

Stan with R for analysis

R in Pharma

Aug 15, 2018

Daniel Lee

daniel@generable.com

Generable

The Abstract that Never Made It

This will be a quick, informal session about:

- ▶ what is Stan?
- ▶ what can it be used for? (what shouldn't it be used for?)
- ▶ why is there a lot of buzz around Stan now?
- ▶ how to use Stan from R
- ▶ examples in pharma
- ▶ time for questions

Before we start

- ▶ Questions
 - ▶ Who's heard of Stan?
 - ▶ Who's used Stan?
 - ▶ When did you start?
- ▶ Please ask questions at any time
 - ▶ Might get an immediate response
 - ▶ A delayed answer
 - ▶ Postponed for scope
(and we'll take it offline)

What is Stan?

What is Stan?

1. Language
2. Inference algorithms
3. Interfaces

Stan is a language

- ▶ **Purpose:** specify statistical models $p(\theta, x)$

- ▶ Influenced by BUGS and JAGS
- ▶ Imperative
- ▶ Statically-typed

- ▶ Ordinary differential equations!

Language

- ▶ Statistical model specification language
 - ▶ specify: data, parameters, joint probability distribution
- ▶ Domain specific language
 - not C++, R, Python, BUGS, JAGS...
- ▶ Imperative language
 - c.f. declarative (BUGS, JAGS), object oriented (Figaro)
- ▶ Language is built to formulate complicated statistical models, include ODEs.

What does Stan look like?

```
data {
  int<lower=0> N;
  int<lower=0,upper=1> y[N];
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
  y ~ bernoulli(theta);
}
```

Technical aside

- ▶ Gradients available for everything in the Stan language!
 - ▶ Reverse mode automatic differentiation
 - ▶ $O(1)$ with respect to function evaluation,
irrespective of the number of parameters!!
- ▶ The Stan compiler accepts a Stan program (.stan)
and generates a C++ header file (.hpp)
- ▶ The backend is all C++

Inference Algorithms

- ▶ Bayesian inference; Markov Chain Monte Carlo (MCMC)
- ▶ Approximate Bayesian inference
- ▶ Optimization

Inference Algorithms

- ▶ Bayesian inference; Markov Chain Monte Carlo (MCMC)
 - ▶ $p(\theta | x)$ approximated with $\{\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(N)}\}$
- ▶ Approximate Bayesian inference
 - ▶ ex: $\hat{p}(\theta | x) \approx q(\hat{\phi})$ where $\hat{\phi} = \operatorname{argmin}_{\phi} D_{\text{KL}}(q(\theta | \phi) || p(\theta | x))$
- ▶ Optimization
 - ▶ $\hat{\theta} = \operatorname{argmax}_{\theta} p(\theta, x)$ (only holds when there's a single optima)

Inference Algorithms

- ▶ Bayesian inference; Markov Chain Monte Carlo (MCMC)
 - ▶ **No-U-Turn Sampler (NUTS)**
 - ▶ Hamiltonian Monte Carlo (HMC)
- ▶ Approximate Bayesian inference
 - ▶ Automatic Differentiation Variational Inference (ADVI)
- ▶ Optimization
 - ▶ Limited memory Brayden-Fletcher-Goldfarb-Shanno (L-BFGS)

Platforms and Interfaces

- ▶ Runs in **Windows, Mac OS X, and Linux**
- ▶ C++ API: portable, standards compliant (C++11)
- ▶ Interfaces
 - ▶ **RStan**: R interface (Rcpp in memory)
 - ▶ CmdStan: command-line or shell access (direct executable)
 - ▶ PyStan: Python interface (Cython in memory)
 - ▶ MatlabStan: Matlab interface (external process)
 - ▶ Stan.jl: Julia interface (external process)
 - ▶ StataStan: Stata interface (external process)

Open source

- ▶ Stan is freedom-respecting, open-source software
 - ▶ Math library, Stan library, **CmdStan: new BSD license**
 - ▶ **RStan, PyStan: GPL**
- ▶ Hosted on GitHub
<http://github.com/stan-dev/>
- ▶ Over 65 contributors since 2011
- ▶ Active development

What can it be used for?

(what shouldn't it be used for?)

What shouldn't it be used for?

- ▶ Stan is not an R replacement.
 - ▶ Doesn't do general-purpose computing well.
 - ▶ No graphing built into the language.
- ▶ If there's already an R package for exactly what you want, you should probably use it.
- ▶ If you need it fast.

When should I use Stan?

- ▶ Limitations in your favorite package
 - ▶ You have a little more structure than the package allows for
 - ▶ It only does some custom inference algorithm
- ▶ Bayesian inference is needed: hierarchical models
 - ▶ Optimization / MLE isn't sensible for hierarchical models with less data
 - ▶ Conditions for approximate algorithms aren't met
- ▶ Have time to get it correct
 - ▶ Most uses are custom, requiring careful coding in Stan
 - ▶ Takes time to write models correctly

**What's with the
buzz?**

**Big advances in
Bayesian computing**

No-U-Turn Sampler
is a game changer

MCMC algorithms

Goal: estimate $p(\theta \mid x)$

► Estimate distribution with draws $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(N)}$

