A Gentle Introduction to Bayesian Estimation

Day 3: Algorithms and Checks

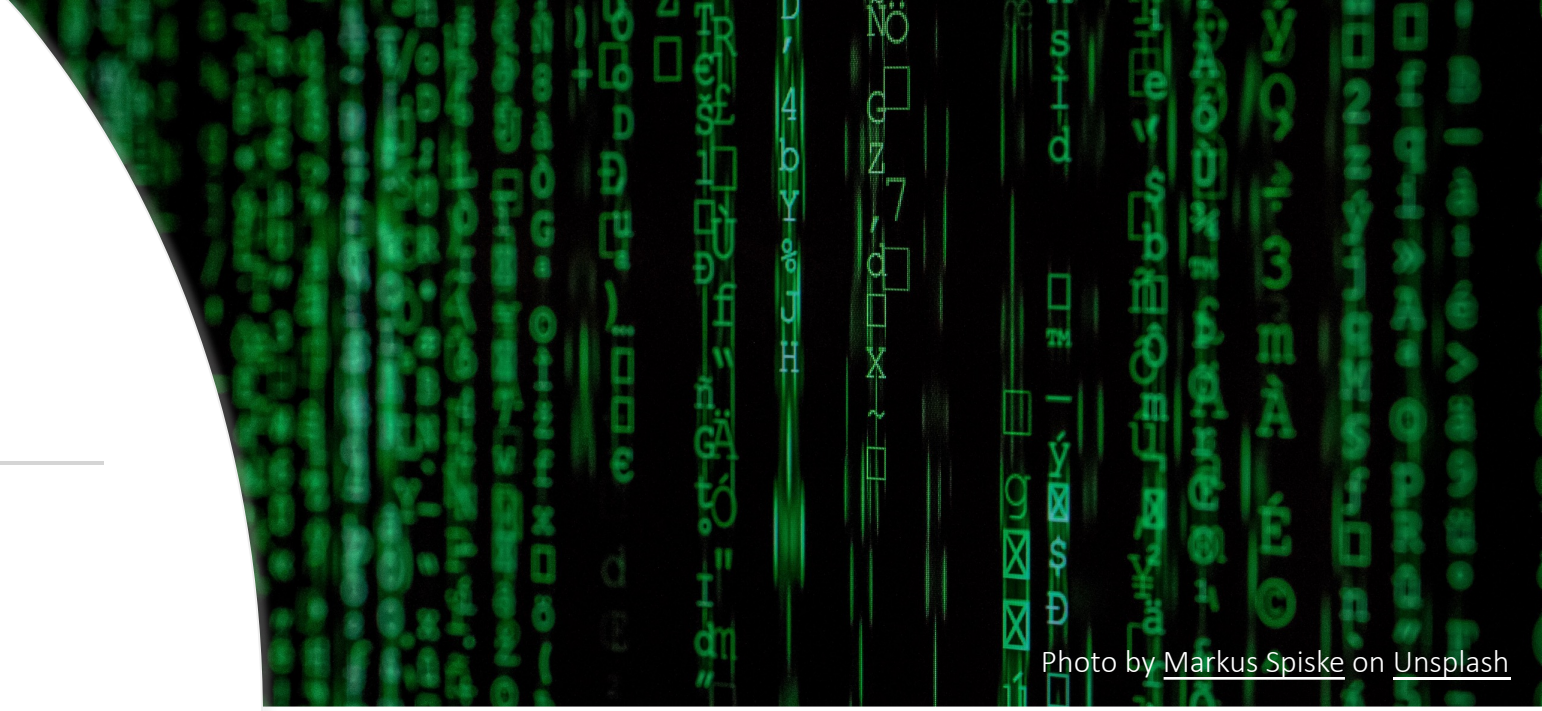


Photo by Markus Spiske on Unsplash

Sara van Erp & Florian van Leeuwen
s.j.vanerp@uu.nl
f.d.vanleeuwen@uu.nl


Photo by Jonathan Greenaway on Unsplash

Recap days 1-2

- Introduction: What is Bayesian analysis? What is a prior?

- How to obtain the posterior?

- Why use Bayes?

- WAMBS-checklist
  - Incl. convergence and prior-predictive checks

Recap days 1-2

- Introduction: What is Bayesian analysis? What is a prior?

- ***How to obtain the posterior?***

- Why use Bayes?

- WAMBS-checklist
  - Incl. ***convergence and prior-predictive checks***

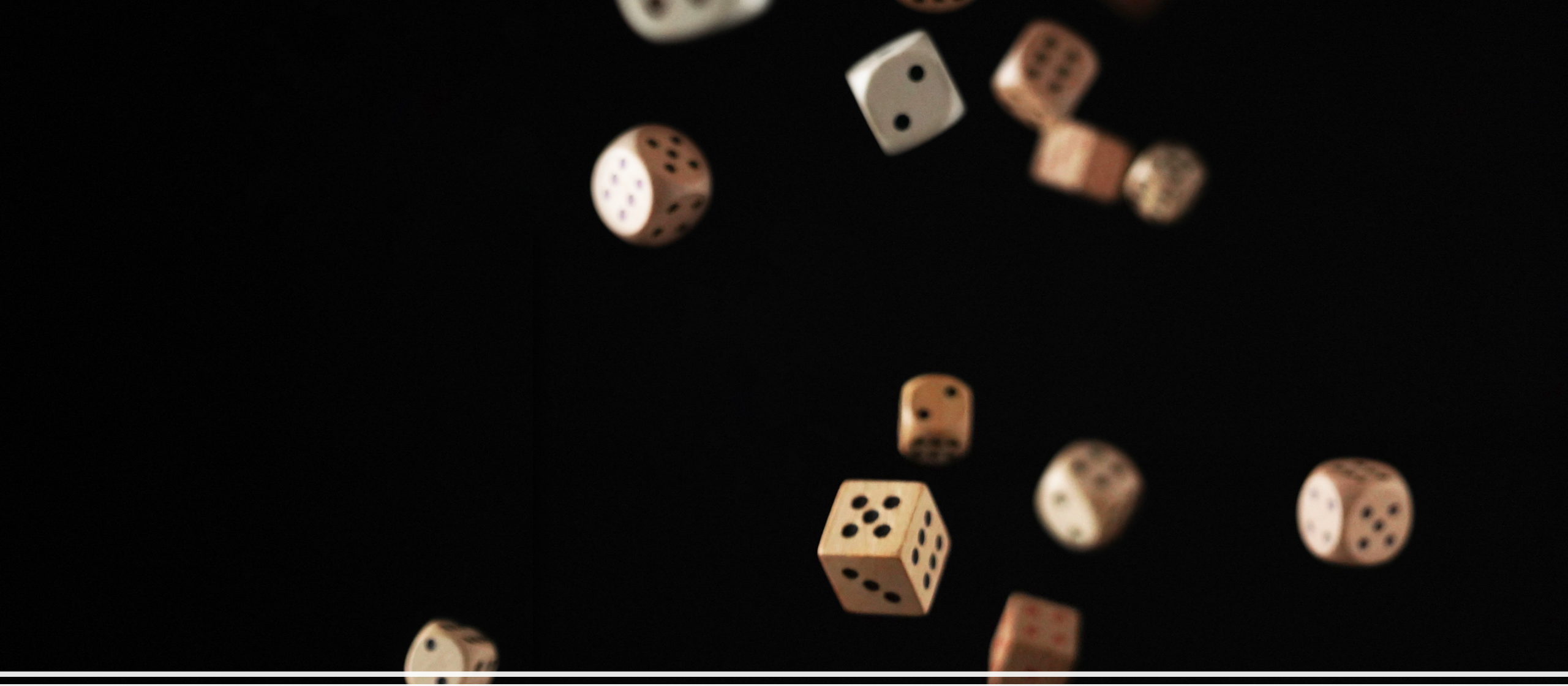
Today

Part 1: Software and algorithms (Florian)

- Different ways to get the posterior
- What is going on (conceptually) under the hood?
- What should you, as user, be aware of?

Part 2: Predictive checks (Sara)

- Posterior predictive checks: how can we check our model?
- Prior predictive checks: how can we check our priors?

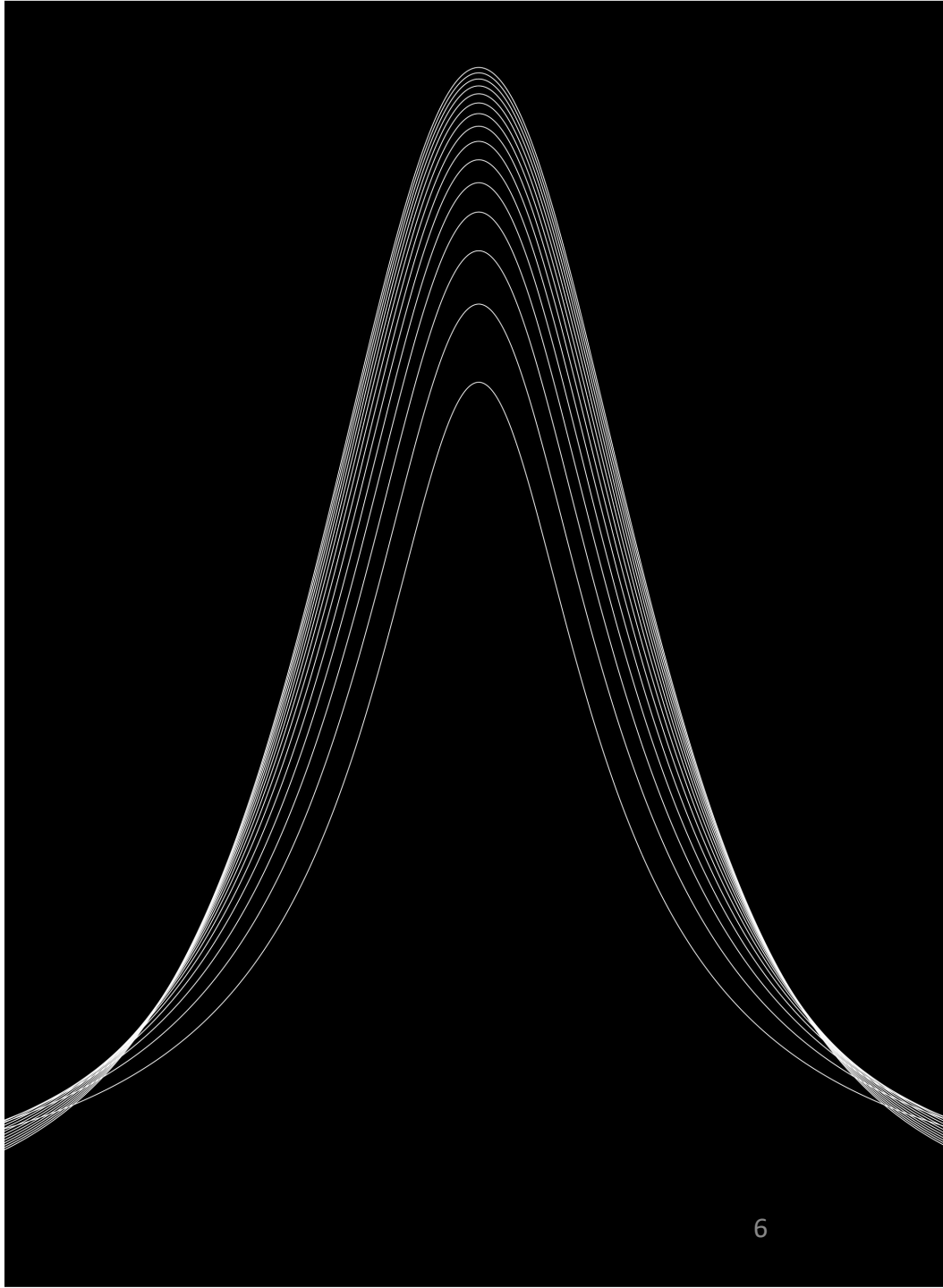


Part 1: Software and algorithms

Photo by Jonathan Greenaway on Unsplash

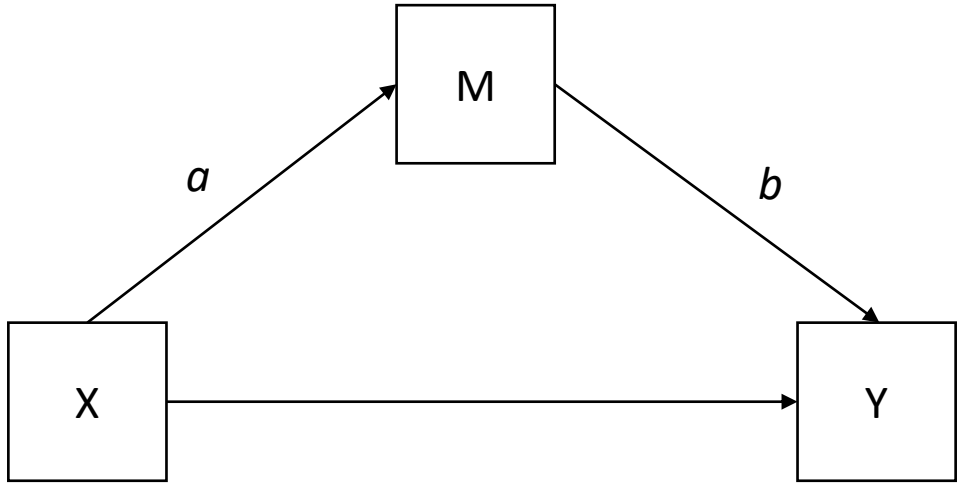
Why use Bayes?

- To include prior information
- More intuitive interpretation
- Technical reasons (estimate more complex models, use smaller samples, model identification)
- ***Full posterior distribution instead of a point estimate***

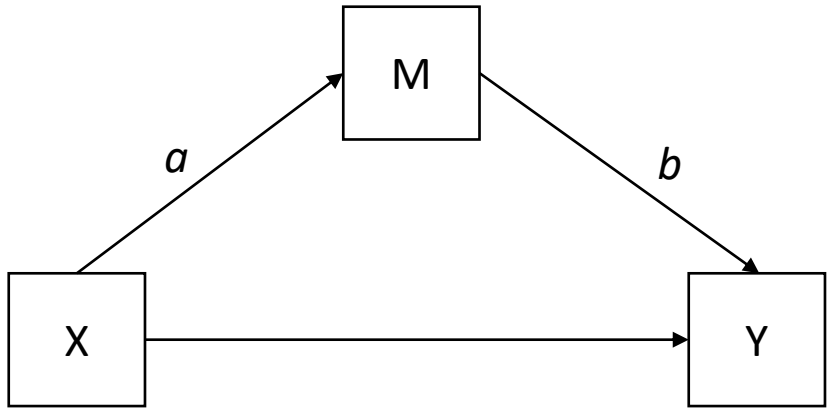


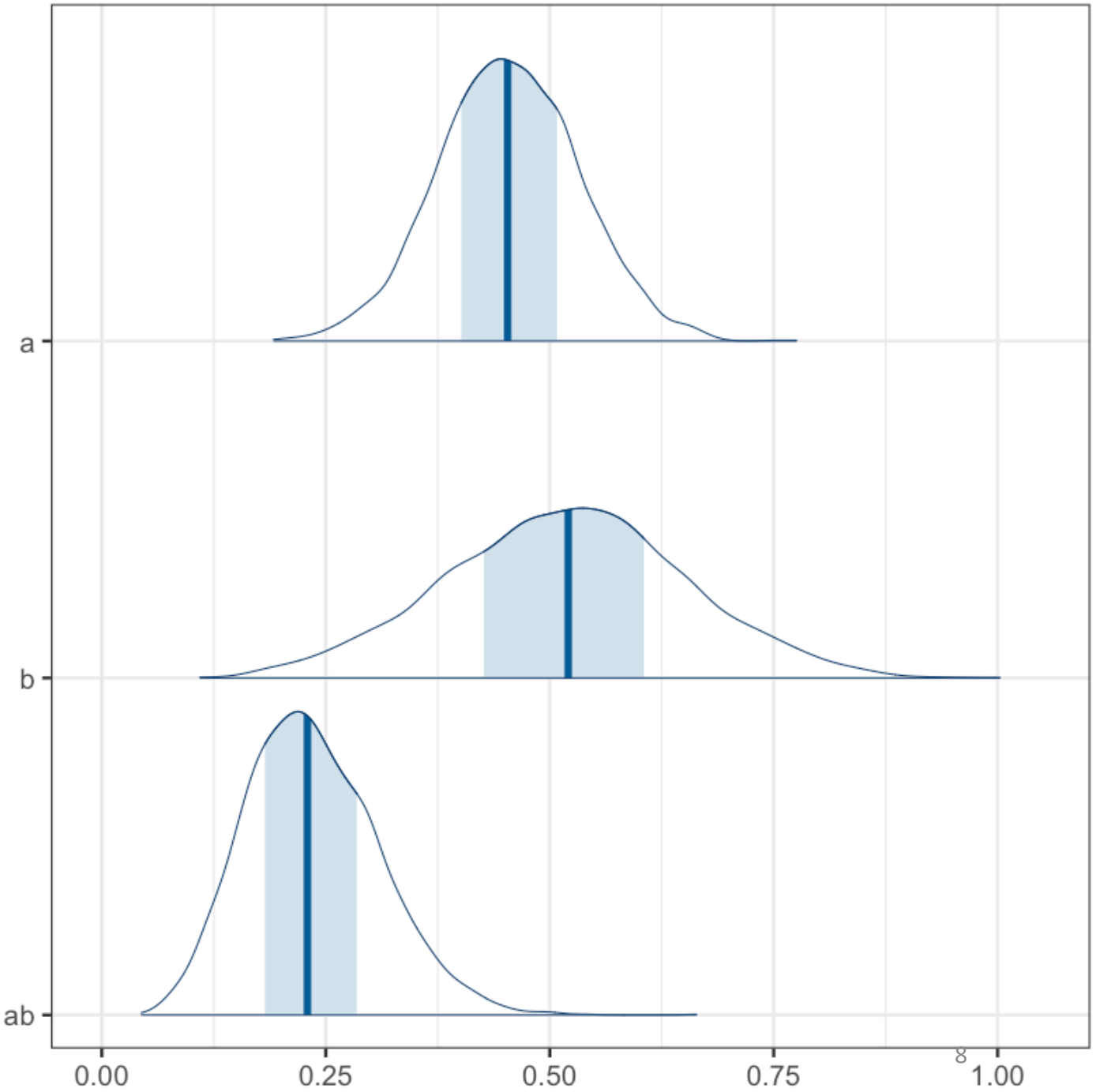
Advantages of the posterior distribution

If we want to estimate an indirect effect, we get automatic uncertainty estimates around functions of parameters.

