Imperial College London

12th January 2023

An Introduction to Bayesian Regression in Stan

Joseph D. Challenger

Introduction

The purpose of this session is to introduce you to specifying models in Stan, using simple regression models as illustrative examples. We'll also briefly introduce how to build a Bayesian regression models, and generate results with the fitted models. We'll also discuss the built-in diagnostic tools in Stan, which tell us whether the MCMC chains are well-behaved.

About Stan

Stan is a widely used platform for statistical modeling and high-performance statistical computation. Amongst other things, it can perform full Bayesian statistical inference with MCMC sampling. It has been used to solve research problems in a range of scientific fields.

Stan interfaces with the most popular data analysis languages (R, Python, shell, MATLAB, Julia, Stata) and runs on all major platforms (Linux, Mac, Windows). Documentation can be found here:

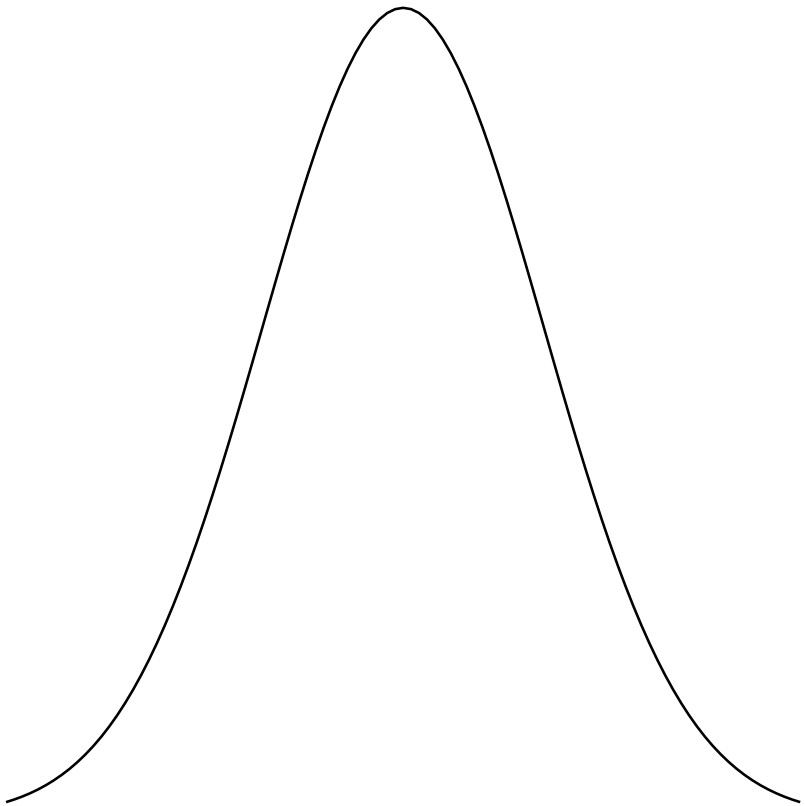
`https://mc-stan.org/users/documentation/`

Hamiltonian MCMC

In Bayesian inference, one challenge always crops up: how to generate random samples that are distributed according to a target probability distribution (*i.e.* the posterior distribution)?

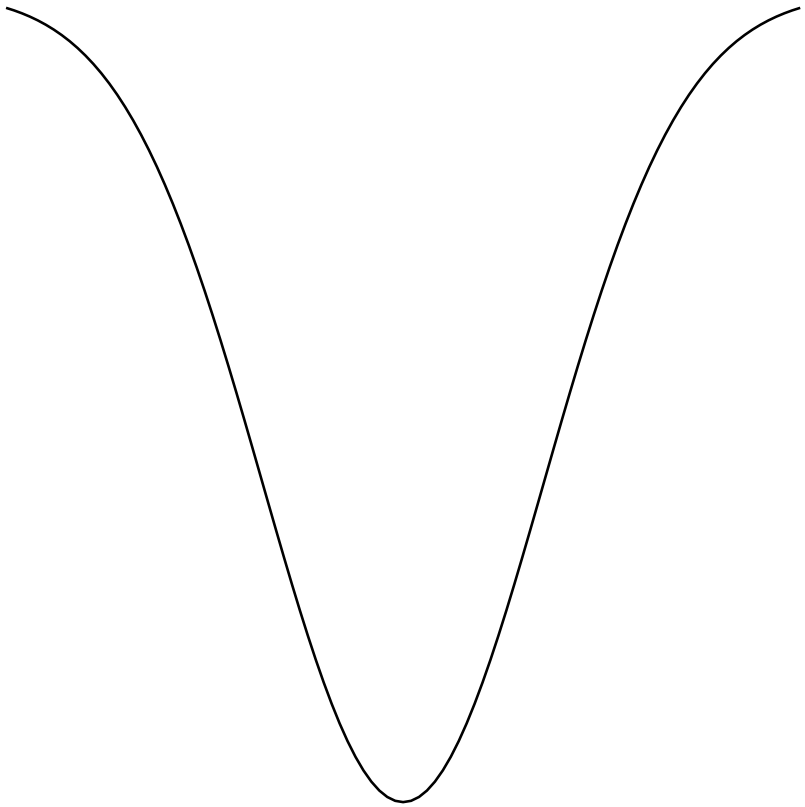
This is challenging, for a distribution of arbitrary shape, and potentially of high dimension (*i.e.* many parameters to infer). Stan uses **Hamiltonian Monte Carlo**. This has its foundations in classical mechanics: Hamilton's equations describe the motion of a physical object when it is acted on by a *conservative* force, such as gravitation.

