

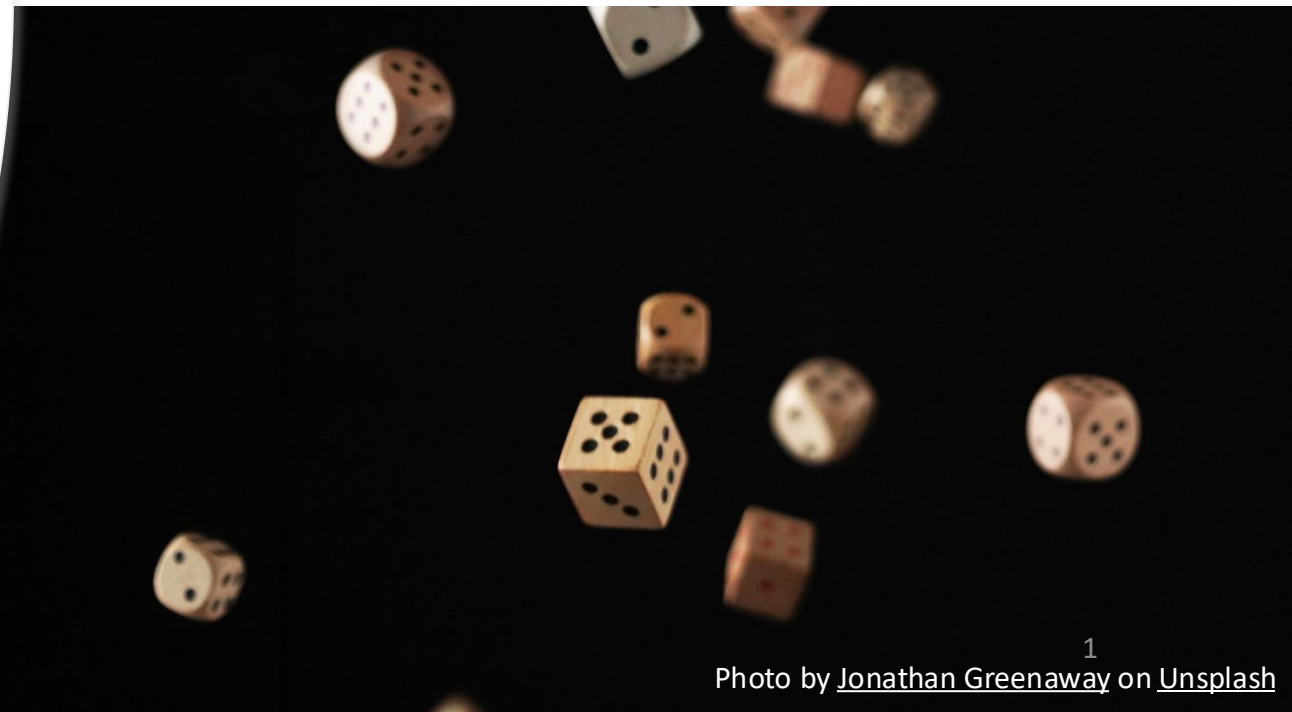
A Gentle Introduction to Bayesian Estimation

Day 3: Algorithms and Checks

Sara van Erp
s.j.vanerp@uu.nl



Photo by [Markus Spiske](#) on [Unsplash](#)



1
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

- 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

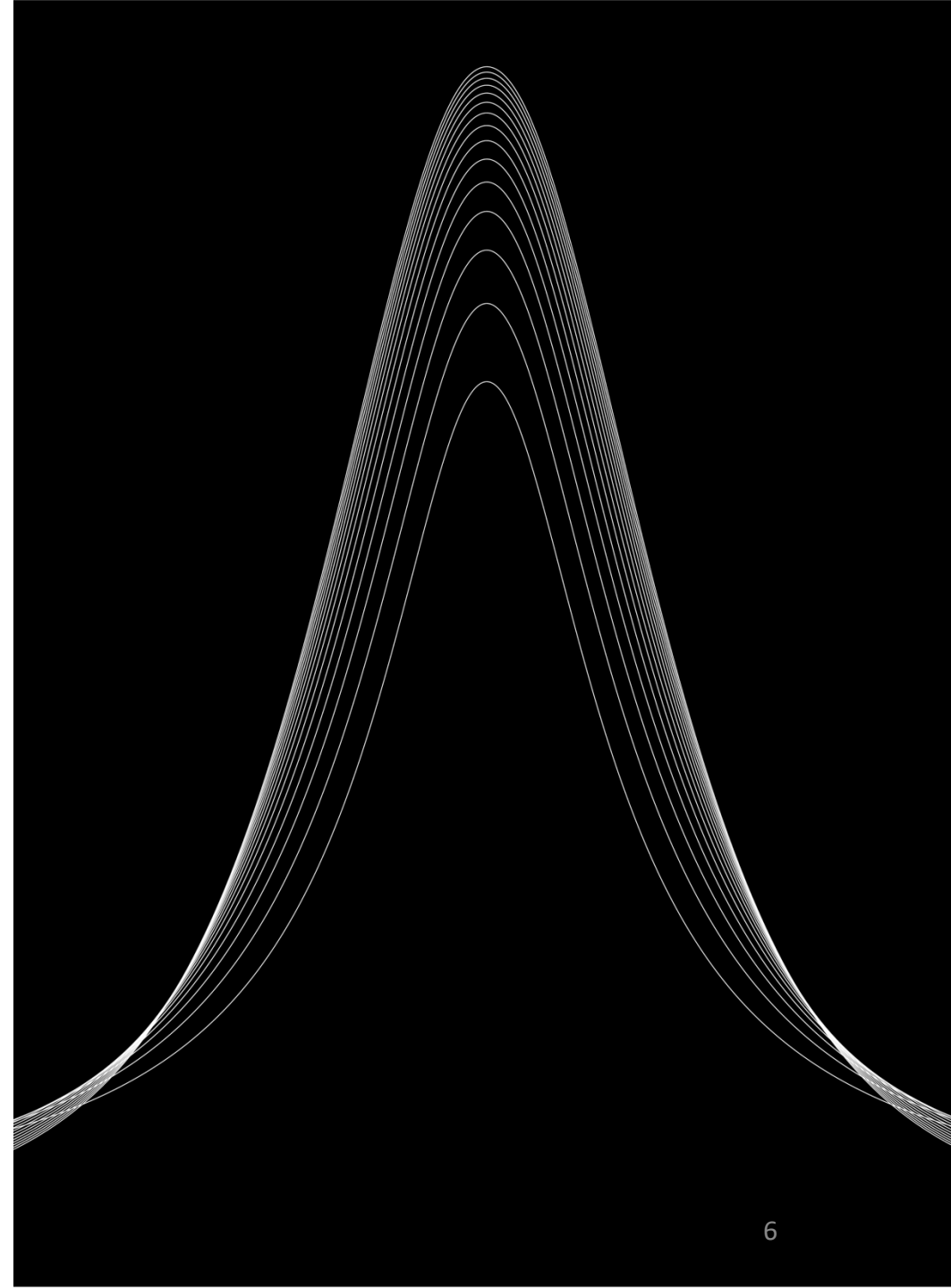
- Posterior predictive checks: how can we check our model?
- Prior predictive checks

A collection of wooden dice in various orientations, appearing to be falling or tossed against a solid black background. The dice are light-colored wood with black pips. Some are in sharp focus, while others are blurred, creating a sense of motion. The dice are scattered across the frame, with some near the top and others near the bottom.

Part 1: Software and algorithms

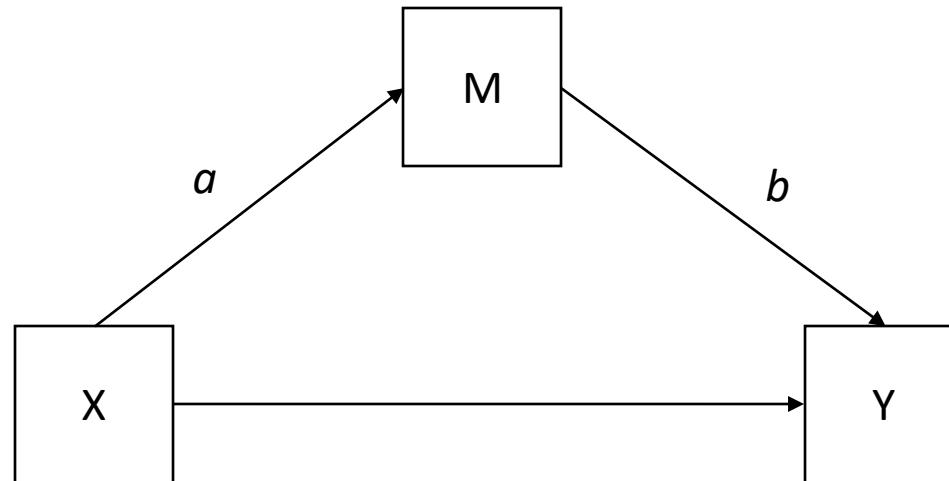
Why use Bayes?

