la loi normale ?

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

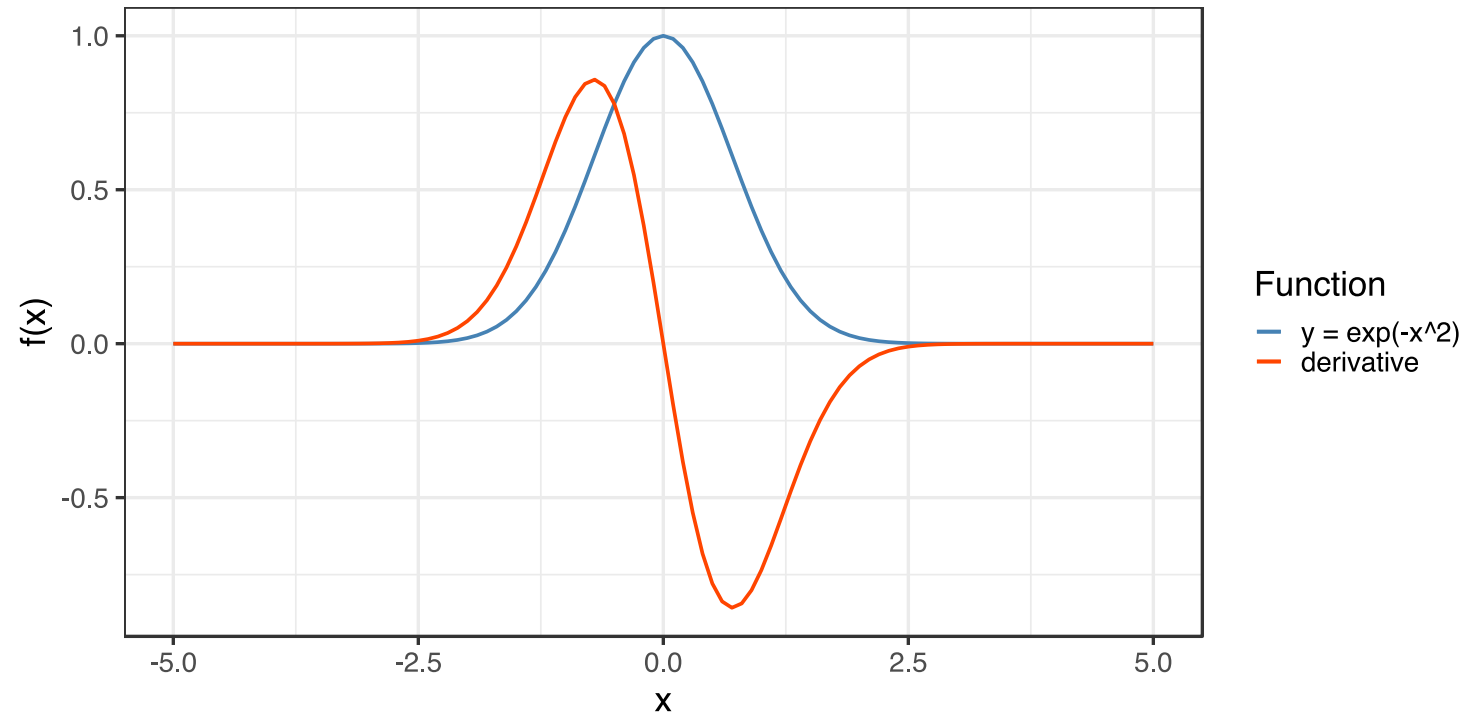
On étend notre fonction aux valeurs négatives.



D'où vient la loi normale ?

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

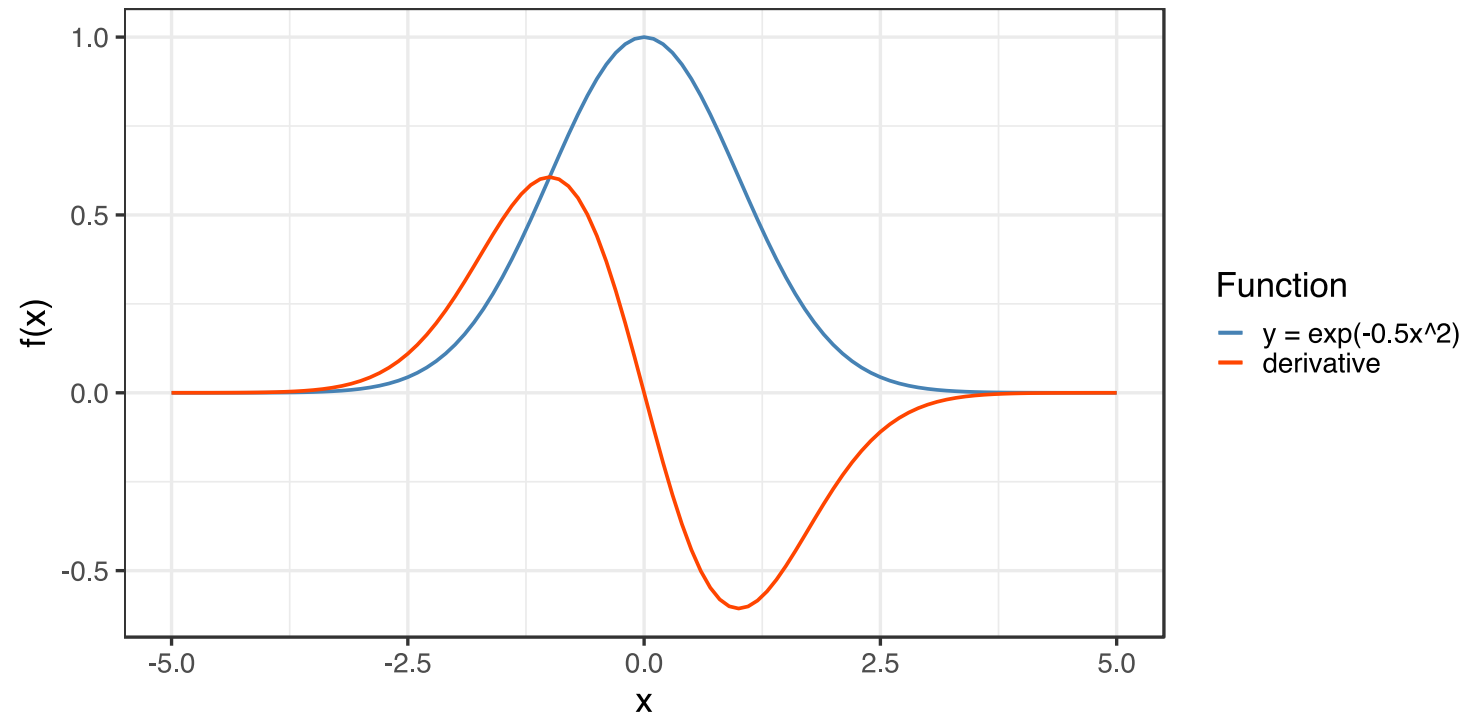
Les points d'inflexion nous donnent une bonne indication de là où la plupart des valeurs se trouvent (i.e., entre les points d'inflexion). Les pics de la dérivée nous montrent les points d'inflexion.



D'où vient la loi normale ?

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

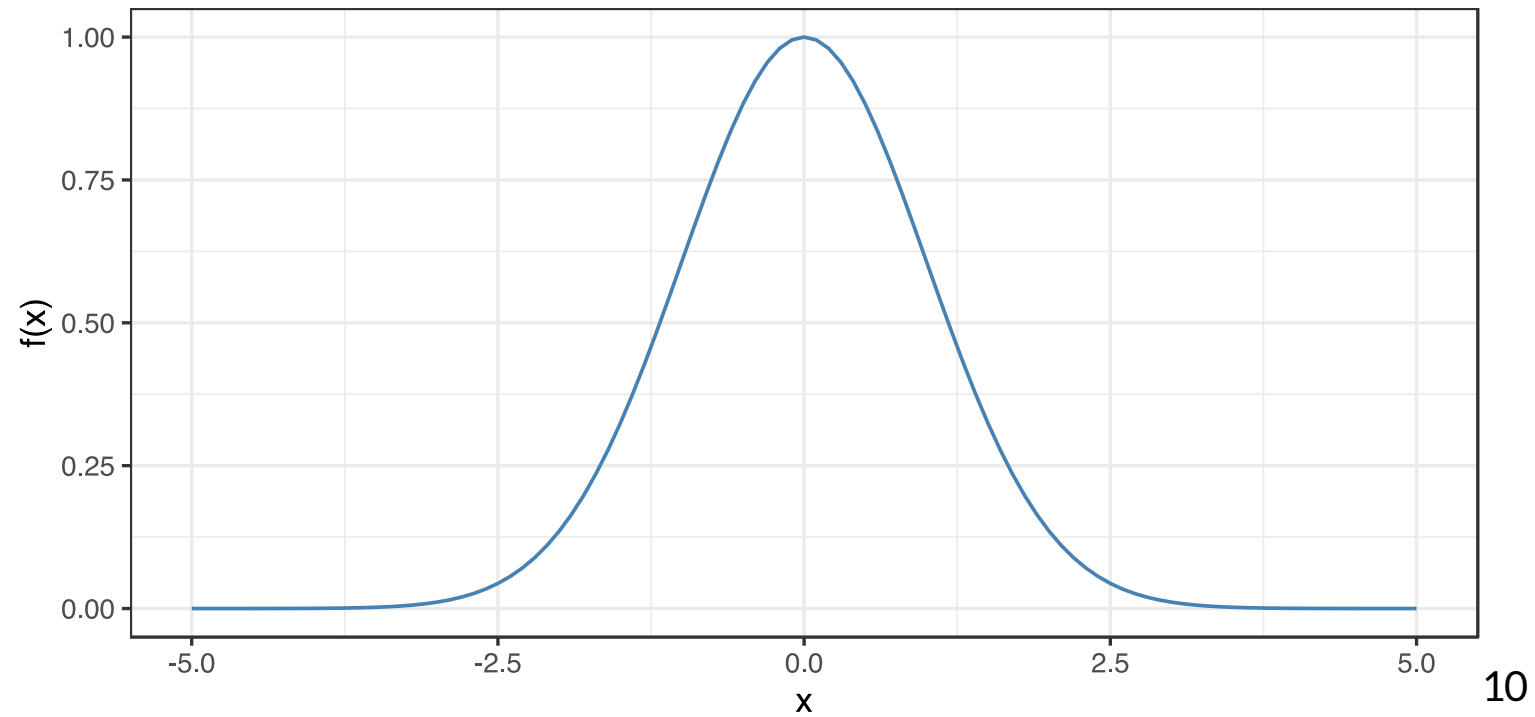
Ensuite on standardise la distribution de manière à ce que les deux points d'inflexion se trouvent à $x = -1$ et $x = 1$.



