

# Introduction à la modélisation statistique bayésienne

Un cours en R, Stan, et brms

Ladislav 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

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

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

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

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

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

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



# Un premier modèle

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

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

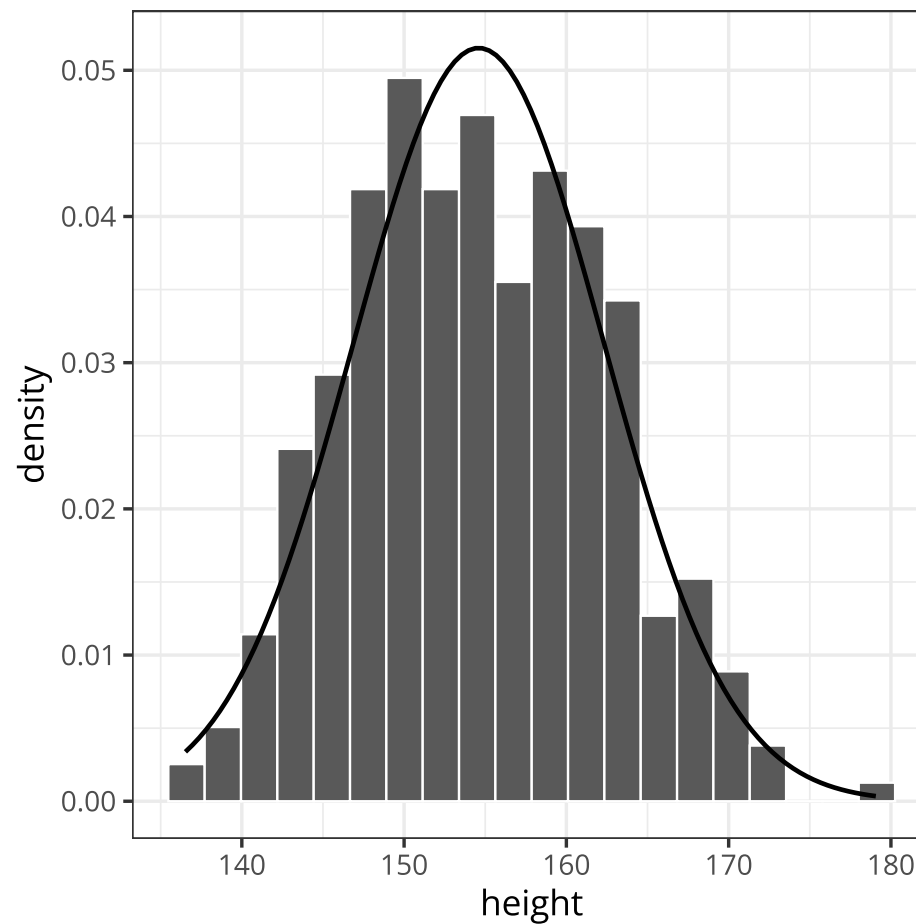
```
1 d2 <- d %>% filter(age >= 18)
2 head(d2)
```

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

# Un premier modèle

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

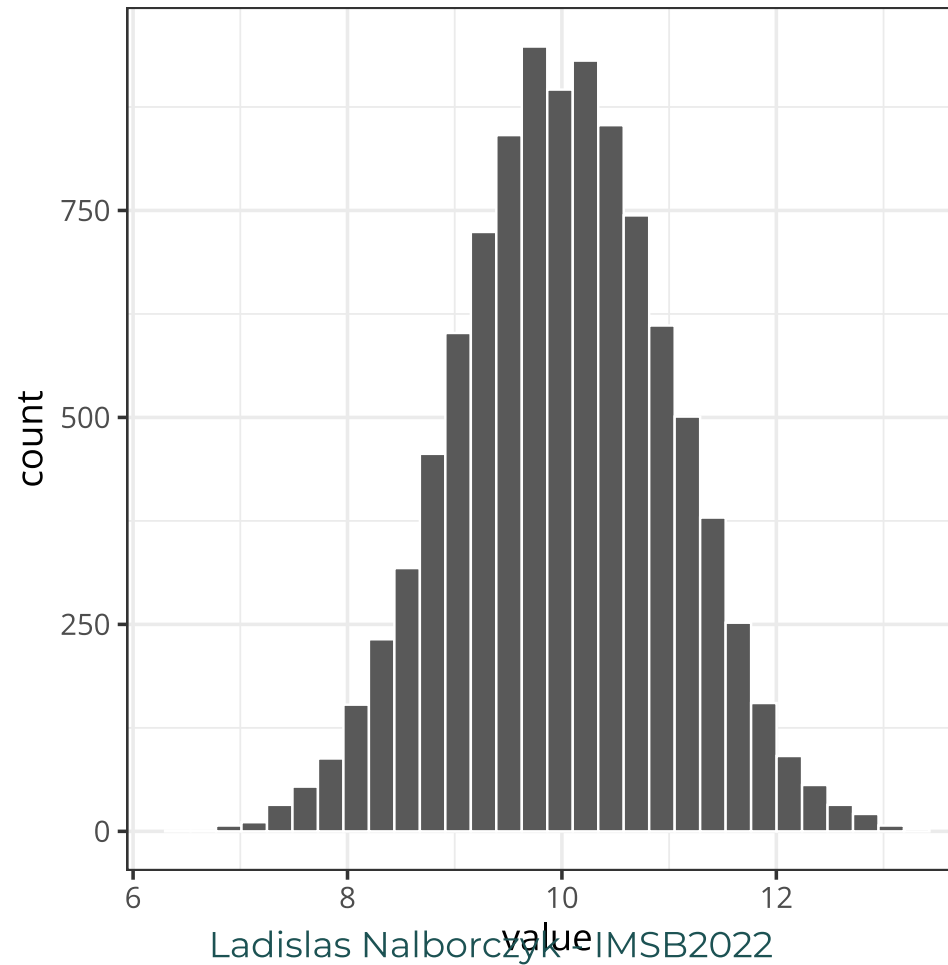
```
1 d2 %>%  
2   ggplot(aes(x = height) ) +  
3   geom_histogram(aes(y = ..density..), bins = 20, col = "white") +  
4   stat_function(fun = dnorm, args = list(mean(d2$height), sd(d2$height) ), size = 1)
```



# Loi normale

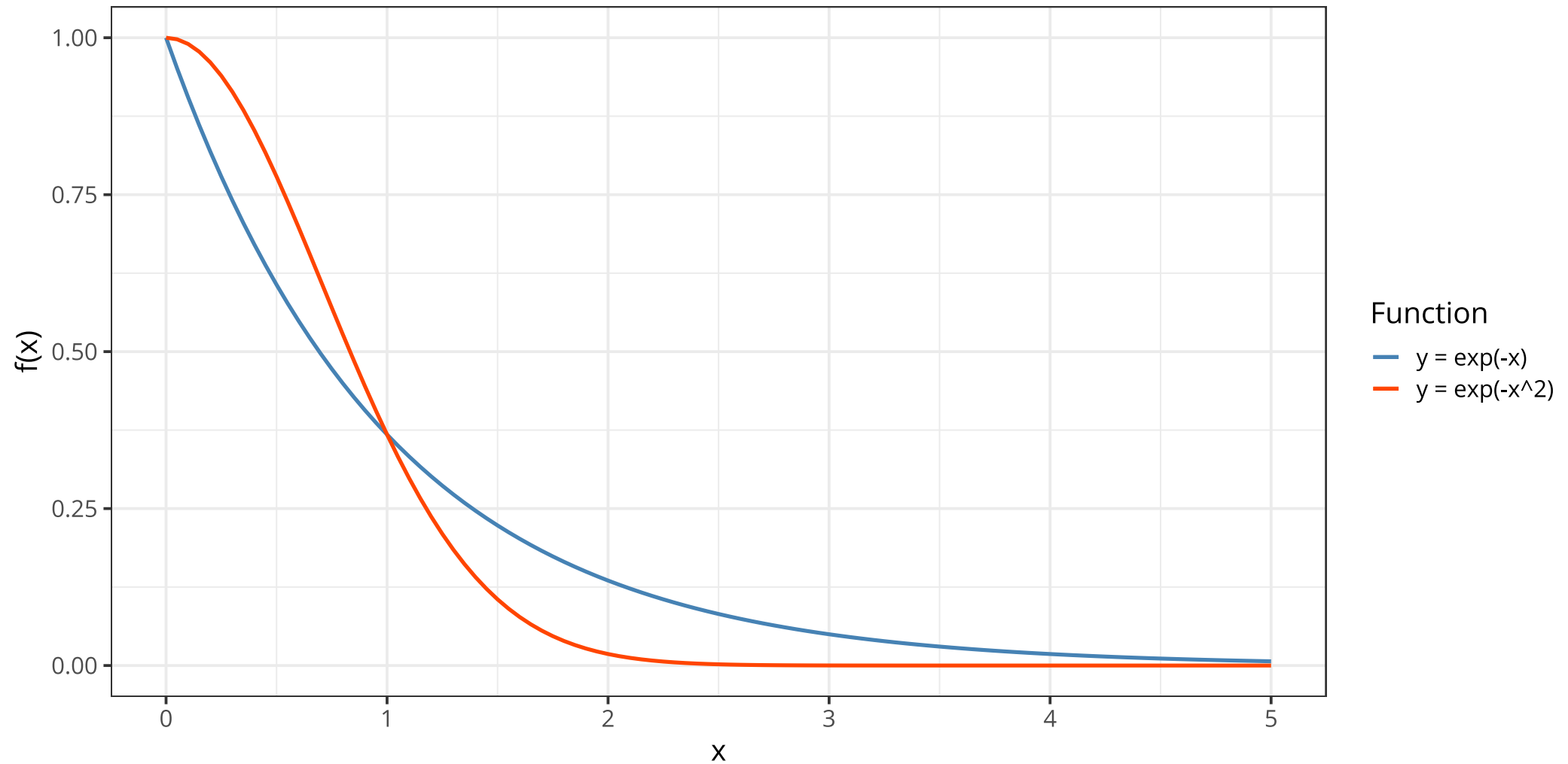
$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ -\frac{1}{2\sigma^2} (\mu - x)^2 \right]$$

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



# D'où vient la loi normale ?

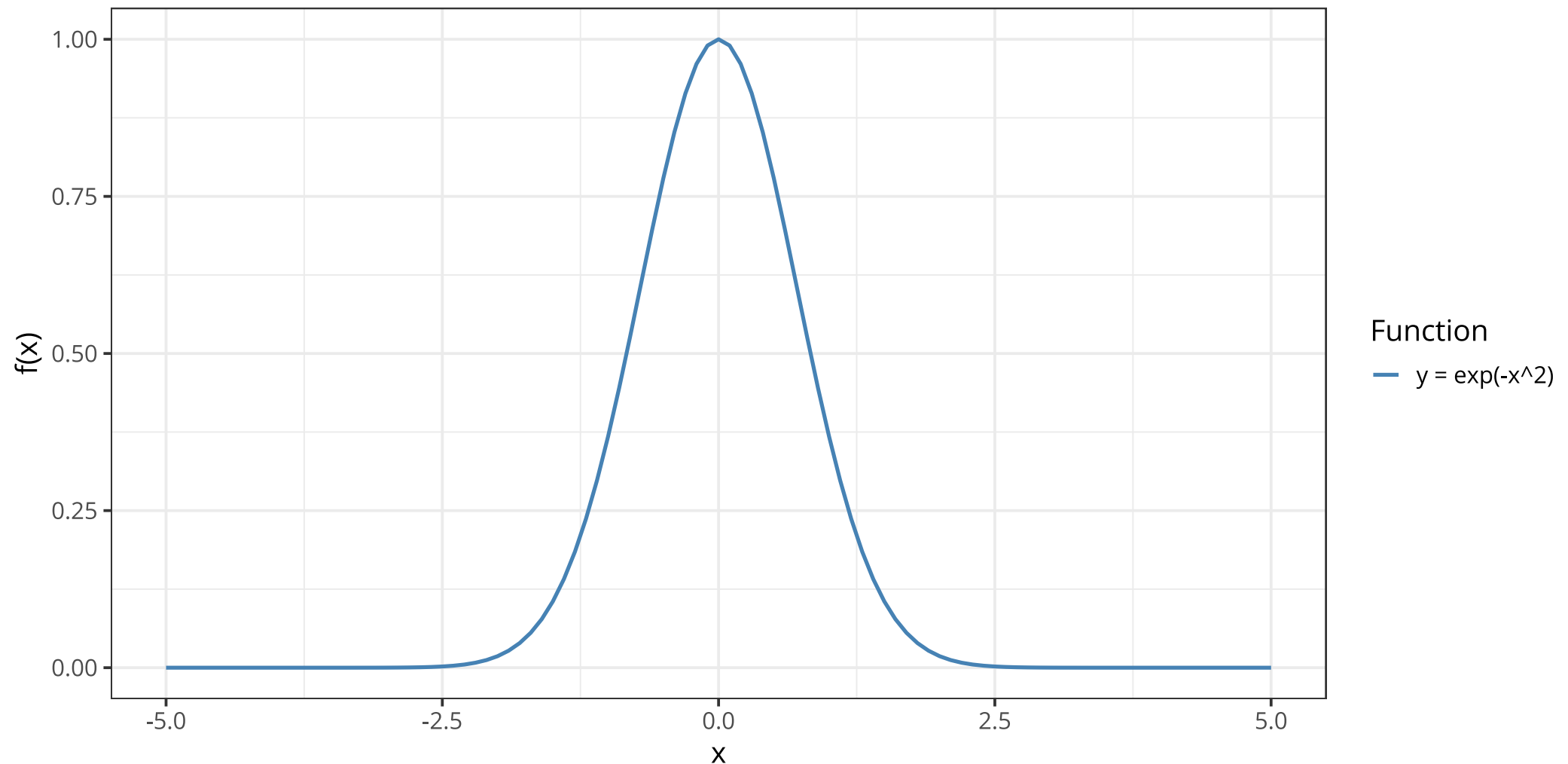
Contraintes : Certaines valeurs soient fortement probables (autour de la moyenne  $\mu$ ). Plus on s'éloigne, moins les valeurs sont probables (en suivant une décroissance exponentielle).



# D'où vient la loi normale ?

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

On étend notre fonction aux valeurs négatives.