- 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***

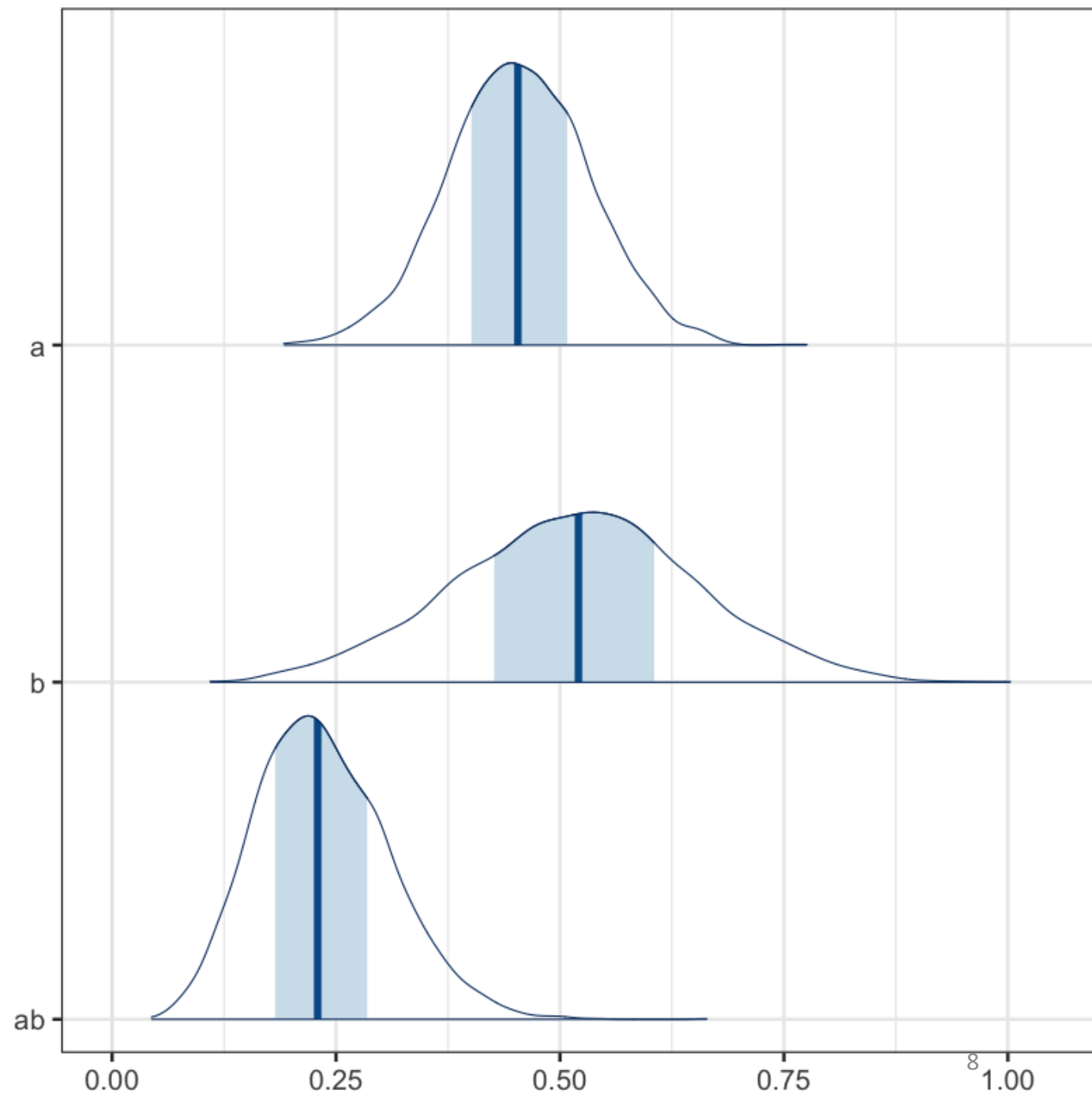
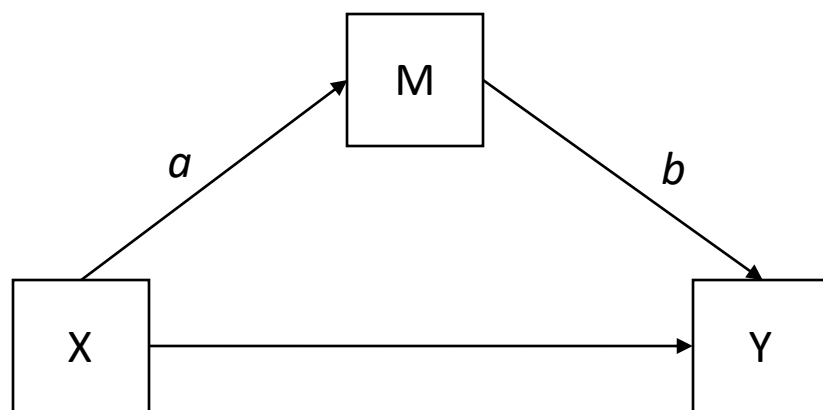


Advantages of the posterior distribution

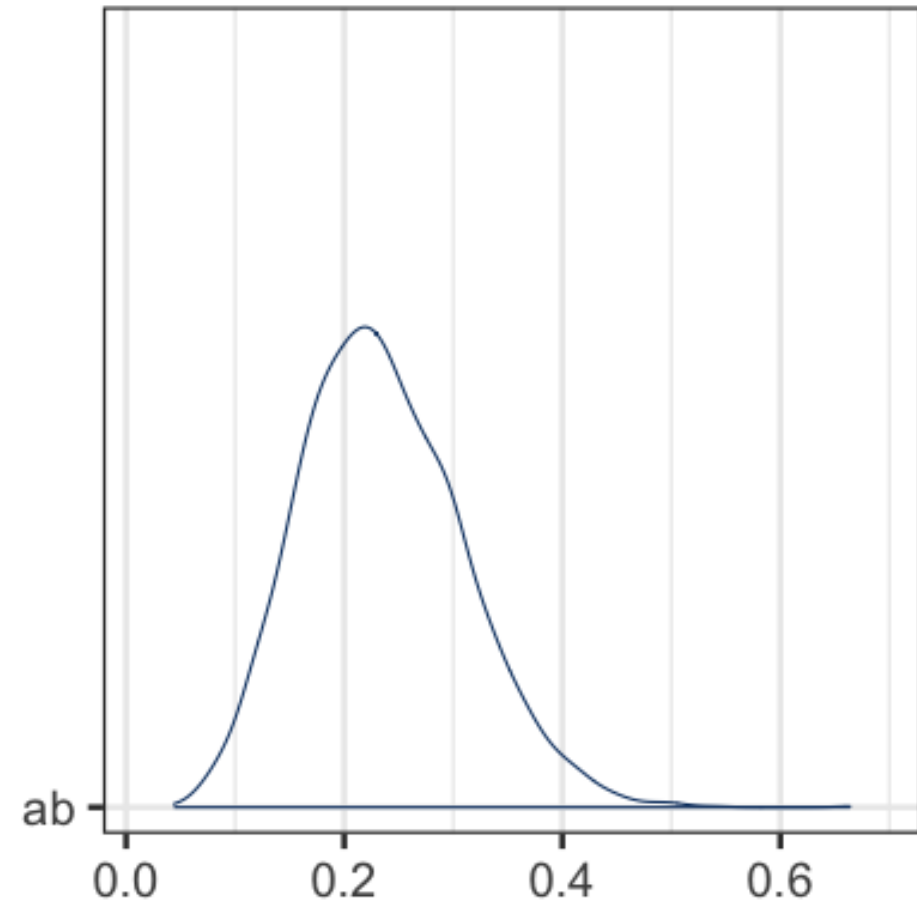
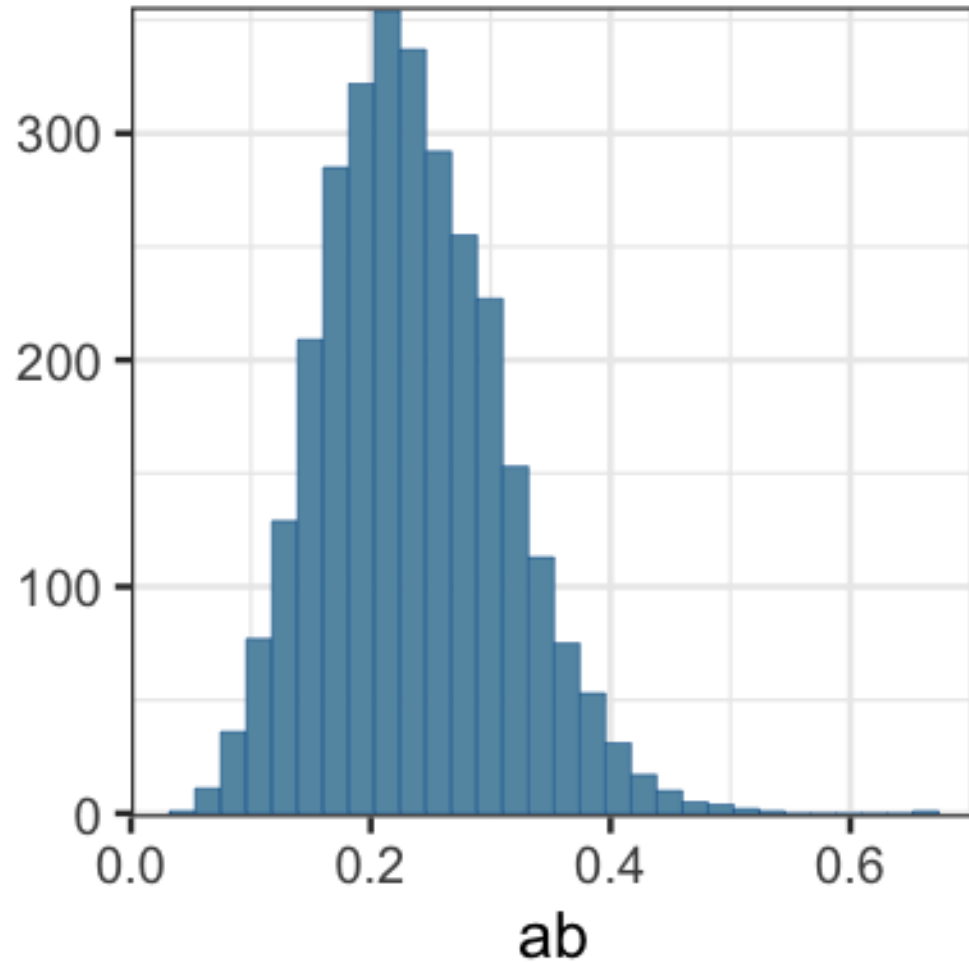
If we want to estimate an indirect effect, we get automatic uncertainty estimates around functions of parameters.