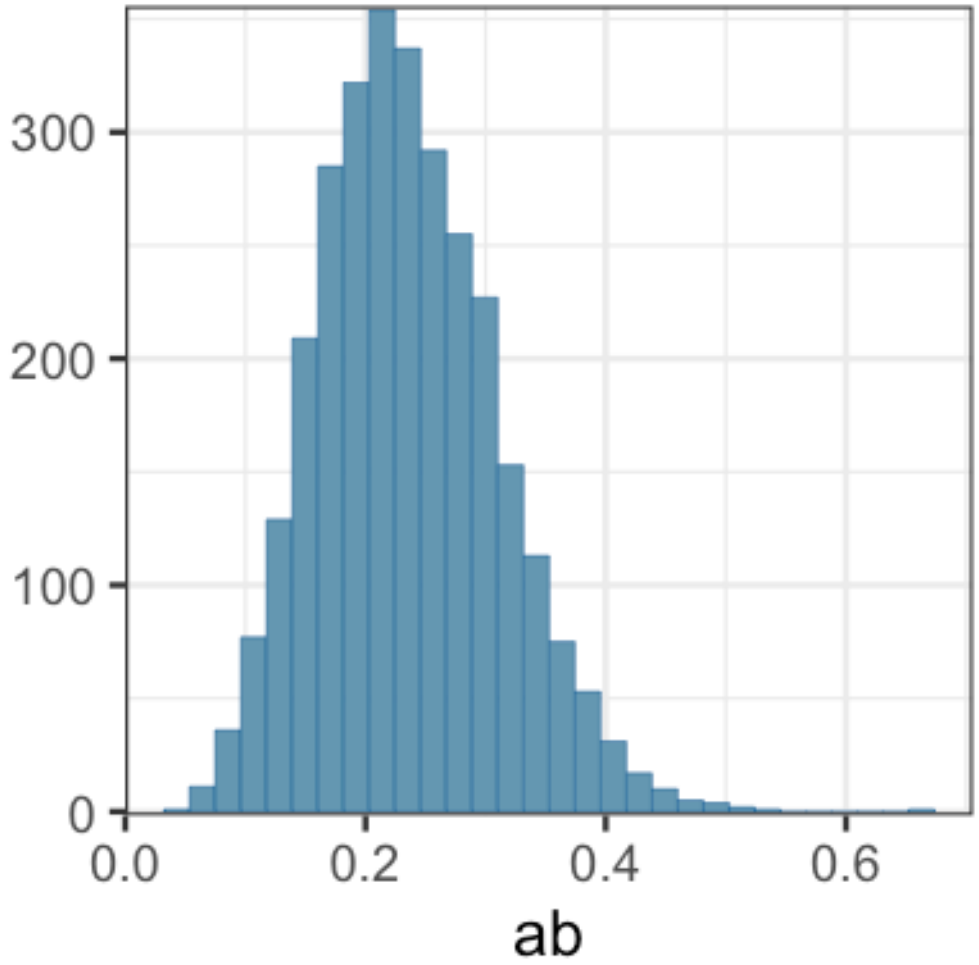


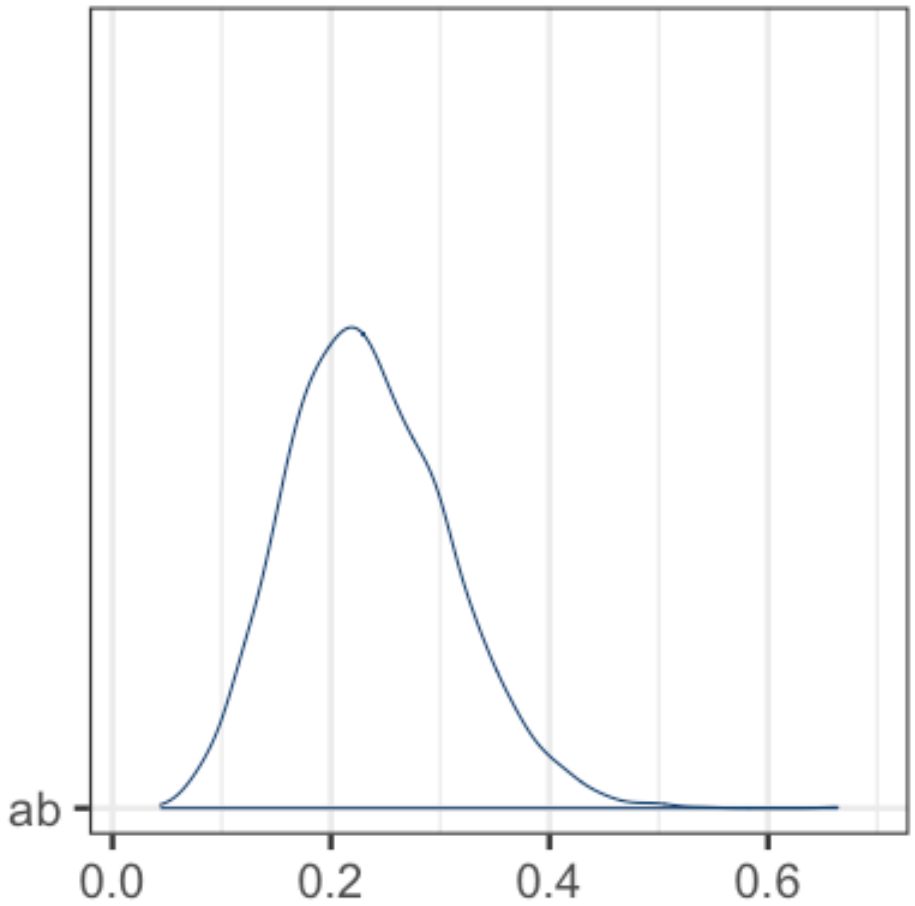
Posterior distribution for the indirect effect *ab*



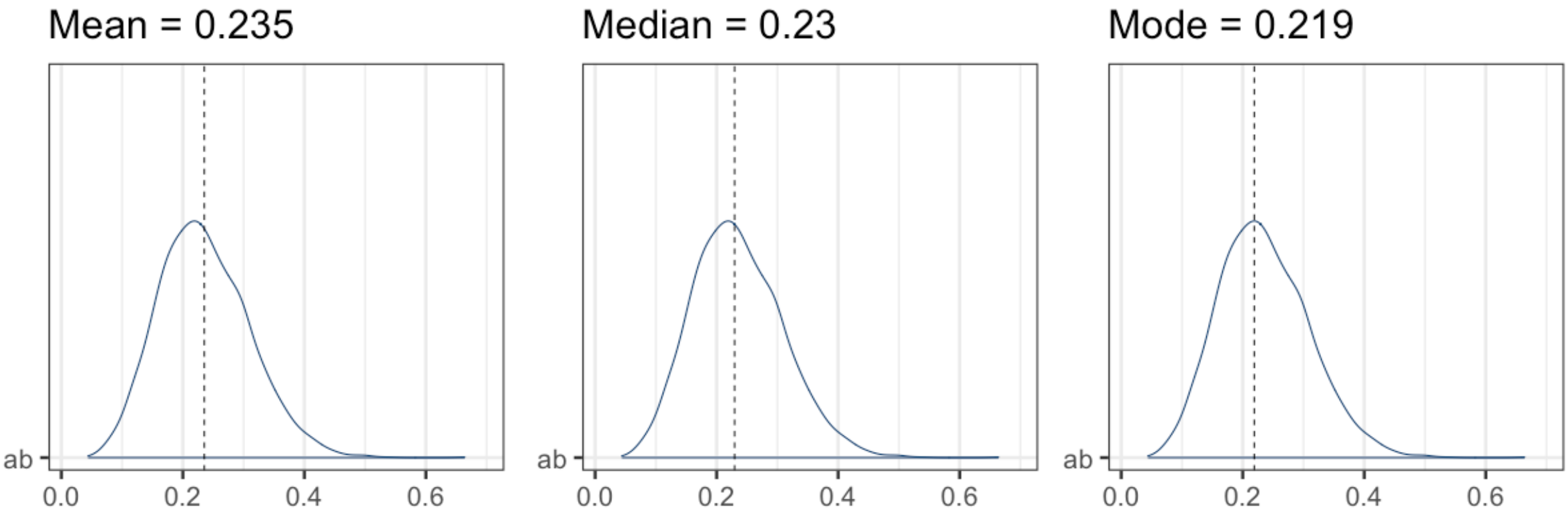


A note on summarizing the posterior





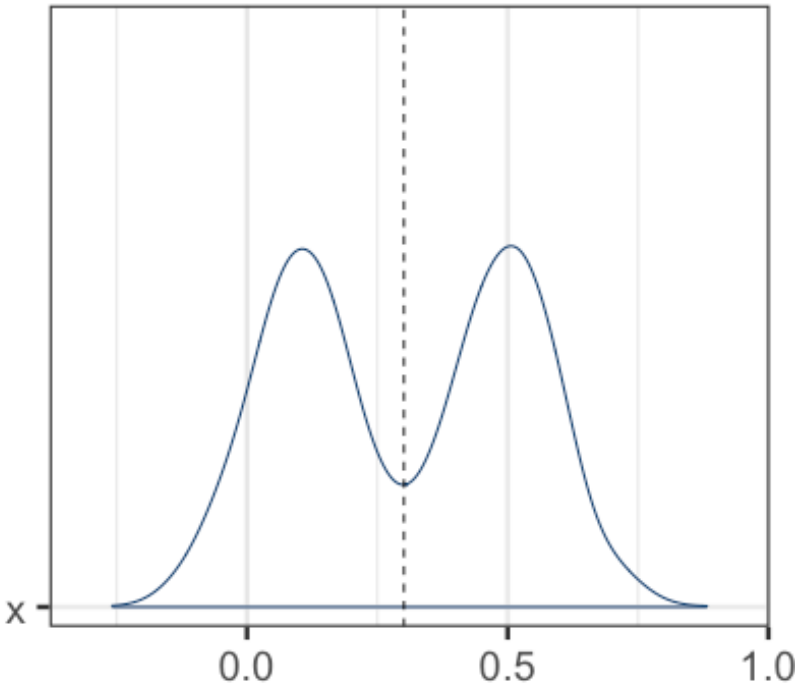
Posterior point estimates



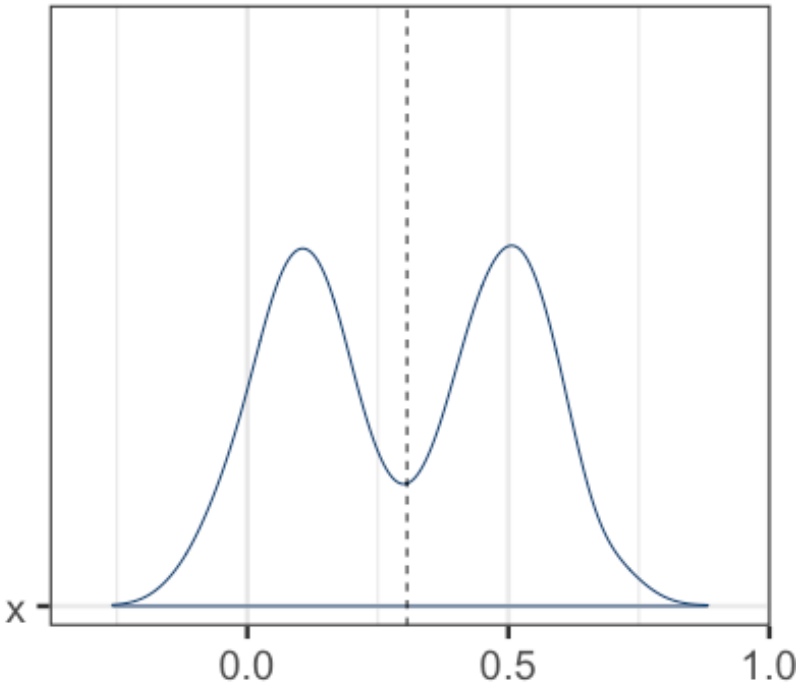
All estimates seem the same, when would this not be the case?

Posterior point estimates: bimodal posterior

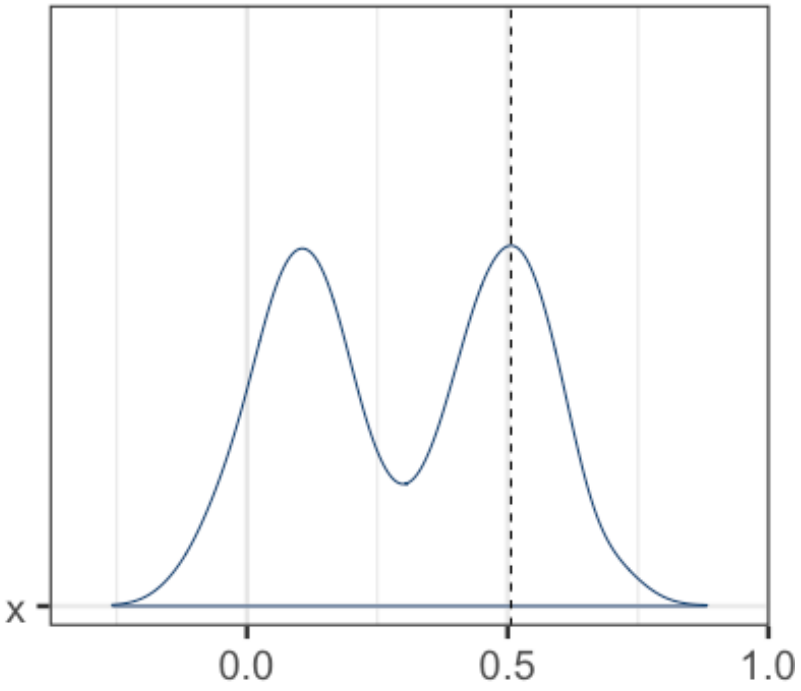
Mean = 0.301



Median = 0.306

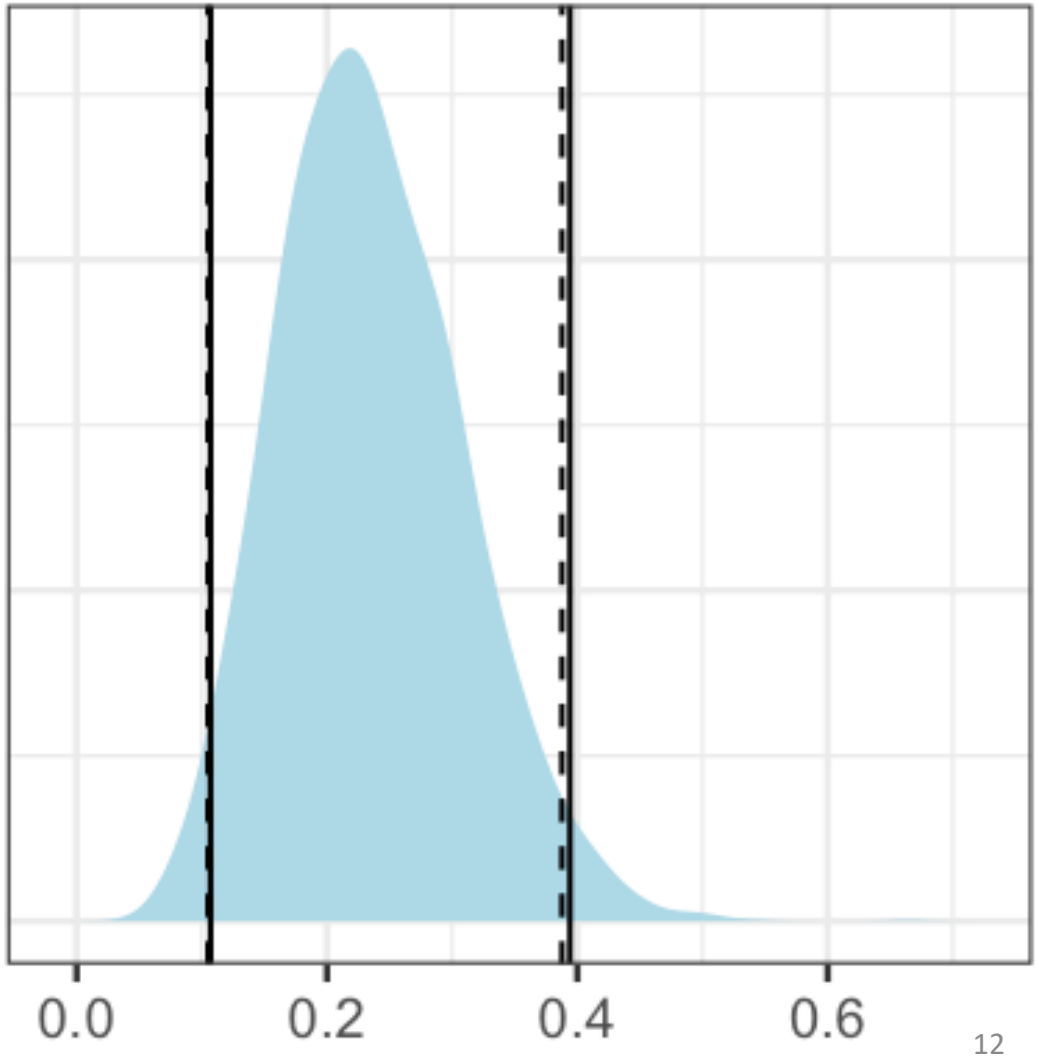


Mode = 0.506?



Posterior credible intervals

Solid line = Equal tailed interval (ETI) -> symmetry
Dashed line = Highest density interval (HDI)

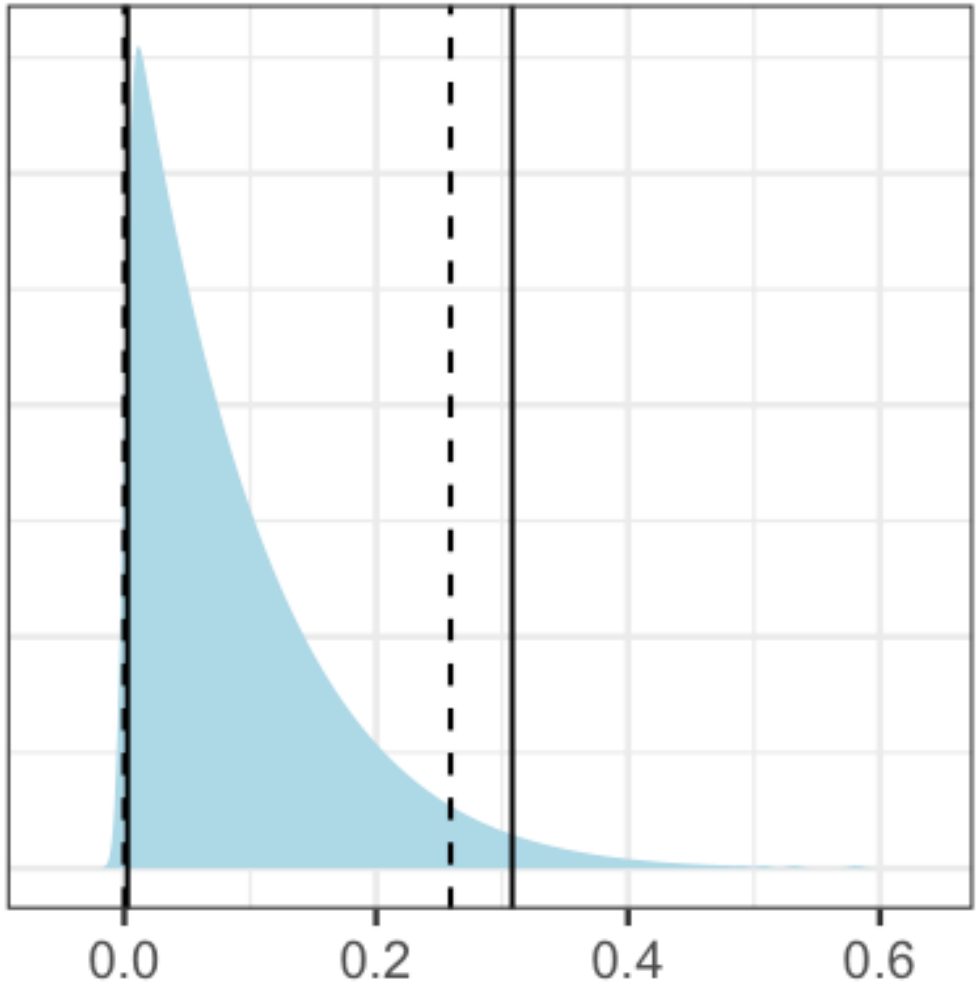


Posterior credible intervals

Solid line = Equal tailed interval (ETI) -> symmetry
Dashed line = Highest density interval (HDI)

Here, the 95% intervals are shown, but we could also compute the 90% intervals, or the 89% intervals... (https://easystats.github.io/bayestestR/articles/credible_interval.html)

ArviZ (the plotting library used in Python) shows 94% HDIs as a friendly reminder of the arbitrary value of choosing any value without justification.



How to obtain the posterior distributions?