D'où vient la loi normale ?

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

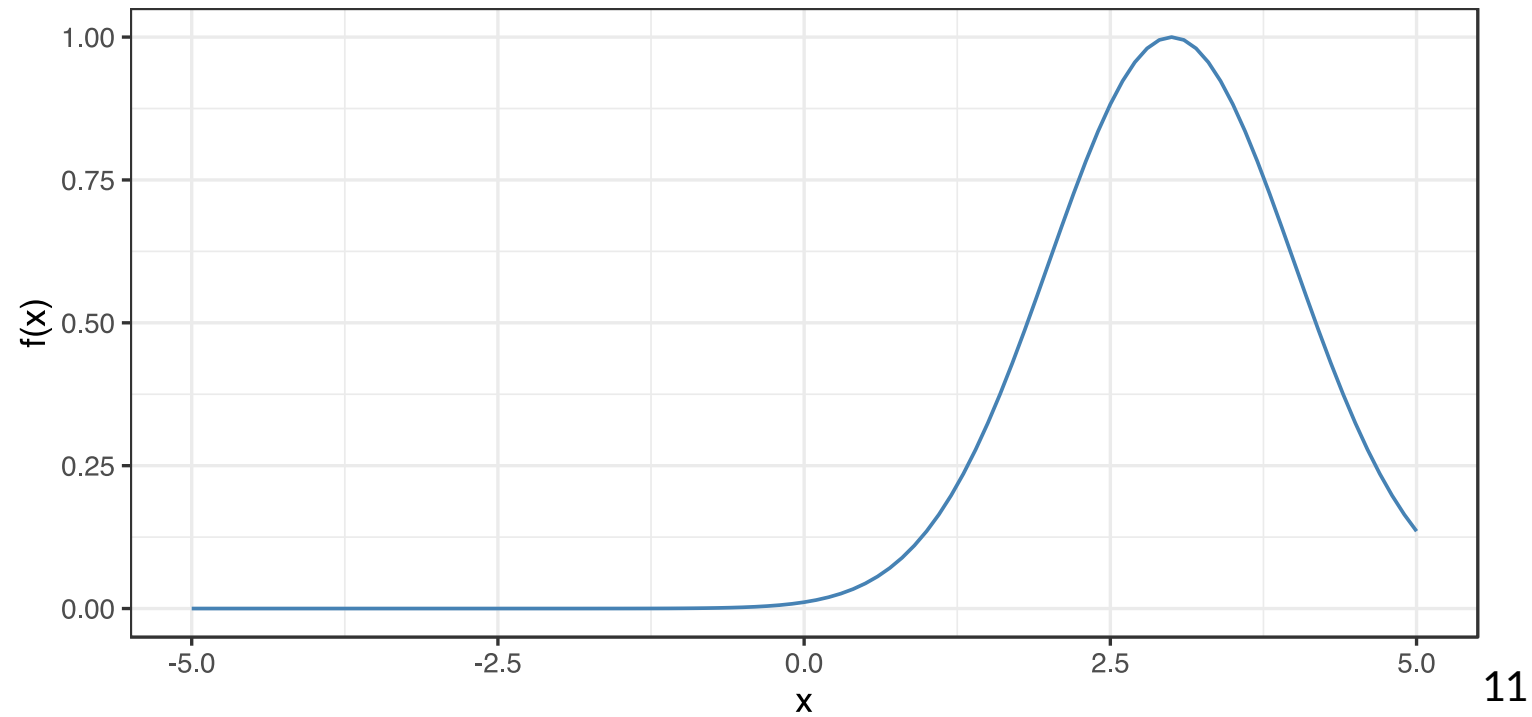
On insère un paramètre σ^2 pour contrôler la distance entre les points d'inflexion.



D'où vient la loi normale ?

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

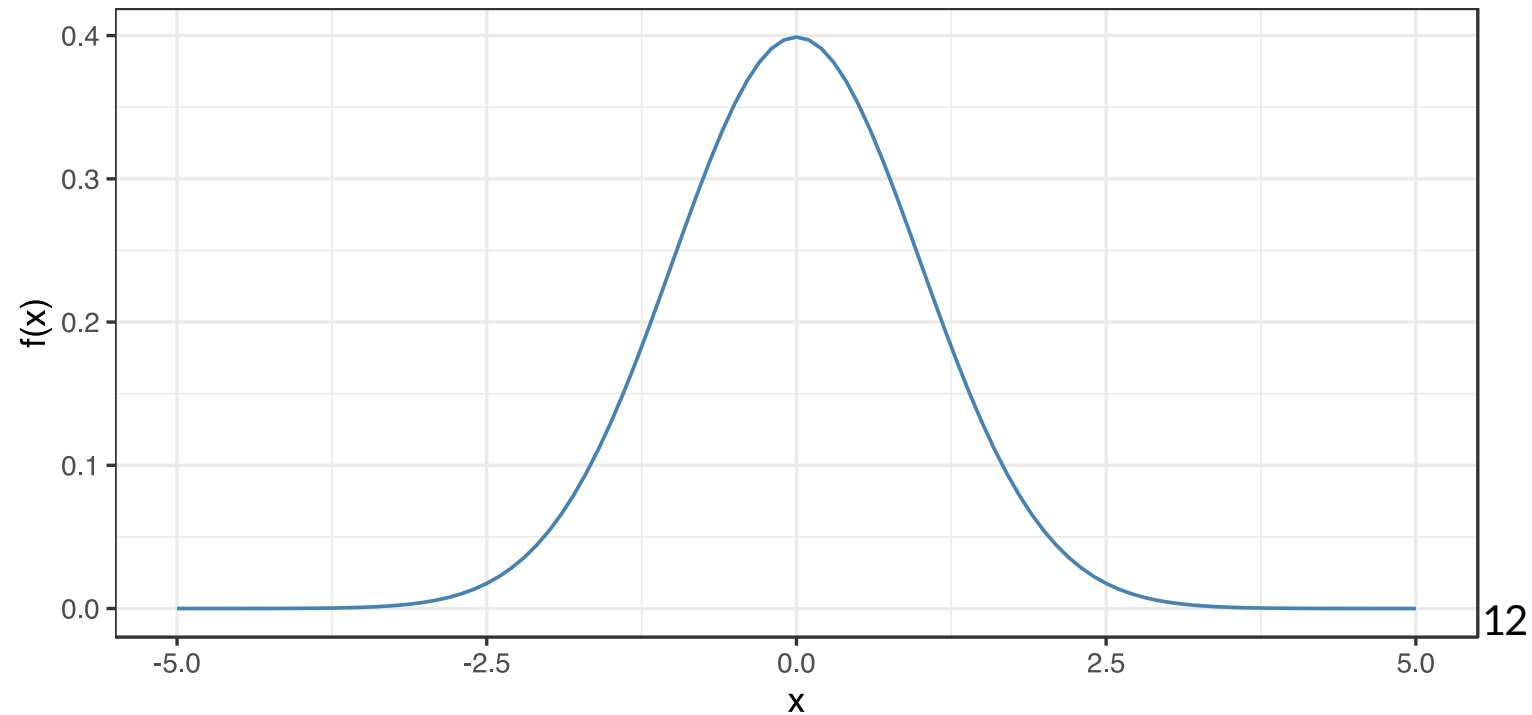
On insère ensuite un paramètre μ afin de pouvoir contrôler la position (la tendance centrale) de la distribution.



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

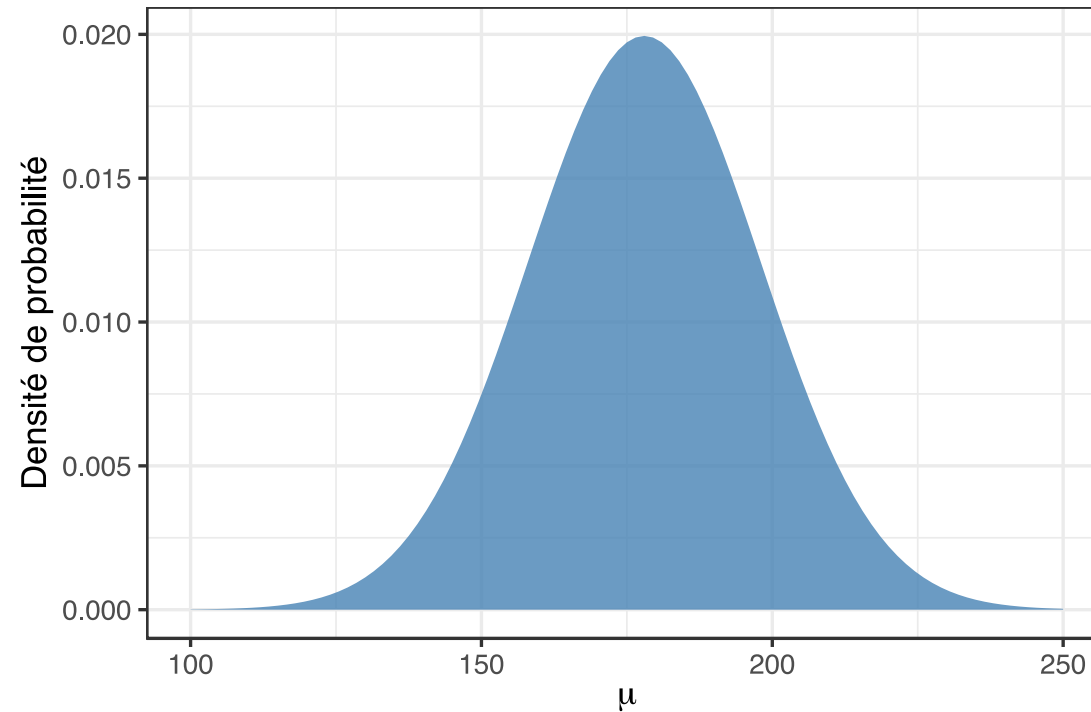
On définit $p(\mu, \sigma)$, la distribution a priori conjointe de tous les paramètres du modèle. On peut spécifier ces priors indépendamment pour chaque paramètre, sachant que $p(\mu, \sigma) = p(\mu)p(\sigma)$.