Posterior distribution for
the indirect effect ab

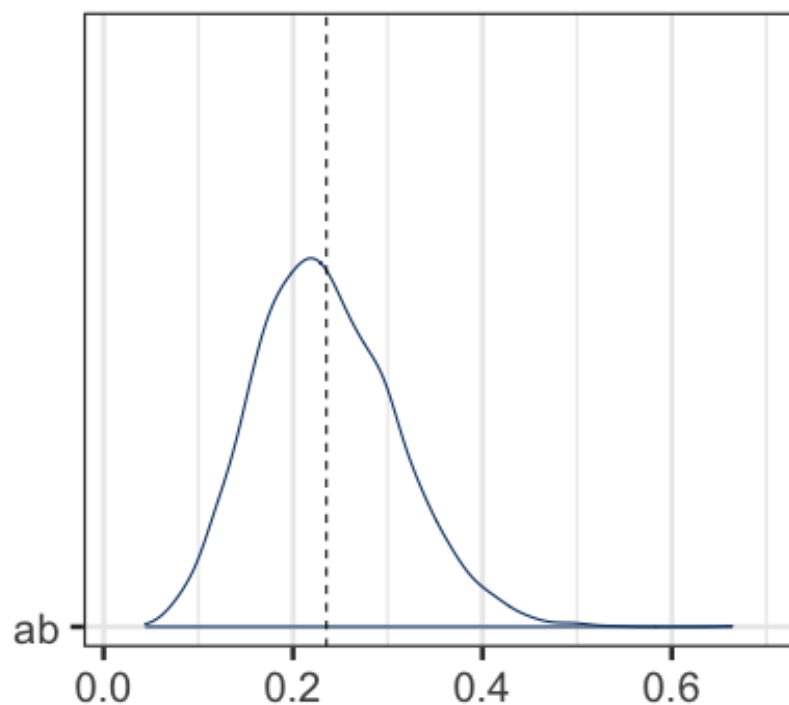


A note on summarizing the posterior

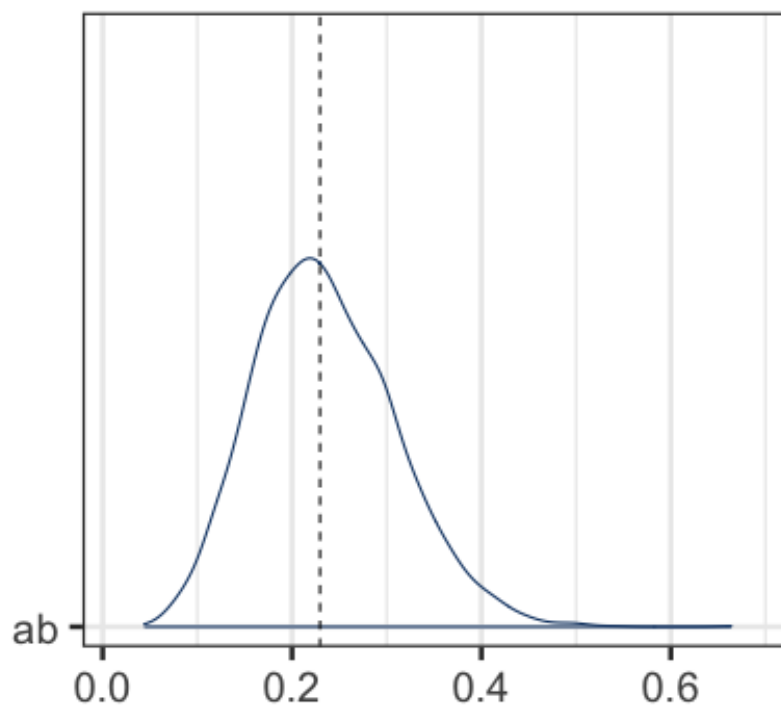


Posterior point estimates

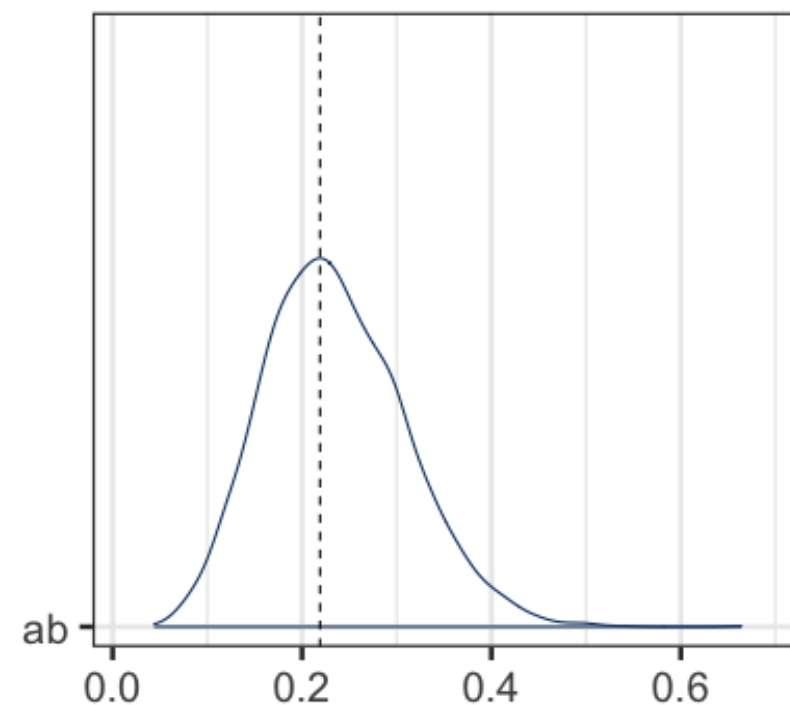
Mean = 0.235



Median = 0.23

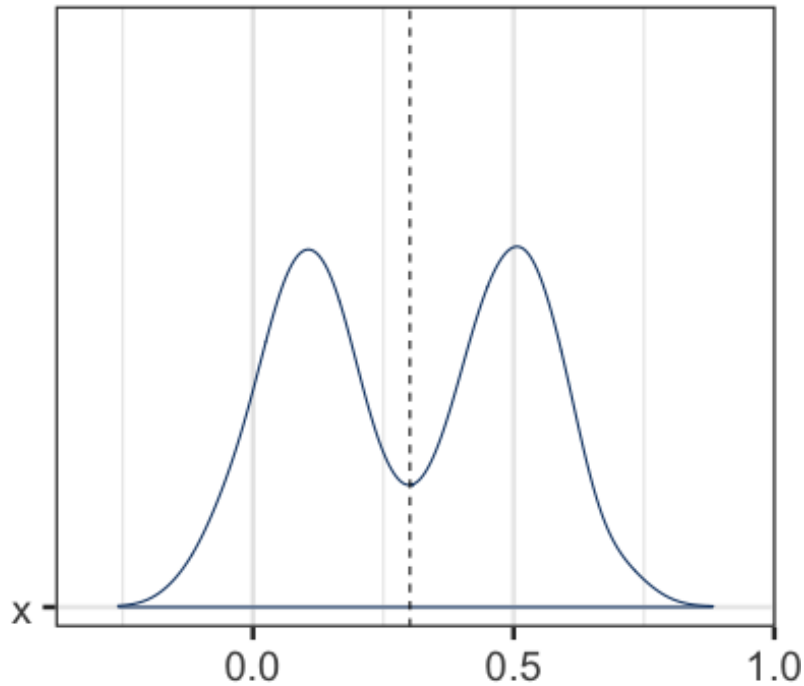


Mode = 0.219

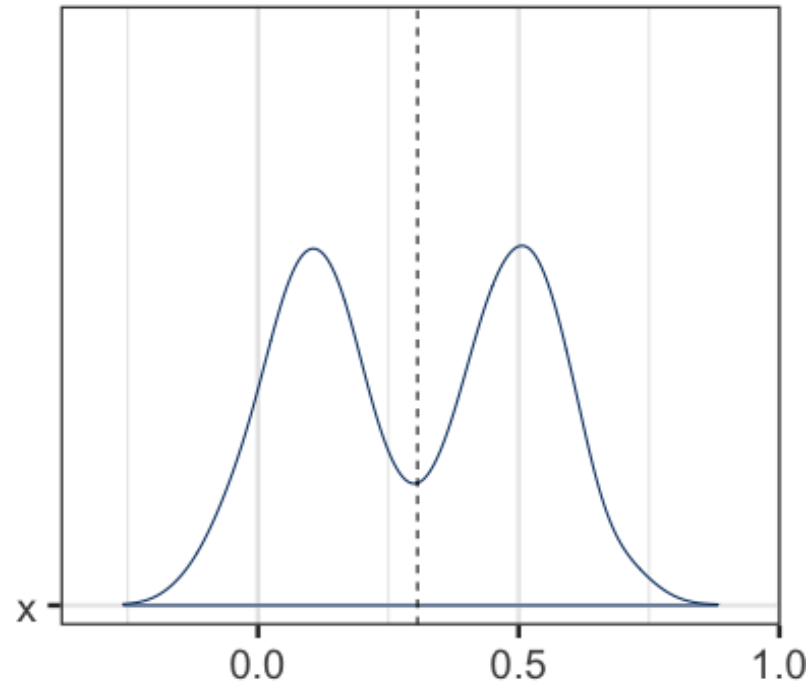


Posterior point estimates

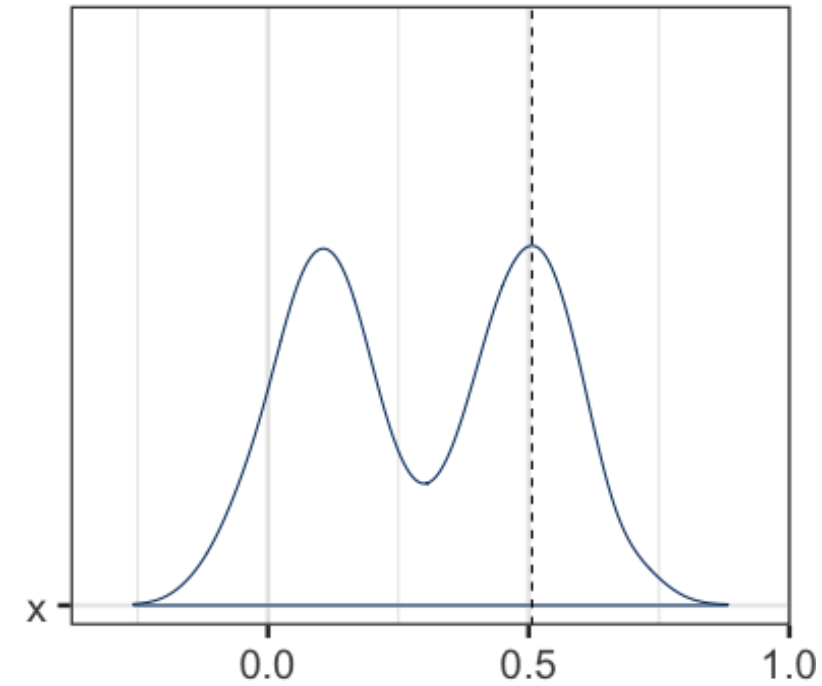
Mean = 0.301



Median = 0.306

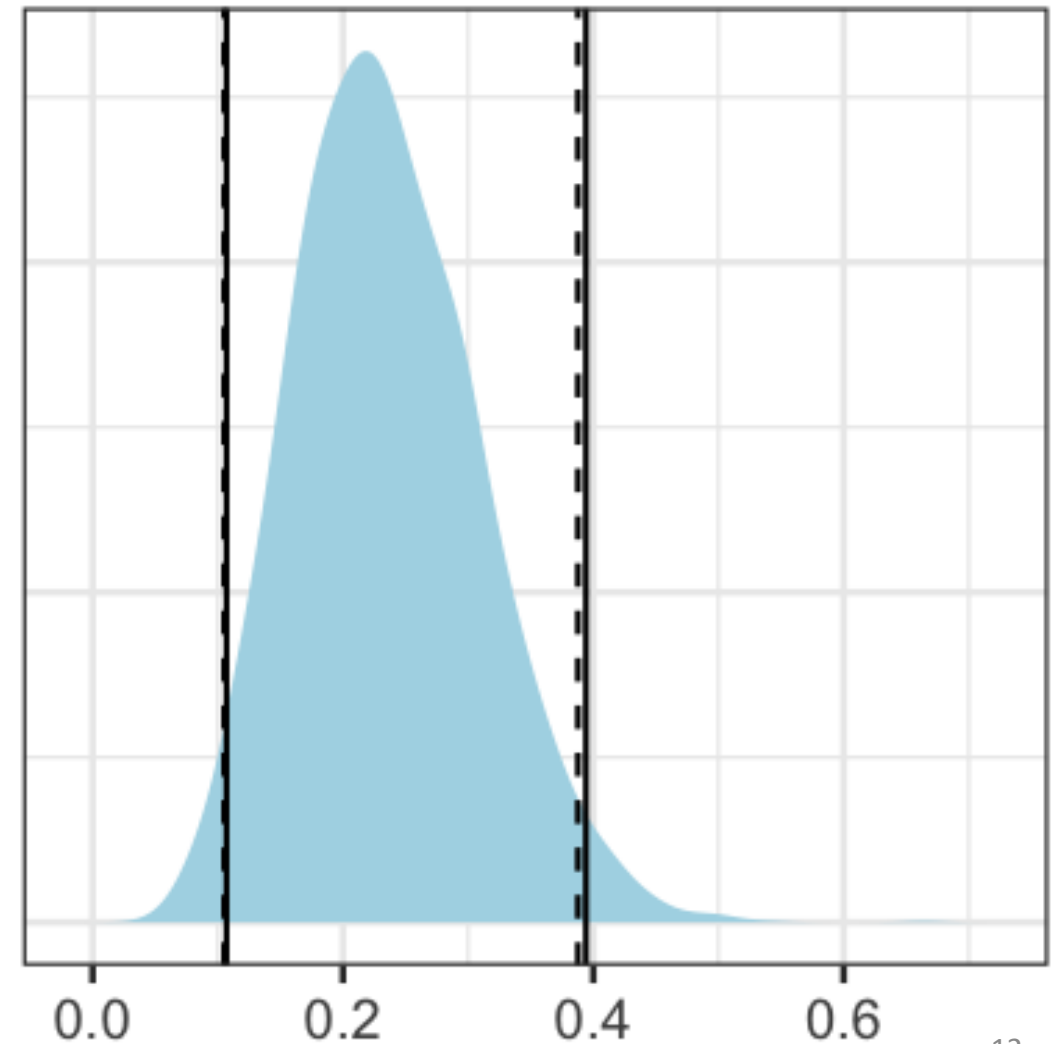


Mode = 0.506?



Posterior credible intervals

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



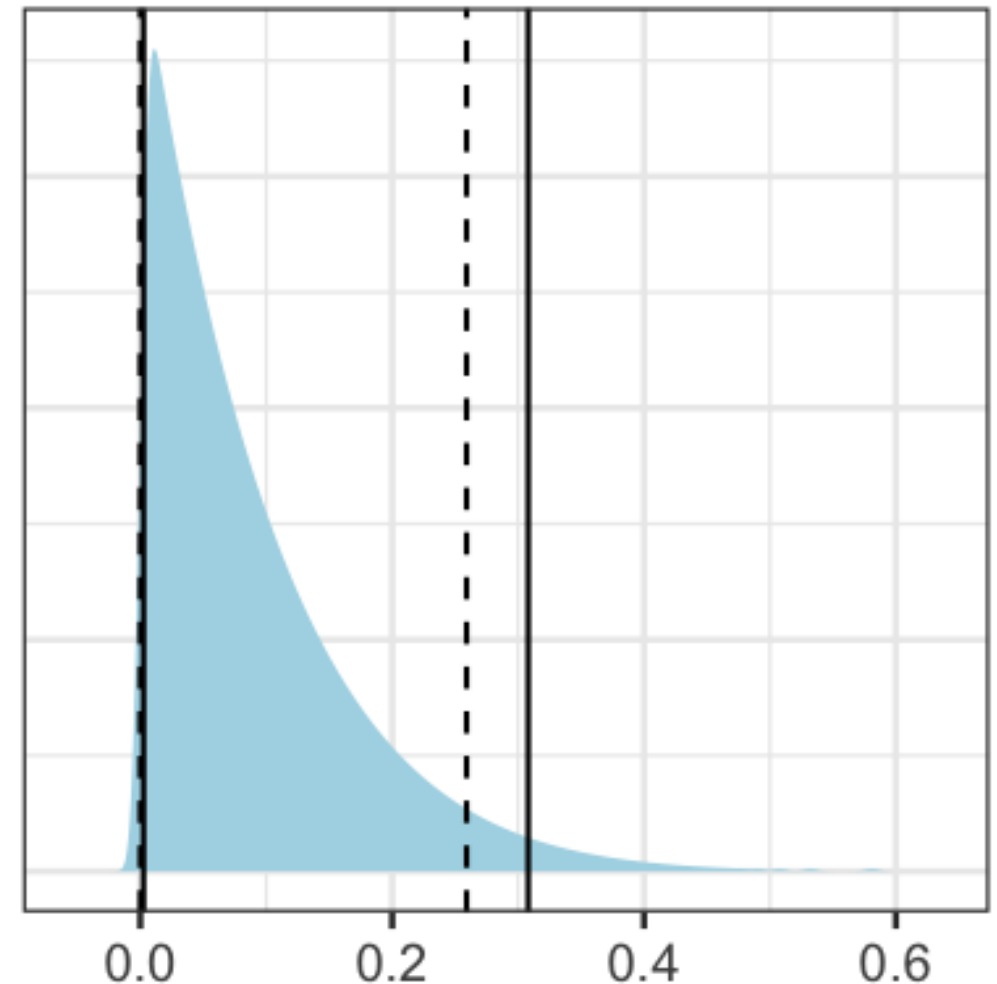
Posterior credible intervals

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)

Solid line = Equal tailed interval (ETI)

Dashed line = Highest density interval (HDI)



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

[nature](#) > [nature reviews methods primers](#) > [primers](#) > [article](#) > [table](#)