Hamiltonian MCMC



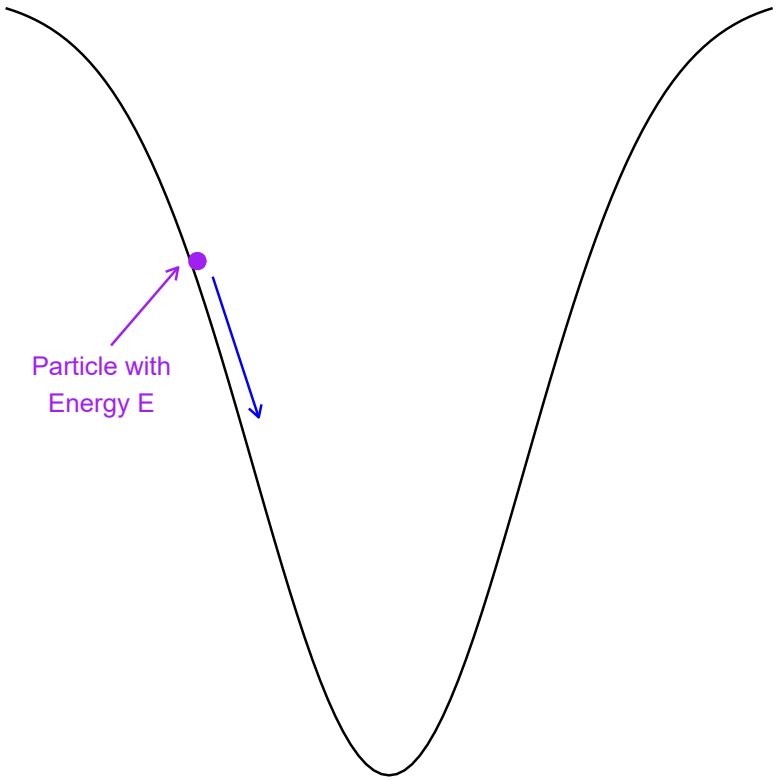
Consider a 1D posterior distribution (or log posterior), which may look something like this

Hamiltonian MCMC



Add a minus sign...