$$\mathbb{E}[f(\theta)] = \int f(\theta) \times p(\theta \mid x) d\theta$$

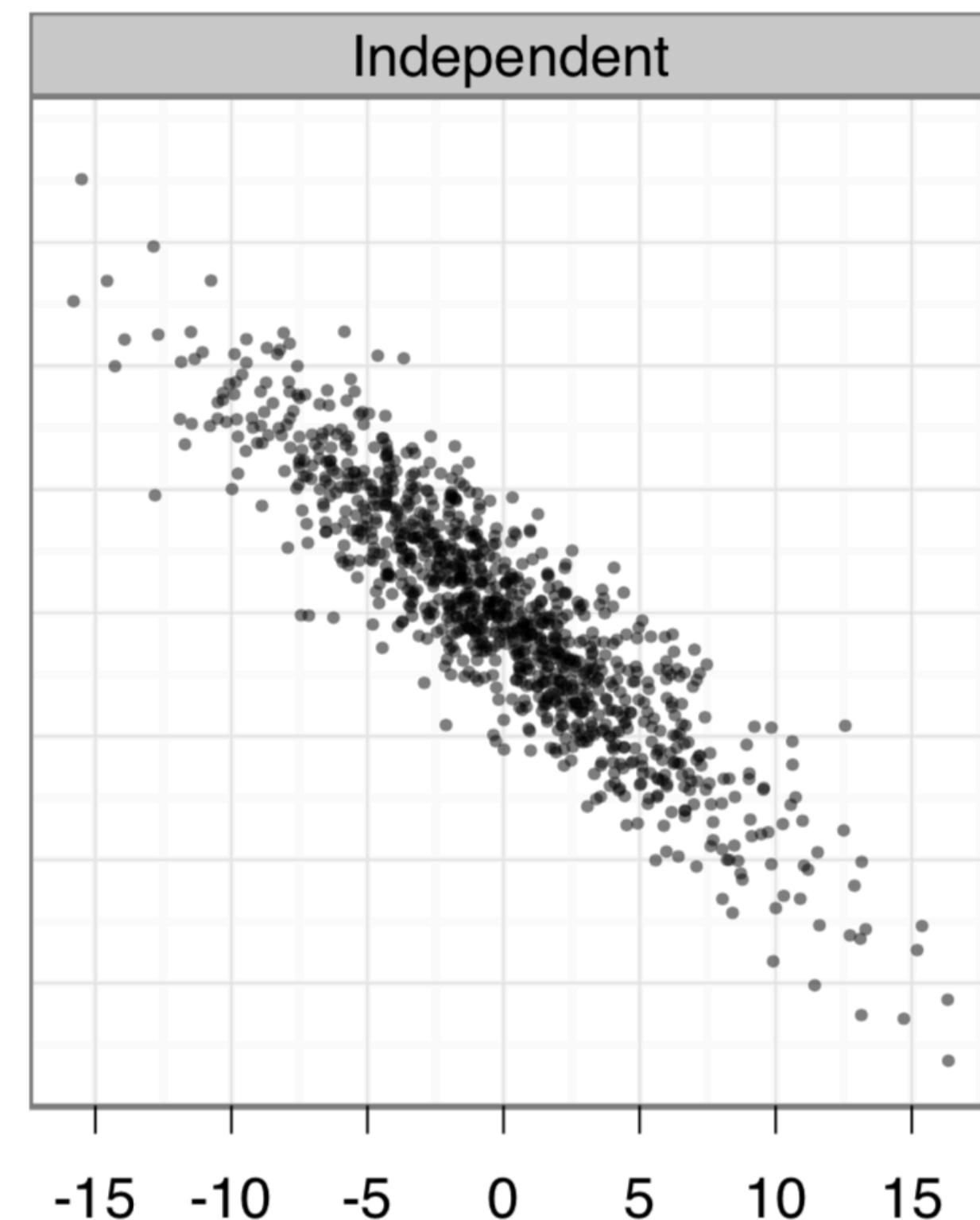
$$\approx \frac{1}{N} \sum_{n=1}^N f(\theta^{(n)})$$

Efficiency of different MCMC algorithms

- ▶ Random-walk Metropolis Hastings
- ▶ Gibbs sampling
- ▶ Hamiltonian Monte Carlo with **No-U-Turn Sampler (NUTS)**

250 dimensional Normal distribution

2d projection



1000 independent draws

250 dimensional Normal distribution

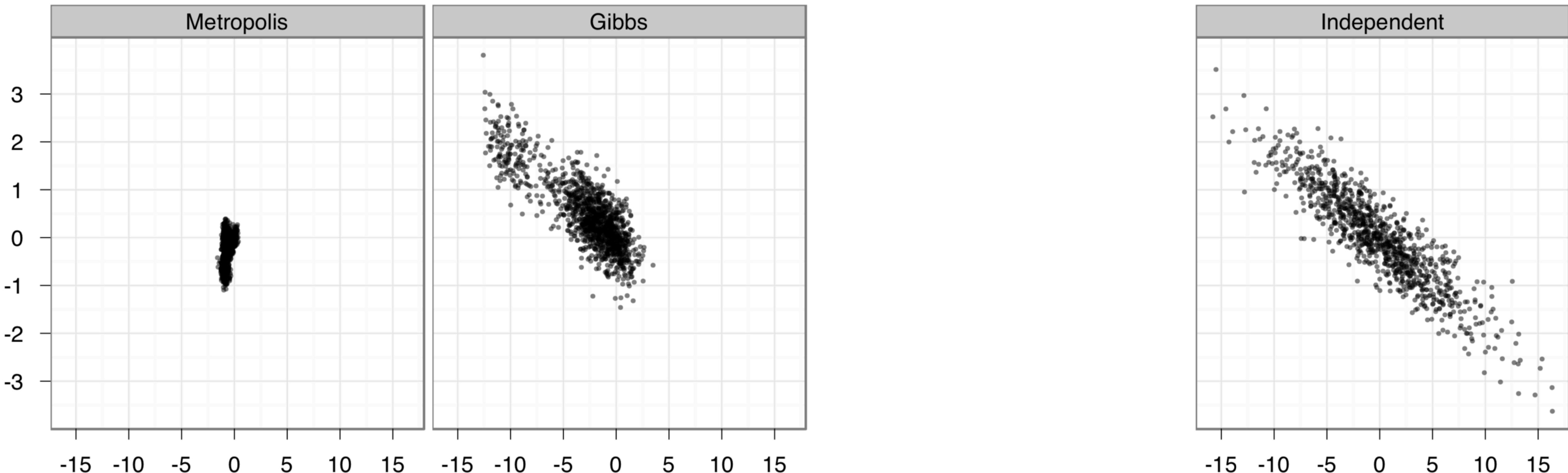
2d projection



1M draws (thinned)

250 dimensional Normal distribution

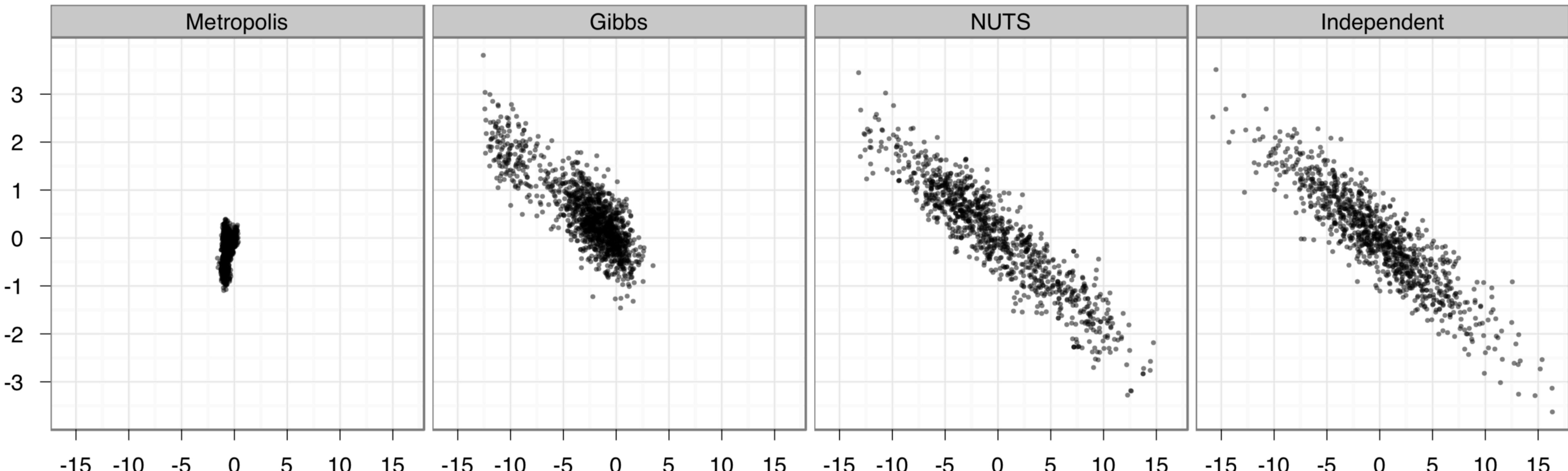
2d projection



1M draws (thinned)