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 ²⁰⁹ 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 Mambaj 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

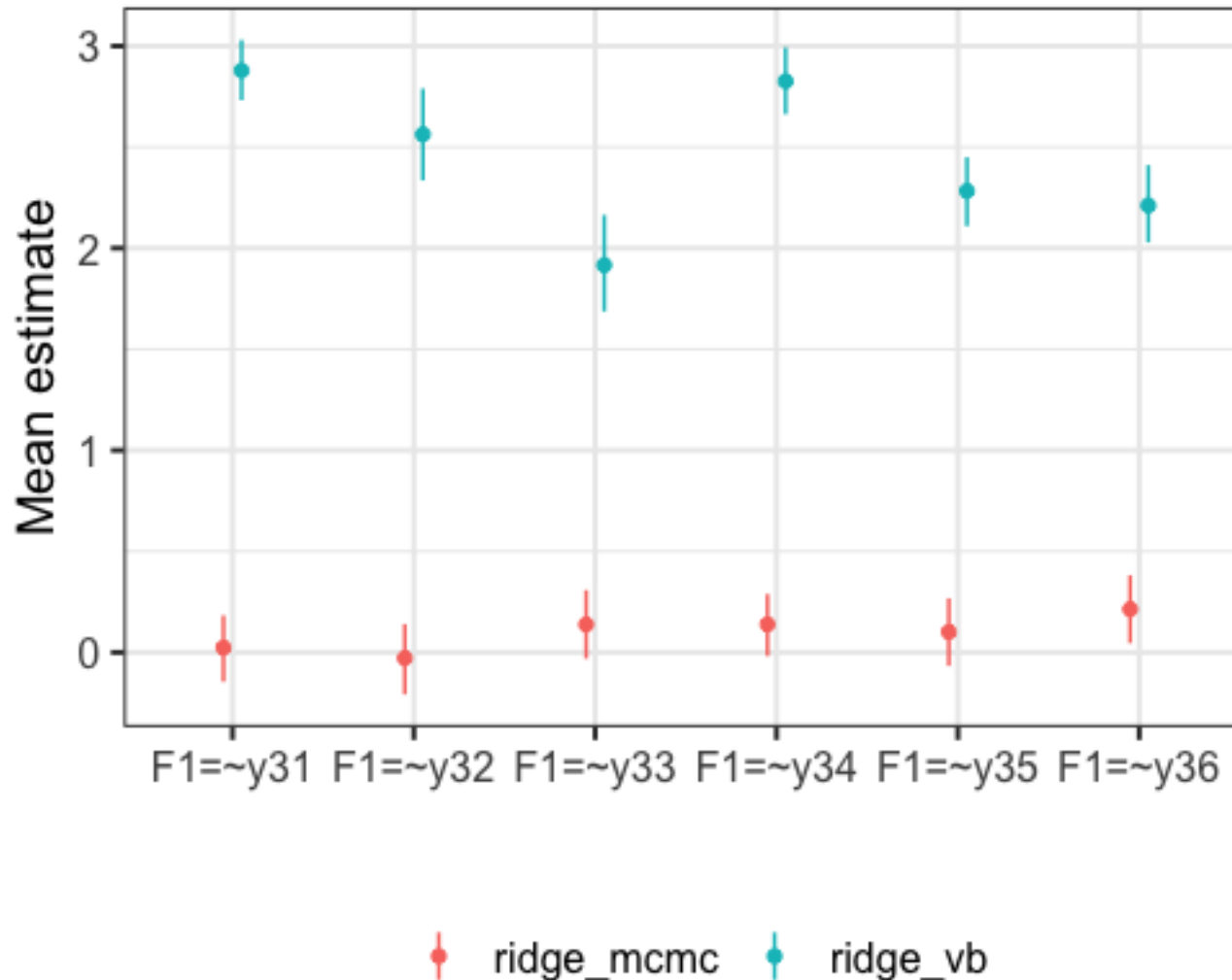
Exact algorithms

- 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 different, comparable distribution and optimize this distribution
- Assess convergence to ensure the approximation is close enough
- Fast and scalable
- E.g., variational inference, INLA

A cautionary note on approximate algorithms



R: Run Stan's variational algorithm for approximate posterior... Find in Topic

vb {rstan} R Documentation

Run Stan's variational algorithm for approximate posterior sampling

Description

Approximately draw from a posterior distribution using variational inference.

This is still considered an experimental feature. We recommend calling `stan` or `sampling` for final inferences and only using `vb` to get a rough idea of the parameter distributions.