In addition:

- Program the conditional posteriors manually
- Closed software, e.g., SPSS, Mplus
- R-packages, e.g., brms, rstanarm, blavaan
- Python libraries: Pyro, bambi

Table 2 A non-exhaustive summary of commonly used and open Bayesian software programs

From: Bayesian statistics and modelling

Software package	Summary
General-purpose Bayesian inference software	
BUGS[231,232]	The original general-purpose Bayesian inference engine, in different incarnations. These use Gibbs and Metropolis sampling. Windows-based software (WinBUGS[233]) with a user-specified model and a black-box MCMC algorithm. Developments include an open-source version (OpenBUGS[234]) also available on Linux and Mac
JAGS[235]	An open-source variation of BUGS that can run cross-platform and can run from R via rjags[236]
PyMC3[237]	An open-source framework for Bayesian modelling and inference entirely within Python; includes Gibbs sampling and Hamiltonian Monte Carlo
Stan[98]	An open-source, general-purpose Bayesian inference engine using Hamiltonian Monte Carlo; can be run from R, Python, Julia, MATLAB and Stata
NIMBLE[238]	Generalization of the BUGS language in R; includes sequential Monte Carlo as well as MCMC. Open-source R package using BUGS/JAGS-model language to develop a model; different algorithms for model fitting including MCMC and sequential Monte Carlo approaches. Includes the ability to write novel algorithms
Programming languages that can be used for Bayesian inference	
TensorFlow Probability[239,240]	A Python library for probabilistic modelling built on Tensorflow[203] from Google
Pyro[241]	A probabilistic programming language built on Python and PyTorch[204]
Julia[242]	A general-purpose language for mathematical computation. In addition to Stan, numerous other probabilistic programming libraries are available for the Julia programming language, including Turing.jl[243] and Mamba.jl[244]
Specialized software doing Bayesian inference for particular classes of models	
JASP[245]	A user-friendly. higher-level interface offering Bayesian analysis. Open source and relies on a collection of open-source R packages
R-INLA[230]	An open-source R package for implementing INLA[246]. Fast inference in R for a certain set of hierarchical models using nested Laplace approximations
GPstuff[247]	Fast approximate Bayesian inference for Gaussian processes using expectation propagation; runs in MATLAB, Octave and R

MCMC, Markov chain Monte Carlo.

Different programs, different algorithms

Exact algorithms (when n -> ∞)

- Simulate from the actual posterior distribution (hopefully)
- Assess convergence to ensure a good representation of the posterior
- Can be slow
- E.g., Gibbs, HMC

Markov Chain Monte Carlo (MCMC) sampling

A class of algorithms to sample from the posterior distribution.

Markov Chain = each state depends only on the previous state

Monte Carlo = repeated sampling

Some examples: Random Walk Metropolis-Hastings, Gibbs sampling, Hamiltonian Monte Carlo.

Monte-Carlo, Monaco by Joachim Lesne on Unsplash

The idea behind MCMC

Bayes' rule: $p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$

With $p(y) = \int p(y|\theta)p(\theta)d\theta$

Or (for the regression model):

$$p(\beta_1, \beta_2, \beta_3, \sigma^2|y) = \frac{p(y|\beta_1, \beta_2, \beta_3, \sigma^2)p(\beta_1, \beta_2, \beta_3, \sigma^2)}{p(y)}$$

With $p(y) = \int\int\int\int p(y|\beta_1, \beta_2, \beta_3, \sigma^2)p(\beta_1, \beta_2, \beta_3, \sigma^2)d\beta_1 d\beta_2 d\beta_3 d\sigma^2$

These intergrals can become untracable (cannot be reduced to "standard" functions).

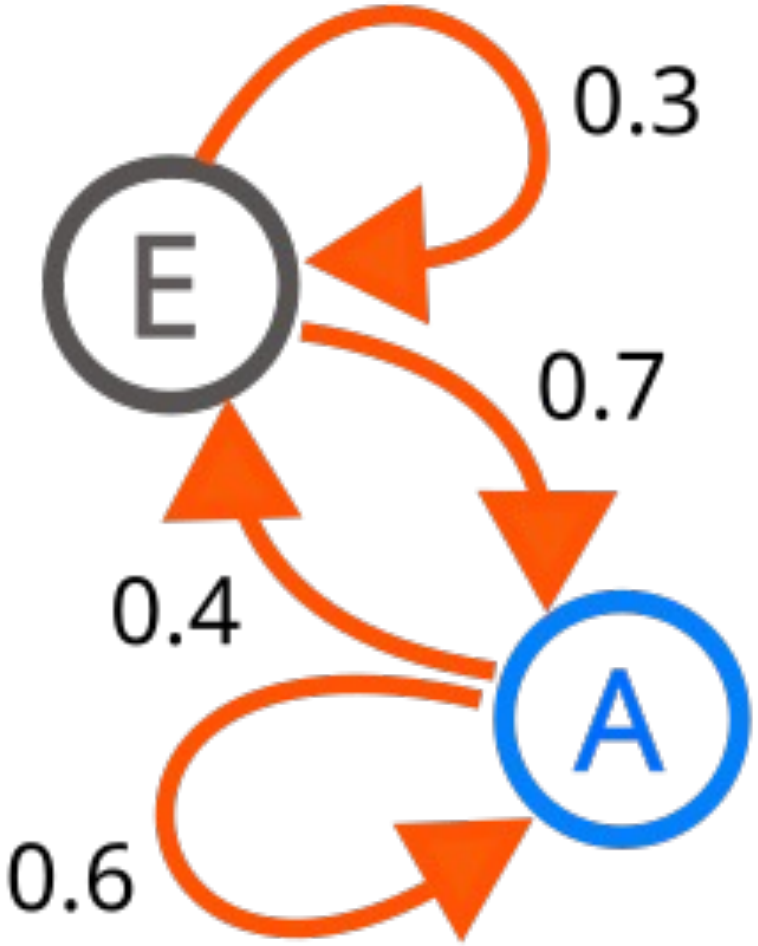
Metropolis-Hastings (MH)

Random walk version is “simplest” MCMC algorithm.

We use some (arbitrary) proposal density to sample from and either accept or reject a new draw.

We can then directly sample from $p(\theta|y)$

Intermezzo: Markov chains



What could be a problem with using Markov Chains?

https://en.wikipedia.org/wiki/Markov_chain

Metropolis-Hastings (MH)

Step 1: Sample from a proposal density (e.g., normal distribution)

Step 2: Propose a new draw from a normal distribution centered around the value from 1, with some SD (stepsize)

Step 3: Calculate r = value unnormalized posterior current iteration/unnormalized posterior previous iteration

Step 4: Decision: if r > u ~ U(0, 1) accept draw, otherwise reject

With r = P(theta_t+1 | D) / P(theta_t | D)

Disadvantage: can be slow (especially if rejection probability is high)

Metropolis-Hastings (MH)

Random walk version is “simplest” MCMC algorithm.

We use some (arbitrary) proposal density to sample from and either accept or reject a new draw.

Gibbs sampling is actually a special case of MH with an acceptance probability of 1.

- Advantage: no risk of rejecting many proposals
- Disadvantage: requires derivation of conditional posteriors

Gibbs sampler (see day 1)

1. Assign starting values
2. Sample β_1 from conditional distribution
3. Sample β_2 from conditional distribution
4. Sample β_3 from conditional distribution
5. Sample σ^2 from conditional distribution
6. Go to step 2 and repeat

Gibbs sampler: Conditional posteriors

Instead of sampling from the difficult $p(\beta_1, \beta_2, \beta_3, \sigma^2|y)$ we use the conditional posteriors:

$$Post(\beta_1|\beta_2, \beta_3, \sigma^2, data) \sim Prior(\beta_1) \times likelihood$$
$$Post(\beta_2|\beta_1, \beta_3, \sigma^2, data) \sim Prior(\beta_2) \times likelihood$$
$$Post(\beta_3|\beta_1, \beta_2, \sigma^2, data) \sim Prior(\beta_3) \times likelihood$$
$$Post(\sigma^2|\beta_1, \beta_2, \beta_3, data) \sim Prior(\sigma^2) \times likelihood$$

These conditional posteriors can be derived when conjugate priors are used.

Gibbs sampler: conjugate priors

Likelihood $p(x_i\|\theta)$	Model parameters θ	Conjugate prior (and posterior) distribution $p(\theta\|\Theta), p(\theta\|\mathbf{x},\Theta) = p(\theta\|\Theta')$	Prior hyperparameters Θ	Posterior hyperparameters[note 1] Θ'	Interpretation of hyperparameters	Posterior predictive[note 5] $p(\tilde{x}\|\mathbf{x},\Theta) = p(\tilde{x}\|\Theta')$
Normal with known variance σ^2	μ (mean)	Normal	$\mu_0,\ \sigma_0^2$	$\frac{1}{\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}}\left(\frac{\mu_0}{\sigma_0^2}+\frac{\sum_{i=1}^n x_i}{\sigma^2}\right), \left(\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}\right)^{-1}$	mean was estimated from observations with total precision (sum of all individual precisions) $1/\sigma_0^2$ and with sample mean μ_0	$\mathcal{N}(\tilde{x}\|\mu_0', \sigma_0^{2\prime}+\sigma^2)$[4]
Normal with known precision τ	μ (mean)	Normal	$\mu_0,\ \tau_0^{-1}$	$\frac{\tau_0\mu_0+\tau\sum_{i=1}^n x_i}{\tau_0+n\tau}, (\tau_0+n\tau)^{-1}$	mean was estimated from observations with total precision (sum of all individual precisions) τ_0 and with sample mean μ_0	$\mathcal{N}\left(\tilde{x}\mid\mu_0', \frac{1}{\tau_0'}+\frac{1}{\tau}\right)$[4]
Normal with known mean μ	σ^2 (variance)	Inverse gamma	$\alpha,\ \beta$ [note 6]	$\alpha+\frac{n}{2}, \beta+\frac{\sum_{i=1}^n (x_i-\mu)^2}{2}$	variance was estimated from 2α observations with sample variance β/α (i.e. with sum of squared deviations 2β, where deviations are from known mean μ)	$t_{2\alpha'}(\tilde{x}\|\mu, \sigma^2=\beta'/\alpha')$ [4]

https://en.wikipedia.org/wiki/Conjugate_prior

Hamiltonian Monte Carlo (HMC)

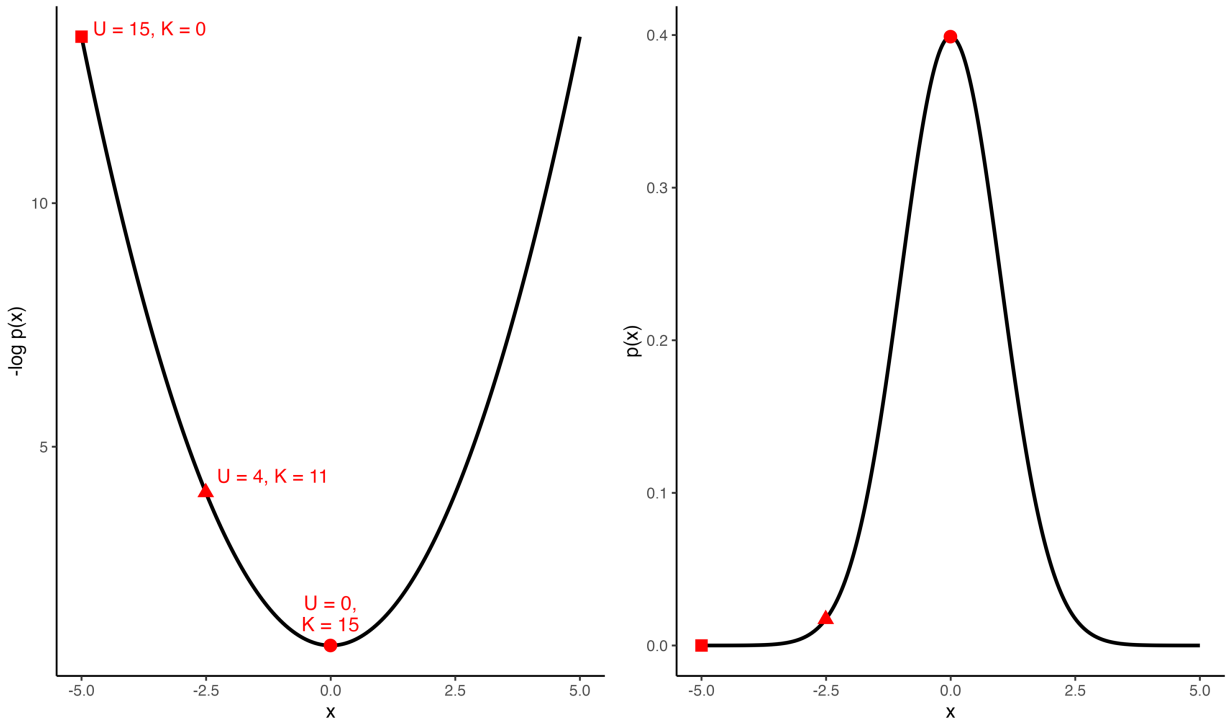
- Another special case of Metropolis-Hastings
- Stan uses the No-U-Turn-Sampler (NUTS), an extension to HMC

Remember the proposal for a next step in MH? HMC uses information (gradient) from the target distribution (the posterior) to inform the proposal.

- Advantage: lower autocorrelation (but can take longer per iteration) and only rejection when the approxiation of the path fails
- Disadvantage: requires the derivatives (discrete parameters not possible)

Hamiltonian Monte Carlo (HMC)

What happens if we would only use the gradient?



Potential + kinetic Energy

$$H(\theta, m) = U(\theta) + K(m)$$

To simulate the path we need the partial derivatives:

$$\frac{d\theta}{dt} = \frac{dH}{dm},$$

$$\frac{dm}{dt} = -\frac{dH}{d\theta},$$

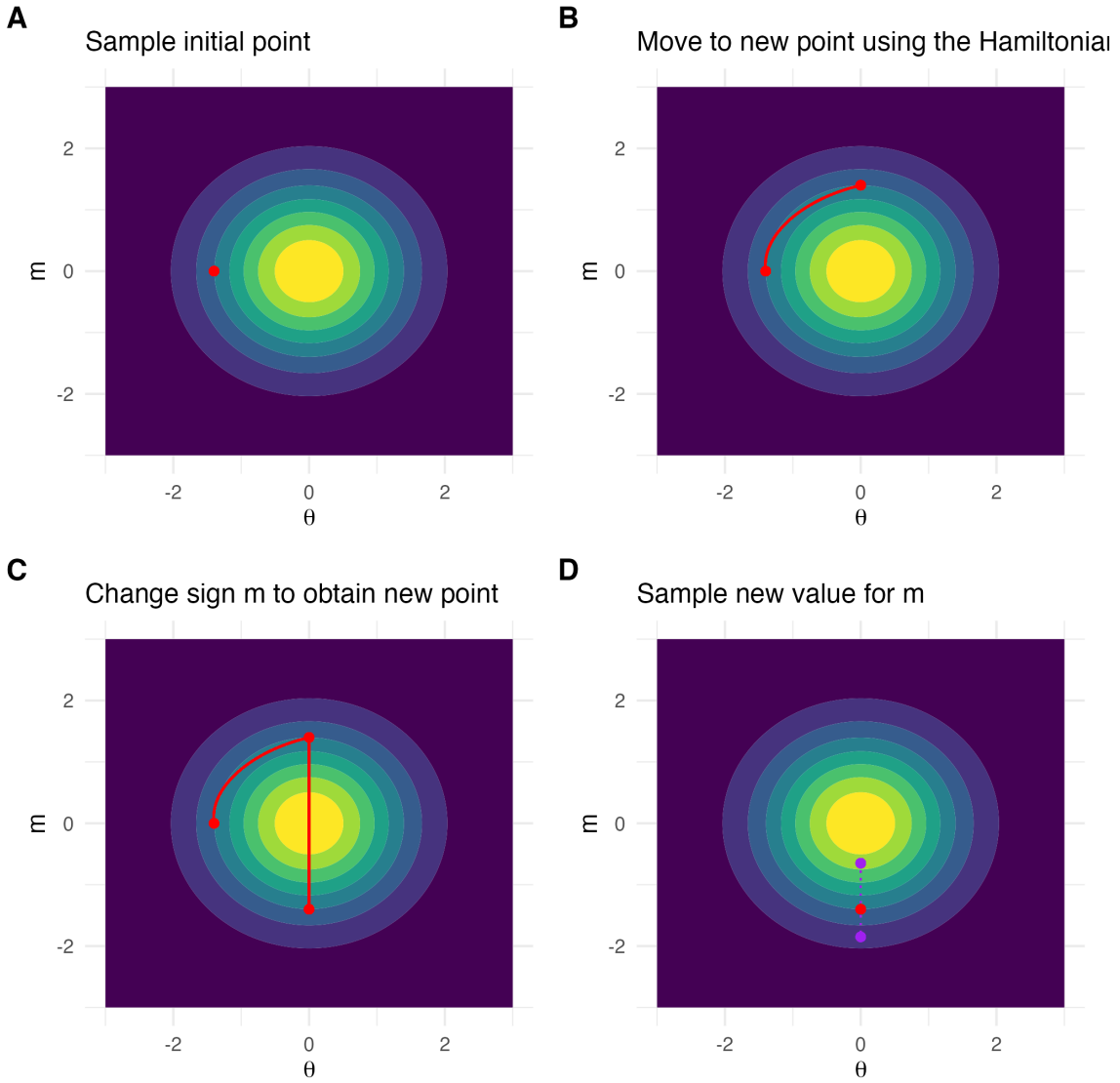
Often this is very hard, it is preffered to estimate the path with leapfrog method. This is not perfect, so we still need to reject some samples:

$$r = \exp(H(\theta_t, m_t) - H(\theta_{t+1}, m_{t+1})$$

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, *2*(11), 2

Monnahan, C. C., Thorson, J. T., & Branch, T. A. (2017). Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo. *Methods in Ecology and Evolution*, *8*(3), 339-348

Hamiltonian Monte Carlo (HMC)



The length of the path is decided by:

- Step size
- Number of steps

Interactive demo

Interactive gallery of various MCMC algorithms:

http://chi-feng.github.io/mcmc-demo/

So, what should I know?

- Traditionally, software relied on Gibbs sampling (e.g., JAGS, Mplus)
- Stan and R-packages using Stan rely on Hamiltonian Monte Carlo (HMC)
- Both are special cases of Metropolis-Hastings
- Generally, HMC exhibits less autocorrelation, so less iterations needed
- HMC offers more convergence diagnostics, but cannot sample discrete parameters.

Different programs, different algorithms

What are ways to obtain quicker estimates?

Exact algorithms (when n -> ∞)

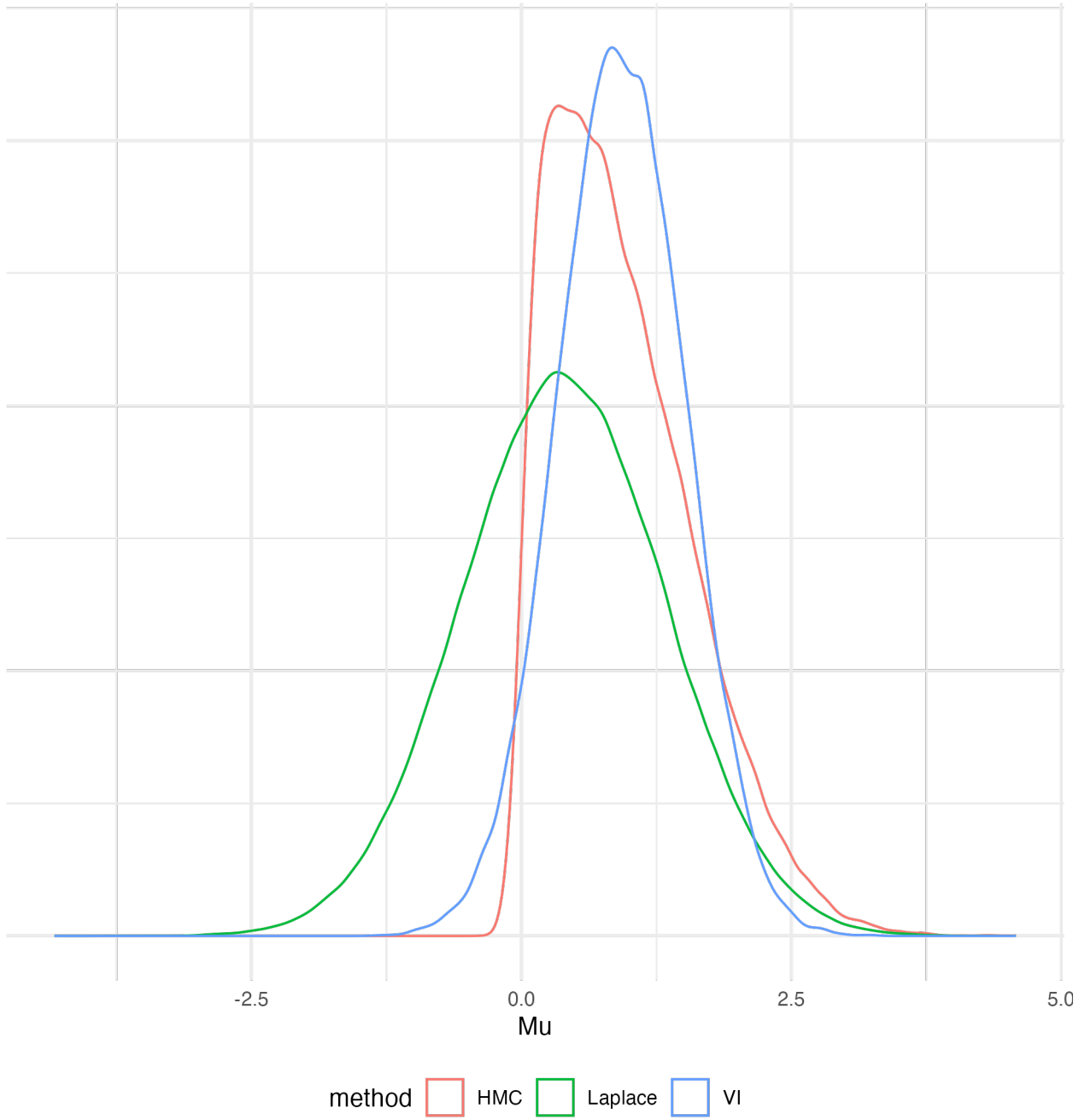
- Simulate from the actual posterior distribution (hopefully)
- Assess convergence to ensure a good representation of the posterior
- Can be slow
- E.g., Gibbs, HMC

Approximate algorithms

- Approximate the posterior distribution with a simple, “comparable” distribution and optimize this distribution
- Assess convergence to ensure the approximation is close enough
- Fast and scalable
- E.g., Laplace (INLA), variational inference, Pathfinder

Non-mcmc methods

- Contrain the shape to a known distribution
- Estimate the parameters with gradient descent (or closed form)



How do we approximate non-mcmc method?