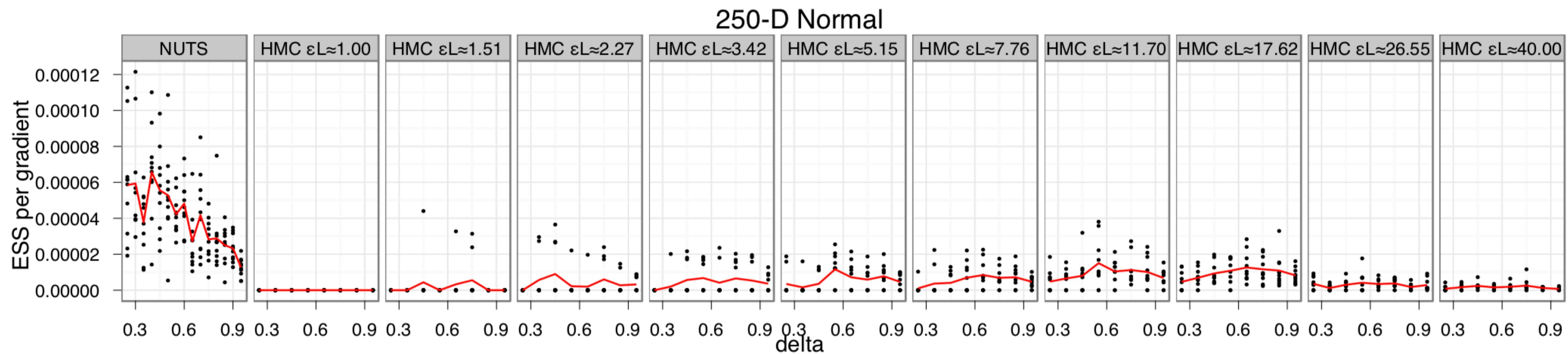
250 dimensional Normal distribution

2d projection

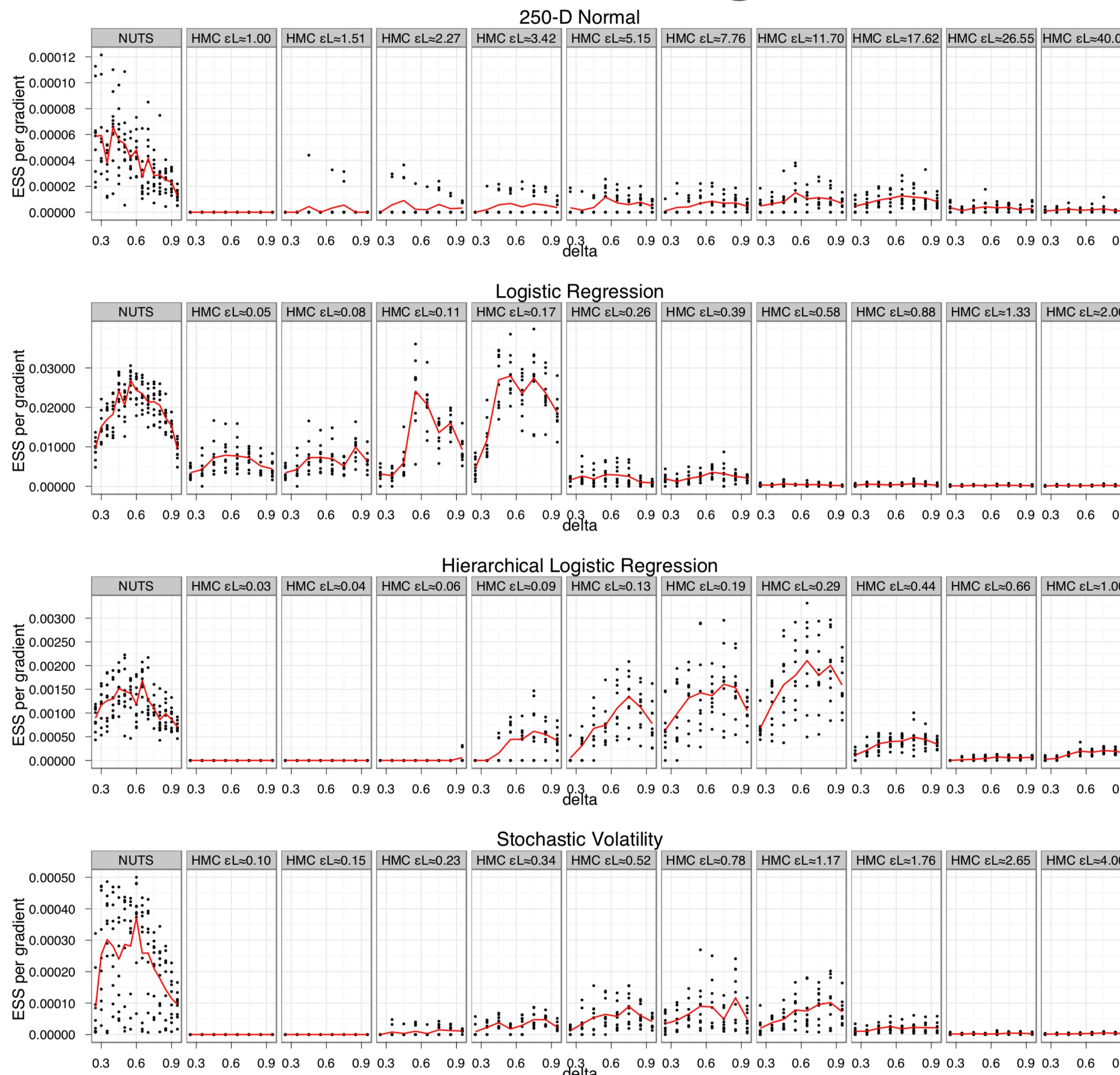


1000 draws

Quick aside: HMC is not enough



Quick aside: HMC is not enough



**Statistical inference
has changed since 2011!**

What's with the buzz?