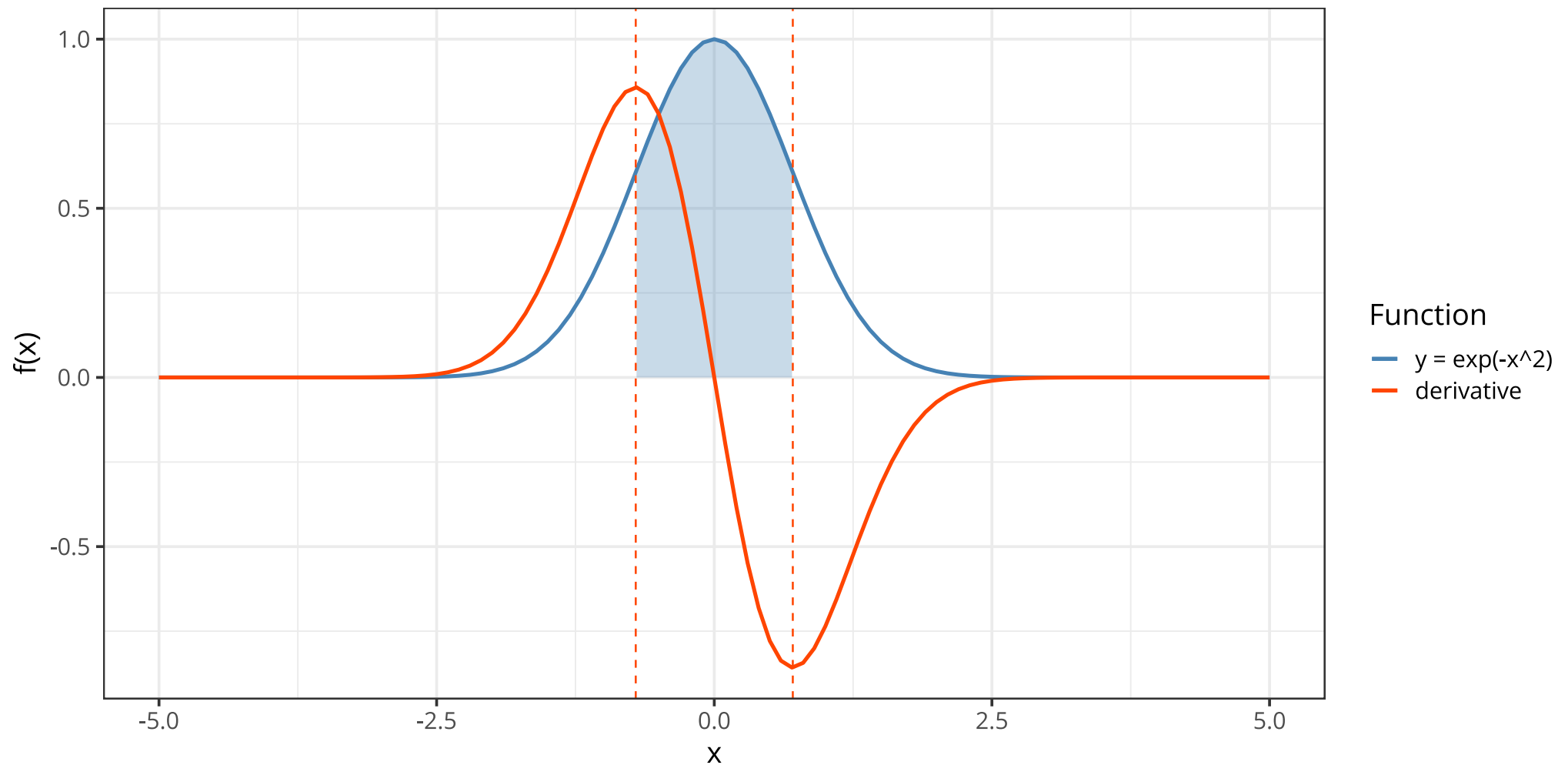




# D'où vient la loi normale ?

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

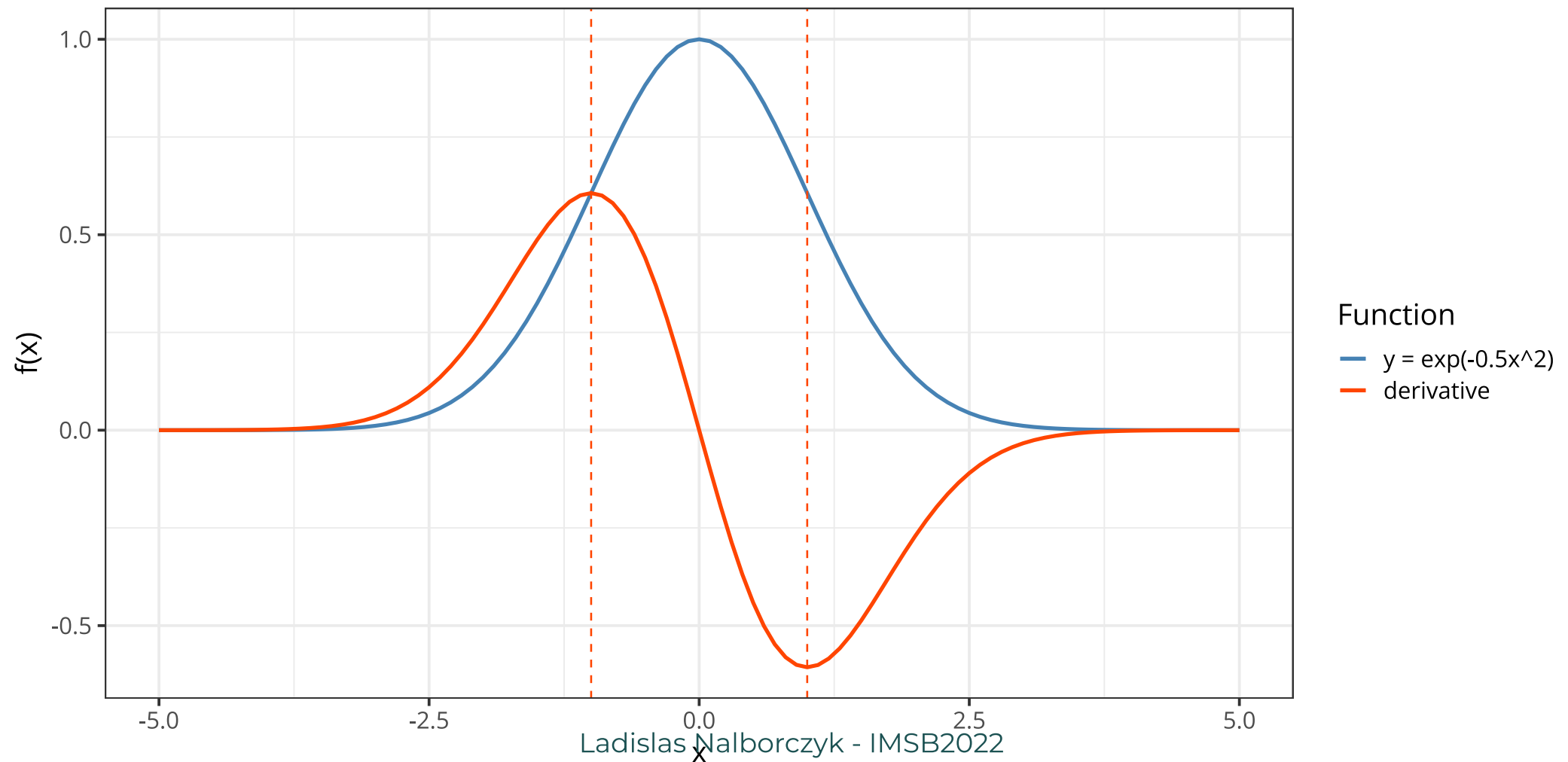
Les points d'inflexion nous donnent une bonne indication de là où la plupart des valeurs se trouvent (i.e., entre les points d'inflexion). Les pics de la dérivée nous montrent les points d'inflexion.



# D'où vient la loi normale ?

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

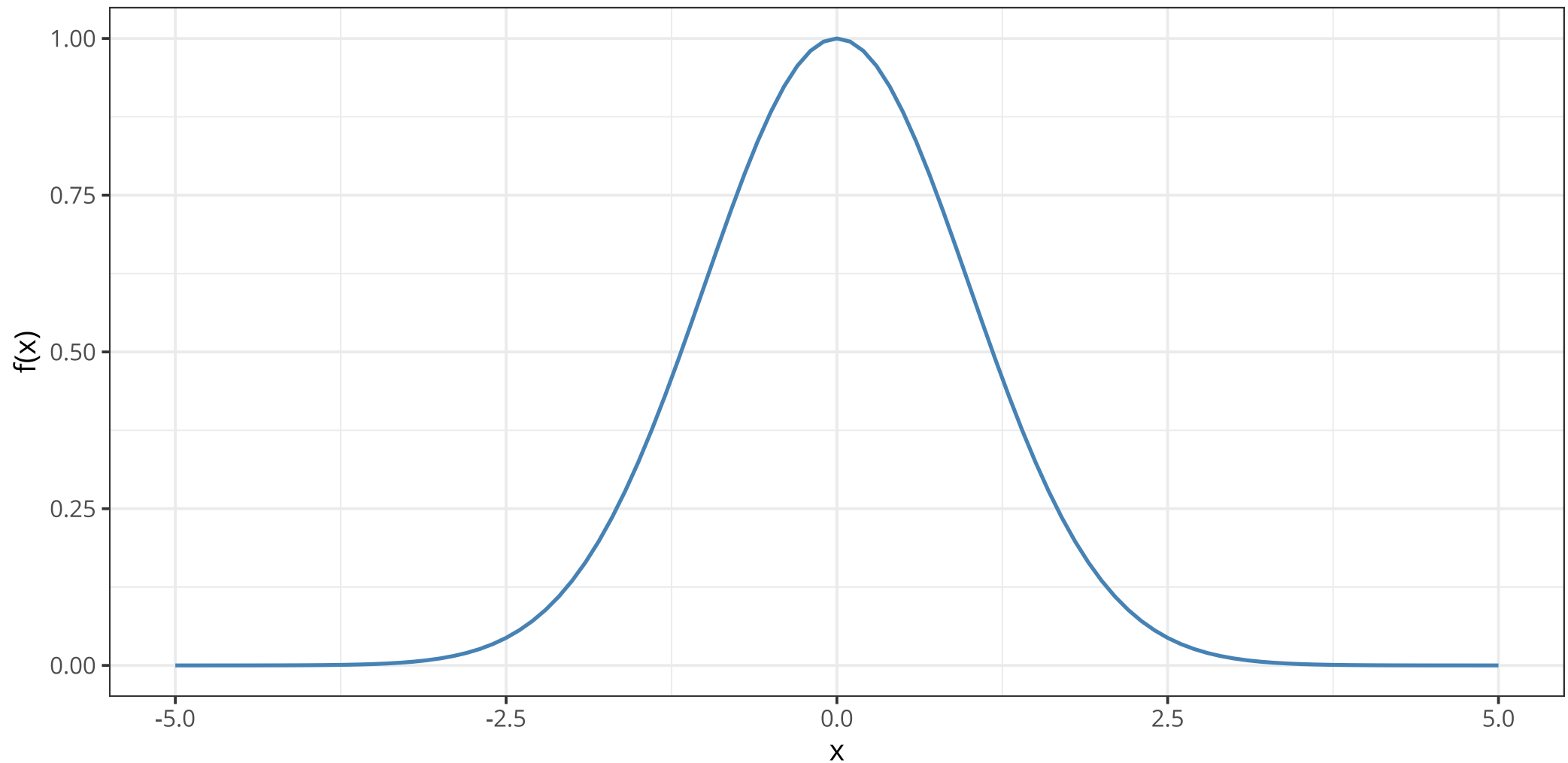
Ensuite on standardise la distribution de manière à ce que les deux points d'inflexion se trouvent à  $x = -1$  et  $x = 1$ .



# D'où vient la loi normale ?

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

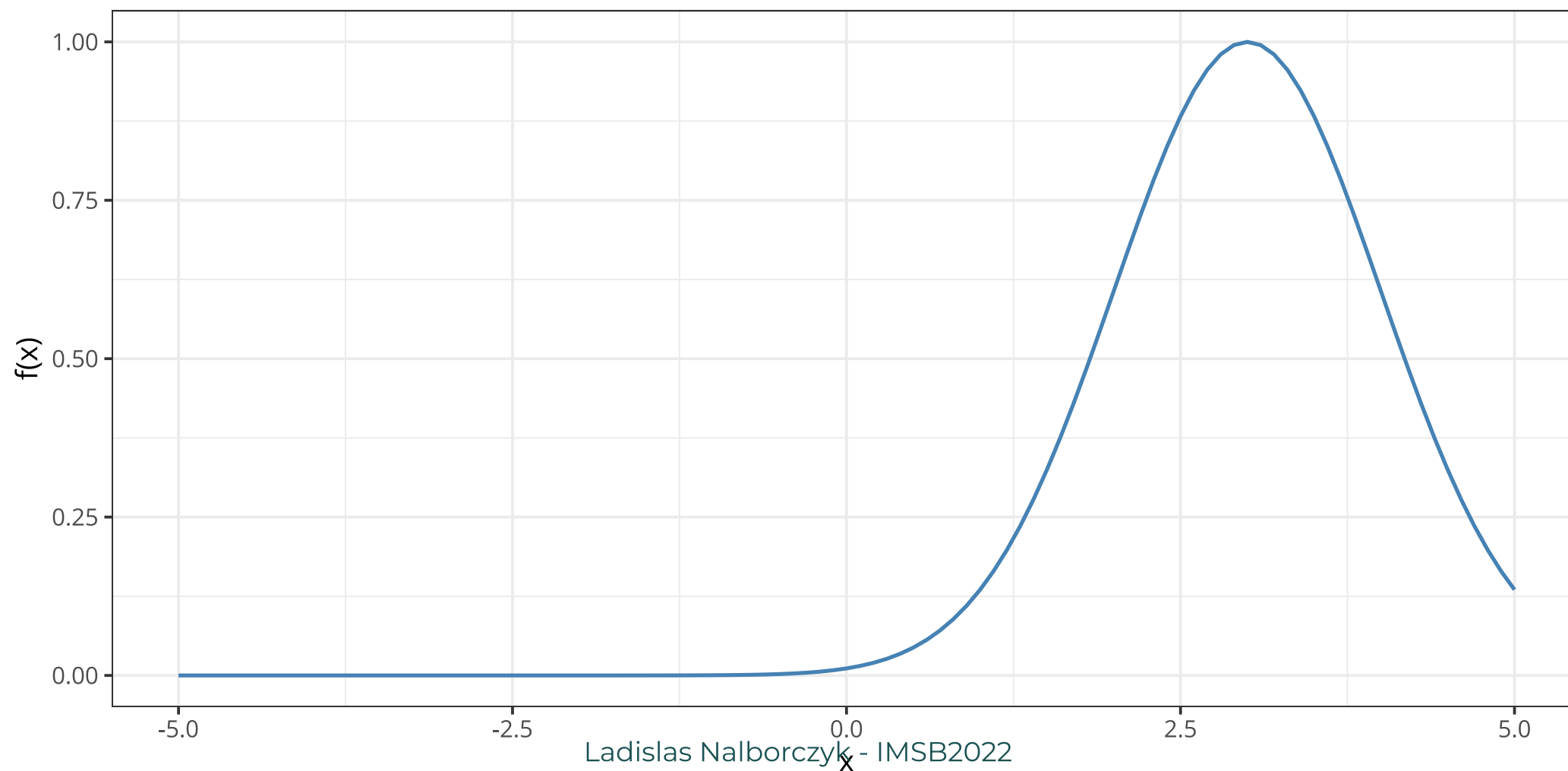
On insère un paramètre  $\sigma^2$  pour contrôler la distance entre les points d'inflexion.



# D'où vient la loi normale ?

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

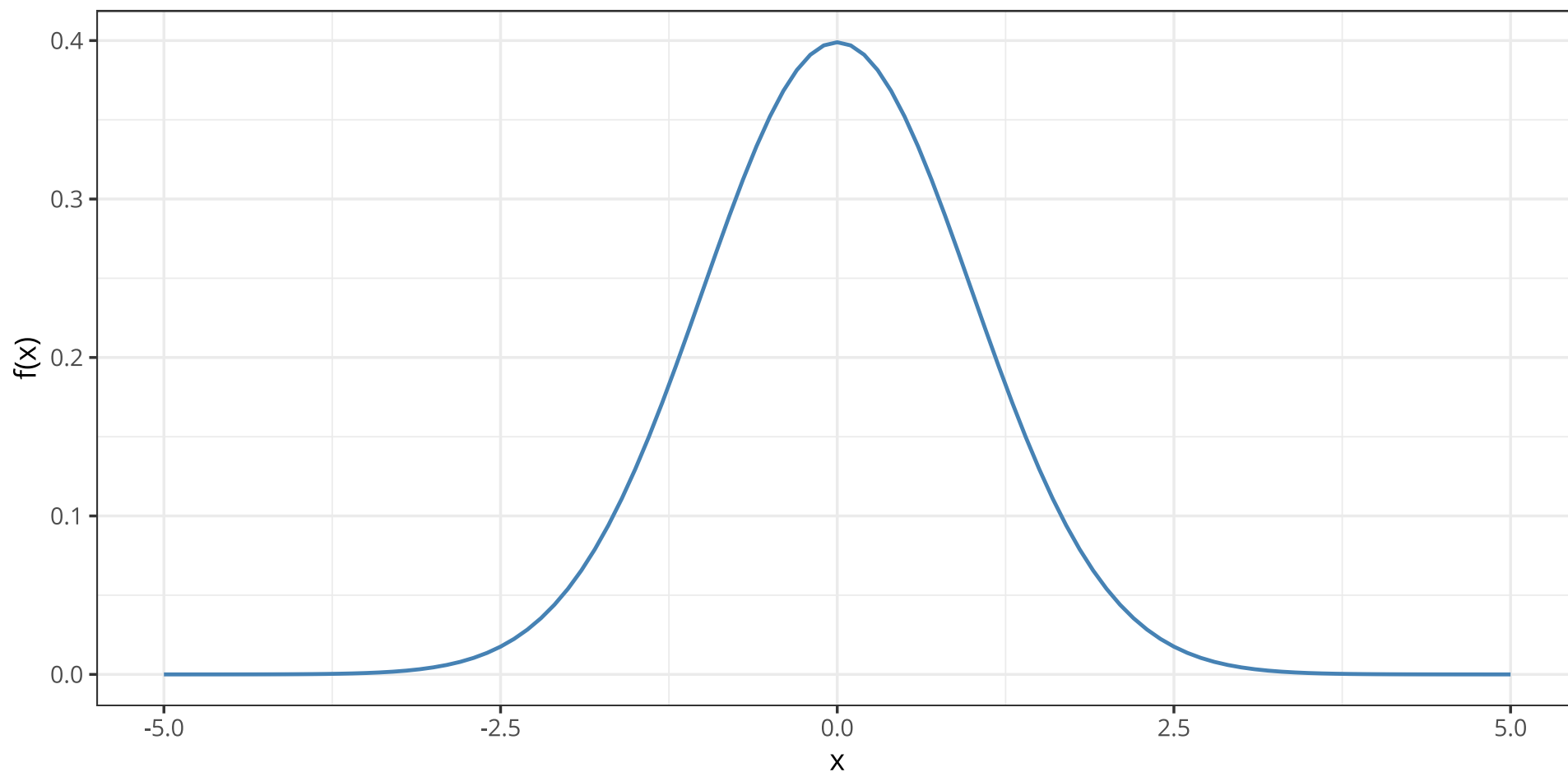
On insère ensuite un paramètre  $\mu$  afin de pouvoir contrôler la position (la tendance centrale) de la distribution.



# D'où vient la loi normale ?

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

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



# Modèle gaussien

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

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

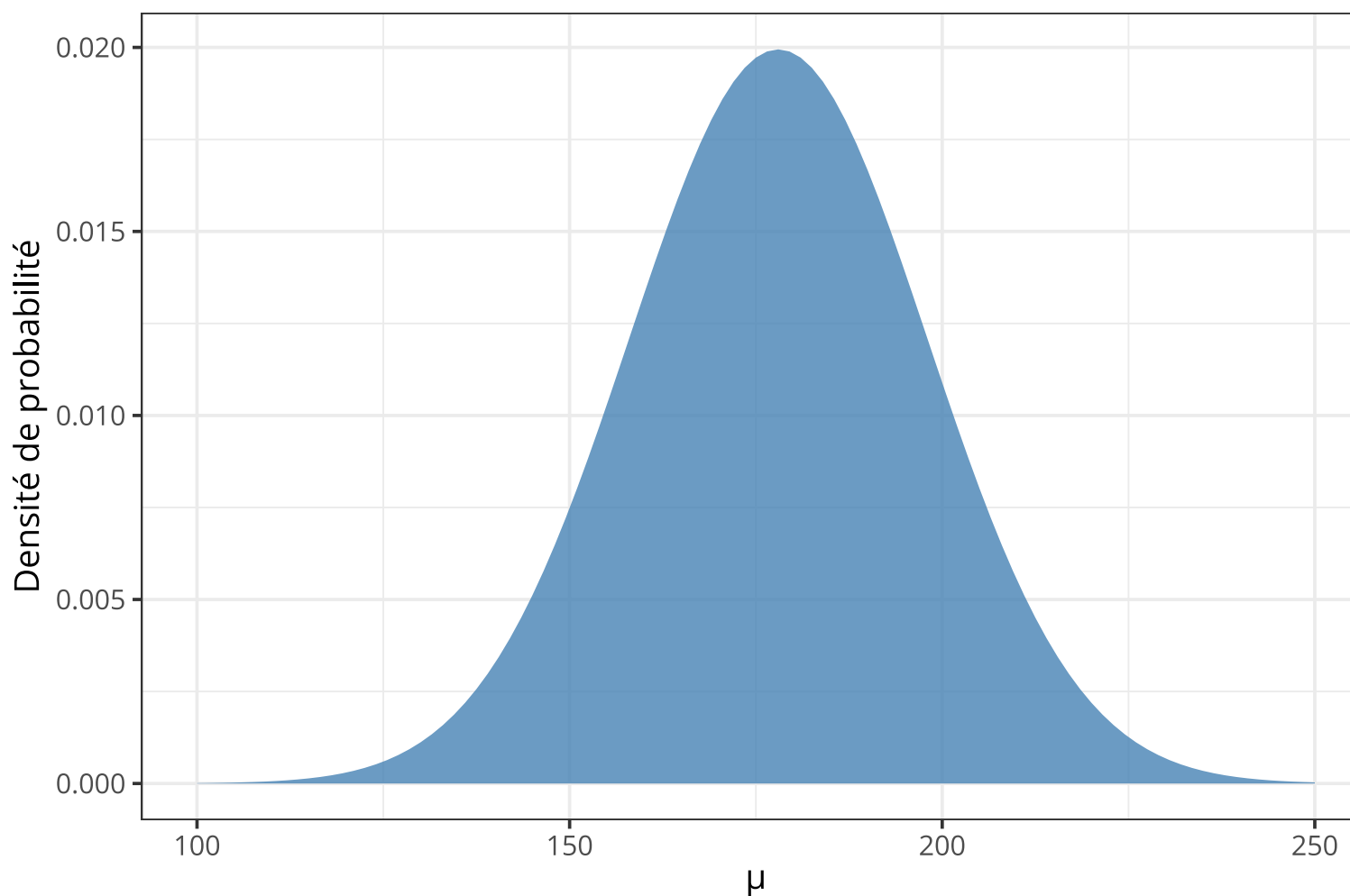
Notre but, une fois encore, est de décrire **la distribution postérieure**, qui sera donc d'une certaine manière **une distribution de distributions**.