Assume a known distribution (e.g., normal) for our approximate posterior q(θ)

Minimize difference between q(θ) and p(θ) using Kullback-Leibler (KL) divergence:

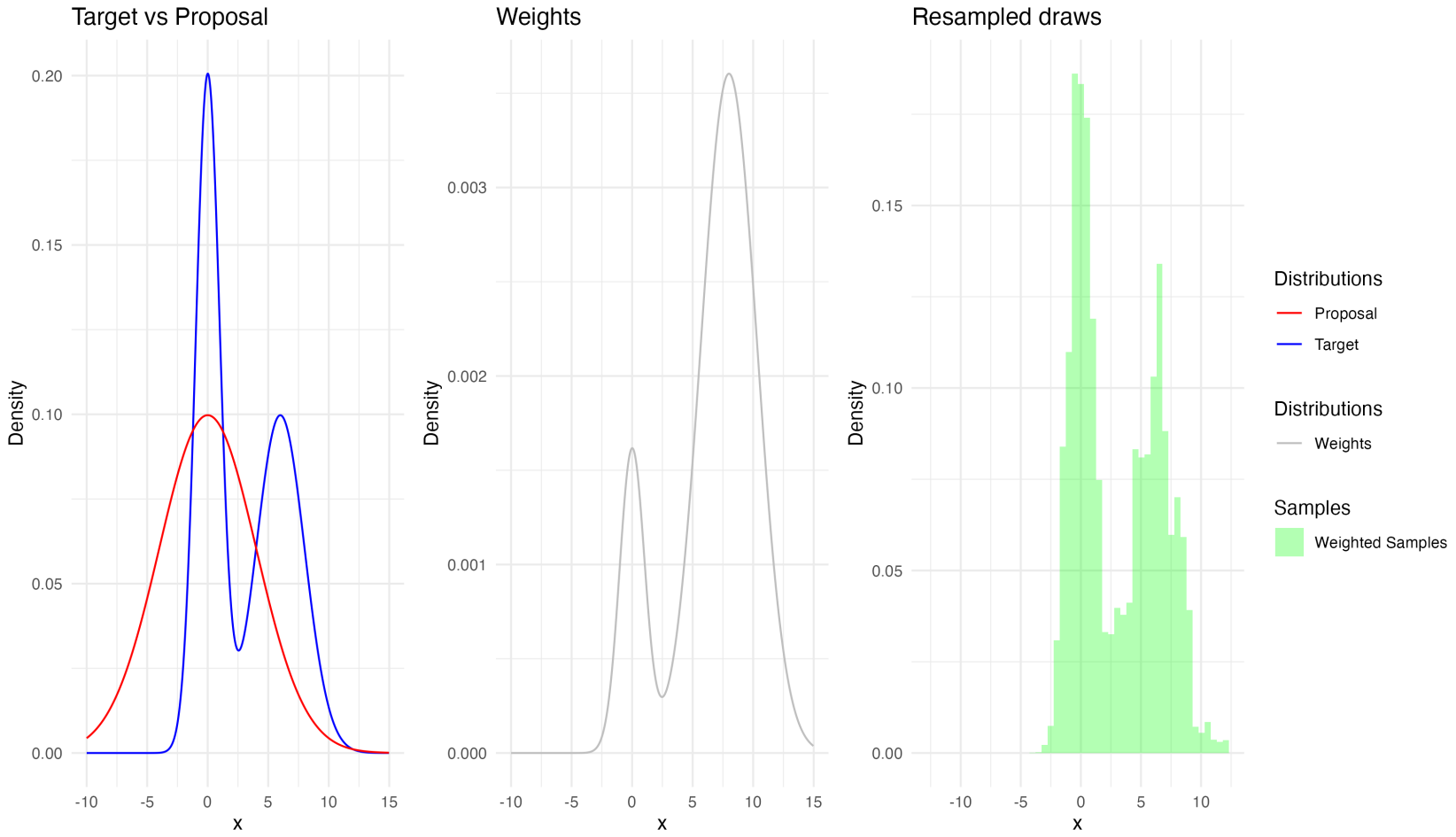
$$\mathrm{KL}(q(\theta)||p(\theta|D)) = E_{q(\theta)}\left[\log \frac{q(\theta)}{p(\theta|D)}\right]$$

Estimate the parameters of the Normal (mu, sigma) of q(θ) by minimising the evidence lower bound:

$$\mathrm{ELBO}(q) = E_{q(\theta)}[\log p(\theta, D)] - E_{q(\theta)}[\log q(\theta)]$$

(pareto smooted) Importance sampling

- Can we make our samples any better?
- Yes! Use importance sampling
- PSIS even better -> Pareto k value
- Downside: this can cause large varience in estimates



Different programs, different algorithms

Exact algorithms (when n -> ∞)

- Usually the way to go with small models

Approximate algorithms

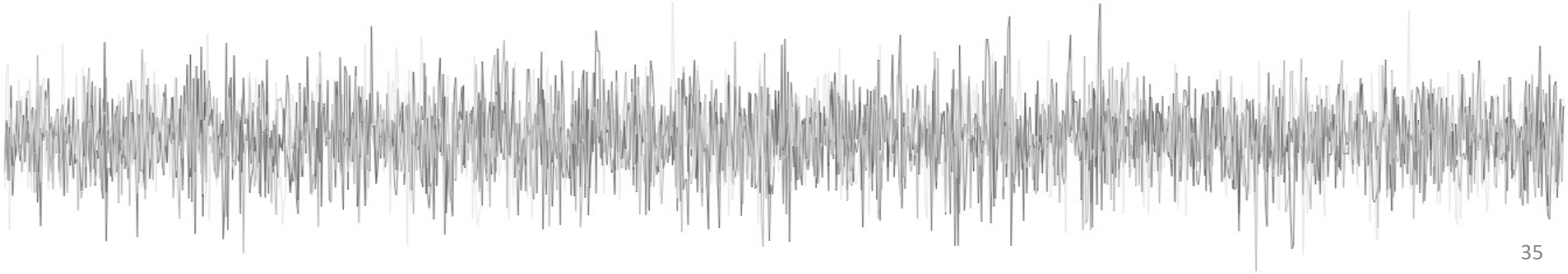
- Helps with model bulding, simulation studies or very complex models

Convergence in Stan: MCMC

Simple models will generally run

Potential solutions more complex, non-converging models:

- Change sampler settings
- Change the prior
- Change the model



Convergence in Stan: MCMC

1. Traceplots should look like fat caterpillars
2. Rhat should be close to 1
3. Effective sample size should be large enough (e.g., 400 with 4 chains)
4. No low BFMI warning
5. No divergent transitions

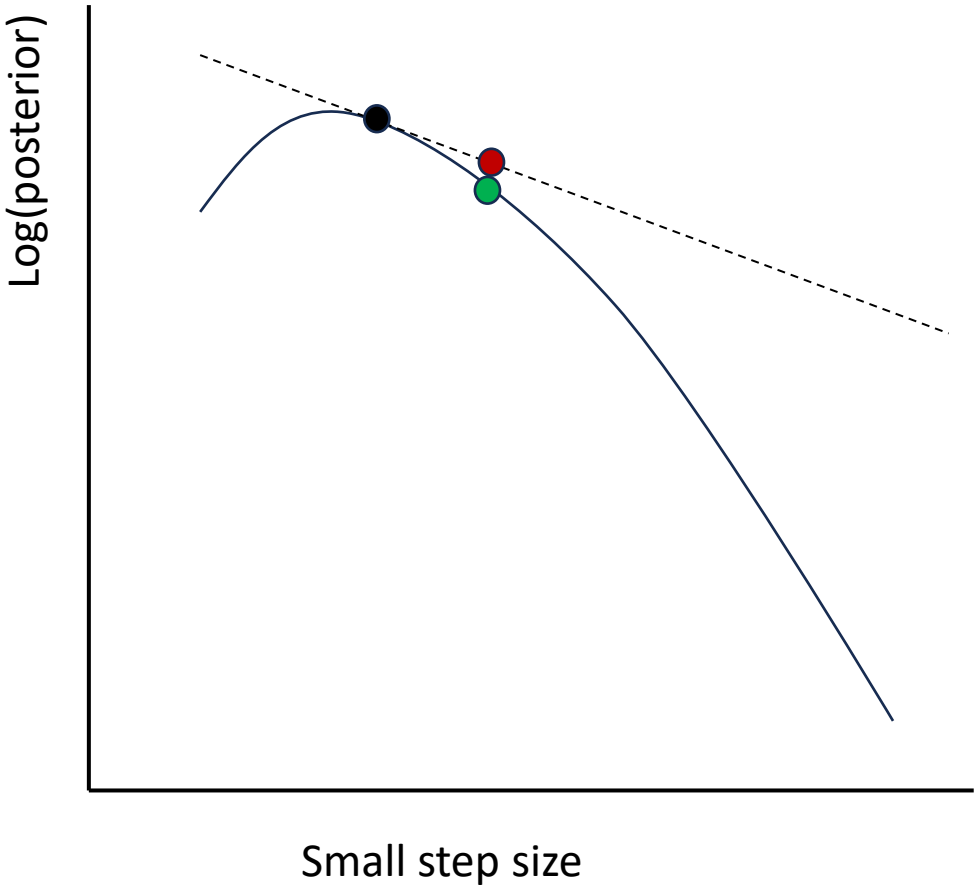
”Max. treedepth” exceeded is an efficiency concern.

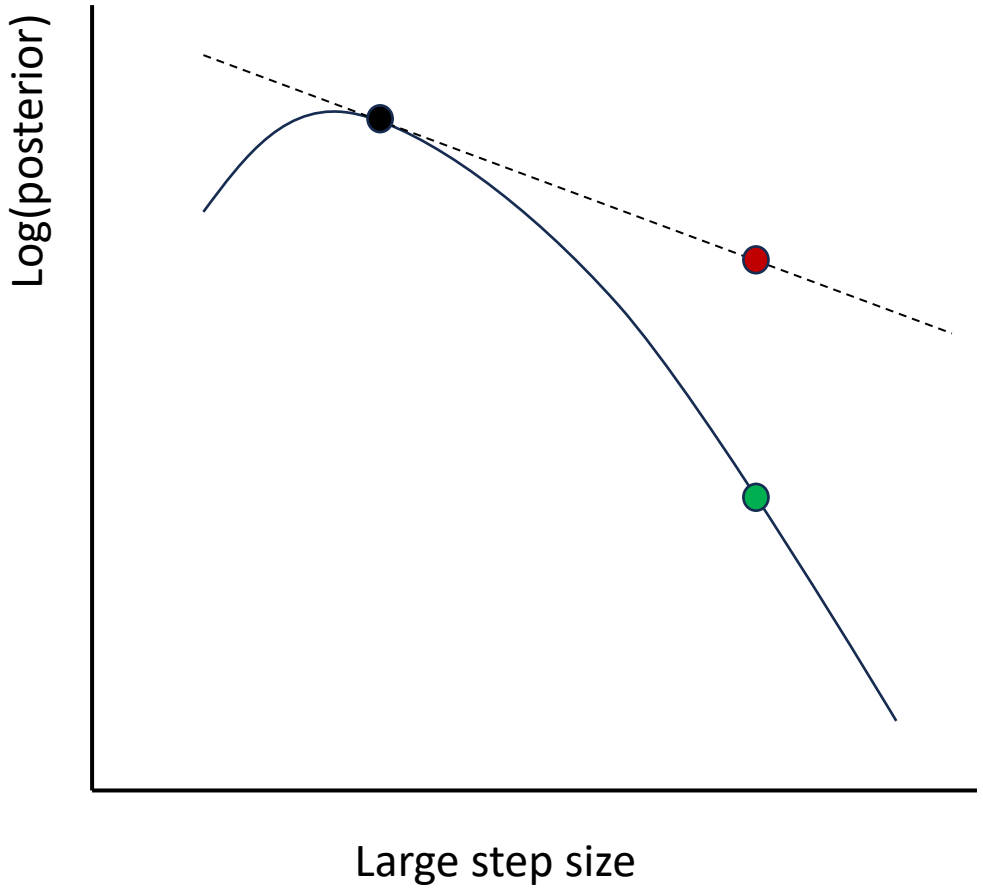
Convergence in Stan: MCMC

1. Traceplots should look like fat caterpillars
2. Rhat should be close to 1
3. Effective sample size should be large enough (e.g., 400 with 4 chains)
4. No low BFMI warning
5. No divergent transitions

Potential solution 1-4: increase number of iterations

Divergent transitions in Stan





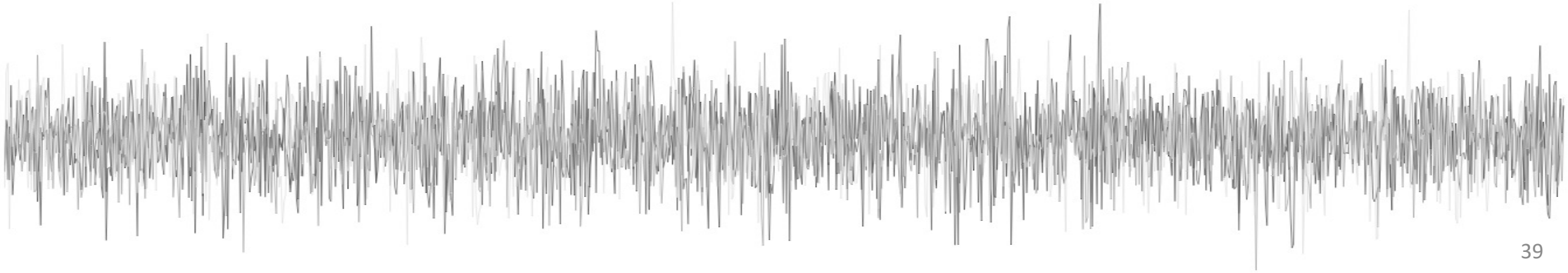
Based on: https://www.martinmodrak.cz/2018/02/19/taming-divergences-in-stan-models/

Convergence in Stan: MCMC

- Important, but difficult topic

- See the Markdown for a brms example (https://utrechtuniversity.github.io/BayesianEstimation/content/wednesday/convergence_checks.html)

- See: https://mc-stan.org/misc/warnings.html for a general overview



Convergence in Stan: non-MCMC

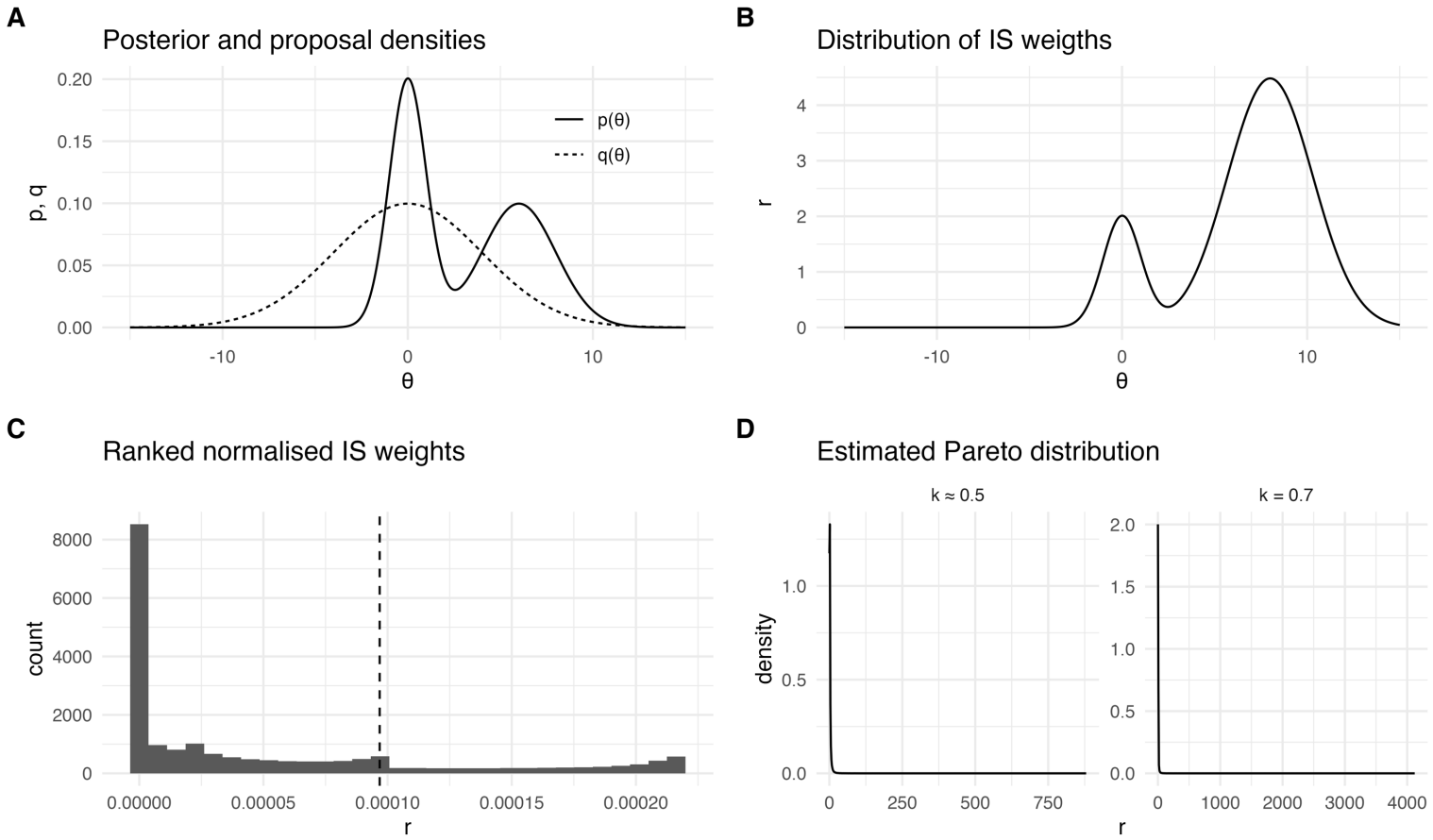
Importance weights: $r_s = \frac{p(\theta_s)}{q(\theta_s)}$

Cutoff : $M = \min(0.2S, 3\sqrt{S}),$

Pareto distribution = GPD(mu, sigma, k)