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

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

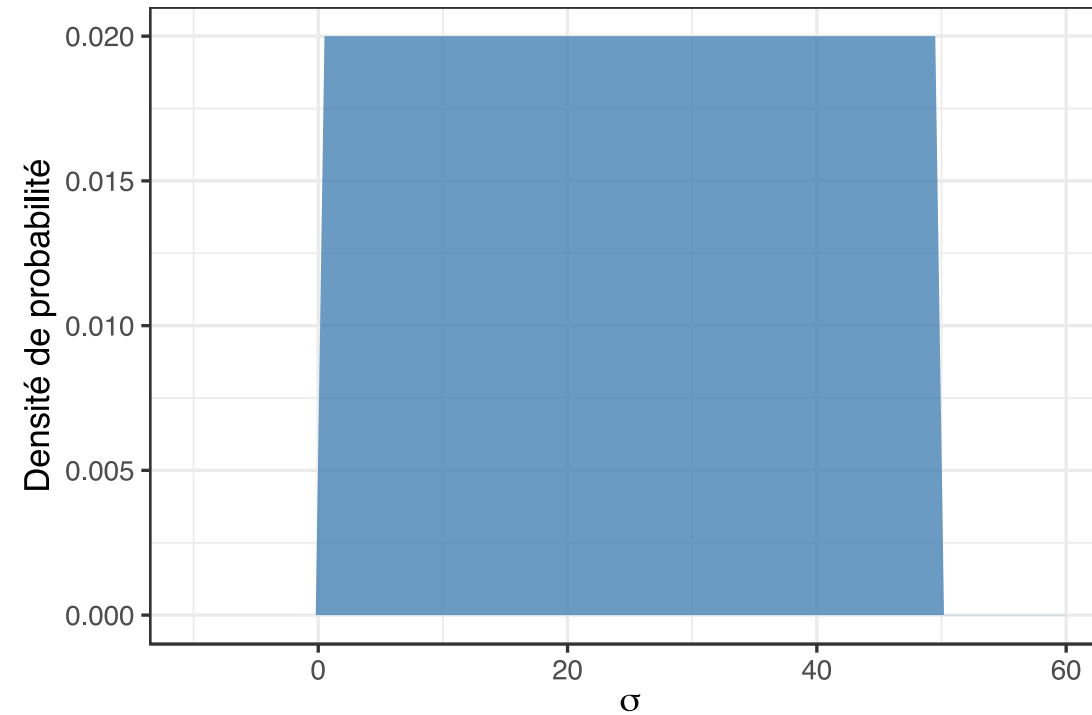
On définit $p(\mu, \sigma)$, la distribution a priori conjointe de tous les paramètres du modèle. On peut spécifier ces priors indépendamment pour chaque paramètre, sachant que $p(\mu, \sigma) = p(\mu)p(\sigma)$.

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

Modèle gaussien

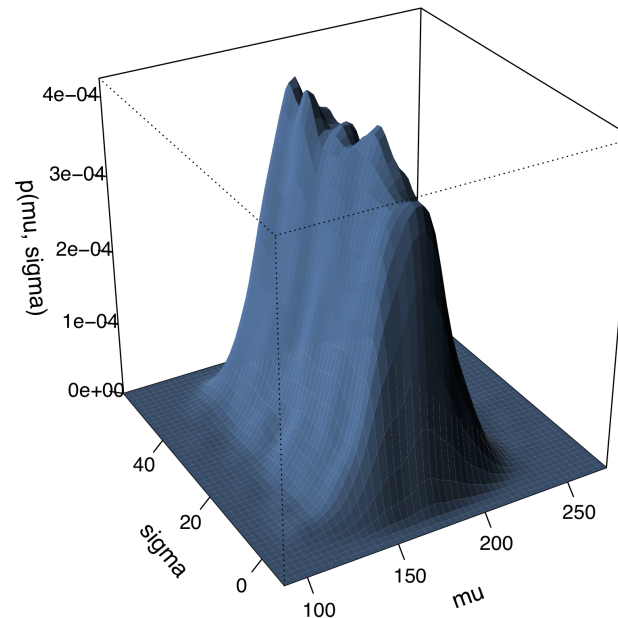
On définit $p(\mu, \sigma)$, la distribution a priori conjointe de tous les paramètres du modèle. On peut spécifier ces priors indépendamment pour chaque paramètre, sachant que $p(\mu, \sigma) = p(\mu)p(\sigma)$.

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



Visualiser le prior

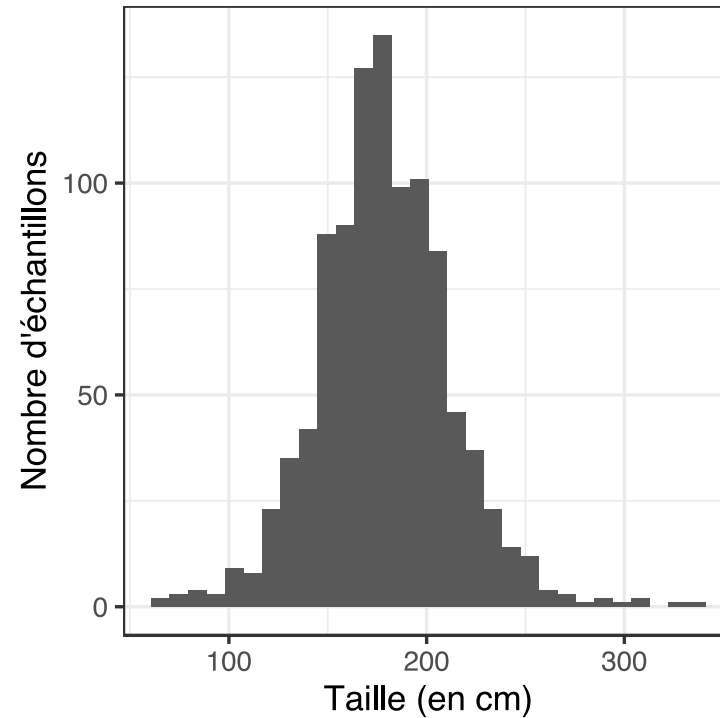
```
library(ks)
sample_mu <- rnorm(1e4, 178, 20) # prior on mu
sample_sigma <- runif(1e4, 0, 50) # prior on sigma
prior <- data.frame(cbind(sample_mu, sample_sigma) ) # multivariate prior
H.scv <- Hscv(x = prior, verbose = TRUE)
fhat_prior <- kde(x = prior, H = H.scv, compute.cont = TRUE)
plot(
  fhat_prior, 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

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

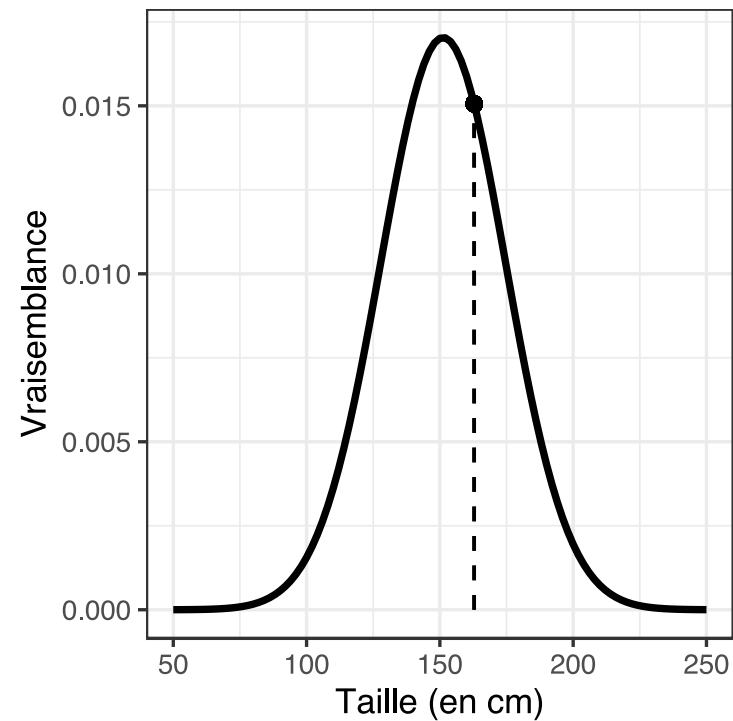
data.frame(x = rnorm(1000, sample_mu, sample_sigma) ) %>%
  ggplot(aes(x) ) +
  geom_histogram() +
  labs(x = "Taille (en cm)", y = "Nombre d'échantillons") +
  theme_bw(base_size = 20)
```



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

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

Distribution postérieure

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

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

Distribution postérieure

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

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

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

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

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

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

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

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

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

```
# 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)

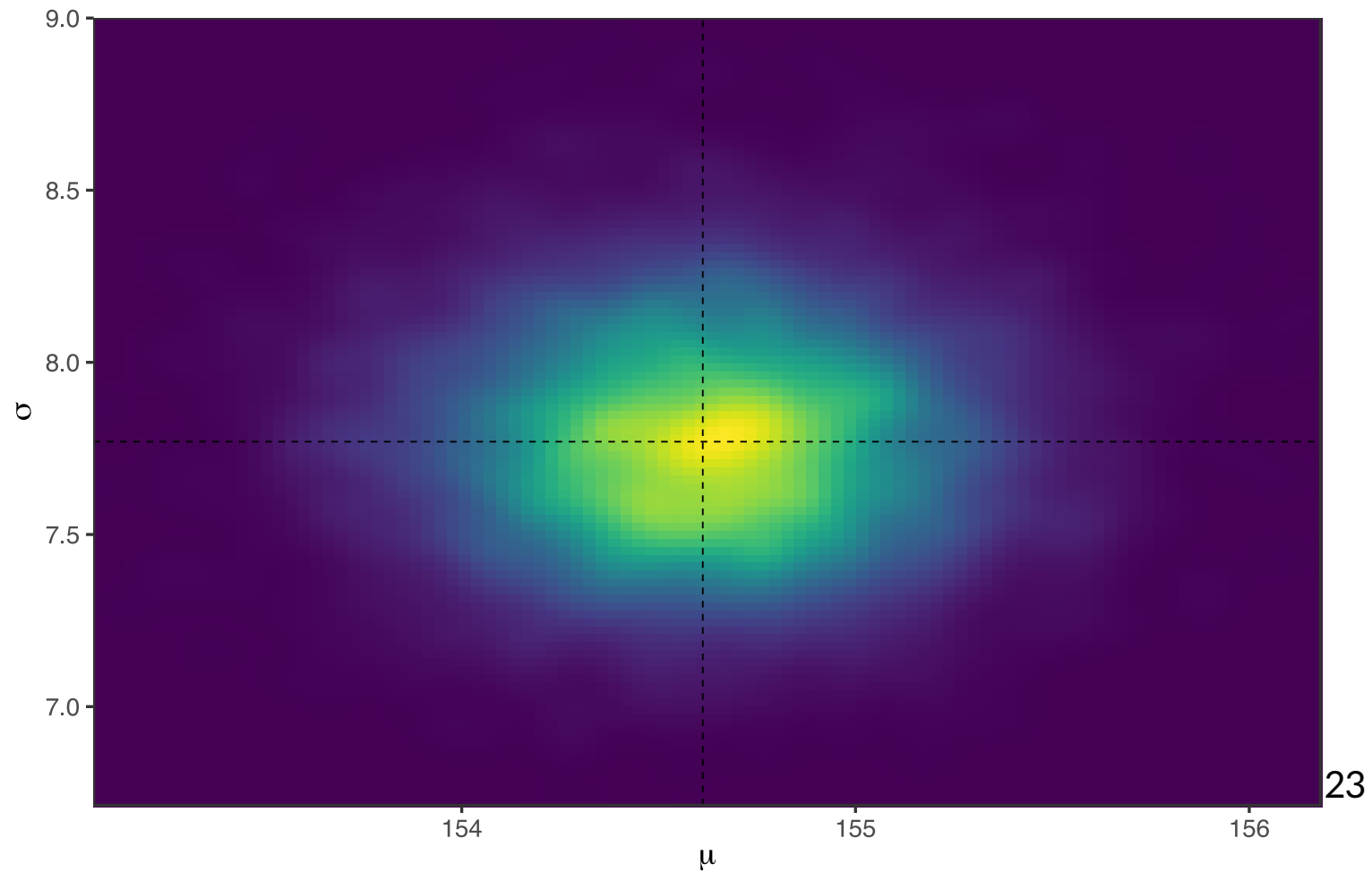
# calcul de la log-vraisemblance (pour chaque couple de mu et sigma)
post$LL <-
  sapply(
    1:nrow(post),
    function(i) sum(dnorm(
      d2$height,
      mean = post$mu[i],
      sd = post$sigma[i],
      log = TRUE)
    )
  )

# 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) )
```