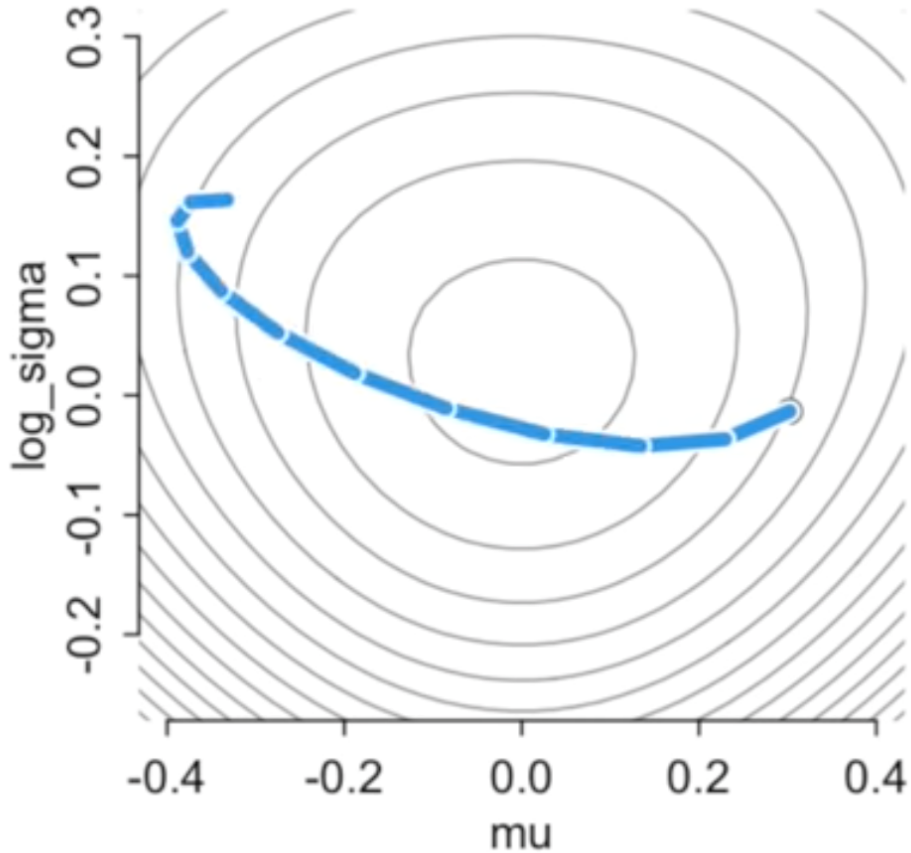
Hamiltonian MCMC



Hamilton's equations tell us how a particle with a certain energy would move in a 1D potential with this shape.

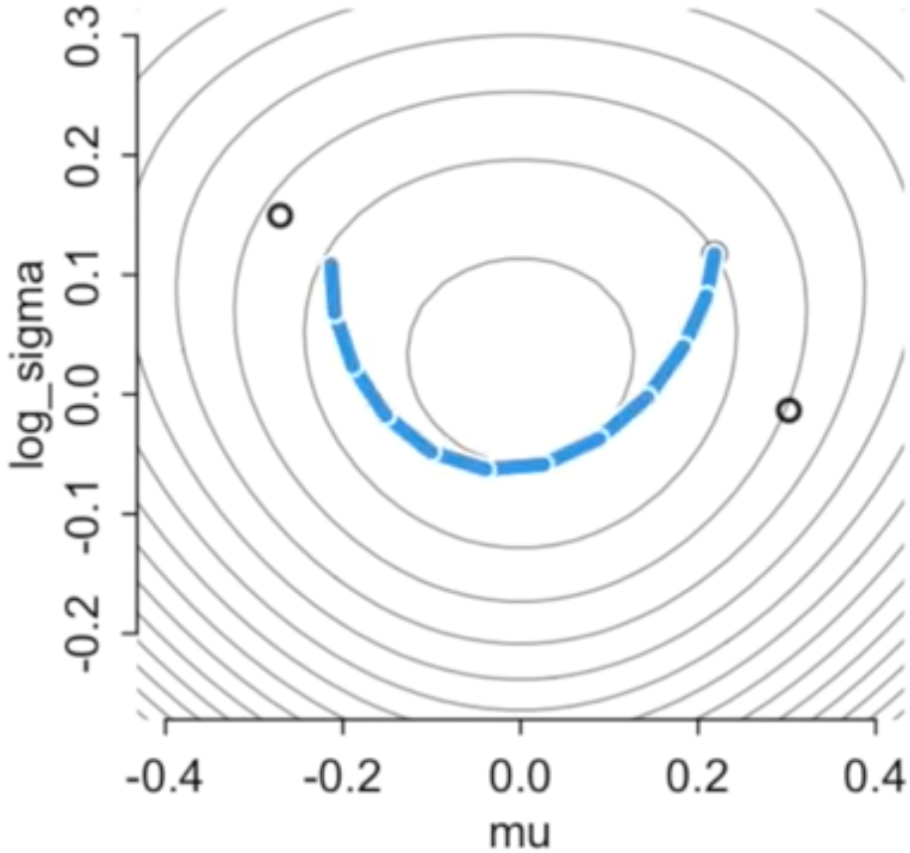
The very technical stuff: `https://arxiv.org/abs/1701.02434`

