Introduction to Bayesian statistical modelling

A course with R, Stan, and brms

Ladislas Nalborczyk (UNICOG, NeuroSpin, CEA, Gif/Yvette, France)

Planning

Course n°01: Introduction to Bayesian inference, Beta-Binomial model

Course n°02: Introduction to brms, linear regression

Course n°03: Markov Chain Monte Carlo, generalised linear model

Course n°04: Multilevel models, cognitive models

Setting up a common language

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

Aim of this session: to understand this type of model.

The components of our models will always be the same and we will always follow the following three steps:

- Build the model (likelihood + priors).
- Update with data to calculate the posterior distribution.
- Interpret the model's estimates and evaluate its predictions. If necessary, modify the model.

A first model

A first model

```r
library(tidyverse)
library(imsb)

d <- open_data(howell)
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 ...
```

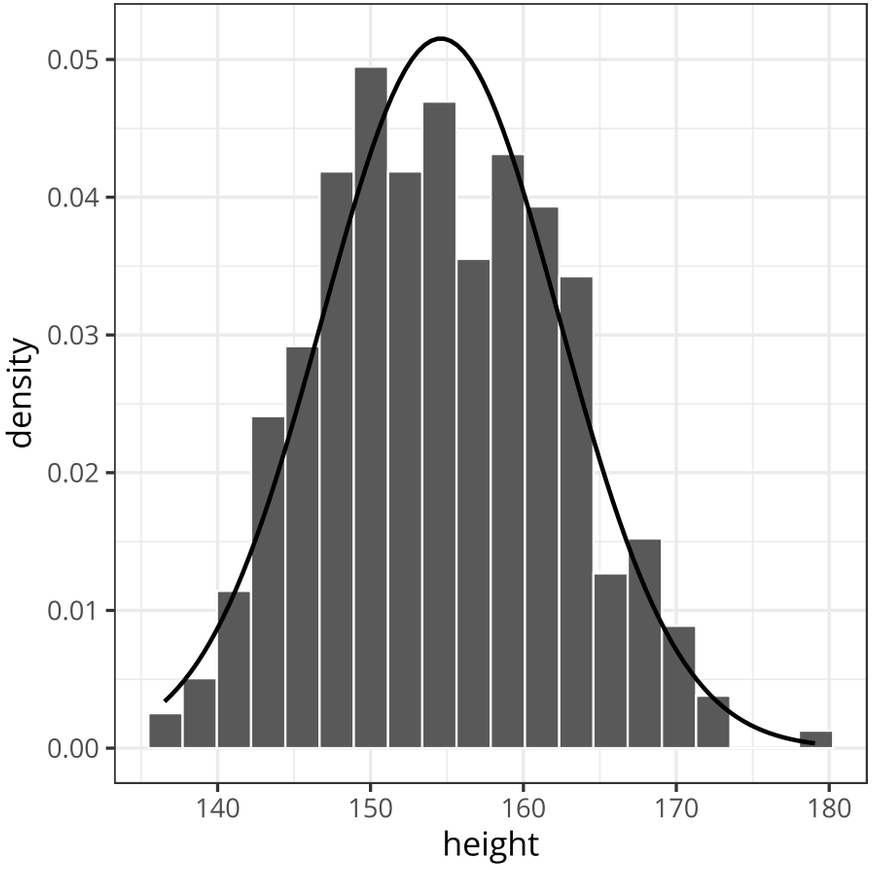
```r
d2 <- d %>% filter(age >= 18)
head(d2)
```

```
   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
```


A first model

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

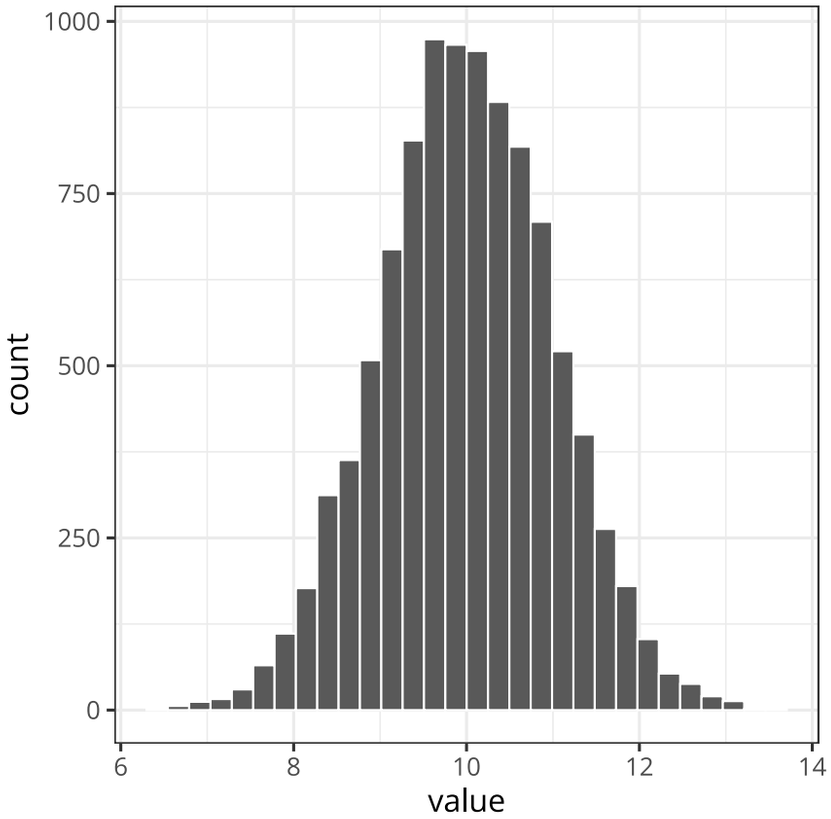
```
d2 %>%
    ggplot(aes(x = height) ) +
    geom_histogram(aes(y = ..density..), bins = 20, col = "white") +
    stat_function(fun = dnorm, args = list(mean(d2$height), sd(d2$height) ), size = 1)
```



Normal distribution

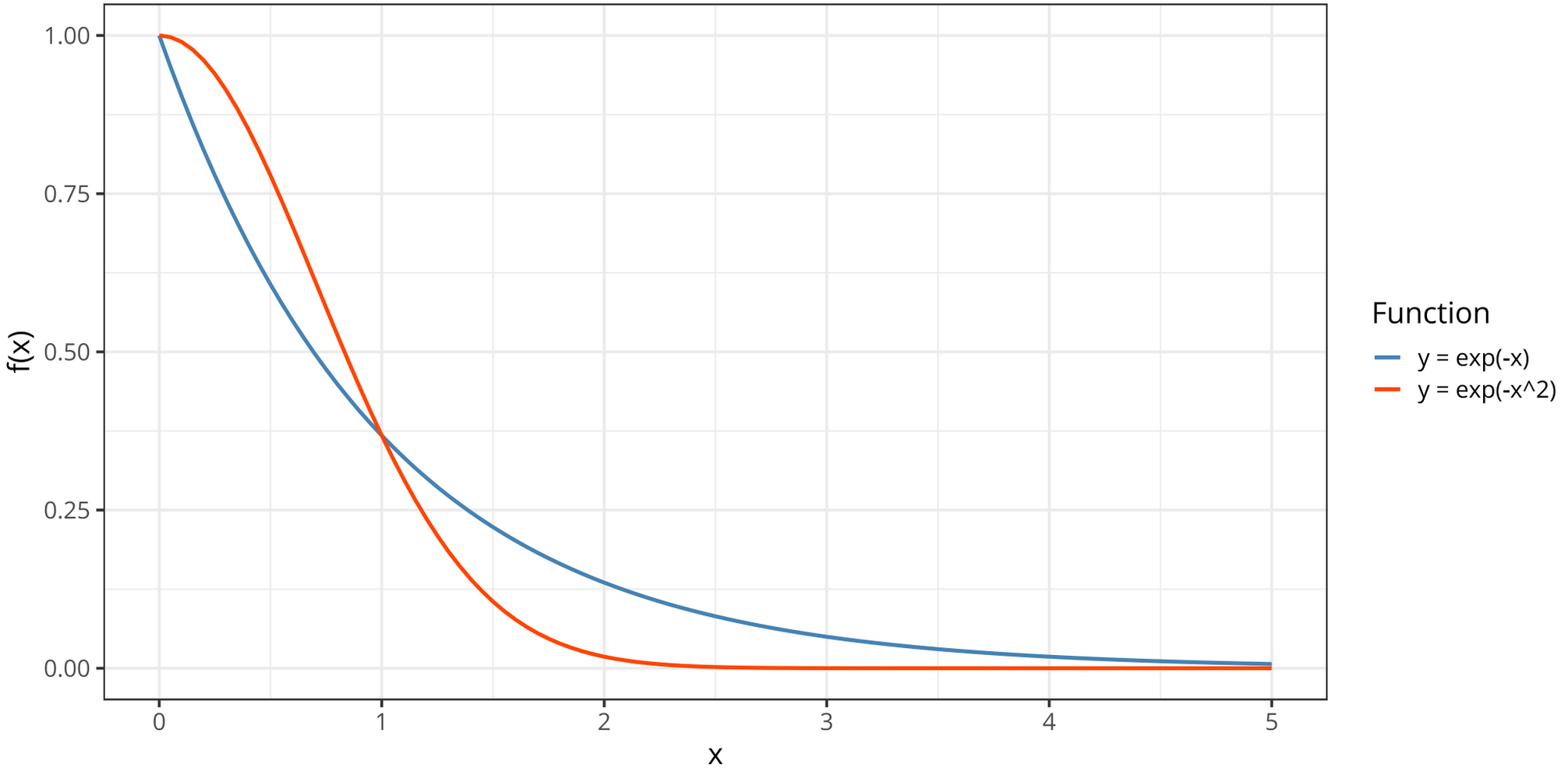
$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\mu - x)^2 \right]$$

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



Where does the normal distribution come from?

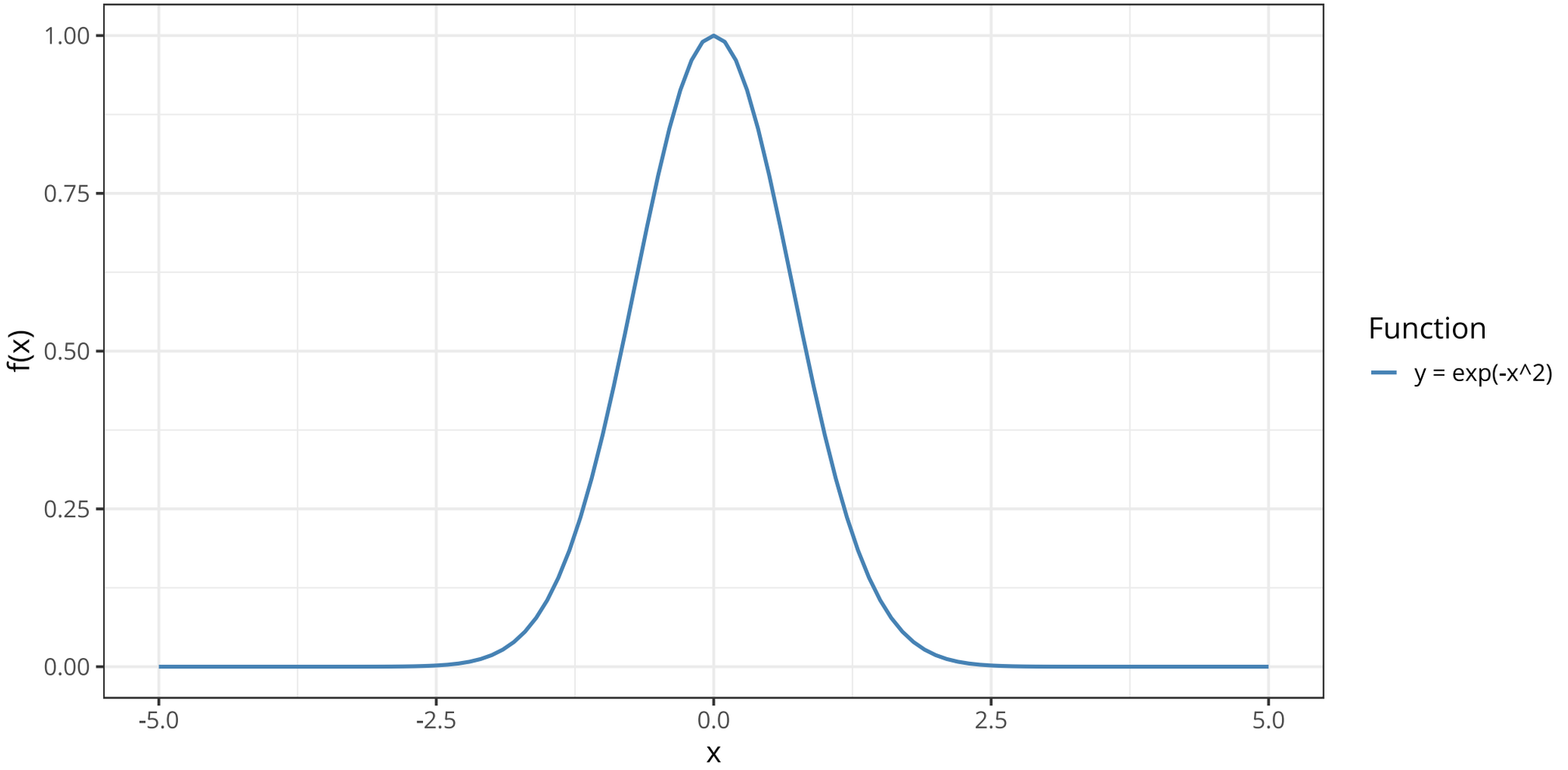
Some constraints: Some values are highly probable (around the mean μ). The further away they are, the less likely they are (following an exponential decay).



Where does the normal distribution come from?

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

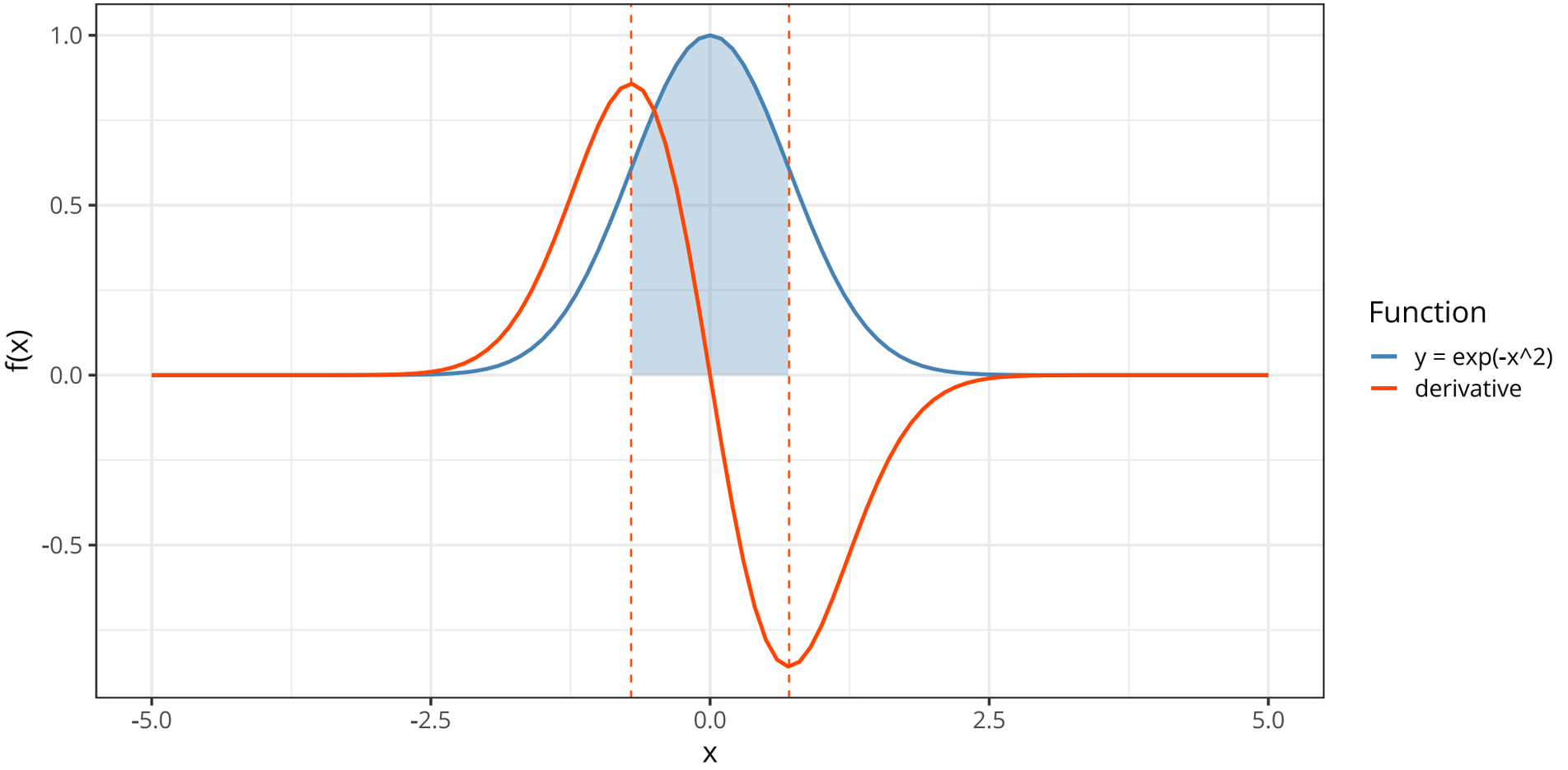
We extend our function to negative values.



Where does the normal distribution come from?

