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

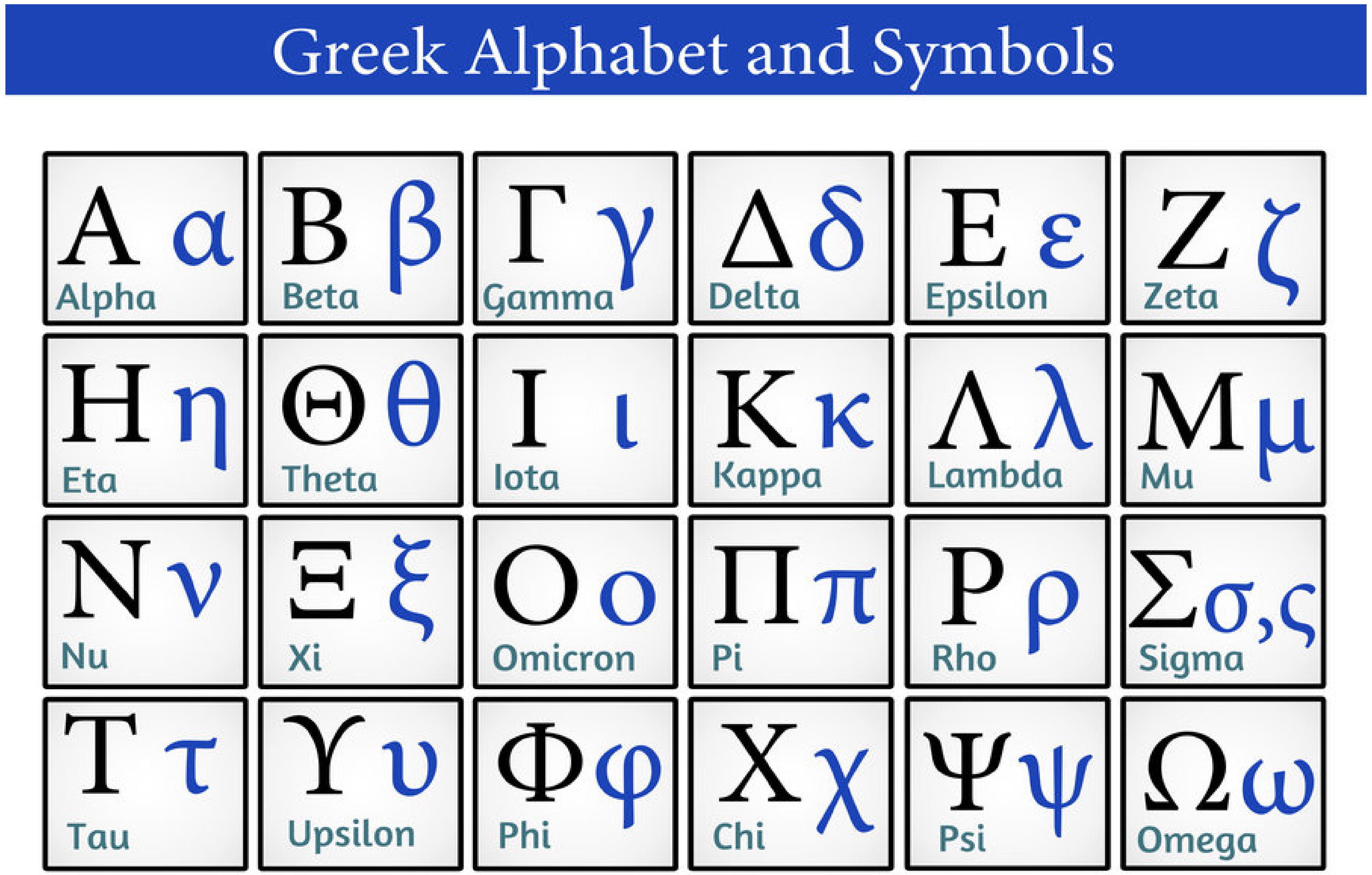
Reminders: notation

The notation $p(y \mid \theta)$ can refer to two things depending on the context: the likelihood function and the observation model. In addition, there are many ambiguous notations in statistics. Let's try to clarify them below.

- $\Pr(Y = y \mid \Theta = \theta)$ refers to a **probability** (e.g., `dbinom(x = 2, size = 10, prob = 0.5)`).
- $p(Y = y \mid \Theta = \theta)$ refers to a probability **density** (e.g., `dbeta(x = 0.4, shape1 = 2, shape2 = 3)`).
- $p(Y = y \mid \Theta)$ refers to a (discrete or continuous) likelihood function, y is given/known/fixed, Θ is a random variable, the sum (or the integral) of this distribution **is not equal to 1** (e.g., `dbinom(x = 2, size = 10, prob = seq(0, 1, 0.1) )`).
- $p(Y \mid \Theta = \theta)$ refers to a probability mass (or density) function (of which the sum or the integral **is equal** to 1) that we call the "observation model" ot "sampling distribution", Y is a random variable, θ is given/known/fixed (e.g., `dbinom(x = 0:10, size = 10, prob = 0.5)`)

The goal of a Bayesian analysis (i.e., what is obtained at the end of such an analysis) is the posterior distribution $p(\theta \mid y)$. It can be summarised to make the communication of results easier, but all the desired information is contained in **the entire distribution** (not just its mean, mode, or whatever).

Reminders: notation



Greek Alphabet and Symbols

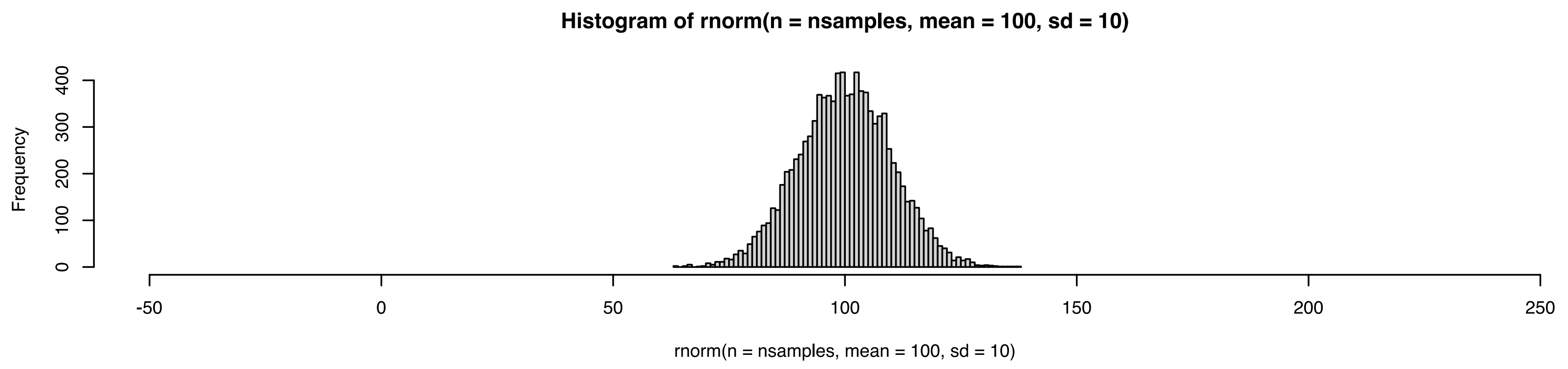
Α α Alpha	Β β Beta	Γ γ Gamma	Δ δ Delta	Ε ε Epsilon	Ζ ζ Zeta
Η η Eta	Θ θ Theta	Ι ι Iota	Κ κ Kappa	Λ λ Lambda	Μ μ Mu
Ν ν Nu	Ξ ξ Xi	Ο ο Omicron	Π π Pi	Ρ ρ Rho	Σ σ,ς Sigma
Τ τ Tau	Υ υ Upsilon	Φ φ Phi	Χ χ Chi	Ψ ψ Psi	Ω ω Omega

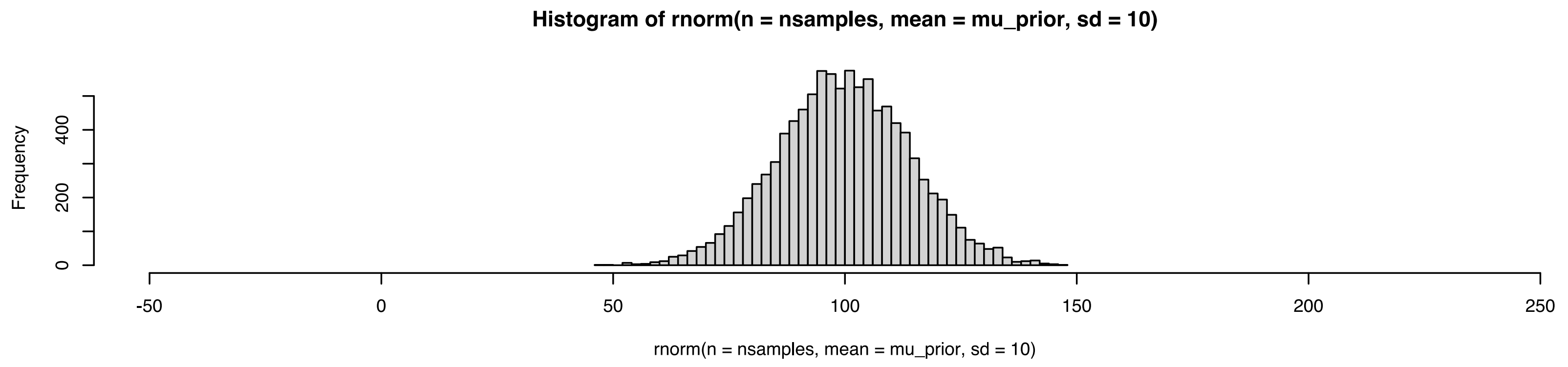
Figure from https://masterofmemory.com/mmem-0333-learn-the-greek-alphabet/.

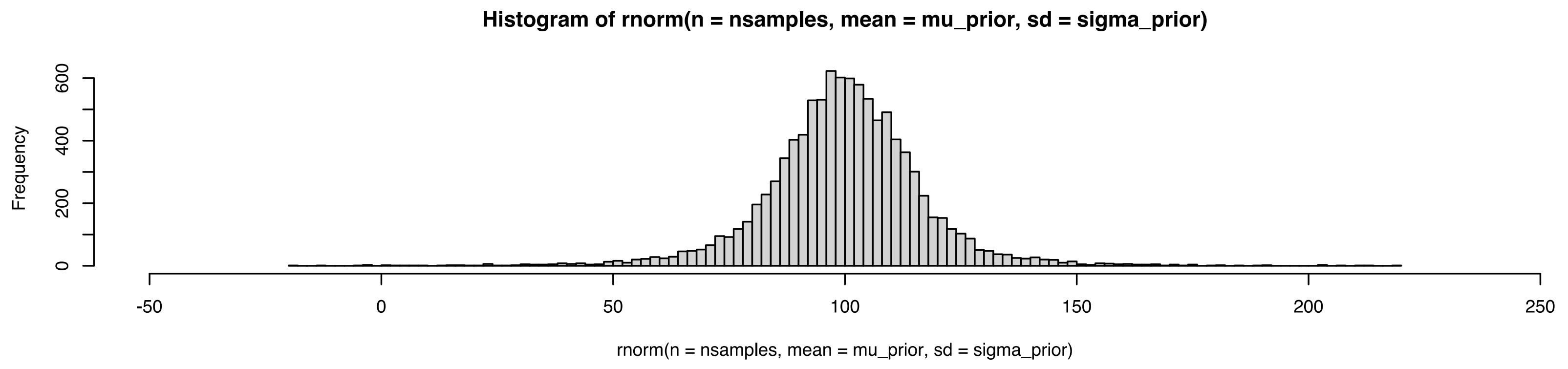
Reminders: prior predictive checking

```r
#####################################################################
# We define a model with:                                           #
# A Gaussian likelihood function: y ~ Normal(mu, sigma)             #
# A Gaussian prior for the mean: mu ~ Normal(100, 10)               #
# An Exponential prior for the dispersion: sigma ~ Exponential(0.1) #
#####################################################################

# drawing 10.000 observations from a Gaussian distribution without (epistemic) uncertainty
rnorm(n = 1e4, mean = 100, sd = 10) |> hist(breaks = "FD")

# drawing 10.000 observations from the Gaussian prior on mu (i.e., p(mu))
# this prior represents what we know about mu before seeing the data...
mu_prior <- rnorm(n = 1e4, mean = 100, sd = 10)

# drawing 10.000 observations from a Gaussian distribution with prior-related (epistemic) uncertainty
rnorm(n = 1e4, mean = mu_prior, sd = 10) |> hist(breaks = "FD")

# drawing 10.000 observations from the Exponential prior on sigma (i.e., p(sigma))
# this prior represents what we know about sigma before seeing the data...
sigma_prior <- rexp(n = 1e4, rate = 0.1)

# drawing 10.000 observations from a Gaussian distribution with prior-related
# (epistemic) uncertainty on mu AND sigma
# this is what the model expects about y given our priors about mu and sigma and the observation model
rnorm(n = 1e4, mean = mu_prior, sd = sigma_prior) |> hist(breaks = "FD")
```

Reminders: prior predictive checking





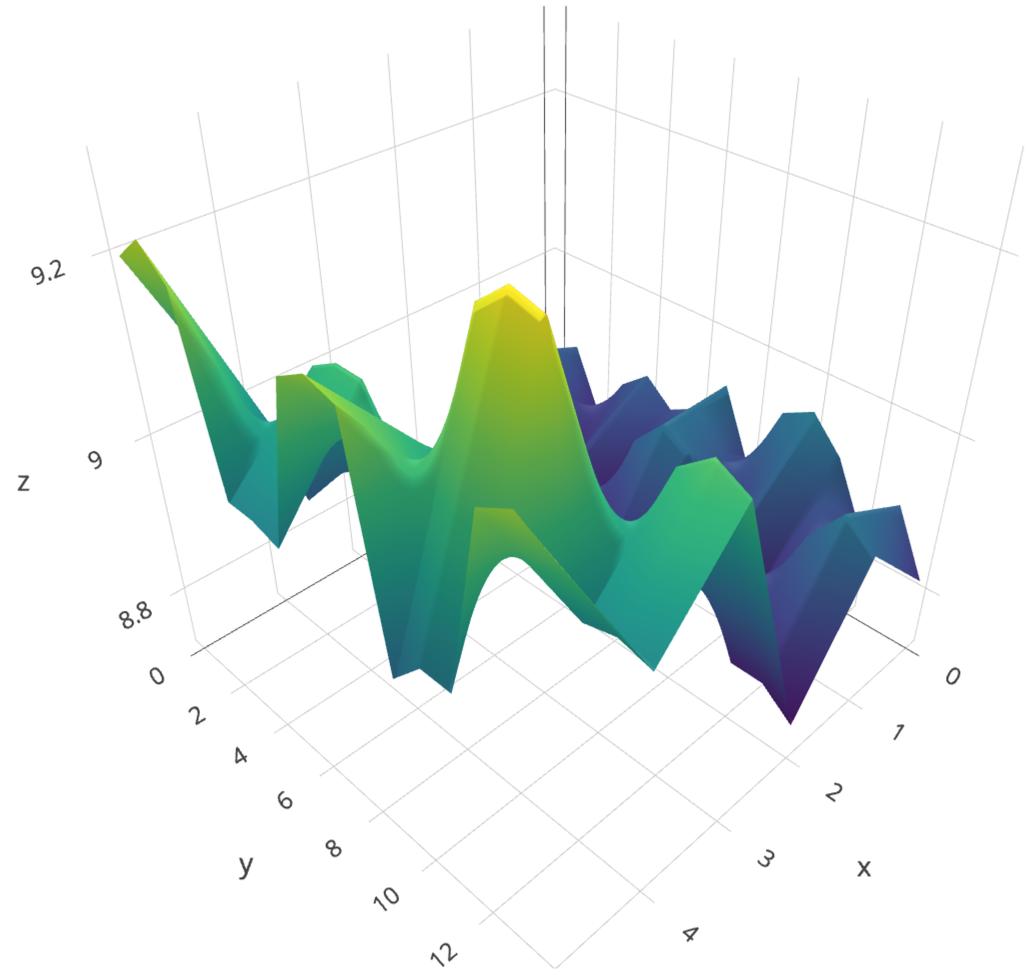


The problem with posterior distributions...

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

Problem: The normalisation constant (in green) is obtained by calculating the sum (for discrete variables) or the integral (for continuous variables) of the joint density $p(\text{data}, \theta)$ over all possible values of θ. This becomes complicated when the model includes several parameters and/or the shape of the posterior distribution is complex...

The problem with posterior distributions...



Reminders from Course n°01

There are three ways of getting around this problem:

- The prior distribution is a **conjugate prior** of the likelihood function (e.g. Beta-Binomial model). In this case, there is an analytical solution (i.e., one that can be calculated exactly) for the the posterior distribution.
- Alternatively, for simple models, we can use the **grid method**. The exact value of the posterior probability is calculated at a finite number of points in the parameter space.
- For more complex models, exploring the entire parameter space space is usually not tractable. Instead, we will sample a large number of points in the parameter space and use these samples as an approximation of the posterior distribution, but we will sample the posterior space in a smart way.

Markov Chain Monte Carlo

Markov Chain Monte Carlo

- Markov chain **Monte Carlo**

  $\longrightarrow$ Random sampling

  $\longrightarrow$ The result is an ensemble of parameter values (samples)

- Markov **chain** Monte Carlo

  $\longrightarrow$ Values are generated in a sequence

  $\longrightarrow$ With a temporal index to identify the position in the chain

  $\longrightarrow$ The result looks like: $\theta^1, \theta^2, \theta^3, \dots, \theta^t$

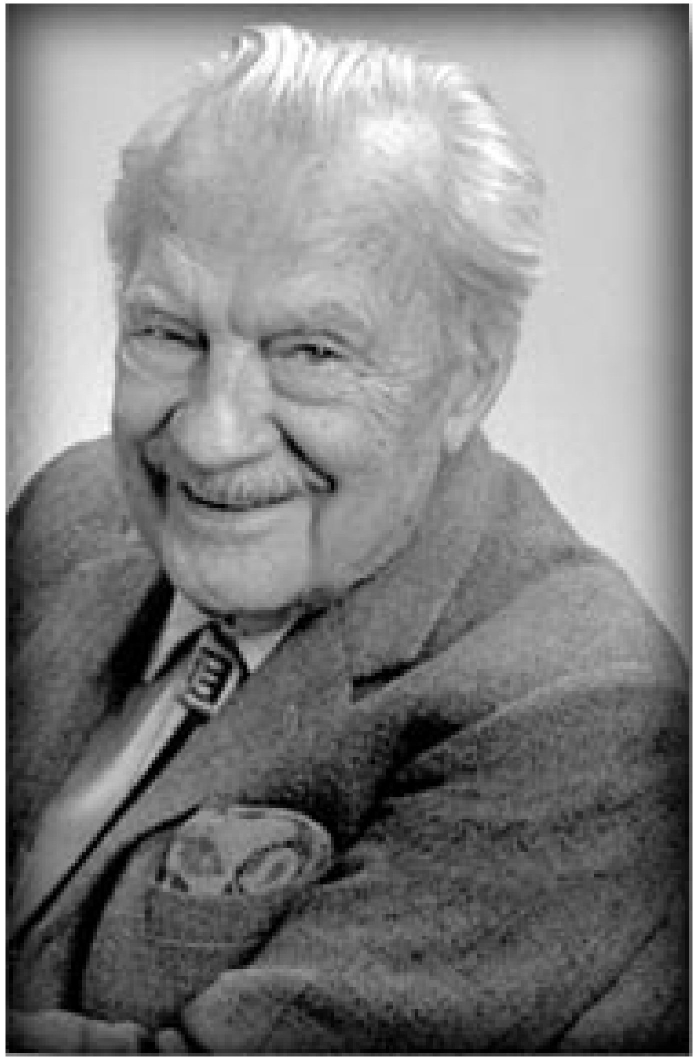
- **Markov** chain Monte Carlo

  $\longrightarrow$ The current parameter value only depends on the previous parameter value:

  $\Pr(\theta^{t+1} \mid \theta^t, \theta^{t-1}, \dots, \theta^1) = \Pr(\theta^{t+1} \mid \theta^t)$

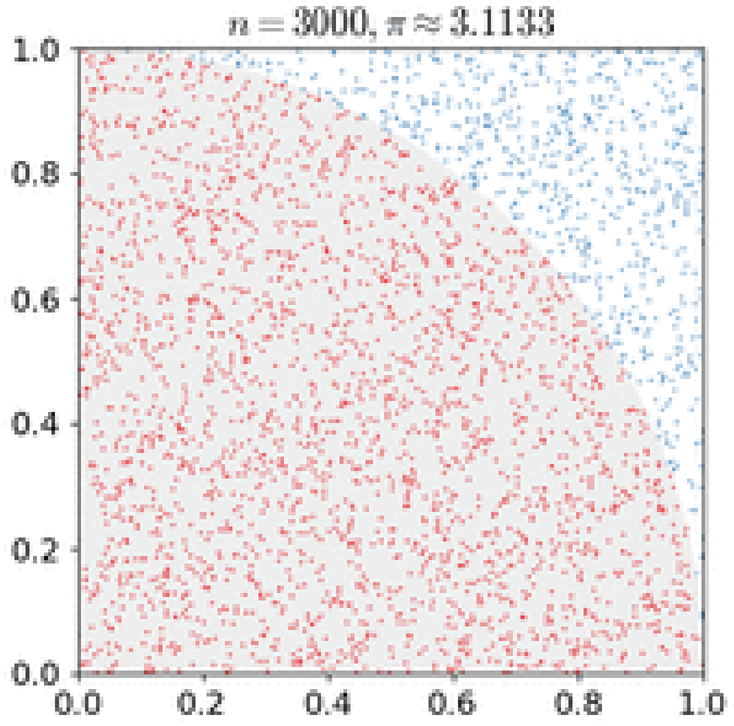
Monte Carlo methods

Monte-Carlo refers to a family of algorithms designed to calculate (or approximate) a numerical value using random processes (i.e., probabilistic techniques). The method was formalised in 1947 by Nicholas Metropolis, and first published in 1949 in an article co-authored with Stanislaw Ulam.



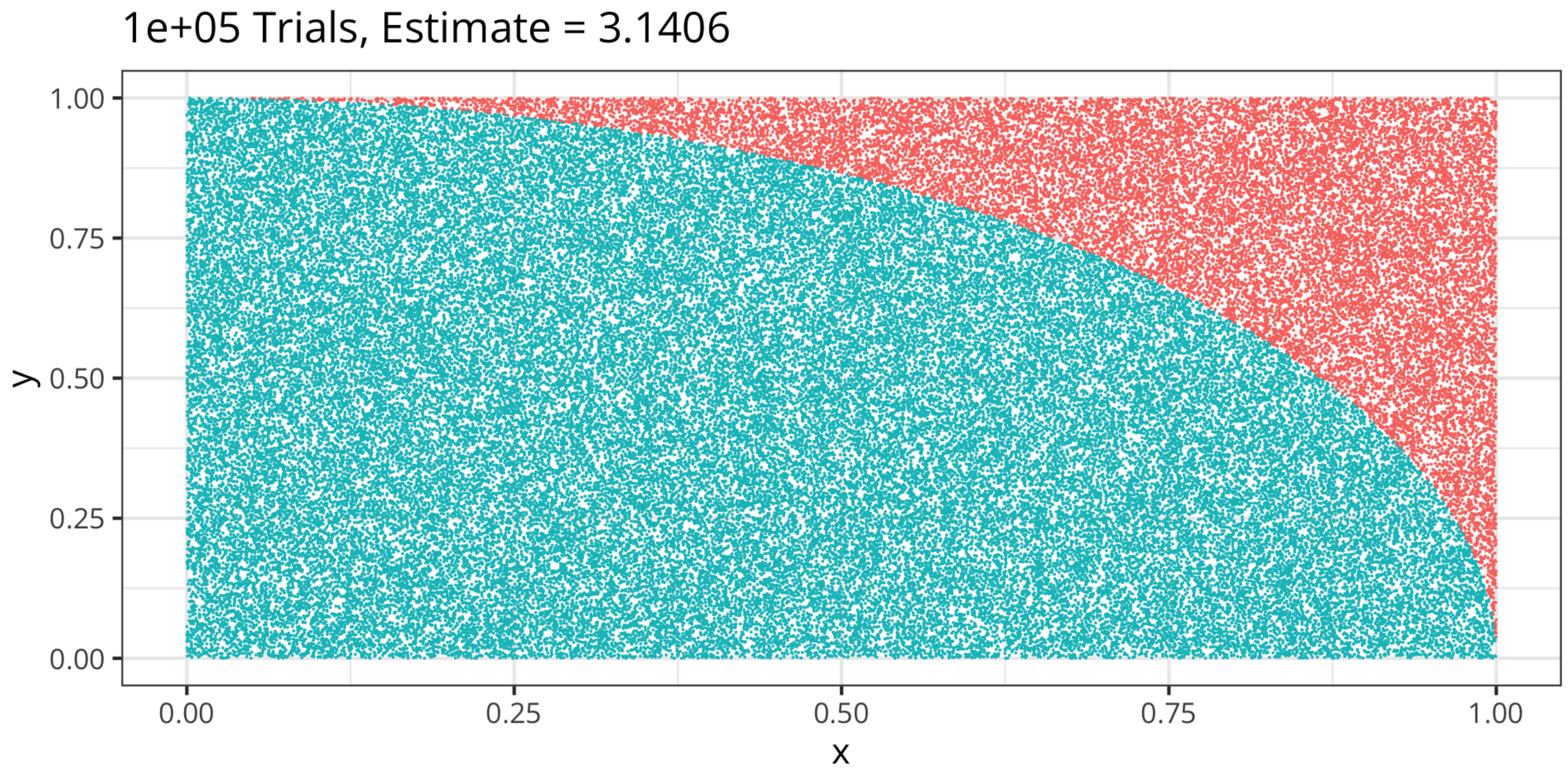
Monte Carlo methods: estimating π

Let M be a point with coordinates (x, y), where $0 < x < 1$ and $0 < y < 1$. We randomly draw the values of x and y between 0 and 1 according to a uniform distribution. The point M belongs to the disc of centre $(0, 0)$ and radius $r = 1$ if and only if $\sqrt{x^2 + y^2} \leqslant 1$. We know that the area of the quarter disc is $\sigma = \pi r^2/4 = \pi/4$ and that the square which contains it has a surface $s = r^2 = 1$. If the probability distribution of which the point is drawn is uniform, then the probability that point M belongs to disc is $\sigma/s = \pi/4$. By dividing the number of points in the disc by the number of draws $\frac{N_{\text{inner}}}{N_{\text{total}}}$, we obtain an approximation of $\pi/4$.



Monte Carlo methods: estimating π

```r
trials <- 1e5 # number of samples
radius <- 1 # radius of the circle
x <- runif(n = trials, min = 0, max = radius) # draws for x
y <- runif(n = trials, min = 0, max = radius) # draws for y
distance <- sqrt(x^2 + y^2) # distance to origin
inside <- distance < radius # is it within the quarter of circle?
pi_estimate <- 4 * sum(inside) / trials # estimated value of pi
```



Méthodes Monte Carlo

Monte-Carlo refers to a family of algorithms designed to calculate (or approximate) a numerical value using random processes (i.e., probabilistic techniques). Can we use this sort of methods to approximate the posterior distribution?

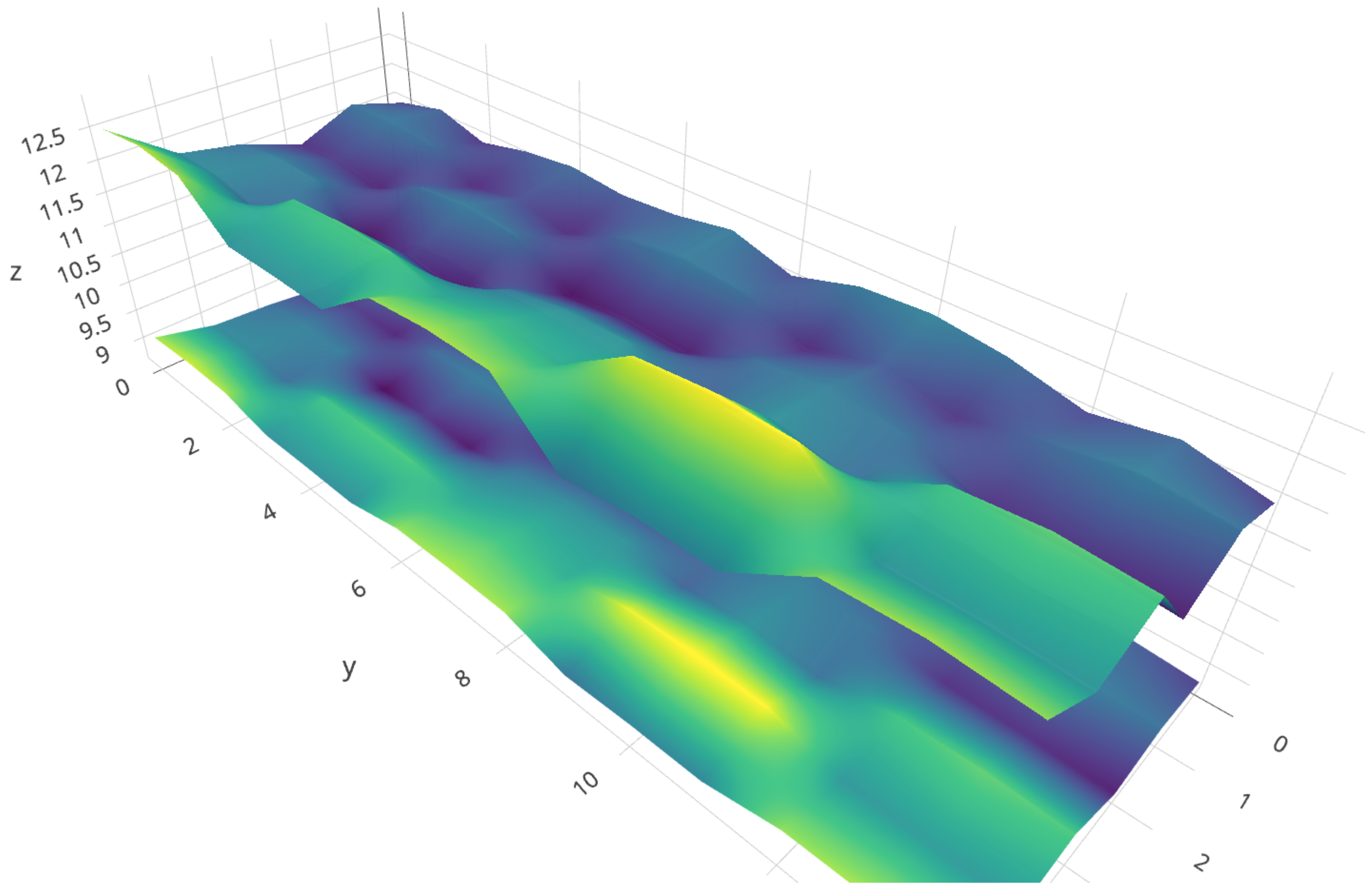
We know the priors $p(\theta_1)$ and $p(\theta_2)$.
We know the likelihood function $p(\text{data} \mid \theta_1, \theta_2)$.