- ▶ New algorithms (NUTS) requires gradients
 - ▶ Every Stan's model has gradients! No need to hand-write gradients.
- ▶ Ordinary Differential Equations are in the language!
 - ▶ ODEs are part of the Stan language.
 - ▶ Specify ODE and a statistical model
 - ▶ Can fit PK/PD models with full Bayesian inference!
- ▶ Stan's a newcomer
 - ▶ But... it's been around since 2011.

How to use Stan from R?

RStan installation

- ▶ It's on CRAN.

```
install.packages("rstan")
```

- ▶ Requires a C++ toolchain

- ▶ [Installing RStan on Mac or Linux](#)
- ▶ [Installing RStan on Windows](#)

RStudio Integration

- ▶ Stan syntax highlighting
- ▶ Stan syntax checking

RStudio: syntax highlighting

The screenshot shows the RStudio interface with a dark theme. The left pane displays the Stan code for a model named `example.stan`. The code includes sections for `functions`, `data`, and `transformed data`. Syntax highlighting is used to distinguish between different types of identifiers: `real` and `int` are in orange, while `functions`, `data`, and `transformed data` are in pink. The right pane shows the `Global Environment` which is currently empty, indicated by the message `Environment is empty`. Below the environment is the `Files` pane, which lists the project files: `rstudio-integration.Rproj` and `example.stan`. The bottom navigation bar includes tabs for `Console` and `Terminal`.

```
functions {
  real ode(real t, real y,
           real theta,
           real x_r, int x_i) {
    real dydt[1];
    real k_a;
    real k;
    real t_D;
    real D;
    real V;
    real dose;
    real elim;

    k_a = theta[1]; // Dosing rate
    k = theta[2];   // Elimination
    t_D = x_r[1];
    D = x_r[2];
    V = x_r[3];

    dose = 0;
    if (t > 0)
      dose = exp(-k_a * t) * D
      * k_a / V;
    elim = k * y[1];

    dydt[1] = dose - elim;
    return dydt;
  }
}

data {
  int T;
  real t0;
  real C0;
  real times[T];
  real D;
  real V;
  real t_D;
  real C_hat[T, 1];
}

transformed data {
  real x_r[3];
  int x_i[0];
  real C_init[1];

  x_r[1] = t_D;
  x_r[2] = D;
  x_r[3] = V;
  C_init[1] = C0;
}
```

RStudio: syntax validation

The screenshot shows the RStudio interface with the file `example.stan` open. Two buttons in the toolbar are highlighted with green circles: "Check on Save" and "Check". The code editor displays Stan code for a pharmacokinetic model. The bottom panel shows the R console output:

```
2:28 | Stan | ~ /syyclk/talks/2018-08-15-R-in-pharma-Stan-tutorial/rstudio-integration/ > rstan:::rstudio_stanc("example.stan")
example.stan is syntactically correct.
> |
```

The screenshot shows the RStudio interface with the file `example.stan` open. The "Check" button in the toolbar is highlighted with a green circle. The code editor displays the same Stan code as the left panel. The bottom panel shows the R console output, which includes an error message:

```
2:28 | Stan | ~ /syyclk/talks/2018-08-15-R-in-pharma-Stan-tutorial/rstudio-integration/ > rstan:::rstudio_stanc("example.stan")
SYNTAX ERROR, MESSAGE(S) FROM PARSER:
first argument to integrate_ode_bdf must be the name of a function with signature (real, real[], real[], real[], int[]) : real[]    error in 'model_example' at line 64, column 26
-----
62:   C = integrate_ode_bdf(code,
63:     C_init, t0, times,
64:     theta, x_r, x_i);
          ^
65: }

PARSER EXPECTED: ")"
Error in stanc(model_code = paste(program, collapse = "\n"), model_name = model_cppname, :
  failed to parse Stan model 'example' due to the above error.
> |
```

Running RStan: basics

1. Load library
`library(rstan)`
2. Write program to file: `example.stan`
3. Create data (in R): `dat`
4. Run
`fit <- stan("example.stan", data = dat)`
5. Posterior analysis
`print(fit)`