# Modèle gaussien

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

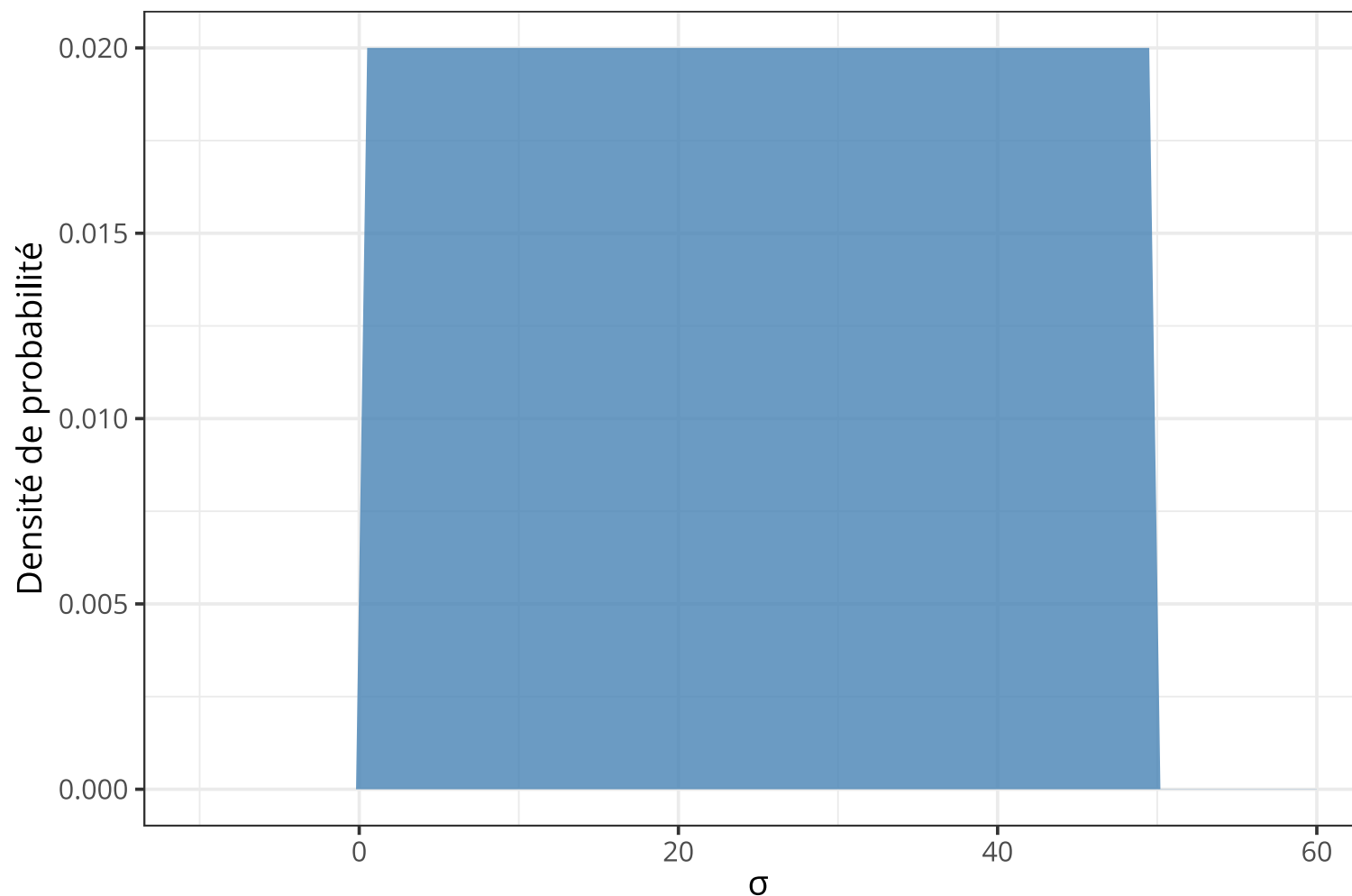
$$\mu \sim \text{Normal}(178, 20)$$



# Modèle gaussien

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

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



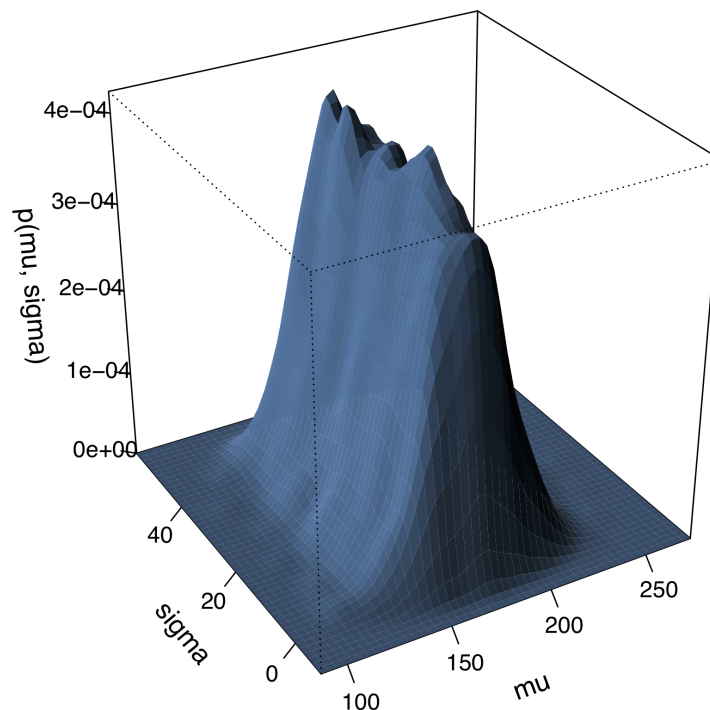


# Visualiser le prior

```

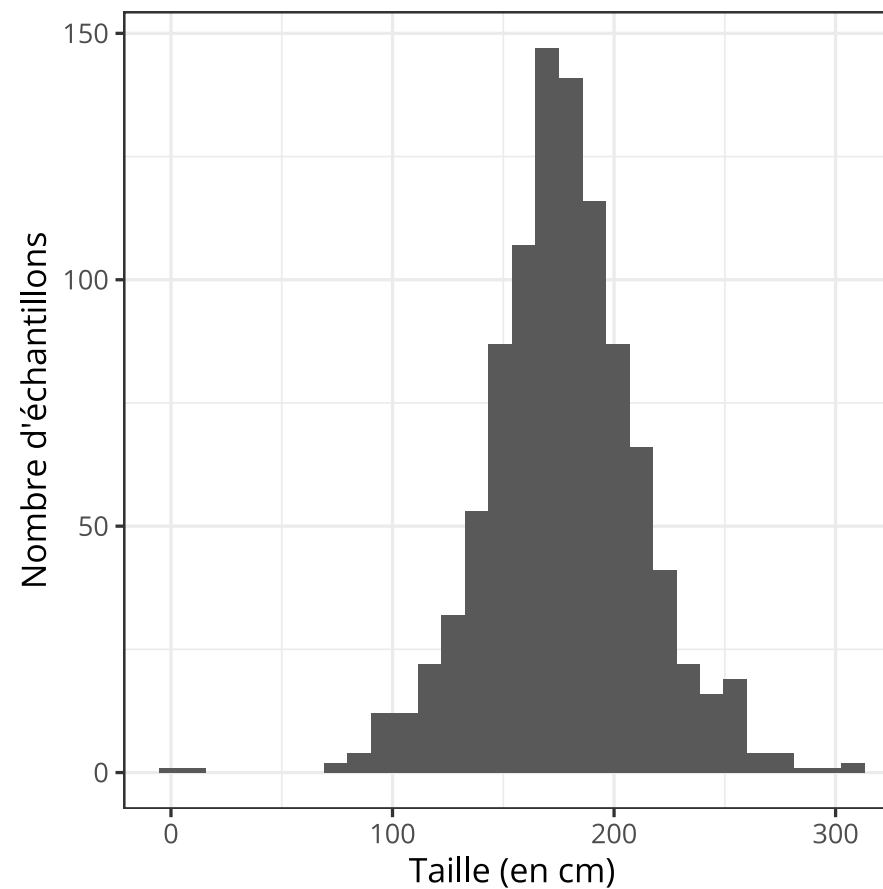
1 library(ks)
2 sample_mu <- rnorm(1e4, 178, 20) # prior on mu
3 sample_sigma <- runif(1e4, 0, 50) # prior on sigma
4 prior <- data.frame(cbind(sample_mu, sample_sigma) ) # multivariate prior
5 H.scv <- Hscv(x = prior, verbose = TRUE)
6 fhat_prior <- kde(x = prior, H = H.scv, compute.cont = TRUE)
7 plot(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

```



# Prior predictive checking

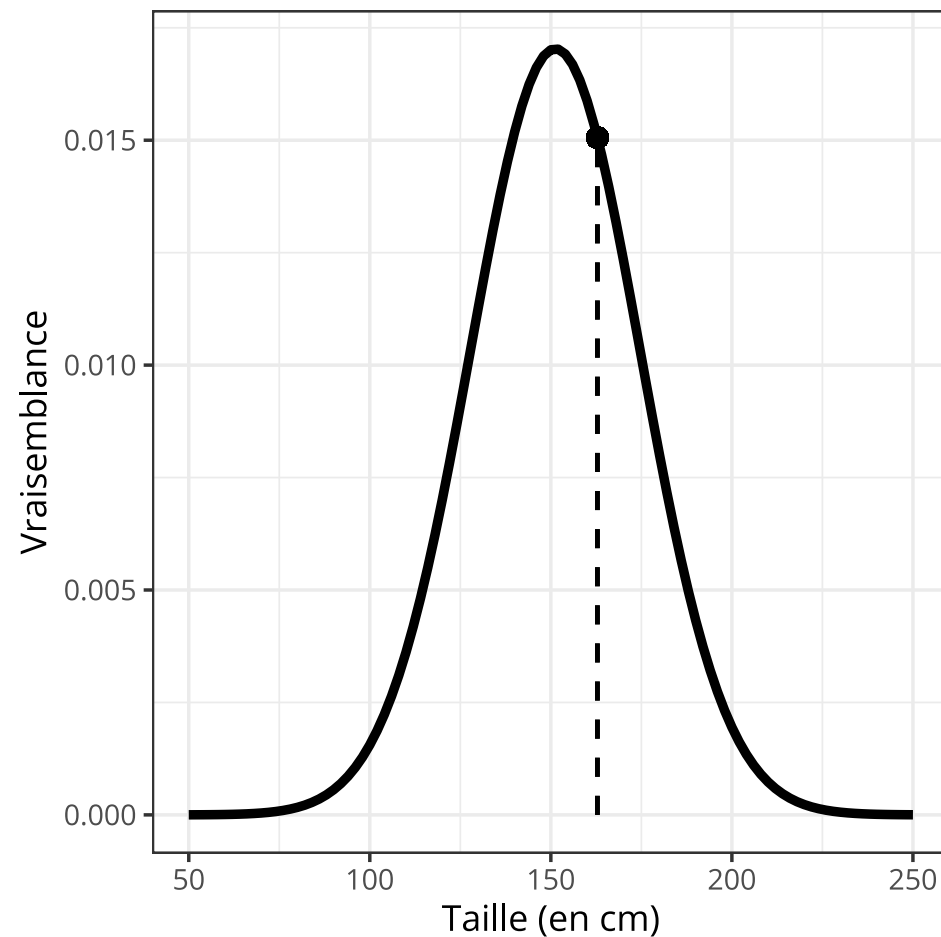
```
1 sample_mu <- rnorm(1000, 178, 20)
2 sample_sigma <- runif(1000, 0, 50)
3
4 data.frame(x = rnorm(1000, sample_mu, sample_sigma) ) %>%
5   ggplot(aes(x) ) +
6   geom_histogram() +
7   labs(x = "Taille (en cm)", y = "Nombre d'échantillons")
```



# Fonction de vraisemblance

```
1 mu_exemple <- 151.23
2 sigma_exemple <- 23.42
3
4 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  $\mu$  et  $\sigma$ , c'est à dire :

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

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

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

```
[1] 0.01505675
```



# Fonction de vraisemblance

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

Ou avec une fonction maison...

```
1 normal_likelihood <- function (x, mu, sigma) {  
2  
3   bell <- exp( (- 1 / (2 * sigma^2) ) * (mu - x)^2 )  
4   norm <- sqrt(2 * pi * sigma^2)  
5  
6   return(bell / norm)  
7  
8 }
```

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

```
[1] 0.01505675
```



# 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 :  $\mu$  et  $\sigma$ .

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



# Distribution postérieure - Grid approximation