But often, we do not know how to compute the exact posterior distribution

$$p(\theta_1, \theta_2 \mid \text{data}) = \frac{p(\text{data} \mid \theta_1, \theta_2) p(\theta_1) p(\theta_2)}{p(\text{data})}.$$

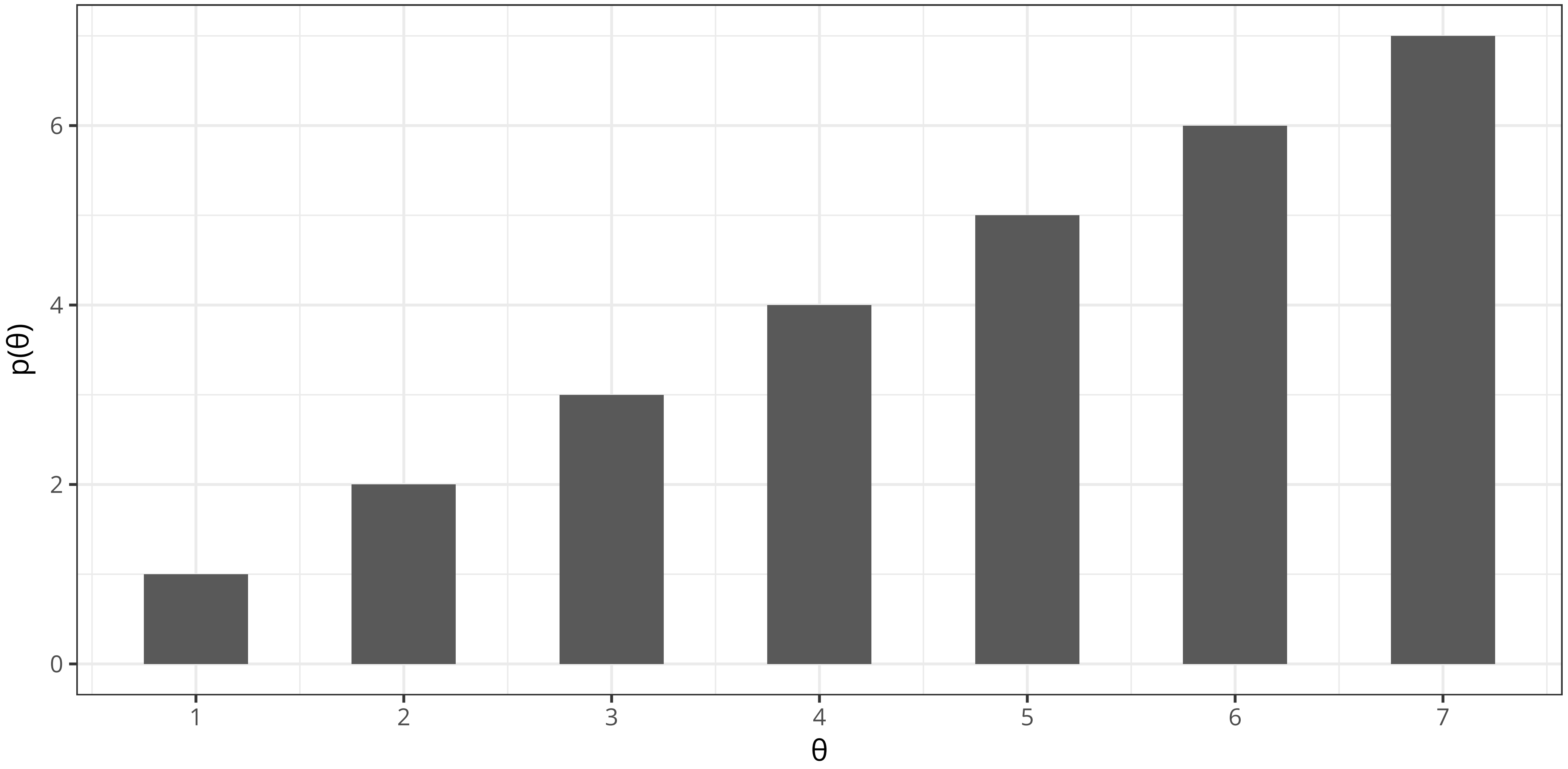
Or rather, we don't know how to compute $p(\text{data})$...! But we can compute something that is proportional to the posterior distribution. Since $p(\text{data})$ is a constant, it does not change the shape of the posterior distribution! So we're going to explore the parameter space and produce samples in proportion to their relative probability (density).

Influence of the normalisation constant



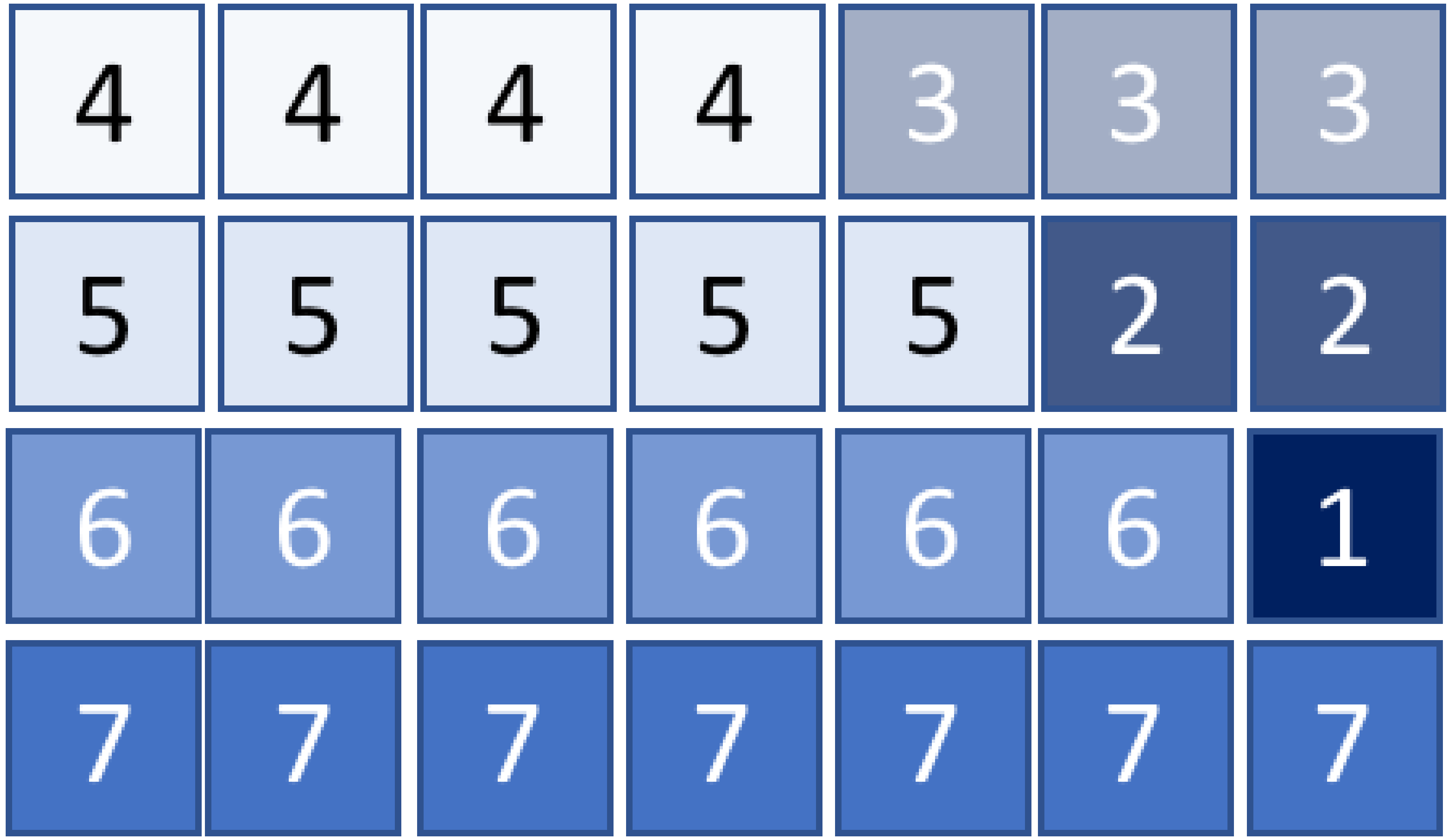
Monte Carle methods: Example

Let's consider a simple example: We have a parameter θ with 7 possible values and the following distribution function, where $p(\theta) = \theta$.



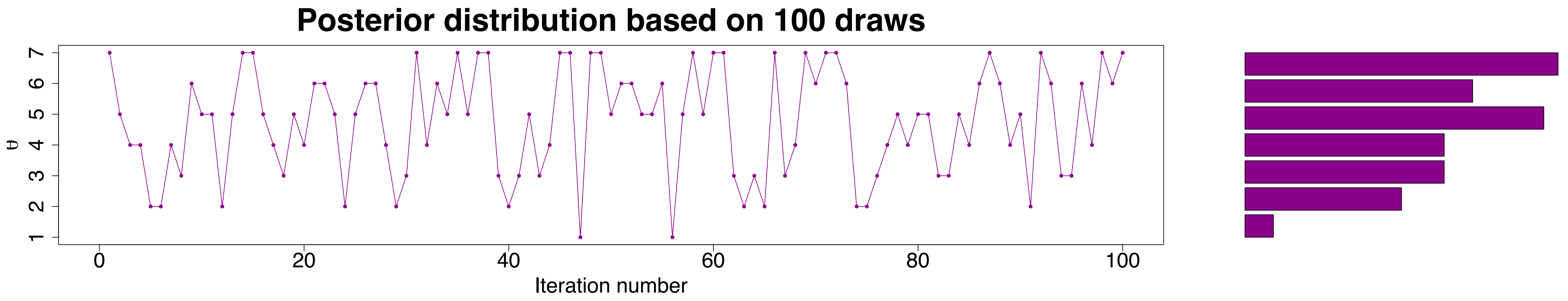
Monte Carle methods: Example

We can approximate this distribution by random draw: This amounts to drawing a large number of points “at random” from among these 28 squares (as in the π example)!



Monte Carle methods: Example

```r
niter <- 100 # number of samples
theta <- 1:7 # possible values for theta
ptheta <- theta # probability of theta
samples <- sample(x = theta, prob = ptheta, size = niter, replace = TRUE) # samples
```



- The distribution of samples converges towards the “true” distribution.
- But this generally requires a lot of samples...
- No control over the speed of convergence...
- Should we abandon independent sampling?

Metropolis algorithm

This algorithm was first presented in Metropolis et al. (1953). The problem with Monte-Carlo algorithms is not convergence, but the speed at which the method converges. To increase the speed of convergence, we want to **facilitate access to the most likely parameter values**.

Principle:

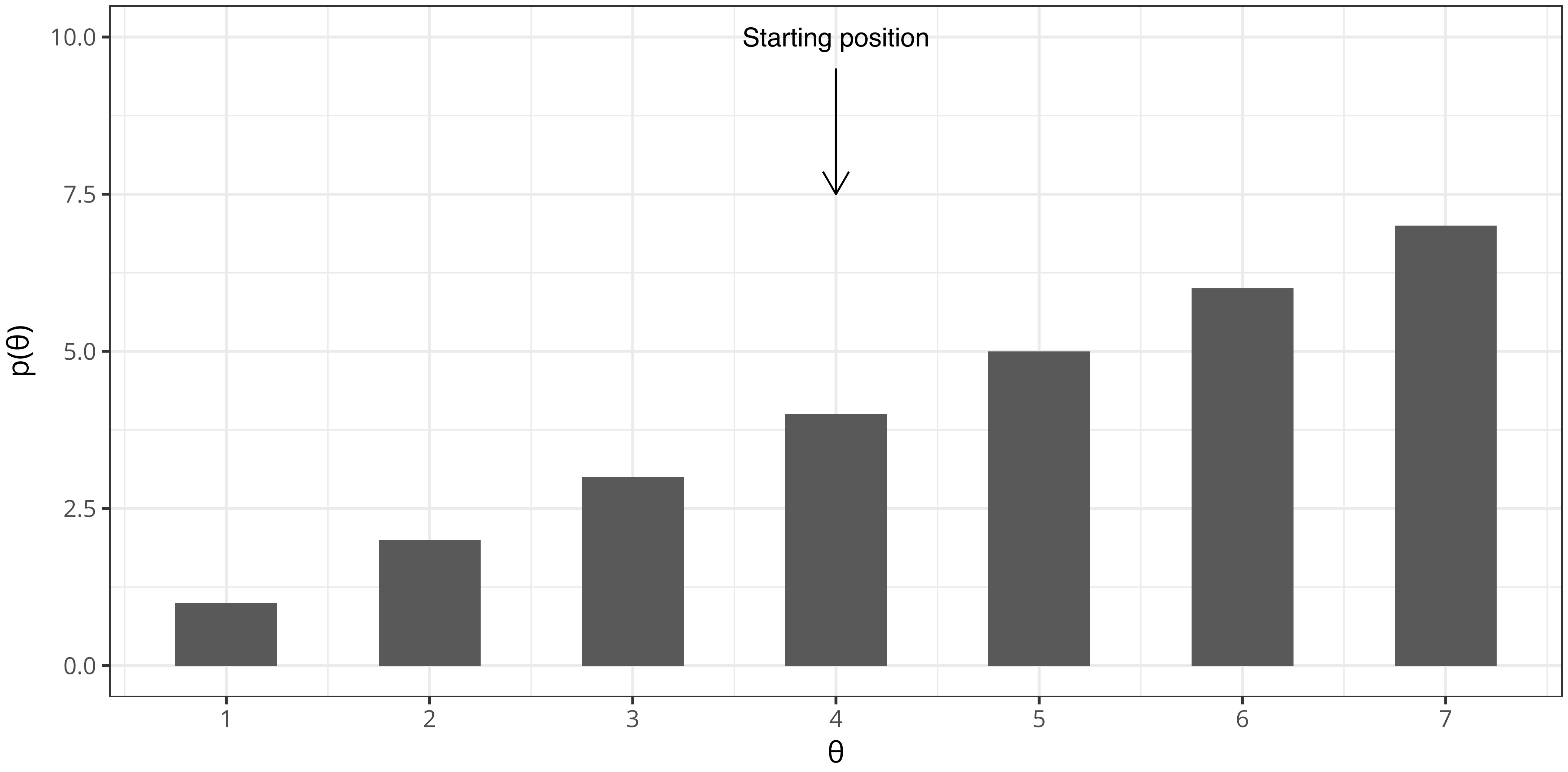
- A proposal (a new position) is made on the basis of the current value of the parameter.
- A random draw is made to accept or reject the new position.

Two central ideas:

- The proposal should favour the most probable parameter values: These parameter values should be proposed more often.
- The proposal should be limited to values adjacent to the current parameter: The speed of convergence is increased by staying where the information is (i.e., by traversing the parameter space **locally** rather than **globally**).

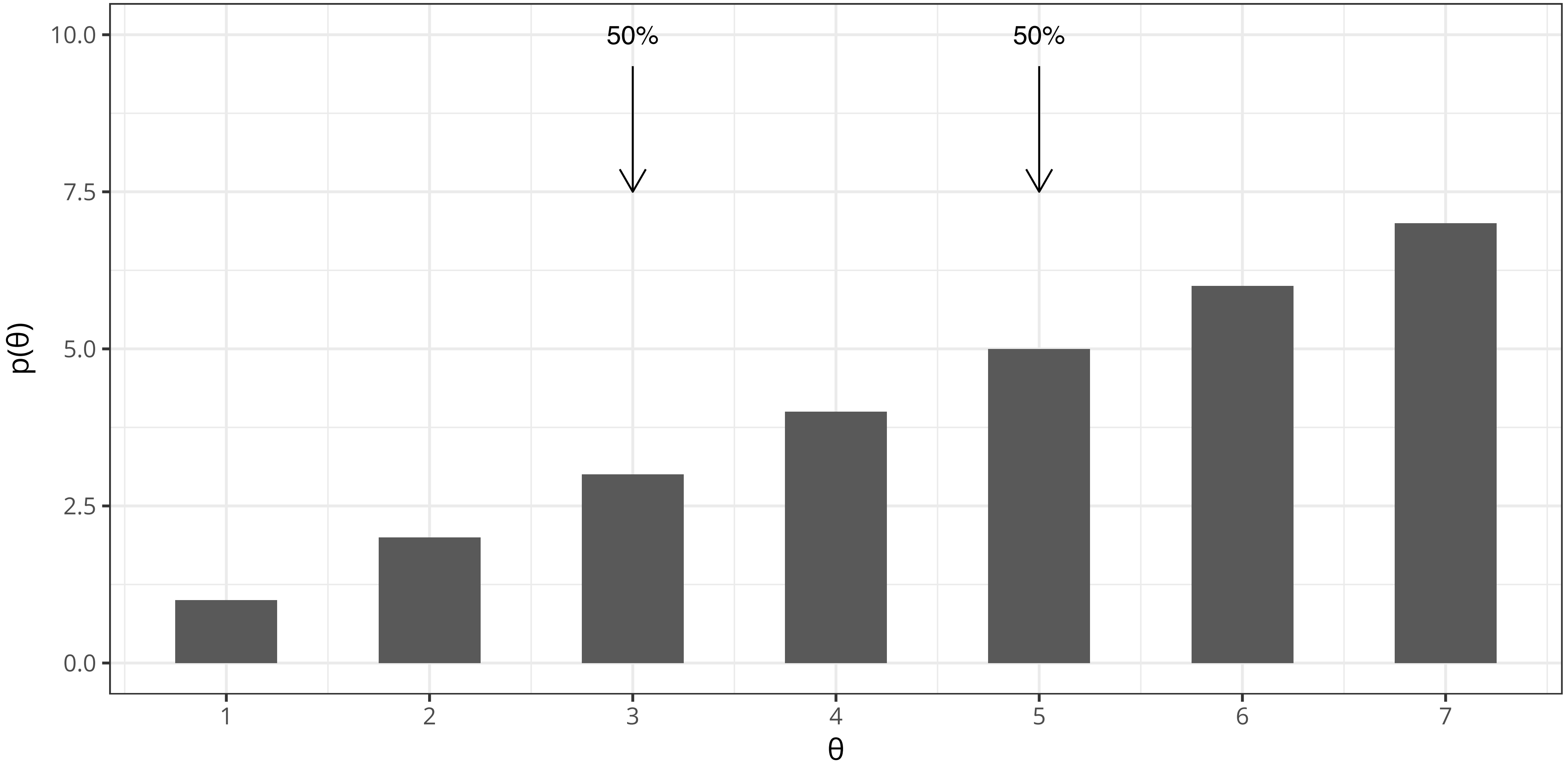
Metropolis algorithm in details

Select a starting point (any value of θ can be selected).



Metropolis algorithm in details

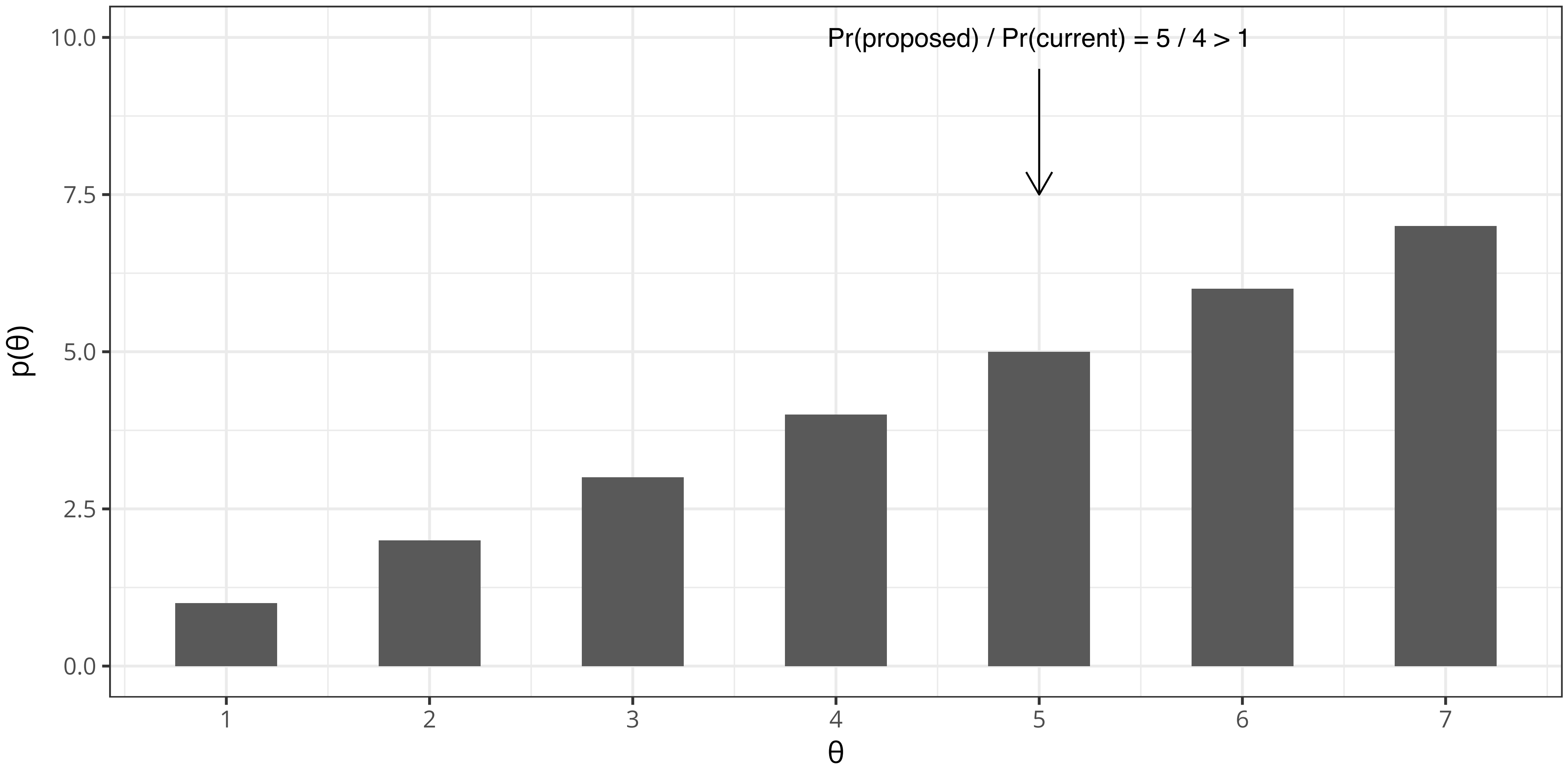
Propose a new position (i.e., a new value for θ) centred on the current value of θ.



Metropolis algorithm in details

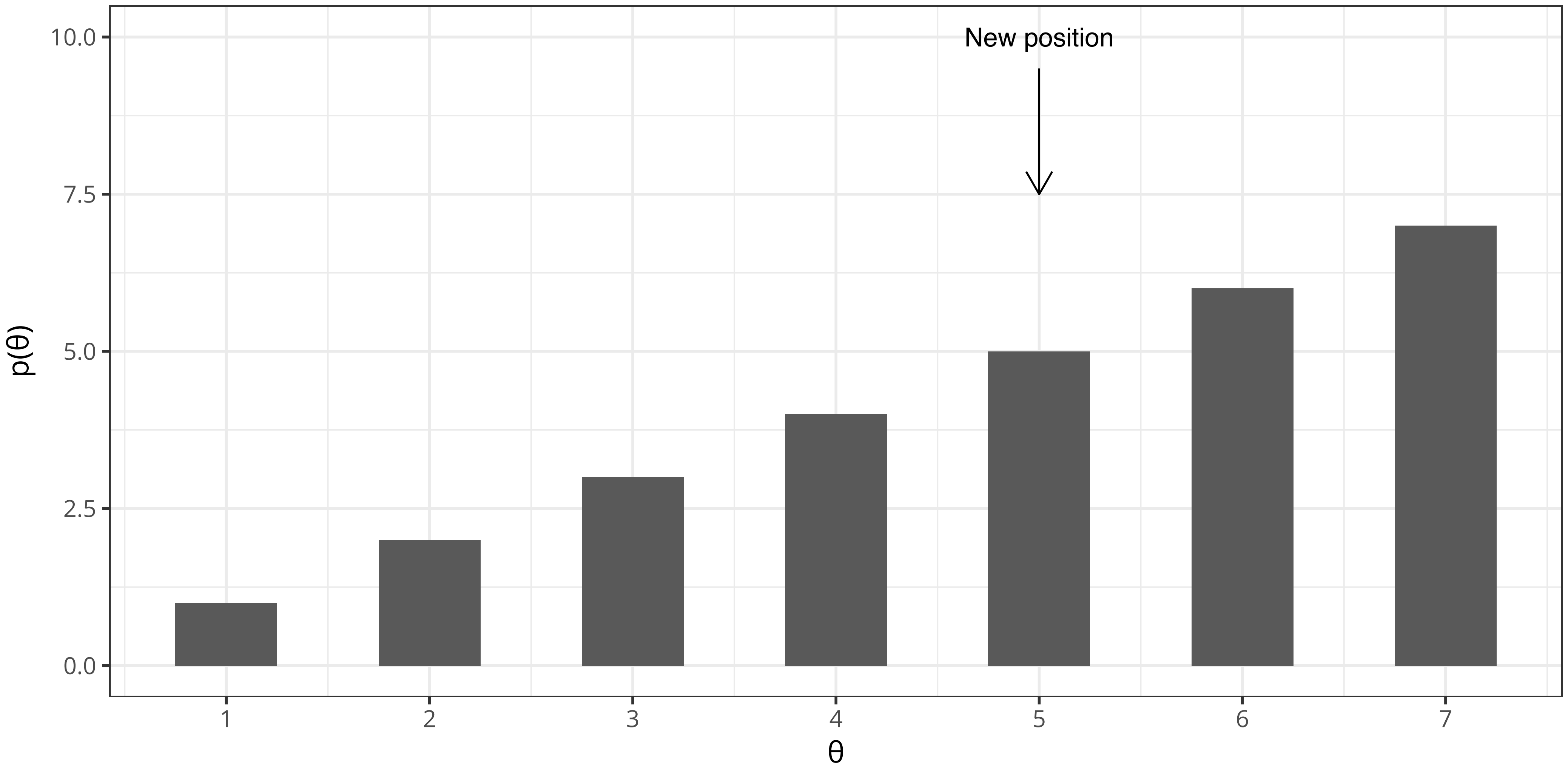
Calculate the **probability** of moving to the new position according to the following rule:

$$\Pr_{\text{move}} = \min\left(\frac{\Pr(\theta_{\text{proposed}})}{\Pr(\theta_{\text{current}})}, 1\right)$$



Metropolis algorithm in details

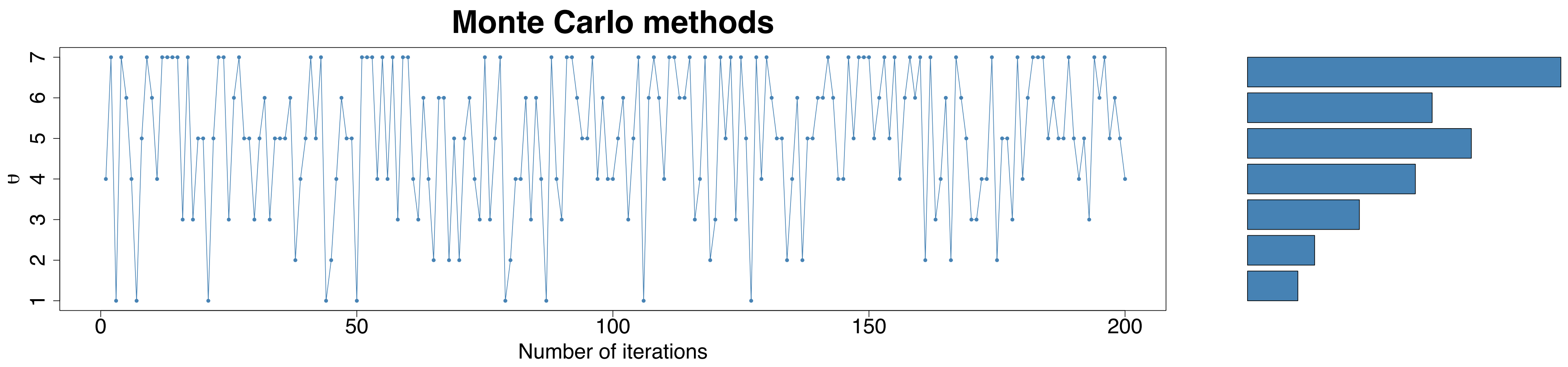
The accepted position becomes the new starting position and the algorithm is repeated.