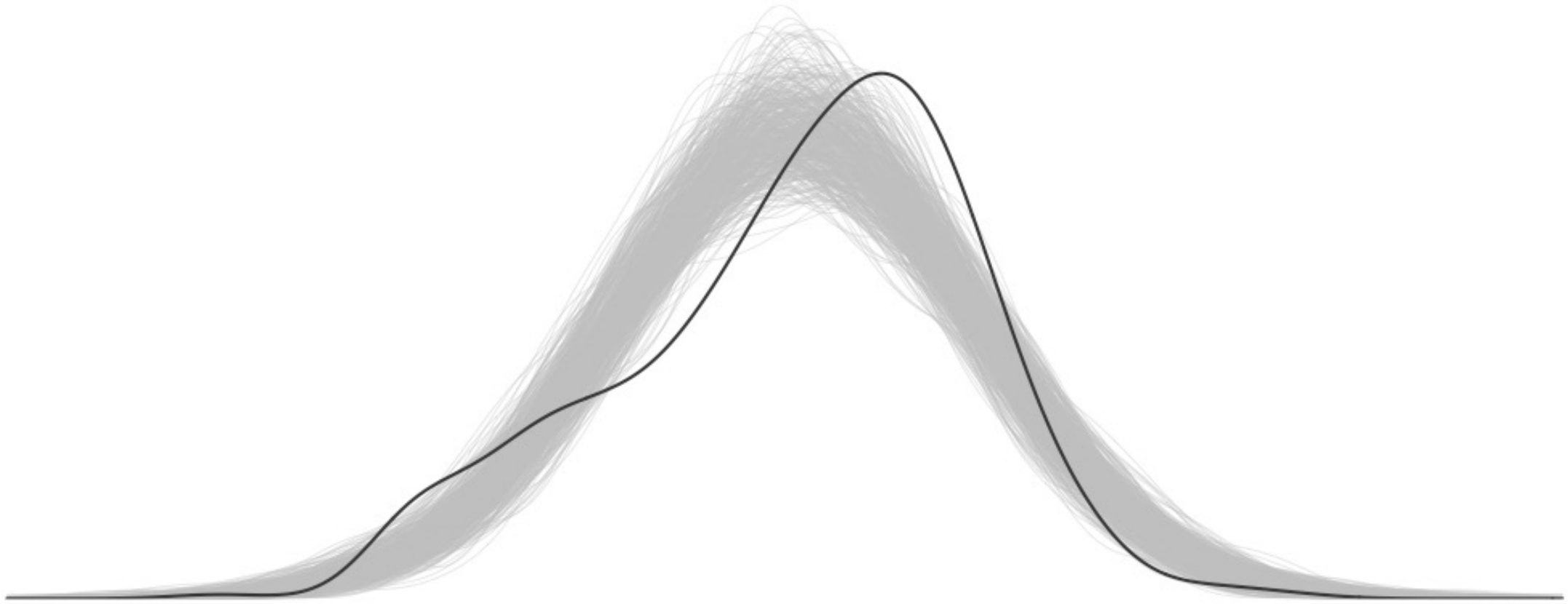
Number of moments = 1/k

k < 0.5 -> variance is known (0.7 also ok)



Vehtari, A., Simpson, D., Gelman, A., Yao, Y., & Gabry, J. (2024). Pareto smoothed importance sampling. *Journal of Machine Learning Research*, *25*(72), 1-58.

Part 2: Predictive checks



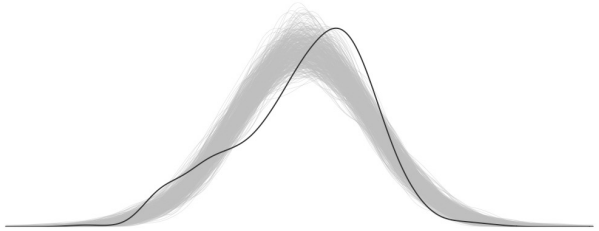
Why check your model?

All models are simplifications -> Do we capture the characteristics we care about?

Important consideration: What is the purpose of our model?

Note: “Model” includes the prior, likelihood, included explanatory variables, hierarchical considerations, etc..

Posterior predictive checks

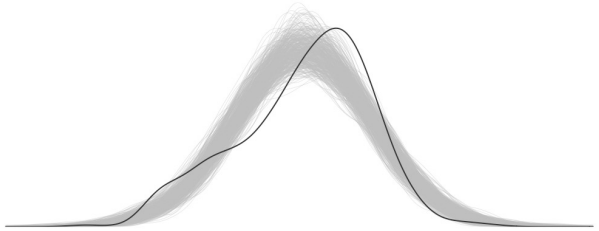


Data generated from the model should resemble the observed data.

Specifically: generate data from the *joint posterior predictive distribution* and compare.

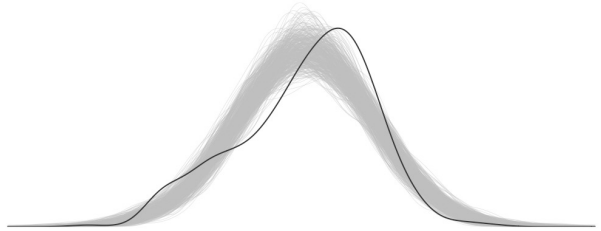
Suppose we have measured the IQ of 20 people. We assume $x \sim N(\mu, \sigma)$ and specify a prior for μ and σ. We sample μ and σ from the posterior distribution and then generate replicated data sets based on these values.

Posterior predictive checks



Posterior sample	μ	σ	Generated data
1	101	16	$x \sim N(101, 16)$
2	98	15	$x \sim N(98, 15)$
3	100	14	$x \sim N(100, 14)$
4	105	18	$x \sim N(105, 18)$
5	97	19	$x \sim N(97, 19)$
6	99	15	$x \sim N(99, 15)$
…	…	…	…

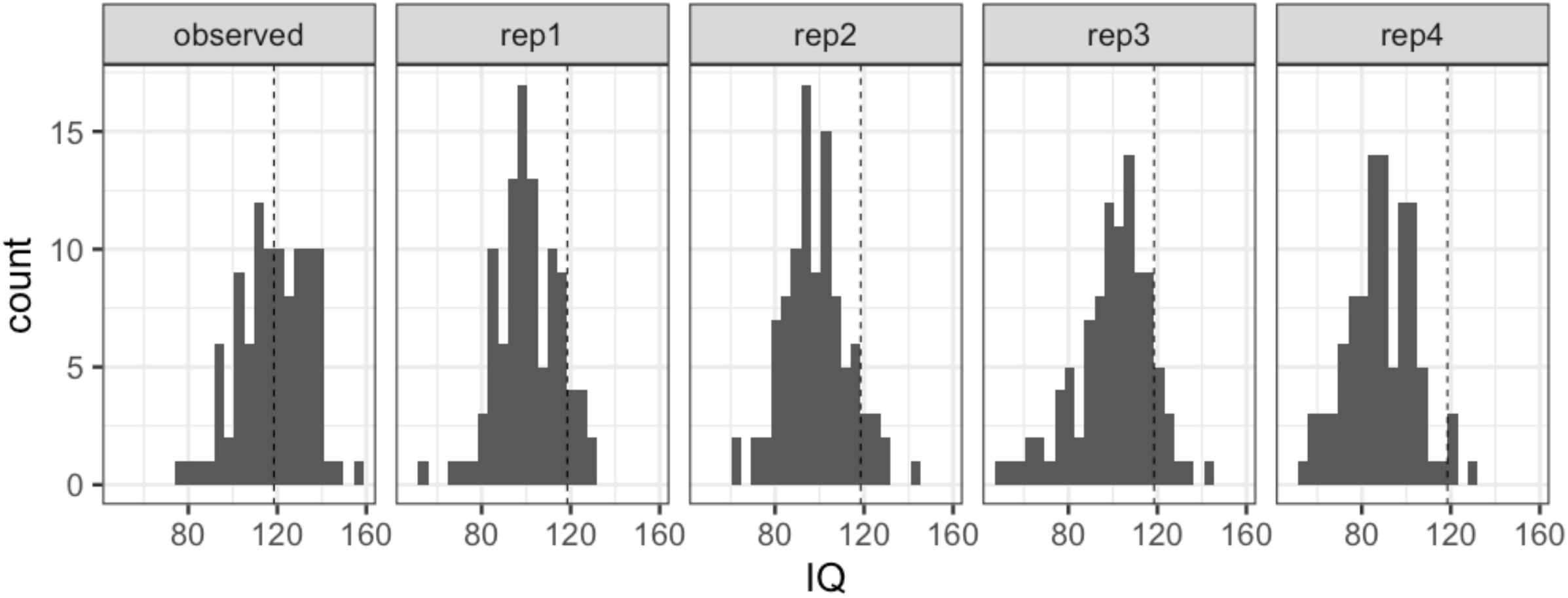
Posterior predictive checks



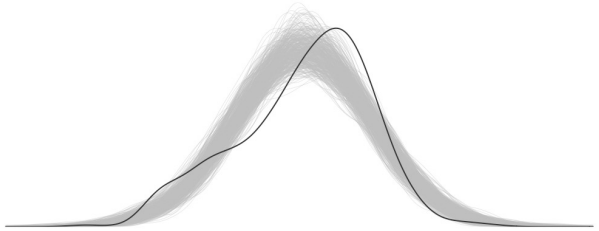
Suppose we have 100 replicated data sets from the posterior predictive distribution. How do we compare them to the observed data?

- Graphical comparisons

Graphical posterior predictive checks



Posterior predictive checks



Suppose we have 100 replicated data sets from the posterior predictive distribution. How do we compare them to the observed data?

- Graphical comparisons
- Numerical comparisons

General: convenient to define a *test statistic* or *discrepancy measure*

Test statistics

- Capture the aspects of the data we want to check
- Problem specific
- Some software offers general test statistics, e.g., likelihood ratio test statistic for SEM
- Examples: mean, standard deviation, distributional asymmetry, autocorrelation, etc.. (see BDA Ch6 for examples).

Posterior predictive p-values (ppp)

- We can directly compare the test statistic of the observed and replicated data sets, or compute a posterior predictive p-value.
- Provides a general summary of the lack of fit
- Interpretation: we want a ppp around 0.50, extreme values indicate a lack of fit

Important caveats

- We are not trying to reject or accept a model, so not concerned with type 1 error rates
- Ppp’s are not necessarily uniformly distributed

An example: Predicting math performance

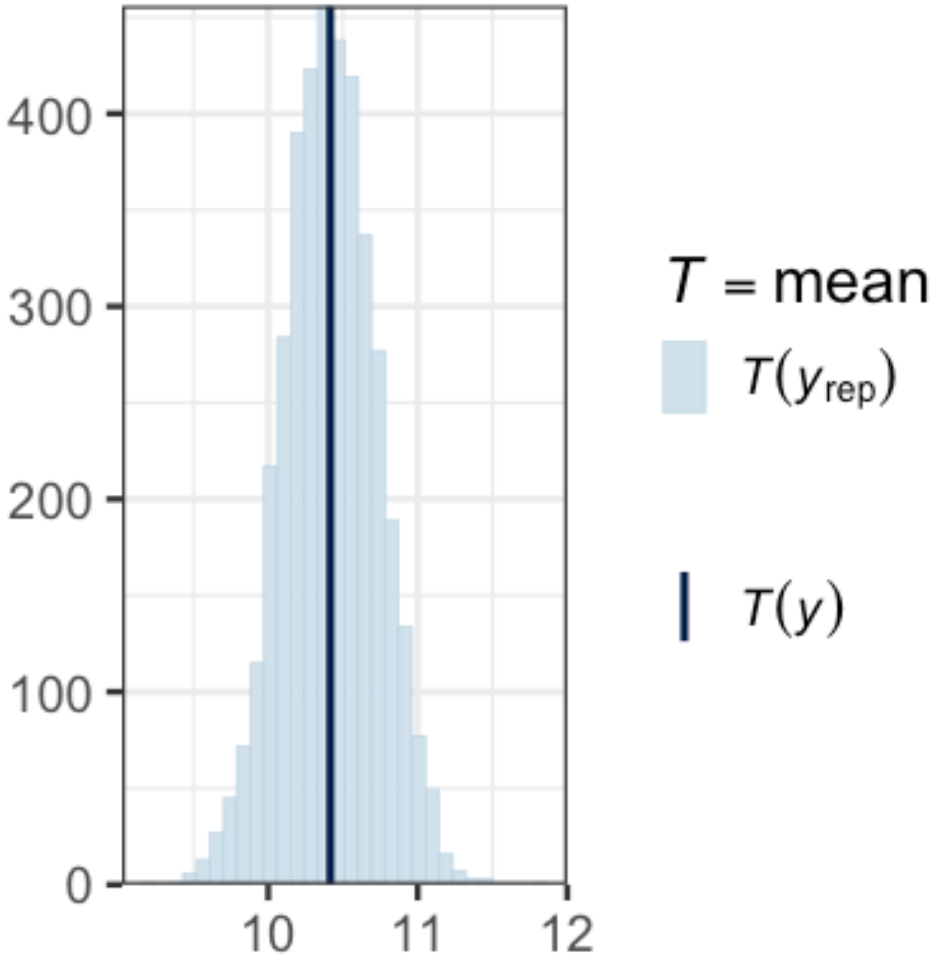
See the Markdown file (https://utrechtuniversity.github.io/BayesianEstimation/content/wednesday/convergence_checks.html)

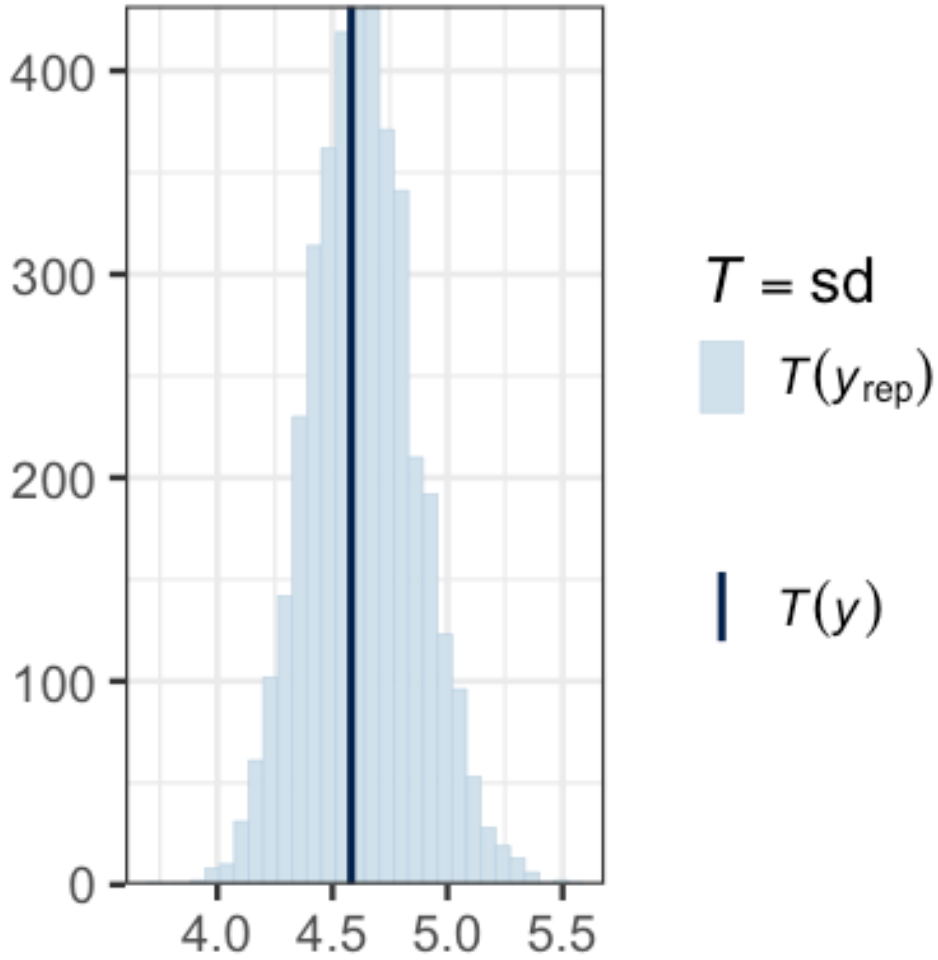
We will use linear regression to predict the math grade of 395 Portugese students in secondary school.

Outcome: Math grade at third period (0-20)

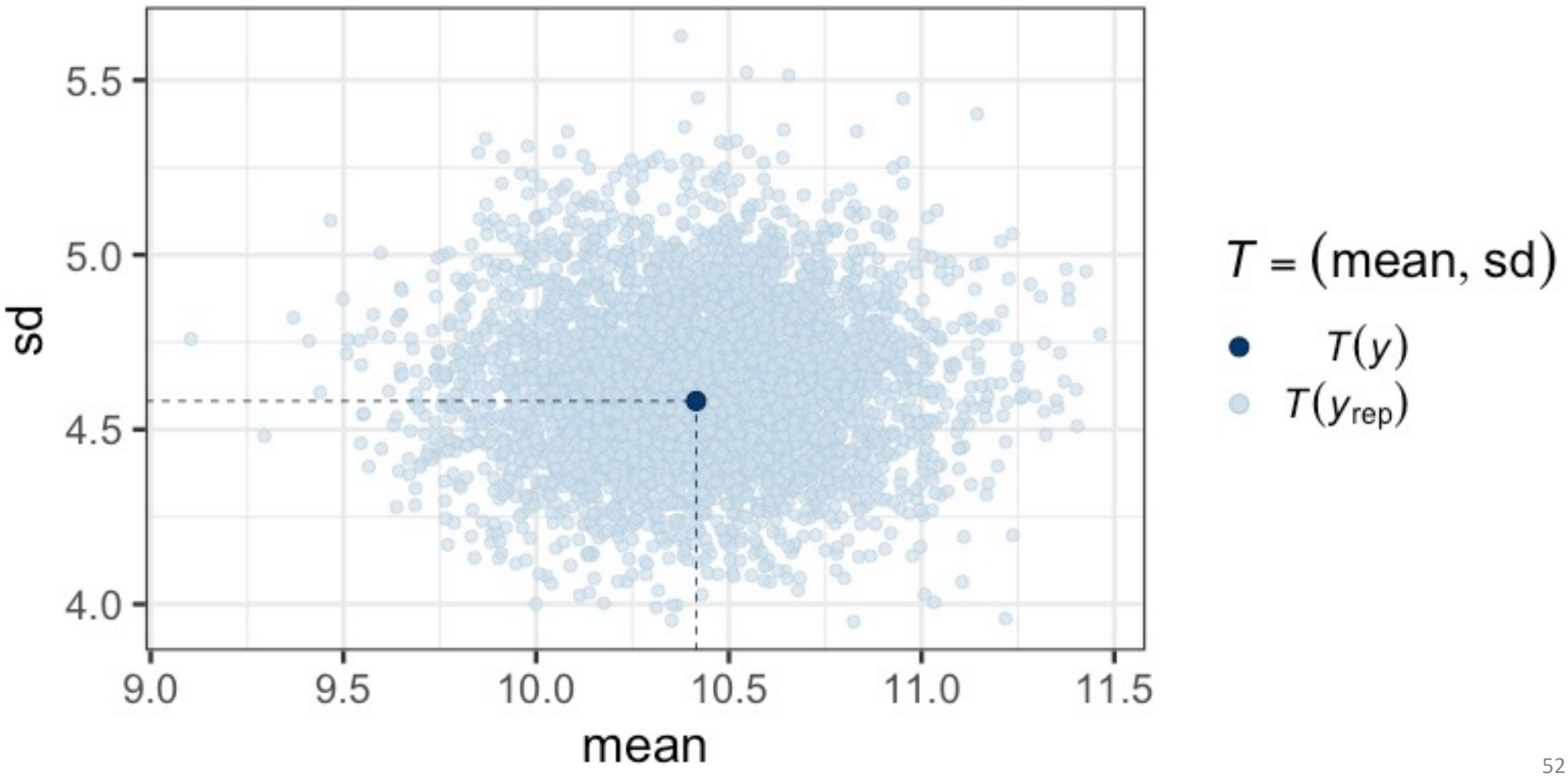
Predictors: sex, weekly time spent studying, additional math class, whether the student wants to take higher education.