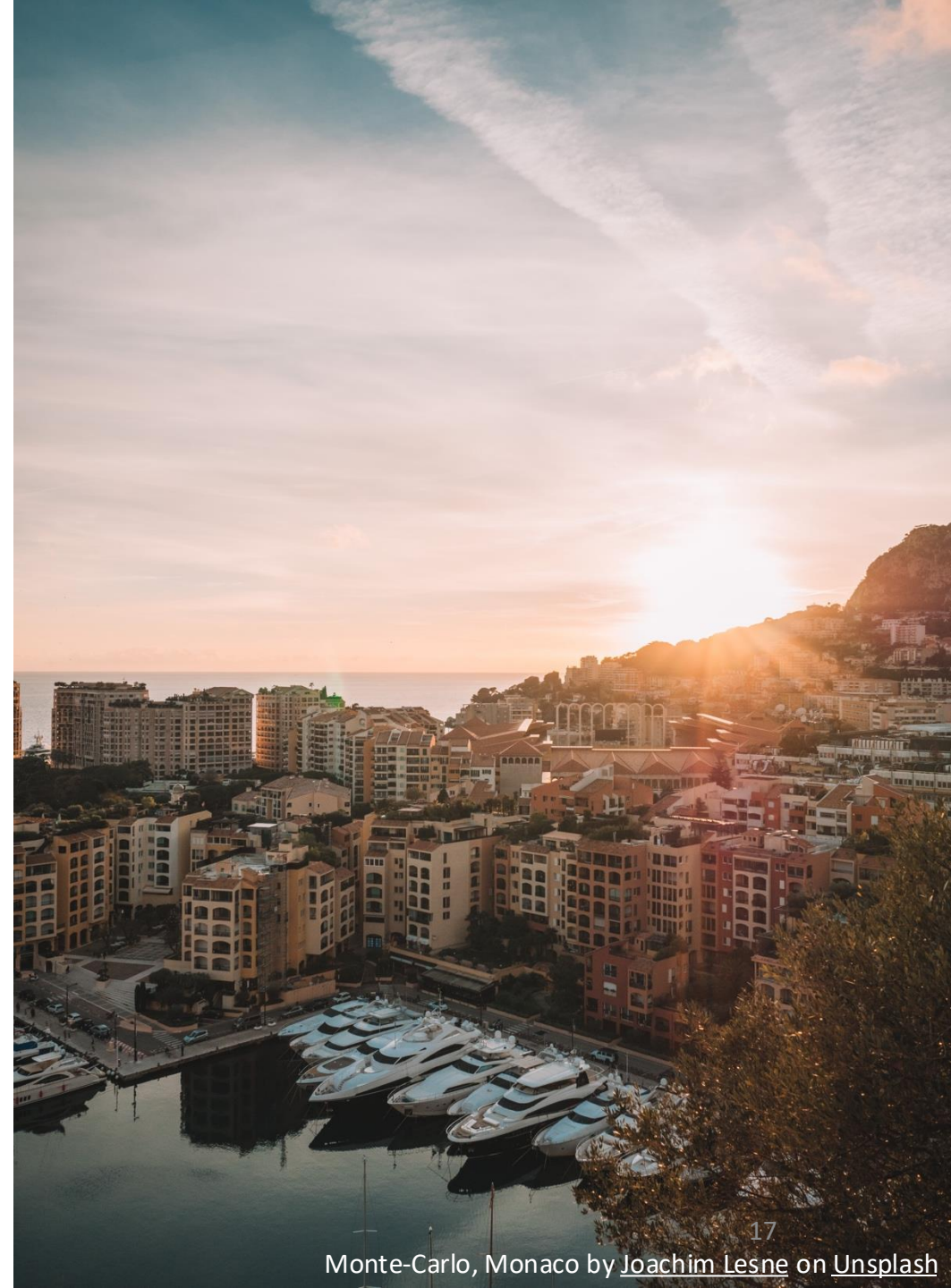
Markov Chain Monte Carlo (MCMC) sampling

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.



The idea behind MCMC

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

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

Or (for the regression model):

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

$$\text{With } p(y) = \int \int \int \int p(y|\beta_1, \beta_2, \beta_3, \sigma^2)p(\beta_1, \beta_2, \beta_3, \sigma^2)d\beta_1d\beta_2d\beta_3d\sigma^2$$

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.

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 $r = \text{value unnormalized posterior current iteration} / \text{unnormalized posterior previous iteration}$

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

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:

$$\begin{aligned}Post(\beta_1 | \beta_2, \beta_3, \sigma^2, data) &\sim Prior(\beta_1) \times likelihood \\Post(\beta_2 | \beta_1, \beta_3, \sigma^2, data) &\sim Prior(\beta_2) \times likelihood \\Post(\beta_3 | \beta_1, \beta_2, \sigma^2, data) &\sim Prior(\beta_3) \times likelihood \\Post(\sigma^2 | \beta_1, \beta_2, \beta_3, data) &\sim Prior(\sigma^2) \times likelihood\end{aligned}$$

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

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 from the target distribution (the posterior) to inform the proposal.

- Advantage: lower autocorrelation (but can take longer per iteration)
- Disadvantage: requires the derivatives (discrete parameters not possible)

Interactive demo



Interactive gallery of various MCMC algorithms:

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

So, what should I know?

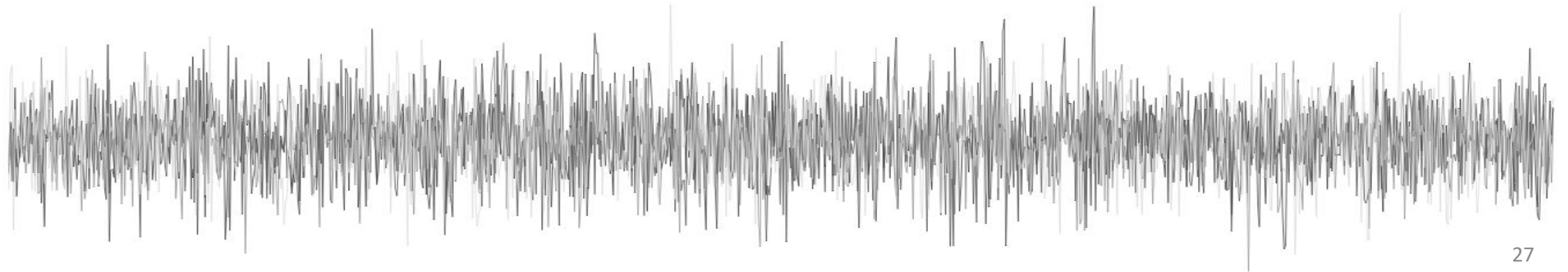
- Traditionally, software relied on Gibbs sampling (e.g., JAGS, Mplus)
- Stan and R-packages using Stan rely on Hamiltonian Monte Carlo (HMC)
- Both are special cases of Metropolis-Hastings
- Generally, HMC exhibits less autocorrelation, so less iterations needed
- HMC offers more convergence diagnostics,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

Convergence in Stan

Simple models will generally run

Potential solutions more complex, non-converging models:

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



Convergence in Stan

1. Traceplots should look like fat caterpillars
2. R_{hat} 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.



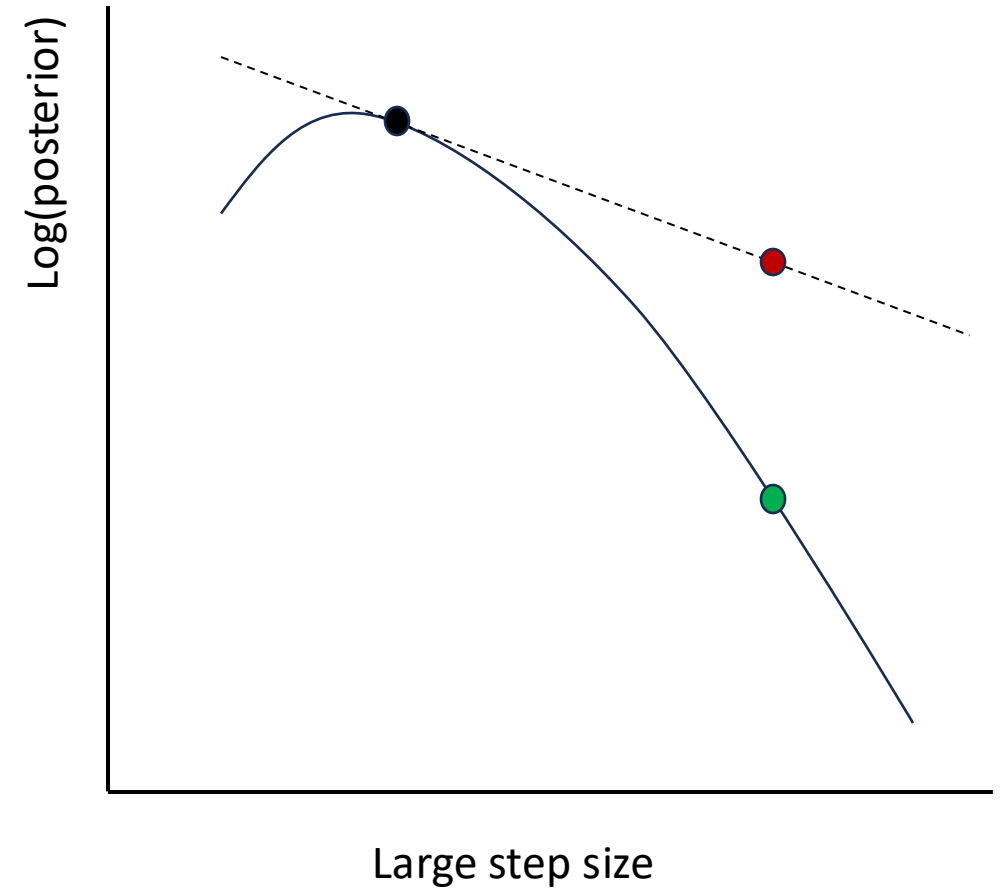
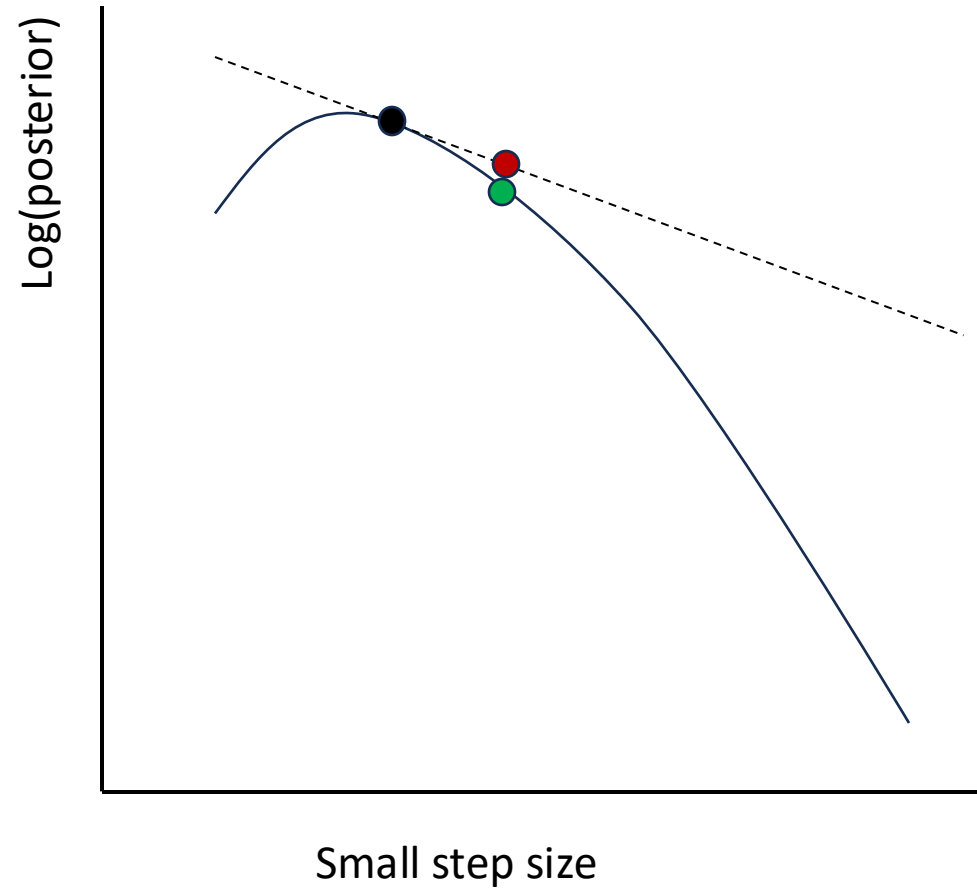
Convergence in Stan