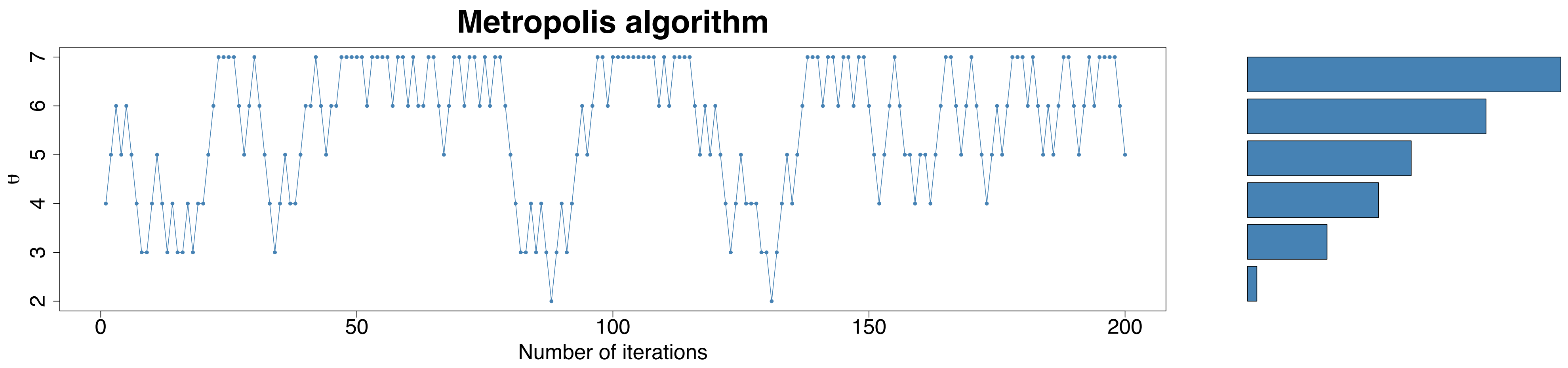


Metropolis algorithm in details

```r
metropolis <- function (niter = 1e2, startval = 4) {

    x <- rep(0, niter) # initialising the chain (vector) of length niter
    x[1] <- startval # defining the initial value of the parameter

    for (i in 2:niter) { # for each iteration

        current <- x[i - 1] # current value of the parameter
        proposal <- current + sample(c(-1, 1), size = 1)
        # we ensure the proposed value is within the [1, 7] interval
        if (proposal < 1) proposal <- 1
        if (proposal > 7) proposal <- 7
        # computing the probability of moving to the proposed position
        prob_move <- min(1, proposal / current)
        # we move (or not) according to this probability
        # x[i] <- ifelse(prob_move > runif(n = 1, min = 0, max = 1), proposal, current)
        x[i] <- sample(c(proposal, current), size = 1, prob = c(prob_move, 1 - prob_move) )

    }

    # returning the entire chain
    return (x)

}
```

Monte Carlo methods vs. Metropolis algorithm





Metropolis algorithm

Application to coin tosses (continuous case)

- The likelihood function: $p(y \mid \theta, n) \propto \theta^{y}(1 - \theta)^{(n-y)}$
- Prior: $p(\theta \mid a, b) \propto \theta^{(a-1)}(1 - \theta)^{(b-1)}$
- The parameter we want to estimate lies in the $[0, 1]$ interval.

Problem n°1: How should we define the proposed move?

The proposal can be modelled by a normal distribution: $\Delta\theta \sim \text{Normal}(0, \sigma)$

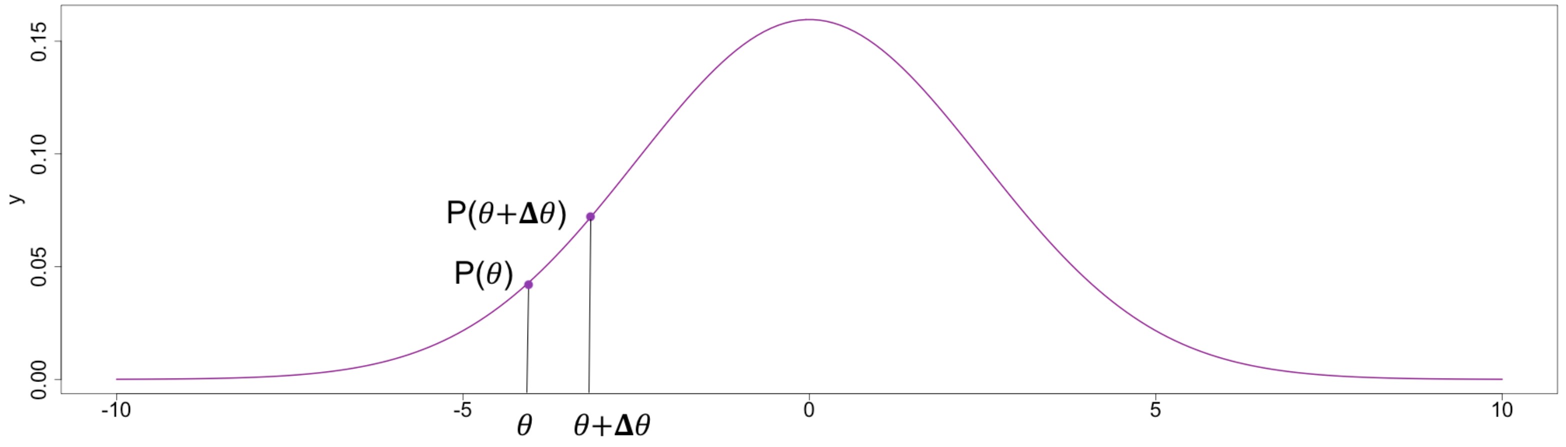
$\longrightarrow$ The mean μ is 0: the proposed position is around the current value of the parameter

$\longrightarrow$ The variance remains to be determined, it controls the distance from the new value.

Metropolis algorithm

Problem n°2 : What probability should we use to accept or refuse the move? We use the product of the likelihood and the prior: $\theta^y(1-\theta)^{(n-y)}\theta^{(a-1)}(1-\theta)^{(b-1)}$

The probability of accepting the move is given by: $\text{Pr}_{\text{move}} = \min\left(\frac{\text{Pr}(\theta_{\text{current}}+\Delta\theta)}{\text{Pr}(\theta_{\text{current}})}, 1\right)$



NOTE: The ratio $\frac{\text{Pr}(\theta_{\text{current}}+\Delta\theta)}{\text{Pr}(\theta_{\text{current}})}$ is the same whether you use the posterior distribution or the product of the prior and the likelihood (because the normalisation constant cancels out)!

Metropolis algorithm

- Select a starting point
- Choose $\theta \in [0, 1]$
- Only constraint: $\Pr(\theta_{\text{initial}}) \neq 0$.

$\longrightarrow$ Choose a direction of movement

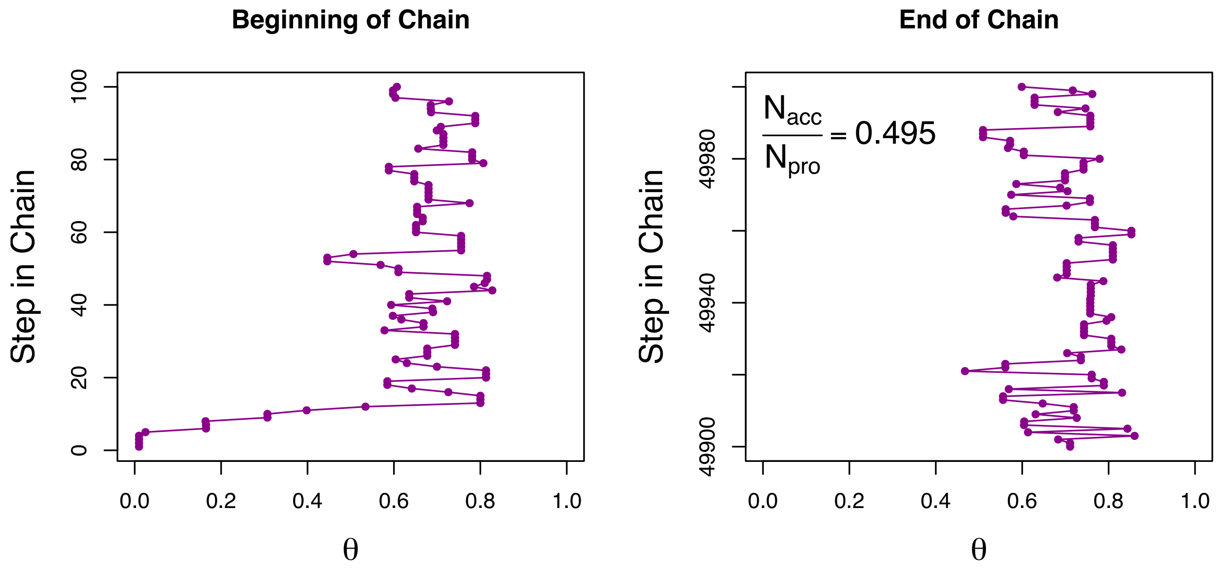
- Make a draw according to $\text{Normal}(0, \sigma)$

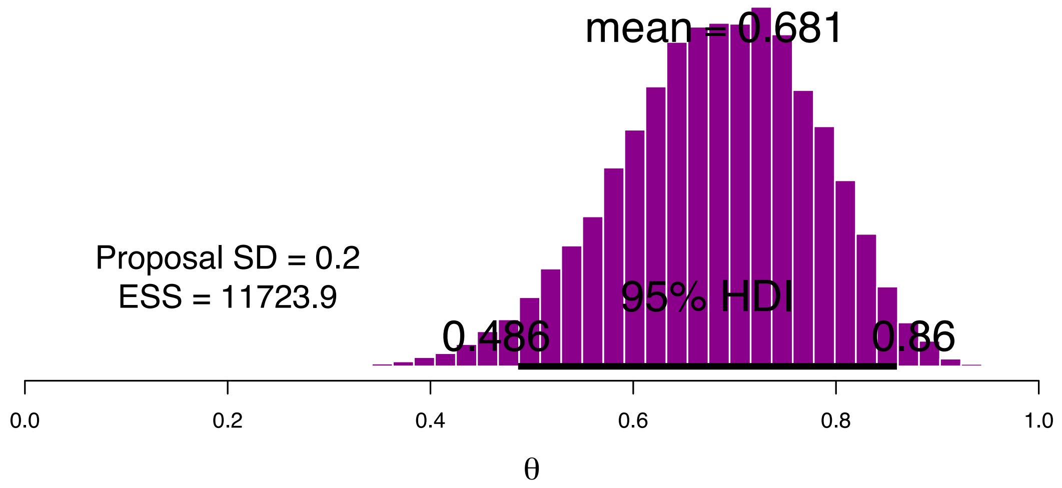
$\longrightarrow$ Accept or reject the proposed move, depending on the probability:

$$\Pr_{\text{move}} = \min\left(\frac{\Pr(\theta_{\text{current}} + \Delta\theta)}{\Pr(\theta_{\text{current}})}, 1\right)$$

$\longrightarrow$ The calculated position becomes the new position

Metropolis algorithm





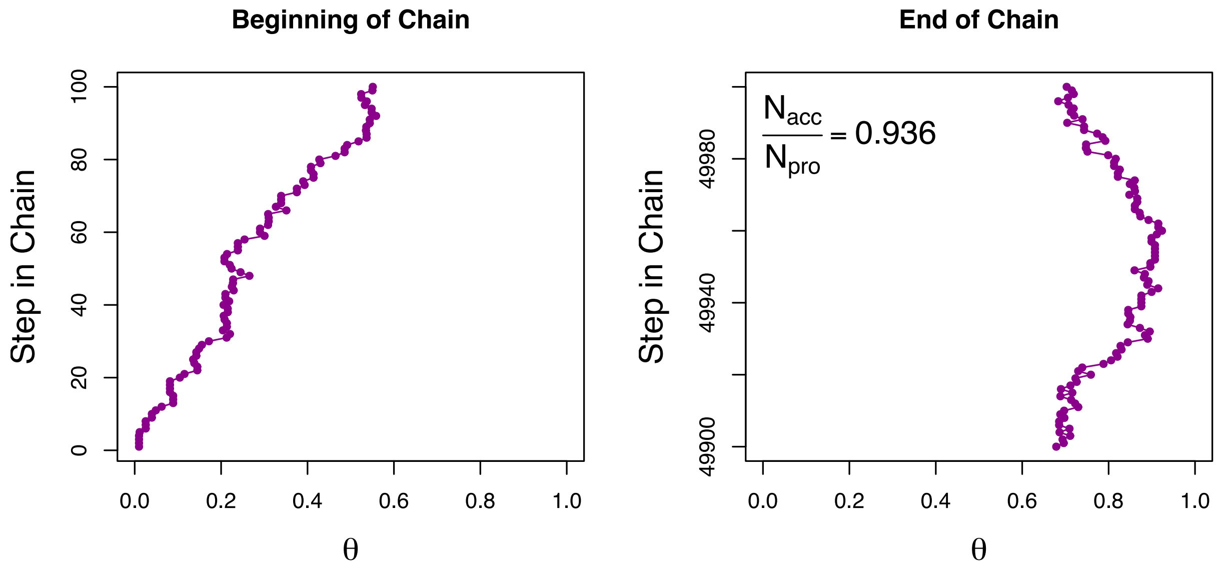
Metropolis algorithm

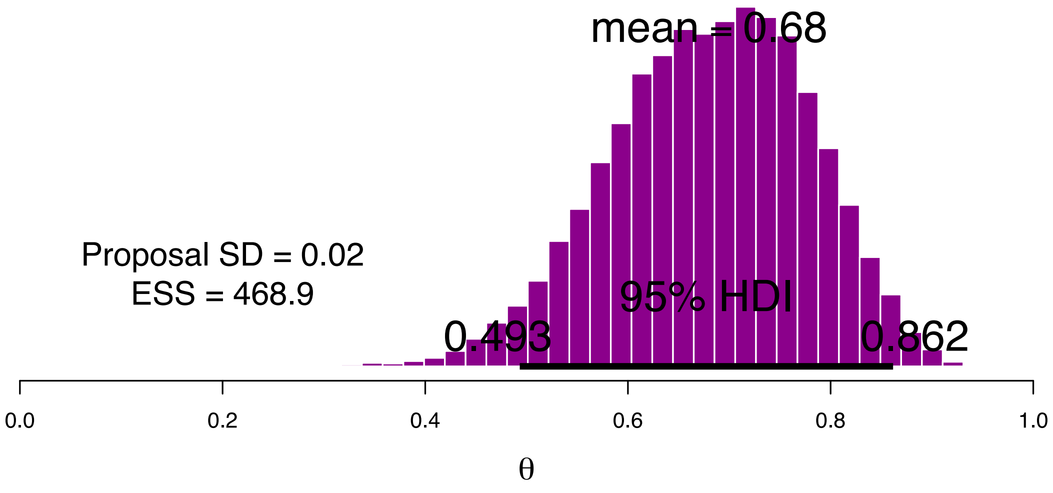
How do you choose σ for the proposal? There are two indicators that can be used to assess the quality of the sampling:

→ The ratio between the number of proposed moves and the number of accepted moves

→ The effective sample size (i.e., the number of moves that are not correlated with the previous ones)

Metropolis algorithm





Metropolis algorithm

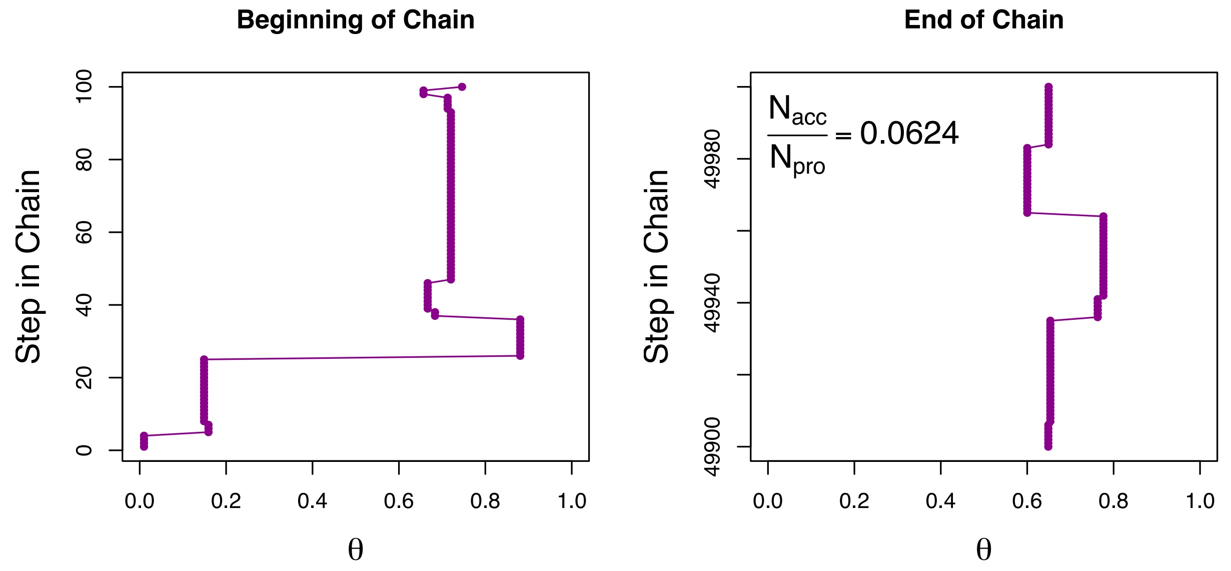
Influence of sigma

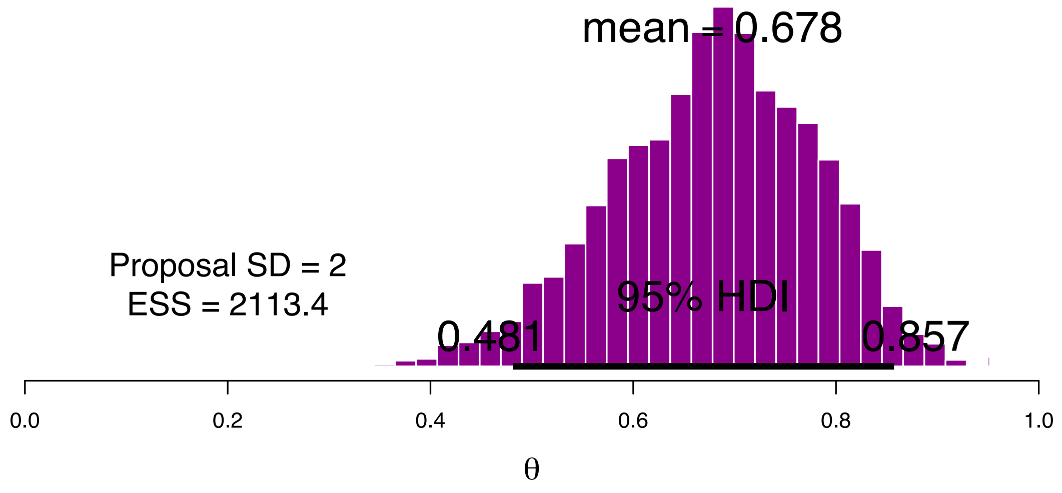
→ All (or almost all) proposals are accepted.

→ Few effective values...

It takes many iterations to get a satisfactory result...

Metropolis algorithm





Metropolis algorithm

Influence of sigma

→ Proposals are rarely accepted...

→ Few effective values...

Many iterations are needed to obtain a satisfactory result...

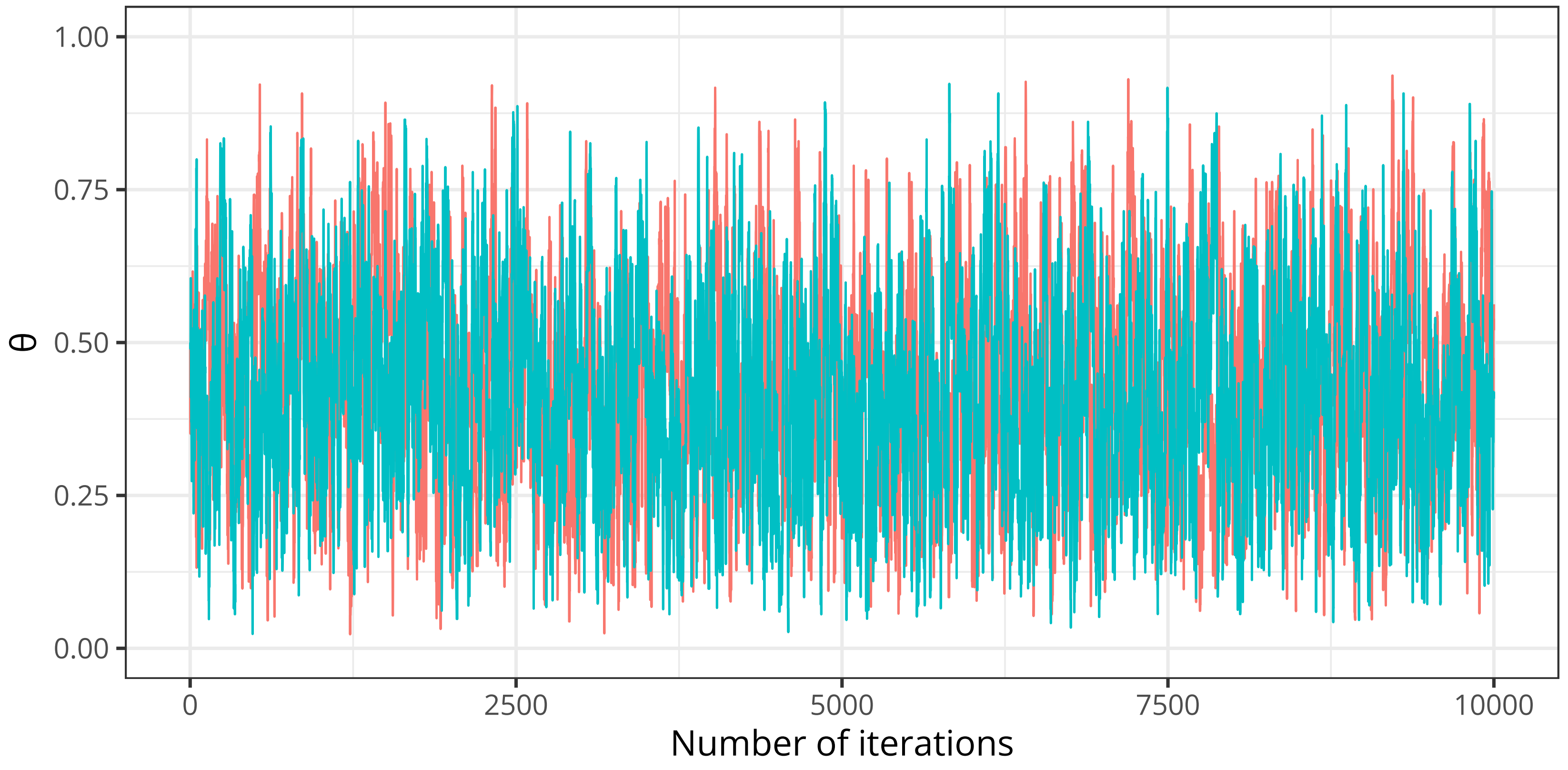
Metropolis algorithm[1]

```r
metropolis_beta_binomial <- function (niter = 1e2, startval = 0.5) {

    x <- rep(0, niter) # initialising the chain (vector) of length niter
    x[1] <- startval # defining the starting/initial value

    for (i in 2:niter) {

        current <- x[i - 1] # current value of the parameter
        current_plaus <- dbeta(current, 2, 3) * dbinom(1, 2, current)
        # proposal <- runif(n = 1, min = current - w, max = current + w) # proposed value
        proposal <- rnorm(n = 1, mean = current, sd = 0.1) # proposed value
        # ensuring that the proposed value is within the [0, 1] interval
        if (proposal < 0) proposal <- 0
        if (proposal > 1) proposal <- 1
        proposal_plaus <- dbeta(proposal, 2, 3) * dbinom(1, 2, proposal)
        # computing the probability of moving
        alpha <- min(1, proposal_plaus / current_plaus)
        # moving (or not) according to this probability
        x[i] <- sample(c(current, proposal), size = 1, prob = c(1 - alpha, alpha) )

    }

    return (x)

}
```

1. The Metropolis-Hastings algorithm is an extension of the Metropolis algorithm allowing for non-

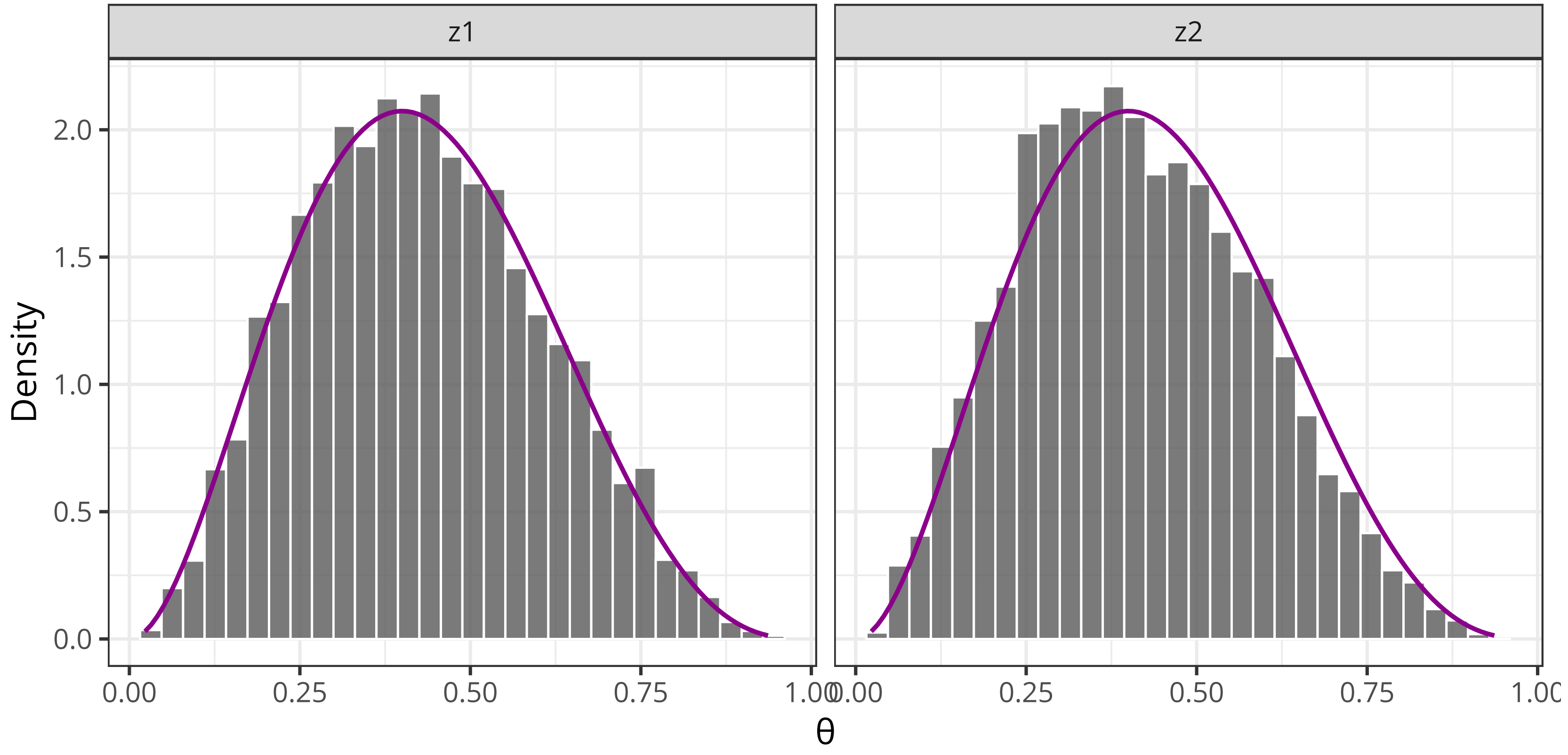
Metropolis algorithm

```r
z1 <- metropolis_beta_binomial(niter = 1e4, startval = 0.5)
z2 <- metropolis_beta_binomial(niter = 1e4, startval = 0.5)

data.frame(z1 = z1, z2 = z2) %>%
  mutate(sample = 1:nrow(.) ) %>%
  pivot_longer(cols = z1:z2) %>%
  ggplot(aes(x = sample, y = value, colour = name) ) +
  geom_line(show.legend = FALSE) +
  labs(x = "Number of iterations", y = expression(theta) ) + ylim(c(0, 1) )
```



Metropolis algorithm

```r
data.frame(z1 = z1, z2 = z2) %>%
  pivot_longer(cols = z1:z2) %>%
  rownames_to_column() %>%
  mutate(rowname = as.numeric(rowname) ) %>%
  ggplot(aes(x = value) ) +
  geom_histogram(aes(y = ..density..), color = "white", alpha = 0.8) +
  stat_function(fun = dbeta, args = list(3, 4), color = "magenta4", size = 1) +
  facet_wrap(~name) +
  labs(x = expression(theta), y = "Density")
```



Metropolis-Hastings algorithm

Hamiltonian Monte Carlo

The Metropolis and Metropolis-Hastings (or Gibbs) algorithms perform poorly when the model parameters are strongly correlated. The **Hamiltonian Monte Carlo** algorithm solves these problems by taking into account the geometry of the posterior space. We adapt the proposal to the geometry of the posterior distribution around the current position.