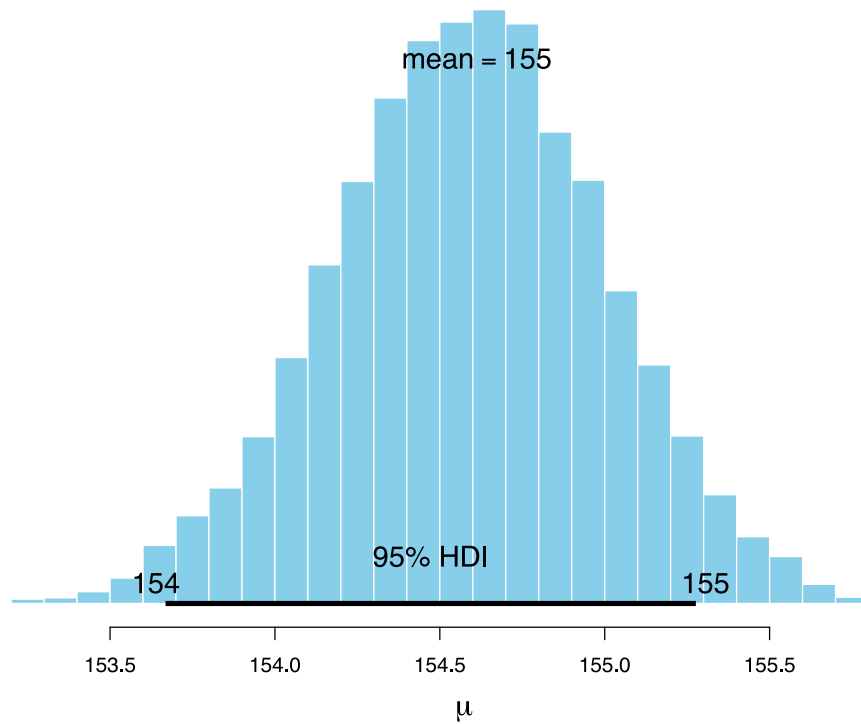
Distribution postérieure - grid approximation

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

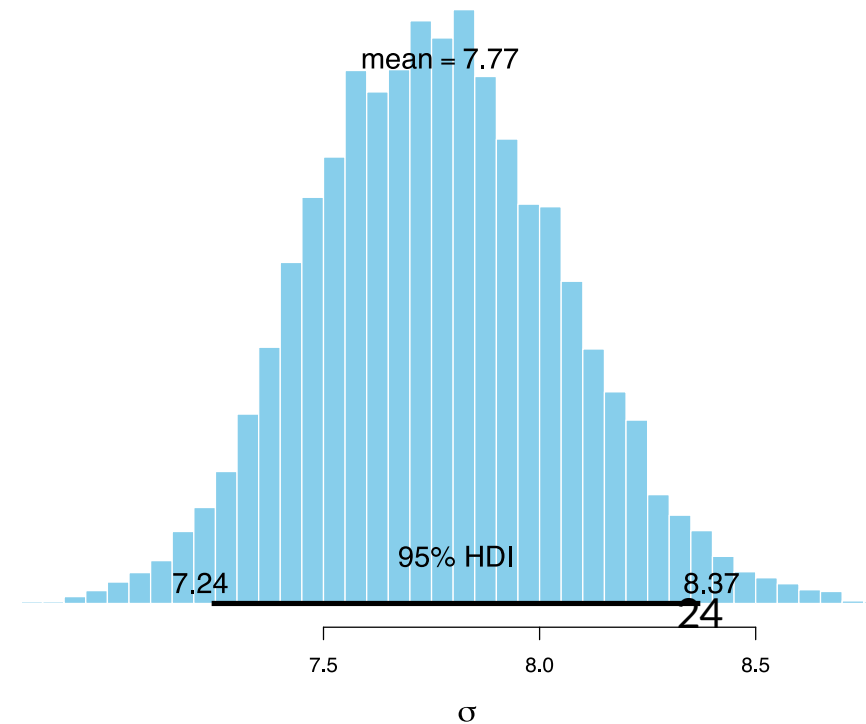


Distribution postérieure - distributions marginales

```
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 :

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

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 :

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

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 :

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

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

```
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 prédictions du modèle
fitted(fit, ...)
predict(fit, ...)
marginal_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	154.581685	0.4130306	153.774346	155.355928
sigma	7.766121	0.2870718	7.225429	8.336149

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 154.58 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.

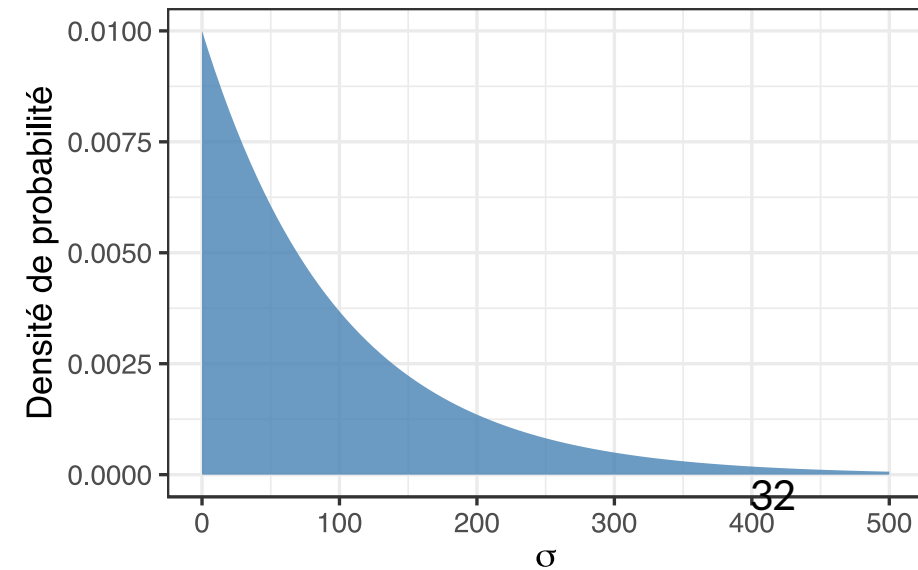
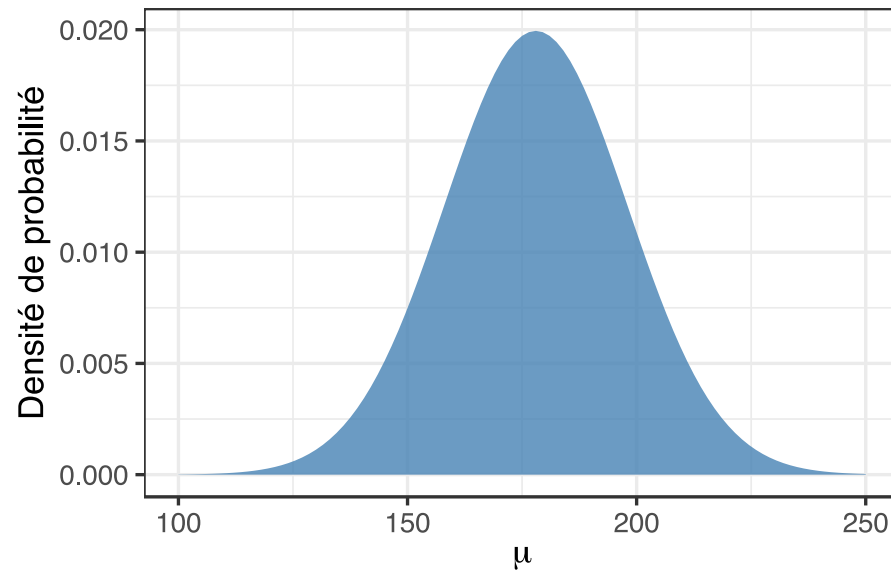
La fonction `get_prior()` permet de visualiser une liste des priors par défaut ainsi que de tous les priors qu'on peut spécifier, sachant une certaine formule (i.e., une manière d'écrire notre modèle) et un jeu de données.