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

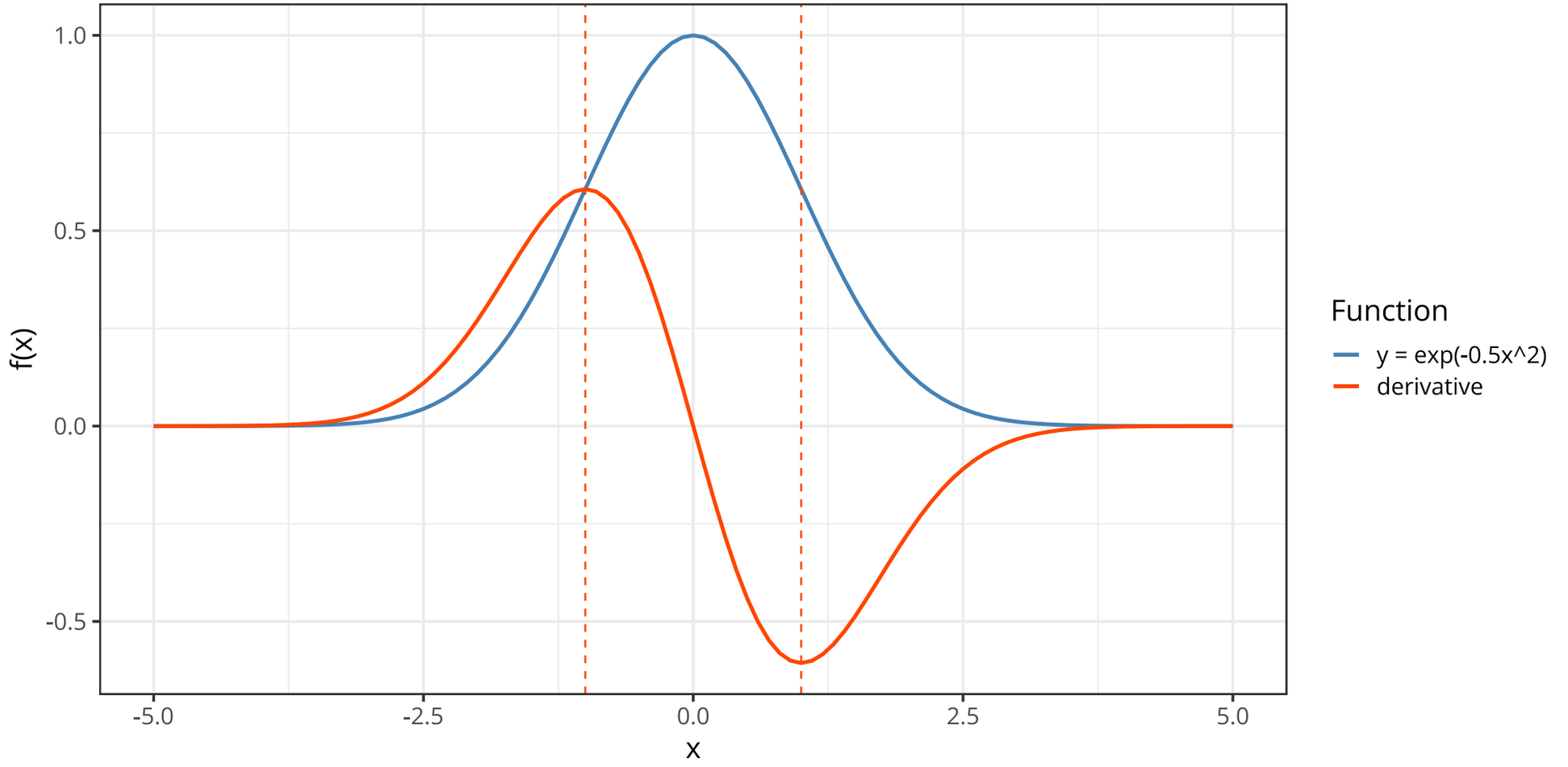
The inflection points give us a good indication of where most of the values lie (i.e., between the inflection points). The peaks of the derivative show us the inflection points.



Where does the normal distribution come from?

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

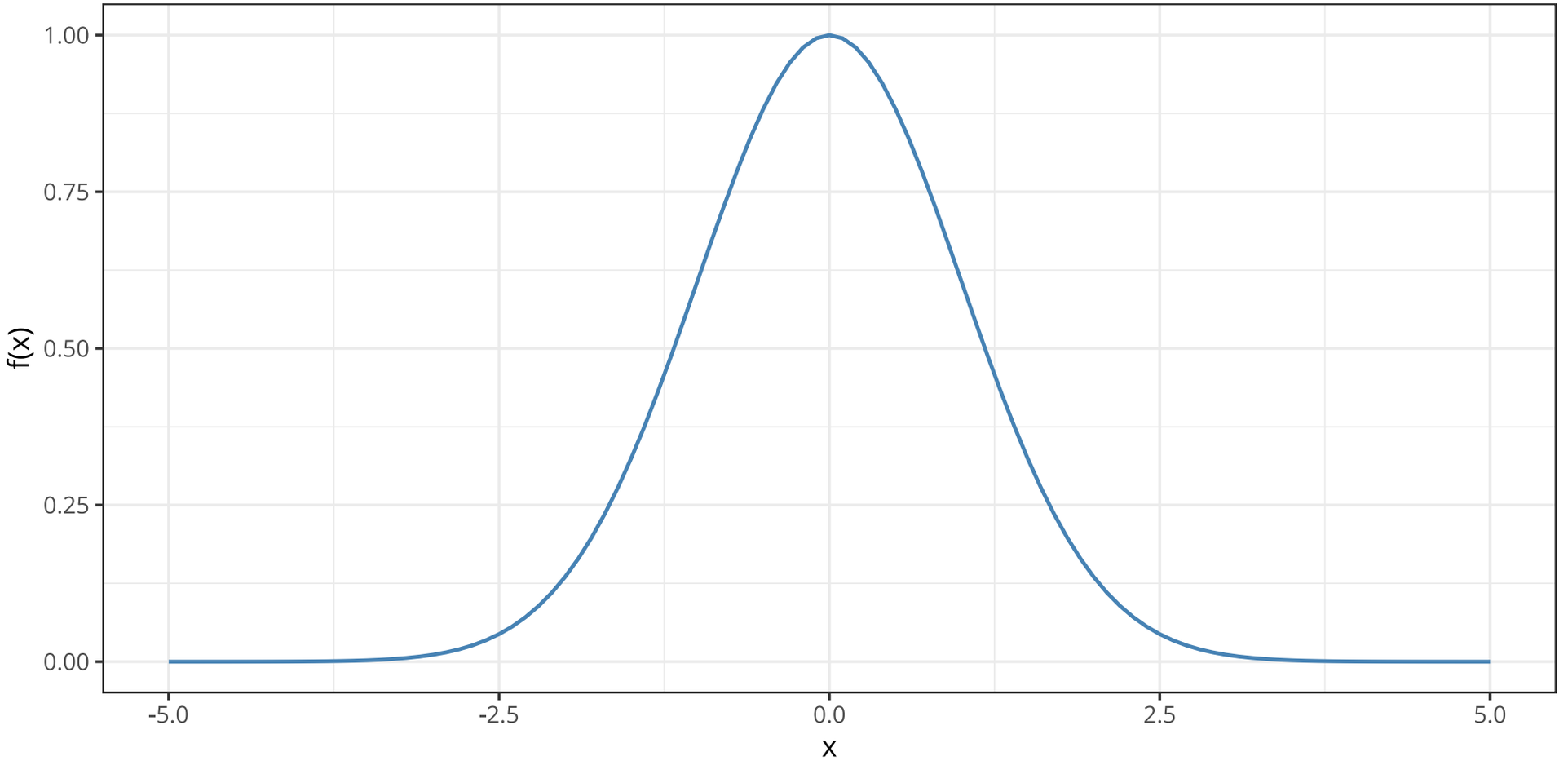
Next, we standardise the distribution so that the two inflection points are located at $x = -1$ and $x = 1$.



Where does the normal distribution come from?

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

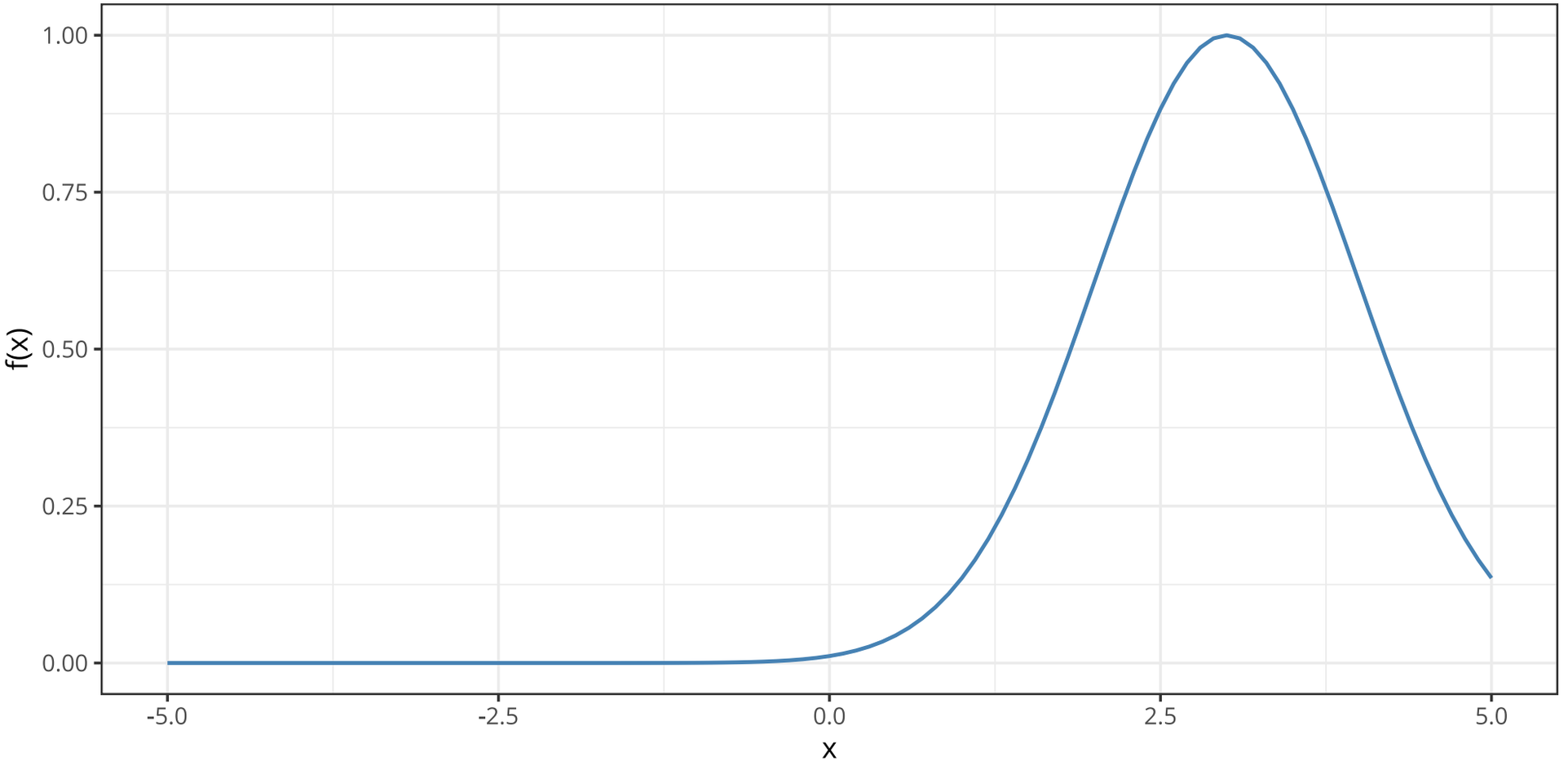
We then add a parameter σ^2 to control the distance between the inflection points.



Where does the normal distribution come from?

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

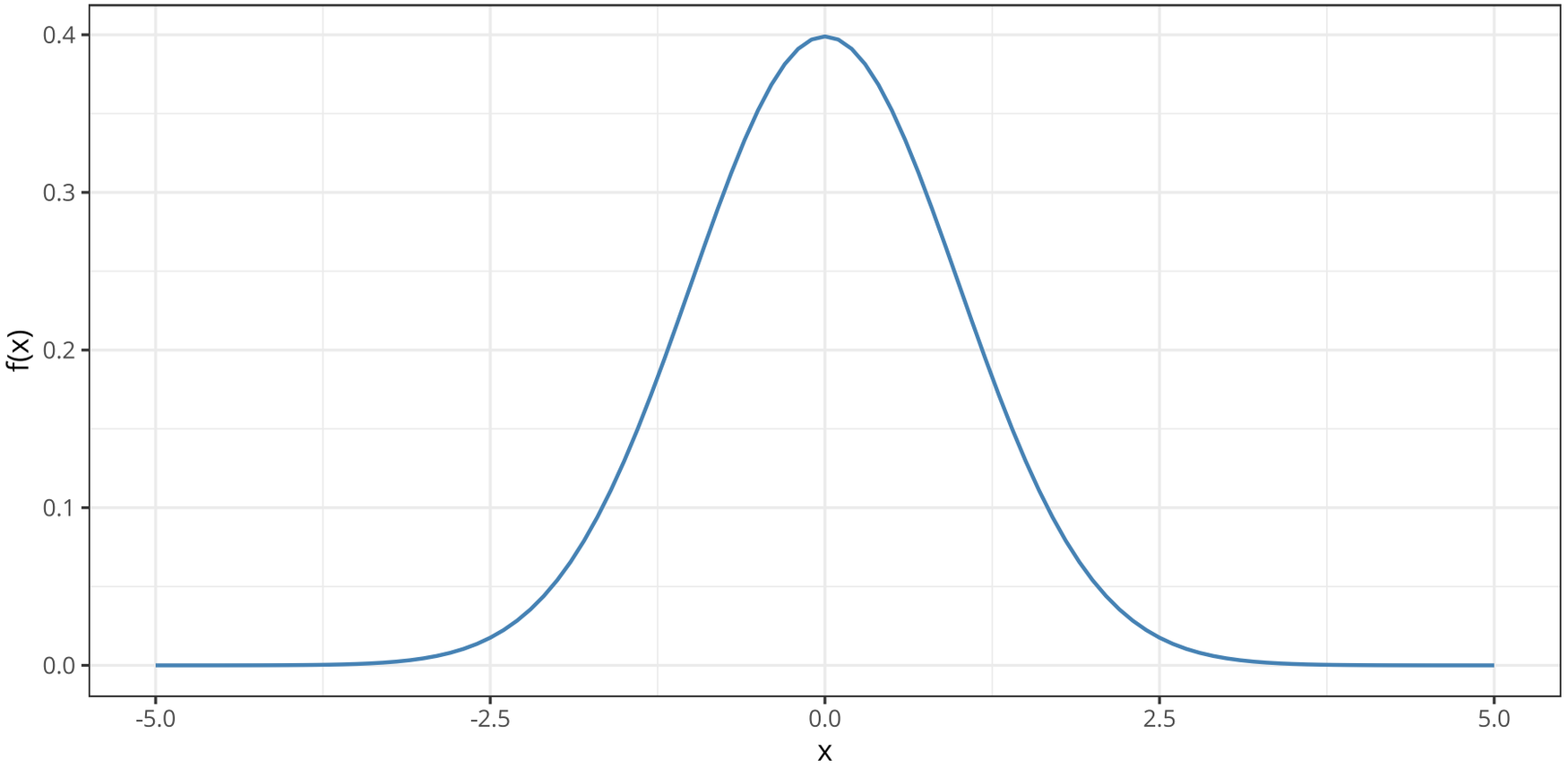
We then insert a parameter μ to control the position (central tendency) of the distribution.



Where does the normal distribution come from?

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

But... this distribution does not integrate to 1. We therefore divide by a normalisation constant (the left-hand side), to obtain a proper (valid) probability distribution.



Gaussian model

We are going to build a regression model, but before we add a predictor, let's try to model the distribution of heights.