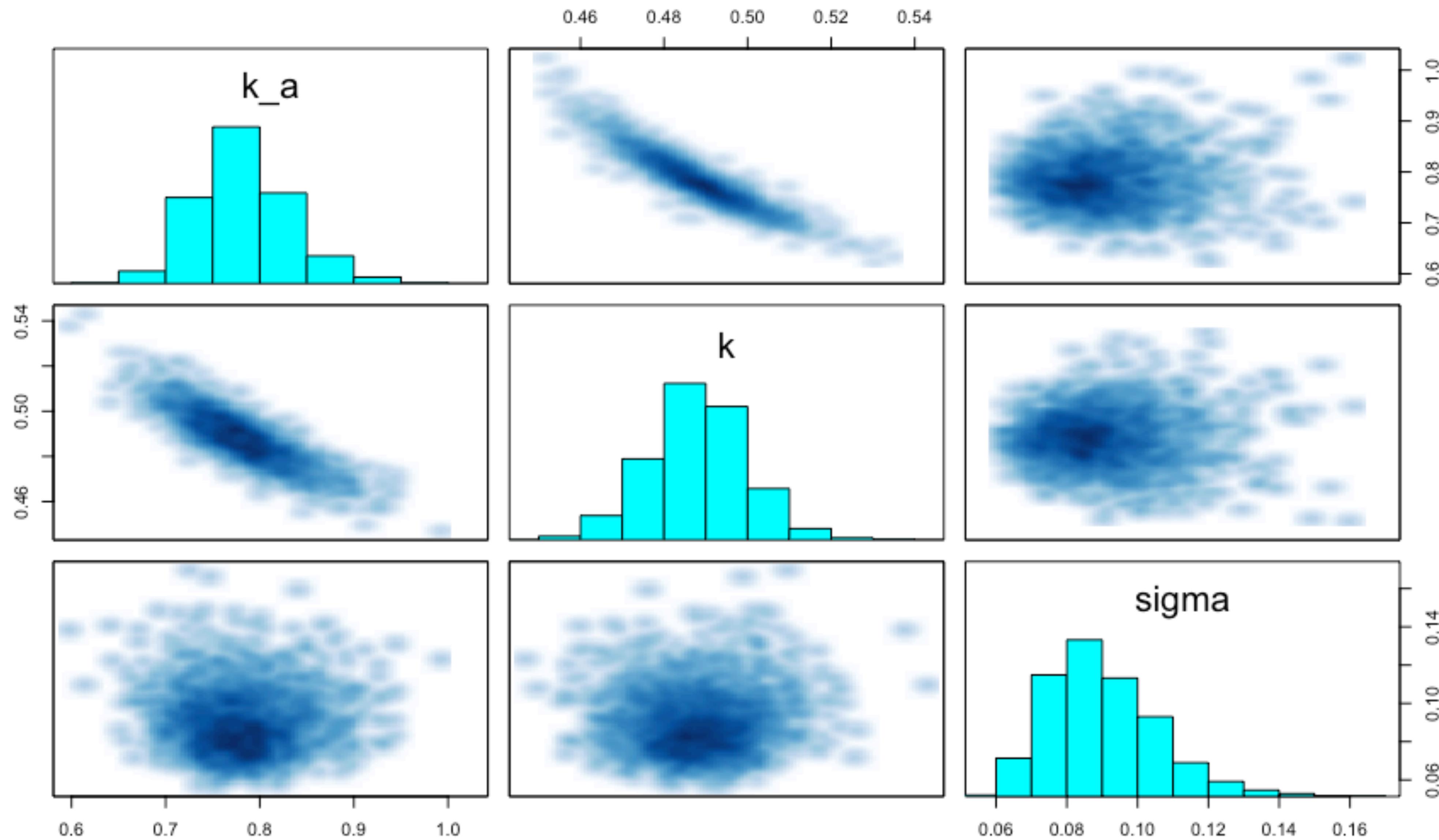
Posterior summary

```
> print(fit, c("k_a", "k", "sigma"))
Inference for Stan model: example.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
k_a	0.78	0	0.05	0.69	0.75	0.78	0.81	0.89	1785	1
k	0.49	0	0.01	0.47	0.48	0.49	0.50	0.51	1742	1
sigma	0.09	0	0.02	0.07	0.08	0.09	0.10	0.13	1788	1

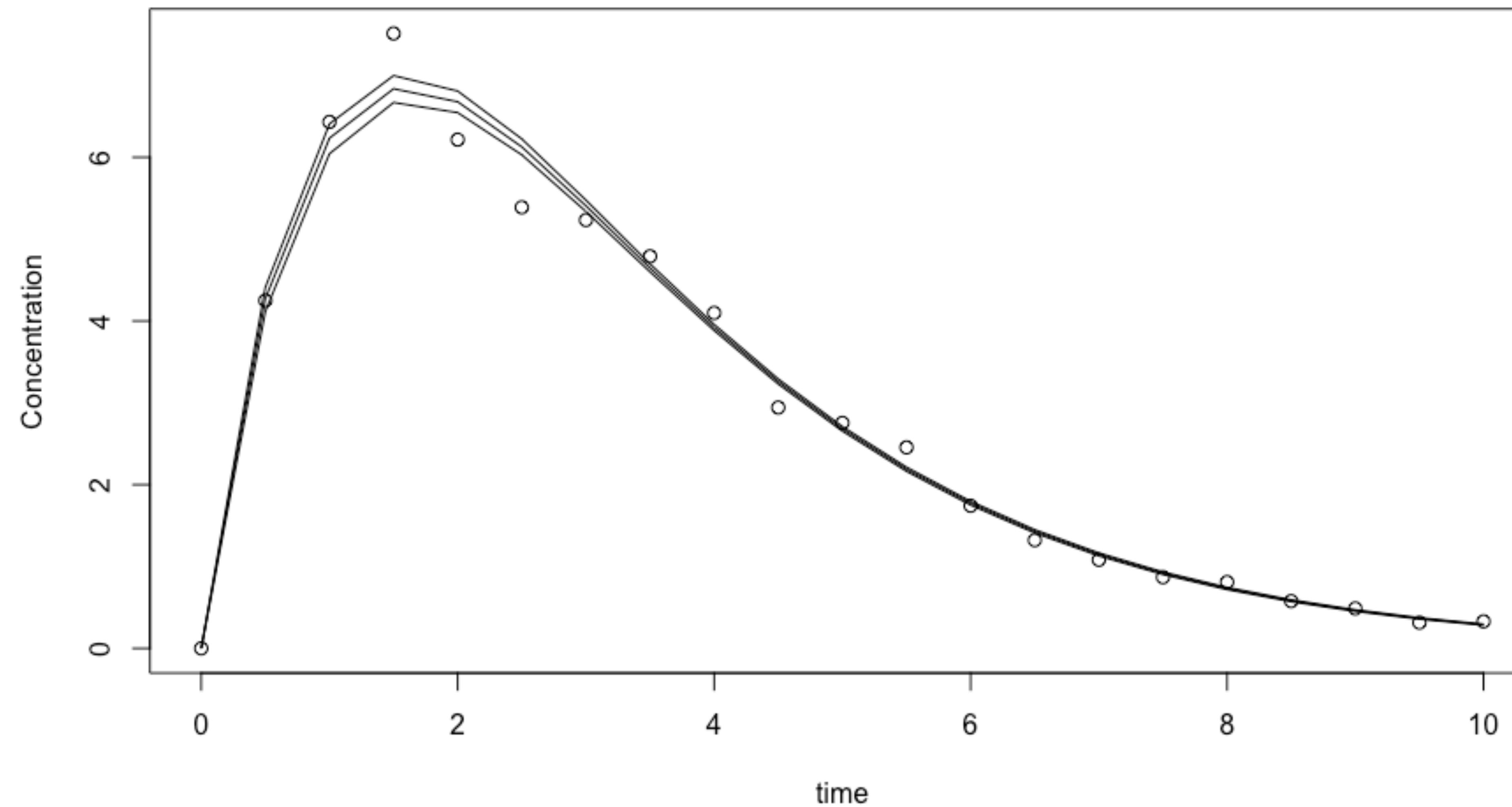
Samples were drawn using NUTS(diag_e) at Tue Aug 14 16:42:20 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).

Access joint posterior distribution



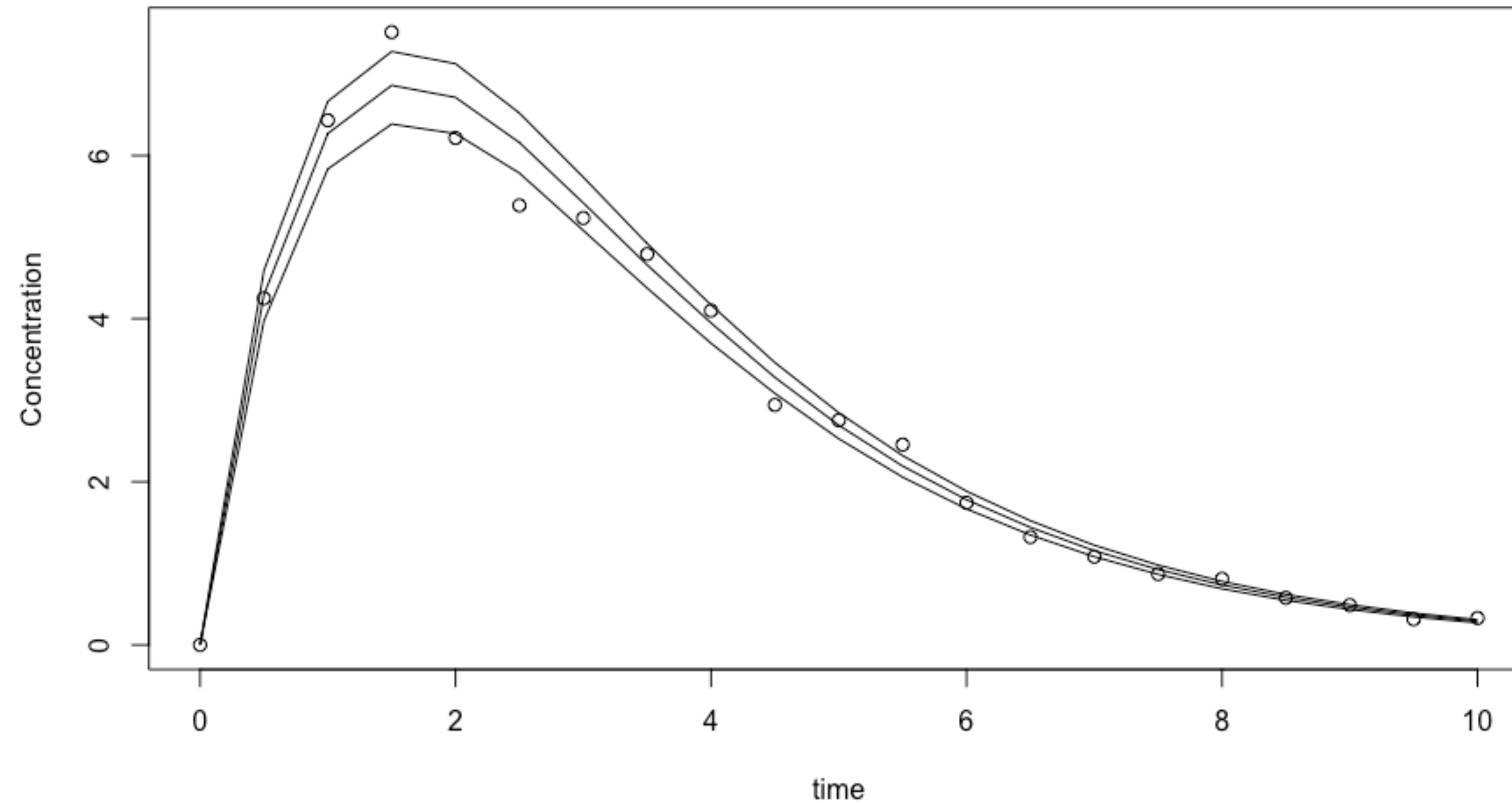
Posterior distribution of concentration

Concentration with 50% interval



Posterior predictive distribution

Posterior interval with 50% interval