```
1 # on définit une grille de valeurs possibles pour mu et sigma
2 mu.list <- seq(from = 140, to = 160, length.out = 200)
3 sigma.list <- seq(from = 4, to = 9, length.out = 200)
4
5 # on étend la grille à toutes les combinaisons possibles de mu et sigma
6 post <- expand.grid(mu = mu.list, sigma = sigma.list)
7
8 # calcul de la log-vraisemblance (pour chaque combinaison de mu et sigma)
9 post$LL <-
10   sapply(
11     1:nrow(post),
12     function(i) sum(dnorm(
13       d2$height,
14       mean = post$mu[i],
15       sd = post$sigma[i],
16       log = TRUE) )
17   )
18
19 # calcul de la probabilité a posteriori (non normalisée)
20 post$prod <-
21   post$LL +
22   dnorm(x = post$mu, mean = 178, sd = 20, log = TRUE) +
23   dunif(x = post$sigma, min = 0, max = 50, log = TRUE)
24
25 # on "annule" le log et on standardise par la valeur maximale (pour éviter les erreurs d'arrondi)
```



# Distribution postérieure - Grid approximation

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

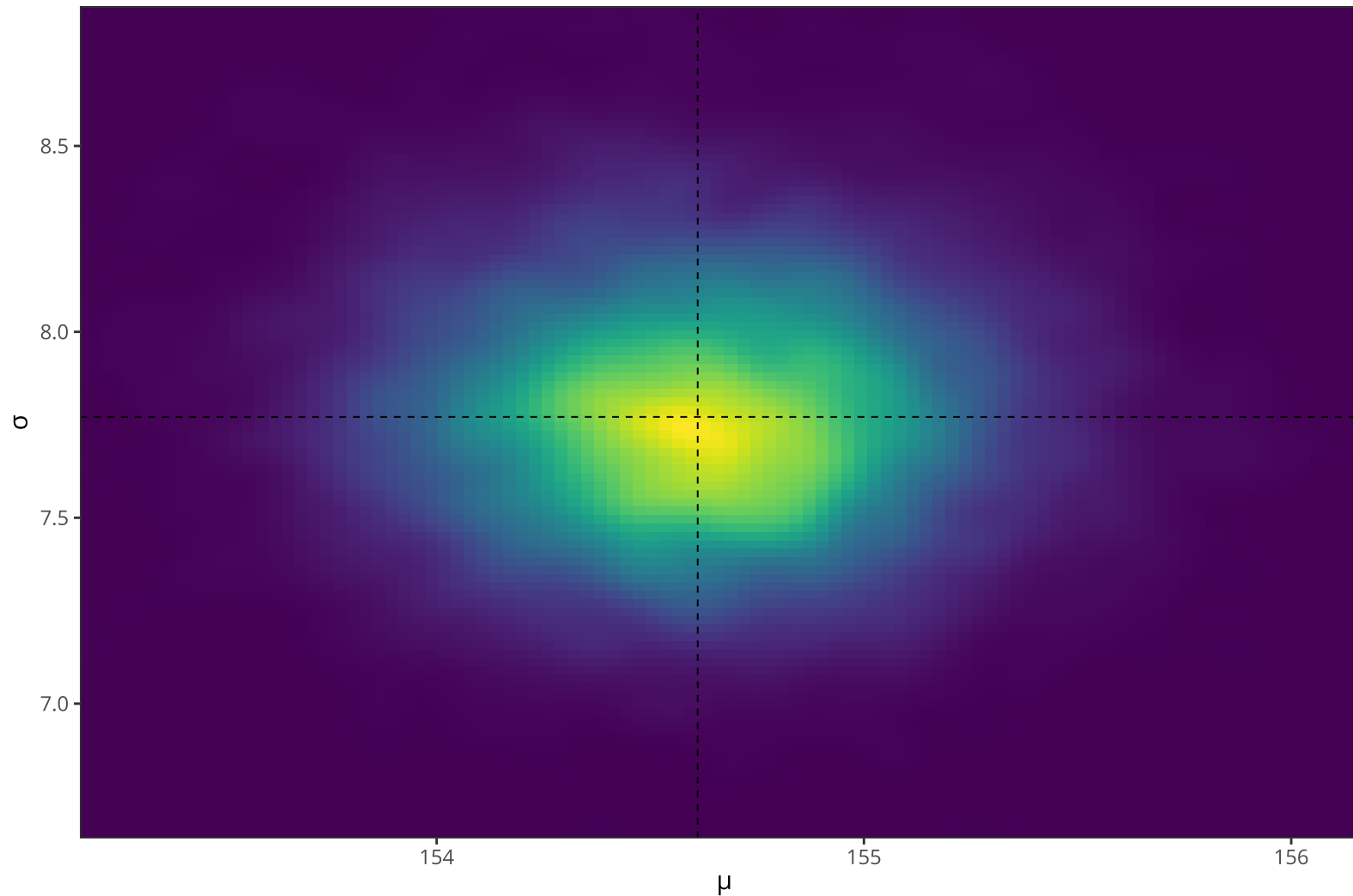
	mu	sigma	LL	prod	prob
1	140.4020	8.949749	-1669.018	-1678.611	1.851262e-196
2	152.3618	8.824121	-1236.343	-1244.991	3.856066e-08
3	155.2764	7.065327	-1224.068	-1232.540	9.856972e-03
4	157.4874	6.939698	-1254.355	-1262.708	7.798220e-16
5	157.7889	6.386935	-1278.010	-1286.347	4.221437e-26
6	155.4774	5.055276	-1310.848	-1319.309	2.044278e-40
7	149.4472	4.025126	-1751.071	-1759.917	9.054090e-232
8	155.3769	6.185930	-1242.628	-1251.094	8.626726e-11
9	148.0402	4.075377	-1907.013	-1915.961	1.540404e-299
10	151.0553	8.698492	-1253.112	-1261.846	1.845524e-15
11	147.8392	5.909548	-1480.217	-1489.181	3.435830e-114
12	152.4623	8.095477	-1232.367	-1241.009	2.068862e-06
13	142.7136	4.075377	-2947.893	-2957.276	0.000000e+00
14	156.7839	8.221106	-1233.134	-1241.523	1.237383e-06
15	152.5628	8.045226	-1231.201	-1239.836	6.683351e-06
16	149.1457	4.075377	-1766.335	-1775.202	2.082933e-238
17	141.8090	6.236181	-1978.355	-1987.818	0.000000e+00
18	159.5980	4.603015	-1565.130	-1573.380	9.310976e-151
19	148.6432	6.939698	-1353.373	-1362.277	4.463307e-59
20	156.4824	5.005025	-1335.275	-1343.681	5.320236e-51





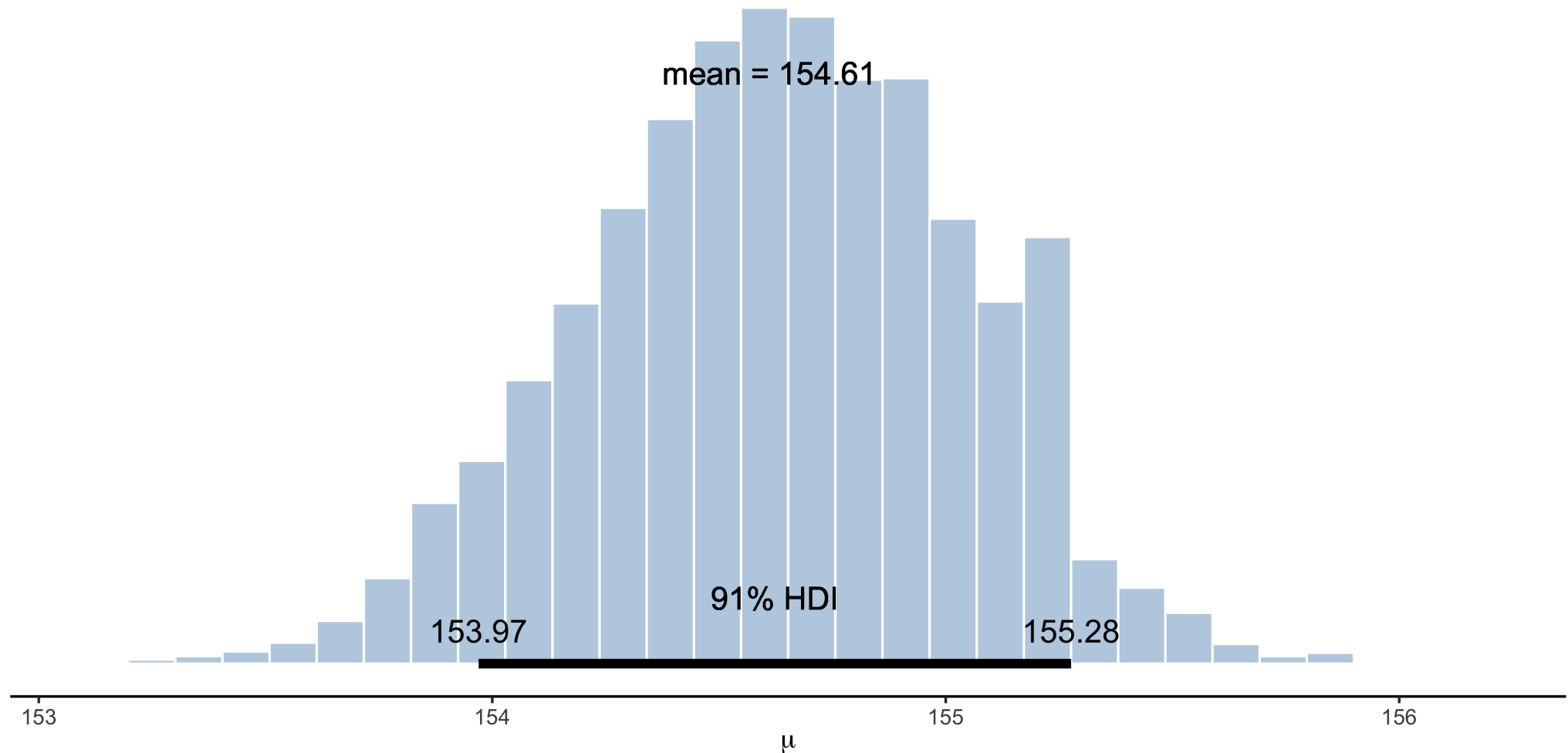
# Distribution postérieure - Grid approximation

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



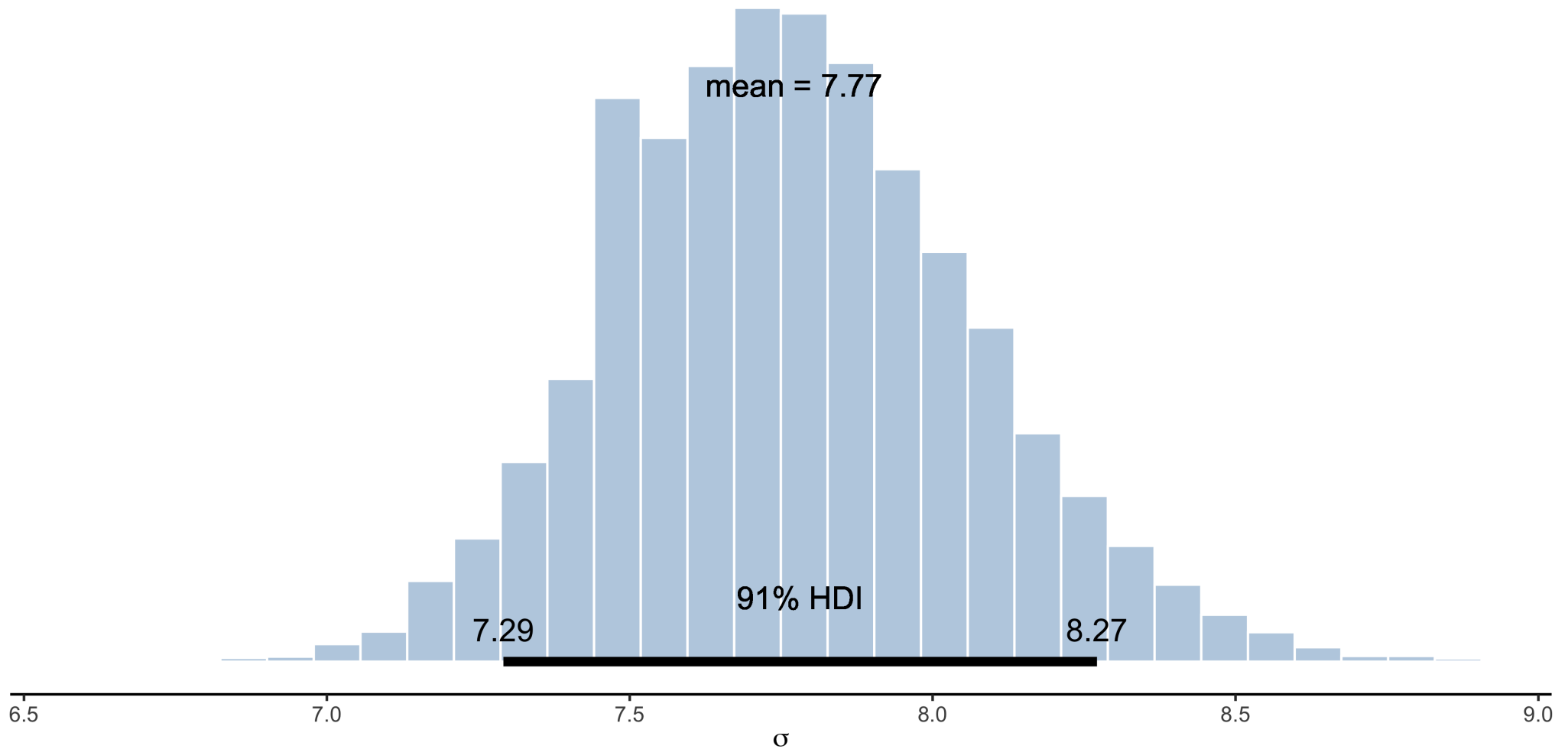
# Distribution postérieure - Distributions marginales

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



# Distribution postérieure - Distributions marginales

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



# Introduction à brms

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

```
1 data {
2   int<lower=0> J; // number of schools
3   real y[J]; // estimated treatment effects
4   real<lower=0> sigma[J]; // s.e. of effect estimates
5 }
6
7 parameters {
8   real mu;
9   real<lower=0> tau;
10  real eta[J];
11 }
12
13 transformed parameters {
14   real theta[J];
15   for (j in 1:J)
16     theta[j] = mu + tau * eta[j];
17 }
18
19 model {
20   target += normal_lpdf(eta | 0, 1);
21   target += normal_lpdf(y | theta, sigma);
22 }
```



# 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 :

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



# Rappels de syntaxe

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

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

La partie gauche représente notre variable dépendante (ou outcome, i.e., ce qu'on essaye de prédire). Le package `brms` permet également de fitter des modèles multivariés (plusieurs outcomes) en les combinant avec `mvbind()`.

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

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

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



# Rappels de syntaxe

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

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

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

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

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



# Quelques fonctions utiles

```
1 # générer le code du modèle en Stan
2 make_stancode(formula, ...)
3 stancode(fit)
4
5 # définir les priors
6 get_prior(formula, ...)
7 set_prior(prior, ...)
8
9 # récupérer les prédictions du modèle
10 fitted(fit, ...)
11 predict(fit, ...)
12 conditional_effects(fit, ...)
13
14 # posterior predictive checking
15 pp_check(fit, ...)
16
17 # comparaison de modèles
18 loo(fit1, fit2, ...)
19 bayes_factor(fit1, fit2, ...)
20 model_weights(fit1, fit2, ...)
21
22 # test d'hypothèse
23 hypothesis(fit, hypothesis, ...)
```





# Un premier exemple

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

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

	Estimate	Est.Error	Q2.5	Q97.5
b_Intercept	154.591780	0.4212333	153.753327	155.420404
sigma	7.762537	0.2969947	7.205733	8.353429

Ces données représentent les distributions marginales de chaque paramètre. En d'autres termes, la probabilité de chaque valeur de  $\mu$ , moyennée sur toutes les valeurs possible de  $\sigma$ , est décrite par une distribution gaussienne avec une moyenne de 154.6 et un écart type de 0.42. L'intervalle de crédibilité ( $\neq$  intervalle de confiance) nous indique les 95% valeurs de  $\mu$  ou  $\sigma$  les plus probables (sachant les données et les priors).



# En utilisant notre prior

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

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

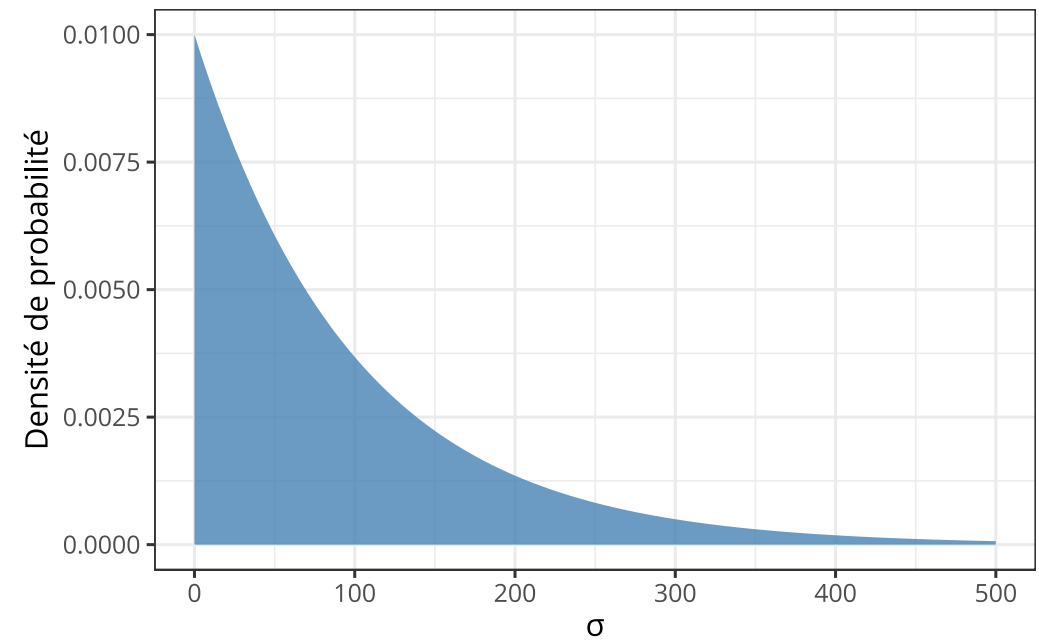
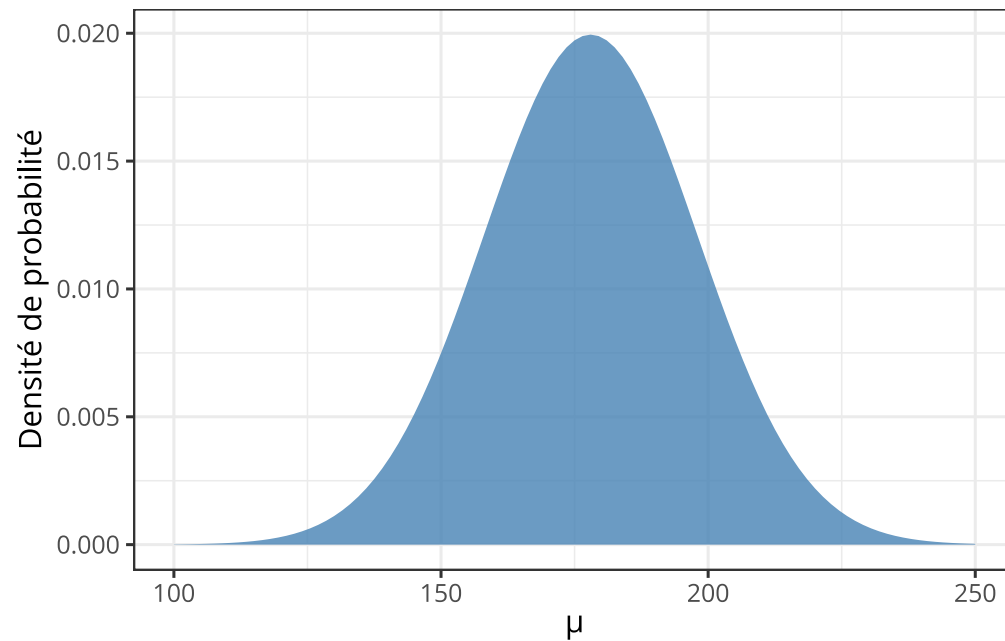
```
1 get_prior(height ~ 1, data = d2)
```

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



# En utilisant notre prior

```
1 priors <- c(  
2   prior(normal(178, 20), class = Intercept),  
3   prior(exponential(0.01), class = sigma)  
4 )  
5  
6 mod2 <- brm(  
7   height ~ 1,  
8   prior = priors,  
9   family = gaussian(),  
10  data = d2  
11 )
```



# En utilisant notre prior

```
1 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   154.62     0.42   153.79   155.43 1.00     3339     2429