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

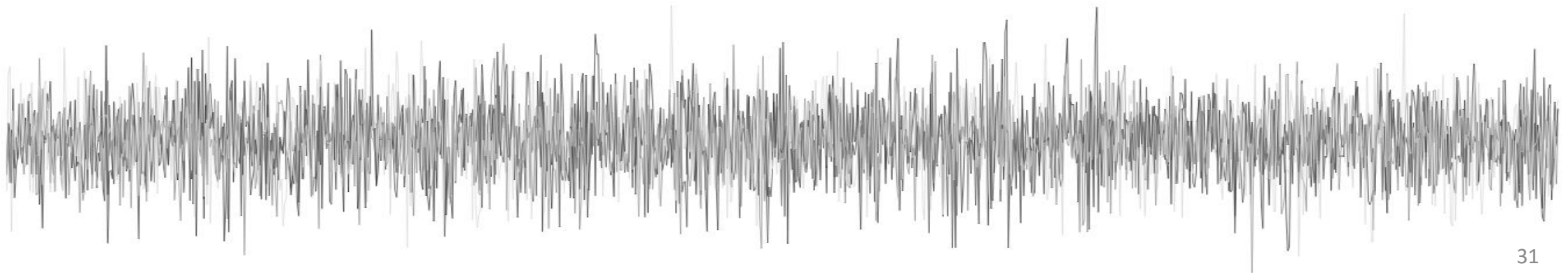


Divergent transitions in Stan

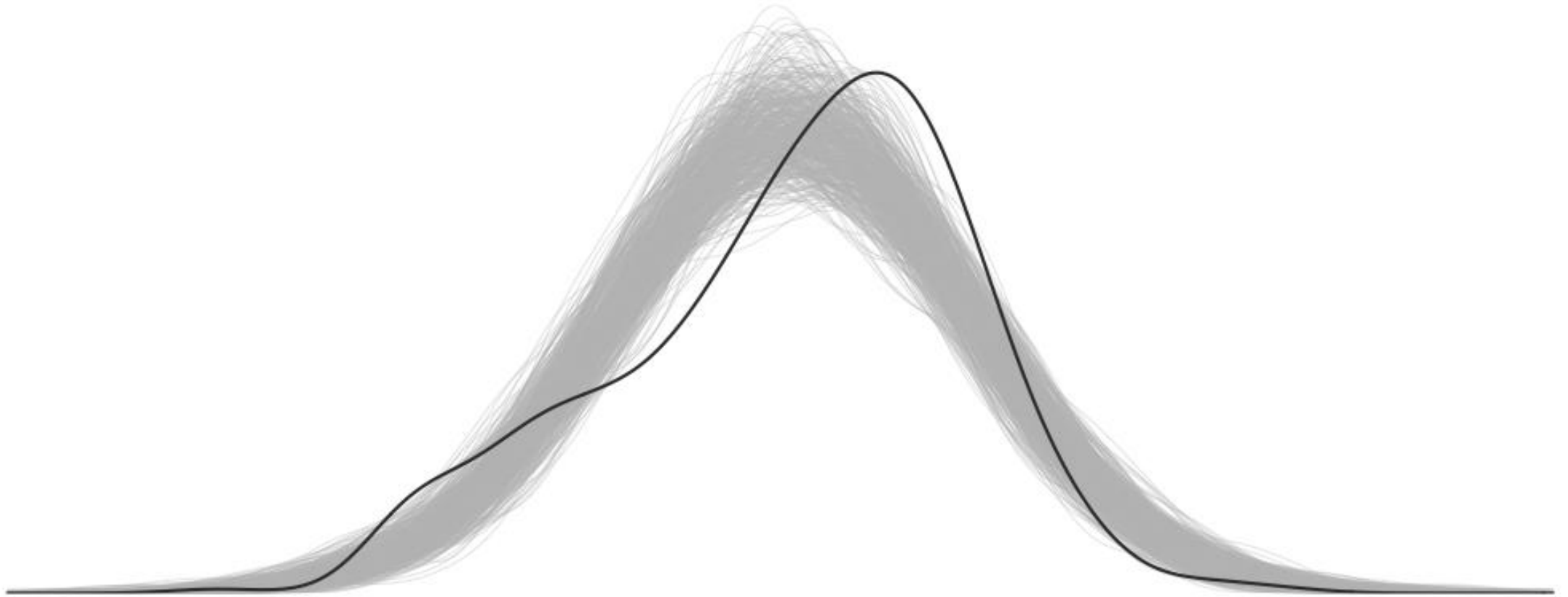


Convergence in Stan

- 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



Part 2: Predictive checks



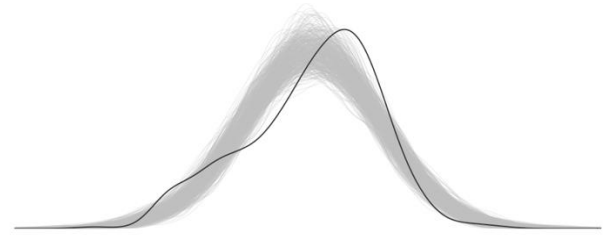
Why check your model?

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

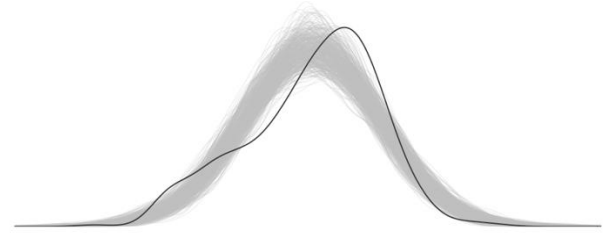


Data generated from the model should resemble the observed data.

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.

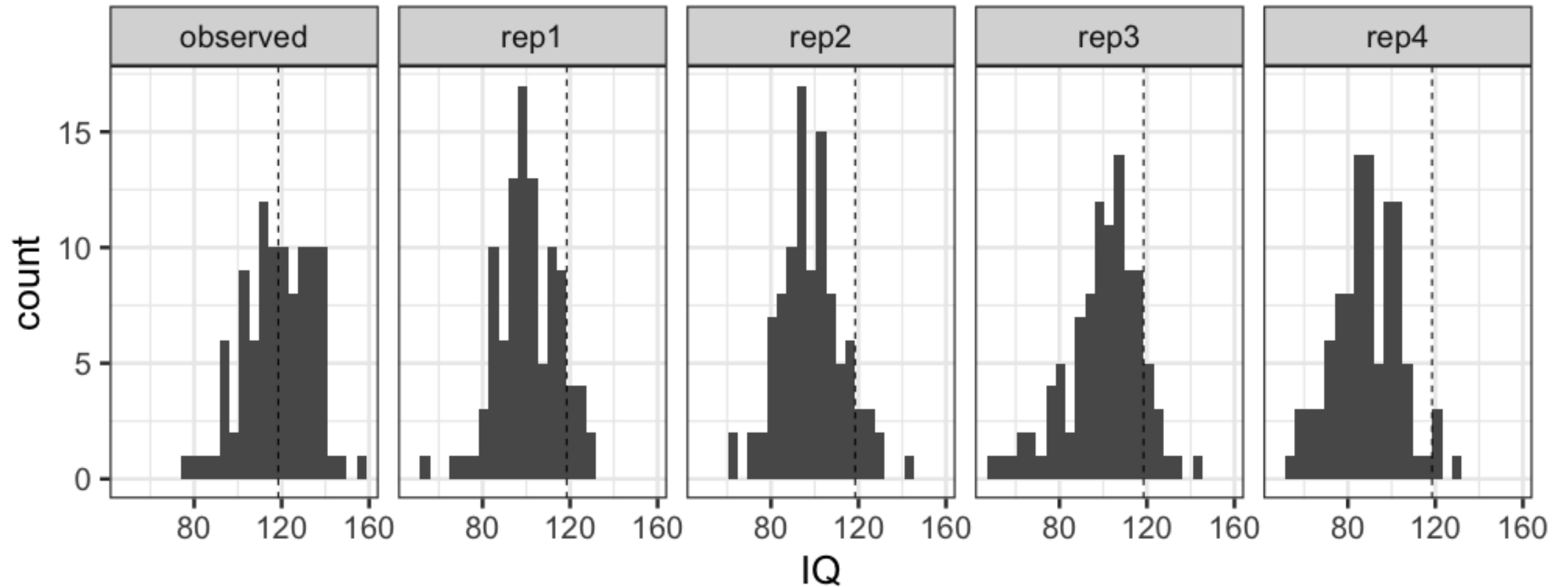
Posterior predictive checks



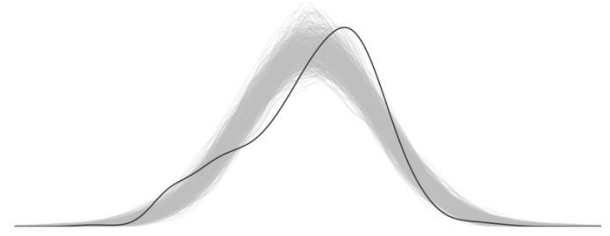
Suppose we have 100 replicated data sets from the posterior predictive distribution. How do we compare them to the observed data?

- 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)

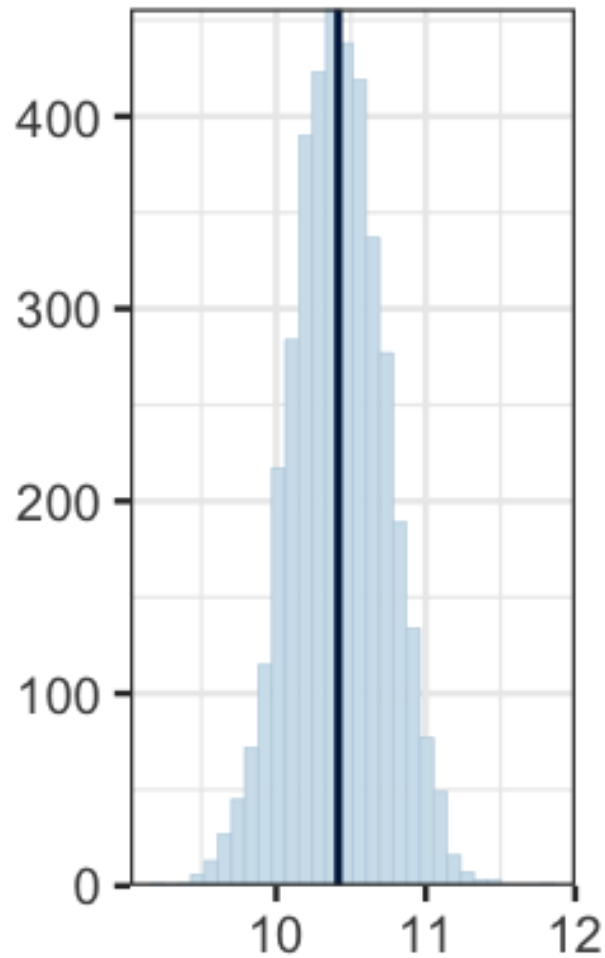
We will use linear regression to predict the math grade of 395 Portuguese students in secondary school.