We use Hamiltonians which represent the total energy of a system. This energy is broken down into its **potential energy** (which depends on its position in parameter space) and its **kinetic energy**, which depends on its **momentum** m:

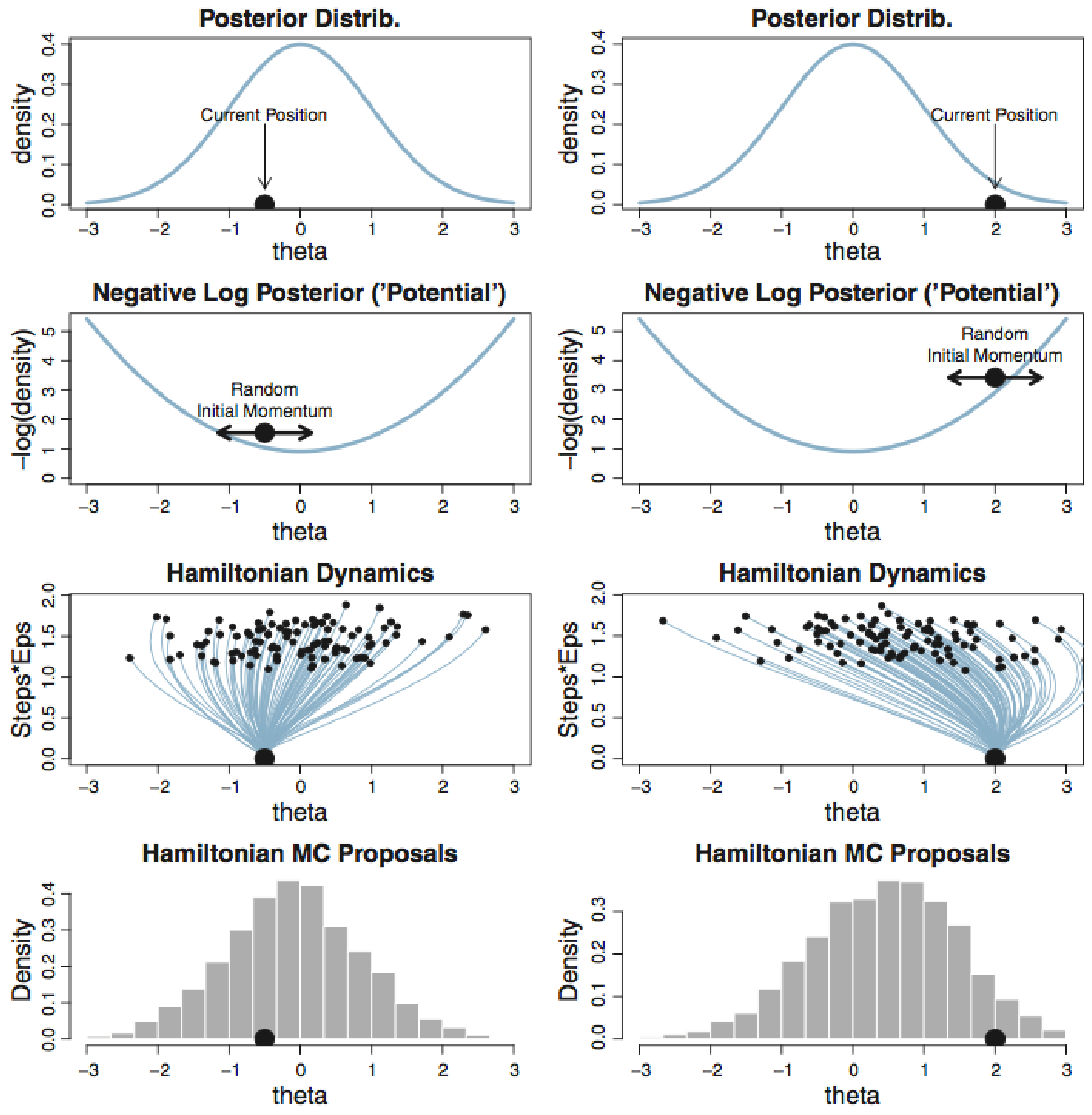
$$H(\theta, m) = \underbrace{U(\theta)}_{\text{potential energy}} + \underbrace{KE(m)}_{\text{kinetic energy}}$$

The potential energy is given by the negative of the log of the posterior density (non-normalised):

$$U(\theta) = -\log[p(\text{data} \mid \theta) \times p(\theta)]$$

As the posterior density increases, the potential energy decreases (i.e., it becomes more negative).

Hamiltonian Monte Carlo



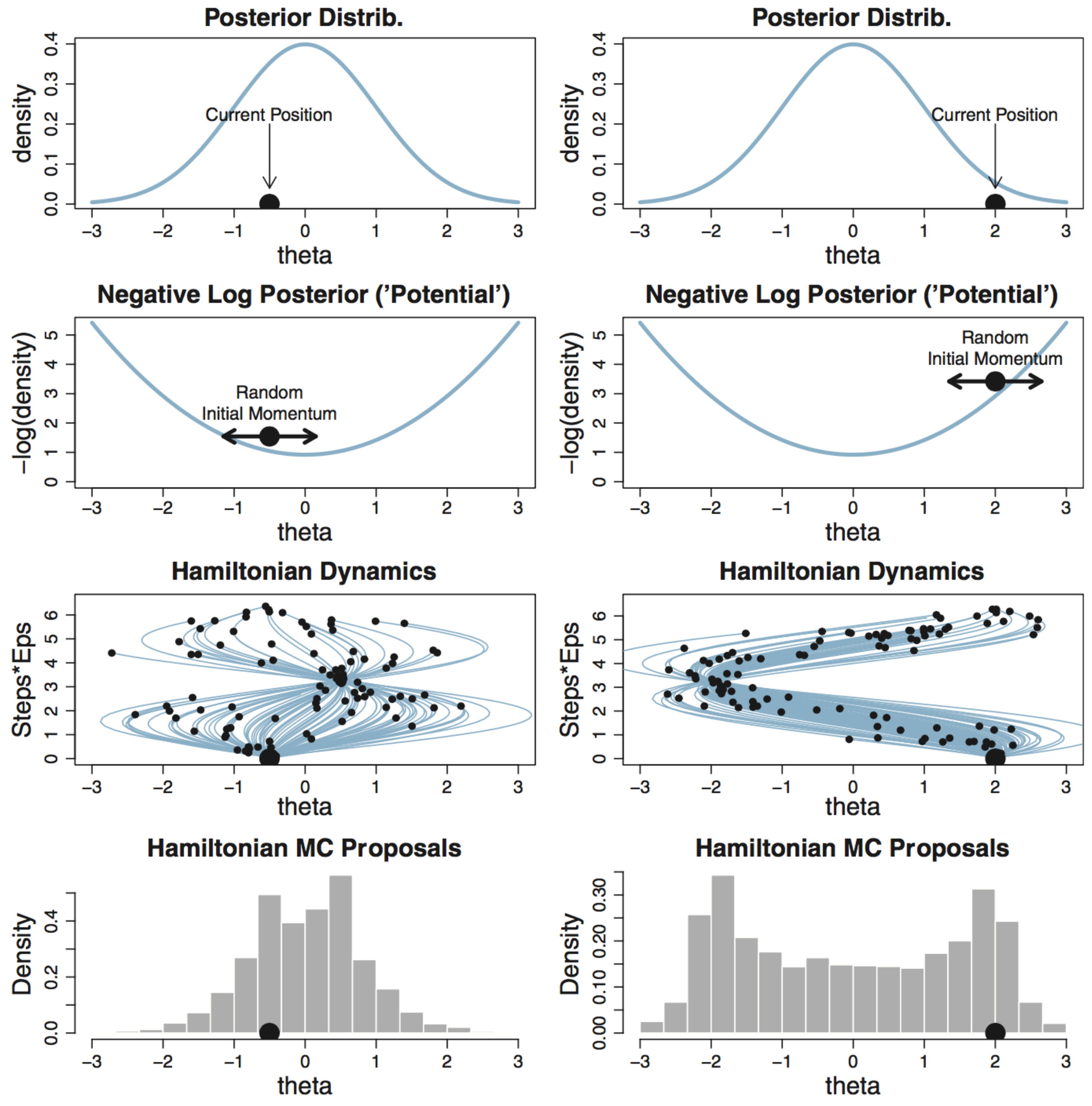
Hamiltonian Monte Carlo

- Select a starting point θ_0: Any value of θ in posterior space can be selected.
- The force with which the ball is thrown (momentum m) is randomly generated, for example from a multivariate normal distribution: $m \sim \text{MVNormal}(\mu, \Sigma)$.
- A trajectory approximation algorithm (e.g., leapfrog) is used to estimate the trajectory and final position of the ball in posterior space for a given trajectory duration.
- After a certain time, the final position of the ball and its moment are recorded.
- The proposed movement is accepted or rejected according to the following probability (where ϕ (phi) is the momentum associated with the marble):

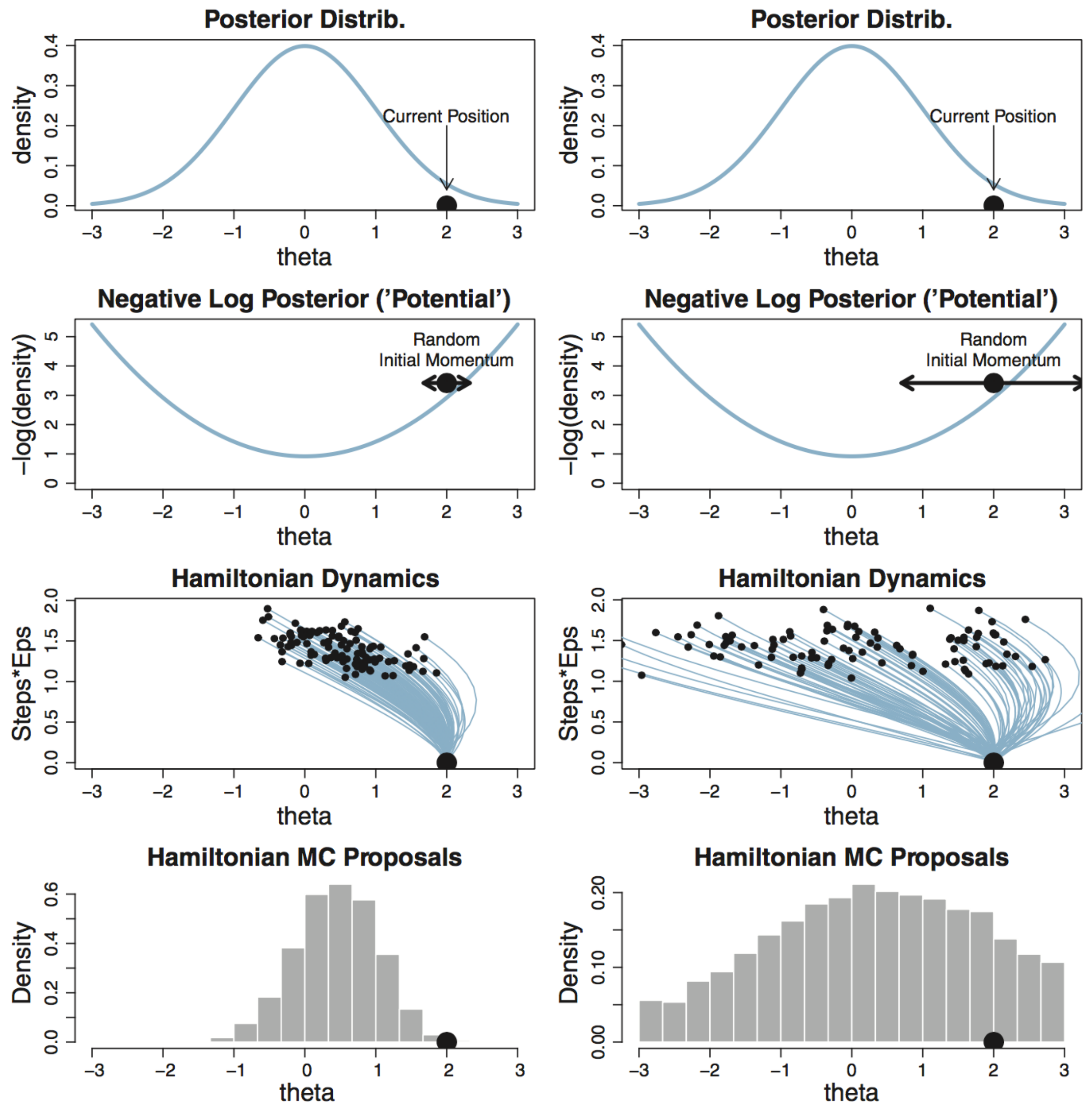
$$\Pr_{\text{move}} = \min\left(\frac{p(\text{data} \mid \theta_{\text{proposed}})\, p(\phi_{\text{proposed}})}{p(\text{data} \mid \theta_{\text{current}})\, p(\phi_{\text{current}})}, 1\right)$$

- We save the new position and start again…

Influence of trajectory duration...



Influence of variability in the initial momentum...



Hamiltonian Monte Carlo

Assessing MCMCs

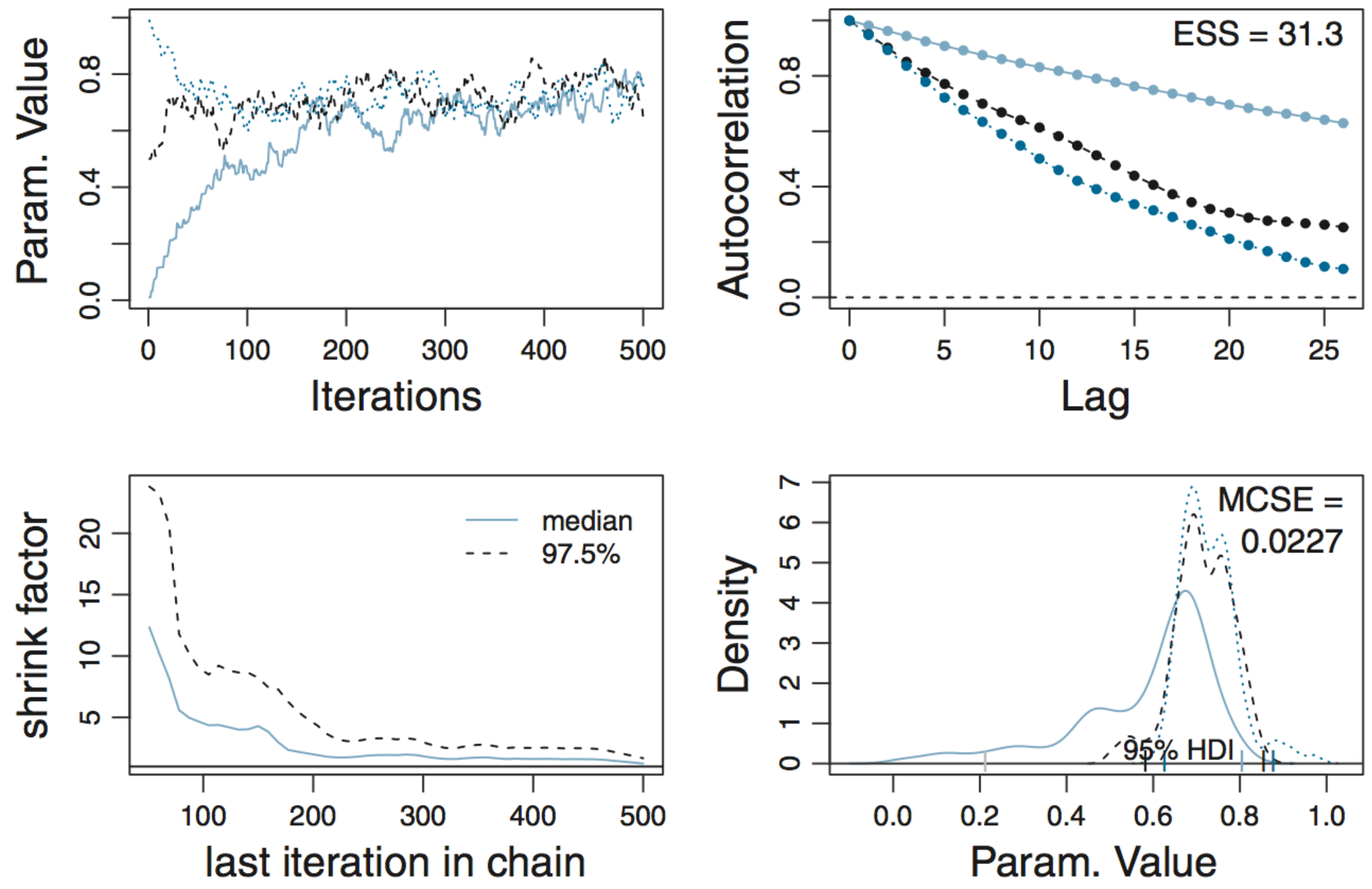
These methods may not converge to the “true” posterior distribution, due to limited computation time, the parametrisation of certain hyper-parameters (e.g., variance of the normal distribution of the proposal, or variance of the initial moment for HMC).

These methods produce chains of parameter values (samples). The use of a particular MCMC algorithm to sample the posterior distribution is based on three objectives:

- The chain values must be representative of the posterior distribution. These values must not depend on the starting point. The values should not be restricted to a particular region of the parameter space.
- The chain must be long enough to ensure the accuracy and stability of the result. The central tendency and HDI calculated from the chain must not change if the procedure is restarted.
- The chain should be generated efficiently (i.e., with as few iterations as possible).

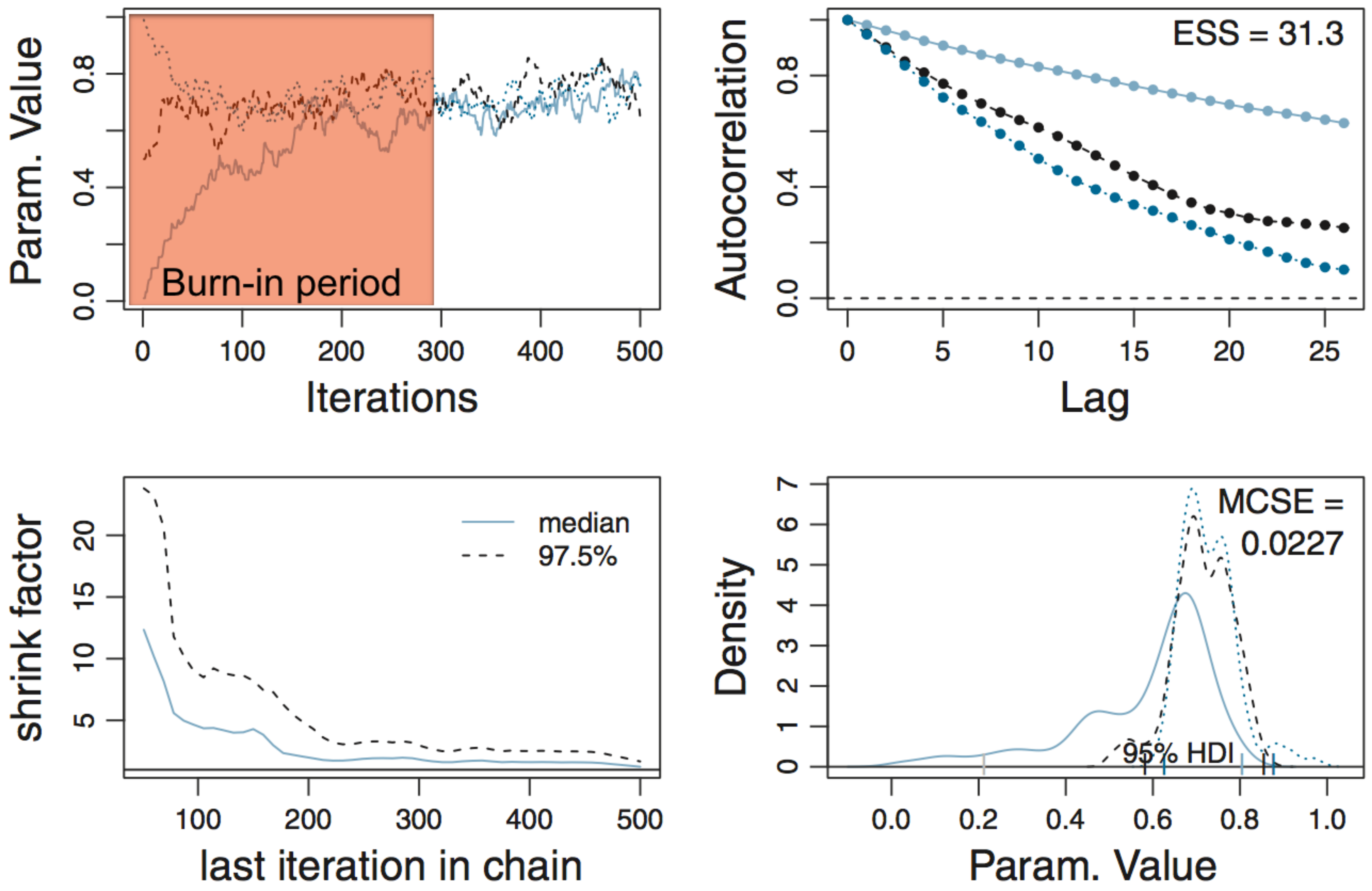
Assessing MCMCs - Representativeness

- Visual verification of trajectories: Chains must occupy the same space, convergence does not depend on the starting point, no chain must have a particular trajectory (e.g., cyclic).
- Visual check of densities: Densities must overlap.



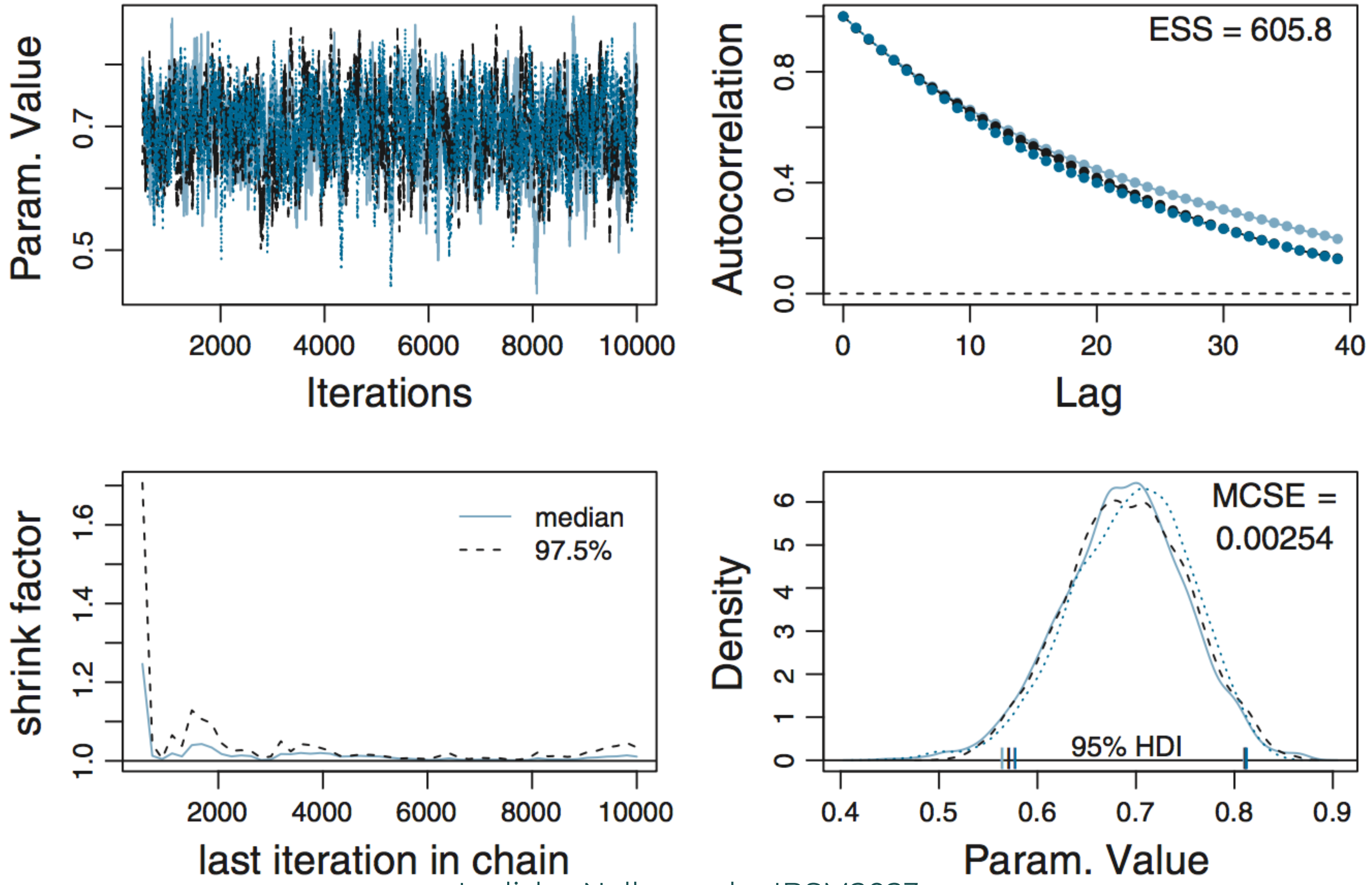
Assessing MCMCs - Representativeness

This display shows only the first 500 iterations. The trajectories do not overlap at the beginning (orange zone). The density is also affected. In practice, these first iterations are suppressed ("burn-in" or "warm-up" period).



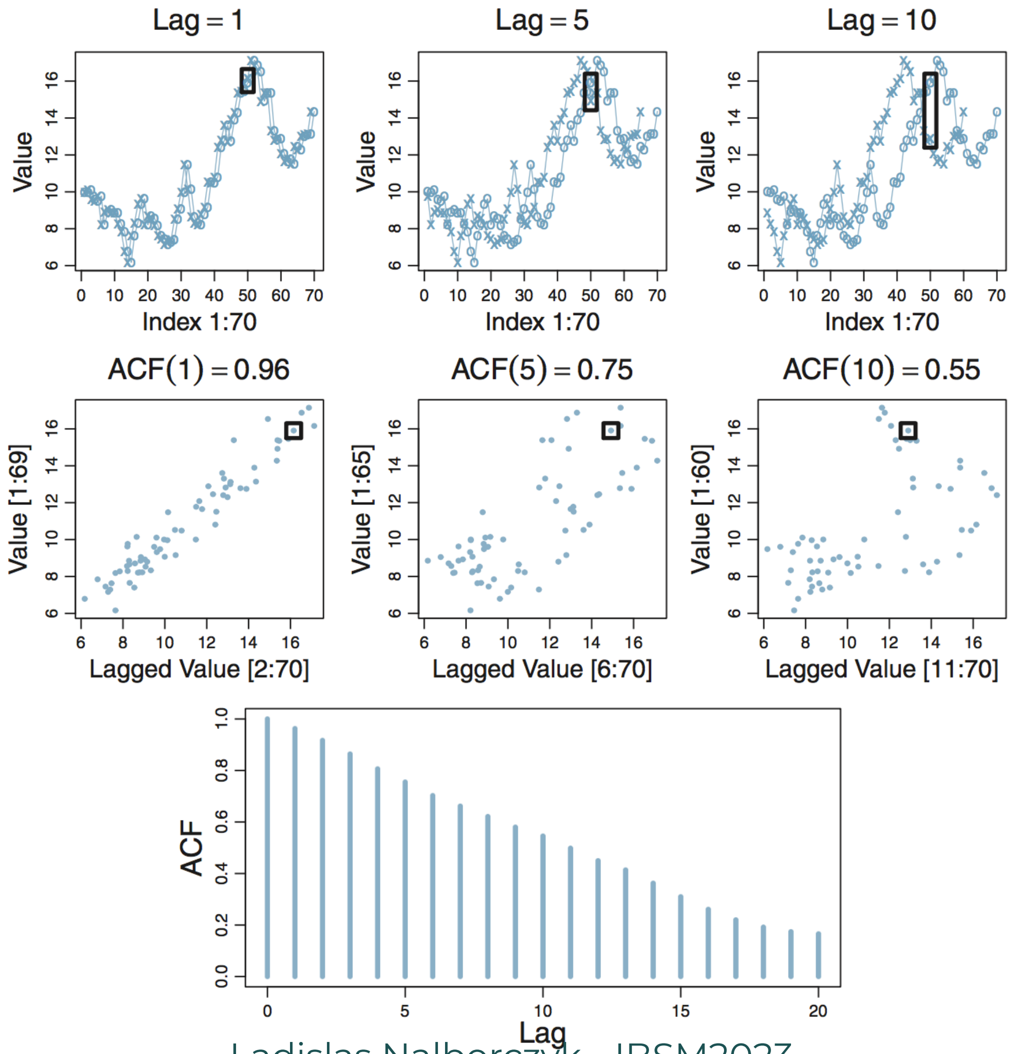
Assessing MCMCs - Representativeness

Numerical verification of chains: The **shrink factor** (also known as $\hat{R}$ or `Rhat`) is the ratio between the inter-chain and intra-chain variance. This value should ideally tend towards 1 (it is considered acceptable up to 1.01).