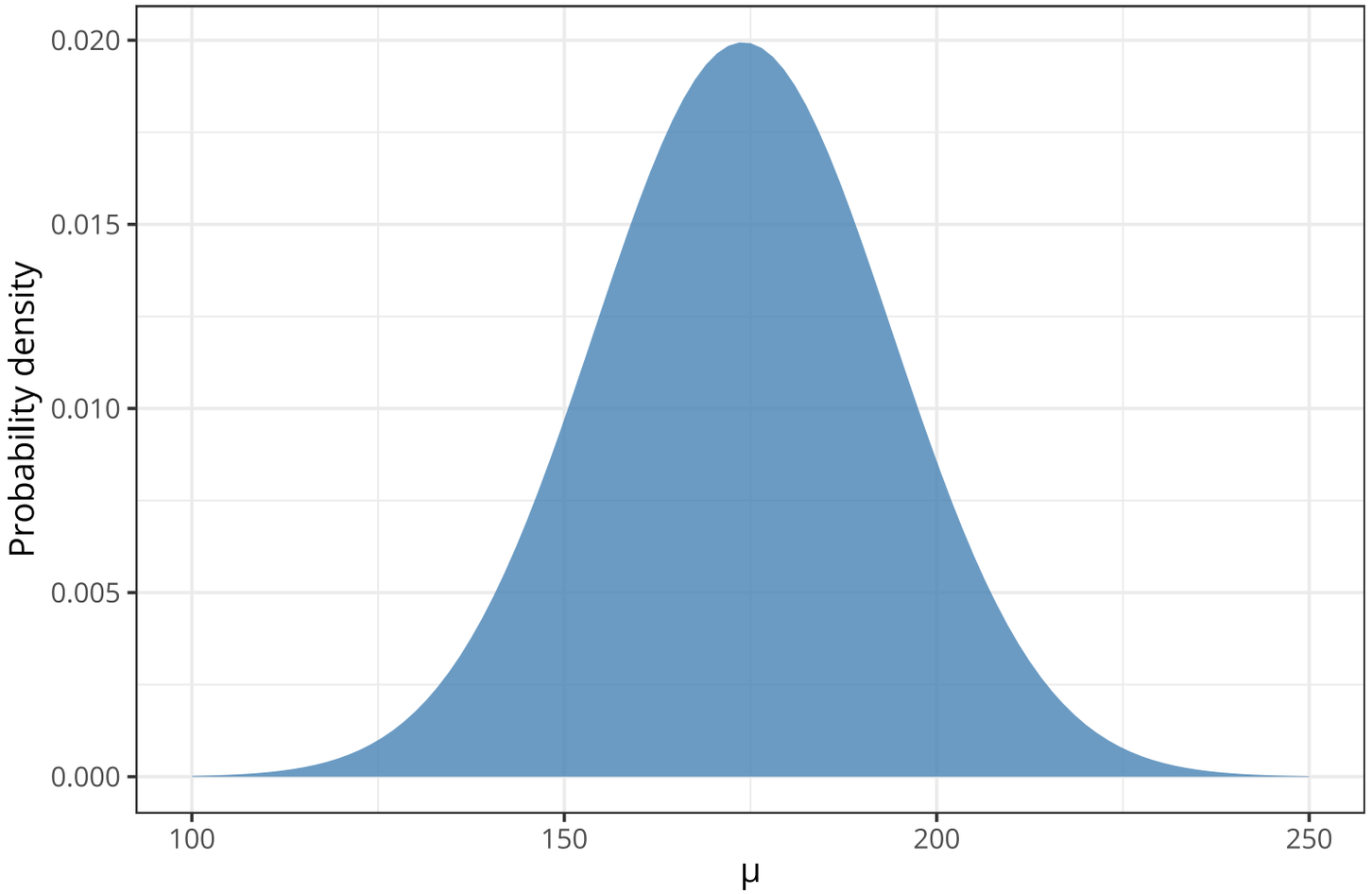
We want to know which model (distribution) best describes the distribution of heights. We will therefore explore all the possible combinations of μ and σ and rank them by their respective probabilities.

Our aim, once again, is to describe **the posterior distribution**, which will therefore be in some way **a distribution of distributions**.

Gaussian model

We define $p(\mu, \sigma)$, the joint prior distribution of all model parameters. These priors can be specified independently for each parameter, given that $p(\mu, \sigma) = p(\mu)p(\sigma)$.

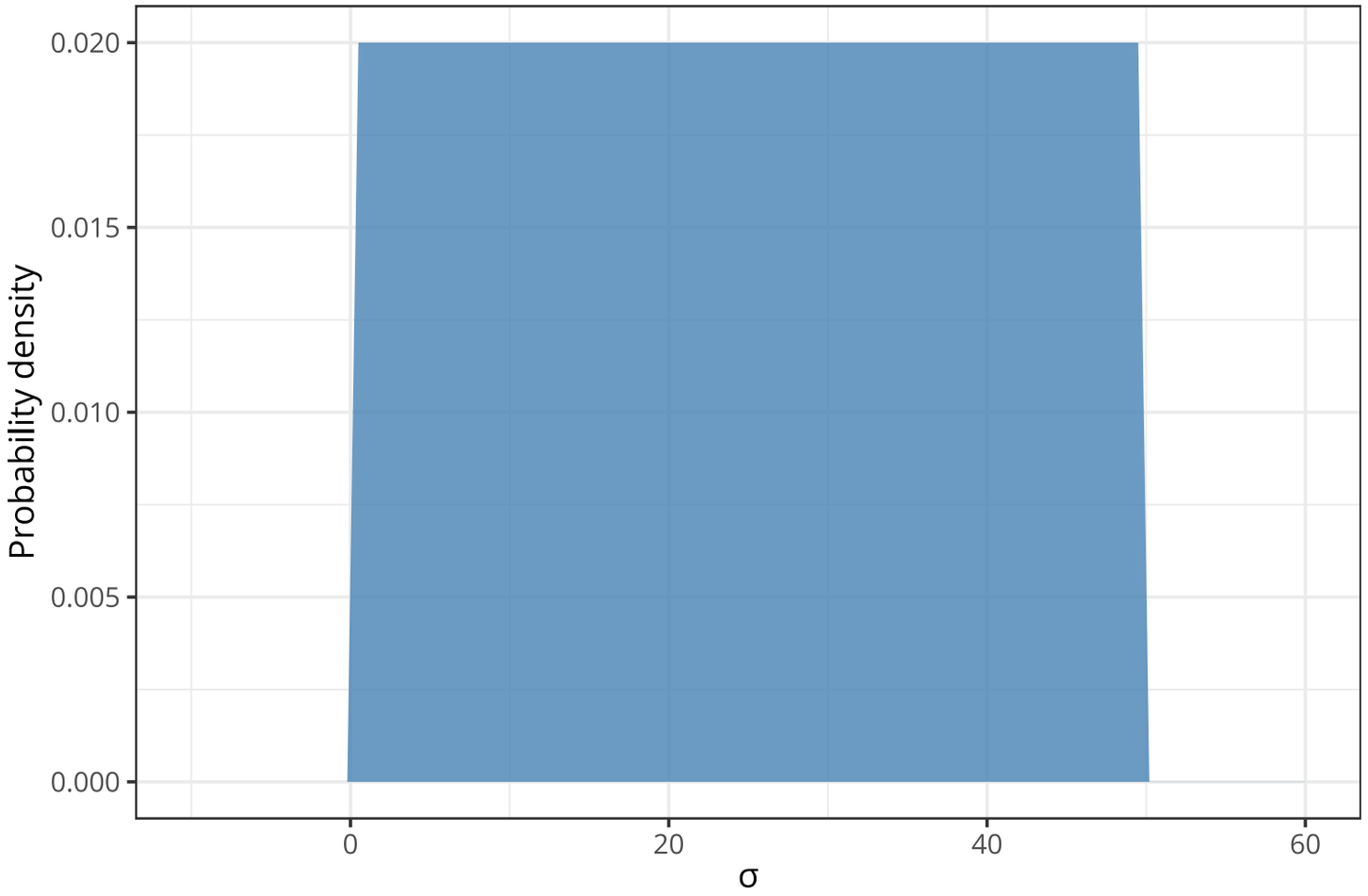
$$\mu \sim \text{Normal}(174, 20)$$



Gaussian model

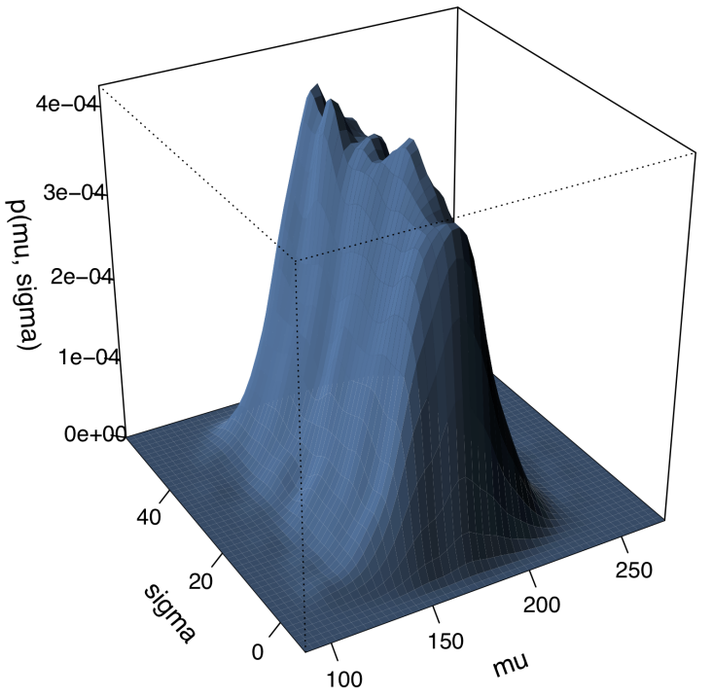
We define $p(\mu, \sigma)$, the joint prior distribution of all model parameters. These priors can be specified independently for each parameter, given that $p(\mu, \sigma) = p(\mu)p(\sigma)$.

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



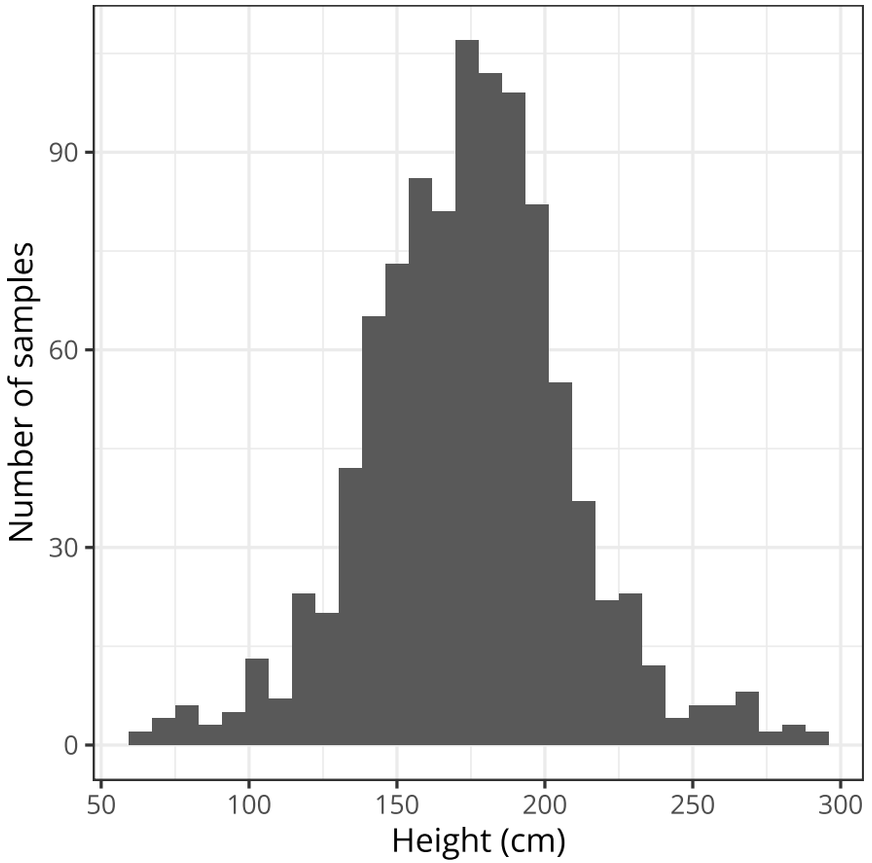
Visualiser le prior

```r
library(ks)
sample_mu <- rnorm(1e4, 174, 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

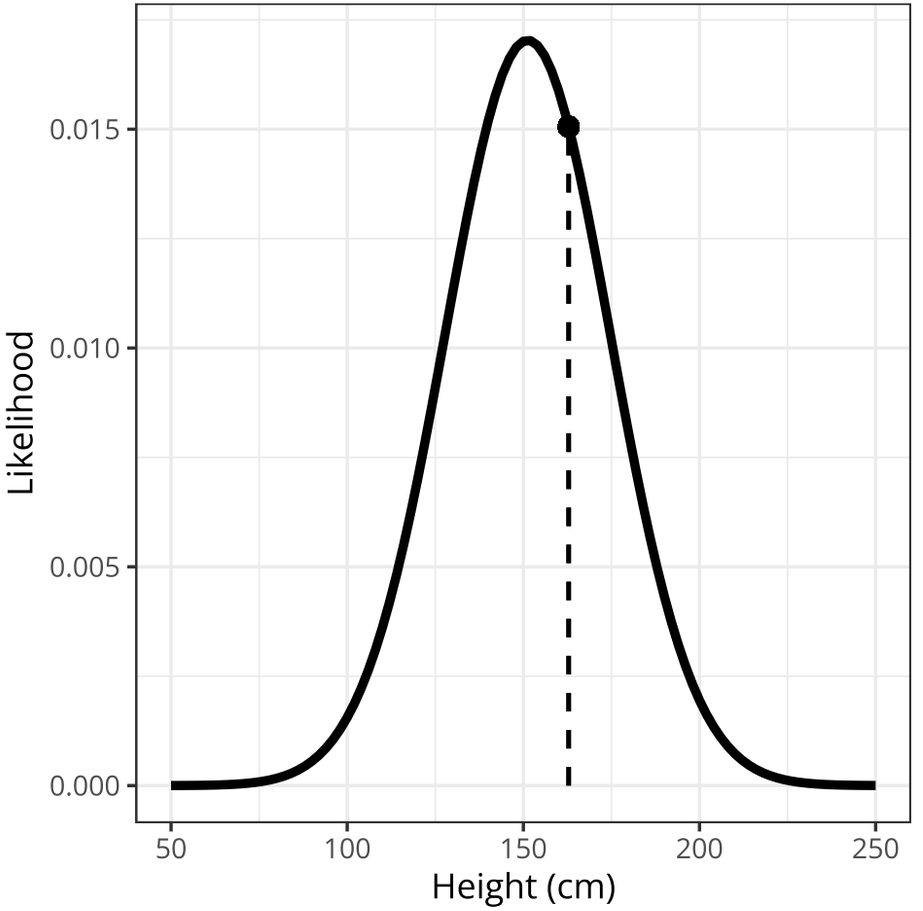
```
sample_mu <- rnorm(1000, 174, 20)
sample_sigma <- runif(1000, 0, 50)

data.frame(x = rnorm(n = 1000, mean = sample_mu, sd = sample_sigma) ) %>%
    ggplot(aes(x) ) +
    geom_histogram() +
    labs(x = "Height (cm)", y = "Number of samples")
```



Likelihood function

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

d2$height[34] # exemplary height observation
```

```
[1] 162.8648
```



Likelihood function

We want to compute the probability of observing a certain value of height knowing certain values of μ and σ, that is:

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

This **probability density** can be computed using the functions `dnorm`, `dbeta`, `dt`, `dexp`, `dgamma`, etc.

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

```
[1] 0.01505675
```


Likelihood function

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

Or using a custom function...

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

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

  return(bell / norm)

}
```

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

```
[1] 0.01505675
```

Posterior distribution

$$p(\mu, \sigma \mid h) = \frac{\prod_i \text{Normal}(h_i \mid \mu, \sigma)\text{Normal}(\mu \mid 174, 20)\text{Uniform}(\sigma \mid 0, 50)}{\int \int \prod_i \text{Normal}(h_i \mid \mu, \sigma)\text{Normal}(\mu \mid 174, 20)\text{Uniform}(\sigma \mid 0, 50)\text{d}\mu\text{d}\sigma}$$

$$p(\mu, \sigma \mid h) \propto \prod_i \text{Normal}(h_i \mid \mu, \sigma)\text{Normal}(\mu \mid 174, 20)\text{Uniform}(\sigma \mid 0, 50)$$

This is the same formula from the previous course, but here considering that there are several observations of height (h_i) and two parameters to be estimated: μ and σ.

To compute the **marginal likelihood** (in green), we therefore need to integrate over two parameters: μ and σ. Here again we realise that the posterior is proportional to the product of the likelihood and the prior.

Posterior distribution - Grid approximation

```r
# we define a grid of possible values for mu and sigma
mu.list <- seq(from = 140, to = 160, length.out = 200)
sigma.list <- seq(from = 4, to = 9, length.out = 200)

# we extend this grid to all possible combinations of mu and sigma
post <- expand.grid(mu = mu.list, sigma = sigma.list)

# we compute the log-likelihood for each observation (under each combination of mu and sigma)
post$LL <-
  sapply(
    1:nrow(post),
    function(i) sum(dnorm(
      d2$height,
      mean = post$mu[i],
      sd = post$sigma[i],
      log = TRUE) )
    )