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

	prior	class	coef	group	resp	dpar	nlpar	bound
1	student_t(3, 154, 10)	Intercept						
2	student_t(3, 0, 10)	sigma						

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)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
```

Population-Level Effects:

	Estimate	Est.Error	l-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
Intercept	154.61	0.41	153.83	155.42	1.00	3604	2502

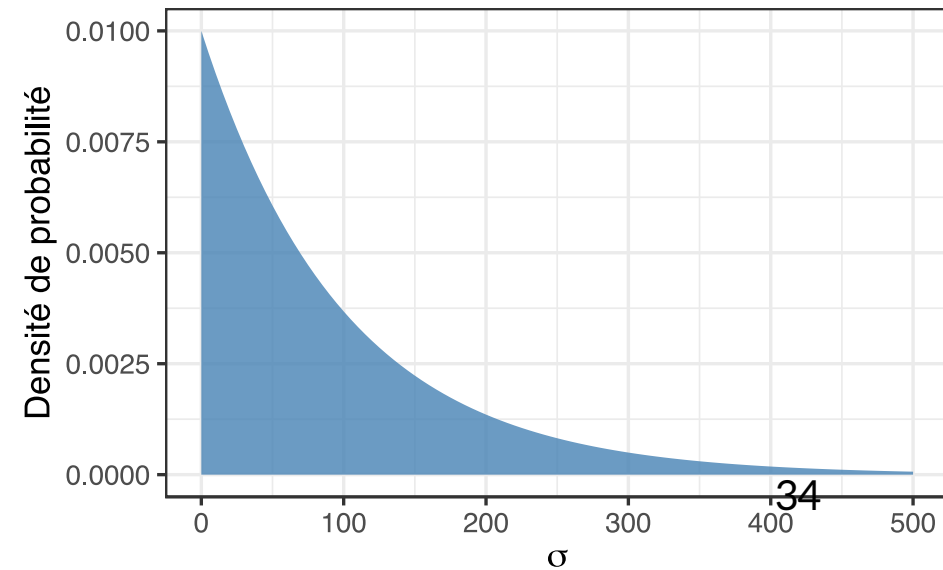
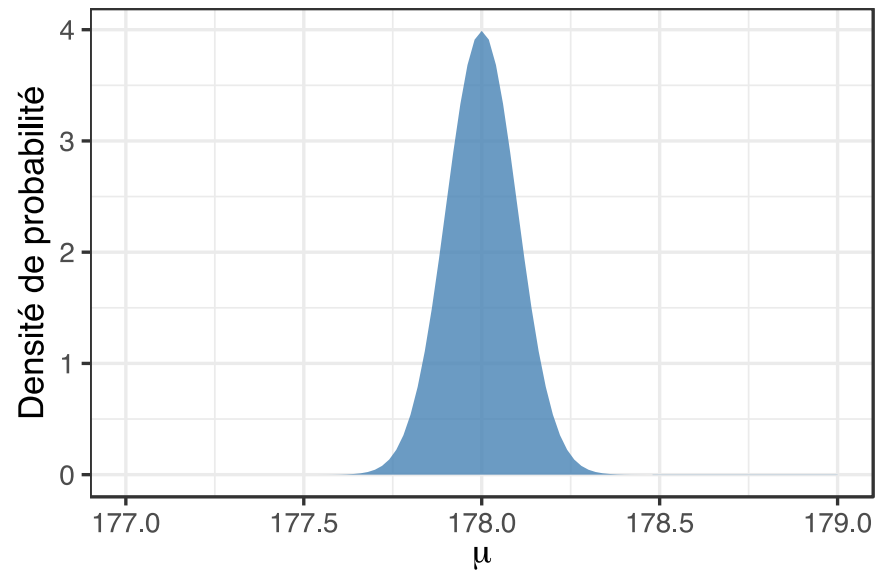
Family Specific Parameters:

	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	3651	2549

Samples were drawn 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)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
```

Population-Level Effects:

	Estimate	Est.Error	l-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
Intercept	177.87	0.10	177.67	178.06	1.00	3053	2313

Family Specific Parameters:

	Estimate	Est.Error	l-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
sigma	24.59	0.93	22.90	26.55	1.00	2619	2542

Samples were drawn 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 :

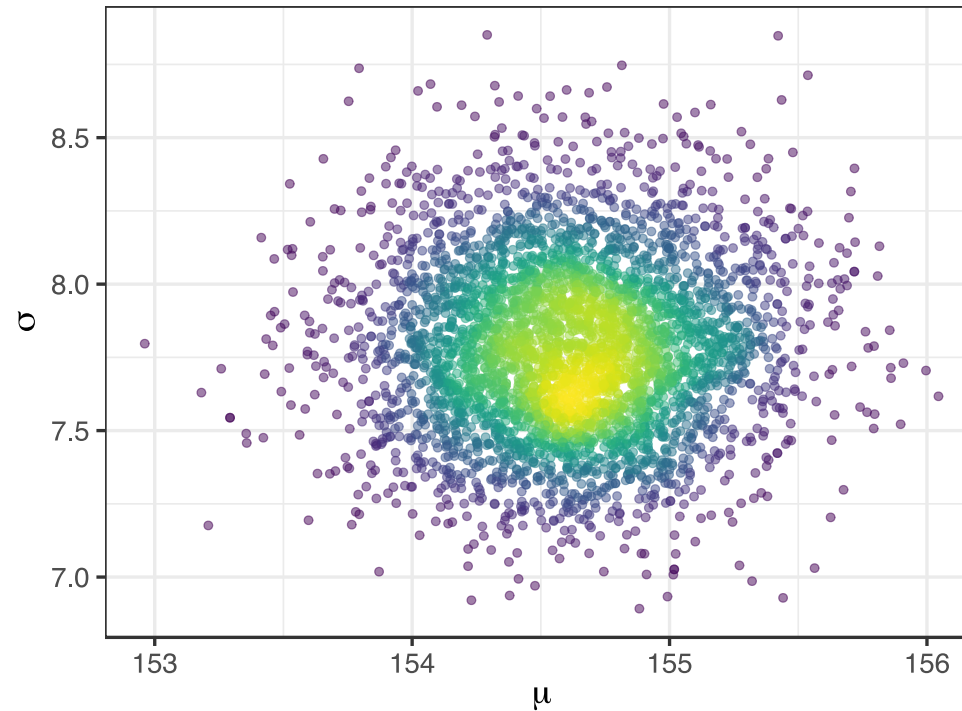
$$\sigma_{post} = 1/\sqrt{n}$$

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) +  
  theme_bw(base_size = 20) +  
  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	154.5120	7.951412	-1226.913	1.0187405
2	154.6809	7.564516	-1226.848	1.1616618
3	154.7534	7.271092	-1228.152	0.3024678
4	154.6569	7.798223	-1226.666	1.0948305
5	154.7599	7.564664	-1226.903	1.0763746
6	154.4327	7.824484	-1226.768	1.0358084

Récupérer les échantillons de la distribution postérieure

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

	b_Intercept	sigma	lp__	density
1	154.5120	7.951412	-1226.913	1.0187405
2	154.6809	7.564516	-1226.848	1.1616618
3	154.7534	7.271092	-1228.152	0.3024678
4	154.6569	7.798223	-1226.666	1.0948305
5	154.7599	7.564664	-1226.903	1.0763746
6	154.4327	7.824484	-1226.768	1.0358084

```
# gets the median and the 95% credible interval  
t(apply(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)
```

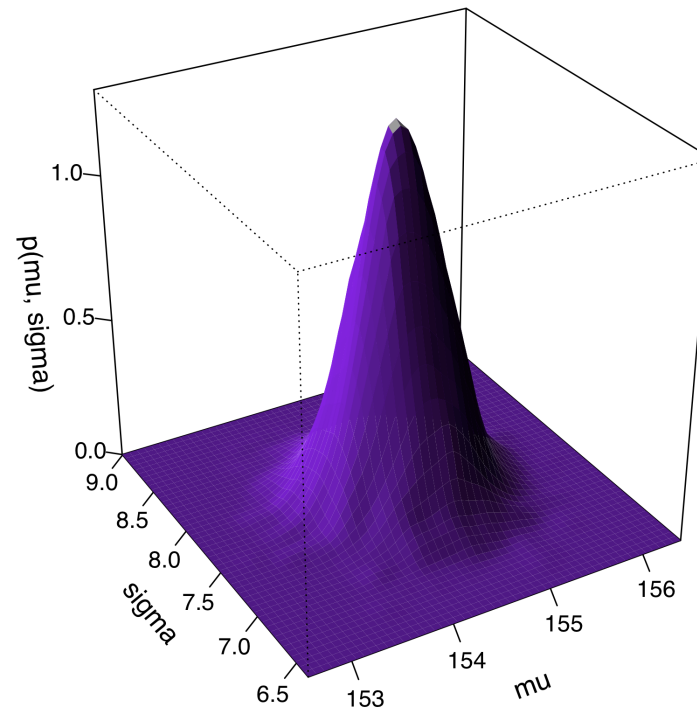
	b_Intercept	sigma	lp__	density
1	154.5120	7.951412	-1226.913	1.0187405
2	154.6809	7.564516	-1226.848	1.1616618
3	154.7534	7.271092	-1228.152	0.3024678
4	154.6569	7.798223	-1226.666	1.0948305
5	154.7599	7.564664	-1226.903	1.0763746
6	154.4327	7.824484	-1226.768	1.0358084