examples

Decompose PK example

- ▶ Switch to RStudio to take a look.
- ▶ Look at types in Stan.
 - ▶ Statically typed.
 - ▶ Different from R.
- ▶ Blocks.
- ▶ Model.
- ▶ Compiling.

questions?

Where to get help?

- ▶ Stan Forums
<http://discourse.mc-stan.org>
- ▶ Stan Home Page
<http://mc-stan.org>
- ▶ Stan Documentation
<http://mc-stan.org/documentation/>

Contact

- ▶ Daniel Lee
- ▶ daniel@generable.com
- ▶ [@djsyclik](https://twitter.com/@djsyclik)
- ▶ <https://www.linkedin.com/in/syclik/>
- ▶ www.generable.com
- ▶ www.syclik.com

Bayesian inference

The Goal of Bayesian Inference

$$p(\theta | x)$$

posterior distribution of parameters
conditioned on data

The Goal of Bayesian Inference

$$p(\theta | x)$$

θ

parameters

x

data, outcomes

$$p(\cdot)$$

probability distribution

The Goal of Bayesian Inference

$$p(\theta | x)$$

θ

parameters

x

data, outcomes

$$p(\cdot)$$

probability distribution

implicit: conditioned on a model

Computing $p(\theta | x)$ is tough

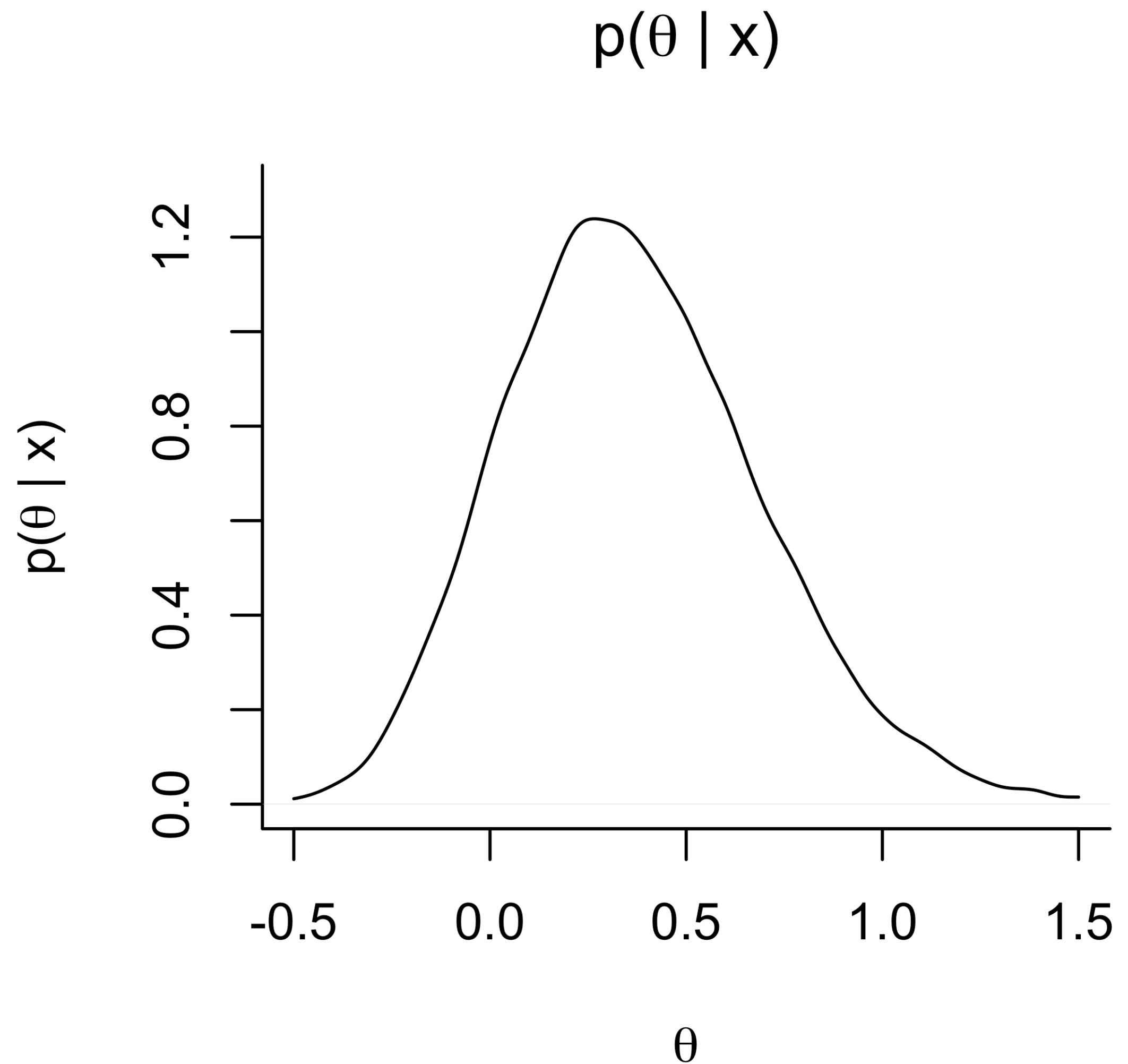
$$p(\theta | x) = \frac{p(\theta, x)}{\int p(\theta, x) d\theta}$$

Computing $p(\theta | x)$ is tough

$$p(\theta | x) = \frac{p(\theta, x)}{\int p(\theta, x) d\theta}$$

Why Bayesian inference?

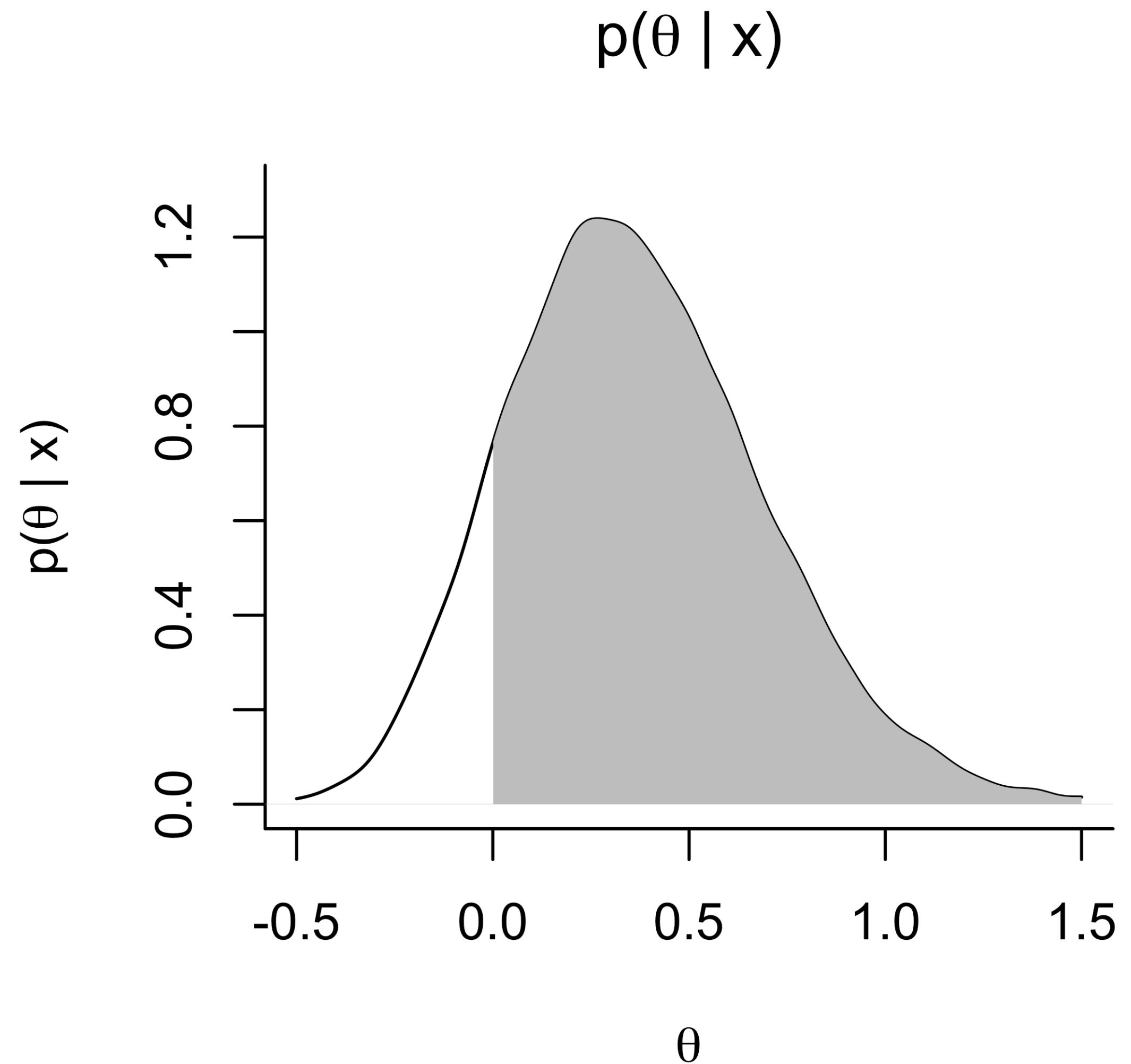
- ▶ Direct interpretation of probabilities
- ▶ Correct measure of uncertainty
- ▶ mode \neq mean



Why Bayesian inference?