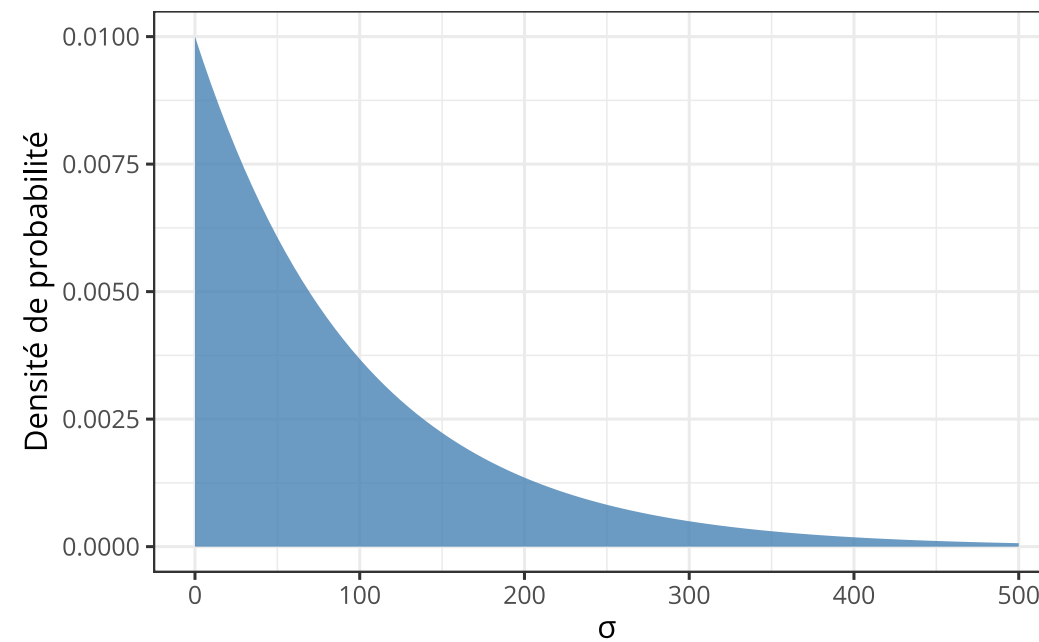
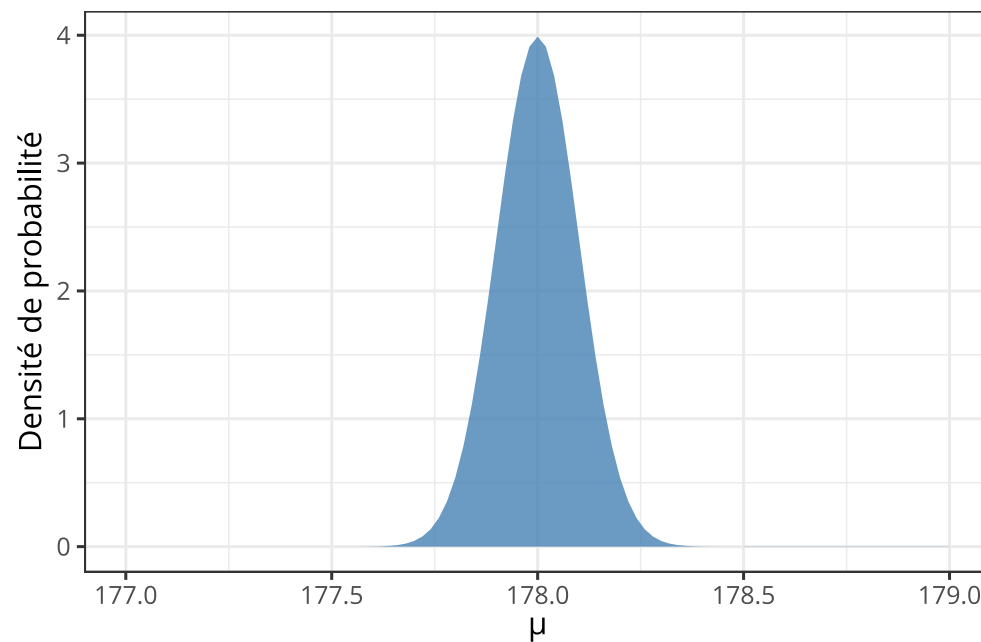
Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma       7.77     0.30     7.19     8.37 1.00     3810     2544

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

```
1 priors <- c(  
2   prior(normal(178, 0.1), class = Intercept),  
3   prior(exponential(0.01), class = sigma)  
4 )  
5  
6 mod3 <- brm(  
7   height ~ 1,  
8   prior = priors,  
9   family = gaussian(),  
10  data = d2  
11 )
```



# En utilisant un prior plus informatif

```
1 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    177.87      0.10   177.67   178.06 1.00    4270    2641

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     24.64      0.92    22.88    26.51 1.00    3696    2687

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



# Précision du prior (heuristique)

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

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

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

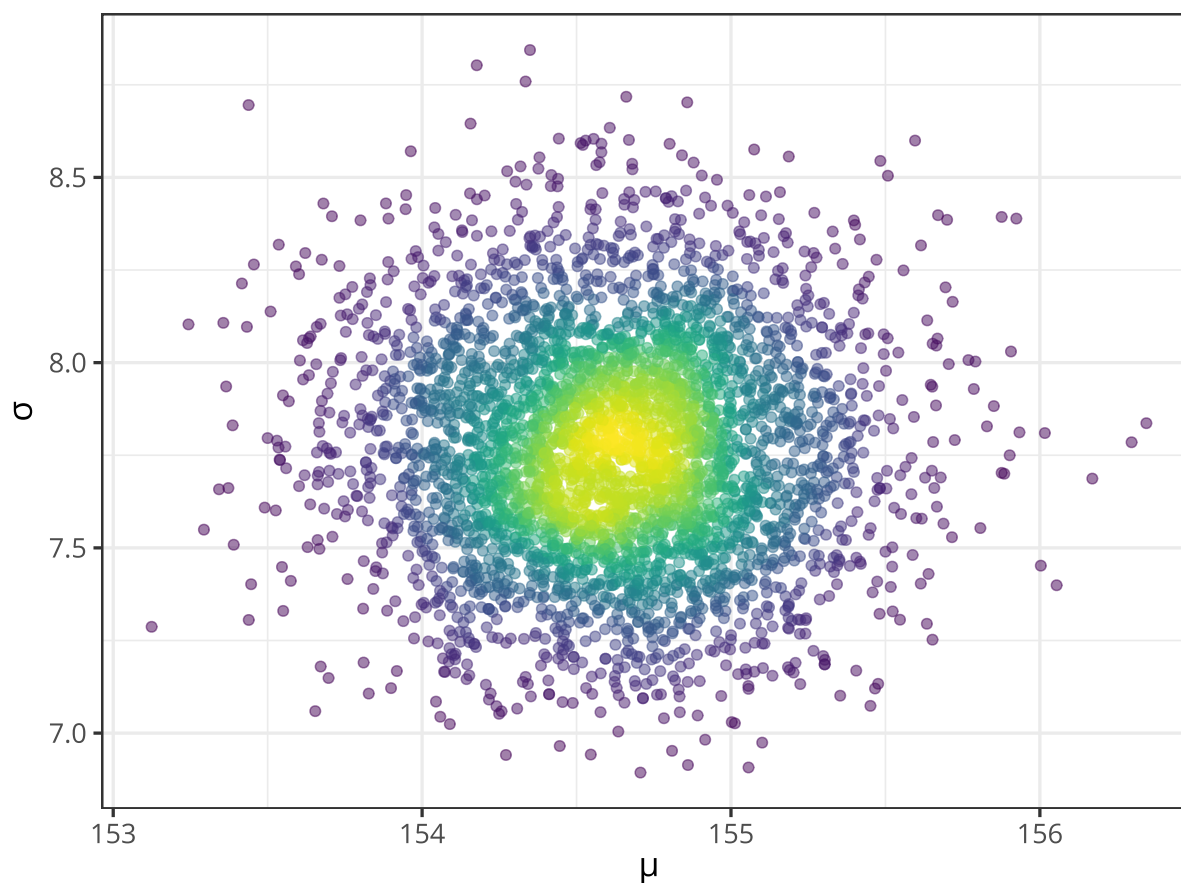
Qui implique une **quantité de données**  $n = 1 / \sigma_{\text{post}}^2$ . Notre prior avait un  $\sigma = 0.1$ , ce qui donne  $n = 1 / 0.1^2 = 100$ .

On peut donc 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.



# Visualiser les échantillons de la distribution postérieure

```
1 post <- as_draws_df(x = mod2) %>%  
2   mutate(density = get_density(b_Intercept, sigma, n = 1e2) )  
3  
4 ggplot(post, aes(x = b_Intercept, y = sigma, color = density) ) +  
5   geom_point(size = 2, alpha = 0.5, show.legend = FALSE) +  
6   labs(x = expression(mu), y = expression(sigma)) +  
7   viridis::scale_color_viridis()
```





# Récupérer les échantillons de la distribution postérieure

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

```
# A draws_df: 6 iterations, 1 chains, and 5 variables
  b_Intercept sigma lprior  lp__ density
1         155    7.8   -9.3 -1227    0.85
2         154    7.7   -9.3 -1227    0.80
3         154    7.4   -9.3 -1228    0.20
4         155    7.7   -9.3 -1227    1.19
5         154    7.7   -9.3 -1227    1.10
6         155    7.4   -9.3 -1228    0.52
# ... hidden reserved variables {'.chain', '.iteration', '.draw'}
```

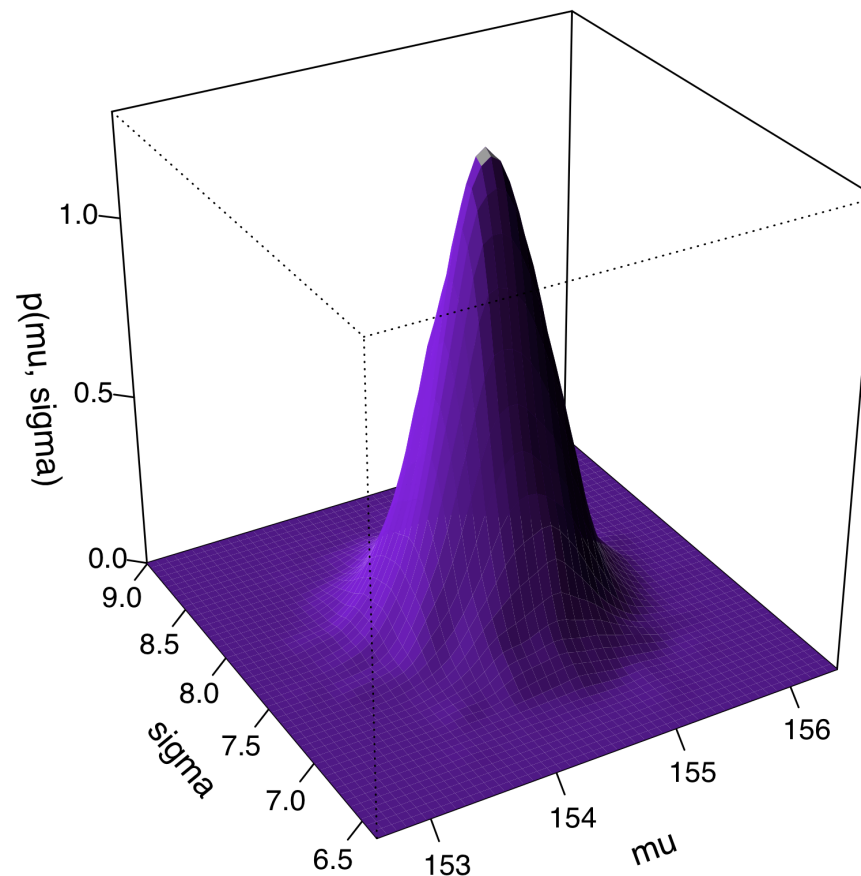
```
1 # gets the median and the 95% credible interval
2 t(sapply(post[, 1:2], quantile, probs = c(0.025, 0.5, 0.975) ) )
```

	2.5%	50%	97.5%
b_Intercept	153.790176	154.620248	155.42834
sigma	7.189477	7.758445	8.36522

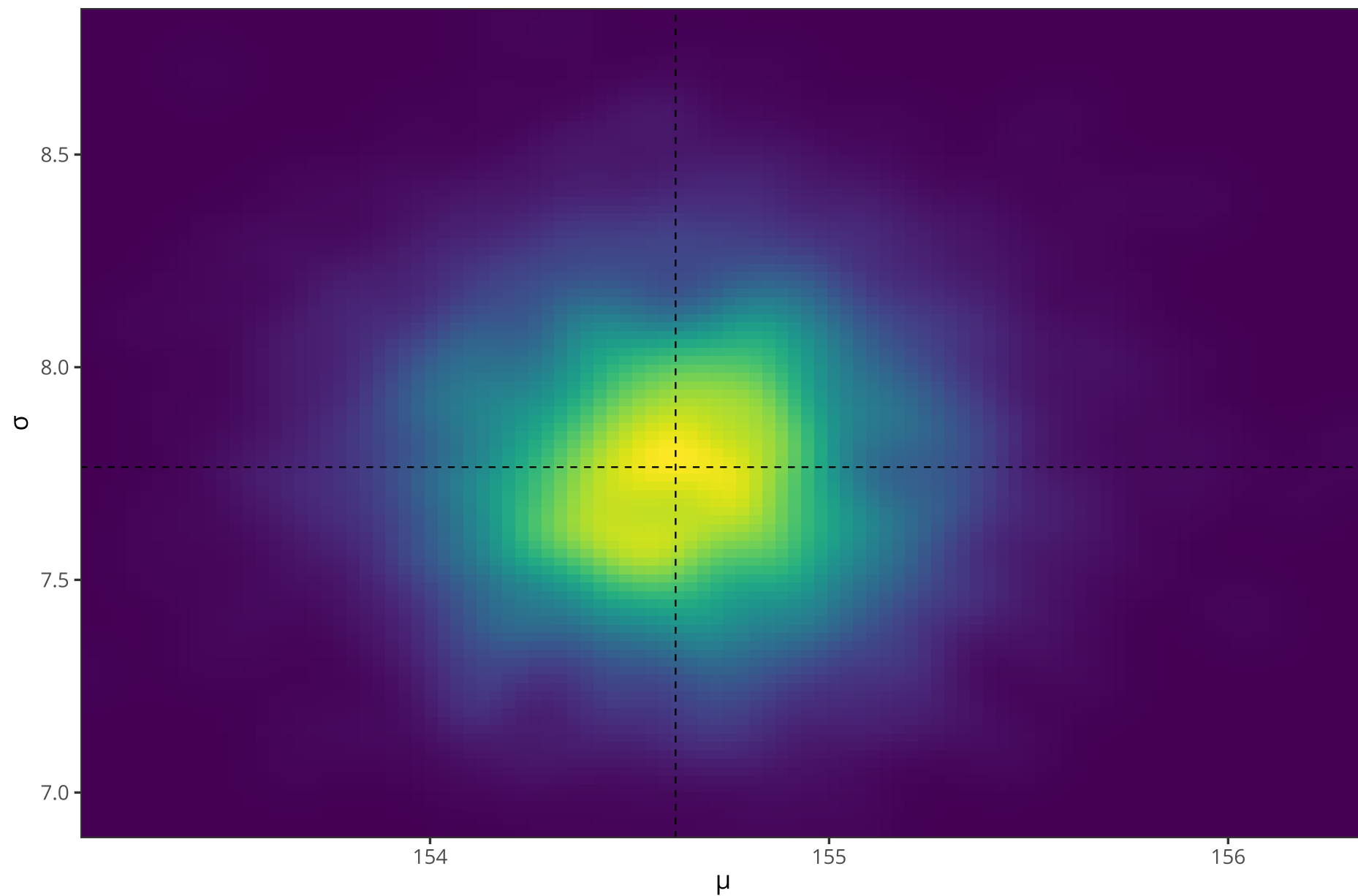


# Visualiser la distribution postérieure