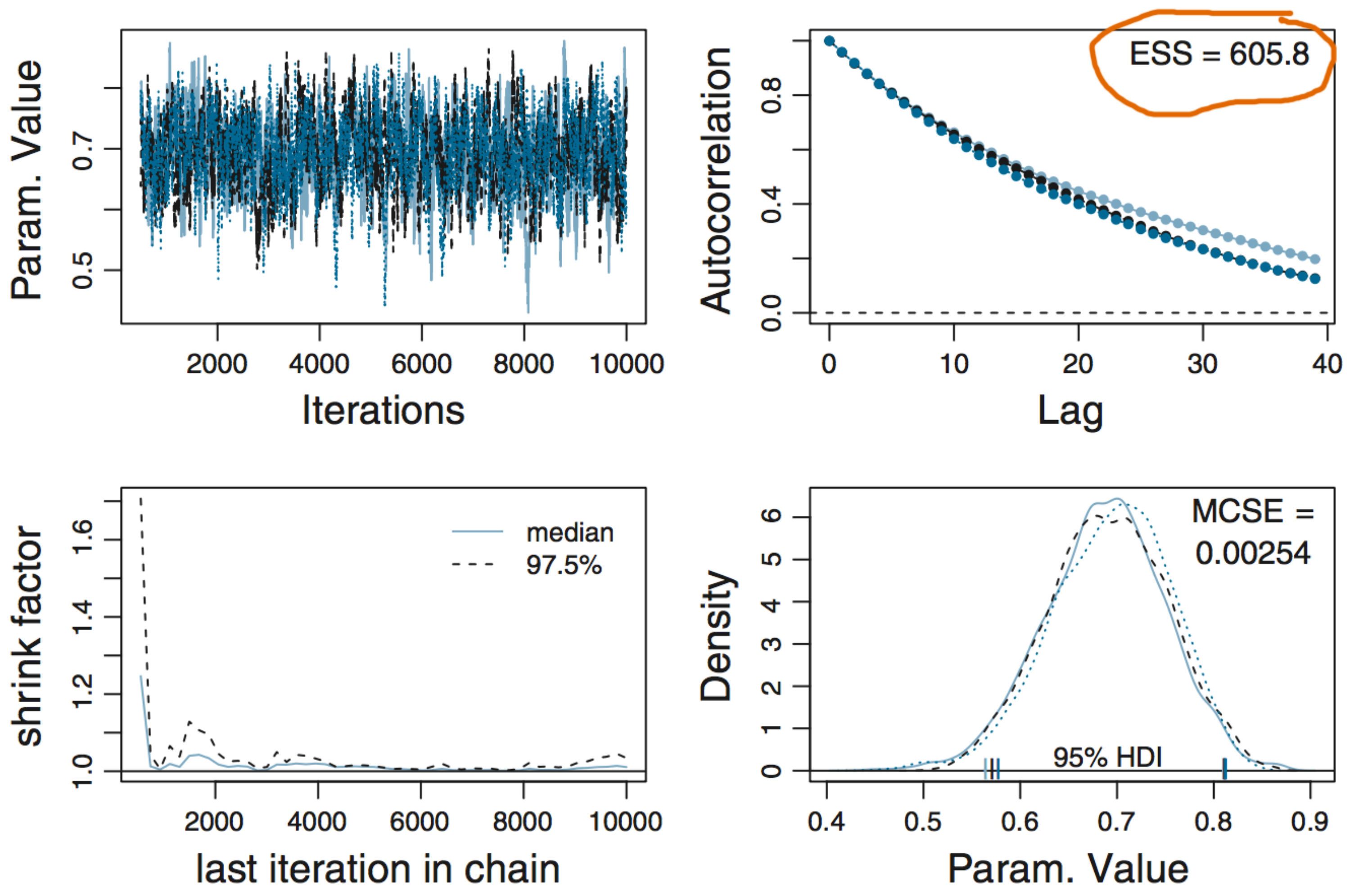
Assessing MCMCs - Stability and precision

The longer the chain, the more accurate and stable the result. If the chain "lingers" on each position, and the number of iterations remains the same, then we lose precision. It will need more iterations to achieve the same level of accuracy. Autocorrelation is the correlation of the chain with itself but shifted by k iterations (lag).



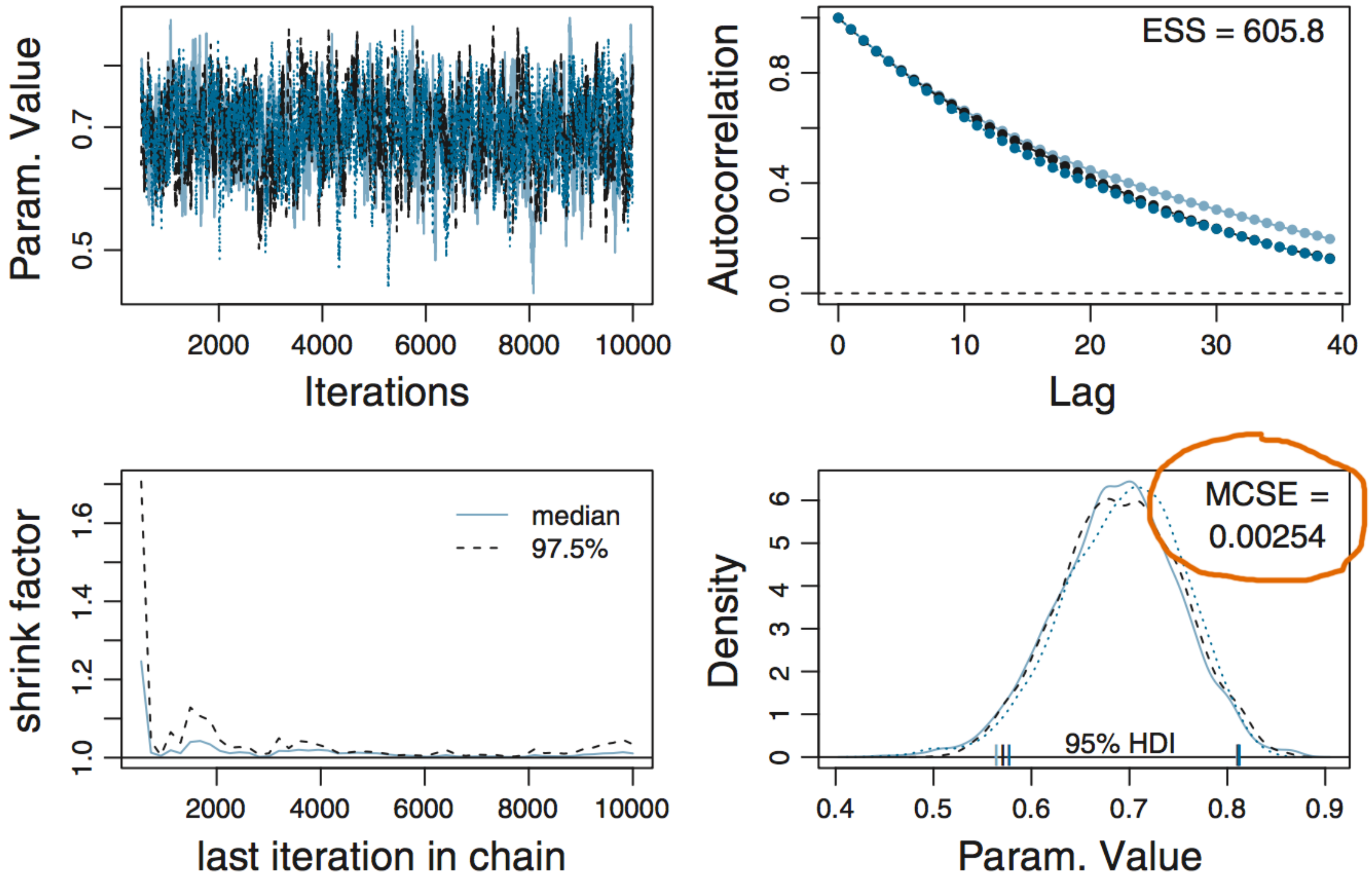
Assessing MCMCs - Stability and precision

The autocorrelation function is shown for each chain (top right). Another result reflects the precision of the sample: the effective sample size, $\text{ESS} = \frac{N}{1+2\sum_k \text{ACF}(k)}$. It represents the size of a non-autocorrelated sample extracted from the sum of all the chains. For reasonable HDI accuracy, an ESS greater than 1000 is recommended.



Assessing MCMCs - Stability and precision

The standard error of a set of samples is given by : $SE = SD/\sqrt{N}$. As N increases, the standard error decreases. We can generalise this idea to Markov chains: $MCSE = SD/\sqrt{ESS}$. For the central tendency to be reasonably accurate, this value must be low.

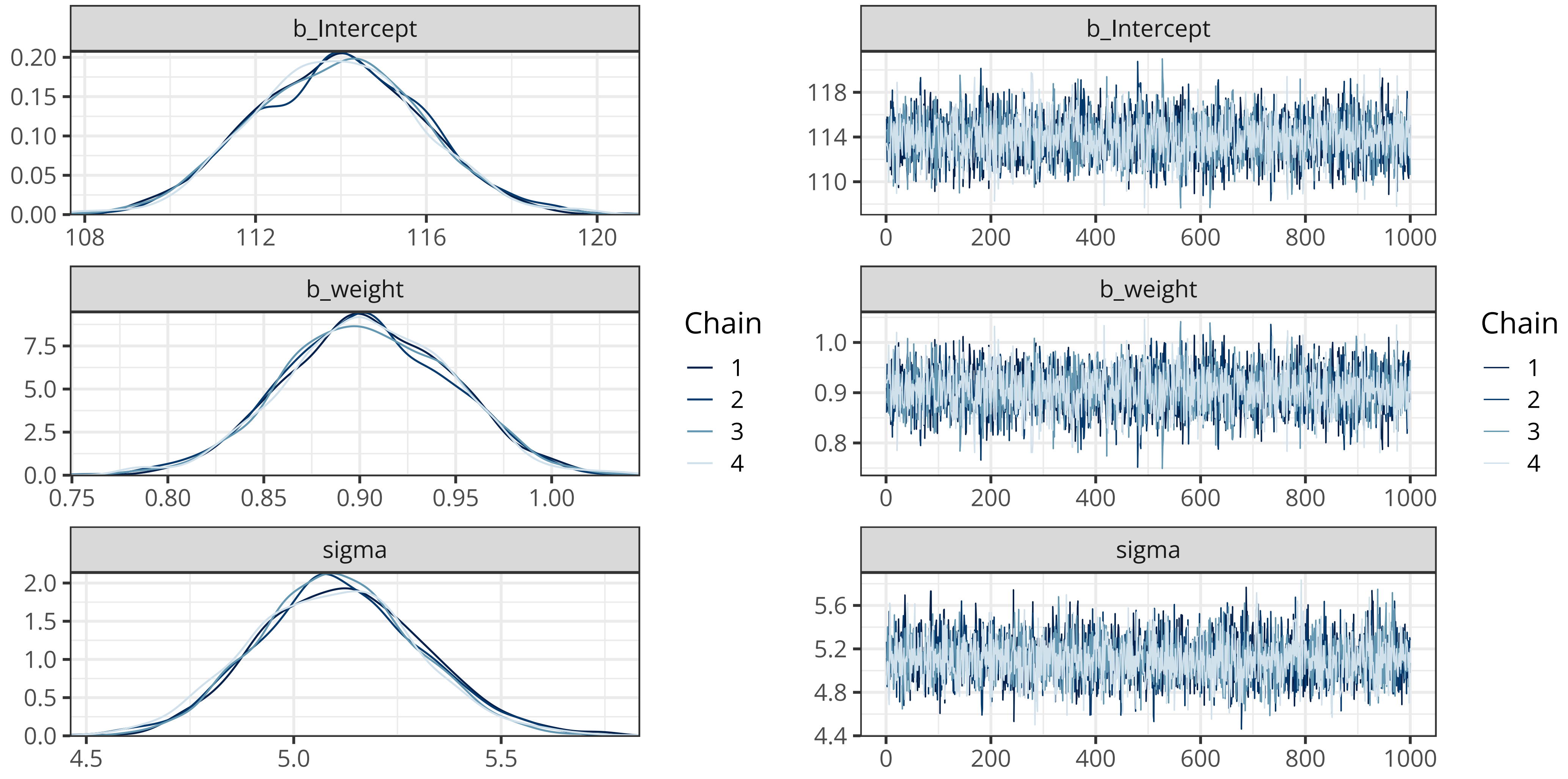


Assessing MCMCs - brms implementation

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

d <- open_data(howell)
d2 <- d %>% filter(age >= 18)

priors <- c(
  prior(normal(150, 20), class = Intercept),
  prior(normal(0, 10), class = b),
  prior(exponential(0.01), class = sigma)
  )

mod1 <- brm(
  formula = height ~ 1 + weight,
  prior = priors,
  family = gaussian(),
  data = d2,
  chains = 4, # number of chains
  iter = 2000, # total number of iteration (per chain)
  warmup = 1000, # number of warm-up iterations
  thin = 1 # thinning (1 = no thinning)
  )
```

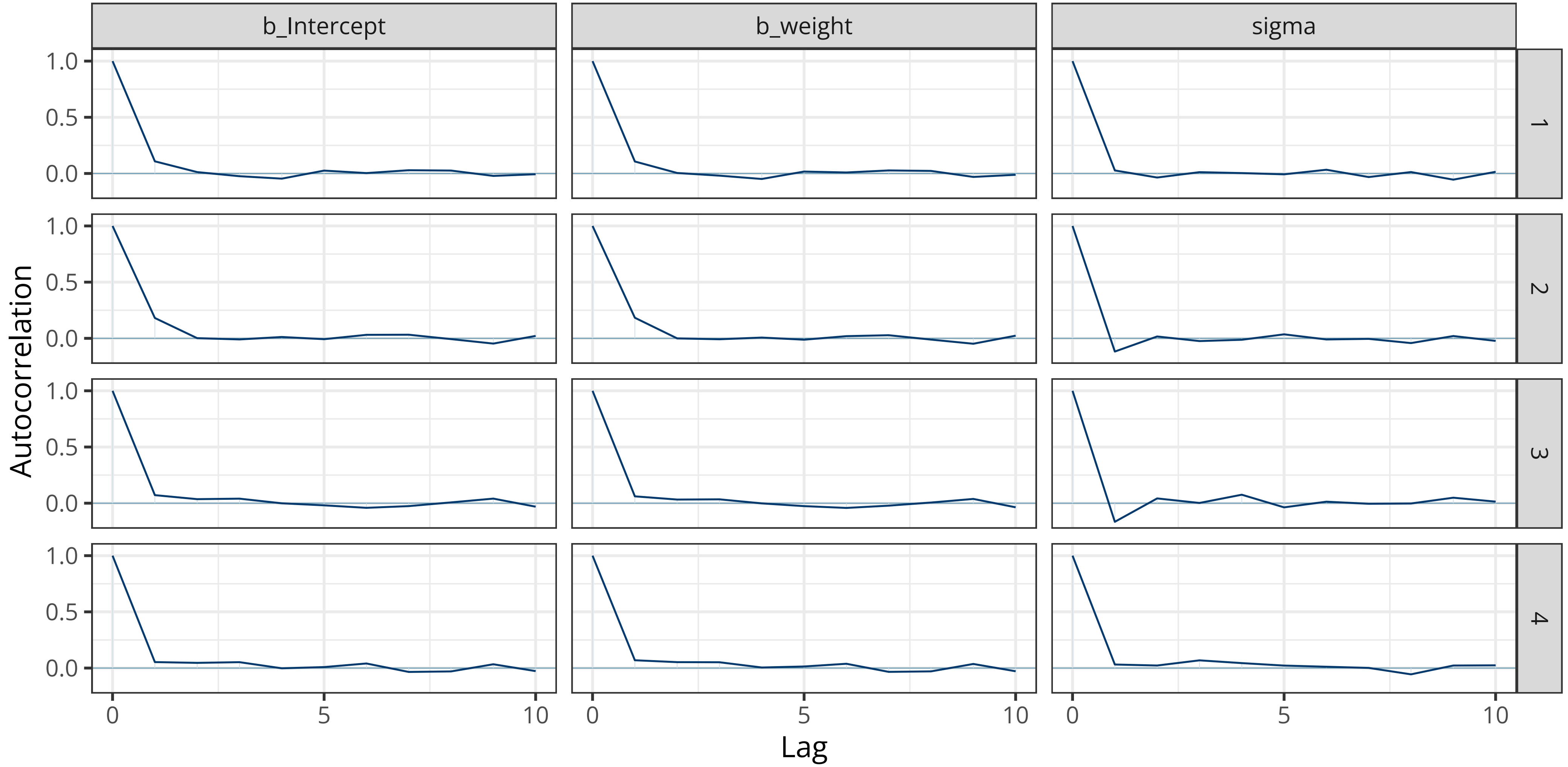
Assessing MCMCs - brms implementation

```r
# combo can be hist, dens, dens_overlay, trace, trace_highlight...
# cf. https://mc-stan.org/bayesplot/reference/MCMC-overview.html
plot(x = mod1, combo = c("dens_overlay", "trace") )
```



Assessing MCMCs - brms implementation

```r
library(bayesplot)
post <- posterior_samples(mod1, add_chain = TRUE)
post %>% mcmc_acf(pars = vars(b_Intercept:sigma), lags = 10)
```



Assessing MCMCs - brms implementation

```r
summary(mod1)
```

```
 Family: gaussian 
  Links: mu = identity; sigma = identity 
Formula: height ~ 1 + weight 
   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   113.90      1.95   110.18   117.70 1.00     3135     2949
weight        0.90      0.04     0.82     0.99 1.00     3140     2817

Family Specific Parameters: 
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     5.11      0.20     4.74     5.51 1.00     3940     2887

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).
```

Assessing MCMCs - brms implementation

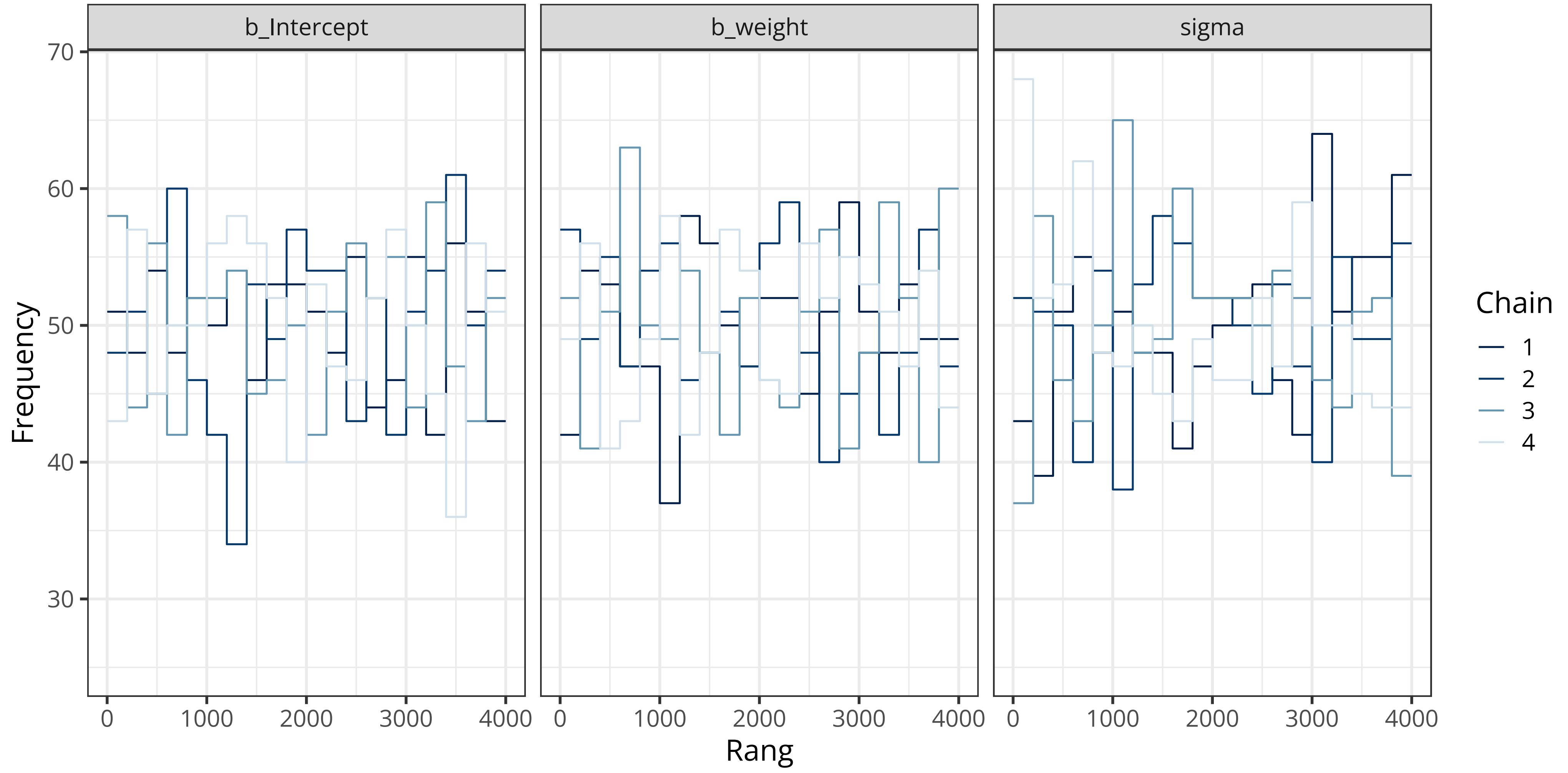
Bulk-ESS refers to the ESS calculated on the distribution of samples normalised by their rank, and more specifically around the central position of this distribution (e.g., mean or median). It is recommended that the Bulk-ESS be at least 100 times greater than the number of chains (i.e., for 4 chains, the Bulk-ESS should be at least 400).

Tail-ESS gives the minimum of the ESS calculated for the quantiles at 5% and 95% (i.e., for the tails of the distribution of samples normalised by their rank). This value must be high if we attach importance to estimating extreme values (for example to compute credible intervals).

When things go wrong, see these recommendations from Stan's team about priority choices, or this guide about frequent error messages. See also recent article or this blog post introducing these new tools.

Assessing MCMCs - brms implementation