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

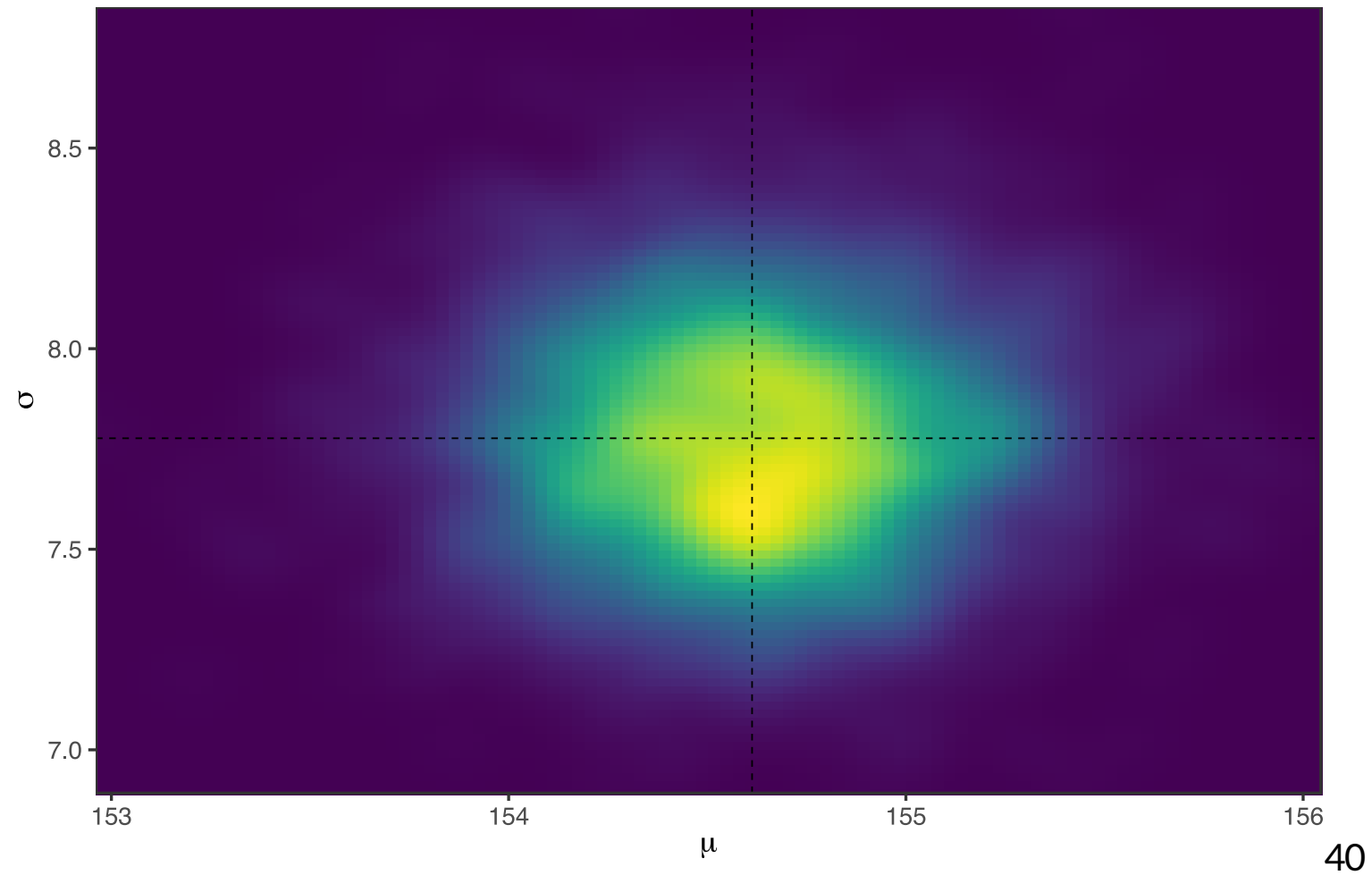
	2.5%	50%	97.5%
b_Intercept	153.830044	154.61231	155.418797
sigma	7.246072	7.76851	8.365361

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")
```



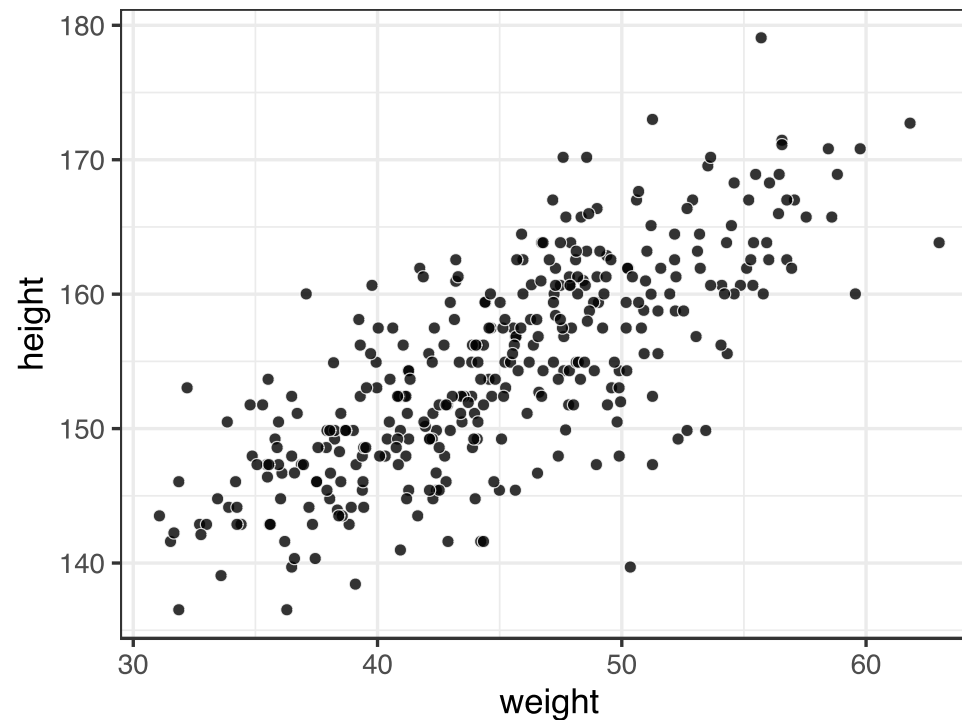
Visualiser la distribution postérieure



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) +  
  theme_bw(base_size = 20)
```



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	StdDev	2.5%	97.5%
(Intercept)	113.88	1.91	110.13	117.63
weight	0.91	0.04	0.82	0.99

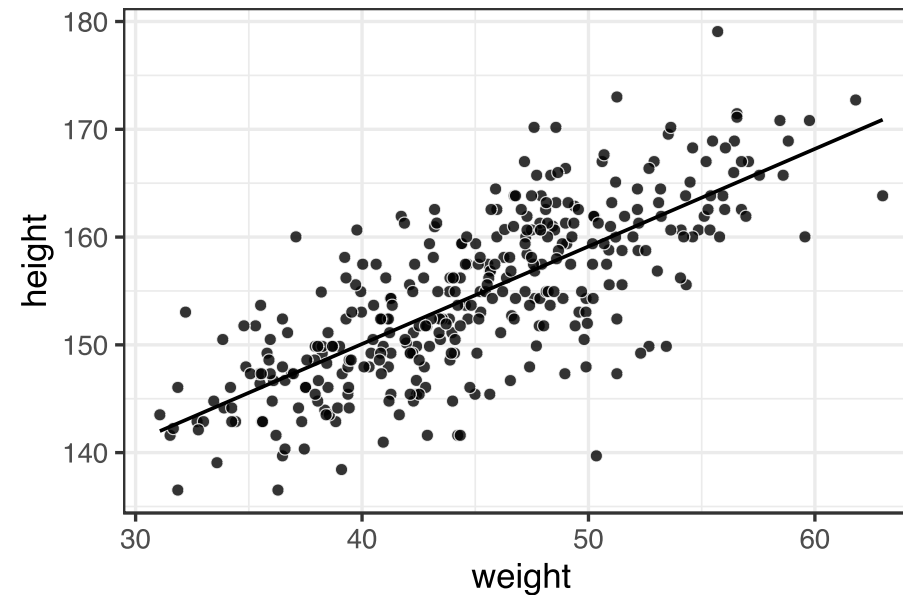
Régression linéaire à un prédicteur continu

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

$$\mu_i = \alpha + \beta x_i$$

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

	Mean	StdDev	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

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

$$\mu_i = \alpha + \beta x_i$$

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

$$\beta \sim \text{Normal}(0, 10)$$

$$\sigma \sim \text{Exponential}(0.01)$$

Régression linéaire à un prédicteur continu

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

$$\mu_i = \alpha + \beta x_i$$

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

$$\beta \sim \text{Normal}(0, 10)$$

$$\sigma \sim \text{Exponential}(0.01)$$

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

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

$$\mu_i = \alpha + \beta x_i$$

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

$$\beta \sim \text{Normal}(0, 10)$$

$$\sigma \sim \text{Exponential}(0.01)$$

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	113.9091739	1.90340939	110.1174643	117.5865082
b_weight	0.9044274	0.04181662	0.8241147	0.9872964
sigma	5.1053312	0.19635602	4.7449906	5.5041008
lp__	-1083.3609326	1.18601814	-1086.4338689	-1081.9880833

- $\beta = 0.90$, 95% CrI [0.82, 0.99] nous indique qu'une augmentation de 1kg entraîne une augmentation de 0.90cm.
- $\alpha = 113.91$, 95% CrI [110.12, 117.59] représente la taille moyenne quand le poids est égal à 0kg...

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  
)
```

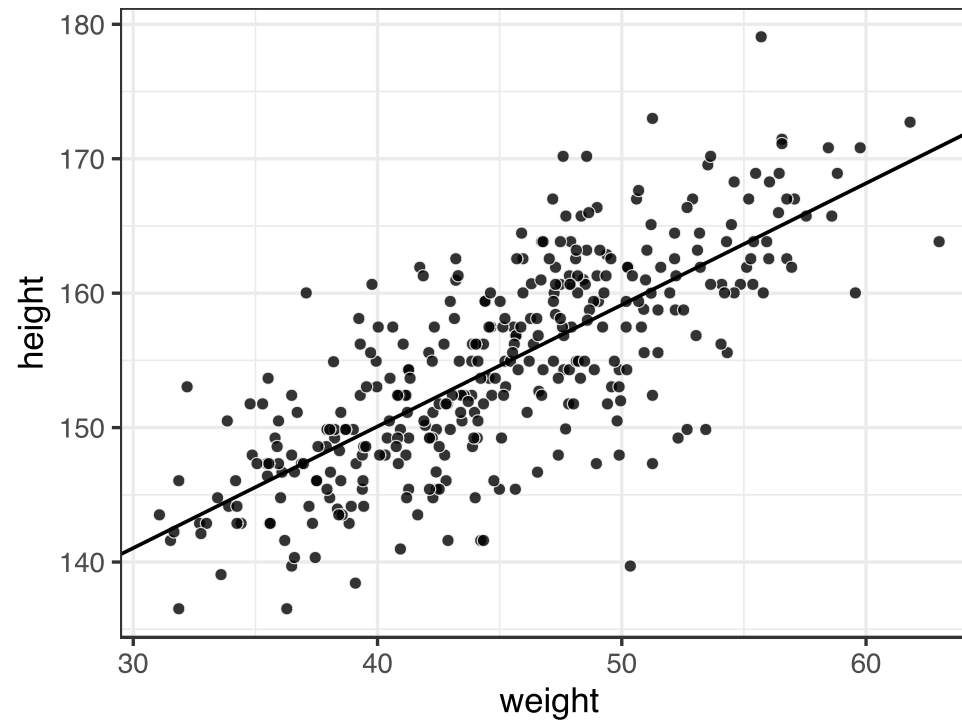
```
fixef(mod5) # retrieves the fixed effects estimates
```

	Estimate	Est.Error	Q2.5	Q97.5
Intercept	154.606246	0.27712175	154.040710	155.133690
weight.c	0.904148	0.04119113	0.819957	0.982267

- Après avoir centré la réponse, l'intercept représente 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) +  
  theme_bw(base_size = 20)
```



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) )