# we compute the (unnormalised) posterior distribution
post$prod <-
  post$LL +
  dnorm(x = post$mu, mean = 174, sd = 20, log = TRUE) +
  dunif(x = post$sigma, min = 0, max = 50, log = TRUE)


```

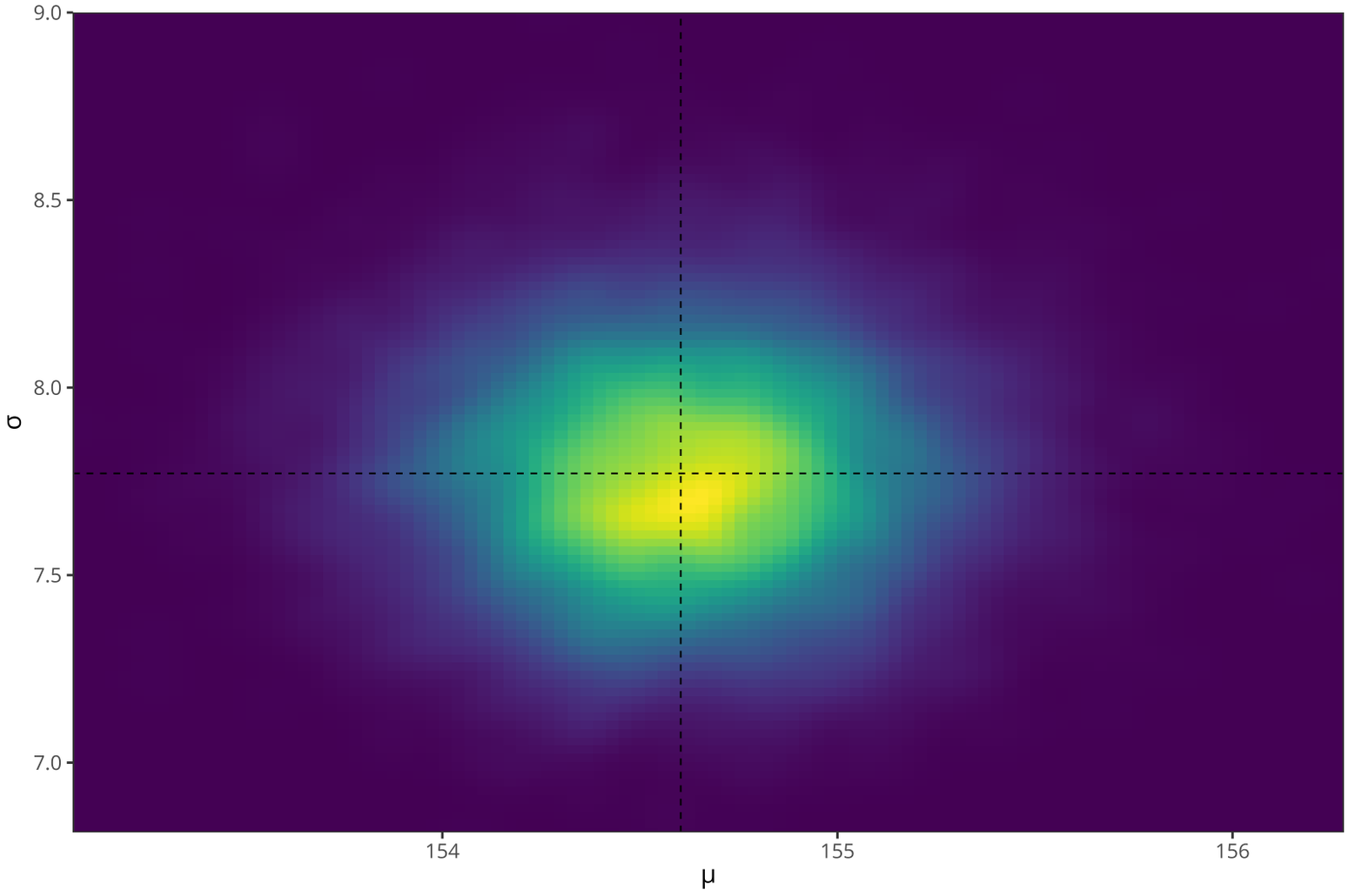

Posterior distribution - Grid approximation

```r
# randomly selecting 20 rows from the resulting dataframe
post %>% slice_sample(n = 20, replace = FALSE)
```

	mu	sigma	LL	prod	prob
1	155.1759	4.251256	-1418.233	-1426.502	4.612132e-87
2	143.2161	6.211055	-1829.978	-1838.989	3.334659e-266
3	156.6834	6.462312	-1250.548	-1258.749	3.297048e-14
4	155.8794	6.537688	-1237.281	-1245.518	1.837911e-08
5	141.0050	8.221106	-1701.767	-1710.955	1.342455e-210
6	159.0955	8.246231	-1273.181	-1281.286	5.378841e-24
7	142.6131	4.728643	-2471.268	-2480.326	0.000000e+00
8	146.3317	5.507538	-1667.237	-1676.021	1.993230e-195
9	160.0000	6.788945	-1337.375	-1345.446	7.347547e-52
10	143.4171	7.517588	-1608.947	-1617.943	3.329394e-170
11	140.6030	6.437186	-2064.584	-2073.805	0.000000e+00
12	151.6583	5.005025	-1370.982	-1379.432	1.276988e-66
13	156.2814	6.814070	-1236.277	-1244.496	5.108761e-08
14	153.7688	8.849246	-1226.826	-1235.164	5.767419e-04
15	144.1206	7.015075	-1615.495	-1624.437	5.033474e-173
16	143.7186	8.572864	-1506.316	-1515.289	1.271644e-125
17	154.8744	7.517588	-1219.926	-1228.210	6.039964e-01
18	146.3317	8.824121	-1379.469	-1388.252	1.886826e-70
19	140.0000	7.844221	-1828.939	-1838.211	7.264600e-266
20	159.8995	5.180905	-1478.782	-1486.857	2.832570e-113

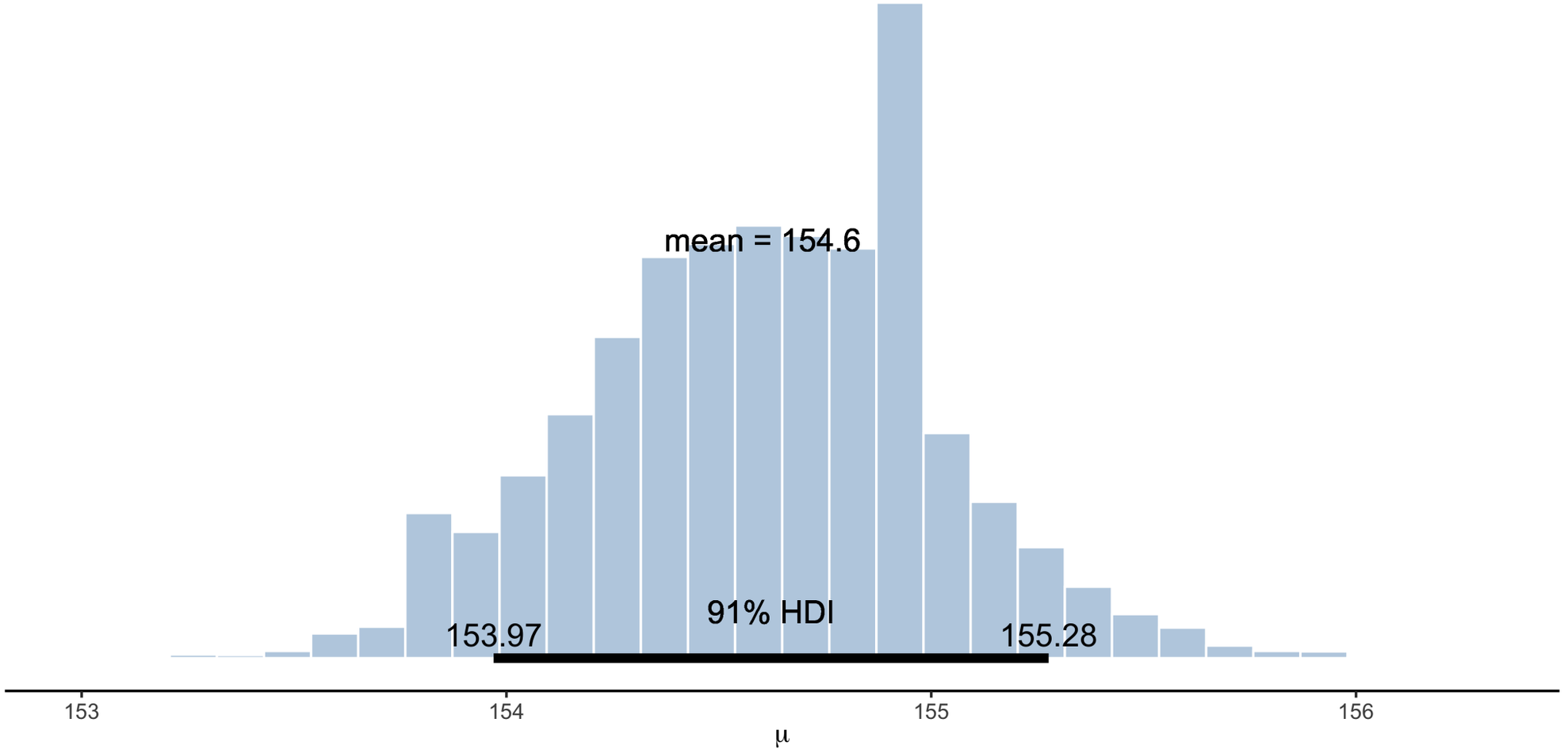
Posterior distribution - Grid approximation

```
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]
```



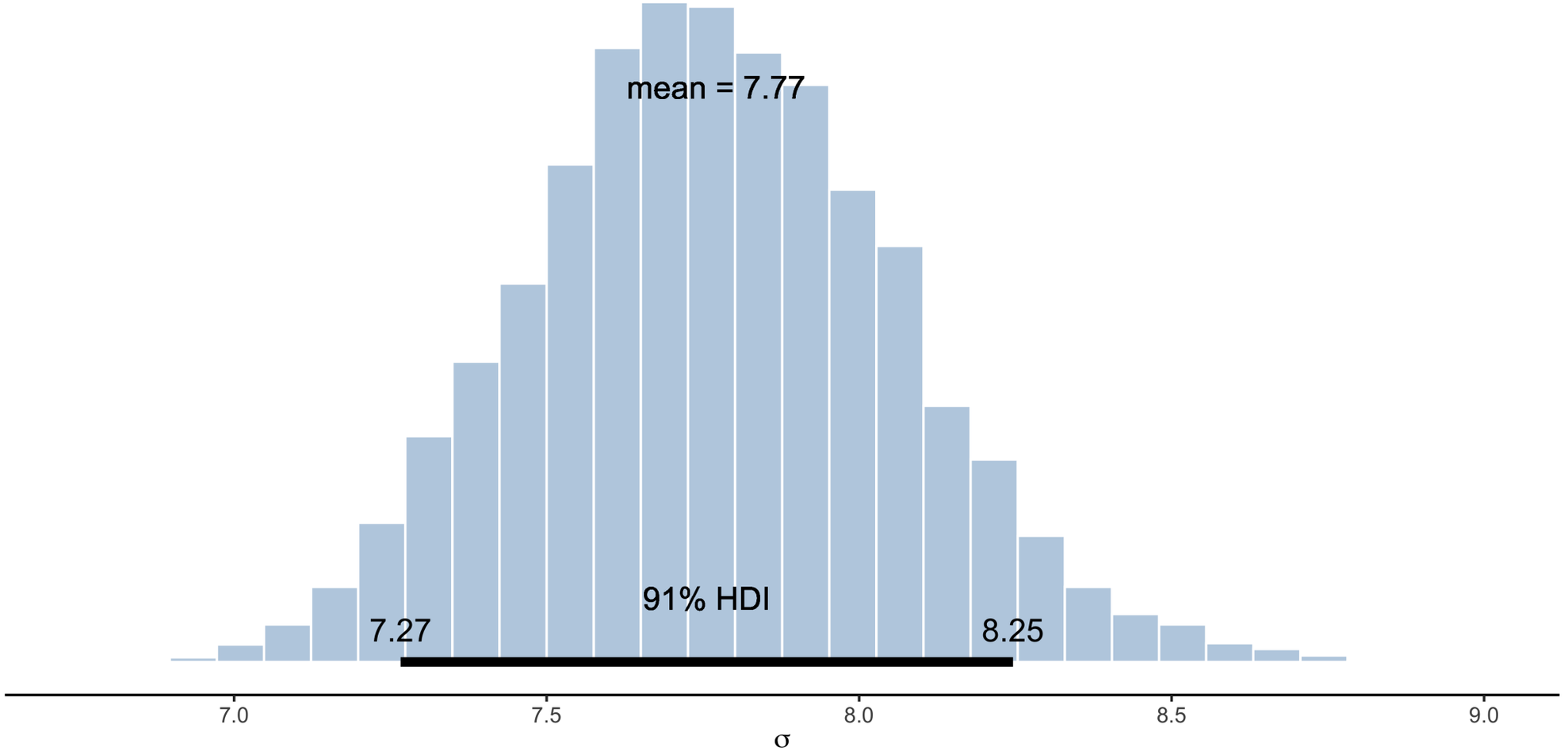
Posterior distribution - Marginal distributions

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



Posterior distribution - Marginal distributions

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



Introduction to brms

Introduction to brms

Under the hood: Stan is a probabilistic programming language written in C++, which implements several MCMC algorithms: 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

The `brms` package (Bürkner, 2017) can be used to fit multilevel (or single-level) linear (or not) Bayesian regression models in `Stan` but using the intuitive syntax of `lme4` (cf. our tutorial paper, Nalborczyk et al., 2019).

For instance, the following model:

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

is specified with `brms` (as with `lme4`) as follows:

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

Syntax reminders

The `brms` package uses the same syntax as the base `R` functions (such as `lm`) or functions from the `lme4` package.

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

The left-hand side represents our dependent variable (or outcome, that is, what we are trying to predict). The `brms` package can also be used to fit multivariate models (several outcomes) by combining them with with `mvbind()`.

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

The right-hand side is used to define the predictors. The intercept is is generally implicit, so that the two formulations below are equivalent.

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


Syntax reminders

If you want to fit a model without an intercept (because why not), you must specify it explicitly, as shown below.

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

By default `brms` assumes a Gaussian likelihood function. This assumption can be modified easily by specifying the desired likelihood via the the `family` argument.

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

In case of doubt, read the documentation (it's very exciting to read) available at `?brm`.

Some useful functions

```r
# retrieving the Stan code generated by brms
make_stancode(formula, ...)
stancode(fit)

# checking and/or defining priors
get_prior(formula, ...)
set_prior(prior, ...)

# retrieving predictions from a brms model
fitted(fit, ...)
predict(fit, ...)
conditional_effects(fit, ...)

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

# model comparison and hypothesis testing
loo(fit1, fit2, ...)
bayes_factor(fit1, fit2, ...)
model_weights(fit1, fit2, ...)
hypothesis(fit, hypothesis, ...)
```


A first example

```
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 154.614878 0.4224360 153.787631 155.452195
sigma         7.763965 0.2936431   7.217446   8.354354
```

These data represent the marginal (posterior) distributions of each parameter. In other words, the posterior probability μ, averaged over all possible values of $sigma$, is best described by a Gaussian distribution with a mean of 154.6 and a standard deviation of 0.42. The credibility interval ($\neq$ confidence interval) indicates the 95% most plausible values of μ or σ (given the data and priors).

Using our prior

By default `brms` uses a very uninformative prior centred on the mean value of the measured variable. We can therefore refine the estimate using our knowledge of the usual distribution of heights in humans.

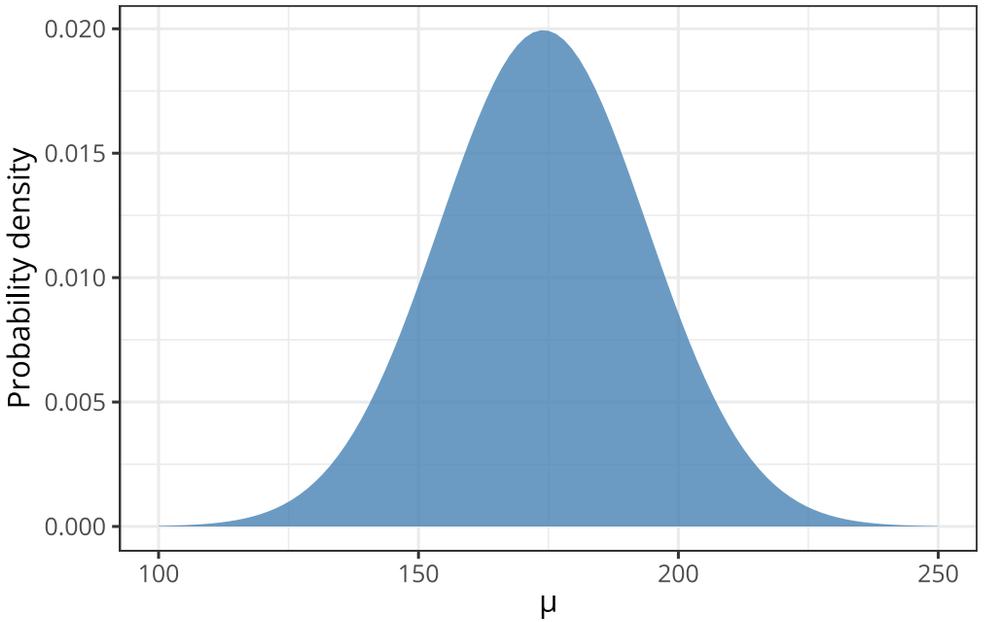
The `get_prior()` function is used to display a list of default priors as well as all the priors we can specify, given a certain model formula (i.e., a way of writing our model) and a set of data.

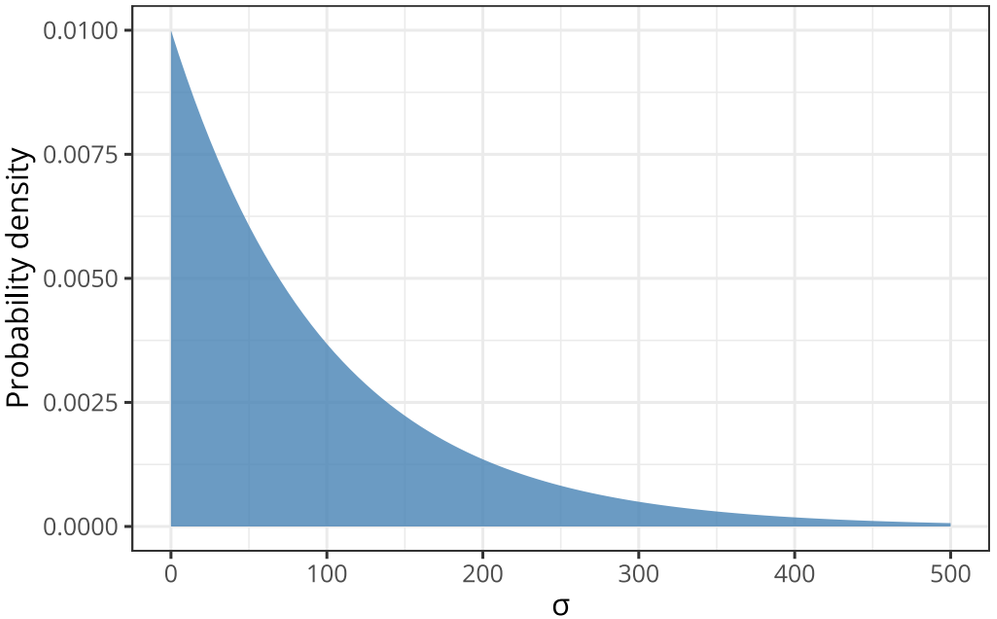
```
1  get_prior(height ~ 1, data = d2)
```