Direct interpretations of probabilities

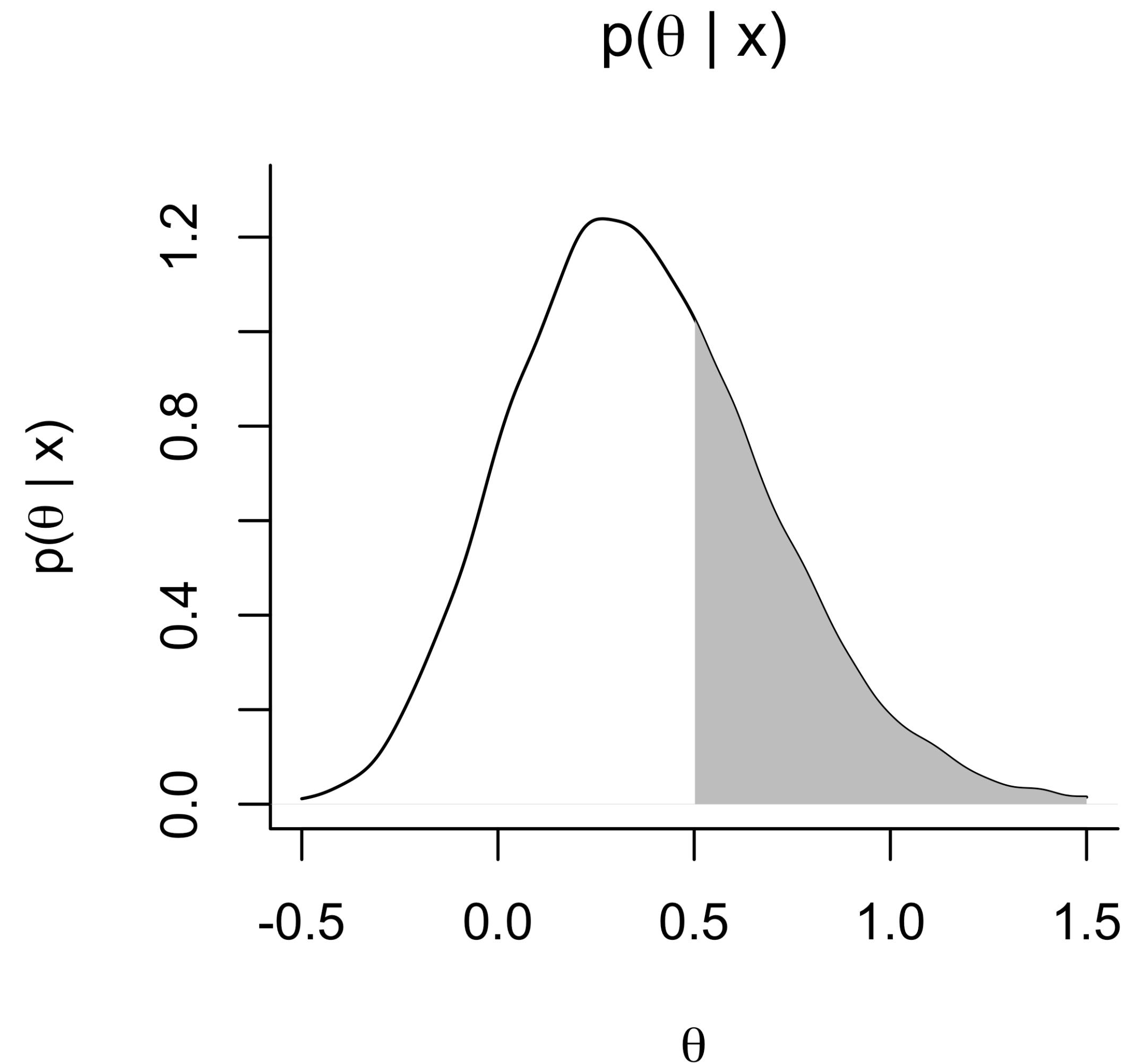
$$p(\theta > 0 | x) = 0.8748$$



Why Bayesian inference?

Direct interpretations of probabilities

$$p(\theta > 0.5 | x) = 0.3210$$

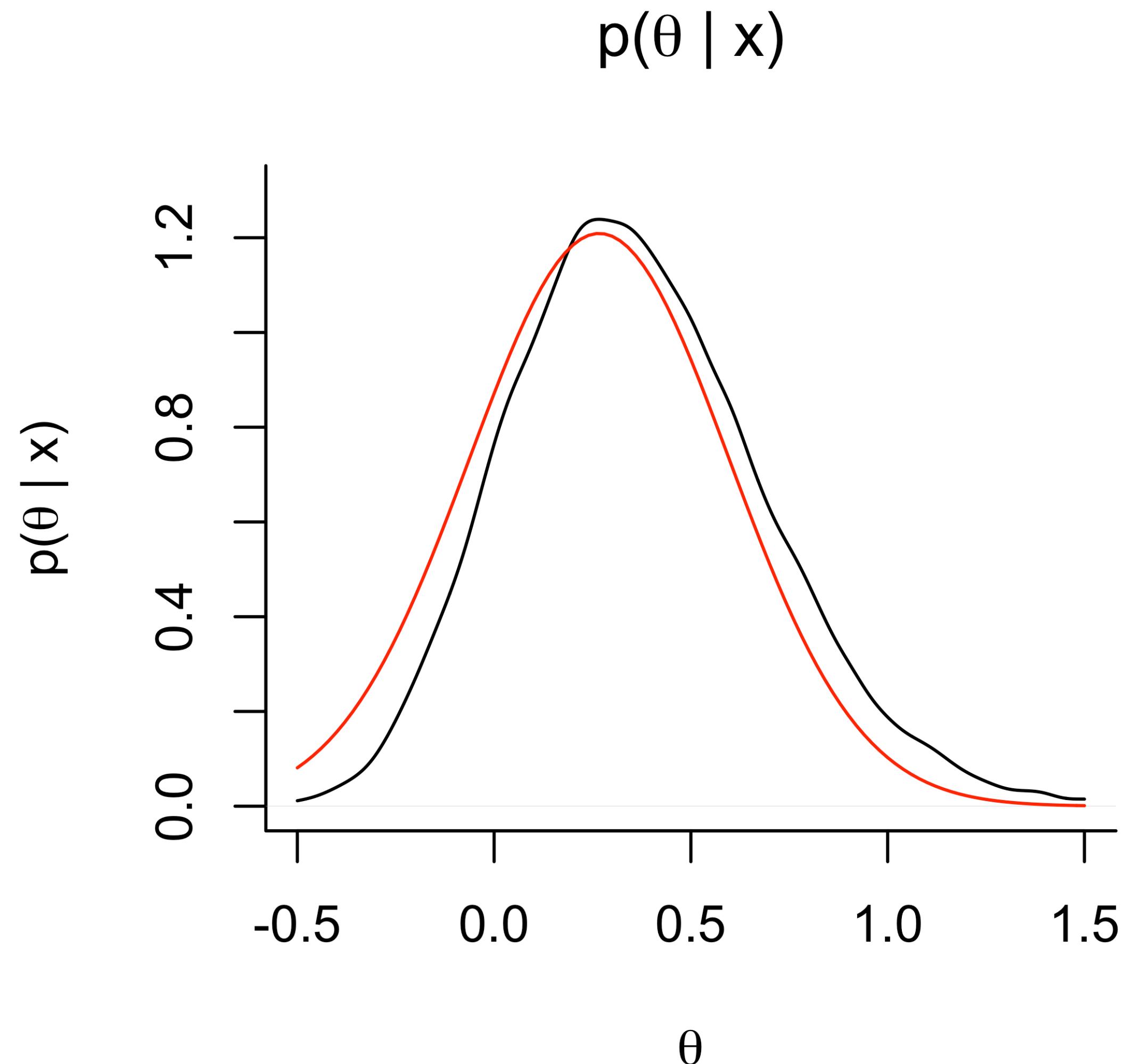


Why Bayesian inference?

Correct measure of uncertainty

- ▶ Red
- ▶ approximating posterior with normal (mode, standard deviation)
- ▶ needs conditions for this to work
- ▶ Estimates accounting for uncertainty:

$$\mathbb{E}[f(\theta)] = \int f(\theta) \times p(\theta | x) d\theta$$



Why Bayesian inference?

mode \neq mean; affects decisions

► blue: $E[\theta] = 0.367$

► red: $\hat{\theta} = 0.267$

► More importantly:
can take any expectation