Basic posterior predictive checks

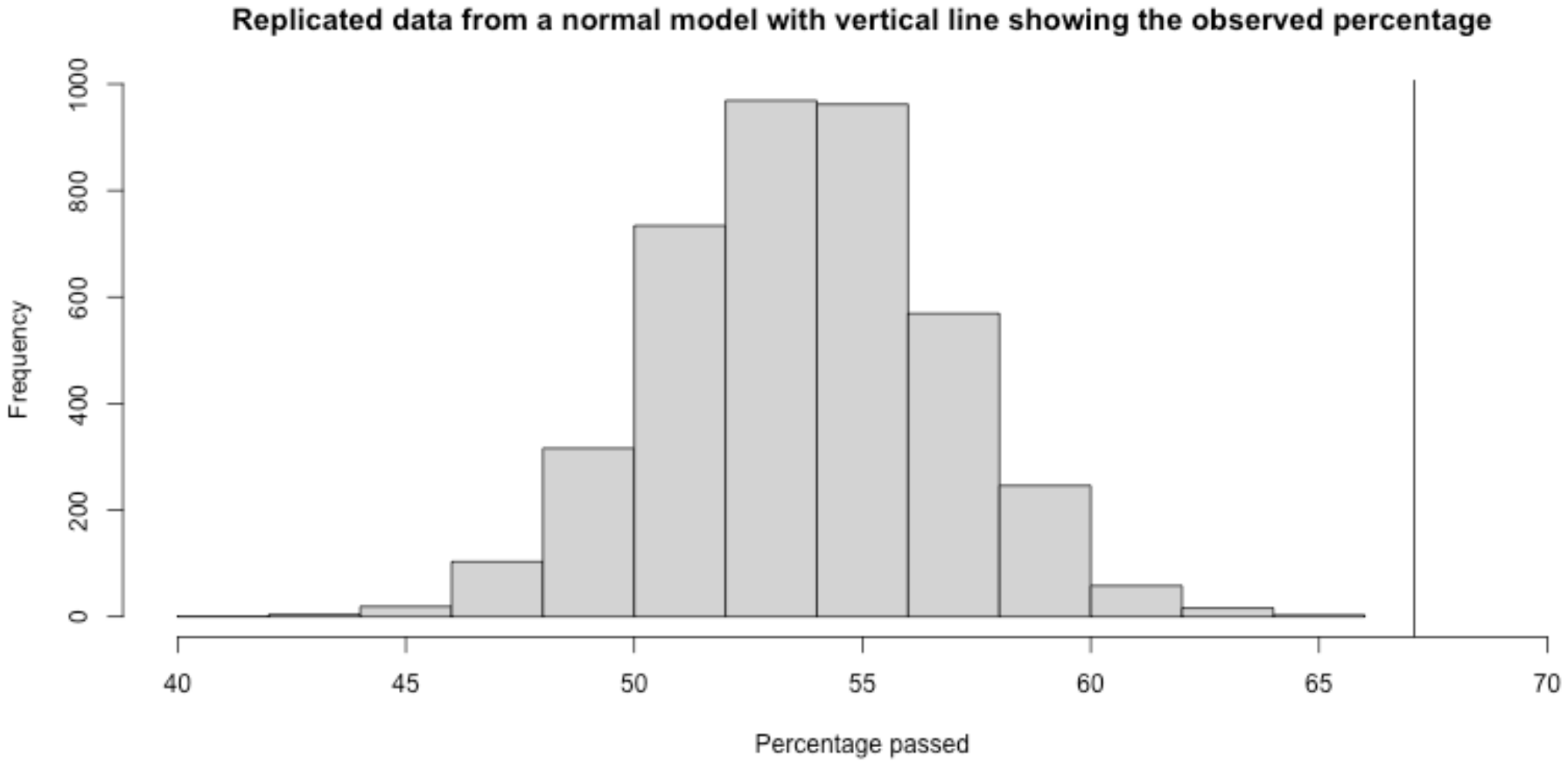




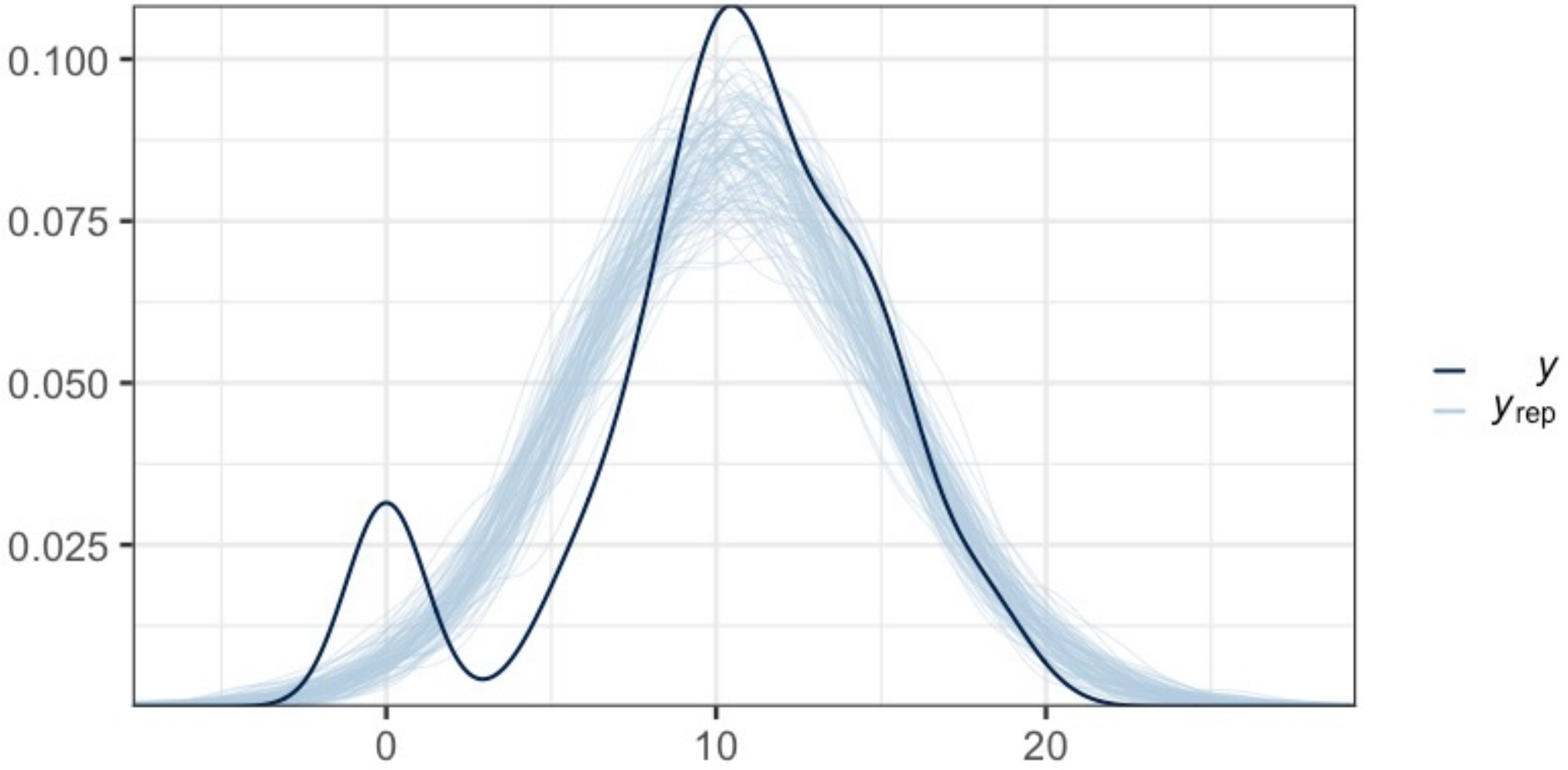
Basic posterior predictive checks



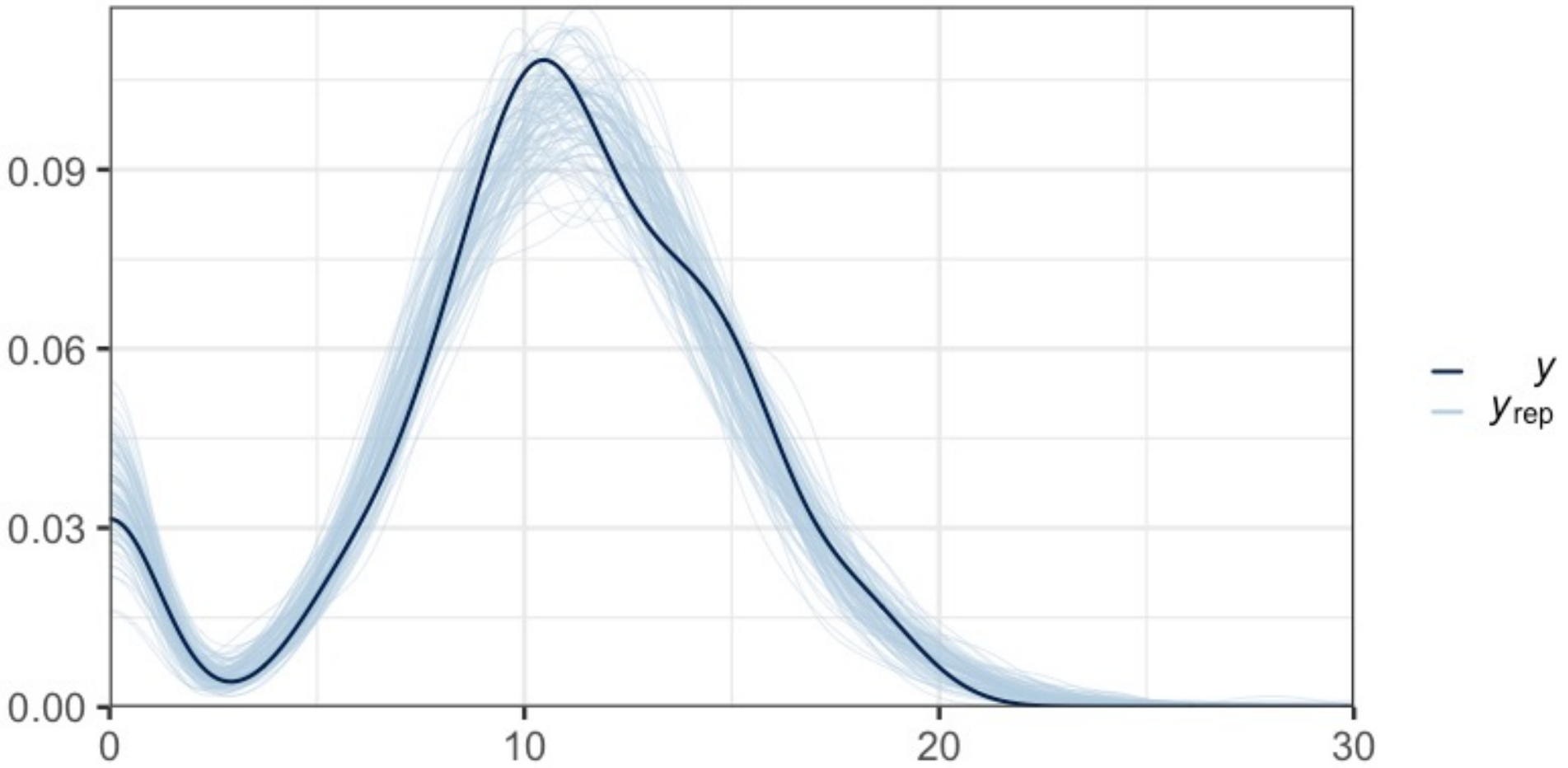
Custom posterior predictive checks



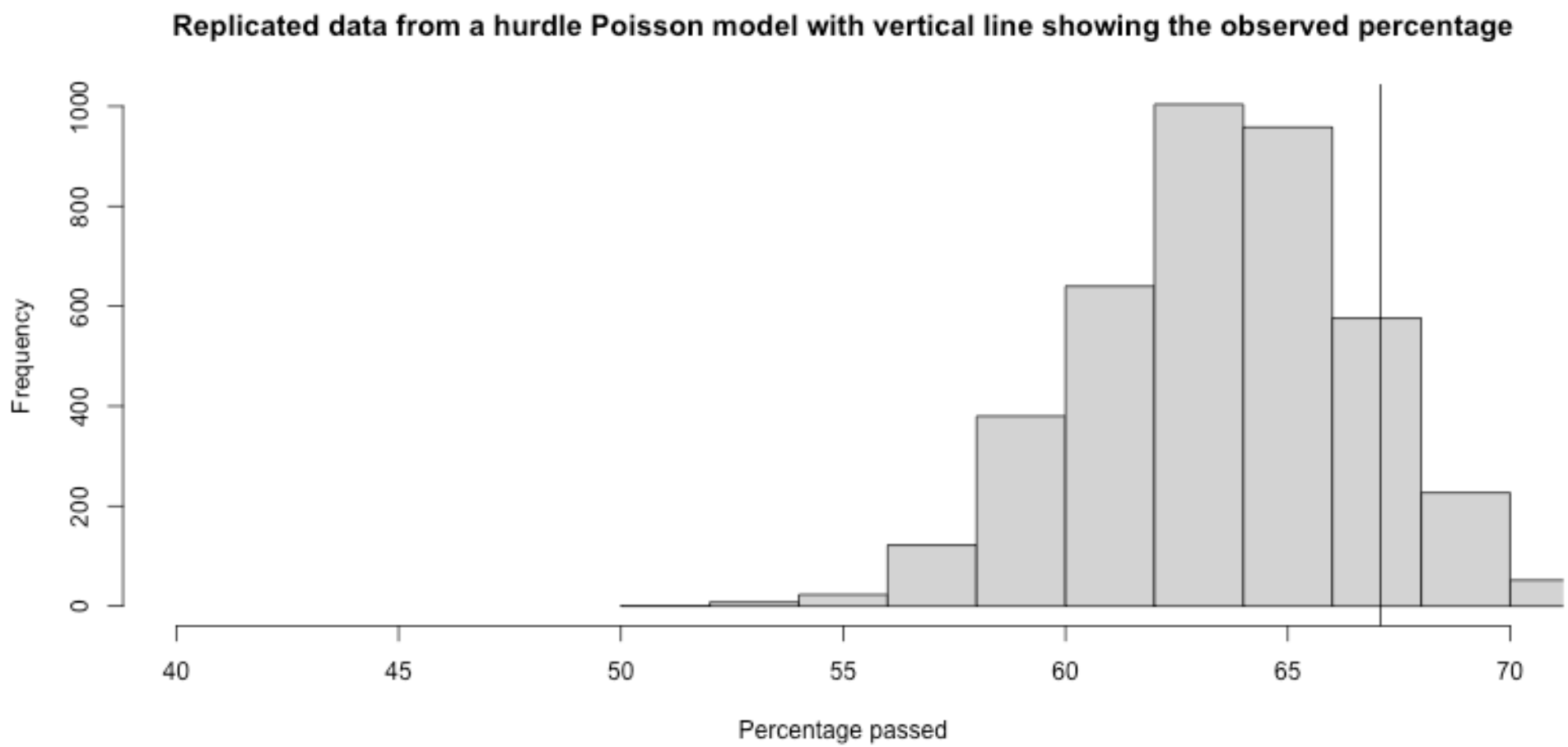
Custom posterior predictive checks



Improving the model: hurdle Poisson



Improving the model: hurdle Poisson



Recommendation: Use posterior predictive checks!

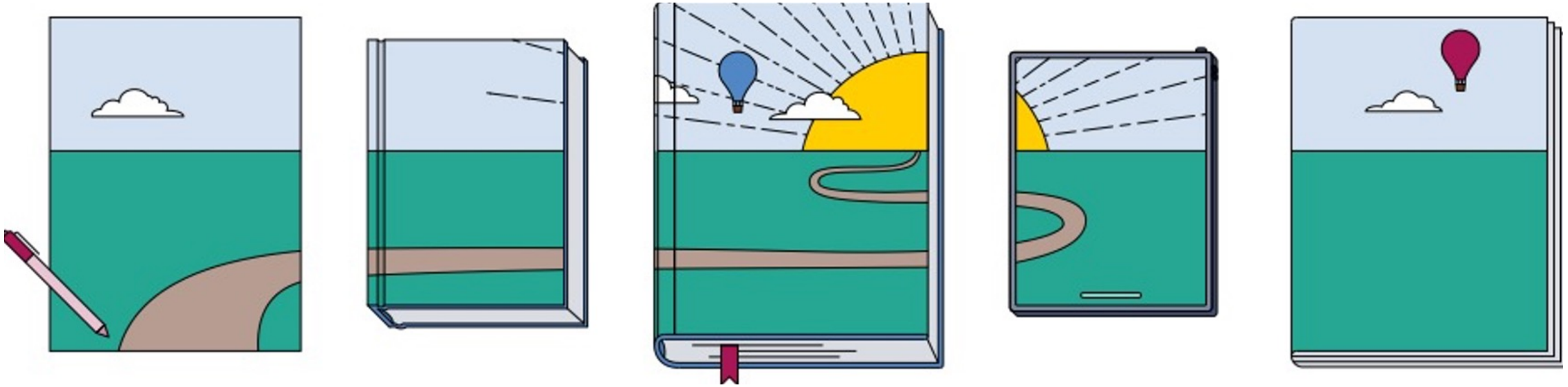
- Posterior predictive checks can provide useful visual and numerical diagnostics of model fit
- Standard posterior predictive checks are available
- Custom posterior predictive checks might be more suitable
- Consider carefully which aspects your model should represent well
- Posterior predictive checks can be done using the training data, or a test set (see also today's practical)

Prior predictive checks

The same idea can be used to see if our priors make sense.

Generate data from the *prior* predictive distribution.

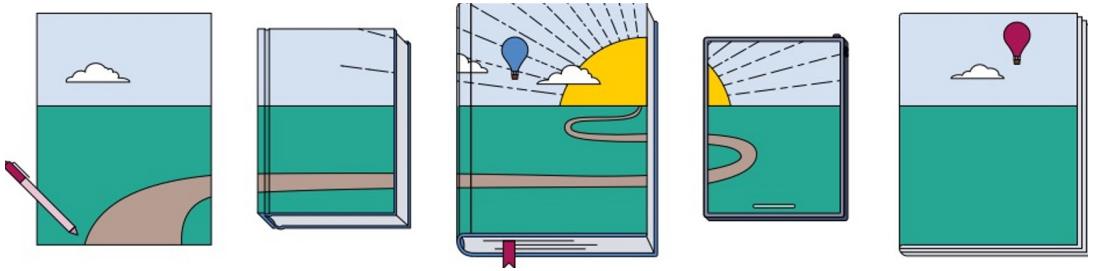
If priors lead to generated data that makes no sense, you might want to revisit them.