```
                  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
```

Using our prior

```
priors <- c(
  prior(normal(174, 20), class = Intercept),
  prior(exponential(0.01), class = sigma)
  )

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





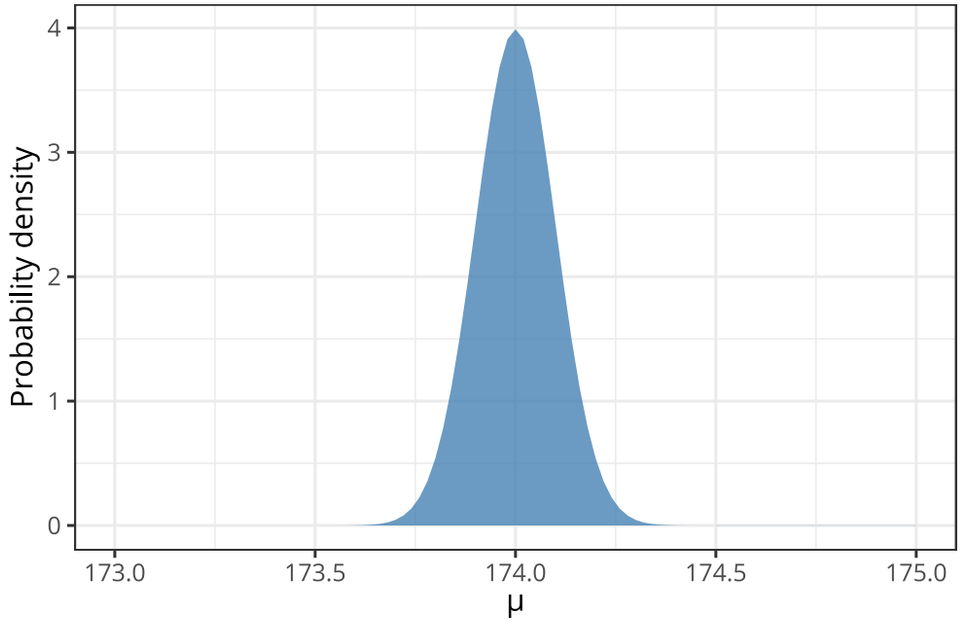
Using our prior

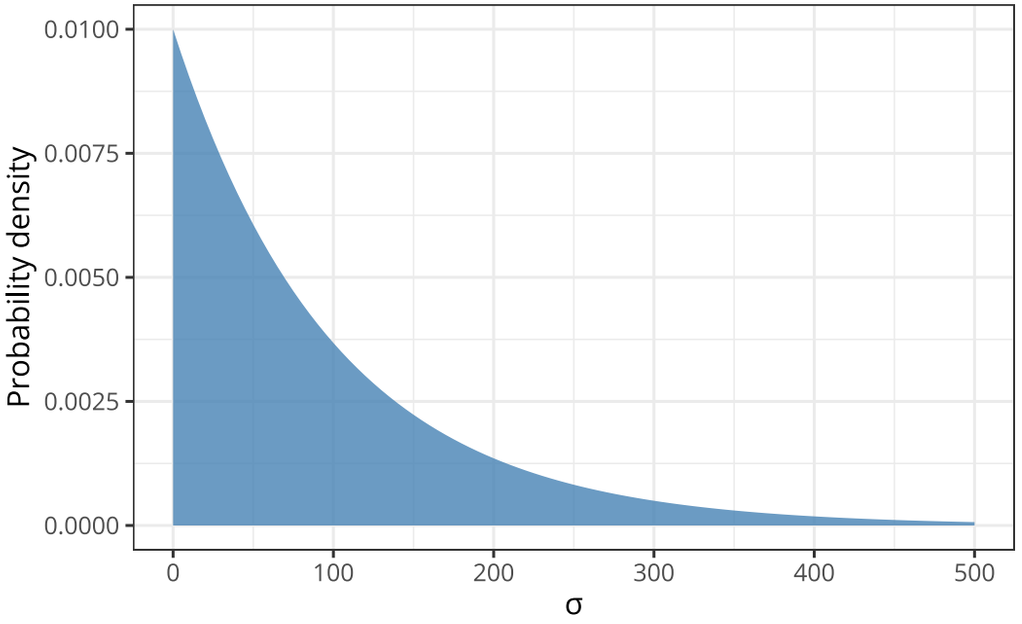
```
1 summary(mod2)
```

```
 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   154.60      0.41   153.80   155.42 1.00     3433     2656

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     7.76      0.30     7.23     8.39 1.00     3032     2499

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).
```


Using a more informative prior

```
priors <- c(
  prior(normal(174, 0.1), class = Intercept),
  prior(exponential(0.01), class = sigma)
  )

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





Using a more informative prior

```
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   173.84      0.10   173.65   174.03 1.00     3108     2389

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma    20.81      0.79    19.33    22.48 1.00     2908     2312

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).
```

We note that the estimated value for μ has hardly "moved" from the prior...but we can also see that the estimated value for σ has greatly increased. What happened? We told the model that we were fairly certain of our μ value, the model then "adapted", which explains the large value of σ...

Prior precision (heuristic)

Prior distributions can generally be considered as posterior distributions obtained from previous data.

We know that the σ of a Gaussian posterior is given by:

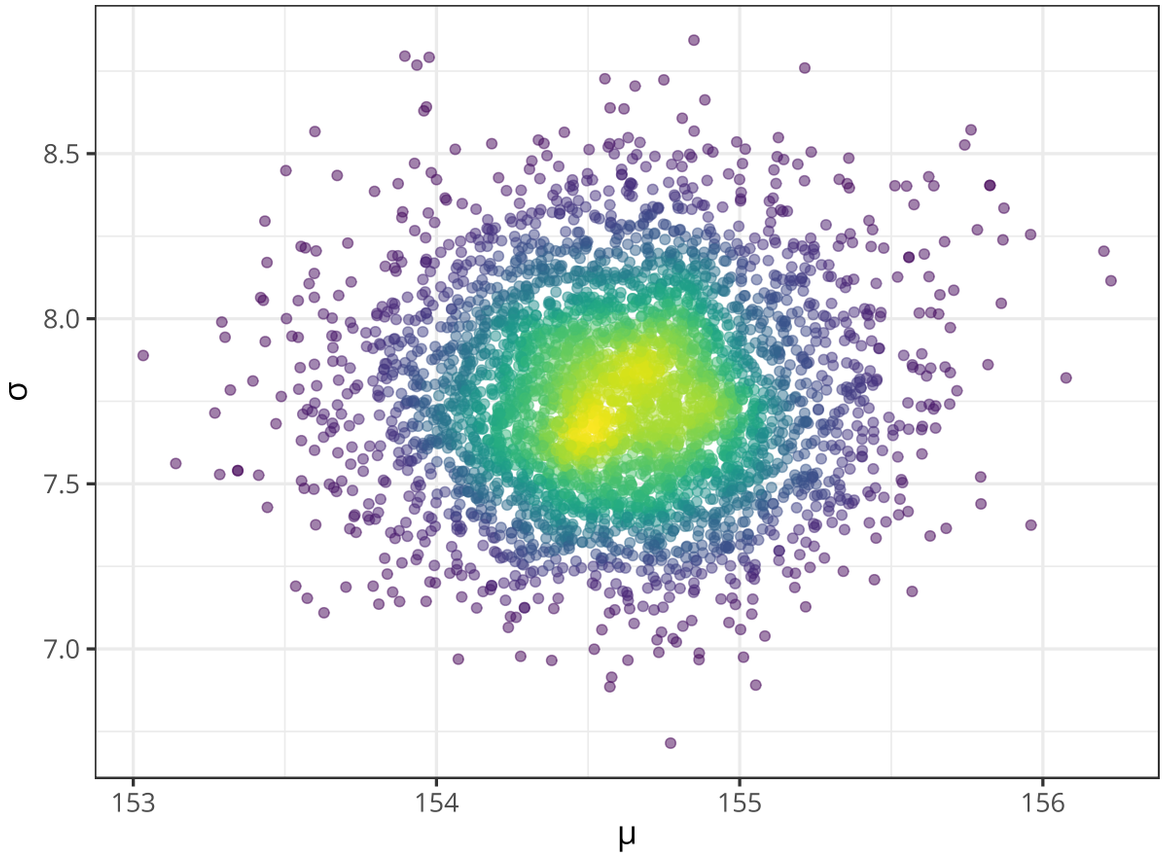
$$\sigma_{\text{post}} = 1 \,/\, \sqrt{n}$$

Which implies a **quantity of data** $n = 1 \,/ \sigma^2_{\text{post}}$. Our prior had a $\sigma = 0.1$, which implies $n = 1 \,/\, 0.1^2 = 100$.

We can therefore consider that the prior $\text{Normal}(174, 0.1)$ is equivalent to the case in which we would have observed 100 heights with mean 174.

Visualising samples from the posterior distribution

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

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



Retrieving samples from the posterior distribution

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

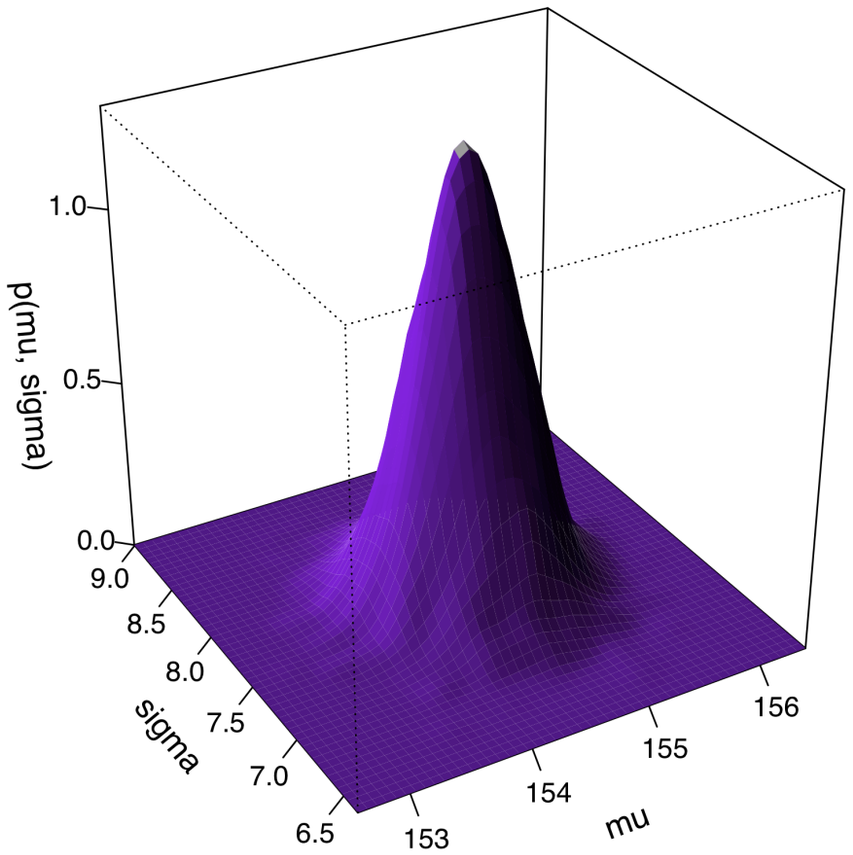
```
# A draws_df: 6 iterations, 1 chains, and 5 variables
  b_Intercept sigma lprior  lp__ density
1         154   8.2   -9.1 -1229   0.082
2         155   7.4   -9.0 -1229   0.150
3         155   7.2   -9.0 -1229   0.139
4         154   8.0   -9.1 -1227   0.582
5         155   8.3   -9.1 -1228   0.263
6         155   7.6   -9.0 -1228   0.215
# ... hidden reserved variables {'.chain', '.iteration', '.draw'}
```

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

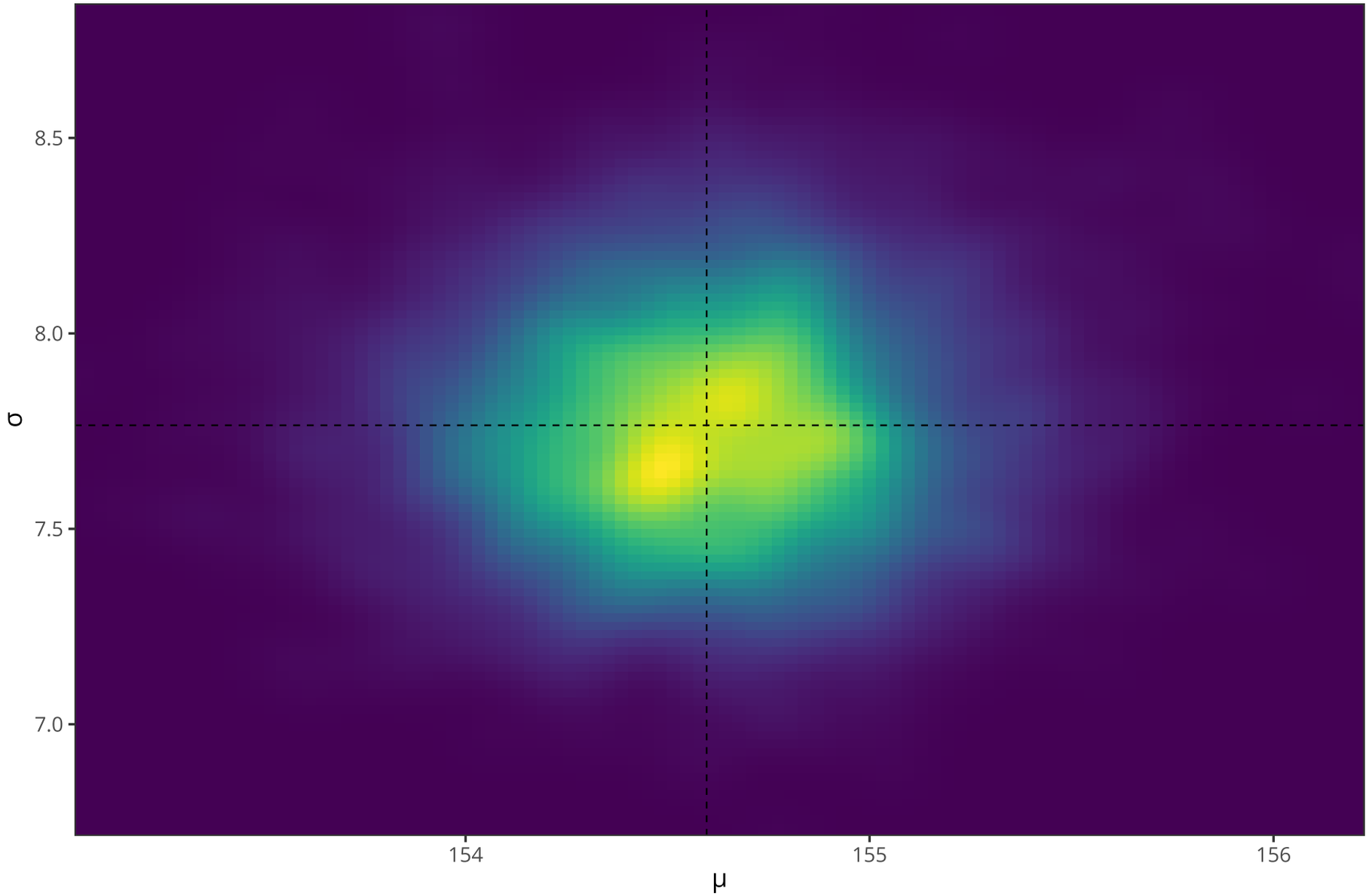
```
                2.5%       50%      97.5%
b_Intercept 153.795472 154.59952 155.418792
sigma         7.227358   7.74985   8.386699
```

Visualising the posterior distribution