```
1 H.scv <- Hscv(post[, 1:2])
2 fhat_post <- kde(x = post[, 1:2], H = H.scv, compute.cont = TRUE)
3
4 plot(fhat_post, display = "persp", col = "purple", border = NA,
5      xlab = "\nmu", ylab = "\nsigma", zlab = "\np(mu, sigma)",
6      shade = 0.8, phi = 30, ticktype = "detailed",
7      cex.lab = 1.2, family = "Helvetica")
```



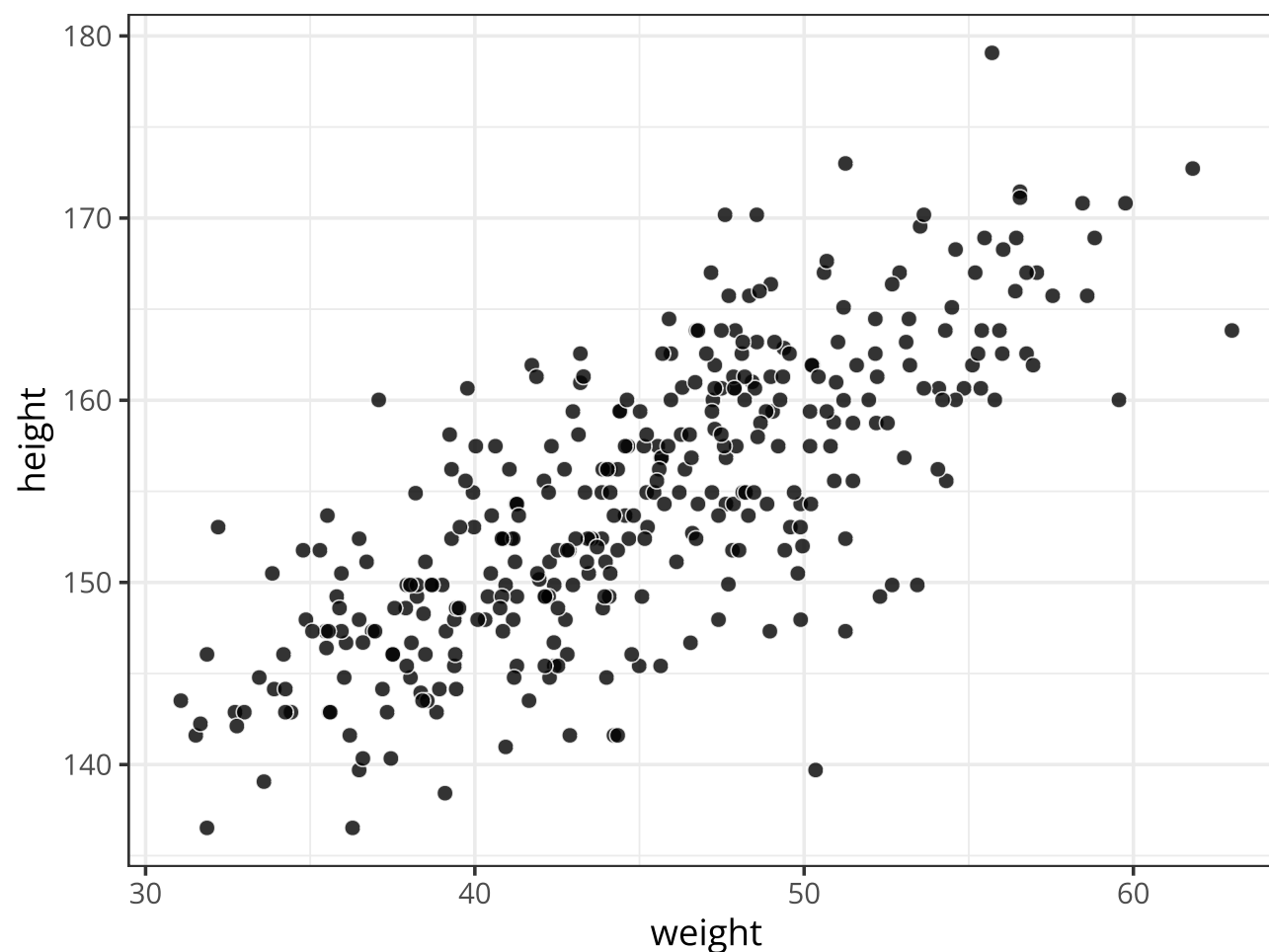
# Visualiser la distribution postérieure



# Ajouter un prédicteur

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

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



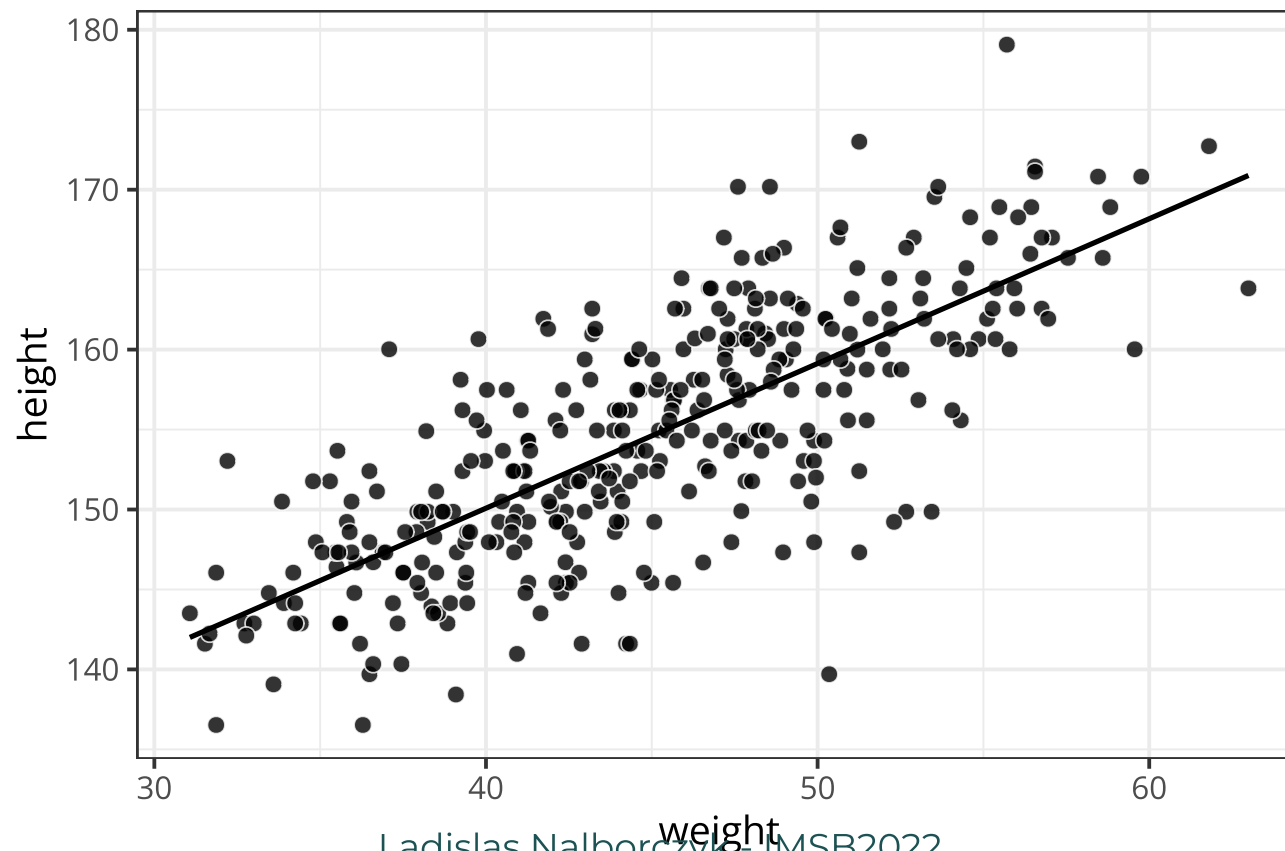
# Régression linéaire à un prédicteur continu

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

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

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

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



# Différentes notations

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

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

est équivalente à :

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

et si on réduit encore un peu :

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

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



# Régression linéaire à un prédicteur continu

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

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

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

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

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

Dans ce modèle  $\mu$  n'est plus un paramètre à estimer (car  $\mu$  est **déterminé** par  $\alpha$  et  $\beta$ ). À la place, nous allons estimer  $\alpha$  et  $\beta$ .

Rappels :  $\alpha$  est l'intercept, c'est à dire la taille attendue, lorsque le poids est égal à 0.  $\beta$  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

```
1 priors <- c(  
2   prior(normal(178, 20), class = Intercept),  
3   prior(normal(0, 10), class = b),  
4   prior(exponential(0.01), class = sigma)  
5 )  
6  
7 mod4 <- brm(  
8   height ~ 1 + weight,  
9   prior = priors,  
10  family = gaussian(),  
11  data = d2  
12 )
```





# Régression linéaire à un prédicteur continu

```
1 posterior_summary(mod4)
```

	Estimate	Est.Error	Q2.5	Q97.5
b_Intercept	113.8890577	1.85606722	110.1498814	117.5807237
b_weight	0.9049617	0.04089134	0.8237468	0.9871104
sigma	5.0997347	0.19666803	4.7455662	5.5033403
lprior	-12.4807870	0.01575025	-12.5121078	-12.4498495
lp__	-1083.3414224	1.20407429	-1086.5553601	-1081.9651776

- $\alpha = 113.89$ , 95% CrI [110.15, 117.58] représente la taille moyenne quand le poids est égal à 0kg...
- $\beta = 0.90$ , 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

```
1 d2$weight.c <- d2$weight - mean(d2$weight)
2
3 mod5 <- brm(
4   height ~ 1 + weight.c,
5   prior = priors,
6   family = gaussian(),
7   data = d2
8 )
```

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

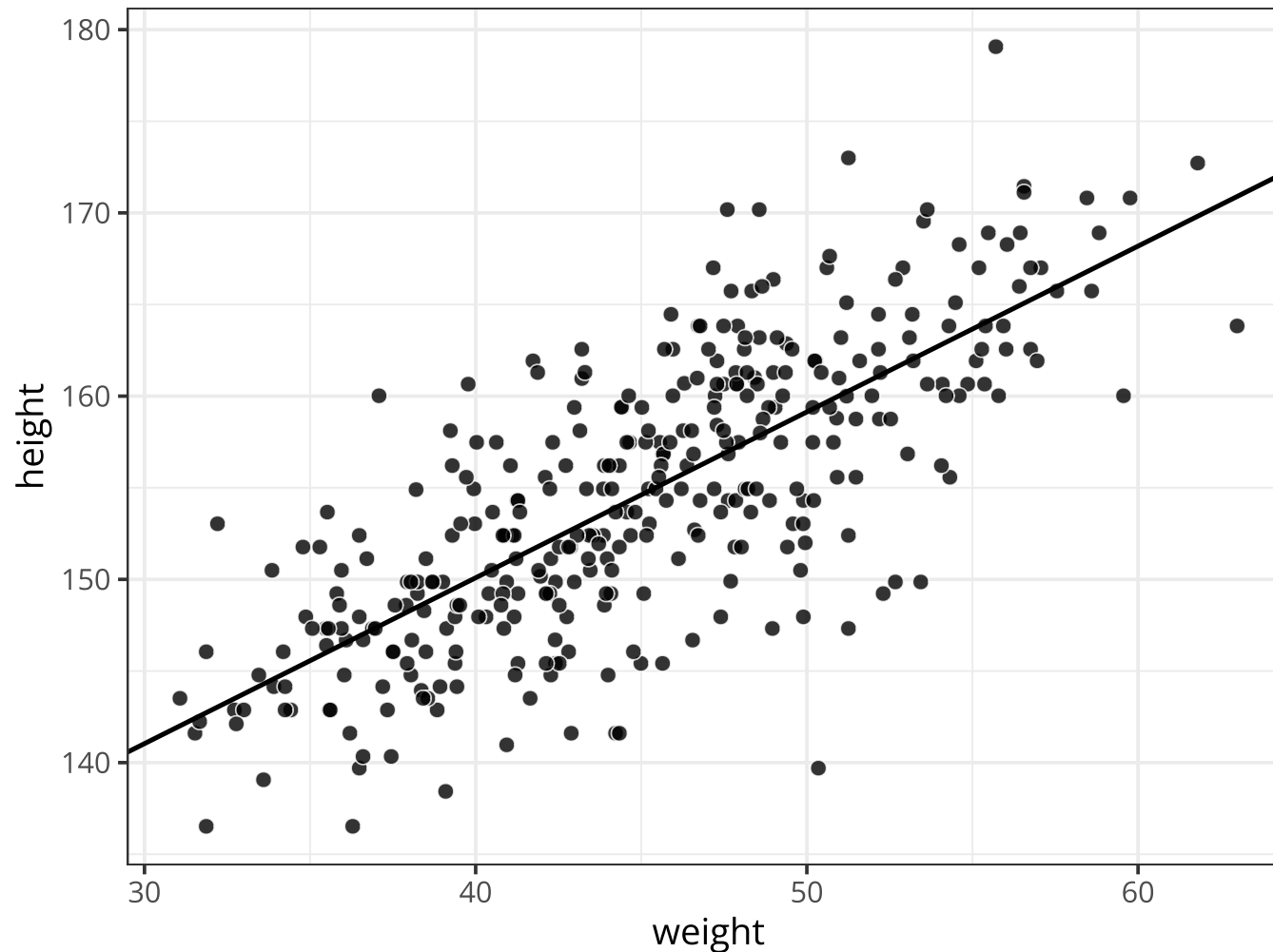
	Estimate	Est.Error	Q2.5	Q97.5
Intercept	154.6031253	0.2737608	154.0670147	155.1436516
weight.c	0.9063922	0.0419498	0.8264103	0.9903136

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



# Représenter les prédictions du modèle

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



# Représenter l'incertitude sur $\mu$ via fitted()

```

1 # on crée un vecteur de valeurs possibles pour "weight"
2 weight.seq <- data.frame(weight = seq(from = 25, to = 70, by = 1) )
3
4 # on récupère les prédictions du modèle pour ces valeurs de poids
5 mu <- data.frame(fitted(mod4, newdata = weight.seq) ) %>% bind_cols(weight.seq)
6
7 # on affiche les 10 premières lignes de mu
8 head(mu, 10)

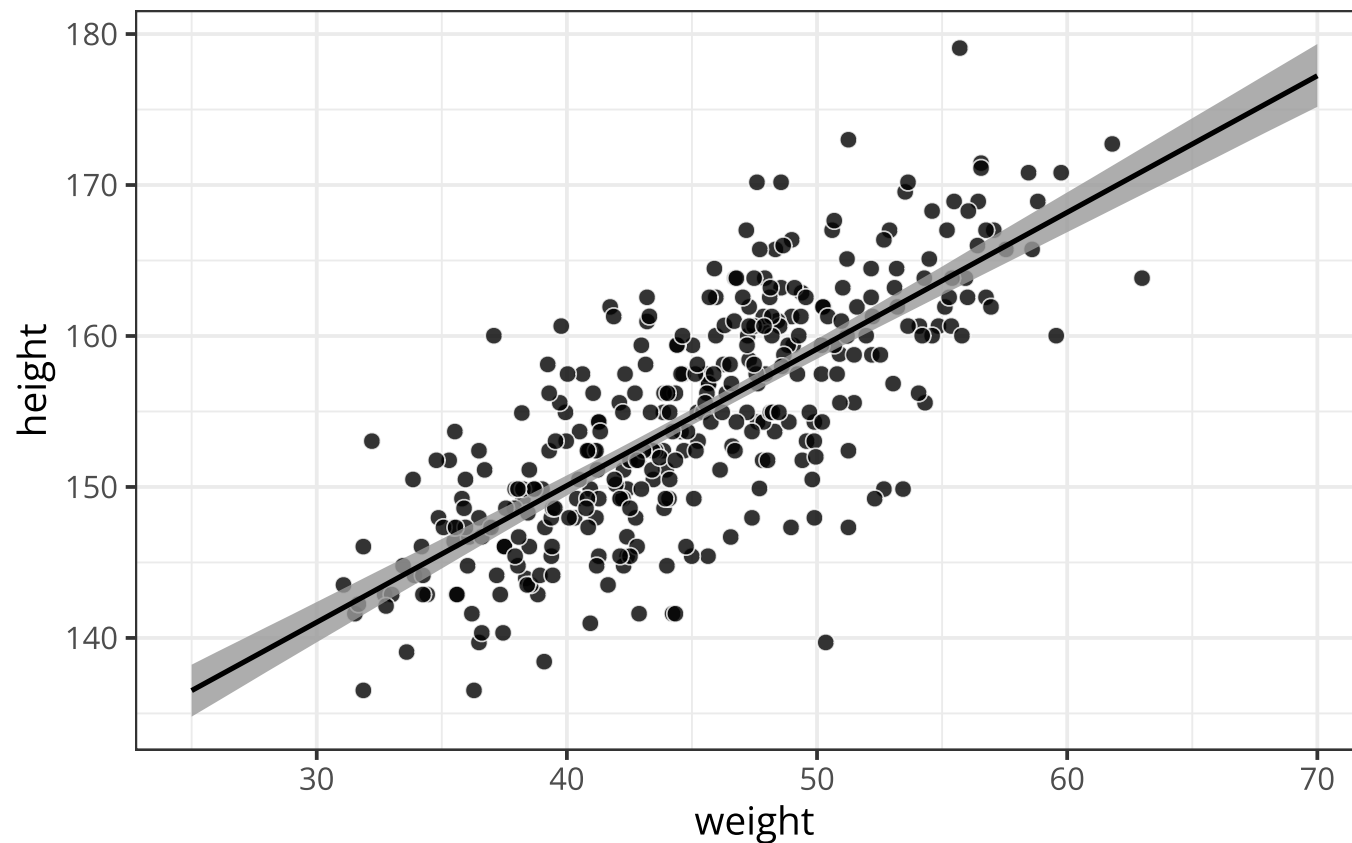
```

	Estimate	Est.Error	Q2.5	Q97.5	weight
1	136.5131	0.8572681	134.7868	138.2284	25
2	137.4181	0.8185231	135.7703	139.0651	26
3	138.3230	0.7799974	136.7587	139.8900	27
4	139.2280	0.7417249	137.7519	140.7220	28
5	140.1329	0.7037471	138.7301	141.5440	29
6	141.0379	0.6661144	139.7110	142.3806	30
7	141.9429	0.6288887	140.6854	143.2139	31
8	142.8478	0.5921467	141.6684	144.0537	32
9	143.7528	0.5559845	142.6534	144.8882	33
10	144.6578	0.5205227	143.6219	145.7115	34



# Représenter l'incertitude sur $\mu$ via fitted()

```
1 d2 %>%  
2   ggplot(aes(x = weight, y = height) ) +  
3   geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +  
4   geom_smooth(  
5     data = mu, aes(y = Estimate, ymin = Q2.5, ymax = Q97.5),  
6     stat = "identity",  
7     color = "black", alpha = 0.8, size = 1  
8   )
```



# Intervalles de prédiction (incorporer $\sigma$ )

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  $\mu$ . Comment incorporer  $\sigma$  dans nos prédictions ?

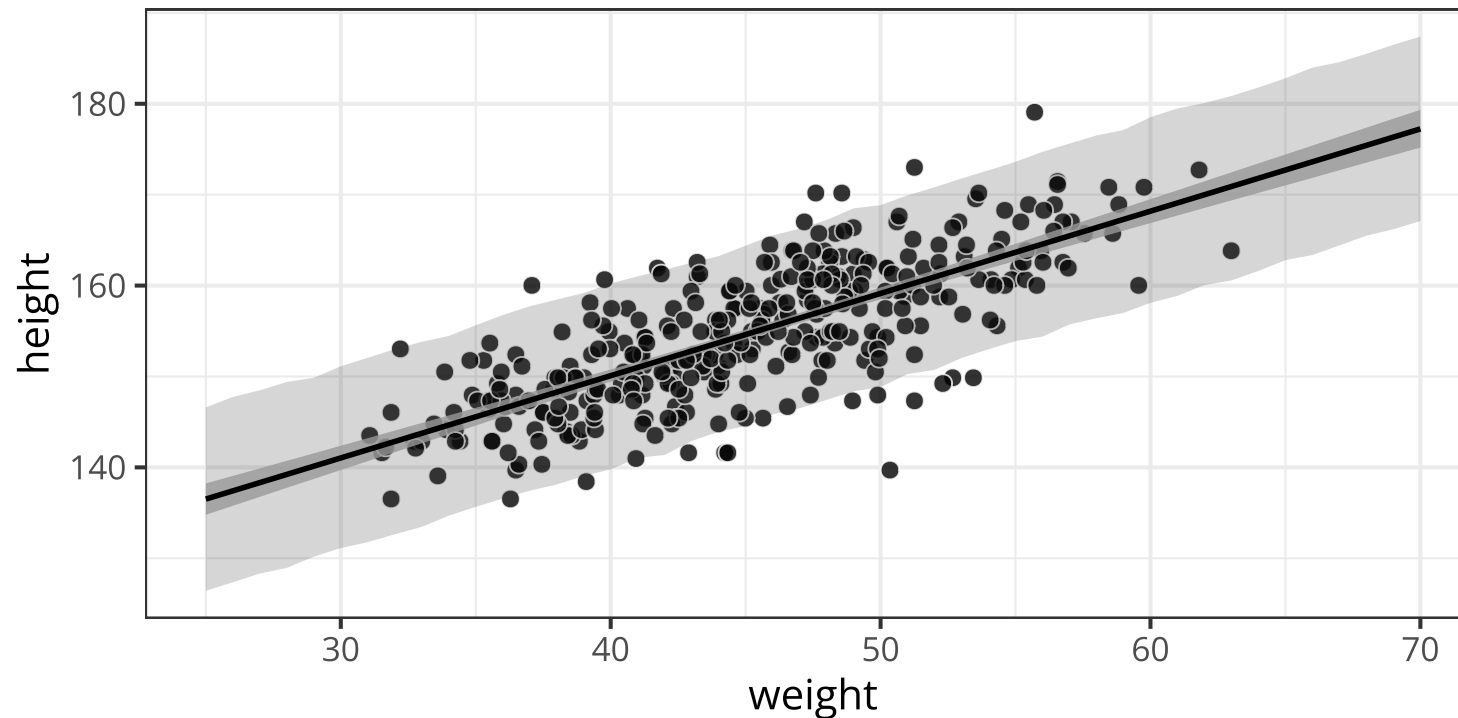
```
1 # on crée un vecteur de valeurs possibles pour "weight"
2 weight.seq <- data.frame(weight = seq(from = 25, to = 70, by = 1) )
3
4 # on récupère les prédictions du modèle pour ces valeurs de poids
5 pred_height <- data.frame(predict(mod4, newdata = weight.seq) ) %>% bind_cols(weight.seq)
6
7 # on affiche les 10 premières lignes de pred_height
8 head(pred_height, 10)
```

	Estimate	Est.Error	Q2.5	Q97.5	weight
1	136.4098	5.186568	126.4272	146.6054	25
2	137.3522	5.269245	127.3612	147.7474	26
3	138.2561	5.219190	128.3187	148.4856	27
4	139.3065	5.225907	128.9418	149.3919	28
5	140.1270	5.172048	130.1793	149.8614	29
6	141.0888	5.159174	131.1203	151.1209	30
7	141.8411	5.200899	131.7751	152.0549	31
8	142.8074	5.175875	132.6406	153.0057	32
9	143.8191	5.132382	133.4520	153.8159	33
10	144.6263	5.051845	134.6868	154.4553	34



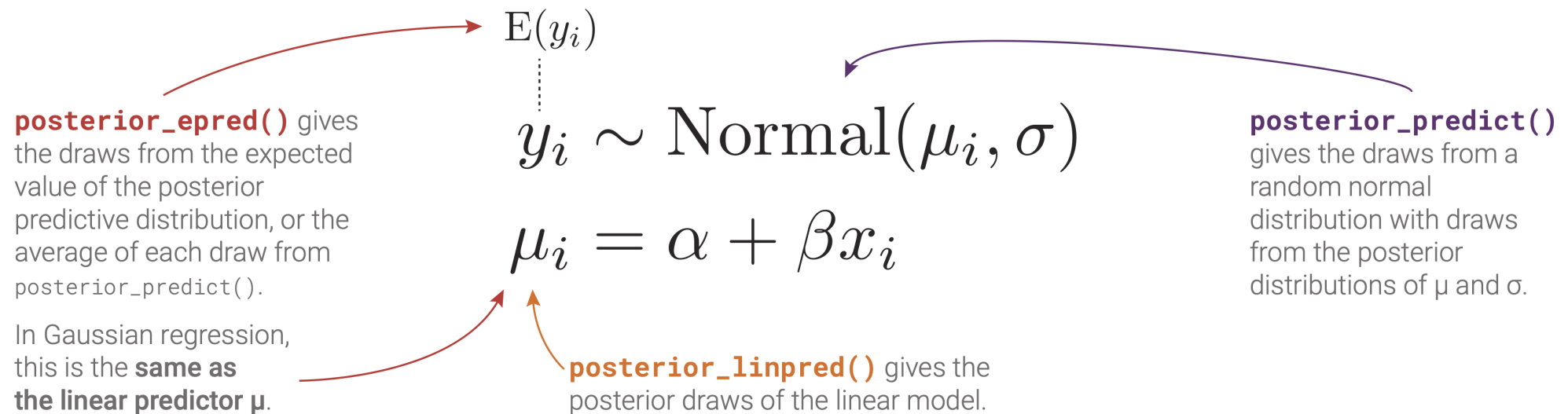
# Intervalles de prédiction (incorporer $\sigma$ )