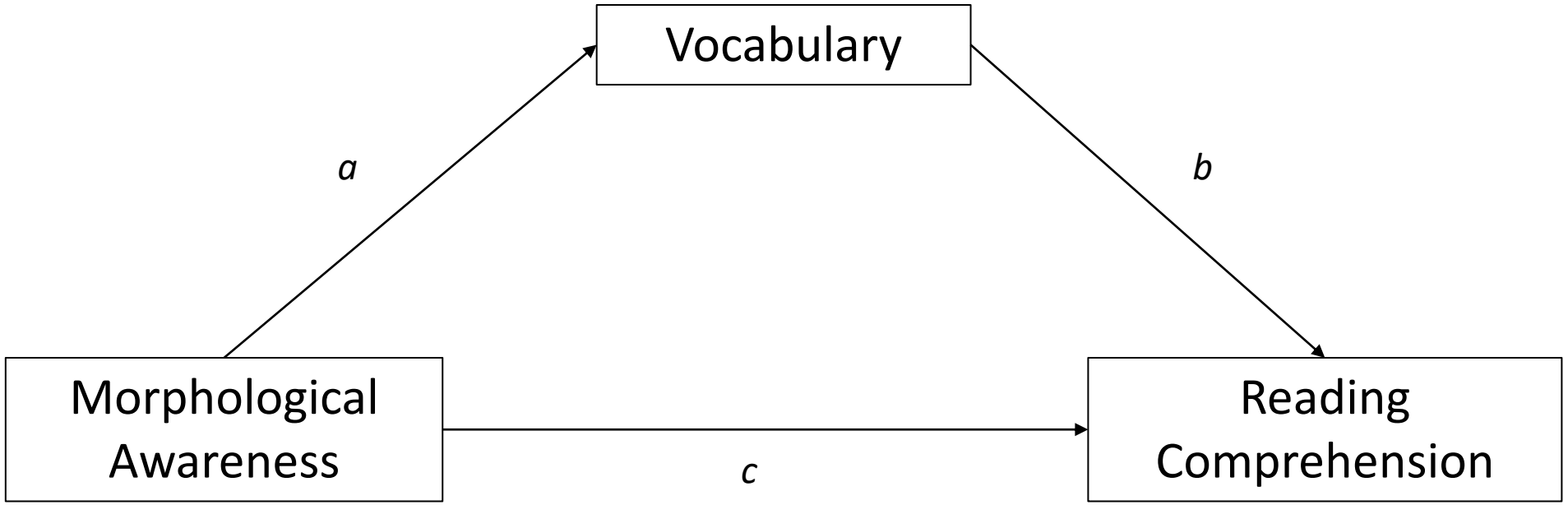


An example

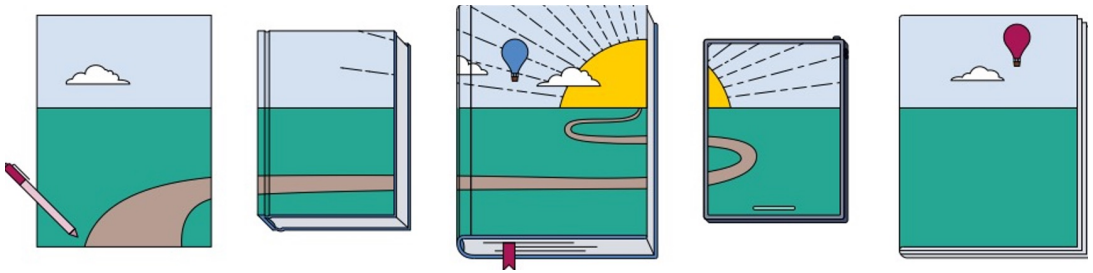
- The *Road to Resilience* project aims to unveil the mechanisms behind resilient trajectories in literacy acquisition.
- Small samples are expected, so Bayesian analysis is being used.
- Data: children in grade 5, 6, and 7 of primary school with and without dyslexia + data from a pilot study.

Mediation model



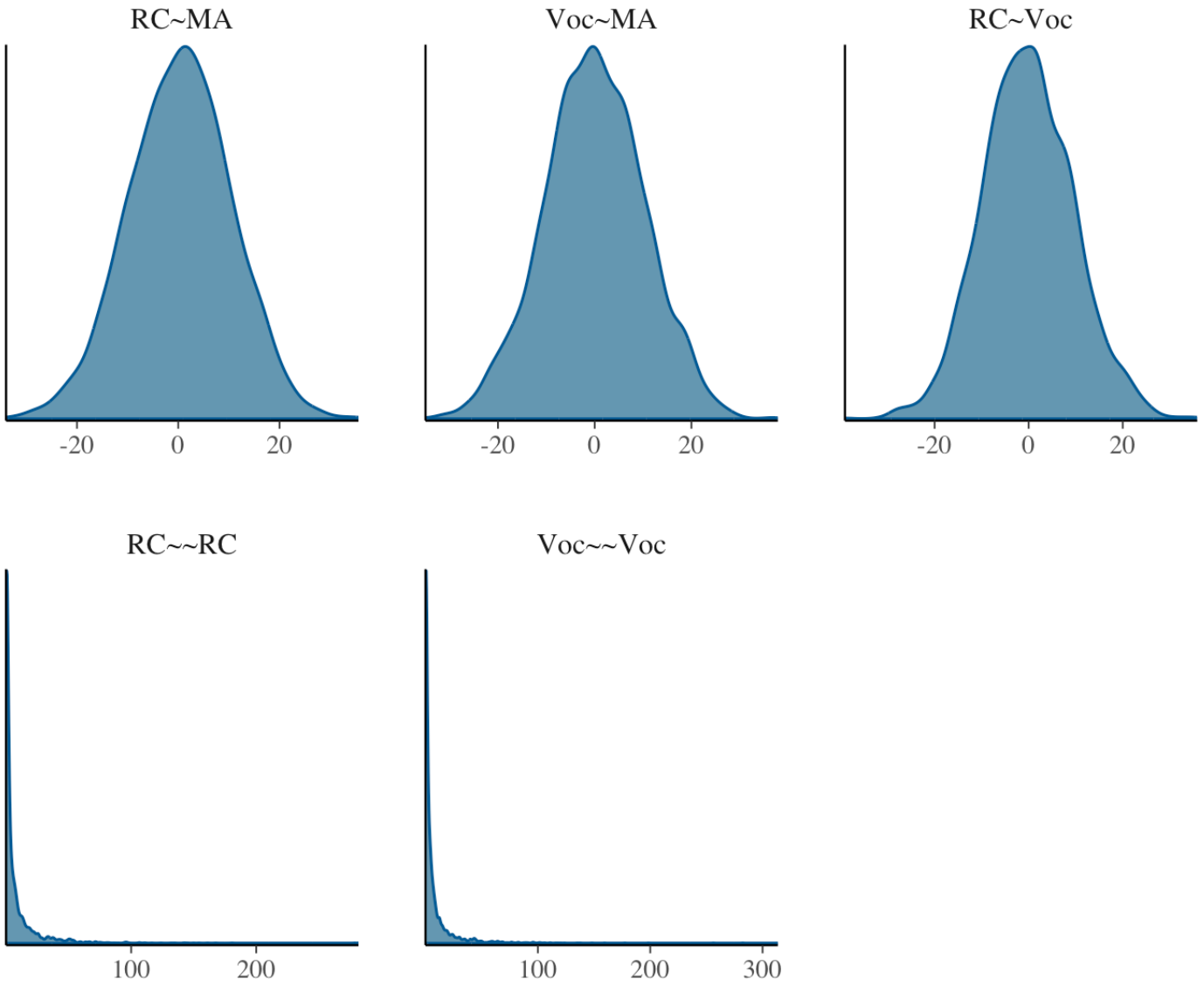


Priors

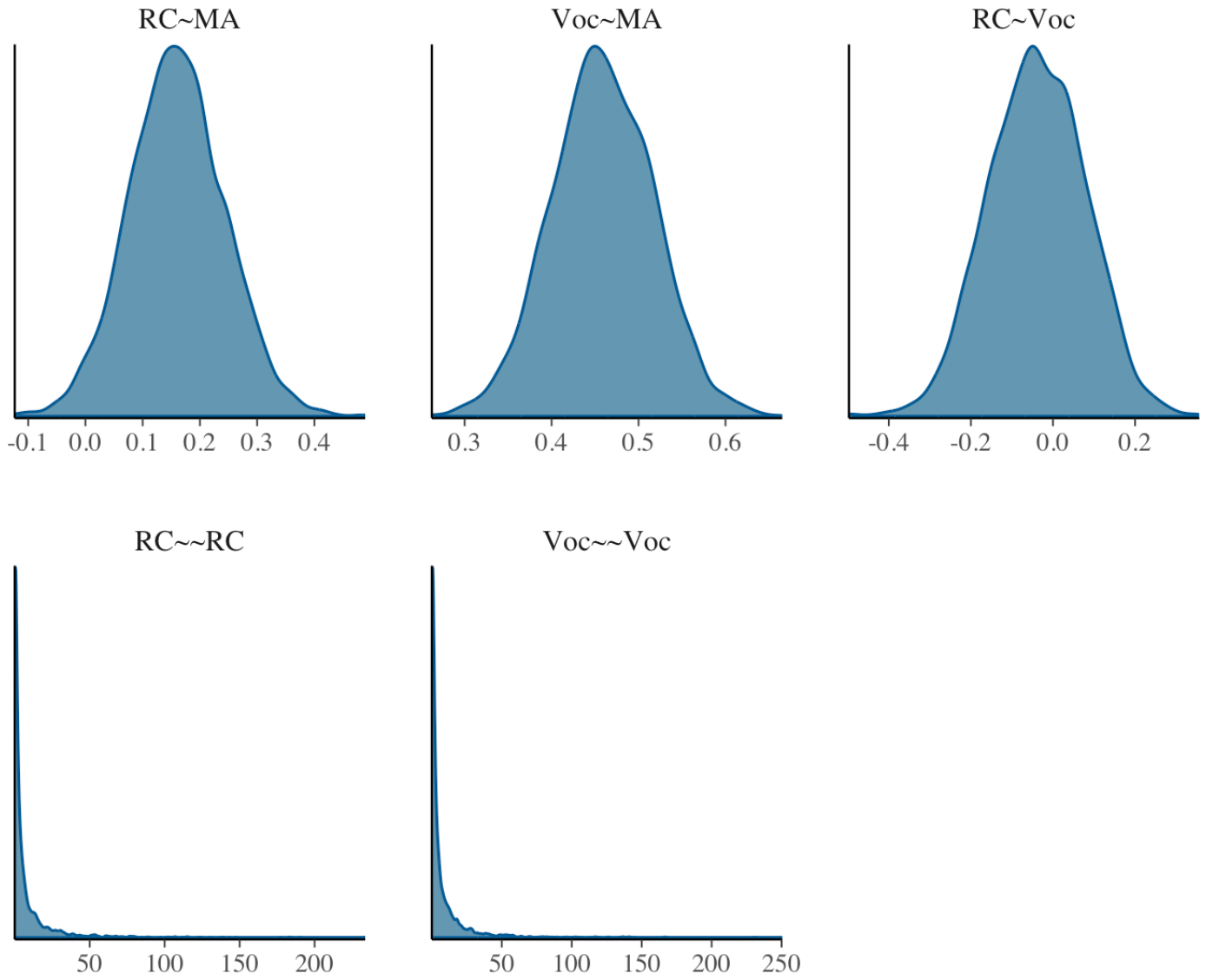


Parameter	Default prior (blavaan)	Informative prior (pilot data)
MA -> Voc *(a)*	N(0, 10)	N(0.46, 0.06)
Voc -> RC *(b)*	N(0, 10)	N(-0.04,0.12)
MA -> RC *(c)*	N(0, 10)	N(0.16, 0.08)
σ_V	G(1, 0.5)	G(1, 0.5)
σ_{RC}	G(1, 0.5)	G(1, 0.5)

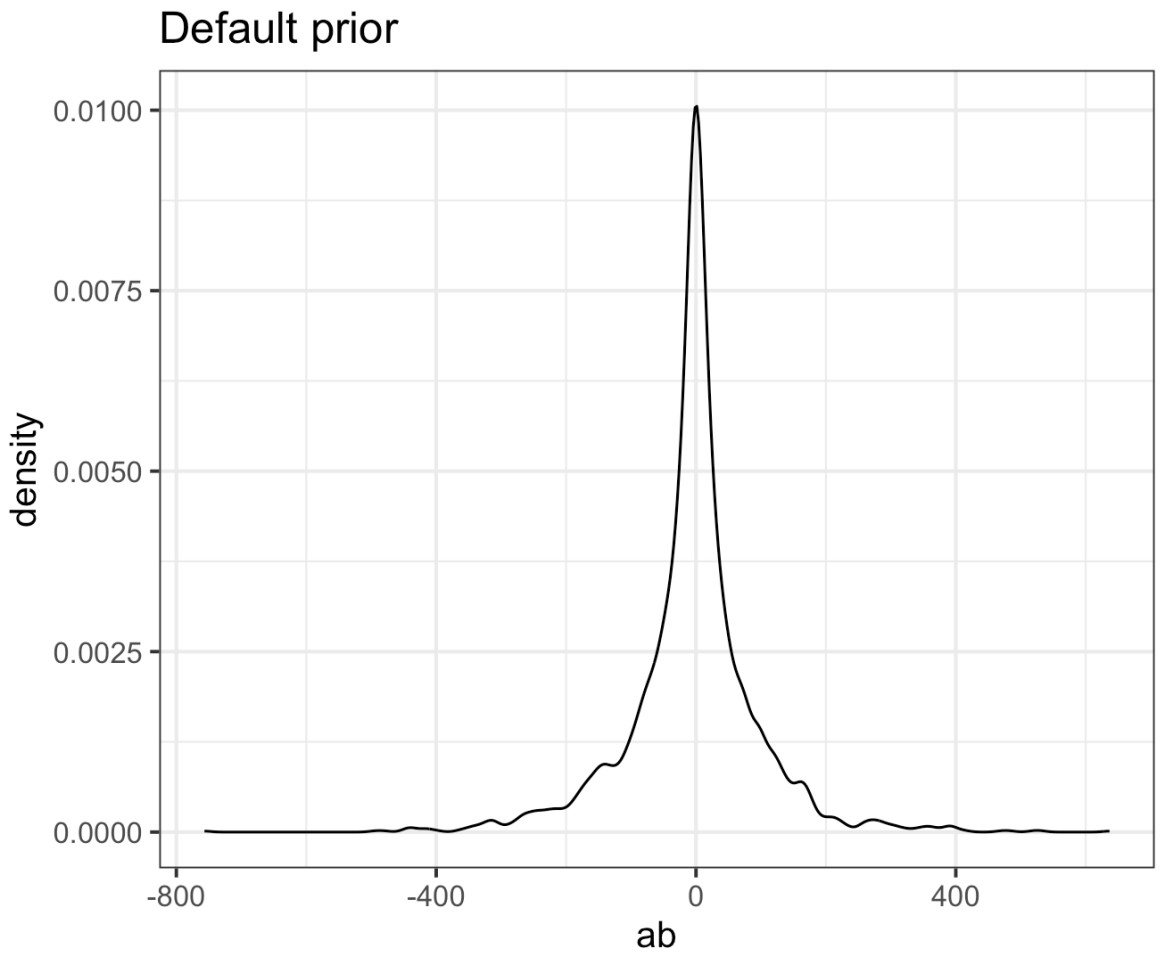
Visualizing the priors (default)

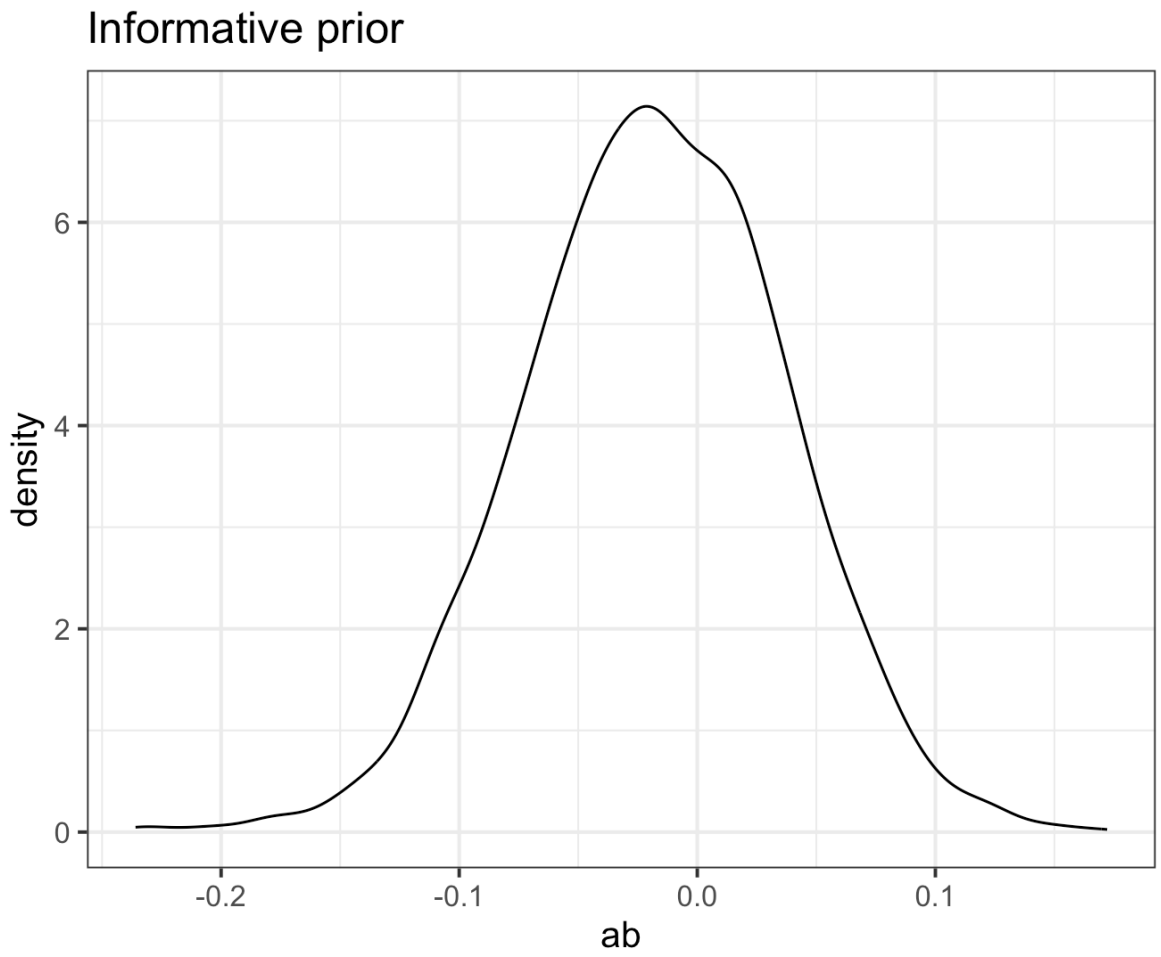


Visualizing the priors (informative)