```
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")
```



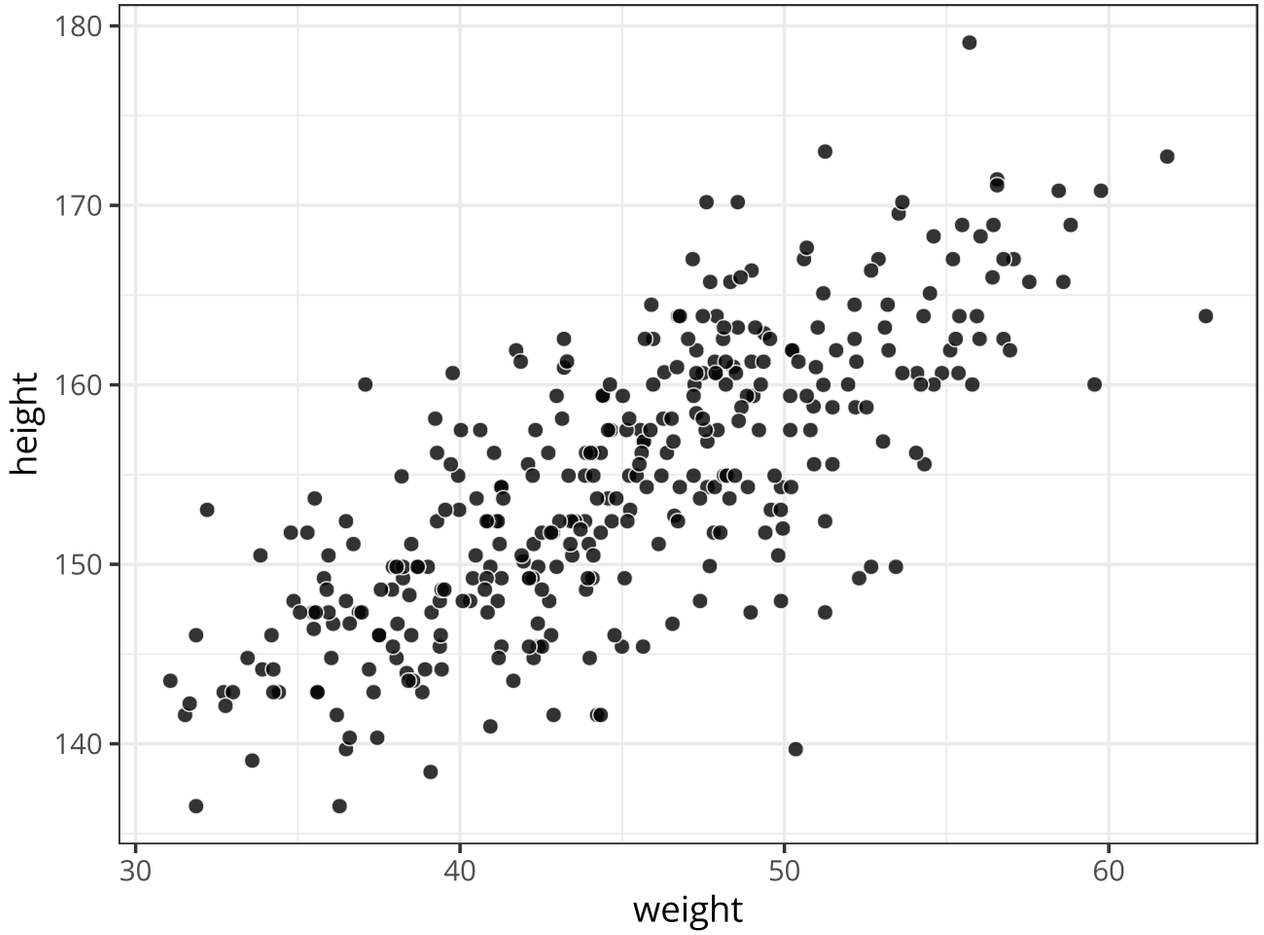
Visualising the posterior distribution



Including predictors

How does height change with weight?

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

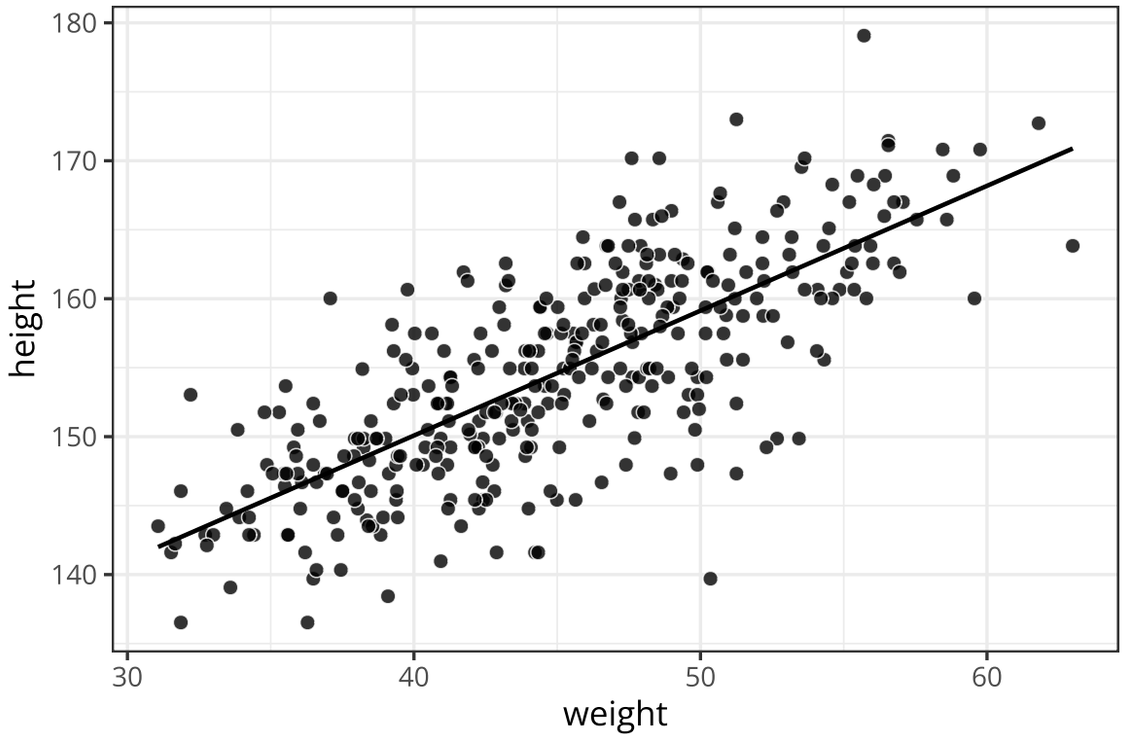


Linear regresion with a single continuous predictor

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

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



Notations

Let's consider a linear regression model with a single predictor, a slope, an intercept, and residuals distributed according to a normal distribution. The notation:

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

is equivalent to:

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

rearranging the expression:

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

The above notations are equivalent, but the last one is more flexible, and will allow us to extend it more intuitively to multilevel models.

Linear regresion with a single continuous predictor

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta x_i \\
\alpha &\sim \text{Normal}(174, 20) \\
\beta &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{Exponential}(0.01)
\end{aligned}
$$

In this model, μ is no longer a parameter to be estimated (because μ is **determined** by α and β). Instead, we will estimate α and β.

Reminders: α is the intercept, that is, the expected height, when the weight is equal to 0. β is the slope, that is, the expected change in height when weight increases by one unit.

Linear regresion with a single continuous predictor

```
priors <- c(
  prior(normal(174, 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
  )
```


Linear regresion with a single continuous predictor

```
posterior_summary(mod4)
```

	Estimate	Est.Error	Q2.5	Q97.5
b_Intercept	113.9022663	1.89453254	110.2751443	117.5863999
b_weight	0.9045325	0.04172372	0.8226575	0.9834808
sigma	5.1023966	0.19607421	4.7309481	5.4967173
lprior	-12.2671372	0.01272490	-12.2925817	-12.2432602
lp__	-1083.1151428	1.14546650	-1086.1172584	-1081.7573733

- $\alpha = 113.90, 95\%$ CrI $[110.28, 117.59]$ represents the expected height when weight equals 0kg...
- $\beta = 0.90, 95\%$ CrI $[0.82, 0.98]$ indicates that for an increase of 1kg, we can expect an increase of 0.90cm.

Linear regresion with a single continuous predictor

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

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

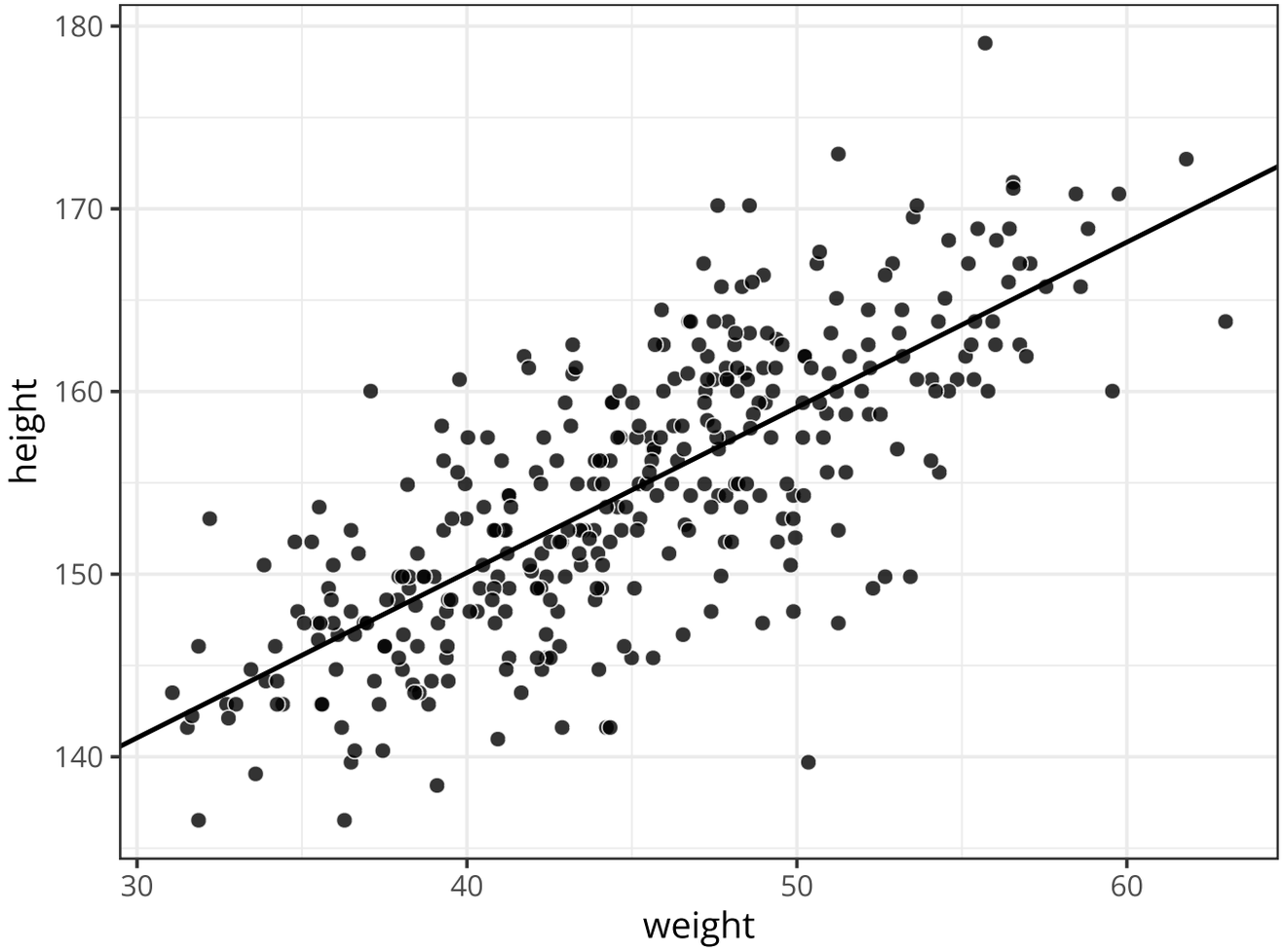
```r
fixef(mod5) # retrieving the fixed effects estimates
```

```
            Estimate  Est.Error        Q2.5       Q97.5
Intercept 154.5892334 0.26511032 154.0580097 155.1021441
weight.c    0.9048262 0.04247021   0.8194467   0.9874537
```

If the predictor (weight) is centred, the intercept represents the expected value of height (in cm) when weight is at its mean value.

Visualising predictions from the model

```
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)
```



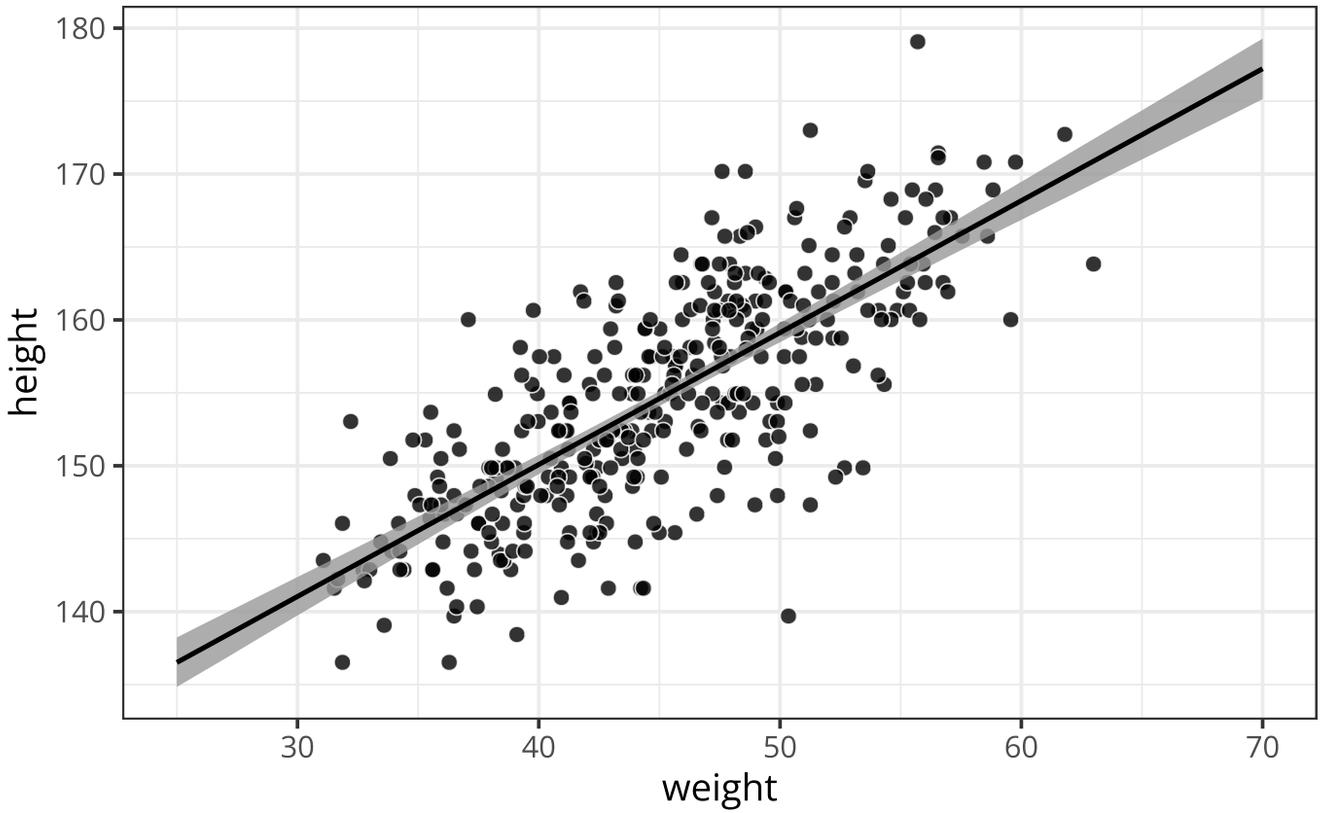
Visualising the uncertainty on μ using fitted()

```r
# defining a grid of possible values for "weight"
weight.seq <- data.frame(weight = seq(from = 25, to = 70, by = 1) )

# retrieving the model's predicted mus (average height) for these values of "weight"
mu <- data.frame(fitted(mod4, newdata = weight.seq) ) %>% bind_cols(weight.seq)

# displaying the first 10 rows of mu
head(mu, 10)
```

```
   Estimate Est.Error     Q2.5    Q97.5 weight
1  136.5156 0.8729446 134.8388 138.2282     25
2  137.4201 0.8331902 135.8118 139.0699     26
3  138.3246 0.7936381 136.7802 139.8924     27
4  139.2292 0.7543200 137.7739 140.7197     28
5  140.1337 0.7152745 138.7559 141.5451     29
6  141.0382 0.6765488 139.7387 142.3767     30
7  141.9428 0.6382011 140.7096 143.1942     31
8  142.8473 0.6003039 141.6932 144.0216     32
9  143.7518 0.5629482 142.6682 144.8488     33
10 144.6564 0.5262493 143.6445 145.6845     34
```


Visualising the uncertainty on μ using fitted()

```r
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
    )
```



Prediction intervals (incorporating σ)

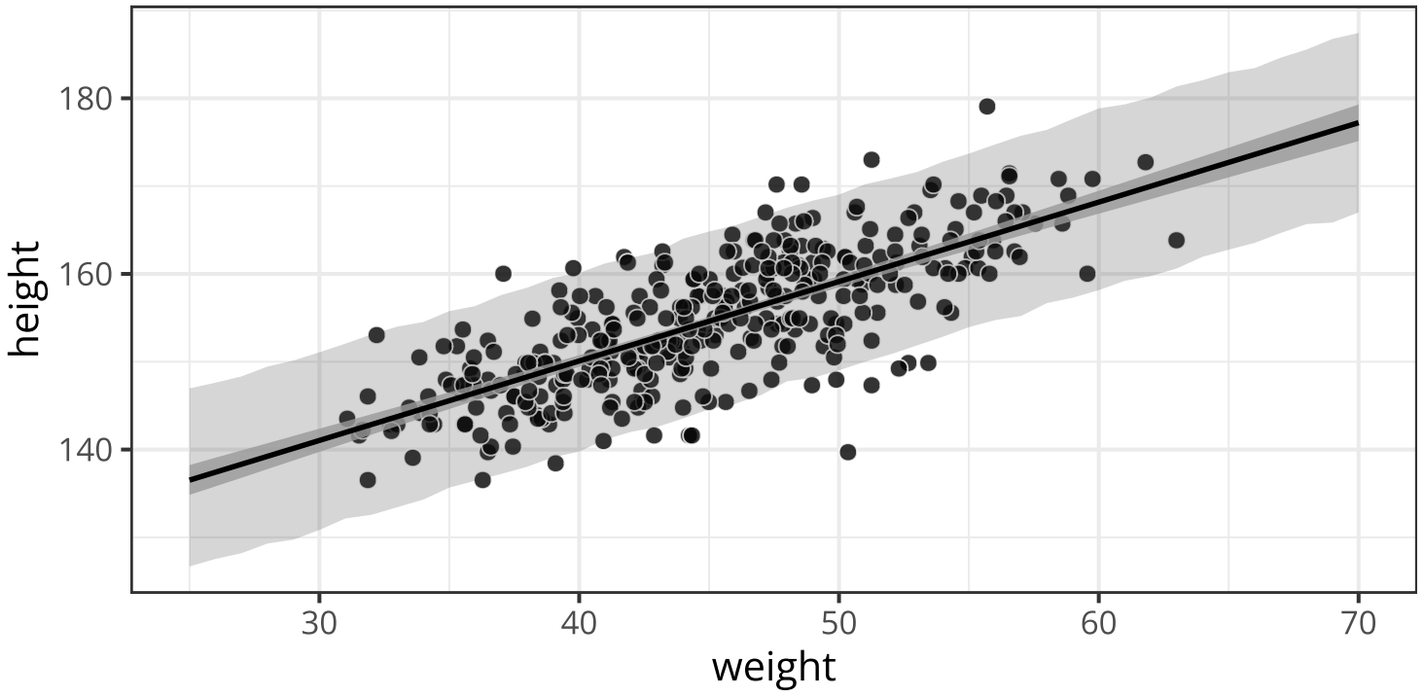
As a reminder, our model is defined as: $h_i \sim \text{Normal}(\alpha + \beta x_i, \sigma)$. For the moment, we have only represented the predictions for μ. How can we incorporate σ into our predictions?