HMCMC trajectories



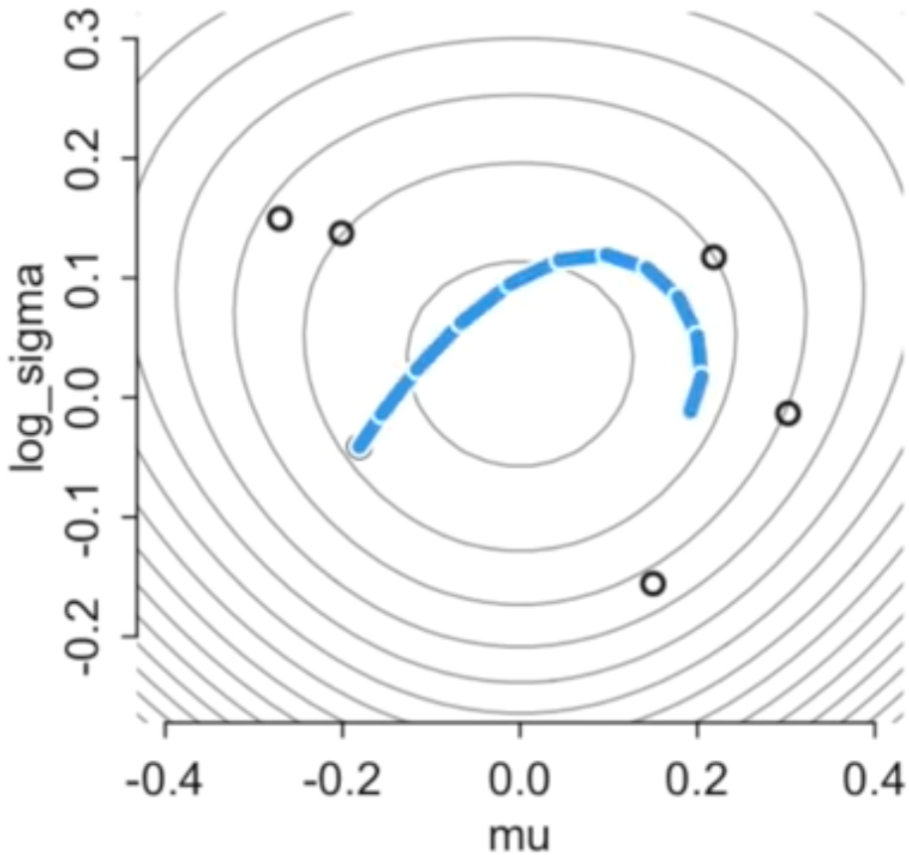
Images taken from Richard McElreath's course:
`https://youtu.be/Qqz5AJjyugM`

HMCMC trajectories



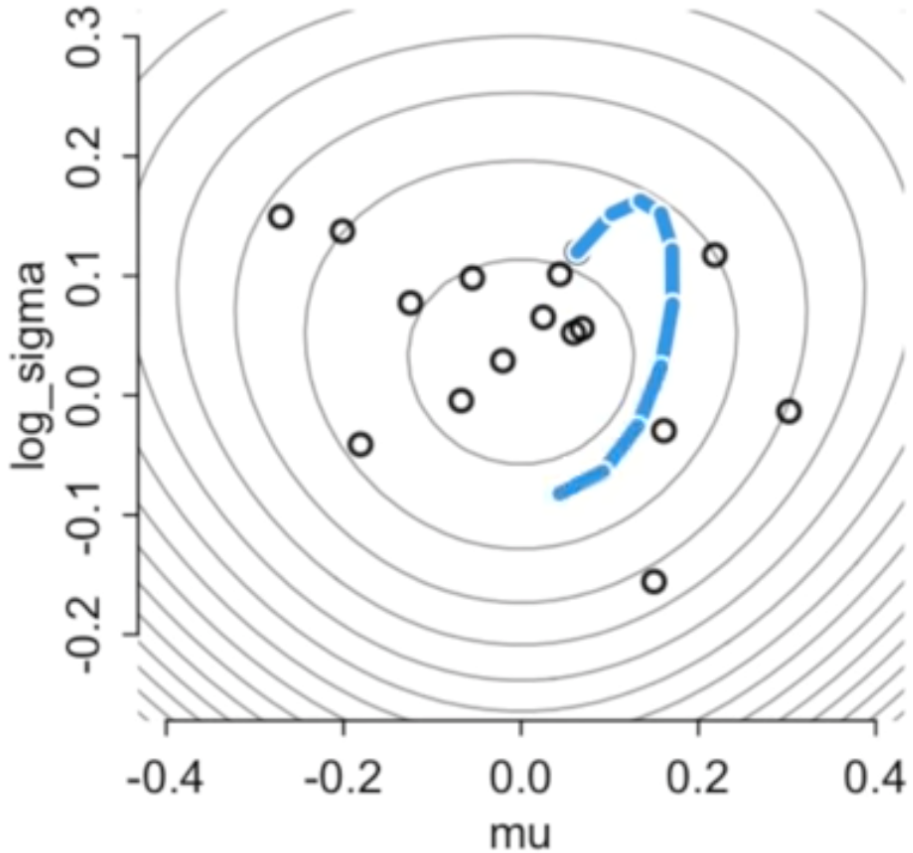
Images taken from Richard McElreath's course:
`https://youtu.be/Qqz5AJjyugM`