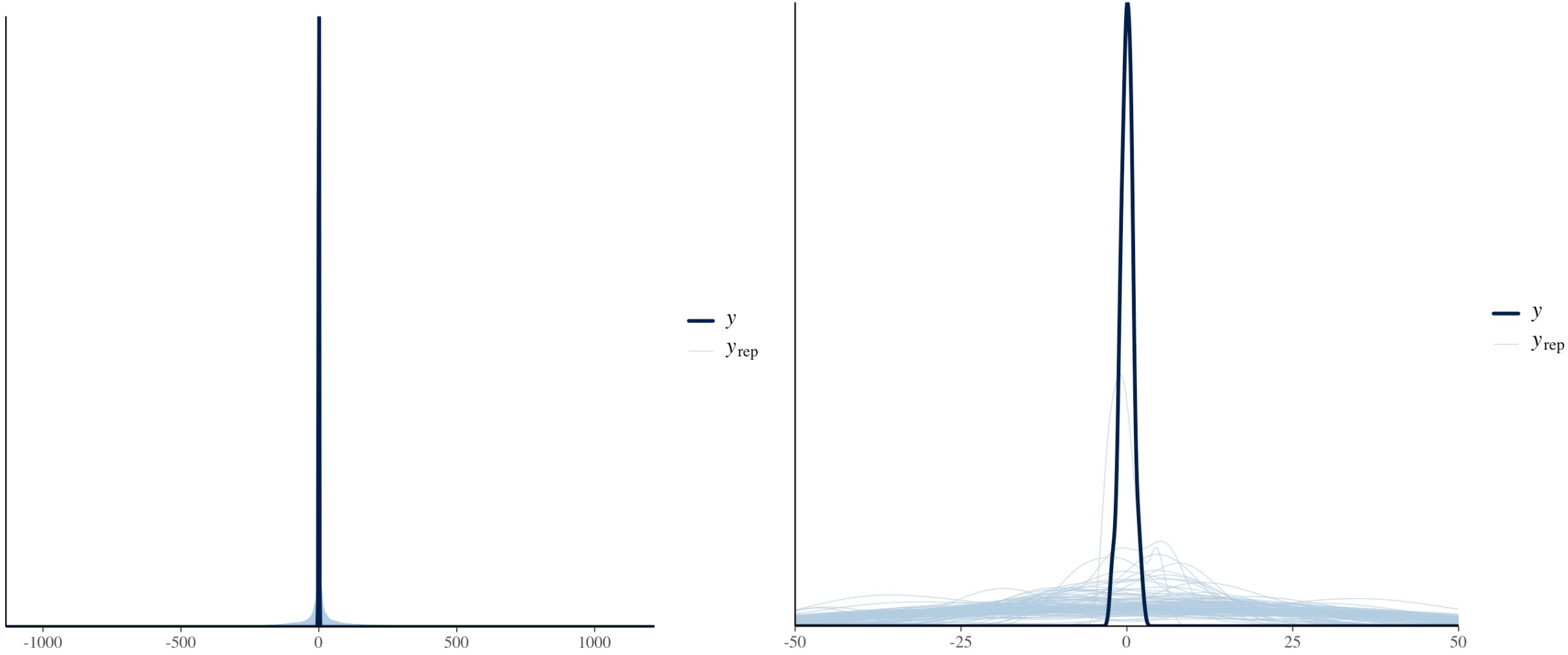


Implied prior indirect effect *ab*



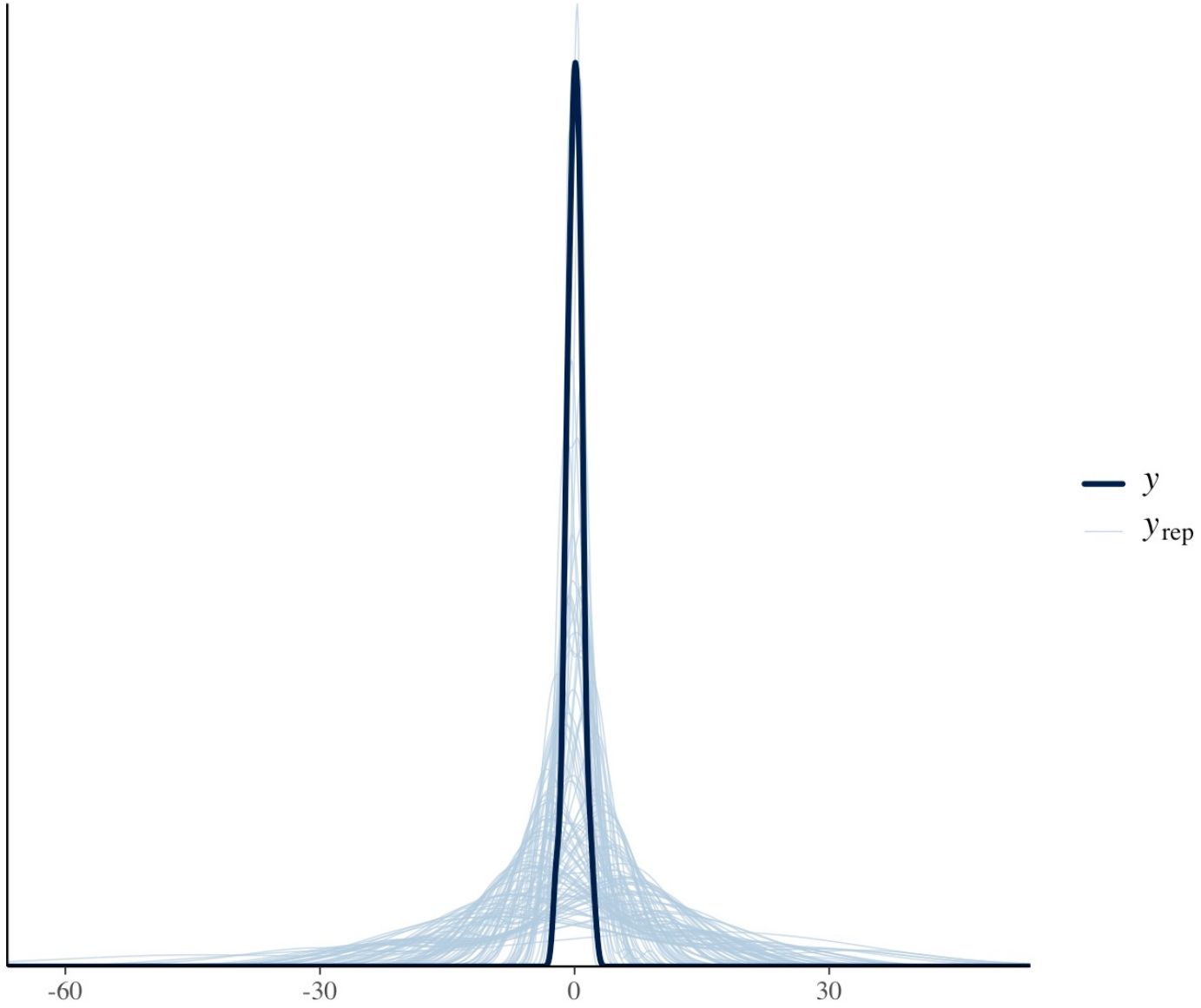


Visualizing RC scores generated based on the default priors

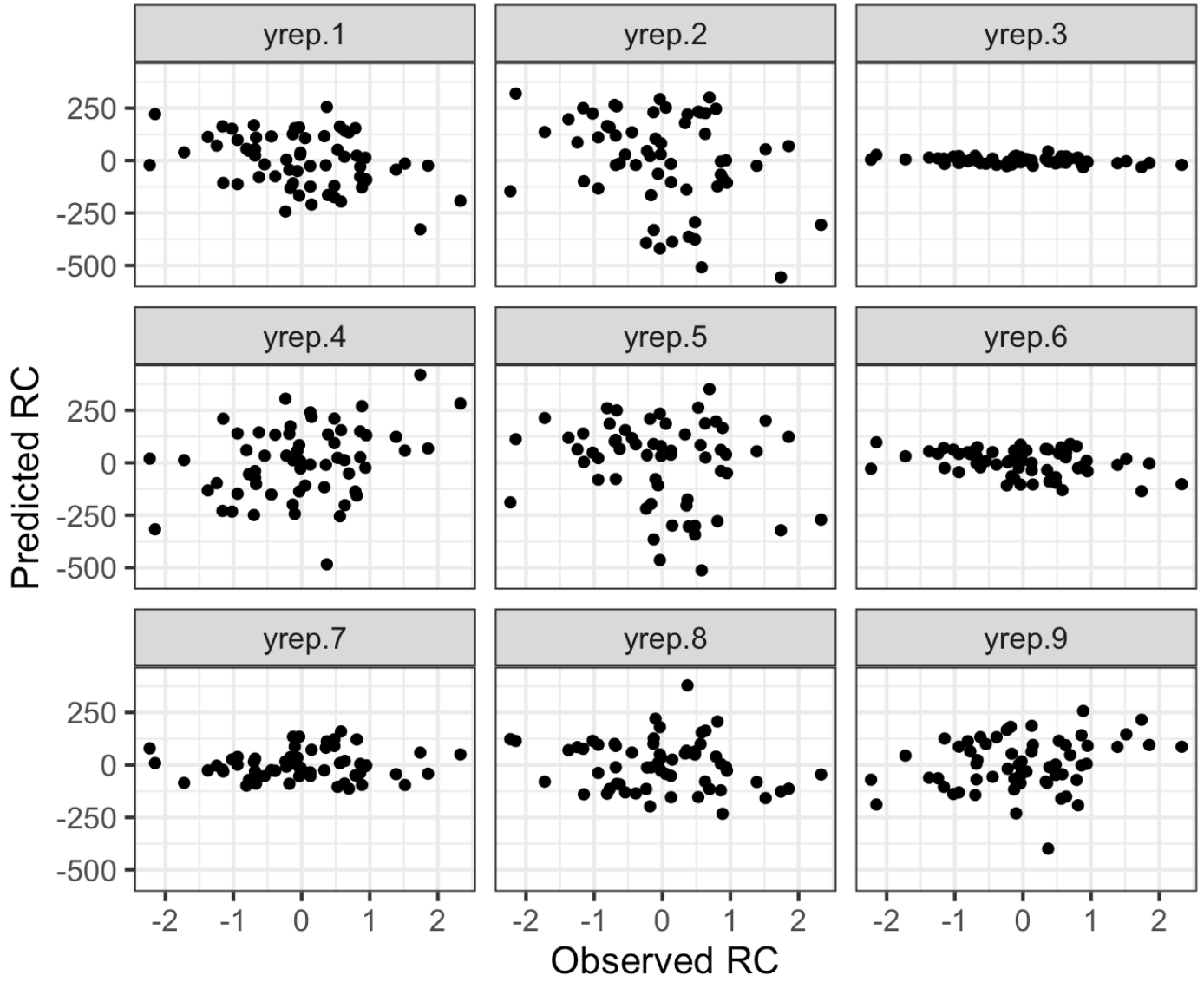


Visualizing RC scores generated based on the informative priors

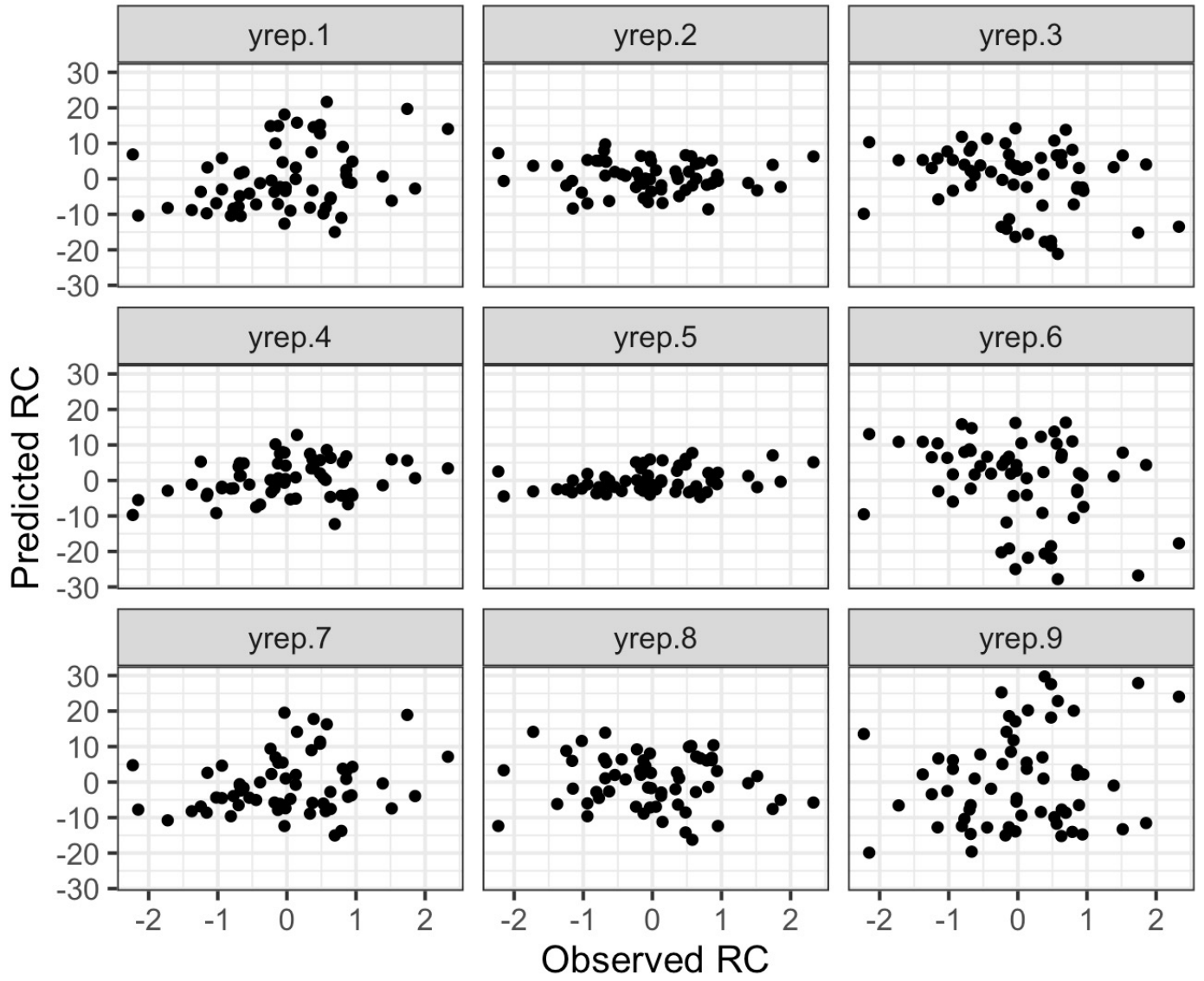
Note the x-axis scale: informative priors lead to less extreme data sets



Examples generated data sets default priors



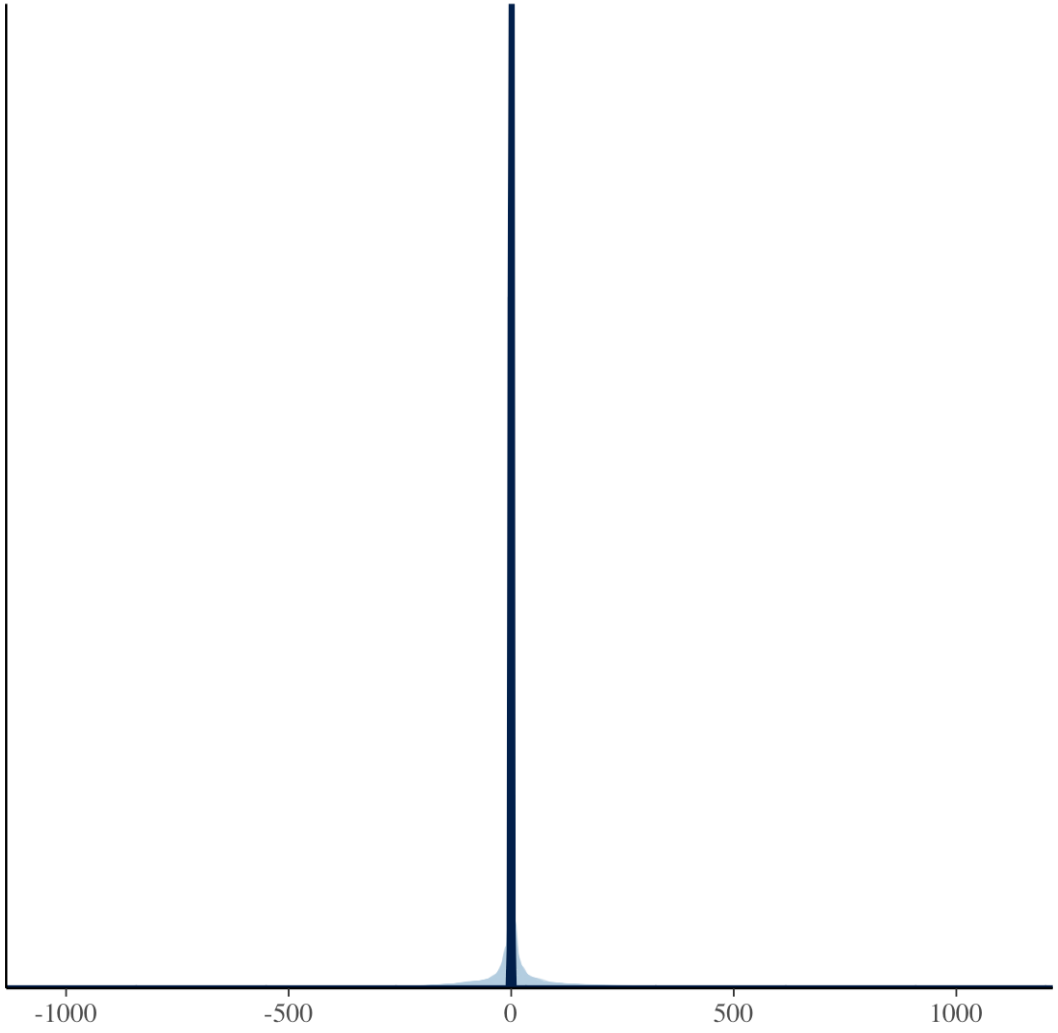
Examples generated data sets informative priors

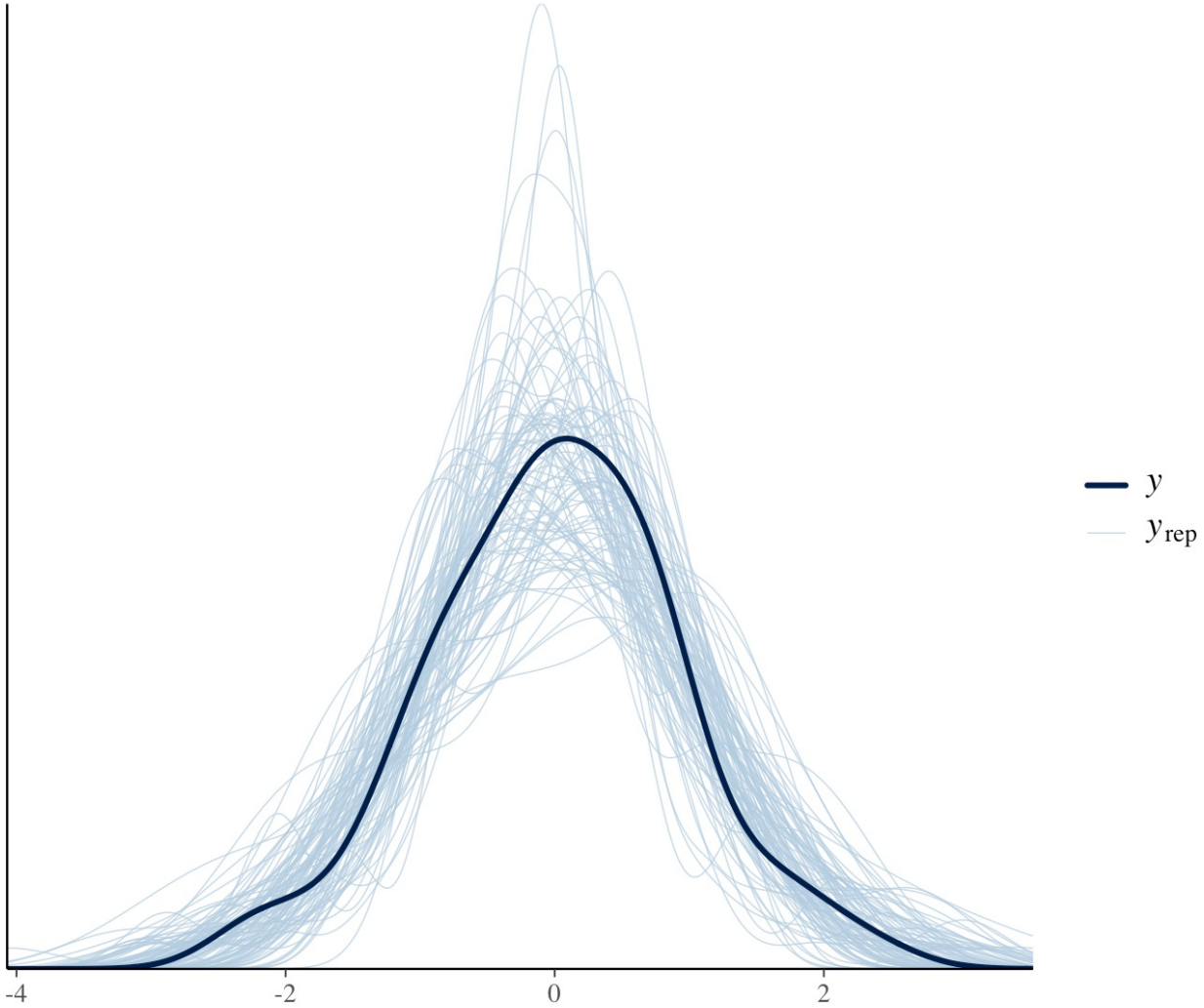


Recommendation: Use prior predictive checks!

- Prior predictive checks help you understand the implications of the prior on possible observations
  - Make you consider which data values are plausible, which might be easier to consider than which parameter values are plausible
- Prior predictive checks are, like the prior, highly context-dependent
- Depend on the chosen prior as well as the model
- Can be compared to the observed data but be careful if you are still in the prior specification stage!

From prior to posterior predictive check





Recap

Part 1: Software and algorithms

- Different ways to get the posterior
- What is going on (conceptually) under the hood?
- What should you, as user, be aware of?

Part 2: Predictive checks

- Posterior predictive checks: how can we check our model?
- Prior predictive checks: how can we check our priors?


Questions?