```r
# defining a grid of possible values for "weight"
weight.seq <- data.frame(weight = seq(from = 25, to = 70, by = 1) )

# retrieving the model's predicted heights for these values of "weight"
pred_height <- data.frame(predict(mod4, newdata = weight.seq) ) %>% bind_cols(weight.seq)

# displaying the first 10 rows of pred_height
head(pred_height, 10)
```

```
   Estimate Est.Error     Q2.5    Q97.5 weight
1  136.6003  5.241255 126.6961 146.9625     25
2  137.5269  5.123556 127.5401 147.6144     26
3  138.2405  5.176174 128.1914 148.3255     27
4  139.3099  5.243116 129.3009 149.4473     28
5  140.0189  5.168394 129.7397 150.1358     29
6  140.8688  5.218597 130.8407 151.0752     30
7  142.0739  5.152610 132.1567 152.0562     31
8  142.7756  5.218983 132.5639 153.1091     32
9  143.7965  5.244761 133.4364 154.0000     33
10 144.6657  5.198395 134.3009 154.5210     34
```

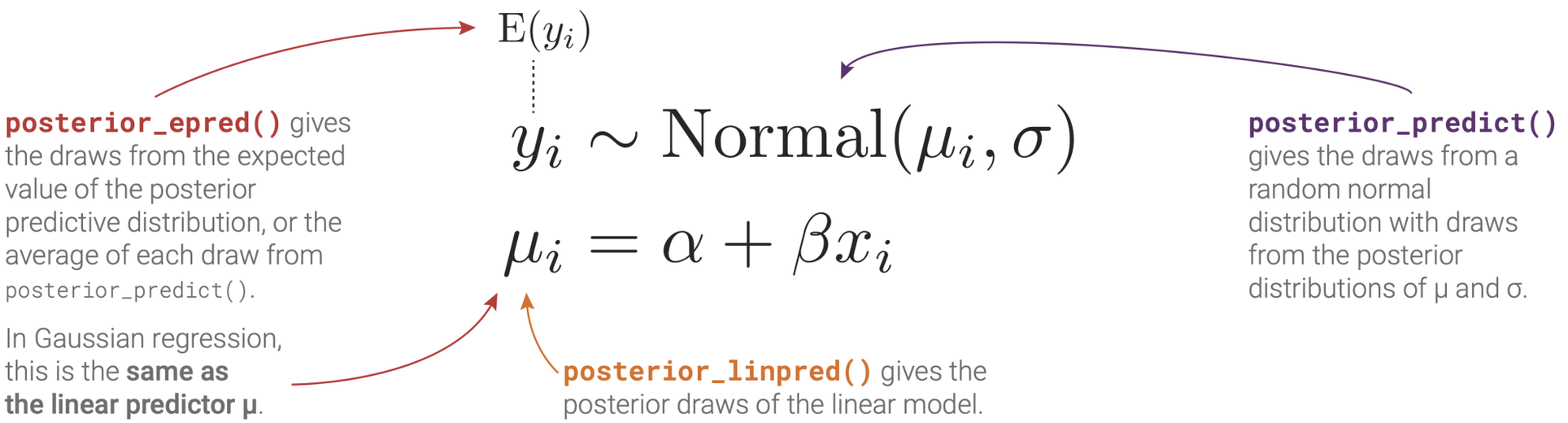

Prediction intervals (incorporating σ)

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



Built-in brms functions

The `brms` package also includes the `posterior_epred()`, `posterior_linpred()`, and `posterior_predict()` functions, which can be used to generate predictions from models fitted with `brms`. Andrew Heiss gives a detailed description of how these functions work in this blog post.



Reminder: Two types of uncertainty

Two sources of uncertainty in the model: uncertainty concerning the estimation of the value of the parameters but also uncertainty concerning the sampling process.

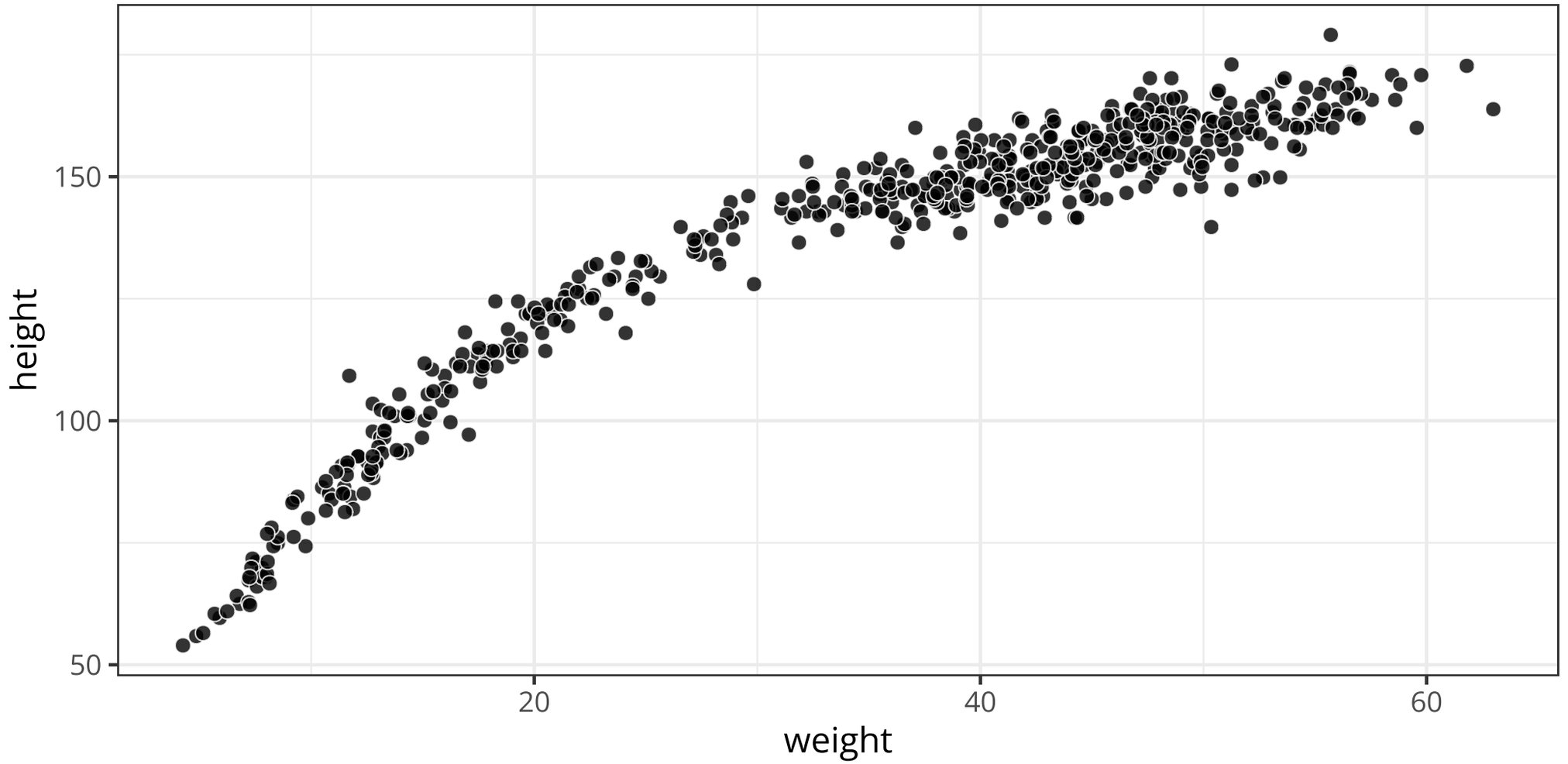
Epistemic uncertainty: The posterior distribution orders all possible combinations of parameter values according to their relative plausibility.

Aleatory uncertainty : The distribution of simulated data is a distribution that contains uncertainty related to a sampling process (i.e., generating data from a Gaussian distribution).

See also this short article by O'Hagan (2004).

Polynomial regression

```
d %>% # note that we use d instead of d2
  ggplot(aes(x = weight, y = height) ) +
  geom_point(colour = "white", fill = "black", pch = 21, size = 3, alpha = 0.8)
```

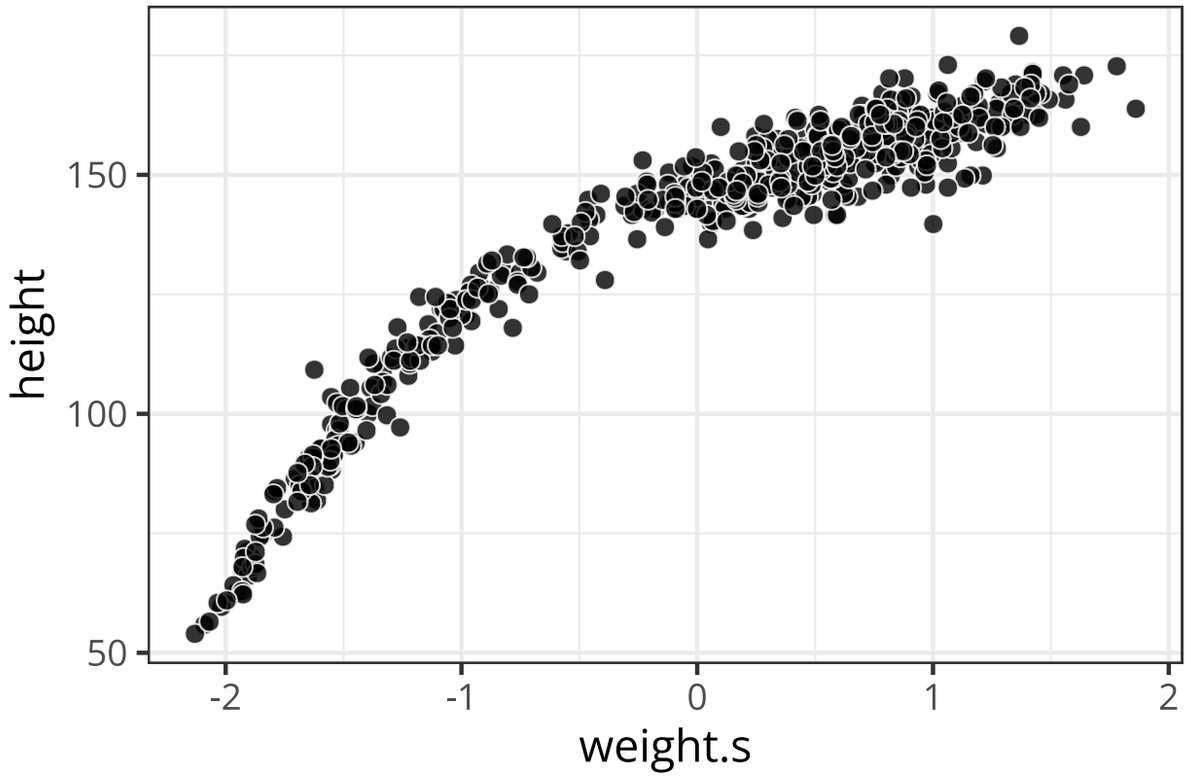


Standardised (scaled) scores

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

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

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

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



Standardised (scaled) scores

Why standardising predictors?

- **Interpretation**. Easier to compare the coefficients of several predictors. A change of one standard deviation in the predictor corresponds to to a change of one standard deviation in the response (if the response is also standardised).
- **Fitting**. When the predictors contain large values (or values that are too different from one another), this can lead to convergence problems (cf. Course n°03).

Polynomial regression - Exercise

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

It's up to you to build and fit this model using `brms::brm()`.

Polynomial regression

```
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
  )
```


Polynomial regression

```
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     146.65      0.37   145.95   147.39 1.00     3438     2694
weight.s       21.40      0.29    20.84    21.99 1.00     3198     2387
Iweight.sE2    -8.41      0.28    -8.96    -7.85 1.00     3117     2845

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     5.78      0.17     5.46     6.15 1.00     3352     3090

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).
```

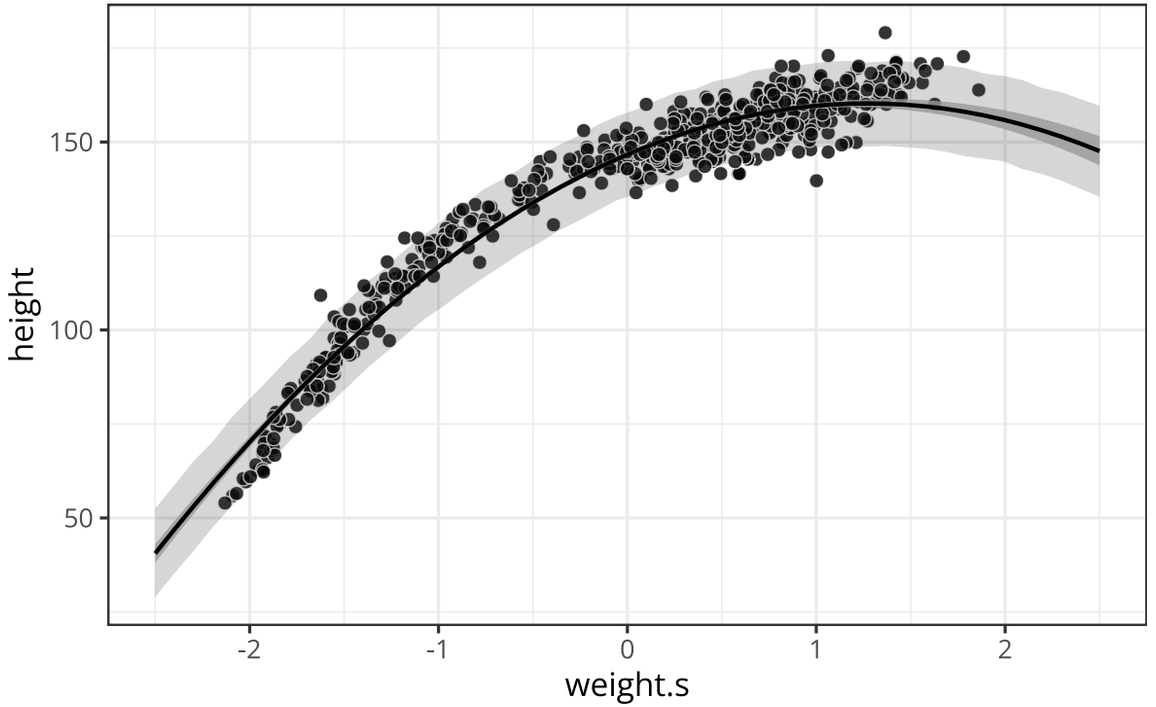

Visualising the model's predictions

```r
# defining a grid of possible (standardised) values for "weight"
weight.seq <- data.frame(weight.s = seq(from = -2.5, to = 2.5, length.out = 50) )

# retrieving the model's predictions for these values of weight
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)

# displaying the first ten rows of pred_height
head(pred_height, 10)
```

```
   Estimate Est.Error     Q2.5     Q97.5  weight.s
1  40.60063  6.010429 28.97013  52.51992 -2.500000
2  47.10254  5.955673 35.32591  58.77250 -2.397959
3  53.19186  6.006070 41.31088  65.14786 -2.295918
4  59.29052  5.760499 47.61468  70.36520 -2.193878
5  65.32630  5.963502 53.34686  76.94875 -2.091837
6  70.76771  5.769429 59.88322  82.18180 -1.989796
7  76.26657  5.815825 64.92816  87.47296 -1.887755
8  81.64095  5.787103 70.60726  92.95473 -1.785714
9  86.67635  5.665041 75.79001  97.45832 -1.683673
10 91.71143  5.880964 80.16754 103.33322 -1.581633
```


Visualising the model's predictions