```
1 d2 %>%
2   ggplot(aes(x = weight, y = height) ) +
3   geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
4   geom_ribbon(
5     data = pred_height, aes(x = weight, ymin = Q2.5, ymax = Q97.5),
6     alpha = 0.2, inherit.aes = FALSE
7   ) +
8   geom_smooth(
9     data = mu, aes(y = Estimate, ymin = Q2.5, ymax = Q97.5),
10    stat = "identity", color = "black", alpha = 0.8, size = 1
11  )
```



# Built-in brms functions

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





# Rappel : Deux types d'incertitude

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

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

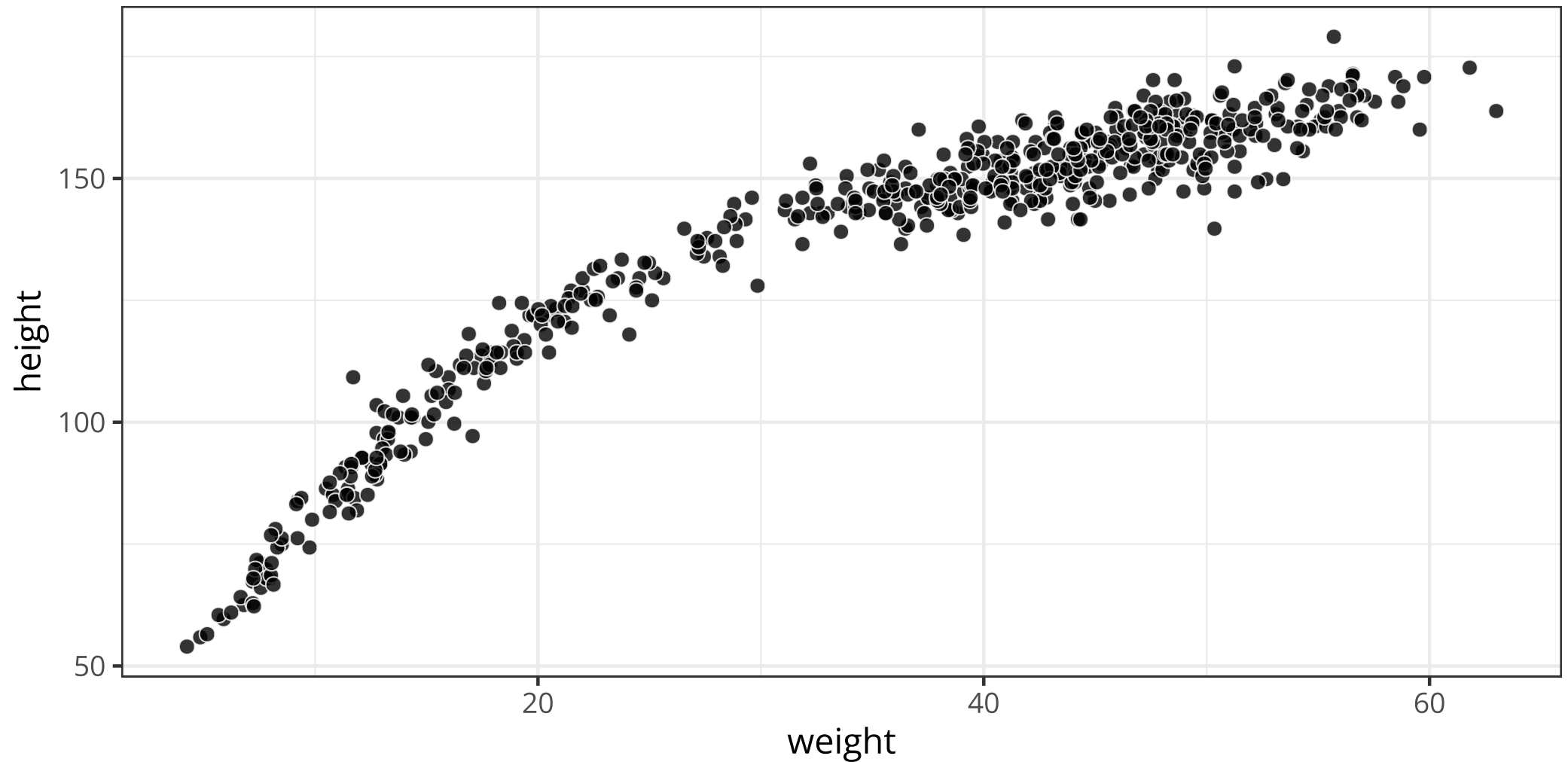
**Incertitude aléatoire** : La distribution des données simulées est elle, une distribution qui contient de l'incertitude liée à un processus d'échantillonnage (i.e., générer des données à partir d'une gaussienne).

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



# Régression polynomiale

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

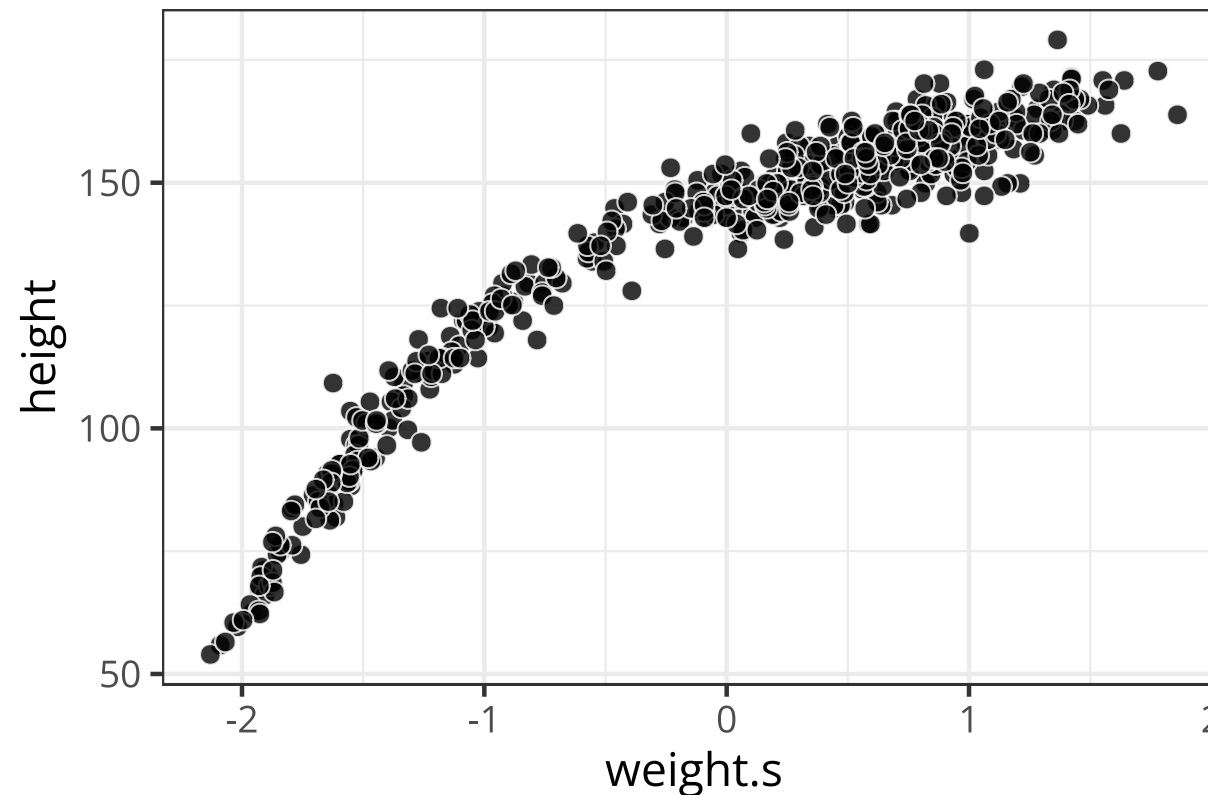


# Scores standardisés

```
1 d <- d %>% mutate(weight.s = (weight - mean(weight) ) / sd(weight) )  
2  
3 d %>%  
4   ggplot(aes(x = weight.s, y = height) ) +  
5   geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8)
```

```
1 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 (ou des valeurs trop différentes les unes des autres), cela peut poser des problèmes de convergence (cf. Cours n°05).



# Modèle de régression polynomiale - Exercice

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

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

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

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

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

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



# Modèle de régression polynomiale

```
1 priors <- c(  
2   prior(normal(156, 100), class = Intercept),  
3   prior(normal(0, 10), class = b),  
4   prior(exponential(0.01), class = sigma)  
5 )  
6  
7 mod6 <- brm(  
8   # NB: polynomials should be written with the I() function...  
9   height ~ 1 + weight.s + I(weight.s^2),  
10  prior = priors,  
11  family = gaussian(),  
12  data = d  
13 )
```



# Modèle de régression polynomiale

```
1 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    146.67     0.37   145.92   147.38 1.00    3849    2603
weight.s      21.40     0.29    20.85    21.97 1.00    3830    2853
Iweight.sE2   -8.41     0.28    -8.96    -7.85 1.00    3652    2763

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma      5.78      0.18     5.45     6.14 1.00    3838    2518

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

```

1 # on crée un vecteur de valeurs possibles pour "weight"
2 weight.seq <- data.frame(weight.s = seq(from = -2.5, to = 2.5, length.out = 50) )
3
4 # on récupère les prédictions du modèle pour ces valeurs de poids
5 mu <- data.frame(fitted(mod6, newdata = weight.seq) ) %>% bind_cols(weight.seq)
6 pred_height <- data.frame(predict(mod6, newdata = weight.seq) ) %>% bind_cols(weight.seq)
7
8 # on affiche les 10 premières lignes de pred_height
9 head(pred_height, 10)

```

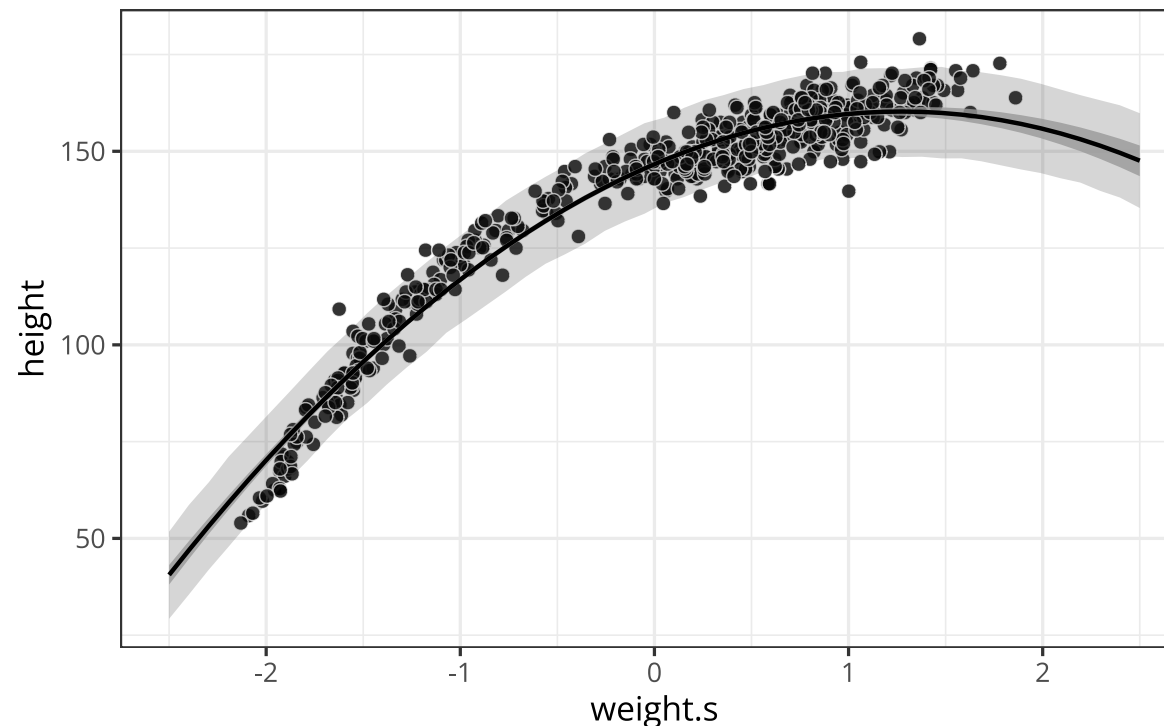
	Estimate	Est.Error	Q2.5	Q97.5	weight.s
1	40.57892	5.780568	29.22709	51.73450	-2.500000
2	46.93764	5.888279	35.50626	58.64793	-2.397959
3	53.19076	5.773016	41.89905	64.46848	-2.295918
4	59.19972	5.999697	47.83277	71.07507	-2.193878
5	65.25943	5.839142	53.87375	76.51191	-2.091837
6	70.79875	5.846949	59.36606	81.99090	-1.989796
7	76.30174	5.809967	64.98177	87.42384	-1.887755
8	81.63006	5.810998	70.14310	92.92906	-1.785714
9	86.69686	5.725256	75.74265	98.34396	-1.683673
10	91.86560	5.698405	80.90886	103.08358	-1.581633





# Représenter les prédictions du modèle