```r
post %>% # rank plots
  mcmc_rank_overlay(pars = vars(b_Intercept:sigma) ) +
  labs(x = "Rang", y = "Frequency") +
  coord_cartesian(ylim = c(25, NA) )
```



Summary

We have introduced and discussed the use of MCMCs to obtain samples from the (un-normalised) posterior distribution. These samples can then be used to calculate various statistics for the posterior distribution (e.g., mean, median, credible interval).

The Metropolis-Hastings algorithm can be used for any problem for which a likelihood can be calculated. However, although this algorithm is simple to code, its convergence can be very slow… Furthermore, this algorithm does not work well when there are strong correlations between the different parameters…

The HMC algorithm avoids these problems by taking into account the geometry of the posterior space as it is explored (i.e., when the algorithm decides where to go next). This algorithm converges much faster and fewer samples will be needed to approximate the posterior distribution.

The result of Bayesian inference is therefore, in practice, a set of samples obtained using MCMCs. The reliability of these estimates must be assessed by verifying (visually and numerically) that the MCMCs have indeed converged towards an optimal solution.

Generalised linear model

Introduction

$$y_i \sim \mathrm{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta_1 \times x_{1i} + \beta_2 \times x_{2i}$$

The linear Gaussian model discussed in Course n°02 is characterised by a number of assumptions, including the following:

- The residuals are distributed according to a Normal distribution.
- The variance of this Normal distribution is constant (homogeneity of variance).
- The predictors act on the mean of this distribution.
- The mean follows a linear or multi-linear model.

Introduction

This model (the choice of a Normal distribution) implies several constraints, for example:

- The observed data are defined on a continuous space.
- This space is not bounded.

How can we model data that do not respect these constraints? For example, the proportion of correct answers to a test (bounded between 0 and 1), a response time (restricted to positive values and often distributed in an approximately lognormal manner), a number of avalanches…

Introduction

We have already encountered a different model: the Beta-Binomial model (cf. Course n°01).

$$
\begin{aligned}
y_i &\sim \mathrm{Binomial}(n, p) \\
p &\sim \mathrm{Beta}(a, b)
\end{aligned}
$$

- The observed data is binary (e.g., 0 vs. 1) or the result of a sum of binary observations (e.g., a proportion).
- The prior probability of success (obtaining 1) is characterised by a Beta distribution. - The probability of success does not depend on any predictor.

Introduction

This model implies two constraints:

- The observed data are defined in a discrete space.
- This space is bounded.

How could predictors be added to this model?

Generalised linear model

$$
\begin{aligned}
y_i &\sim \mathrm{Binomial}(n, p_i) \\
f(p_i) &= \alpha + \beta \times x_i
\end{aligned}
$$

Objectives:

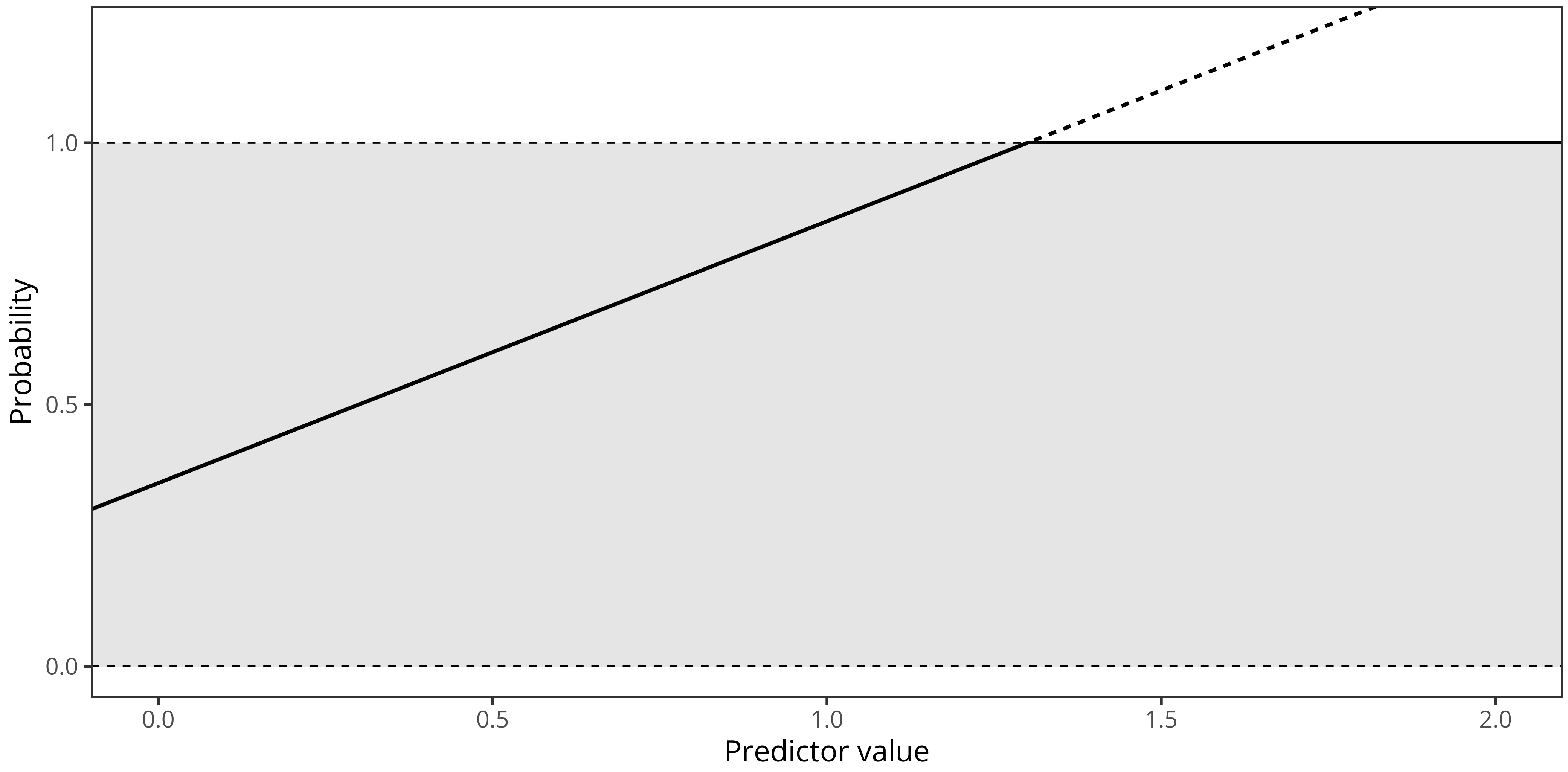
- Accounting for discrete data (e.g., failure/success) generated by a single process.
- Introducing predictors into the model.

Two changes from the Gaussian model:

- The use of a Binomial probability distribution.
- The linear model is no longer used to directly describe one of the parameters of the distribution, but a function of this parameter (the Gaussian model can also be considered to have been formulated with an identity link function).

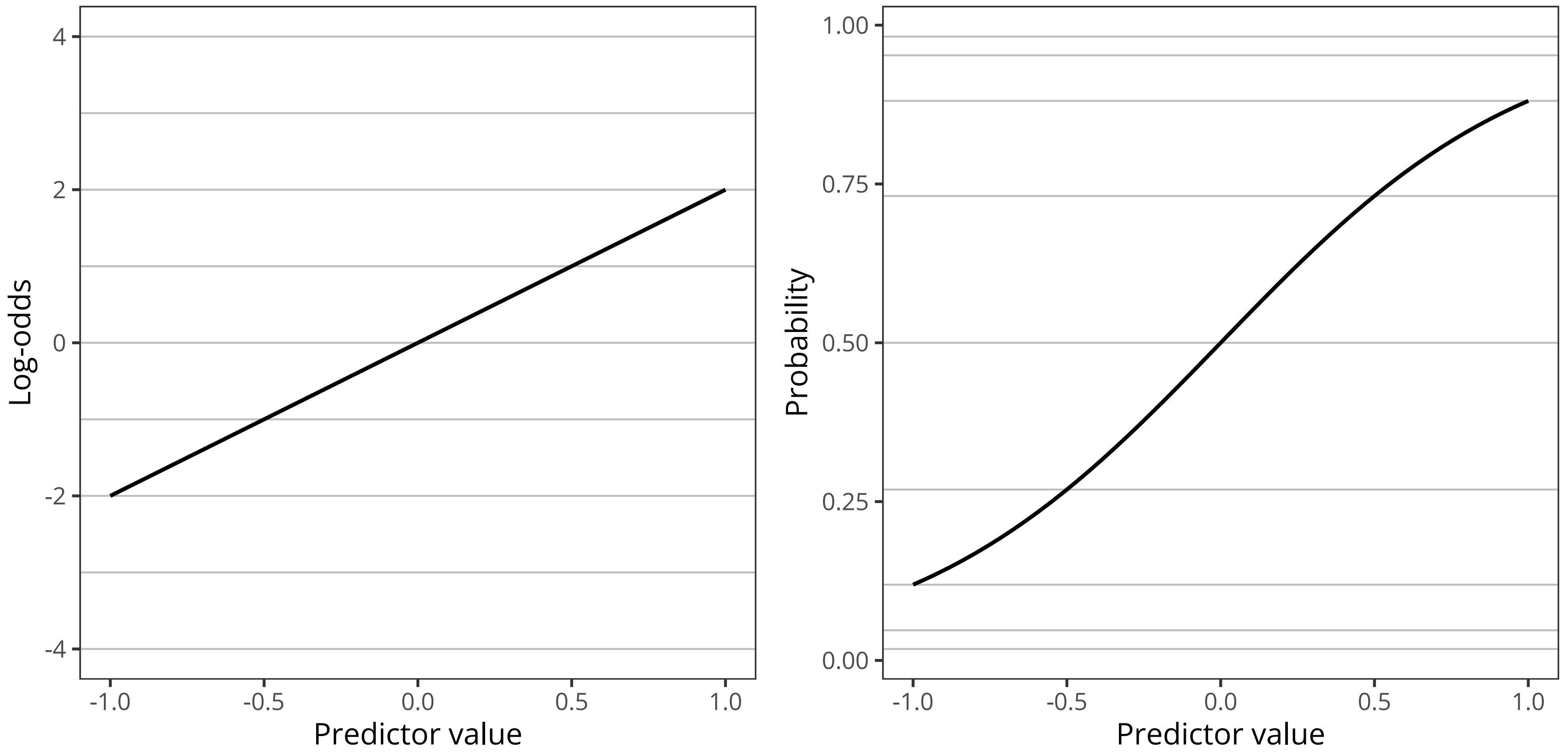
Link function

We use a link function to map the space of a linear (unbounded) model to the space of a potentially bounded parameter such as a probability, defined on the interval $[0, 1]$.



Link function

We use a link function to map the space of a linear (unbounded) model to the space of a potentially bounded parameter such as a probability, defined on the interval $[0, 1]$.



Logistic regression

The logit function of the binomial GLM (known as “log-odds”):

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1 - p_i}\right)$$

The odds of an event are the ratio between the probability of the event occurring and the probability of it not occurring. The logarithm of this odds is predicted by a linear model.

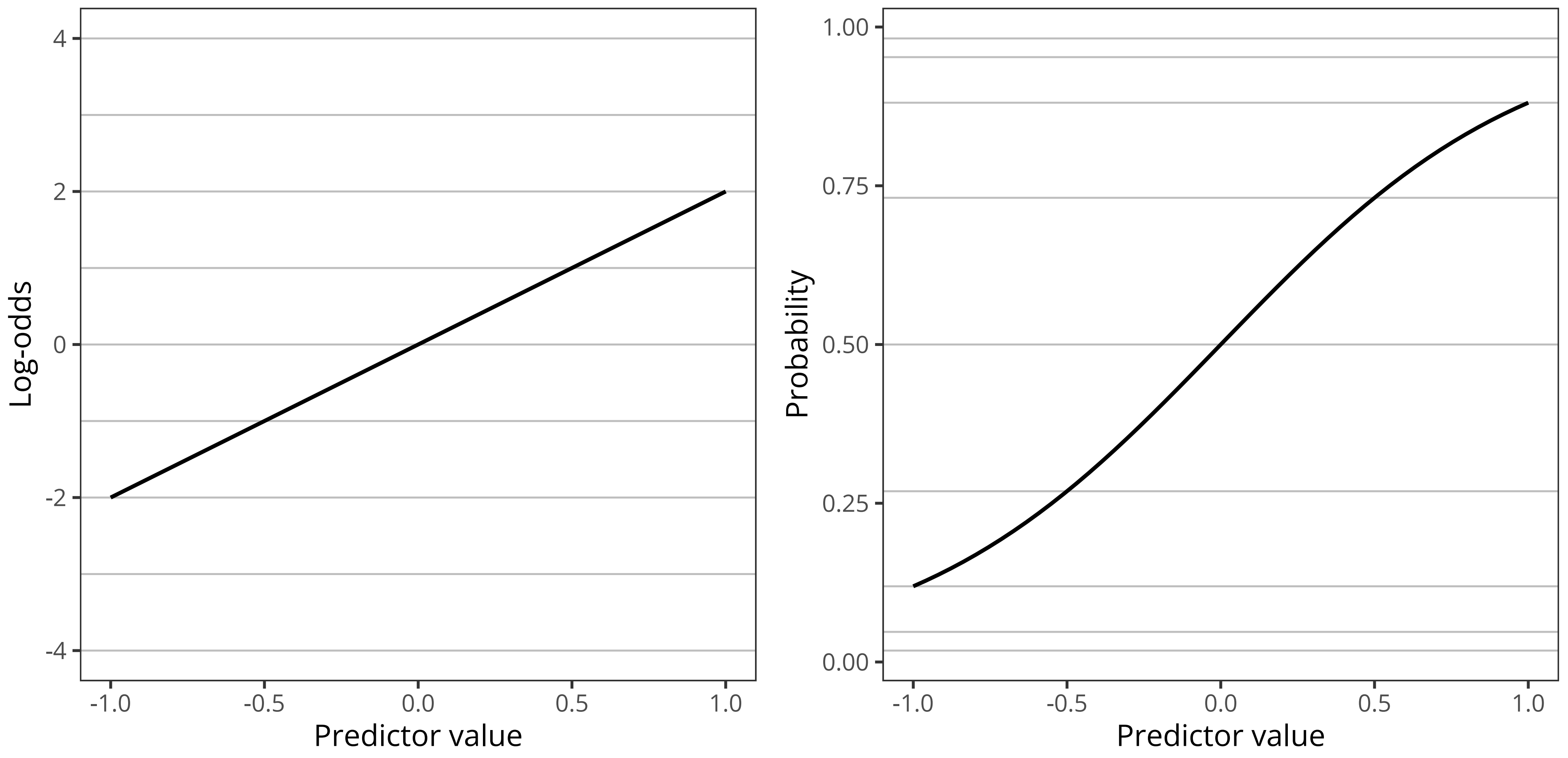
$$\log\left(\frac{p_i}{1 - p_i}\right) = \alpha + \beta \times x_i$$

To retrieve the probability of an event, we use the **inverse link** function, the **logistic** (or logit-inverse) function:

$$p_i = \frac{\exp(\alpha + \beta \times x_i)}{1 + \exp(\alpha + \beta \times x_i)}$$

Complications caused by the link function

Such link functions pose problems of interpretation: a change of one unit in a predictor no longer has a constant effect on the probability but impacts it more or less according to its distance from the origin. When $x = 0$, an increase of half a unit (i.e., $\Delta x = 0.5$) results in an increase in probability of 0.25. Then, each half-unit increase results in a smaller and smaller increase in p...



Complications caused by the link function

Second complication: this link function "forces" each predictor to interact with itself and with ALL the other predictors, even if the interactions are not explicit...

In a Gaussian model, the rate of change of y as a function of x is given by $\partial(\alpha + \beta x) / \partial x = \beta$ and does not depend on x (i.e., β is constant).

In a binomial GLM (with a logit link function), the probability of an event is given by the logistic function:

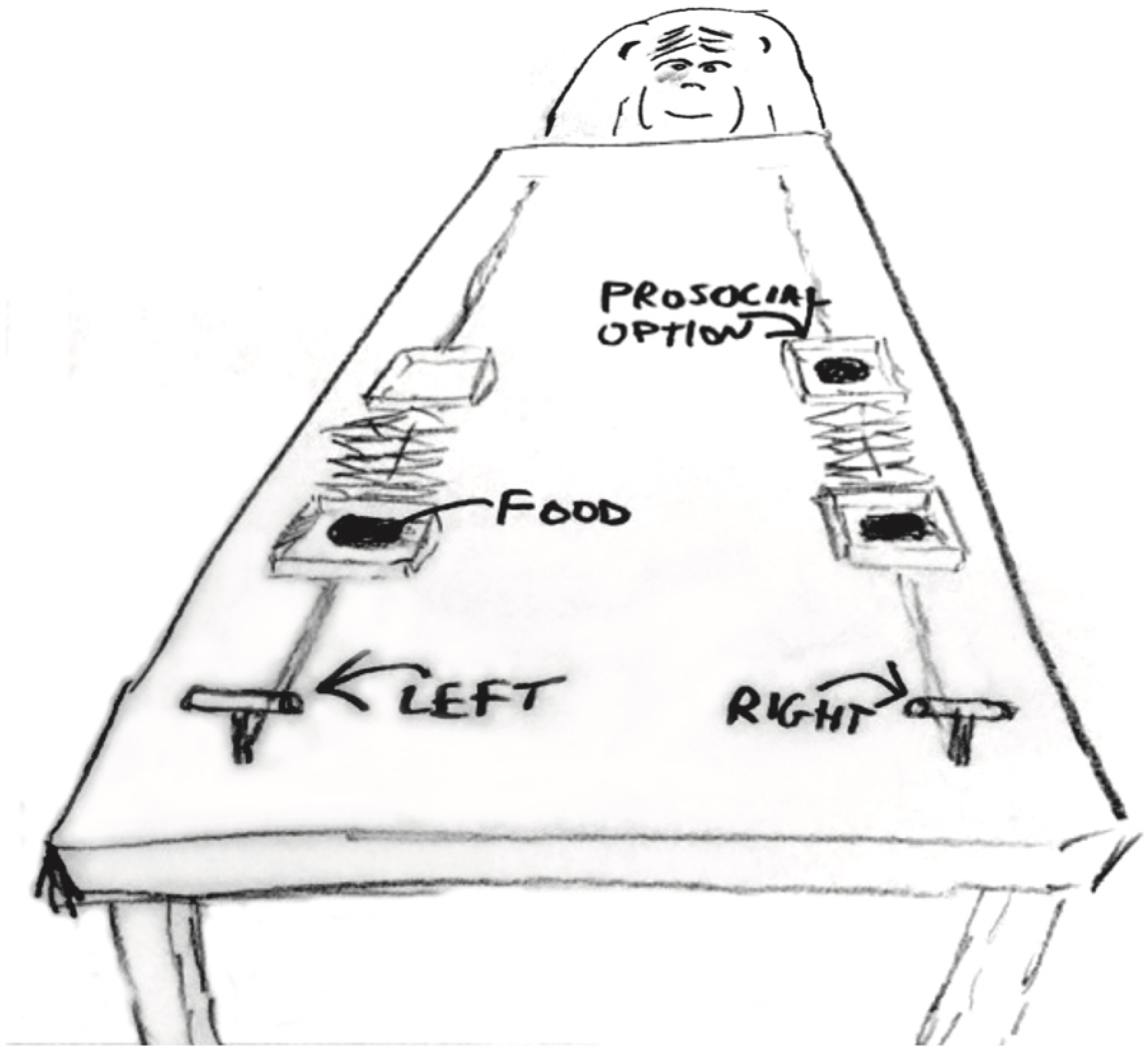
$$p_i = \frac{\exp(\alpha + \beta \times x_i)}{1 + \exp(\alpha + \beta \times x_i)}$$

And the rate of change of p as a function of the predictor x is given by:

$$\frac{\partial p}{\partial x} = \frac{\beta}{2(1 + \cosh(\alpha + \beta \times x))}$$

We can see that the variation on p due to the predictor x is a function of the predictor x, and also depends on the value of α... !

Logistic regression example: Prosociality in chimpanzees



Logistic regression example

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

df1 <- open_data(chimpanzees)
str(df1)
```

```
'data.frame':	504 obs. of  8 variables:
 $ actor       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ recipient   : int  NA NA NA NA NA NA NA NA NA NA ...
 $ condition   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ block       : int  1 1 1 1 1 1 2 2 2 2 ...
 $ trial       : int  2 4 6 8 10 12 14 16 18 20 ...
 $ prosoc_left : int  0 0 1 0 1 1 1 1 0 0 ...
 $ chose_prosoc: int  1 0 0 1 1 1 0 0 1 1 ...
 $ pulled_left : int  0 1 0 0 1 1 0 0 0 0 ...
```

- **pulled_left**: 1 when the chimpanzee pulled the left lever, 0 otherwise.
- **prosoc_left**: 1 when the left lever was associated with the prosocial option, 0 otherwise.
- **condition**: 1 when a partner was present, 0 otherwise.

Logistic regression example

The question

We want to know whether the presence of a partner chimpanzee encourages the chimpanzee to press the prosocial lever, that is, the lever that gives food to both individuals. In other words, is there an interaction between the effect of laterality and the effect of the presence of another chimpanzee on the probability of pulling the left lever?

The variables

- Observations (`pulled_left`): These are Bernoulli variables (0 or 1).
- Predictor (`prosoc_left`): Are the two meals on the left or the right?
- Predictor (`condition`): Is a partner present?

Logistic regression example

$$
\begin{aligned}
L_i &\sim \mathrm{Binomial}(1, p_i) \\
\text{(equivalent to)} \quad L_i &\sim \mathrm{Bernoulli}(p_i) \\
\mathrm{logit}(p_i) &= \alpha \\
\alpha &\sim \mathrm{Normal}(0, \omega)
\end{aligned}
$$

Mathematical model without any predictor. How do you pick a value for ω...?

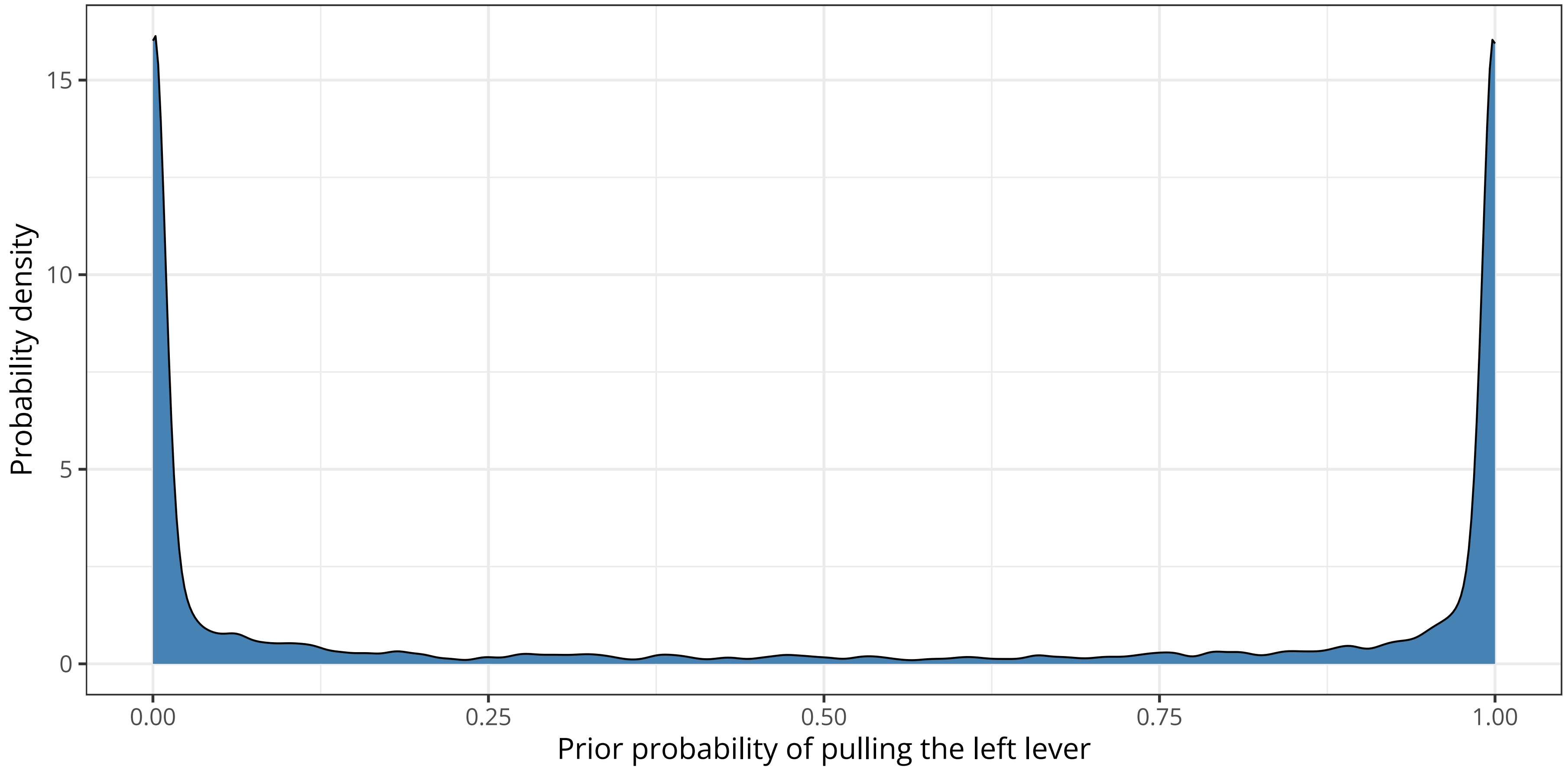
Prior predictive check

We write the previous model with `brms` and sample from the prior to check that the model's predictions (based on the prior and likelihood function alone) match our expectations.

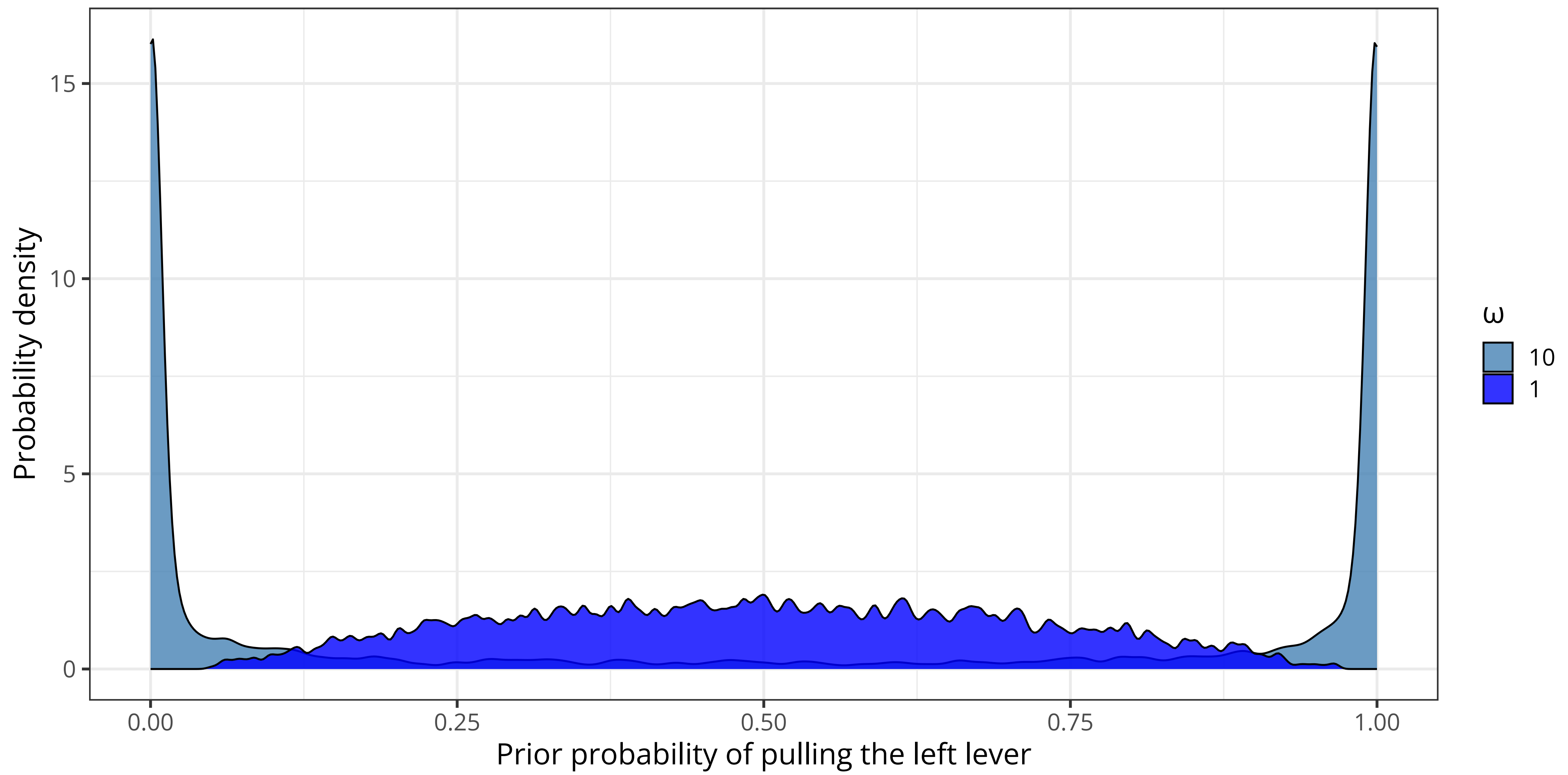
```r
library(brms)

mod1.1 <- brm(
  # "trials" allows defining the number of trials (i.e., n)
  formula = pulled_left | trials(1) ~ 1,
  family = binomial(),
  prior = prior(normal(0, 10), class = Intercept),
  data = df1,
  # we samples from the prior
  sample_prior = "yes"
  )
```

Prior predictive check

```r
# retrieving samples from the prior predictive distribution
prior_draws(x = mod1.1) %>%
  # applying the inverse link function
  mutate(p = brms::inv_logit_scaled(Intercept) ) %>%
  ggplot(aes(x = p) ) +
  geom_density(fill = "steelblue", adjust = 0.1) +
  labs(x = "Prior probability of pulling the left lever", y = "Probability density")
```



Prior predictive check



Logistic regression

The intercept is interpreted in the log-odds space… to interpret it as a probability, we should apply the inverse link function. We can use the `brms::inv_logit_scaled()` or the `plogis()` function.

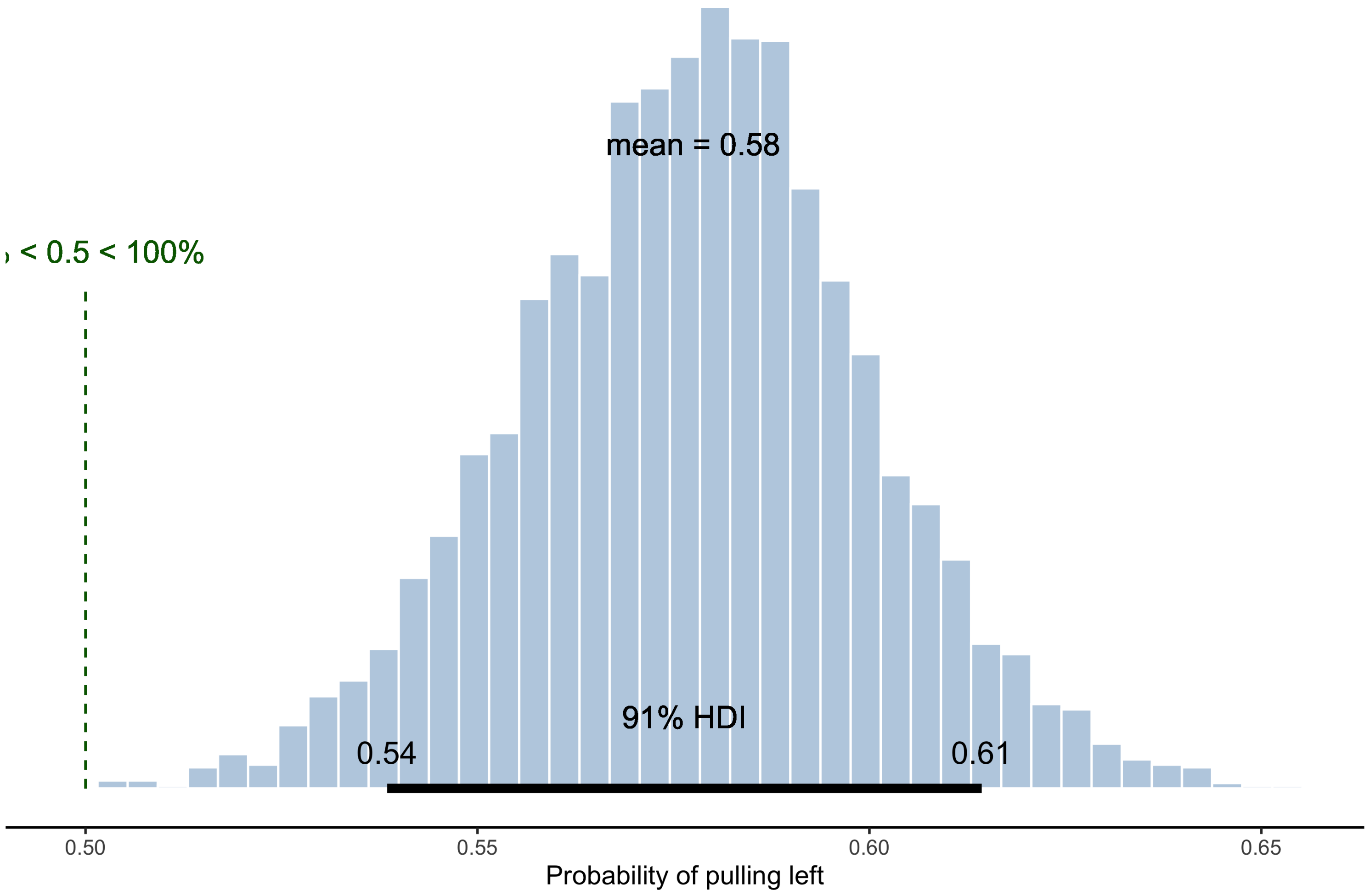
```r
fixed_effects <- fixef(mod1.2) # fixed effects (i.e., the intercept)
plogis(fixed_effects) # inverse link function
```

```
           Estimate Est.Error      Q2.5     Q97.5
Intercept 0.5776786 0.5228274 0.5327255 0.622261
```

On average (without considering the predictors), it seems that chimpanzees are slightly more likely to pull the left lever than the right one…

Logistic regression

```r
post <- as_draws_df(x = mod1.2) # retrieving the posterior samples
intercept_samples <- plogis(post$b_Intercept) # posterior samples for the intercept

posterior_plot(samples = intercept_samples, compval = 0.5) + labs(x = "Probability of pulling left")
```



Logistic regression

How can we pick a value for ω (in the priors on the slopes)?

$$
\begin{aligned}
L_i &\sim \mathrm{Binomial}(1, p_i) \\
\mathrm{logit}(p_i) &= \alpha + \beta_P P_i + \beta_C C_i + \beta_{PC} P_i C_i \\
\alpha &\sim \mathrm{Normal}(0, 1) \\
\beta_P, \beta_C, \beta_{PC} &\sim \mathrm{Normal}(0, \omega)
\end{aligned}
$$

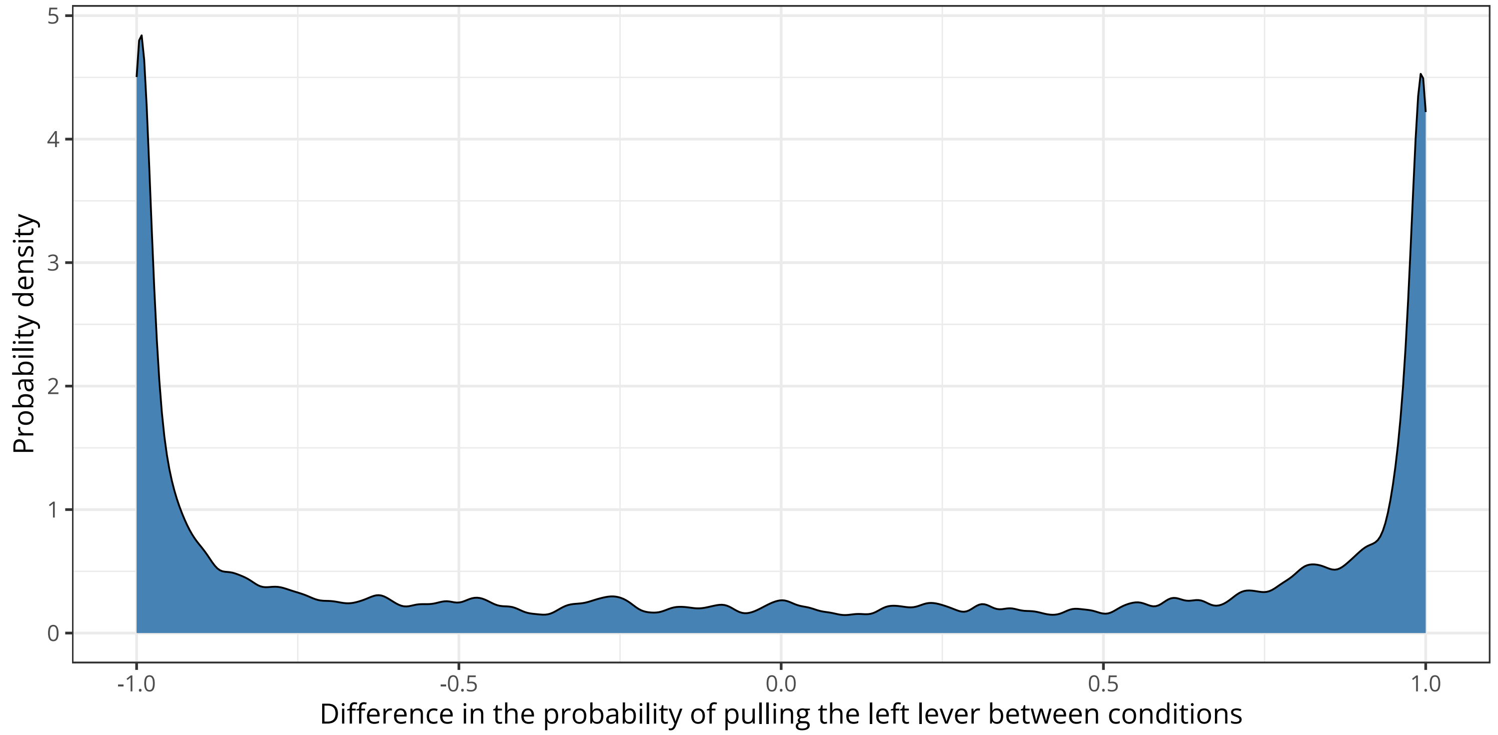
- L_i indicates whether the monkey pulled the left lever (`pulled_left`).
- P_i indicates whether the left side corresponded to the prosocial side.
- C_i indicates the presence of a partner.