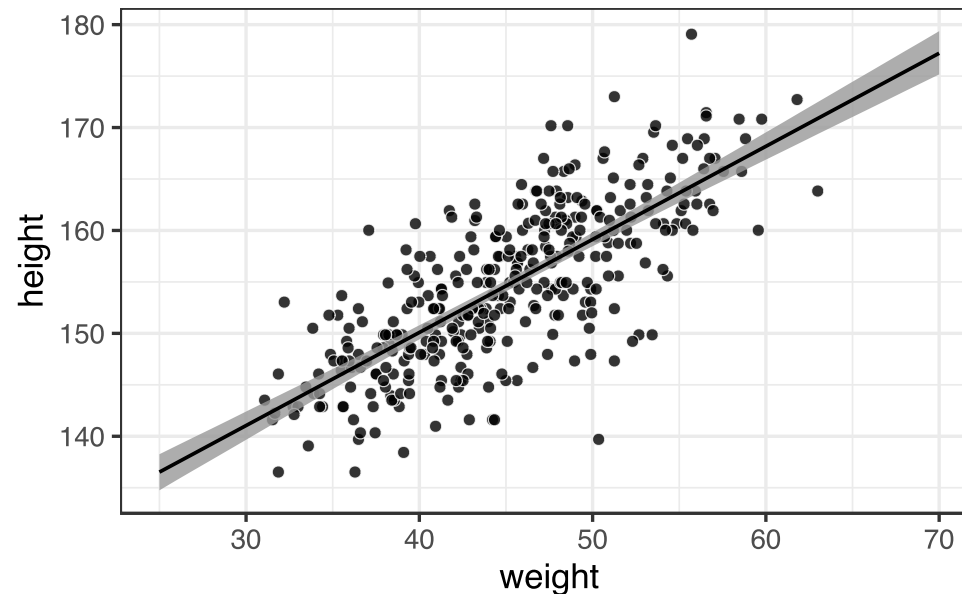
# on récupère les prédictions du modèle pour ces valeurs de poids
mu <- data.frame(fitted(mod4, newdata = weight.seq) ) %>% bind_cols(weight.seq)

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

	Estimate	Est.Error	Q2.5	Q97.5	weight
1	136.5199	0.8807902	134.7469	138.2552	25
2	137.4243	0.8410521	135.7460	139.0875	26
3	138.3287	0.8015256	136.7262	139.9143	27
4	139.2331	0.7622435	137.6986	140.7338	28
5	140.1376	0.7232456	138.6741	141.5626	29
6	141.0420	0.6845806	139.6550	142.4029	30
7	141.9464	0.6463082	140.6439	143.2329	31
8	142.8509	0.6085024	141.6297	144.0550	32
9	143.7553	0.5712560	142.6001	144.8849	33
10	144.6597	0.5346857	143.5895	145.7123	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  
  ) +  
  theme_bw(base_size = 20)
```



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) )

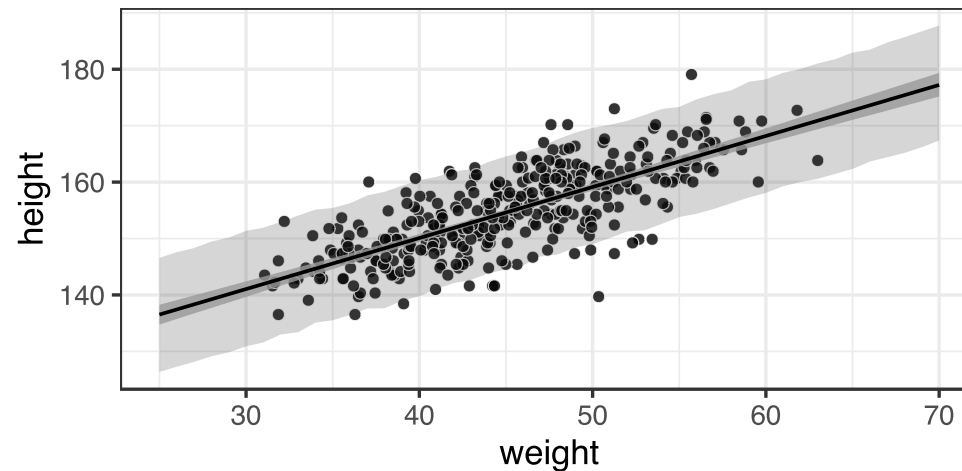
# on récupère les prédictions du modèle pour ces valeurs de poids
pred_height <- data.frame(predict(mod4, newdata = weight.seq) ) %>% bind_cols(weight.seq)

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

	Estimate	Est.Error	Q2.5	Q97.5	weight
1	136.4334	5.156982	126.3597	146.5687	25
2	137.4228	5.172193	127.2720	147.5865	26
3	138.2495	5.196426	128.1297	148.3102	27
4	139.2875	5.149738	129.1324	149.4651	28
5	140.0878	5.108313	129.8216	150.1605	29
6	141.0650	5.250530	130.8891	151.3670	30
7	141.8594	5.093503	131.5897	151.9159	31
8	142.8869	5.048090	132.9920	152.9313	32
9	143.7208	5.226576	133.3834	154.0676	33
10	144.6954	5.202136	135.0953	155.1228	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
  ) +
  theme_bw(base_size = 20)
```



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.

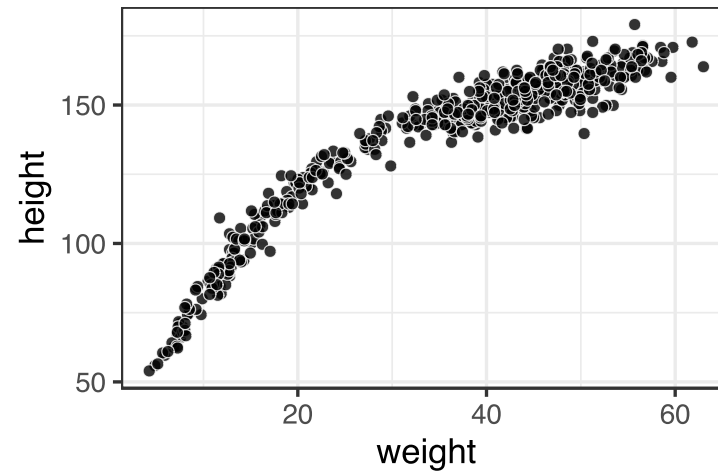
Incertitude épistémique : La distribution a posteriori ordonne toutes les combinaisons possibles des valeurs des paramètres selon leurs plausibilités relatives.

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

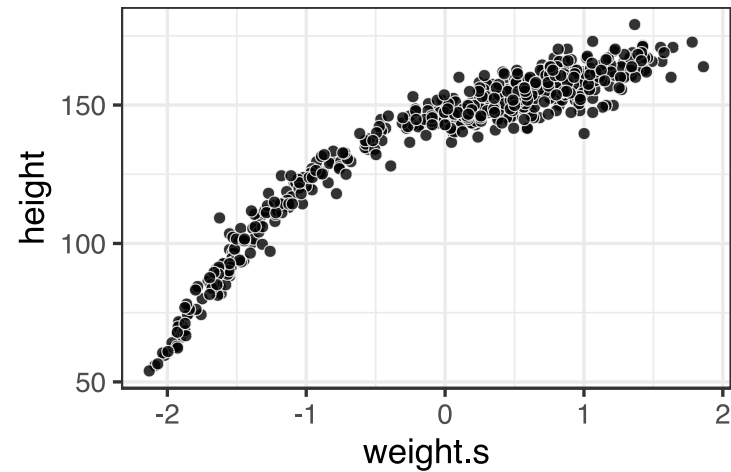
```
d %>% # on utilise d au lieu de d2
  ggplot(aes(x = weight, y = height) ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
  theme_bw(base_size = 20)
```



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) +  
  theme_bw(base_size = 20)
```



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

```
[1] -2.712698e-18  1.000000e+00
```

Scores standardisés

Pourquoi standardiser les prédicteurs ?

- **Interprétation.** Un changement d'une unité du prédicteur correspond à un changement d'un écart-type sur la réponse.
Permet de comparer les coefficients de plusieurs prédicteurs.
- **Fitting.** Quand les prédicteurs contiennent de grandes valeurs, cela peut poser des problèmes...

Modèle de régression polynomiale - exercice

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

$$\mu_i = \alpha + \beta_1 x_i + \beta_2 x_i^2$$

$$\alpha \sim \text{Normal}(156, 100)$$

$$\beta_1, \beta_2 \sim \text{Normal}(0, 10)$$

$$\sigma \sim \text{Exponential}(0.01)$$

À vous de construire 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)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
```

Population-Level Effects:

	Estimate	Est.Error	l-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
Intercept	146.67	0.38	145.93	147.40	1.00	3571	2769
weight.s	21.39	0.29	20.83	21.97	1.00	3451	2955
Iweight.sE2	-8.42	0.28	-8.97	-7.90	1.00	3536	2972

Family Specific Parameters:

	Estimate	Est.Error	l-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
sigma	5.78	0.18	5.44	6.14	1.00	4182	2934

Samples were drawn 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) )

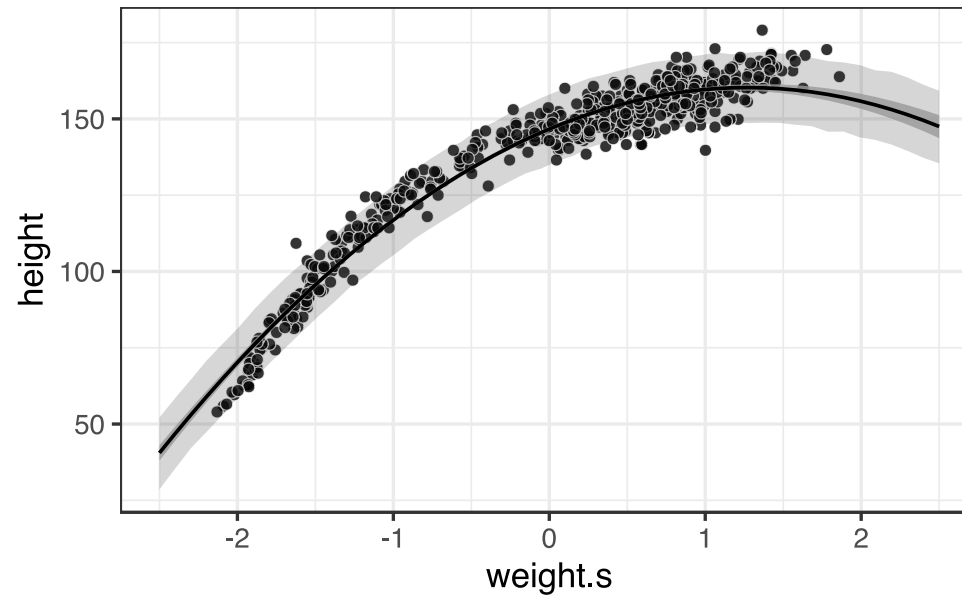
# on récupère les prédictions du modèle pour ces valeurs de poids
mu <- data.frame(fitted(mod6, newdata = weight.seq) ) %>% bind_cols(weight.seq)
pred_height <- data.frame(predict(mod6, newdata = weight.seq) ) %>% bind_cols(weight.seq)