HMCMC trajectories



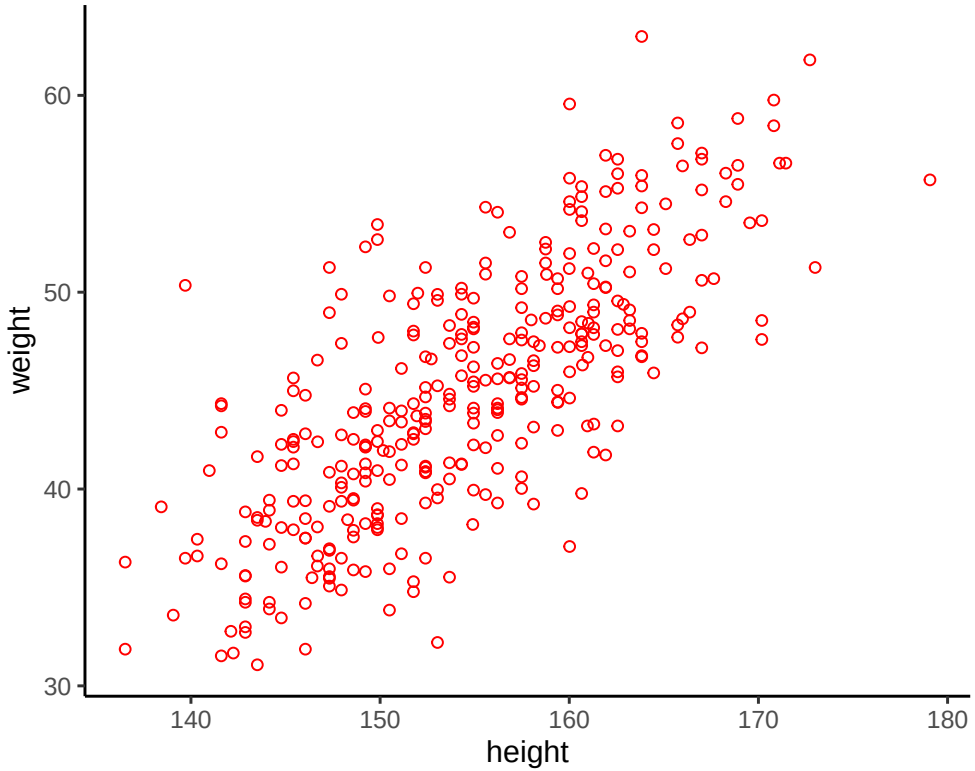
Images taken from Richard McElreath's course:
`https://youtu.be/Qqz5AJjyugM`

HMCMC trajectories



Images taken from Richard McElreath's course:
`https://youtu.be/Qqz5AJjyugM`

Example dataset




```
data("Howell1")
d <- Howell1[Howell1$age>=18,]
```

Let's fit a regression model to check the association between height & weight.

Writing the model

In R, you may have fitted a regression model like this: `lm(W~H)`.
What parameters does this model fit?

$$W_i = a + b * H_i + \epsilon_i$$

Here's how we could write the Bayesian model:

$$\begin{aligned} W_i &\sim \text{Normal}(\mu_i, \sigma) \\ \mu_i &= a + b(H - \bar{H}) \\ a &\sim \text{Normal}(30, 15) \\ b &\sim \text{Normal}(0, 5) \\ \sigma &\sim \text{Exp}(0.5) \end{aligned}$$

Now let's introduce the model in Stan code

Writing the model

Reminder of Bayes theorem:

$$\Pr(\theta|D) \propto \Pr(D|\theta)\Pr(\theta)$$

In our example:

$$\Pr(a, b, \sigma|H, W) \propto \Pr(H, W|a, b, \sigma)\Pr(a, b, \sigma)$$

Question: could we do a grid approximation to estimate $\Pr(a, b, \sigma|H, W)$?

Writing the model

To write a Stan model, we can either (i) define it within an R script or (ii) Write it in a separate `.stan` file. Either way, we define a stan model in code *blocks*:

- **Data**: fully specify the data (already loaded in R) to be used
- Functions: user defined functions
- **Parameters**: Parameters to be fitted
- Transformed Parameters
- **Model**: Define the model likelihood & priors
- Generalised Quantities: Anything we want to generate using the fitted parameters. For example, pointwise likelihoods for model comparison (*e.g.* WAIC, LOO-CV)