Logistic regression

```r
# recoding predictors
df1 <- df1 %>%
  mutate(
    prosoc_left = ifelse(prosoc_left == 1, 0.5, -0.5),
    condition = ifelse(condition == 1, 0.5, -0.5)
    )

priors <- c(
  prior(normal(0, 1), class = Intercept),
  prior(normal(0, 10), class = b)
  )

mod2.1 <- brm(
  formula = pulled_left | trials(1) ~ 1 + prosoc_left * condition,
  family = binomial,
  prior = priors,
  data = df1,
  sample_prior = "yes"
  )
```

Prior predictive check

```r
prior_draws(x = mod2.1) %>% # prior samples
  mutate(
    condition1 = plogis(Intercept - 0.5 * b), # p in condition 1
    condition2 = plogis(Intercept + 0.5 * b) # p in condition 0
    ) %>%
  ggplot(aes(x = condition2 - condition1) ) + # plotting the difference
  geom_density(fill = "steelblue", adjust = 0.1) +
  labs(
    x = "Difference in the probability of pulling the left lever between conditions",
    y = "Probability density"
    )
```

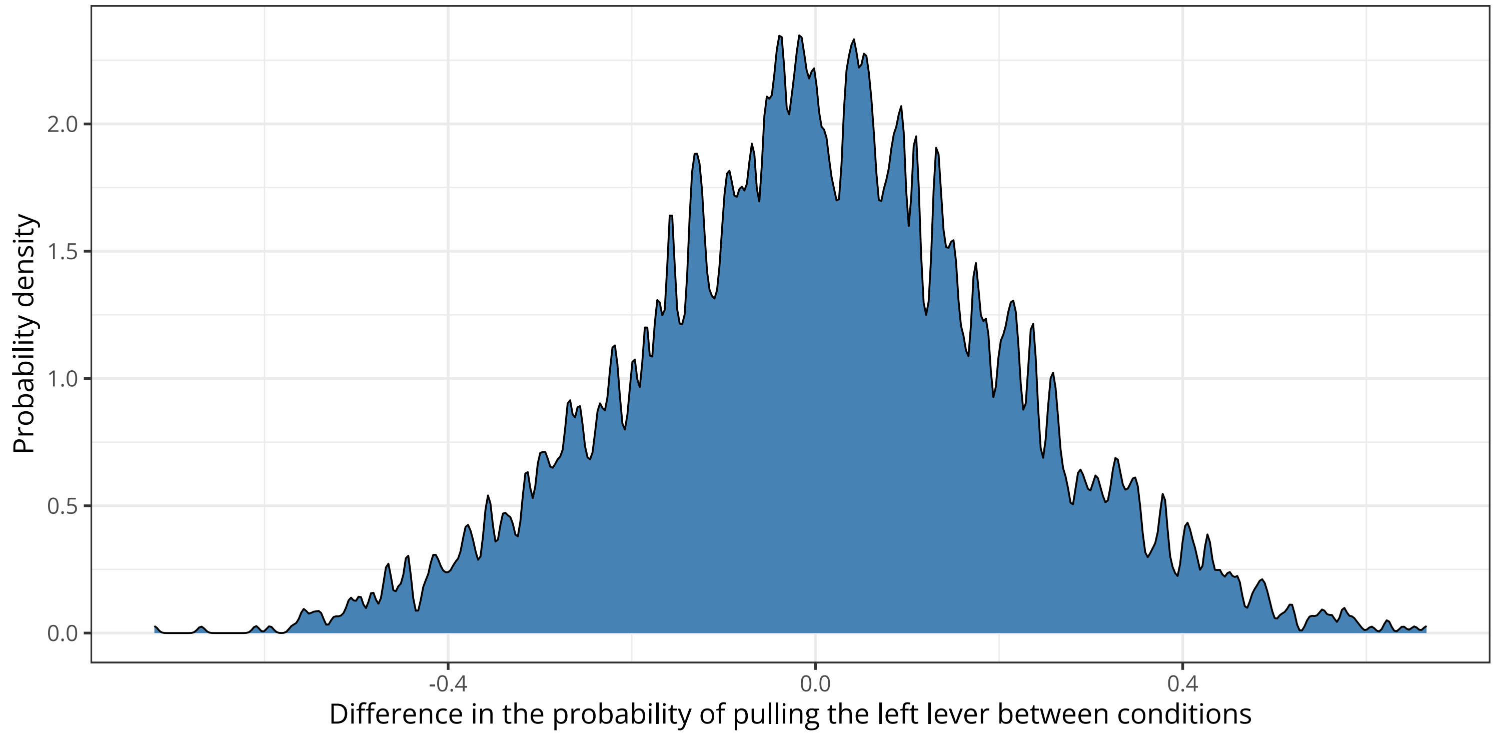


Logistic regression

```r
priors <- c(
  prior(normal(0, 1), class = Intercept),
  prior(normal(0, 1), class = b)
  )

mod2.2 <- brm(
  formula = pulled_left | trials(1) ~ 1 + prosoc_left * condition,
  family = binomial,
  prior = priors,
  data = df1,
  sample_prior = "yes"
  )
```

Prior predictive check

```r
prior_draws(mod2.2) %>% # prior samples
  mutate(
    condition1 = plogis(Intercept - 0.5 * b), # p in condition 1
    condition2 = plogis(Intercept + 0.5 * b) # p in condition 0
    ) %>%
  ggplot(aes(x = condition2 - condition1) ) +
  geom_density(fill = "steelblue", adjust = 0.1) +
  labs(
    x = "Difference in the probability of pulling the left lever between conditions",
    y = "Probability density"
    )
```



Logistic regression

```r
summary(mod2.2)
```

```
 Family: binomial 
  Links: mu = logit 
Formula: pulled_left | trials(1) ~ 1 + prosoc_left * condition 
   Data: df1 (Number of observations: 504) 
  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
Intercept                 0.33      0.09     0.15     0.50 1.00     4652
prosoc_left               0.55      0.18     0.20     0.91 1.00     4661
condition                -0.19      0.18    -0.56     0.16 1.00     5310
prosoc_left:condition     0.17      0.35    -0.52     0.84 1.00     4909
                      Tail_ESS
Intercept                 3142
prosoc_left               2898
condition                 2737
prosoc_left:condition     3222

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
```

Relative effects vs. absolute effects

The linear model does not directly predict the probability but the log-odds of the probability:

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1 - p_i}\right) = \alpha + \beta \times x_i$$

Two types of effect can be distinguished and interpreted.

Relative effect: The relative effect relates to the logarithm of the probability ratio. It indicates the proportion of change induced by the predictor on the **chances** of success (or rather, on the odds). It tells us nothing about the probability of the event, in absolute terms.

Absolute effect: Effect which directly affects the probability of an event. It depends on all the parameters of the model and gives us the effective impact of a change of one unit of a predictor (in probability space).

Relative effect

This is a **proportion** of change induced by the predictor on the odds ratio. Illustration with a model without interaction.

$$
\begin{aligned}
\log\left(\frac{p_i}{1 - p_i}\right) &= \alpha + \beta x_i \\
\frac{p_i}{1 - p_i} &= \exp(\alpha + \beta x_i)
\end{aligned}
$$

The proportional odds q of an event is the number by which the odds are multiplied when x_i increases by one.

$$
q = \frac{\exp(\alpha + \beta(x_i + 1))}{\exp(\alpha + \beta x_i)} = \frac{\exp(\alpha)\exp(\beta x_i)\exp(\beta)}{\exp(\alpha)\exp(\beta x_i)} = \exp(\beta)
$$

When $q = 2$ (for example), a one-unit increase in x_i doubles the odds.

Interpreting relative effects

The relative effect of a parameter **also depends on the other parameters**. In the previous model, the predictor `prosoc_left` increases the log odds by about 0.54, which translates into an increase in odds of $\exp(0.54) \approx 1.72$, that is, an increase in odds of about 72%.

Let's assume that the intercept $\alpha = 4$.

- The probability of pulling the lever without further consideration is $\text{logit}^{-1}(4) \approx 0.98$.
- Considering the effect of `prosoc_left`, we obtain $\text{logit}^{-1}(4 + 0.54) \approx 0.99$.

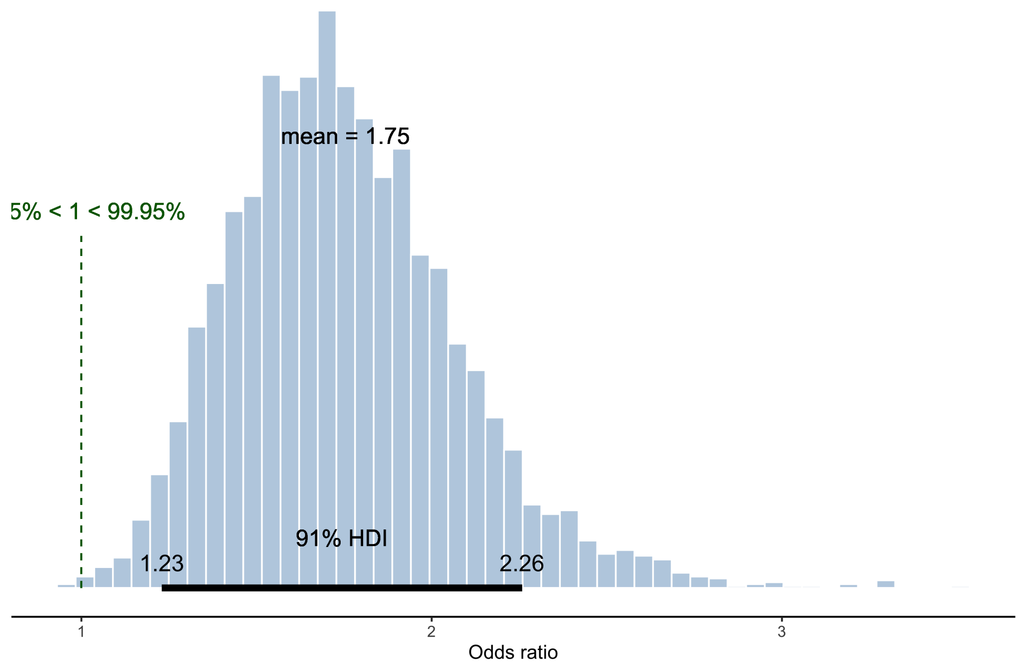
An increase of 72% in the log-odds translates into an increase of just 1% in the effective probability... Relative effects can lead to misinterpretations when the scale of the variable being measured is not taken into account.

Interpreting relative effects

```
fixef(mod2.2) # retrieving estimates for "fixed effects"
```

```
                       Estimate  Est.Error       Q2.5     Q97.5
Intercept             0.3272953 0.09039531  0.1494670 0.5041133
prosoc_left           0.5457259 0.17928616  0.2024788 0.9110874
condition            -0.1915692 0.18382309 -0.5581443 0.1578334
prosoc_left:condition  0.1662294 0.34577459 -0.5209200 0.8445441
```

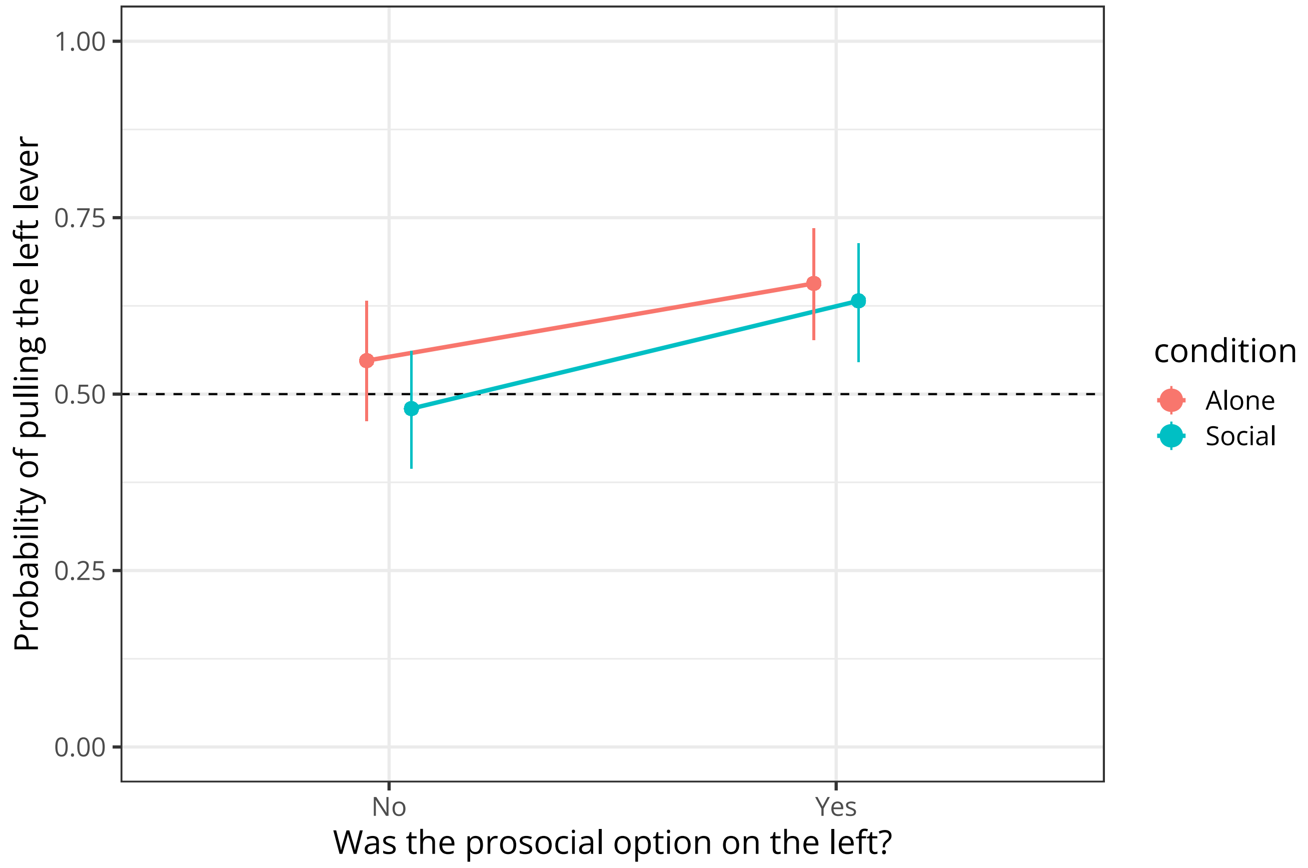
```
post <- as_draws_df(x = mod2.2) # posterior samples
posterior_plot(samples = exp(post$b_prosoc_left), compval = 1) + labs(x = "Odds ratio")
```



Absolute effects

The absolute effect depends on all the parameters of the model and gives us the effective impact of a change of one unit in a predictor (in probability space).

```r
model_predictions <- fitted(mod2.2) %>% # prediction for p (i.e., the probability)
  data.frame() %>%
  bind_cols(df1) %>%
  mutate(condition = factor(condition), prosoc_left = factor(prosoc_left) )
```



Aggregated binomial regression

These data represent the number of applications to UC Berkeley by gender and department. Each application was either accepted or rejected and the results are aggregated by department and gender.

```r
(df2 <- open_data(admission) )
```

```
   dept gender admit reject applications
1     A   Male   512    313          825
2     A Female    89     19          108
3     B   Male   353    207          560
4     B Female    17      8           25
5     C   Male   120    205          325
6     C Female   202    391          593
7     D   Male   138    279          417
8     D Female   131    244          375
9     E   Male    53    138          191
10    E Female    94    299          393
11    F   Male    22    351          373
12    F Female    24    317          341
```

We want to know whether there is a gender bias in recruitment?

Aggregated binomial regression

We will build a model of the admissions decision using the gender of the applicant as a predictor.

$$
\begin{aligned}
\text{admit}_i &\sim \text{Binomial}(n_i, p_i) \\
\text{logit}(p_i) &= \alpha + \beta_m \times m_i \\
\alpha &\sim \text{Normal}(0, 1) \\
\beta_m &\sim \text{Normal}(0, 1)
\end{aligned}
$$

Variables:

- admit_i: The number of successful applications (`admit`).
- n_i: The total number of applications (`applications`).
- m_i: The aplicant's gender (`1 = Male`).

Aggregated binomial regression

```r
priors <- c(prior(normal(0, 1), class = Intercept) )

mod3 <- brm(
  formula = admit | trials(applications) ~ 1,
  family = binomial(link = "logit"),
  prior = priors,
  data = df2,
  sample_prior = "yes"
  )
```

Aggregated binomial regression

```r
priors <- c(
  prior(normal(0, 1), class = Intercept),
  prior(normal(0, 1), class = b)
  )

# dummy-coding
df2$male <- ifelse(df2$gender == "Male", 1, 0)

mod4 <- brm(
  formula = admit | trials(applications) ~ 1 + male,
  family = binomial(link = "logit"),
  prior = priors,
  data = df2,
  sample_prior = "yes"
  )
```

Aggregated binomial regression

```r
summary(mod4)
```

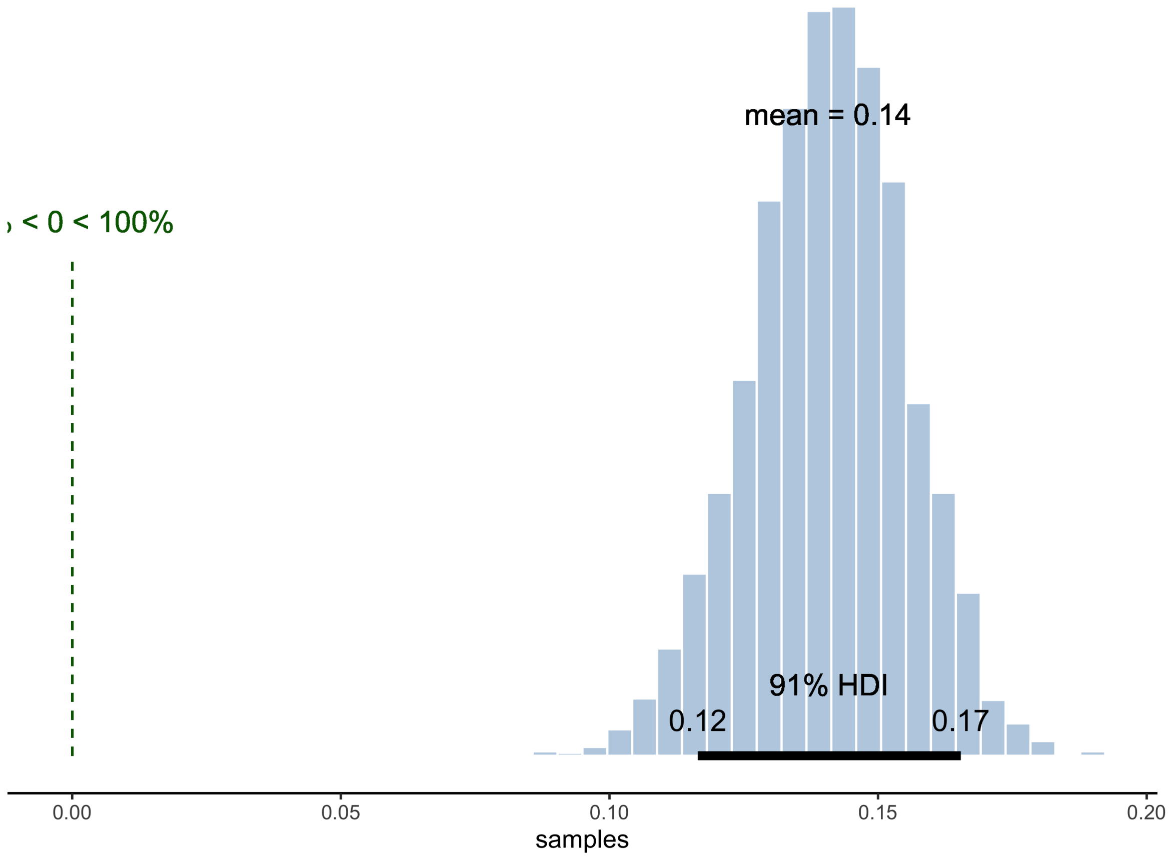
```
 Family: binomial
  Links: mu = logit
Formula: admit | trials(applications) ~ 1 + male
   Data: df2 (Number of observations: 12)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept    -0.83      0.05    -0.93    -0.72 1.00     2225     1853
male          0.61      0.06     0.48     0.73 1.00     2628     2185

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).
```

Being a man seems to be an advantage…! The odds ratio is $\exp(0.61) \approx 1.84$.

Aggregated binomial regression

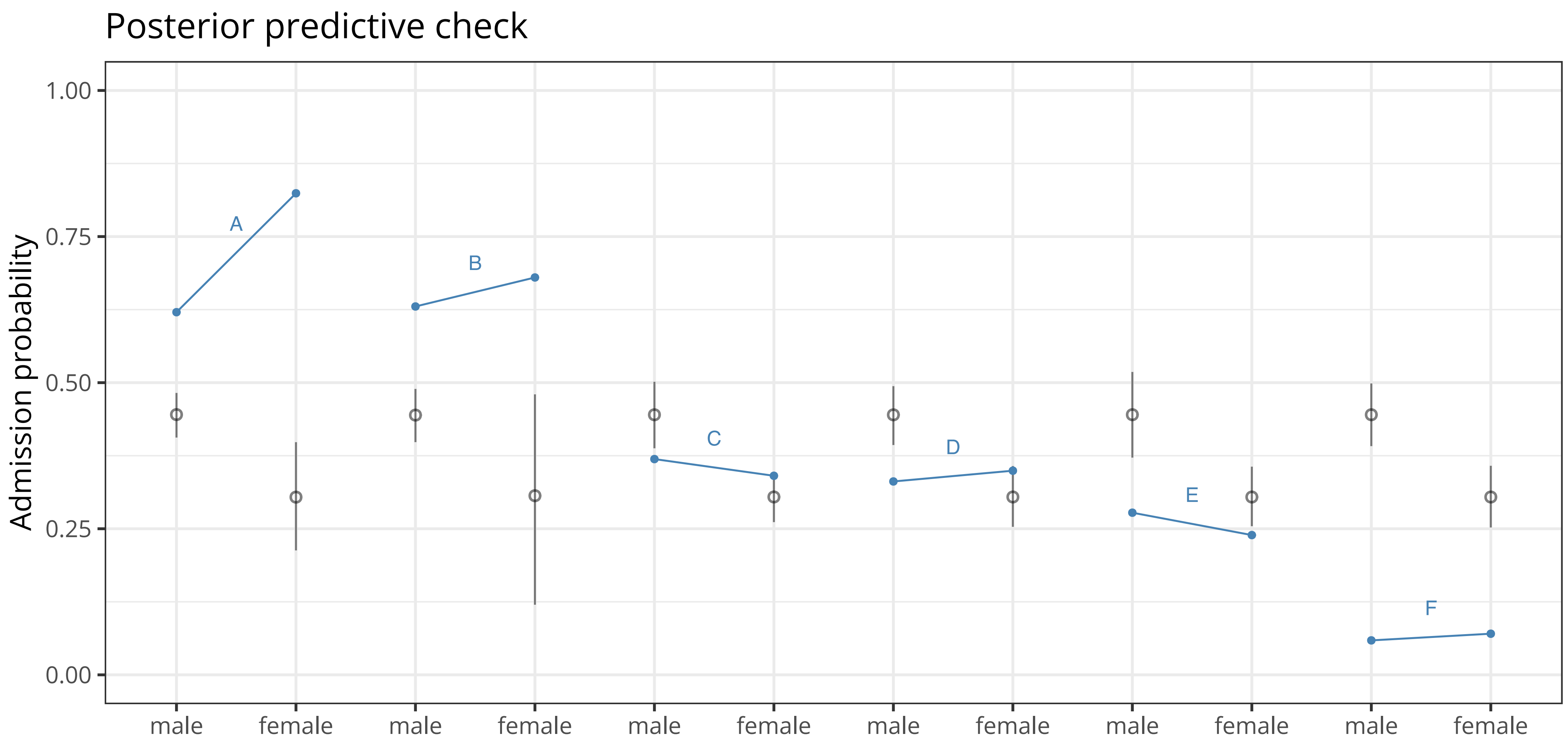
Let’s calculate the difference in probability of admission between men and women.

```r
post <- as_draws_df(x = mod4)
p.admit.male <- plogis(post$b_Intercept + post$b_male)
p.admit.female <- plogis(post$b_Intercept)
diff.admit <- p.admit.male - p.admit.female
posterior_plot(samples = diff.admit, compval = 0)
```



Visualising the model's predictions

Let's examine the model's predictions by department.



Aggregated binomial regression

The model's predictions are very poor… There are only two departments for which women have a lower probability of admission than men (C and E), whereas the model predicts a lower probability of admission for all departments…

The problem is twofold:

- Men and women do not apply to the same departments.
- The departments do not all have the same number of students.