```
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
    )
```



Effect sizes in regression models

There are several methods for computing effect sizes in Bayesian models. Gelman & Pardoe (2006) propose a method for computing a sample-based R^2.

Marsman & Wagenmakers (2017) and Marsman et al. (2019) generalise existing methods to compute a ρ^2 for ANOVA designs (i.e., with categorical predictors), which represents an estimate of the effect size in the population.

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

Effect sizes in regression models

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

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

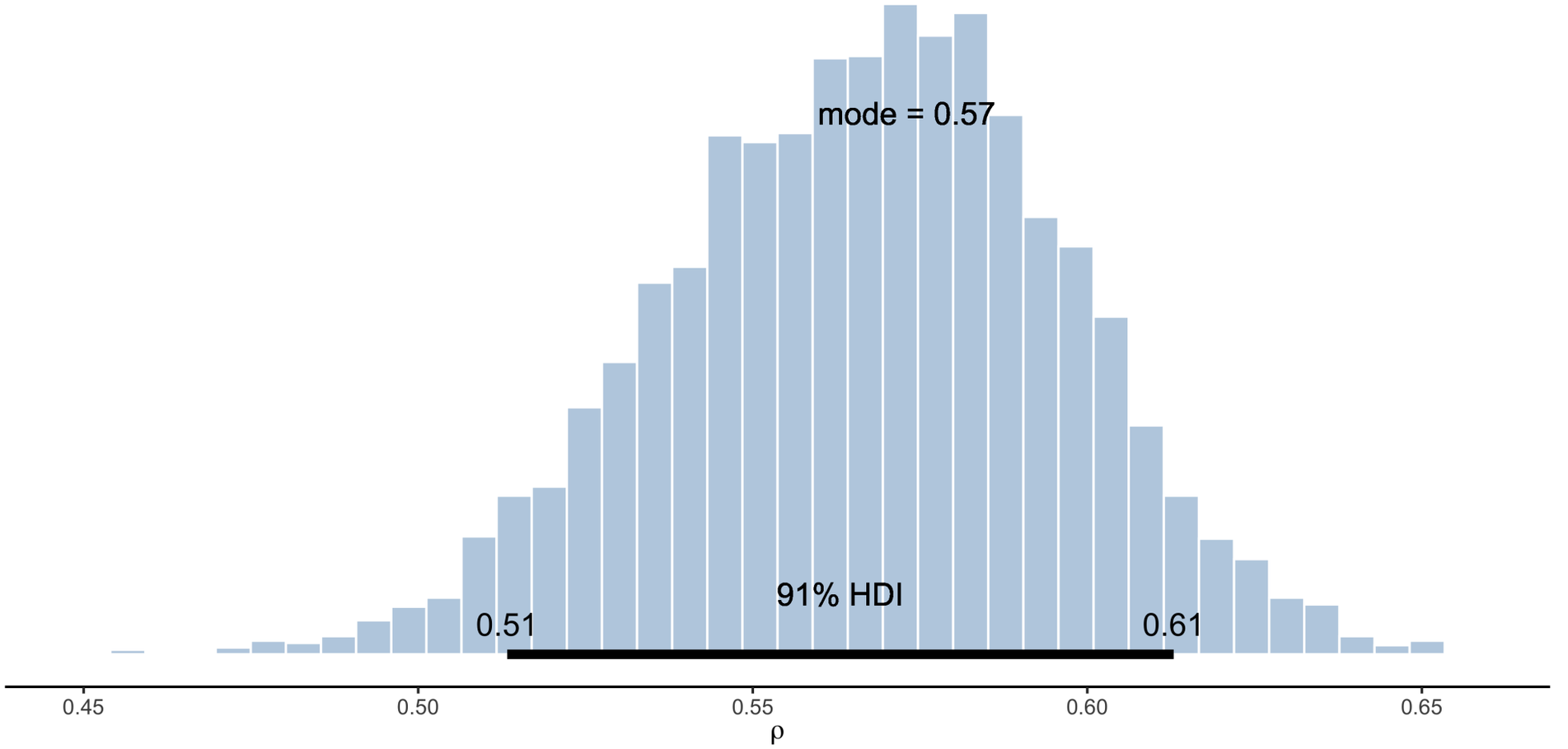
$$\rho^2 = \frac{\beta^2\tau^2}{\sigma^2 + \beta^2\tau^2}$$

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

Caution: if there are several predictors, it depends on the covariance structure...

Effect sizes in regression models

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




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

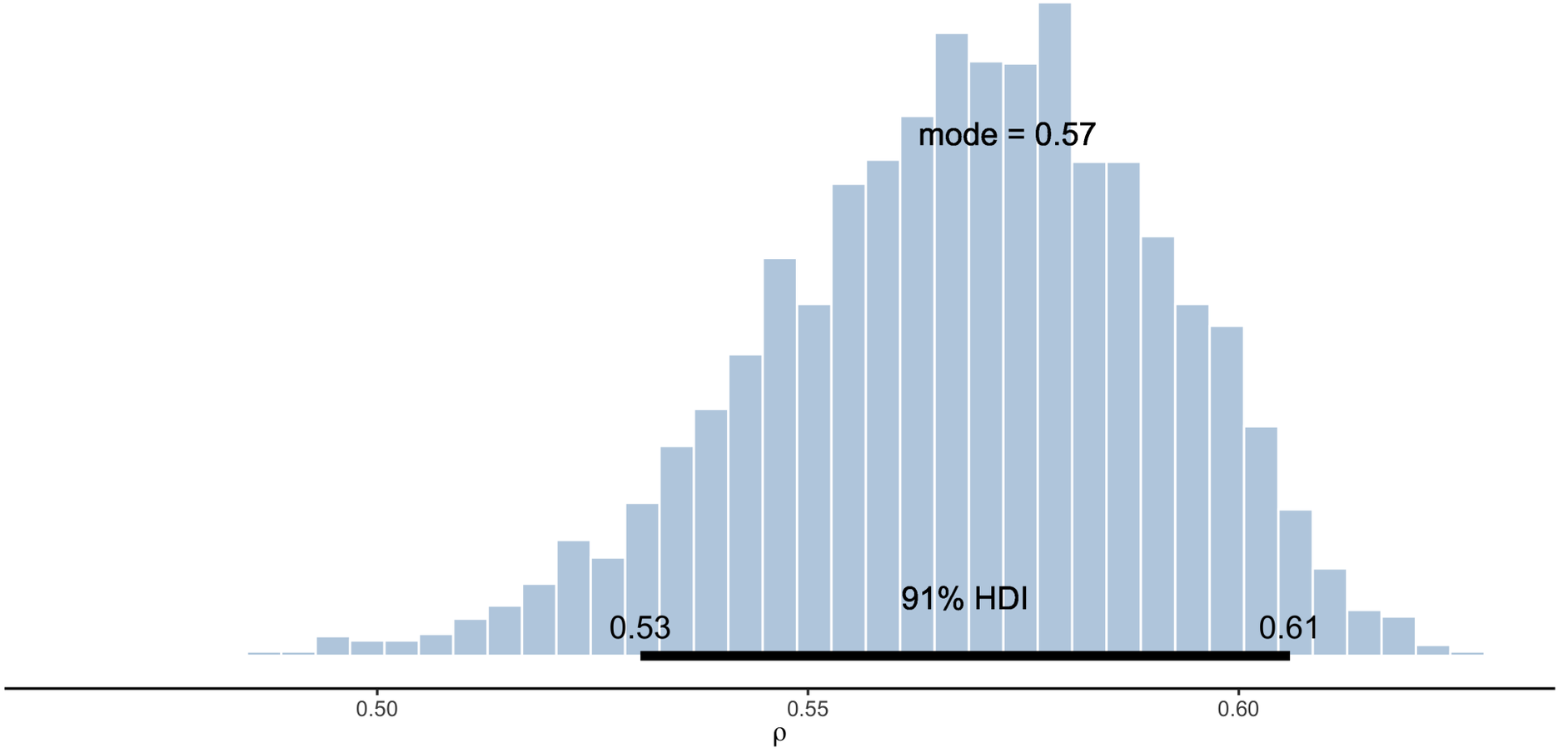
```
[1] 0.5696444
```

Effect sizes in regression models

```
bayes_R2(mod4)
```

```
   Estimate  Est.Error    Q2.5    Q97.5
R2 0.568025 0.02283433 0.51965 0.607488
```

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



mode = 0.57
91% HDI
0.53
0.61
0.50
0.55
0.60
ρ

Summary

A new model with two and then three parameters was presented: the Gaussian model, then Gaussian linear regression model, making it possible to relate two continuous variables.

As previously, Bayes' theorem is used to update our prior knowledge of parameter values into posterior knowledge, a synthesis between our priors and the information contained in the data.

The `brms` package can be used to fit all sorts of models with a syntax similar to that used by `lm()`.

The `fitted()` function is used to retrieve the predictions of a model.

The `predict()` function is used to simulate data from a model fitted with `brms`.

Practical work - 1/2

Select all rows in the `howell` dataset corresponding to minors individuals (age < 18). This should result in a dataframe of 192 rows.

Fit a linear regression model using the `brms::brm()` function. Report and interpret the estimates from this model. For an increase of 10 units in `weight`, what increase in height (`height`) does the model predict?

Plot the raw data with weight on the x-axis and height on the y-axis. Overlay the model's regression line of the model and an 89% credibility interval for the mean. Add an 89% credibility interval for the predicted sizes.

What do you think of the 'fit' of the model? What assumptions of the model would you be willing to modify in order to improve the model's fit?

Practical work - 2/2

Let's say you've consulted a colleague who's an expert in allometry (i.e., the phenomena of differential organ growth) and she explains to you that it doesn't make sense to model the relationship between weight and height... because we know that it's the logarithm of weight which is (linearly) related to height!

Model the relationship between height (cm) and log weight (log-kg). Use the entire `howell` dataframe (all 544 rows). Fit the following model using `brms::brm()`.

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta \cdot \log(w_i) \\
\alpha &\sim \text{Normal}(174, 100) \\
\beta &\sim \text{Normal}(0, 100) \\
\sigma &\sim \text{Exponential}(0.01)
\end{aligned}
$$

Where h_i is the height of individual i and w_i the weight of individual i. The function for calculating the log in R is simply `log()`. Can you interpret the results? Hint: plot the raw data and superimpose the model's predictions...

Proposed solution

```r
# we keep only the individuals younger than 18
d <- open_data(howell) %>% filter(age < 18)

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

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


Proposed 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    58.22      1.40    56.00    60.47 1.00     3853     3011
weight        2.72      0.07     2.61     2.83 1.00     3445     2962

Family Specific Parameters:
      Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
sigma     8.53      0.45     7.84     9.27 1.00     4283     2872

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).
```

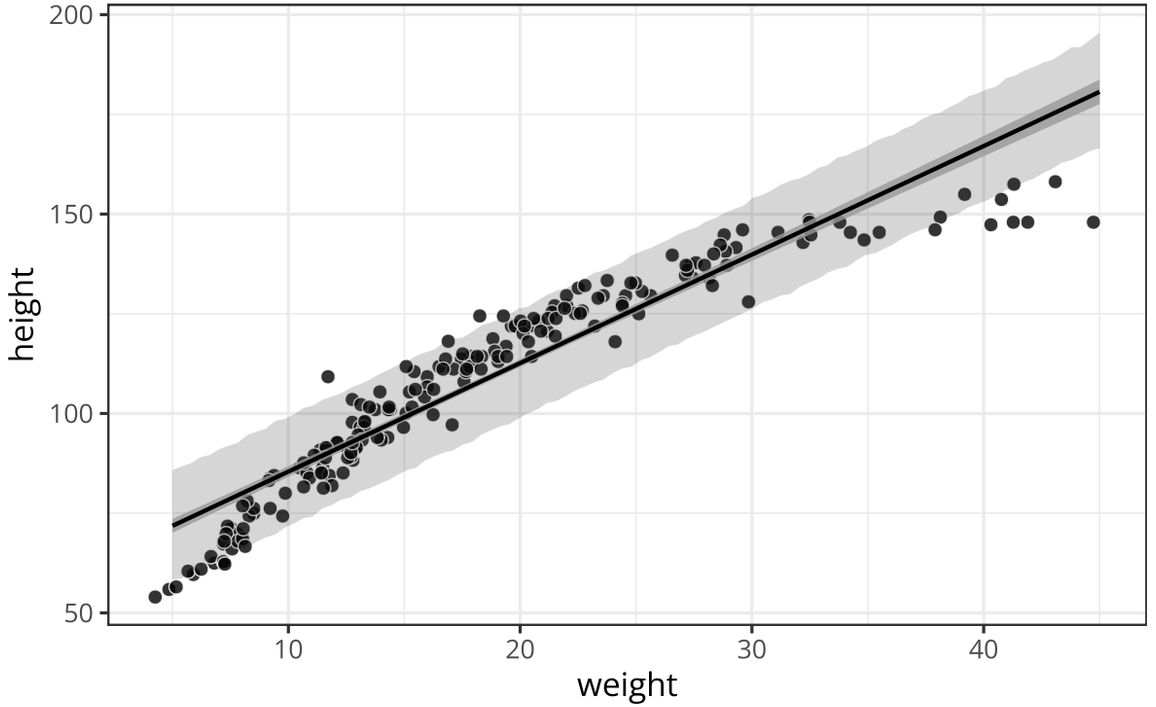

Proposed solution

```r
# defining a grid of possible values for "weight"
weight.seq <- data.frame(weight = seq(from = 5, to = 45, length.out = 1e2) )

# retrieving the model's predictions for these values fo "weight"
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)

# displaying the first 6 rows of pred_height
head(pred_height)
```

```
  Estimate Est.Error     Q5.5    Q94.5   weight
1 71.90158  8.558478 58.38327 85.76683 5.000000
2 72.80918  8.809837 58.64175 86.74542 5.404040
3 74.07137  8.558394 60.51732 87.58434 5.808081
4 75.08137  8.717473 61.08438 89.05932 6.212121
5 75.95542  8.633522 62.39603 89.61429 6.616162
6 77.39934  8.498564 63.69033 91.05000 7.020202
```


Proposed solution

```
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
    )
```



Proposed solution

```r
# we now consider all individuals
d <- open_data(howell)

mod8 <- brm(
  # we now predict height using the logarithm of weight
  height ~ 1 + log(weight),
  prior = priors,
  family = gaussian(),
  data = d
  )
```


Proposed 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   -23.56      1.34   -25.67   -21.42 1.00     4356     3128
logweight    47.01      0.38    46.39    47.63 1.00     4396     2952

Family Specific Parameters:
      Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
sigma     5.15      0.15     4.91     5.41 1.00     4641     2770

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).
```

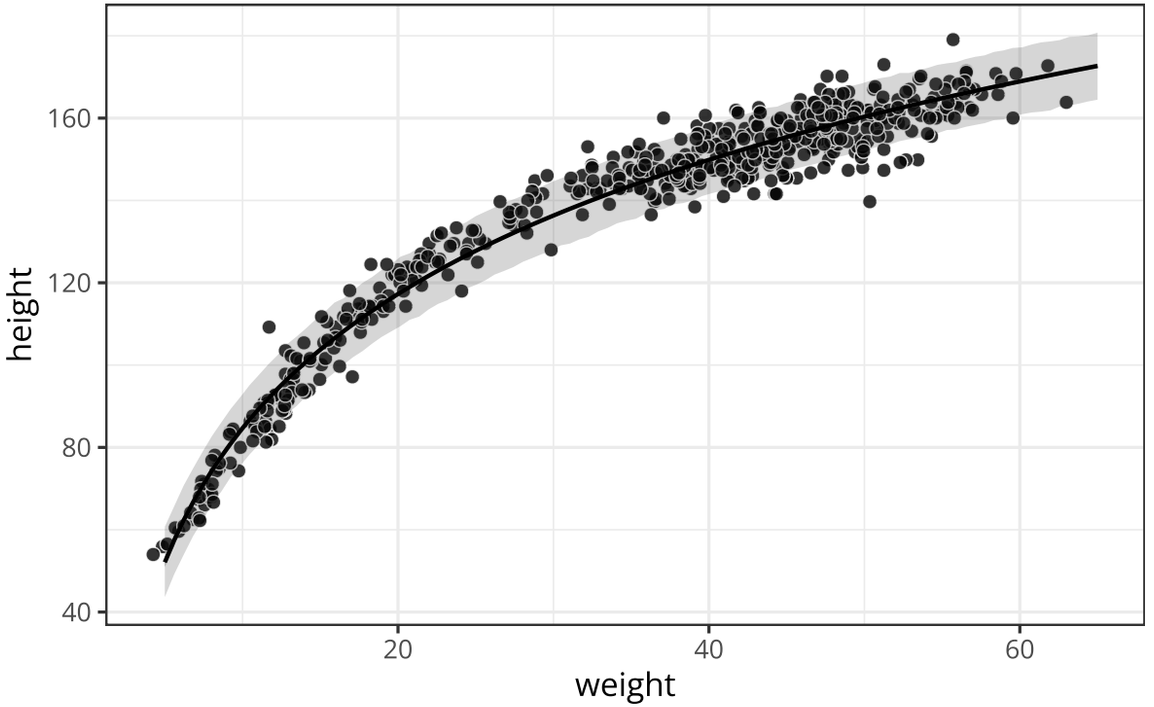

Proposed solution

```r
# defining a grid of possible values for "weight"
weight.seq <- data.frame(weight = seq(from = 5, to = 65, length.out = 1e2) )

# retrieving the model's predictions for these values of "weight"
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)

# displaying the first 6 rows of "pred_height"
head(pred_height)
```

```
  Estimate Est.Error     Q5.5    Q94.5   weight
1 52.08114  5.364763 43.64659 60.73506 5.000000
2 57.47366  5.180466 49.20607 65.72344 5.606061
3 62.29115  5.289523 53.91334 70.72491 6.212121
4 66.70296  5.104535 58.45216 74.90719 6.818182
5 70.49918  5.249748 62.17113 78.91841 7.424242
6 74.31789  5.177680 66.23217 82.75413 8.030303
```


Proposed solution

```
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
    )
```



References

Bürkner, P.-C. (2017). **brms** : An *R* Package for Bayesian Multilevel Models Using *Stan*. *Journal of Statistical Software*, *80*(1). https://doi.org/10.18637/jss.v080.i01

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

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

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

Nalborczyk, L., Batailler, C., Lœvenbruck, H., Vilain, A., & Bürkner, P.-C. (2019). An Introduction to Bayesian Multilevel Models Using brms: A Case Study of Gender Effects on Vowel Variability in Standard Indonesian. *Journal of Speech, Language, and Hearing Research*, *62*(5), 1225–1242. https://doi.org/10.1044/2018_jslhr-s-18-0006

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