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.



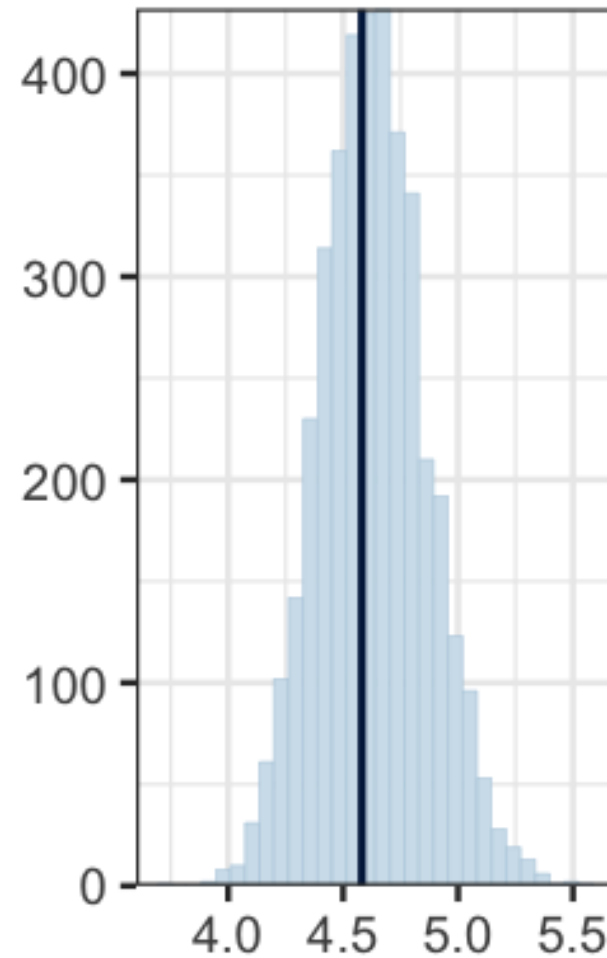
Basic posterior predictive checks



$T = \text{mean}$

■ $T(y_{\text{rep}})$

■ $T(y)$

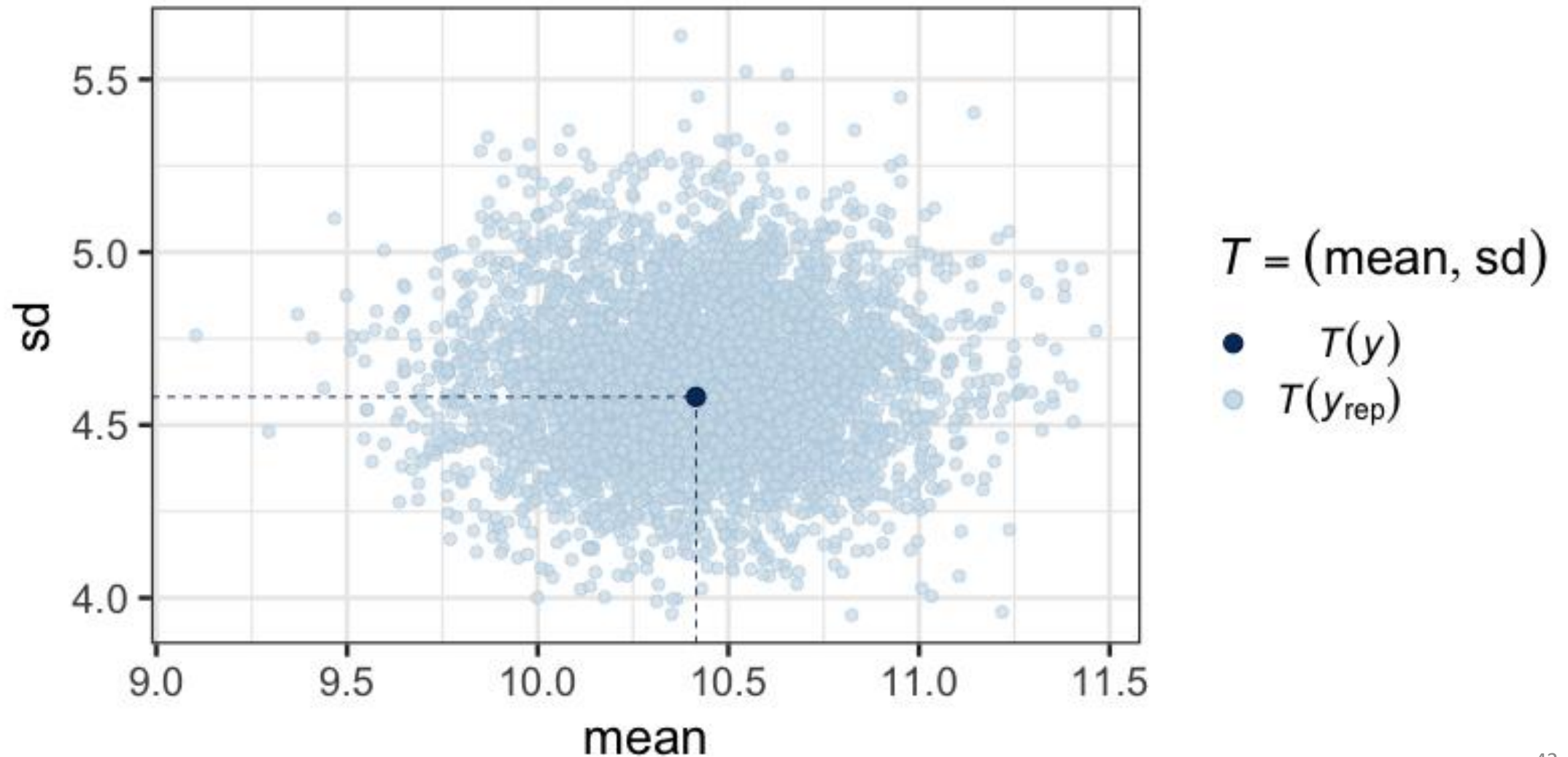


$T = \text{sd}$

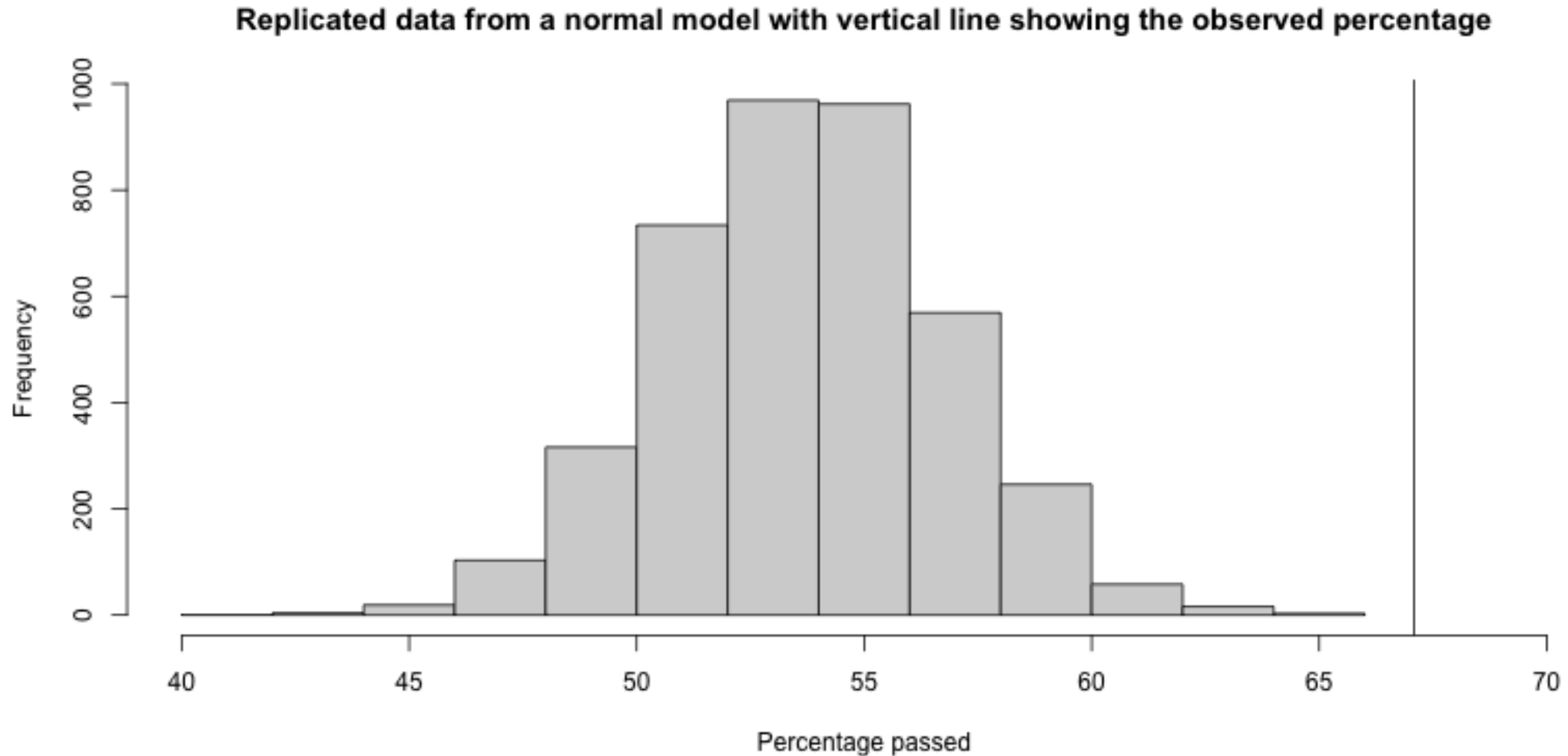
■ $T(y_{\text{rep}})$

■ $T(y)$

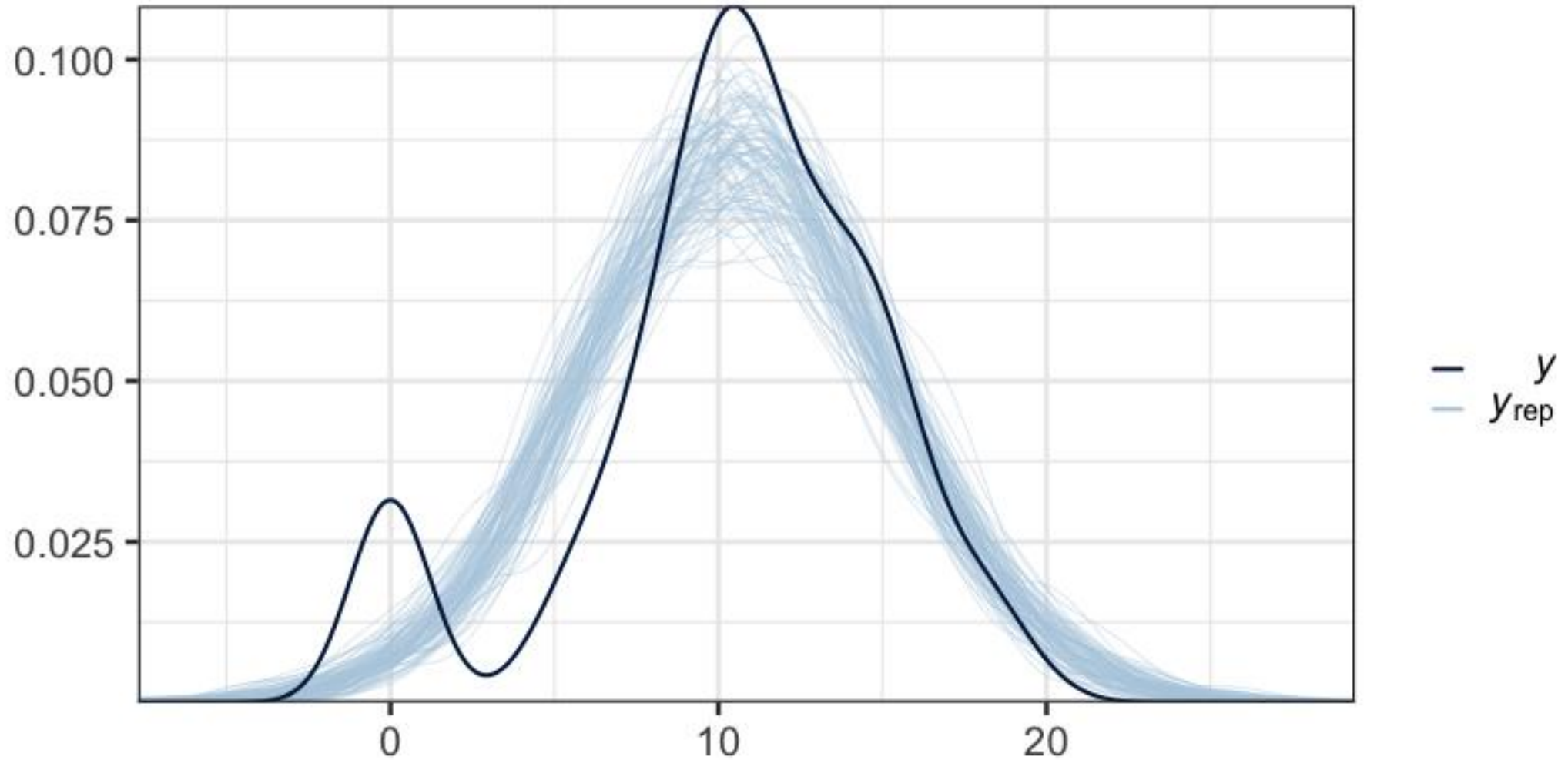
Basic posterior predictive checks