This is Simpson's "paradox"… remarks:

- The posterior distribution alone would not have detected this problem.
- We were able to pinpoint the problem by examining the detailed model's predictions…

Aggregated binomial regression

We therefore build a model of admission decisions by gender, within each department.

$$
\begin{aligned}
\text{admit}_i &\sim \text{Binomial}(n_i, p_i) \\
\text{logit}(p_i) &= \alpha_{\text{dept}[i]} + \beta_m \times m_i \\
\alpha_{\text{dept}[i]} &\sim \text{Normal}(0, 1) \\
\beta_m &\sim \text{Normal}(0, 1)
\end{aligned}
$$

Aggregated binomial regression

```r
# model without any predictor
mod5 <- brm(
  admit | trials(applications) ~ 0 + dept,
  family = binomial(link = "logit"),
  prior = prior(normal(0, 1), class = b),
  data = df2
  )

# model with one predictor (sex)
mod6 <- brm(
  admit | trials(applications) ~ 0 + dept + male,
  family = binomial(link = "logit"),
  prior = prior(normal(0, 1), class = b),
  data = df2
  )
```

Aggregated binomial regression

```r
summary(mod6)
```

```
 Family: binomial 
  Links: mu = logit 
Formula: admit | trials(applications) ~ 0 + dept + male 
   Data: df2 (Number of observations: 12) 
  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
deptA     0.68      0.10     0.50     0.87 1.00     1965     2929
deptB     0.64      0.11     0.42     0.87 1.00     2191     2846
deptC    -0.58      0.08    -0.73    -0.43 1.00     3009     2535
deptD    -0.61      0.09    -0.78    -0.44 1.00     2834     2758
deptE    -1.05      0.10    -1.24    -0.86 1.00     3101     2982
deptF    -2.57      0.16    -2.89    -2.26 1.00     3722     2448
male     -0.11      0.08    -0.27     0.06 1.00     1619     2403

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).
```

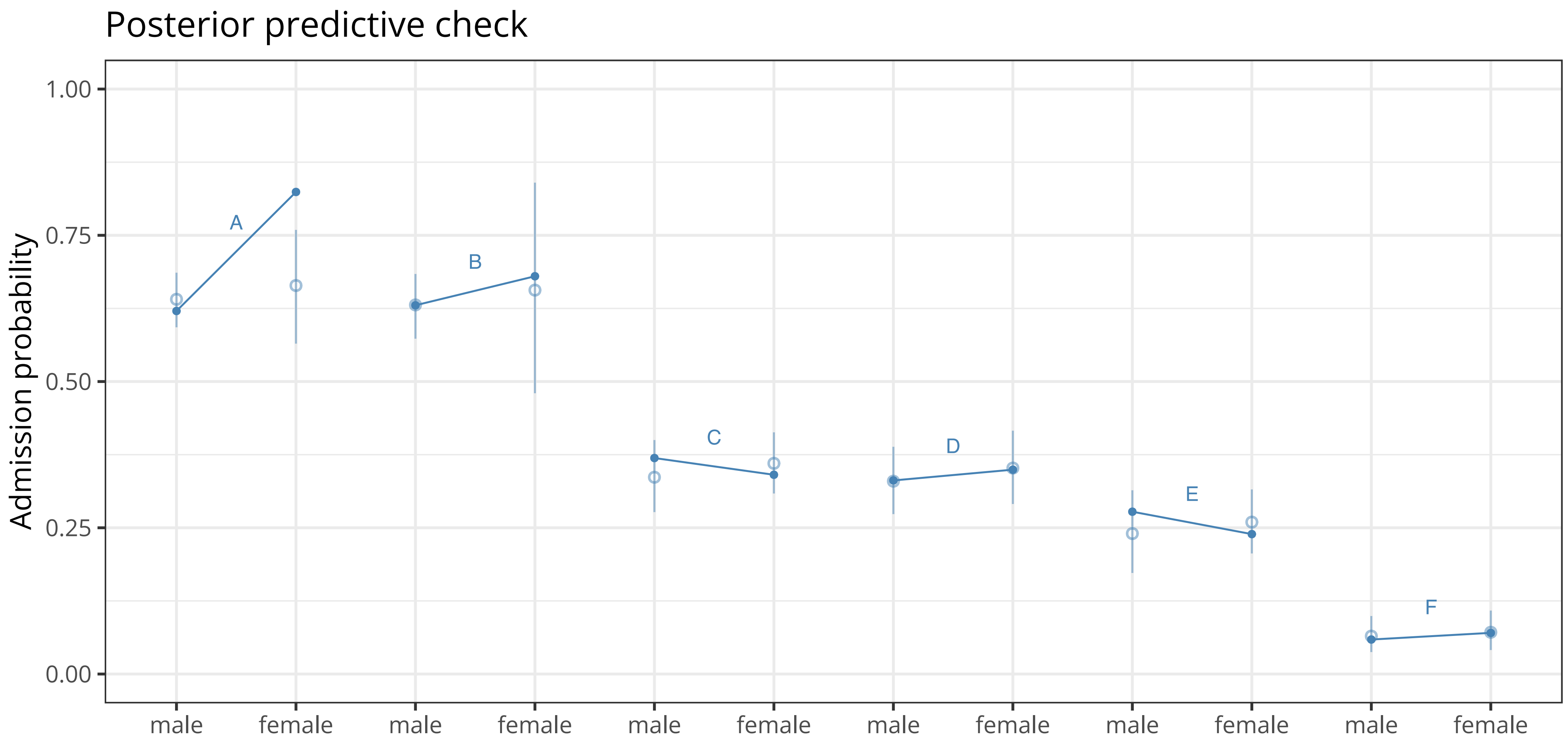
Aggregated binomial regression

```r
fixef(mod6)
```

```
        Estimate  Est.Error       Q2.5       Q97.5
deptA  0.6844546 0.09699335  0.4955171  0.87378332
deptB  0.6416951 0.11488191  0.4201809  0.86578455
deptC -0.5769302 0.07513928 -0.7272142 -0.42587992
deptD -0.6073597 0.08545915 -0.7823102 -0.44428476
deptE -1.0488372 0.09630802 -1.2376921 -0.86091445
deptF -2.5741081 0.16103756 -2.8928532 -2.26372318
male  -0.1053356 0.08111418 -0.2700577  0.05695022
```

Now, the prediction for β_m goes the other way... The odds ratio is $\exp(-0.1) = 0.9$, the odds of admission for men are estimated to be 90% of the odds for women.

Aggregated binomial regression



Conclusions

Men and women do not apply to the same departments and the departments vary in their probability of admission. In this case, women applied more to departments E and F (with a lower probability of admission) and applied less to departments A or B, with a higher probability of admission.

To assess the effect of gender on the probability of admission, we therefore need to ask the following question: “What is the difference in probability of admission between men and women **within each department**?” (rather than in general).

Remember that the regression model can be generalised to different data generation models (i.e., different probability distributions, such as Normal, Binomial, Poisson, etc) and that the parameter space can be “connected” to the predictor space (measured variables) using link functions (e.g., logarithm, exponential, logit, etc).

Remember the distinction between **relative effects** (e.g., a change in odds) and **absolute effects** (e.g., a difference in probability).

Practical work - Experimental absenteeism

Working with human subjects implies a minimum of mutual cooperation. But this is not always the case. A non-negligible proportion of students who register for Psychology experiments do not turn up on the day they are supposed to... We wanted to estimate the **probability of a registered student's attendance** as a function of whether or not a reminder email was sent (this example is presented in detail in two blog articles, accessible here and here).

```r
df3 <- open_data(absence)
df3 %>% sample_frac %>% head(10)
```

```
         day inscription reminder absence presence total
1     Monday      doodle       no       5        4     9
2    Tuesday       panel      yes       0        9     9
3     Monday      doodle      yes       2        6     8
4     Friday       panel      yes       0       10    10
5    Tuesday      doodle      yes       1        7     8
6  Wednesday      doodle      yes       0        4     4
7    Tuesday      doodle       no       4       10    14
8     Friday      doodle      yes       0        2     2
9   Thursday      doodle       no       3       11    14
10    Friday      doodle       no       7       11    18
```

Practical work

- **What is the probability that a participant who has registered on his or her own initiative will actually come and take part in the experiment?**
- What is the effect of the reminder?
- What is the effect of the registration method?
- What is the joint effect of these two predictors?

Practical work

Write the model that predicts the presence of a participant without a predictor.

$$
\begin{aligned}
y_i &\sim \mathrm{Binomial}(n_i, p_i) \\
\mathrm{logit}(p_i) &= \alpha \\
\alpha &\sim \mathrm{Normal}(0, 1)
\end{aligned}
$$

Practical work

```r
mod7 <- brm(
    presence | trials(total) ~ 1,
    family = binomial(link = "logit"),
    prior = prior(normal(0, 1), class = Intercept),
    data = df3,
    # using all available parallel cores
    cores = parallel::detectCores()
    )
```

```r
fixef(mod7) # relative effect (log-odds)
```

```
          Estimate Est.Error      Q2.5   Q97.5
Intercept 1.146064 0.1930107 0.7875779 1.53671
```

```r
fixef(mod7) %>% plogis # absolute effect (probability of presence)
```

```
           Estimate Est.Error     Q2.5     Q97.5
Intercept 0.7587913 0.5481034 0.687311 0.8229859
```

Practical work

- What is the probability that a participant who has registered on his or her own initiative will actually come and take part in the experiment?
- **What is the effect of the reminder?**
- What is the effect of the registration method?
- What is the joint effect of these two predictors?

Practical work

We start by recoding into dummy variables `reminder` and `inscription`.

```r
df3 <-
  df3 %>%
  mutate(
    reminder = ifelse(reminder == "no", 0, 1),
    inscription = ifelse(inscription == "panel", 0, 1)
    )

head(df3, n = 10)
```

```
        day inscription reminder absence presence total
1    Friday           1        0       7       11    18
2    Friday           1        1       0        2     2
3    Friday           0        1       0       10    10
4    Monday           1        0       5        4     9
5    Monday           1        1       2        6     8
6    Monday           0        1       6       12    18
7  Thursday           1        0       3       11    14
8   Tuesday           1        0       4       10    14
9   Tuesday           1        1       1        7     8
10  Tuesday           0        1       0        9     9
```

Practical work

Write the model that predicts presence as a function of recall.

$$
\begin{aligned}
y_i &\sim \mathrm{Binomial}(n_i, p_i) \\
\mathrm{logit}(p_i) &= \alpha + \beta \times \mathrm{reminder}_i \\
\alpha &\sim \mathrm{Normal}(0, 1) \\
\beta &\sim \mathrm{Normal}(0, 1)
\end{aligned}
$$

Practical work

Write the model that predicts the probability of presence as a function of reminder.

```r
priors <- c(
  prior(normal(0, 1), class = Intercept),
  prior(normal(0, 1), class = b)
  )

mod8 <- brm(
    presence | trials(total) ~ 1 + reminder,
    family = binomial(link = "logit"),
    prior = priors,
    data = df3,
    cores = parallel::detectCores()
    )
```

Practical work

What is the **relative** effect of the reminder email?

```r
exp(fixef(mod8)[2]) # odds ratio with and without the reminder e-mail
```

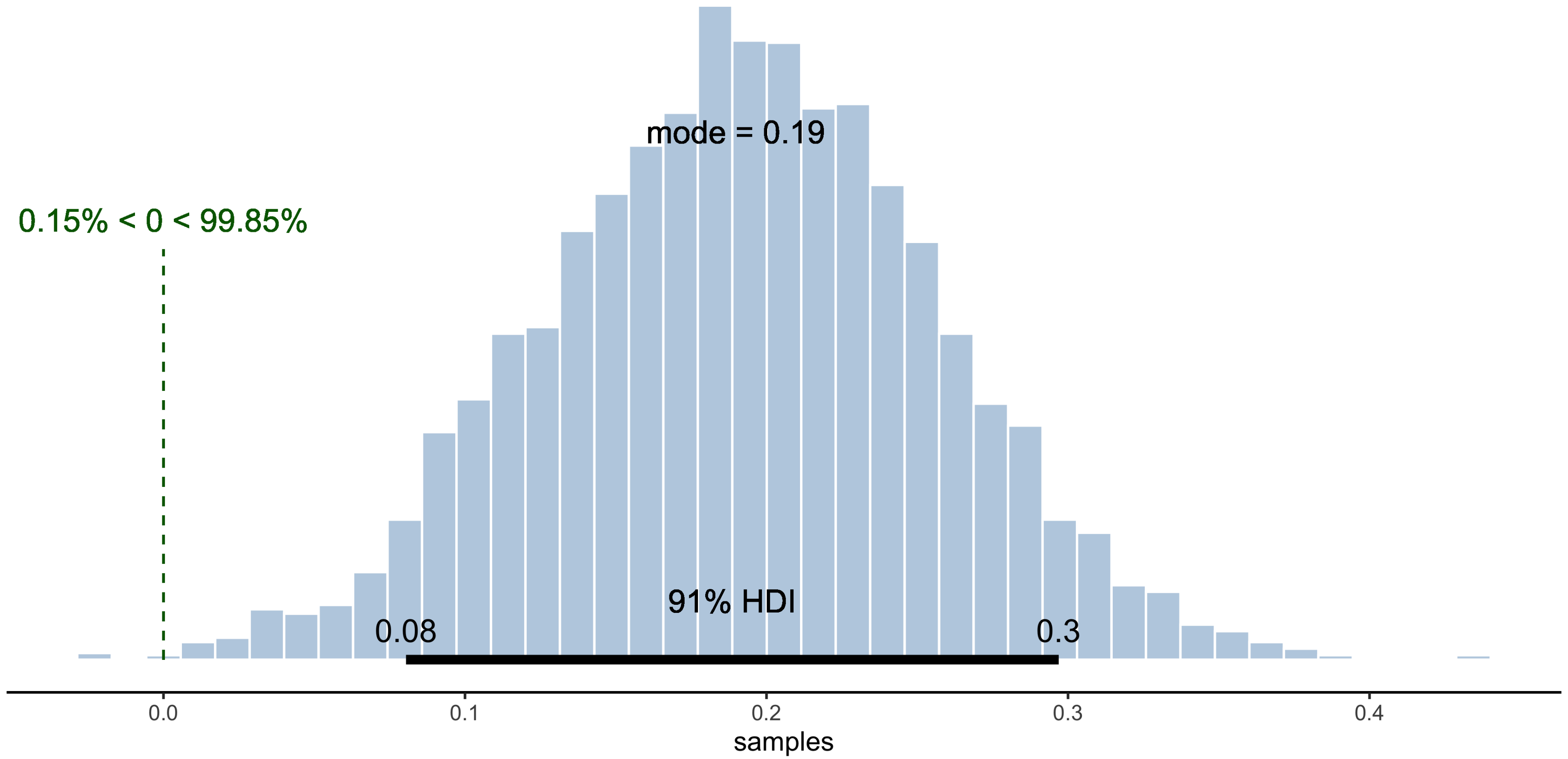
```
[1] 3.021774
```

Sending a reminder e-mail increases the odds by about 3.

Practical work

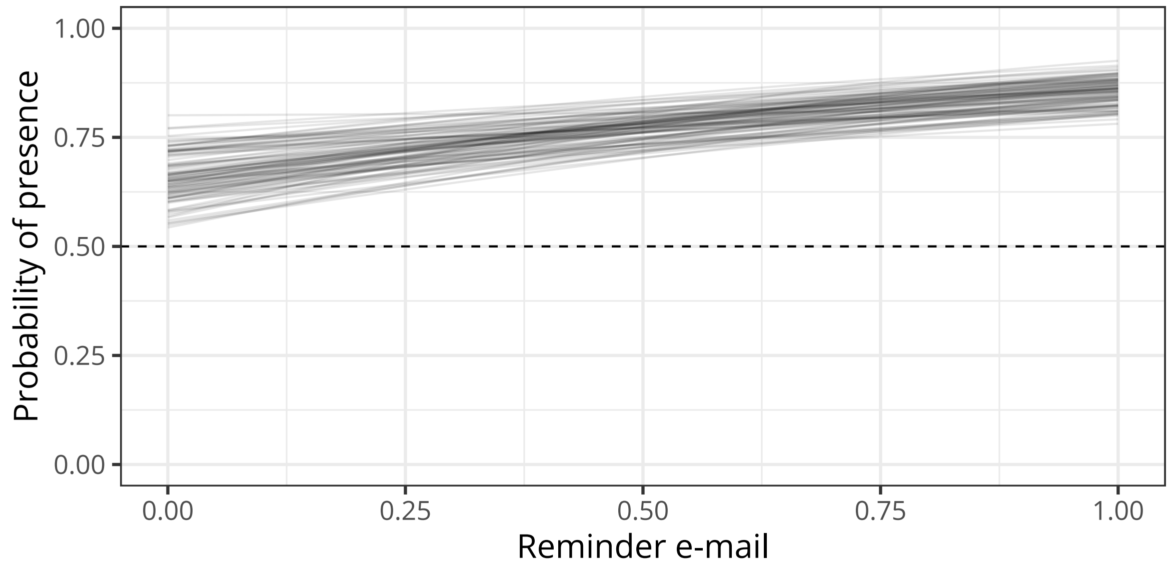
What is the **absolute** effect of the reminder email?

```r
post <- as_draws_df(x = mod8) # retrieving posterior samples
p.no <- plogis(post$b_Intercept) # probability of presence without reminder e-mail
p.yes <- plogis(post$b_Intercept + post$b_reminder) # probability of presence with reminder e-mail
posterior_plot(samples = p.yes - p.no, compval = 0, usemode = TRUE)
```



Practical work

```r
library(tidybayes)
library(modelr)

df3 %>%
  group_by(total) %>%
  data_grid(reminder = seq_range(reminder, n = 1e2) ) %>%
  add_fitted_draws(mod8, newdata = ., n = 100, scale = "linear") %>%
  mutate(estimate = plogis(.value) ) %>%
  group_by(reminder, .draw) %>%
  summarise(estimate = mean(estimate) ) %>%
  ggplot(aes(x = reminder, y = estimate, group = .draw) ) +
  geom_hline(yintercept = 0.5, lty = 2) +
  geom_line(aes(y = estimate, group = .draw), size = 0.5, alpha = 0.1) +
  ylim(0, 1) +
  labs(x = "Reminder e-mail", y = "Probability of presence")
```



Practical work

- What is the probability that a participant who has registered on his or her own initiative will actually come and take part in the experiment?
- What is the effect of the reminder?
- **What is the effect of the registration method?**
- What is the joint effect of these two predictors?

Practical work

Write the model that predicts the probability of presence as a function of registration mode.

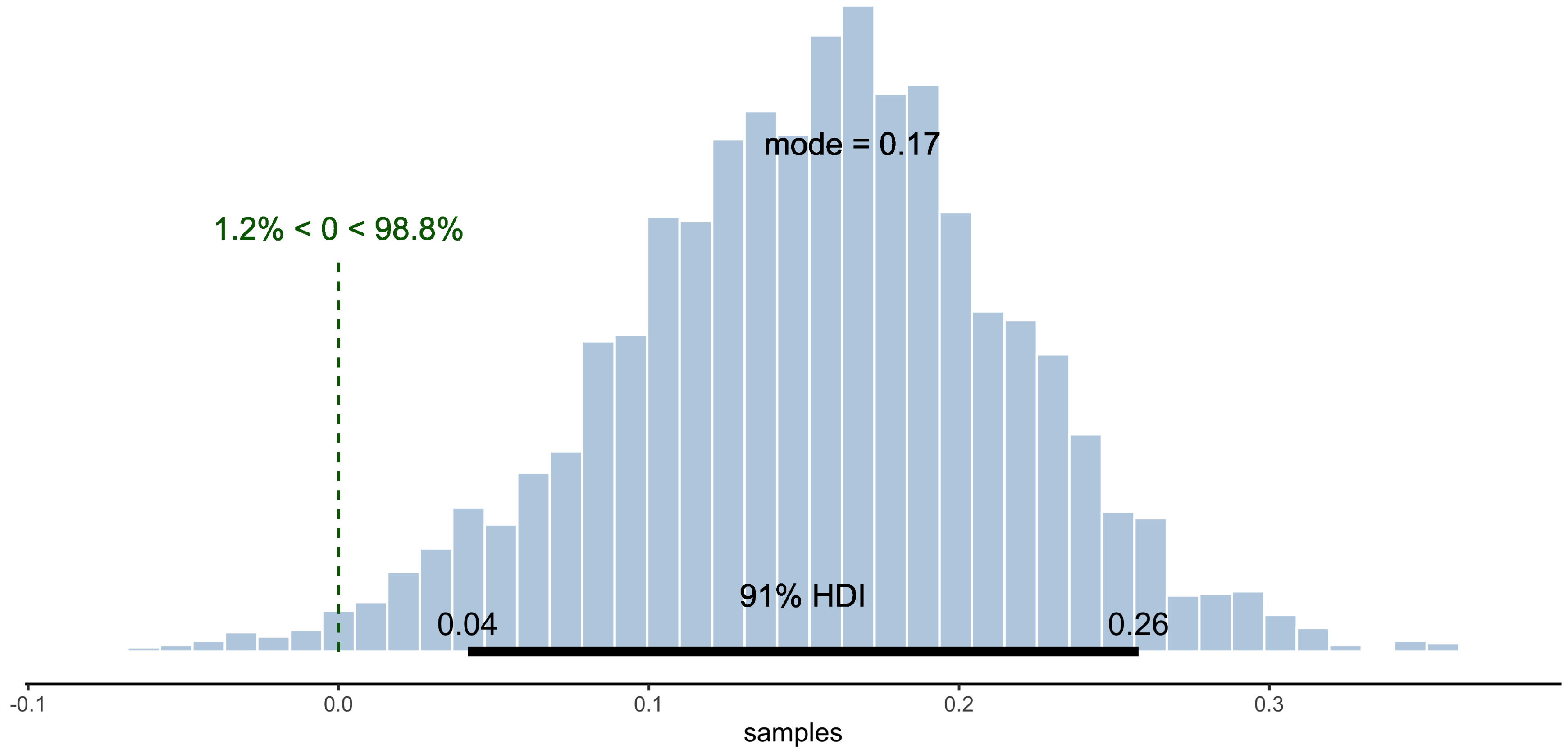
$$
\begin{aligned}
y_i &\sim \mathrm{Binomial}(n_i, p_i) \\
\mathrm{logit}(p_i) &= \alpha + \beta \times \mathrm{inscription}_i \\
\alpha &\sim \mathrm{Normal}(0, 1) \\
\beta &\sim \mathrm{Normal}(0, 1)
\end{aligned}
$$

Practical work

```r
priors <- c(
  prior(normal(0, 1), class = Intercept),
  prior(normal(0, 1), class = b)
  )

mod9 <- brm(
    presence | trials(total) ~ 1 + inscription,
    family = binomial(link = "logit"),
    prior = priors,
    data = df3,
    cores = parallel::detectCores()
    )
```

Practical work

```r
post <- as_draws_df(x = mod9)
p.panel <- plogis(post$b_Intercept) # average probability of presence - panel
p.doodle <- plogis(post$b_Intercept + post$b_inscription) # average probability of presence- doodle
posterior_plot(samples = p.panel - p.doodle, compval = 0, usemode = TRUE)
```



The probability of presence is increased by around 0.17 when registering on a panel compared with registering on a Doodle (slightly smaller effect than for the reminder).

Practical work

- What is the probability that a participant who has registered on his or her own initiative will actually come and take part in the experiment?
- What is the effect of the reminder?
- What is the effect of the registration method?
- **What is the joint effect of these two predictors?**

Practical work

Write the full model.

$$
\begin{aligned}
y_i &\sim \mathrm{Binomial}(n_i, p_i) \\
\mathrm{logit}(p_i) &= \alpha + \beta_1 \times \mathrm{reminder}_i + \beta_2 \times \mathrm{inscription}_i \\
\alpha &\sim \mathrm{Normal}(0, 1) \\
\beta_1, \beta_2 &\sim \mathrm{Normal}(0, 1)
\end{aligned}
$$

Practical work

```r
priors <- c(
  prior(normal(0, 1), class = Intercept),
  prior(normal(0, 1), class = b)
  )

mod10 <- brm(
    presence | trials(total) ~ 1 + reminder + inscription,
    family = binomial(link = "logit"),
    prior = priors,
    data = df3,
    cores = parallel::detectCores()
    )
```

Practical work

```r
summary(mod10)
```

```
 Family: binomial 
  Links: mu = logit 
Formula: presence | trials(total) ~ 1 + reminder + inscription 
   Data: df3 (Number of observations: 13) 
  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       1.02      0.57    -0.11     2.11 1.00     2523     2363
reminder        0.92      0.48     0.02     1.92 1.00     2436     2415
inscription    -0.35      0.54    -1.39     0.75 1.00     2577     2306

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).
```

Practical work

The reminder e-mail seems to have less effect in the full model than in the simple model... why is this?

```r
fixef(mod8) %>% exp() # computing the odds ratio
```

```
          Estimate Est.Error     Q2.5    Q97.5
Intercept 1.958003  1.274121 1.226752 3.179808
reminder  3.021774  1.468636 1.435682 6.527387
```

```r
fixef(mod9) %>% exp() # computing the odds ratio
```

```
             Estimate Est.Error      Q2.5      Q97.5
Intercept   6.3070616  1.479543 3.1012980 14.3532331
inscription 0.3807339  1.547852 0.1583898  0.8753162
```

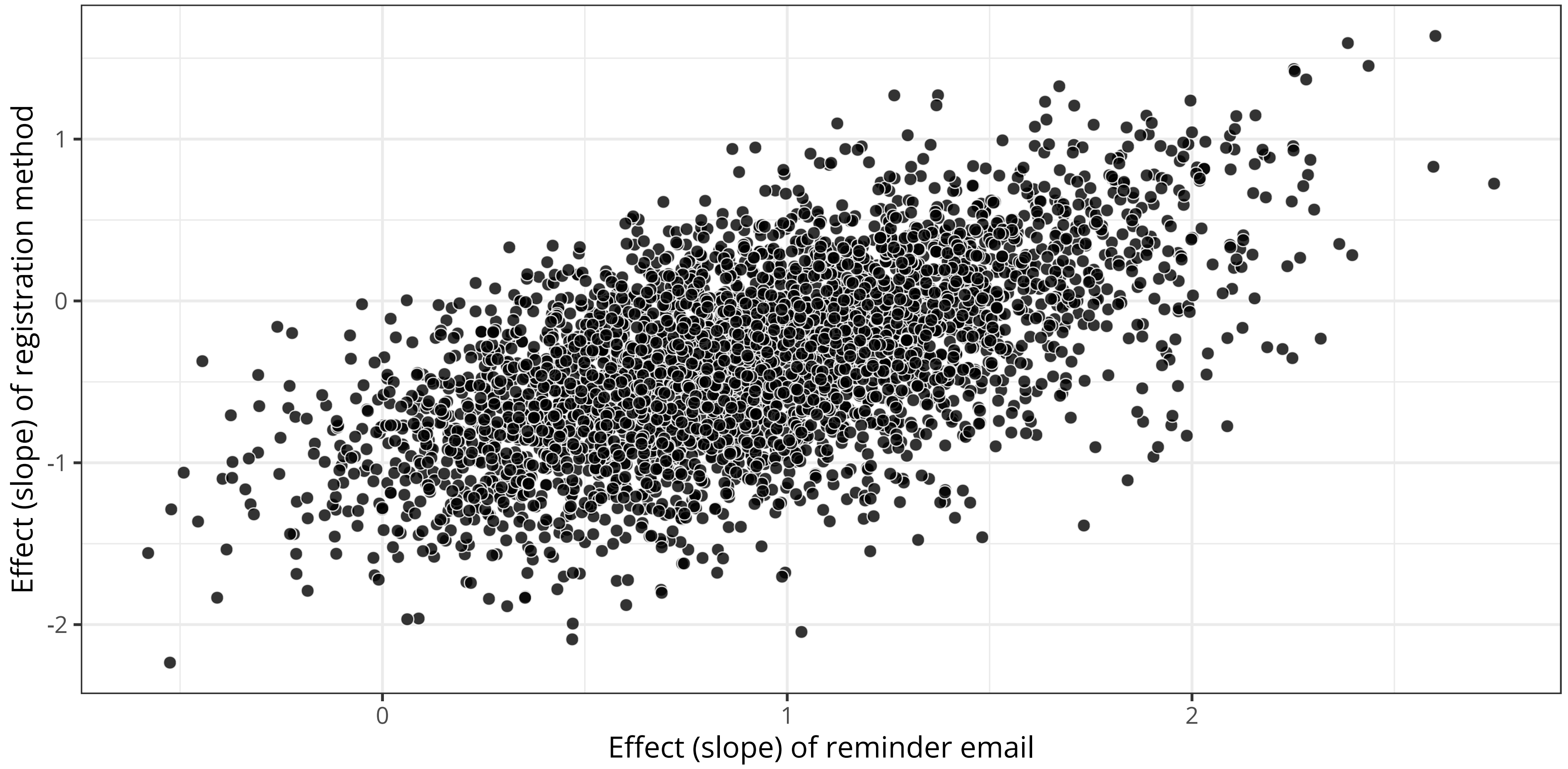
```r
fixef(mod10) %>% exp() # computing the odds ratio
```

```
             Estimate Est.Error      Q2.5    Q97.5
Intercept   2.7724678  1.771534 0.8995797 8.209809
reminder    2.5058259  1.616034 1.0199602 6.851798
inscription 0.7016501  1.712775 0.2500480 2.115074
```

Practical work

When two predictors share some of the same information, the slope estimates are correlated...

```r
as_draws_df(x = mod10) %>%
    ggplot(aes(b_reminder, b_inscription) ) +
    geom_point(size = 3, pch = 21, alpha = 0.8, color = "white", fill = "black") +
    labs(x = "Effect (slope) of reminder email", y = "Effect (slope) of registration method")
```



Practical work

Indeed, the data were collected by two experimenters. One of them recruited all her participants via Doodle, and did not often send a reminder email. The second experimenter recruited all her participants via a physical sign in the laboratory and systematically sent a reminder email. In other words, these two variables are almost perfectly identical.

```r
open_data(absence) %>%
  group_by(inscription, reminder) %>%
  summarise(n = sum(total) ) %>%
  spread(key = reminder, value = n) %>%
  data.frame()
```

```
  inscription no yes
1      doodle 72  22
2       panel NA  51
```

References

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114