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

	Estimate	Est.Error	Q2.5	Q97.5	weight.s
1	40.38125	5.959401	28.60277	52.16339	-2.500000
2	46.88754	5.924126	35.44837	58.54749	-2.397959
3	53.26769	5.803171	42.21123	64.71465	-2.295918
4	59.24837	5.920341	47.63192	71.00610	-2.193878
5	65.01056	5.839967	53.12504	76.45687	-2.091837
6	70.75569	5.746343	59.25683	81.78785	-1.989796
7	76.34011	5.892929	64.94176	87.82885	-1.887755
8	81.65611	5.869747	70.34955	93.23042	-1.785714
9	86.87202	5.815200	75.39549	98.25968	-1.683673
10	91.77386	5.785625	80.44357	102.82518	-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  
  ) +  
  theme_bw(base_size = 20)
```



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 prédictors 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

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

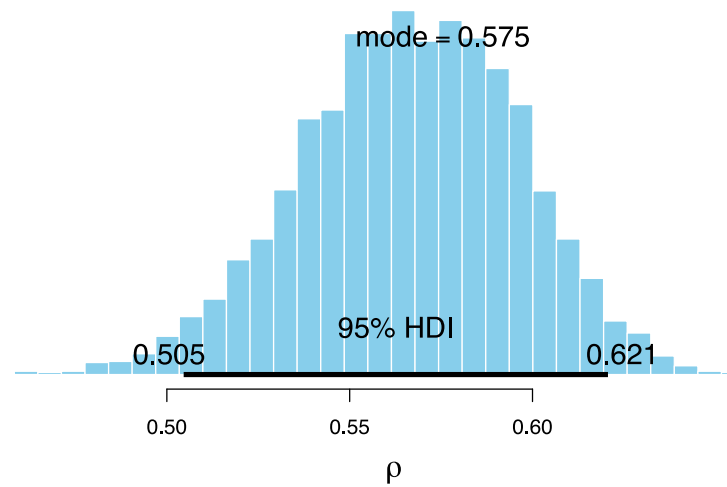
```
post <- posterior_samples(mod4)
beta <- post$b_weight
sigma <- post$sigma

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
```

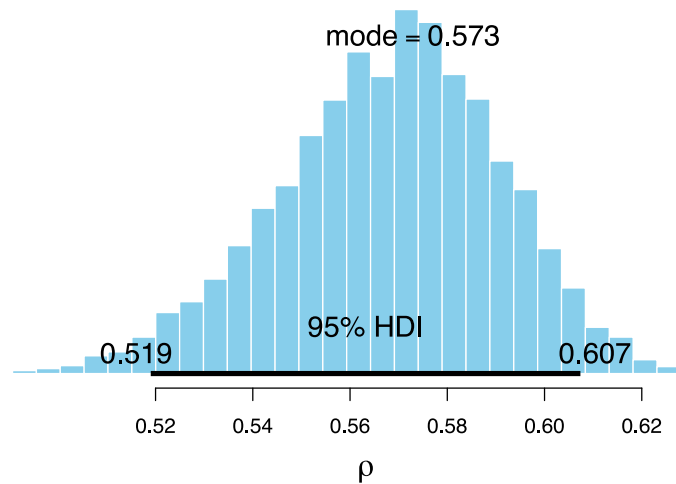
```
[1] 0.5696444
```

Modèle de régression, taille d'effet

```
bayes_R2(mod4)
```

	Estimate	Est.Error	Q2.5	Q97.5
R2	0.5679634	0.02278667	0.5206285	0.6090996

```
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 `Howell11` correspondant à des individus mineurs ($\text{age} < 18$). 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 `Howell11` 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)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
```

Population-Level Effects:

	Estimate	Est.Error	l-89% CI	u-89% CI	Rhat	Bulk_ESS	Tail_ESS
Intercept	58.23	1.39	56.00	60.48	1.00	4234	3068
weight	2.72	0.07	2.61	2.83	1.00	4154	2543

Family Specific Parameters:

	Estimate	Est.Error	l-89% CI	u-89% CI	Rhat	Bulk_ESS	Tail_ESS
sigma	8.55	0.44	7.87	9.29	1.00	4177	2547

Samples were drawn 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)

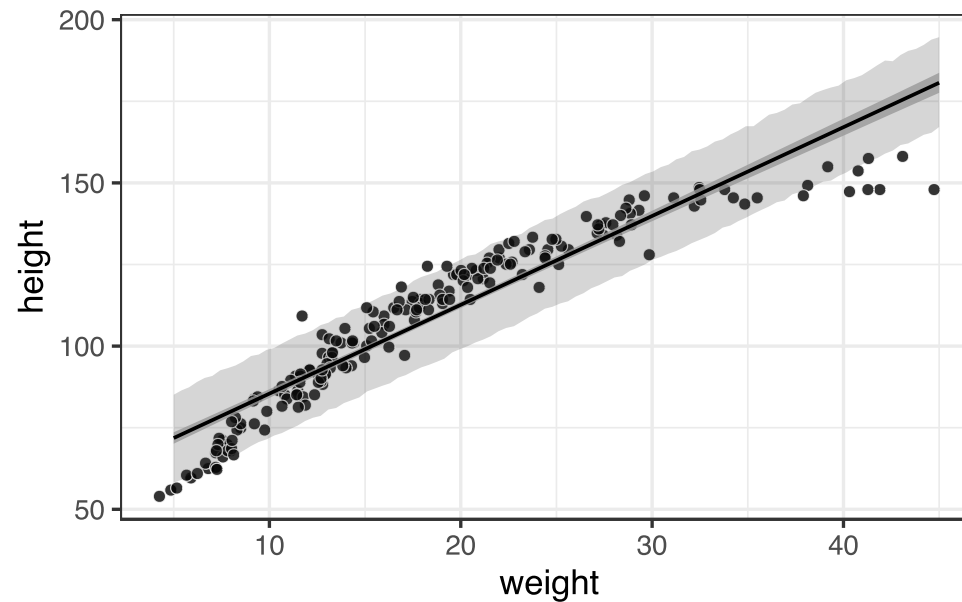
pred_height <- data.frame(
  predict(mod7, newdata = weight.seq, probs = c(0.055, 0.945) )
) %>%
  bind_cols(weight.seq)

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

	Estimate	Est.Error	Q5.5	Q94.5	weight
1	71.75561	8.535062	58.20070	85.10458	5.000000
2	73.02142	8.590257	59.01128	86.42330	5.404040
3	73.96282	8.382328	60.68347	87.40413	5.808081
4	75.10487	8.673878	60.94780	89.00061	6.212121
5	76.37712	8.762384	62.12747	90.25081	6.616162
6	77.26529	8.554469	63.74763	90.86176	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  
  ) +  
  theme_bw(base_size = 20)
```



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)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
```

Population-Level Effects:

	Estimate	Est.Error	l-89% CI	u-89% CI	Rhat	Bulk_ESS	Tail_ESS
Intercept	-23.61	1.37	-25.81	-21.40	1.00	4354	3192
logweight	47.02	0.39	46.40	47.65	1.00	4409	3217

Family Specific Parameters:

	Estimate	Est.Error	l-89% CI	u-89% CI	Rhat	Bulk_ESS	Tail_ESS
sigma	5.15	0.15	4.92	5.41	1.00	3972	2674

Samples were drawn 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)

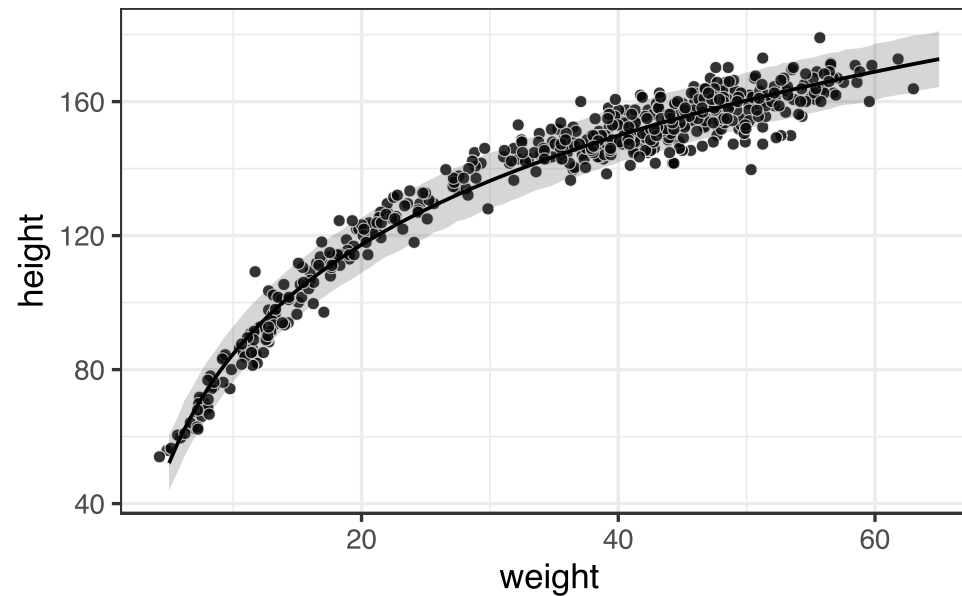
pred_height <- data.frame(
  predict(mod8, newdata = weight.seq, probs = c(0.055, 0.945) )
) %>%
  bind_cols(weight.seq)

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

	Estimate	Est.Error	Q5.5	Q94.5	weight
1	52.23800	5.265516	43.96558	60.61920	5.000000
2	57.34252	5.175260	48.88584	65.17004	5.606061
3	62.32088	5.197434	54.15168	70.79880	6.212121
4	66.70109	5.088207	58.56606	74.75198	6.818182
5	70.64608	5.280222	62.08034	78.96780	7.424242
6	74.40413	5.198329	66.21465	82.77536	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  
  ) +  
  theme_bw(base_size = 20)
```



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  
  ) +  
  theme_bw(base_size = 20)
```