Model blocks

```
data {
  int<lower=0> N;
  vector[N] H;
  vector[N] W;
  real Hbar;
}
parameters {
  real a;
  real b;
  real<lower=0> sigma;
}
model {
  a ~ normal(0,5);
  b ~ normal(0,5); //lognormal(0,1);
  sigma ~ exponential(1);
  W ~ normal(a + b*(H - Hbar), sigma);
  //equivalently
  //for (i in 1:N) {
  //  W[i] ~ normal(a + b*(H[i] - Hbar), sigma),
  //}
}
```

Running Stan model

Save the data you need to fit the model as a list:

```
dat <- list(H = d$height, W = d$weight, Hbar =
mean(d$height), N = length(d$height))
```

```
fit = stan('stan_model1.stan', data = dat, iter =
3000, chains = 3)
```

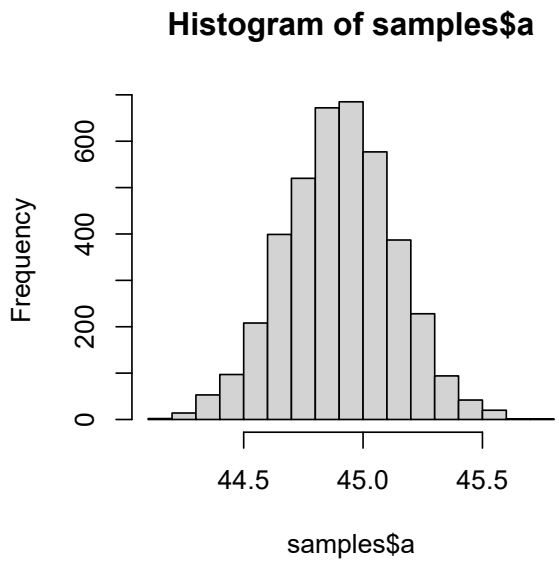
Does the model run?

Interpret & use the model output

In the R script, I've included some useful functions to inspect the fitted posterior distribution. These include: `print()`, `plot()`, `traceplot()`, `pairs()`.

We can also extract the posterior samples, like this:

```
samples <- extract(fit)
hist(samples$a)
```