```
1 d %>%  
2   ggplot(aes(x = weight.s, y = height) ) +  
3   geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +  
4   geom_ribbon(  
5     data = pred_height, aes(x = weight.s, ymin = Q2.5, ymax = Q97.5),  
6     alpha = 0.2, inherit.aes = FALSE  
7   ) +  
8   geom_smooth(  
9     data = mu, aes(y = Estimate, ymin = Q2.5, ymax = Q97.5),  
10    stat = "identity", color = "black", alpha = 0.8, size = 1  
11  )
```



# Modèle de régression, taille d'effet

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

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

“

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



# Modèle de régression, taille d'effet

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

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

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

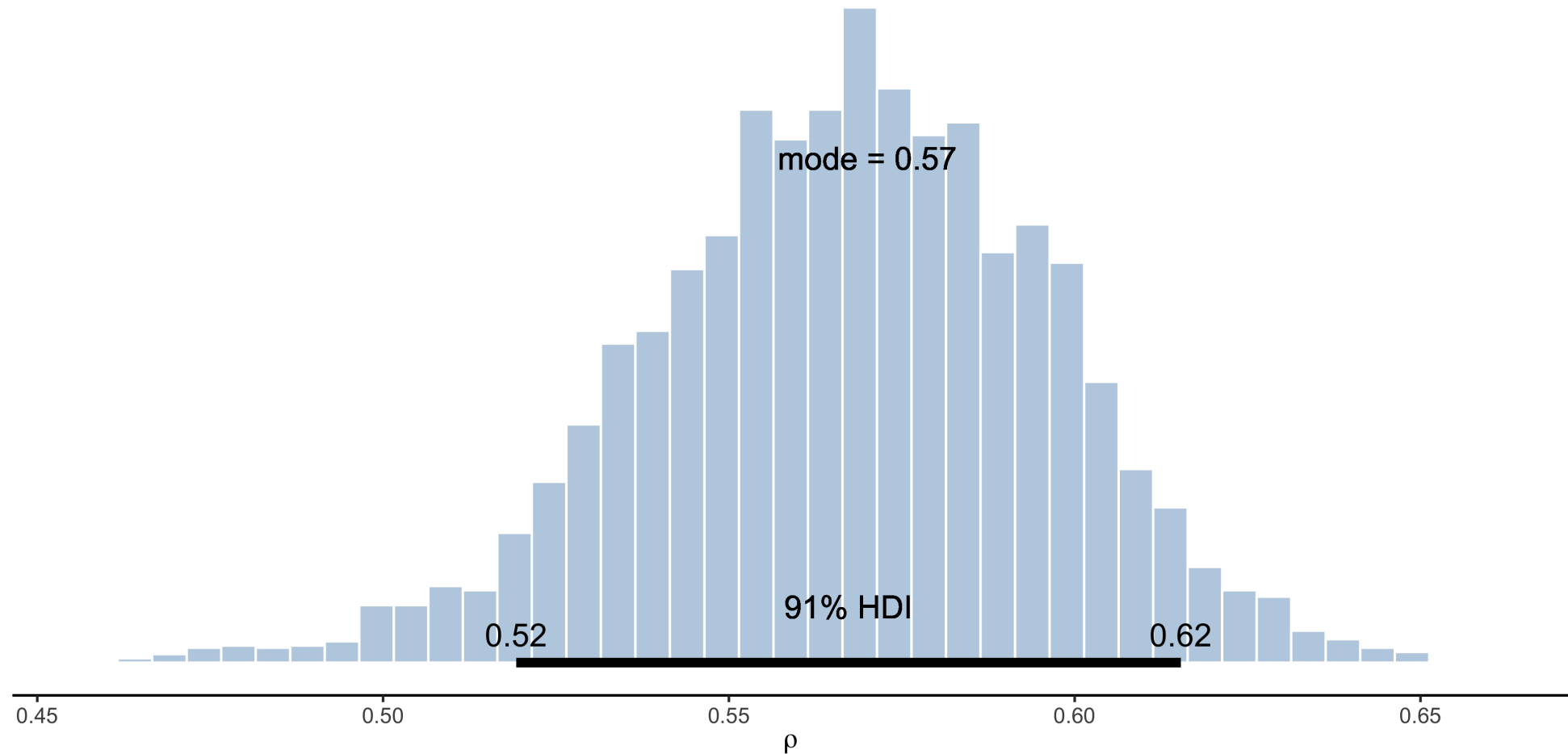
```
1 post <- as_draws_df(x = mod4)
2 beta <- post$b_weight
3 sigma <- post$sigma
4 rho <- beta^2 * var(d2$weight) / (sigma^2 + beta^2 * var(d2$weight) )
```

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



# Modèle de régression, taille d'effet

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



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

```
[1] 0.5696444
```



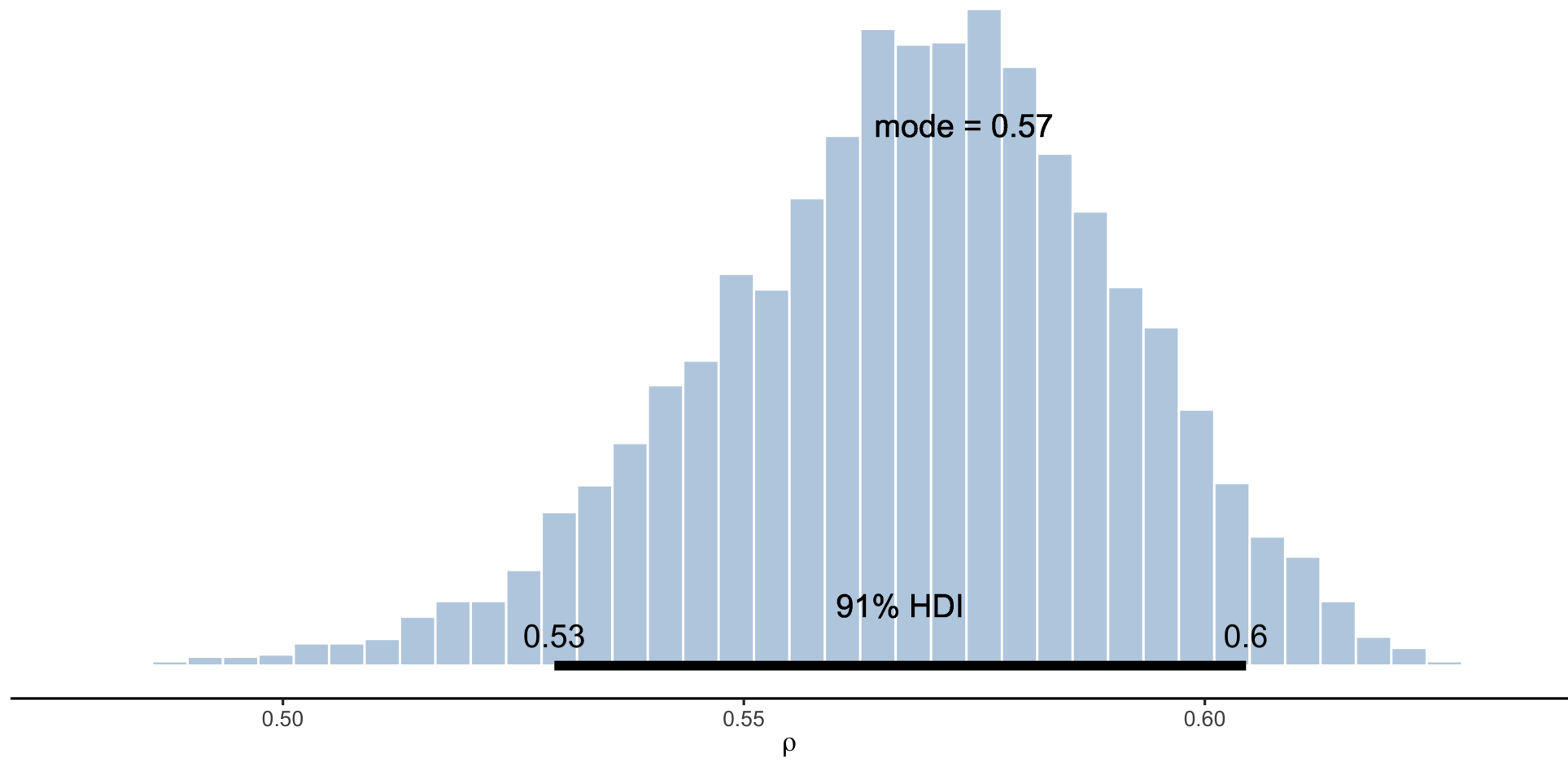
# Modèle de régression, taille d'effet

```
1 bayes_R2(mod4)
```

	Estimate	Est.Error	Q2.5	Q97.5
R2	0.568312	0.02230833	0.5203817	0.6090215

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





# Résumé du cours

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

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

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

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

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



# Travaux pratiques - 1/2

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

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

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

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





## Travaux pratiques - 2/2

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

Modéliser alors la relation entre la taille (cm) et le log du poids (log-kg). Utiliser la dataframe `howell1` en entier (les 544 lignes). Fitter le modèle suivant en utilisant `brms::brm()`.

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

$$\mu_i = \alpha + \beta \cdot \log(w_i)$$

$$\alpha \sim \text{Normal}(178, 100)$$

$$\beta \sim \text{Normal}(0, 100)$$

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

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

```
1 # on garde seulement les individus ayant moins de 18 ans
2 d <- open_data(howell) %>% filter(age < 18)
3
4 priors <- c(
5   prior(normal(150, 100), class = Intercept),
6   prior(normal(0, 10), class = b),
7   prior(exponential(0.01), class = sigma)
8 )
9
10 mod7 <- brm(
11   height ~ 1 + weight,
12   prior = priors,
13   family = gaussian(),
14   data = d
15 )
```



# Proposition de solution

```
1 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    58.20     1.40   55.92   60.47 1.00    4066    3128
weight        2.72     0.07    2.61    2.83 1.00    3954    2910

Family Specific Parameters:
      Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
sigma      8.53     0.45    7.83    9.27 1.00    3781    2772

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

```

1 # on crée un vecteur de valeurs possibles pour "weight"
2 weight.seq <- data.frame(weight = seq(from = 5, to = 45, length.out = 1e2) )
3
4 # on récupère les prédictions du modèle pour ces valeurs de poids
5 mu <- data.frame(
6   fitted(mod7, newdata = weight.seq, probs = c(0.055, 0.945) )
7 ) %>%
8   bind_cols(weight.seq)
9
10 pred_height <- data.frame(
11   predict(mod7, newdata = weight.seq, probs = c(0.055, 0.945) )
12 ) %>%
13   bind_cols(weight.seq)
14
15 # on affiche les 6 premières lignes de pred_height
16 head(pred_height)

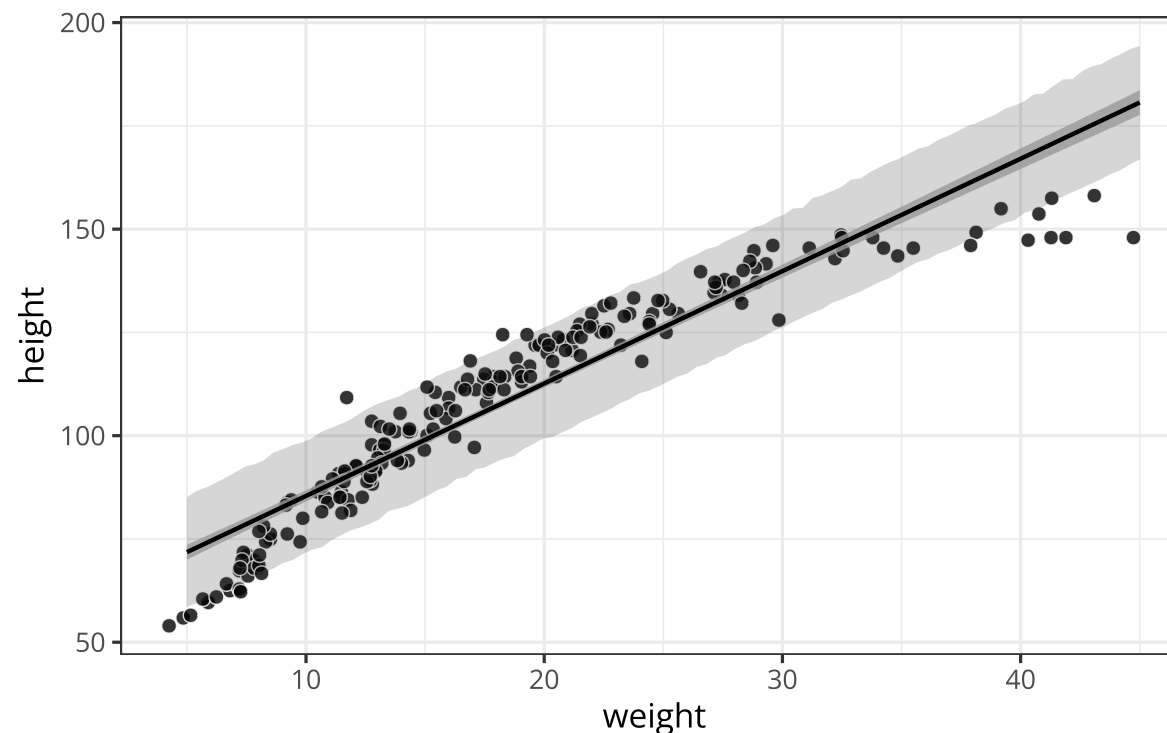
```

	Estimate	Est.Error	Q5.5	Q94.5	weight
1	71.66877	8.423002	58.55938	85.12889	5.000000
2	73.08458	8.607555	59.45684	86.61345	5.404040
3	73.74920	8.534452	60.15777	87.46164	5.808081
4	75.09736	8.569803	61.52997	88.58663	6.212121
5	76.33358	8.573160	62.58027	89.77716	6.616162
6	77.32169	8.525510	63.42130	91.10404	7.020202



# Représenter les prédictions du modèle

```
1 d %>%  
2   ggplot(aes(x = weight, y = height) ) +  
3   geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +  
4   geom_ribbon(  
5     data = pred_height, aes(x = weight, ymin = Q5.5, ymax = Q94.5),  
6     alpha = 0.2, inherit.aes = FALSE  
7   ) +  
8   geom_smooth(  
9     data = mu, aes(y = Estimate, ymin = Q5.5, ymax = Q94.5),  
10    stat = "identity", color = "black", alpha = 0.8, size = 1  
11  )
```



# Proposition de solution

```
1 # on considère maintenant tous les individus
2 d <- open_data(howell)
3
4 mod8 <- brm(
5   # on prédit la taille par le logarithme du poids
6   height ~ 1 + log(weight),
7   prior = priors,
8   family = gaussian(),
9   data = d
10 )
```



# Proposition de solution

```
1 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      1.34   -25.80   -21.45 1.00     3743     2855
logweight     47.02      0.39    46.40    47.65 1.00     3709     2819

Family Specific Parameters:
      Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
sigma      5.16      0.16     4.92     5.42 1.00     3780     2869

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

```

1 # on crée un vecteur de valeurs possibles pour "weight"
2 weight.seq <- data.frame(weight = seq(from = 5, to = 65, length.out = 1e2) )
3
4 # on récupère les prédictions du modèle pour ces valeurs de poids
5 mu <- data.frame(
6   fitted(mod8, newdata = weight.seq, probs = c(0.055, 0.945) )
7 ) %>%
8   bind_cols(weight.seq)
9
10 pred_height <- data.frame(
11   predict(mod8, newdata = weight.seq, probs = c(0.055, 0.945) )
12 ) %>%
13   bind_cols(weight.seq)
14
15 # on affiche les 6 premières lignes de pred_height
16 head(pred_height)

```

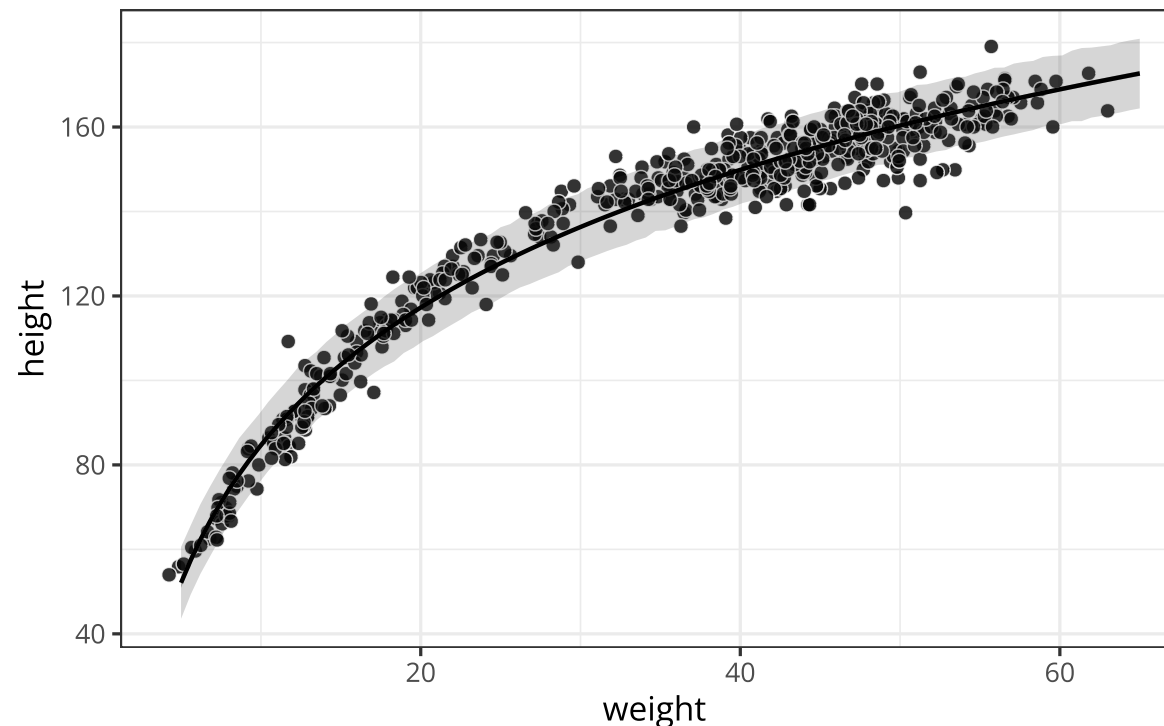
	Estimate	Est.Error	Q5.5	Q94.5	weight
1	52.02360	5.320630	43.55784	60.56774	5.000000
2	57.53741	5.100738	49.31835	65.68519	5.606061
3	62.47234	5.196777	54.19279	70.69335	6.212121
4	66.74357	5.256288	58.36315	74.89908	6.818182
5	70.61061	5.189019	62.46373	78.88959	7.424242
6	74.32148	5.123331	66.33714	82.59116	8.030303





# Représenter les prédictions du modèle

```
1 d %>%
2   ggplot(aes(x = weight, y = height) ) +
3   geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8) +
4   geom_ribbon(
5     data = pred_height, aes(x = weight, ymin = Q5.5, ymax = Q94.5),
6     alpha = 0.2, inherit.aes = FALSE
7   ) +
8   geom_smooth(
9     data = mu, aes(y = Estimate, ymin = Q5.5, ymax = Q94.5),
10    stat = "identity", color = "black", alpha = 0.8, size = 1
11  )
```



# Références

- Bürkner, P.-C. (2017). **brms** : An R Package for Bayesian Multilevel Models Using Stan. *Journal of Statistical Software*, 80(1). <https://doi.org/10.18637/jss.v080.i01>
- Gelman, A., & Pardoe, I. (2006). Bayesian measures of explained variance and pooling in multilevel (hierarchical) models. *Technometrics*, 48(2), 241–251. <https://doi.org/10.1198/0040170050000000517>
- Marsman, M., & Wagenmakers, E.-J. (2017). Three Insights from a Bayesian Interpretation of the One-Sided *P* Value. *Educational and Psychological Measurement*, 77(3), 529–539. <https://doi.org/10.1177/0013164416669201>
- Marsman, M., Waldorp, L., Dablander, F., & Wagenmakers, E.-J. (2019). Bayesian estimation of explained variance in ANOVA designs. *Statistica Neerlandica*, 0(0), 1–22. <https://doi.org/10.1111/stan.12173>
- O'Hagan, T. (2004). Dicing with the unknown. *Significance*, 1(3), 132–133. <https://doi.org/10.1111/j.1740-9713.2004.00050.x>