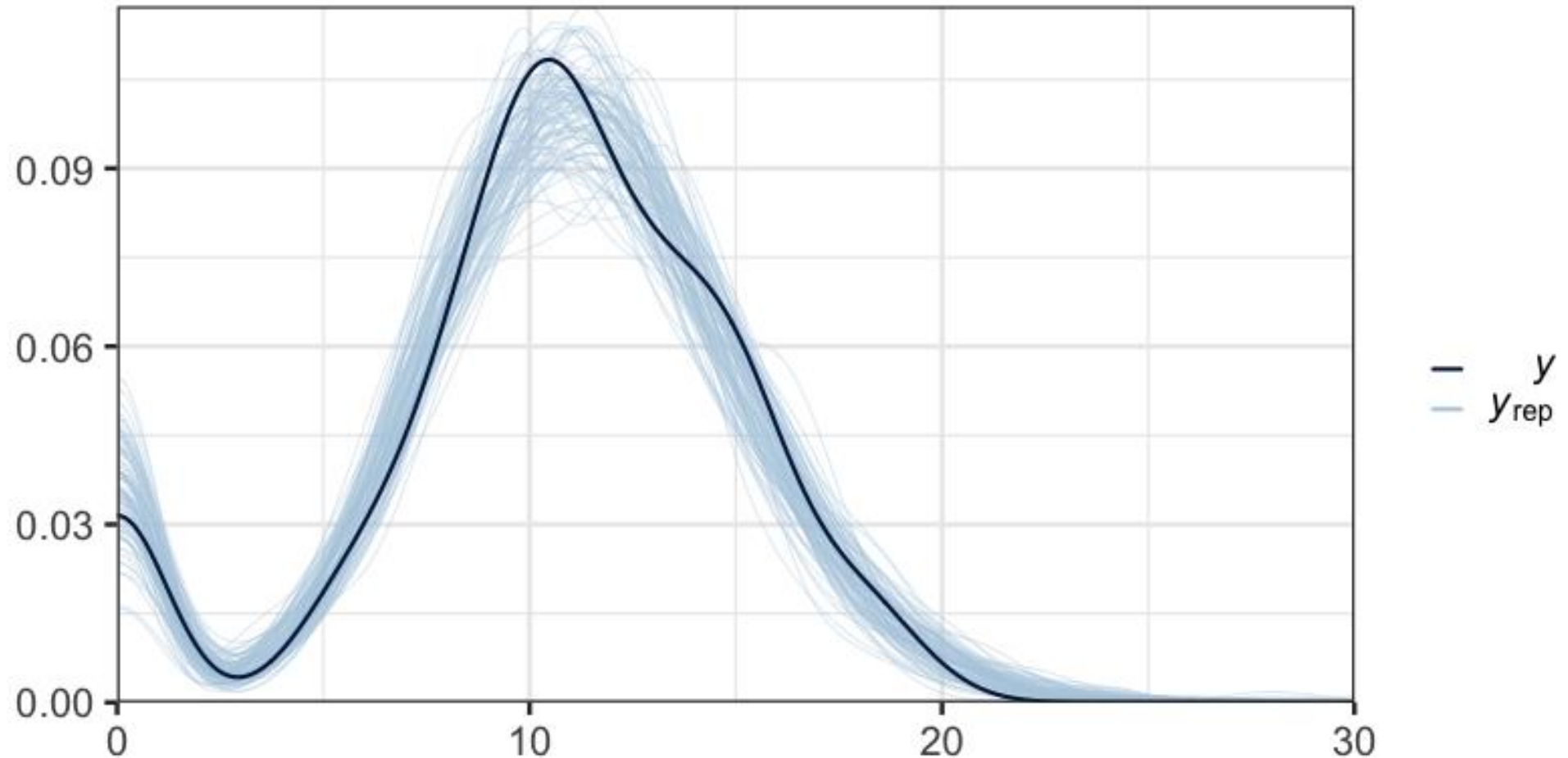
Custom posterior predictive checks



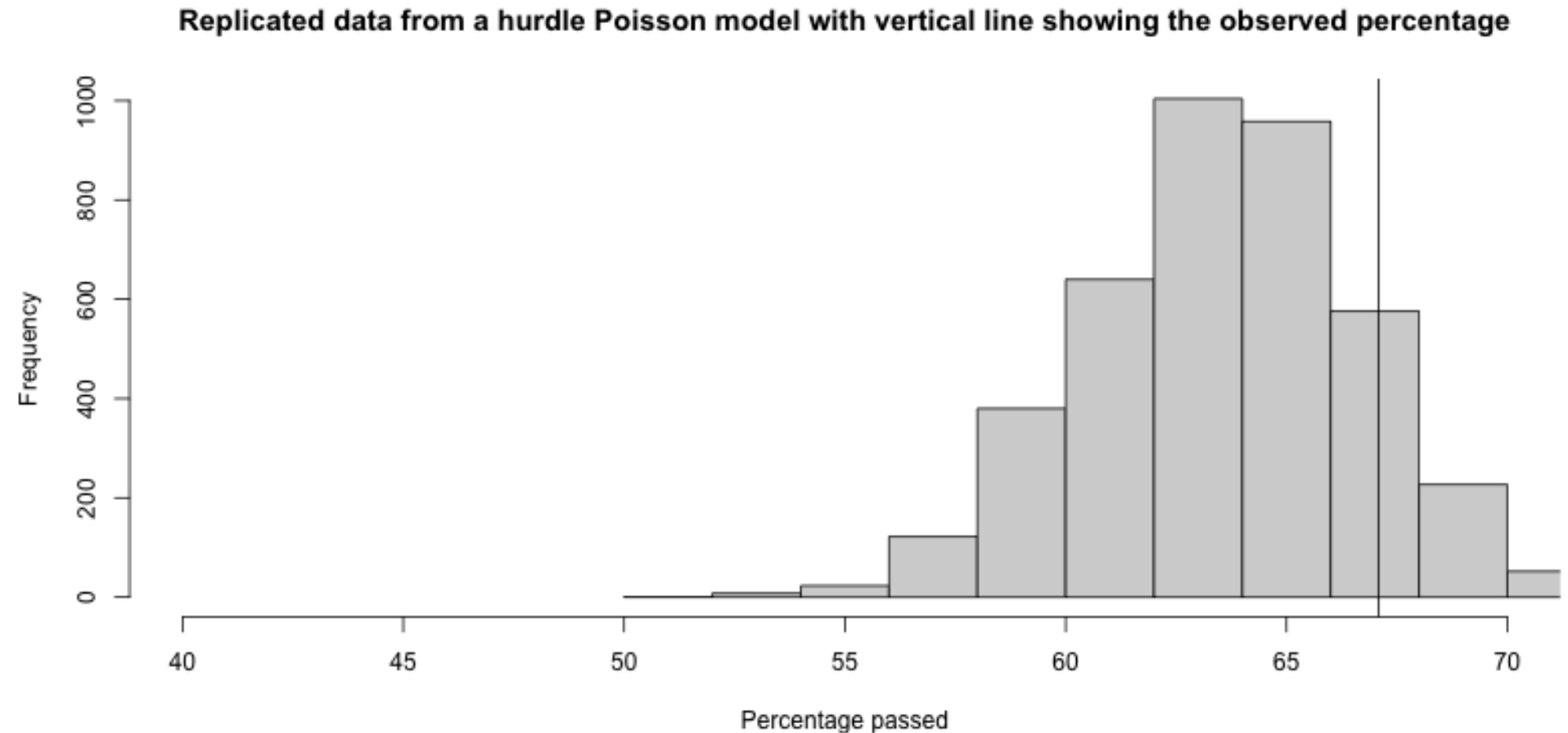
Custom posterior predictive checks



Improving the model: hurdle Poisson



Improving the model: hurdle Poisson



Recap

- 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

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.

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



Questions?