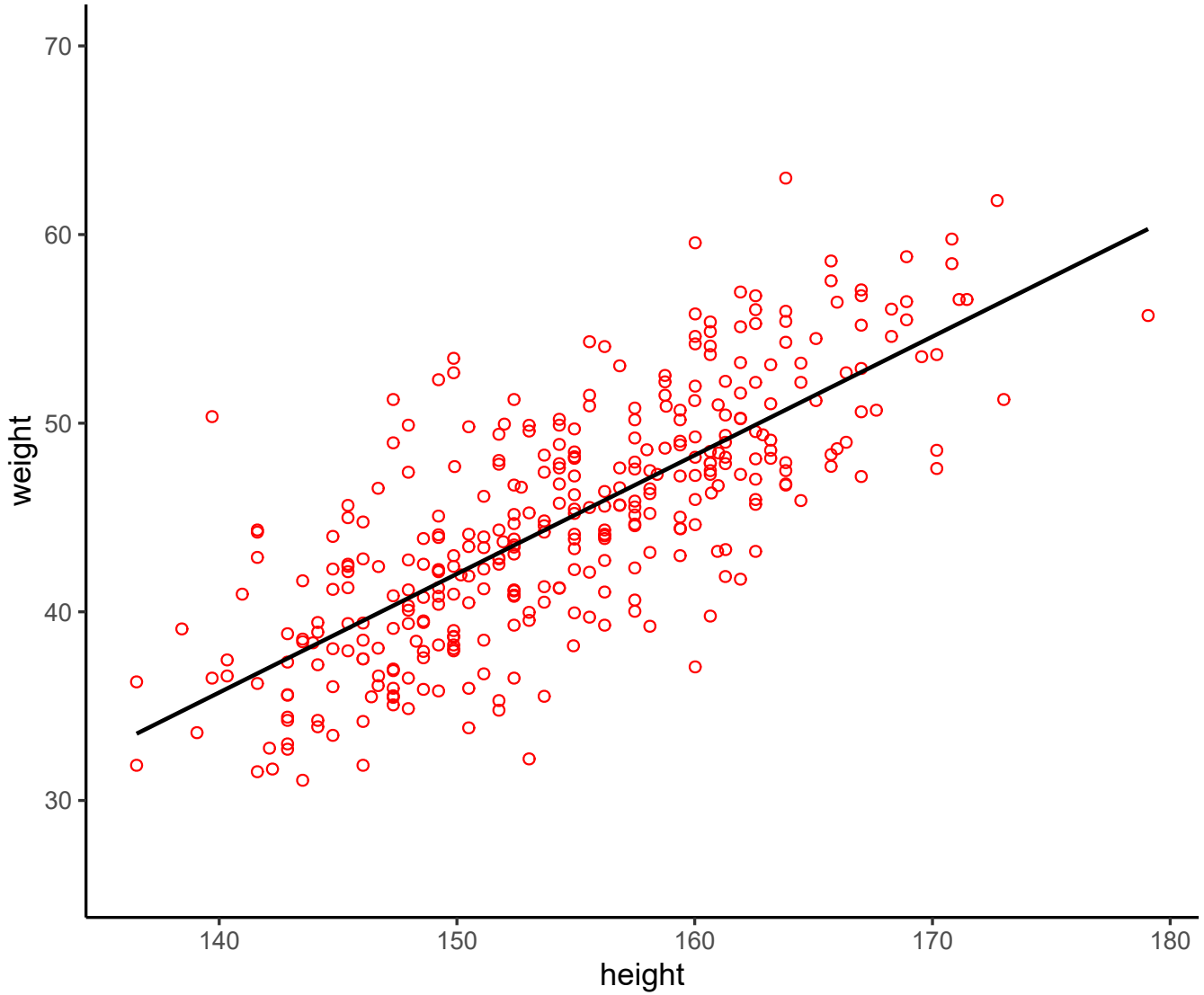


Compare model to data

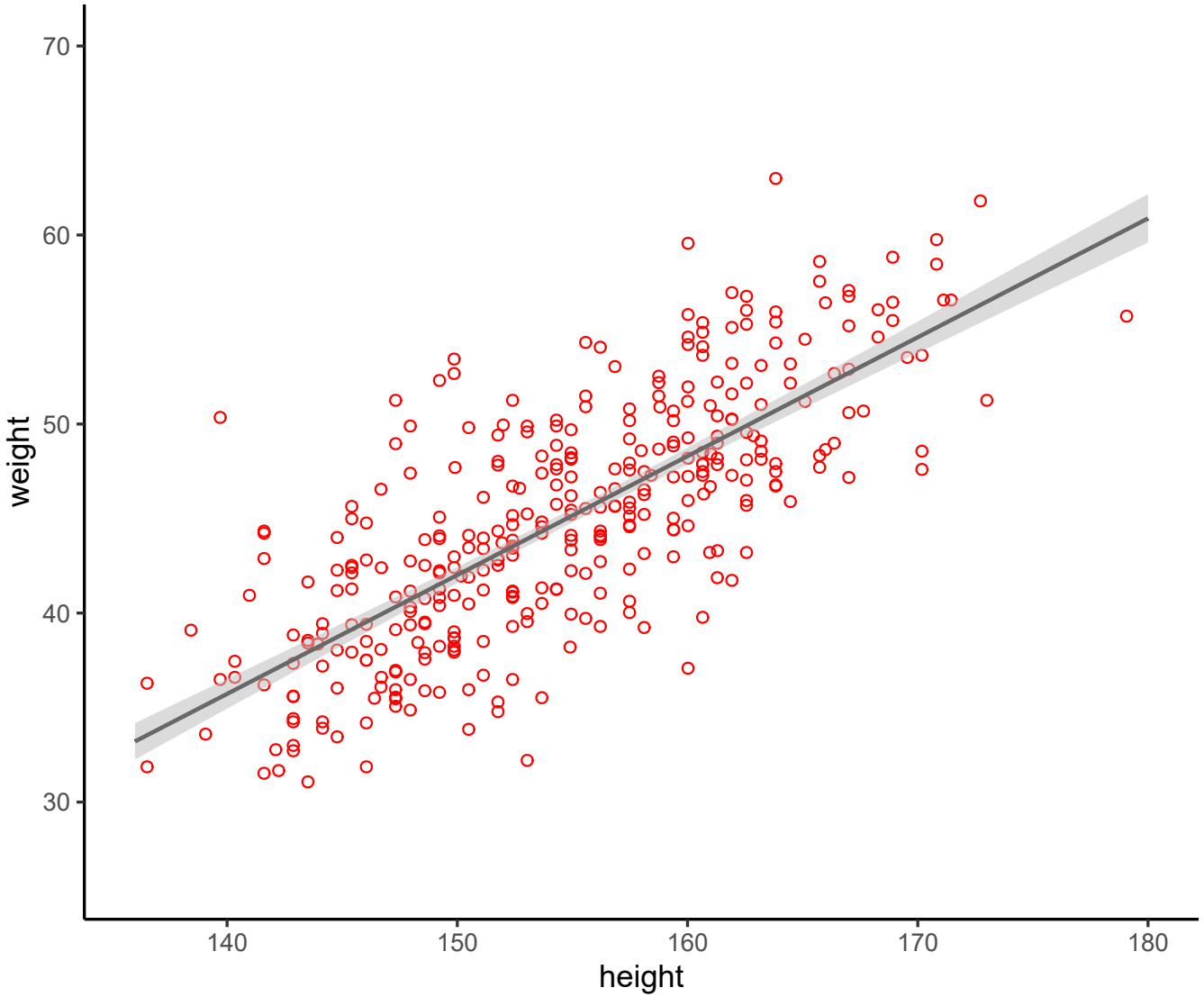
We'll look at three things that we might want to with the posterior distribution:

- Plot the straight line generated by the mean values of a and b
- Use the uncertainty in a and b to generate a credible interval for this line
- Simulate synthetic data using the posterior distribution

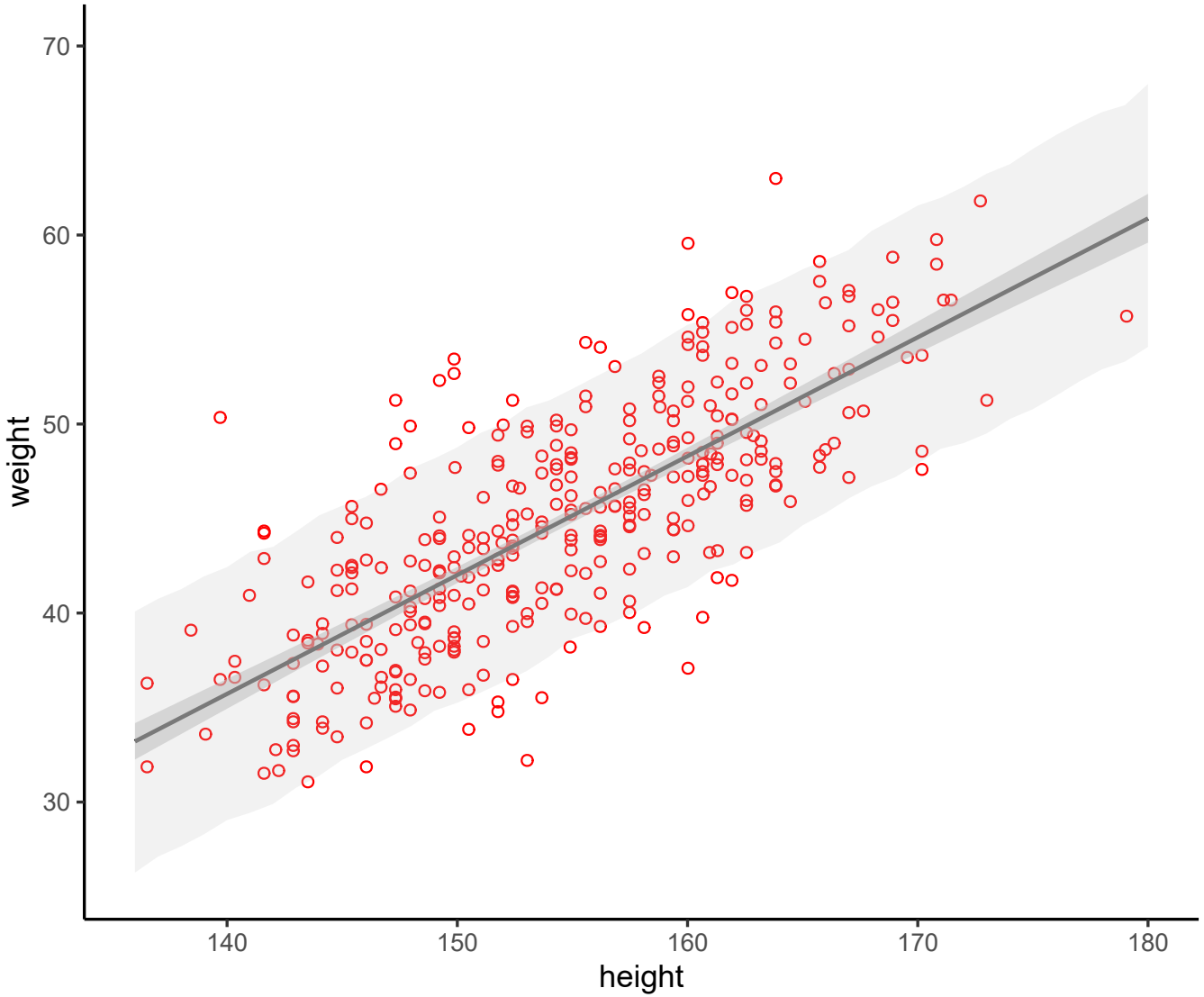
Compare model to data (i)



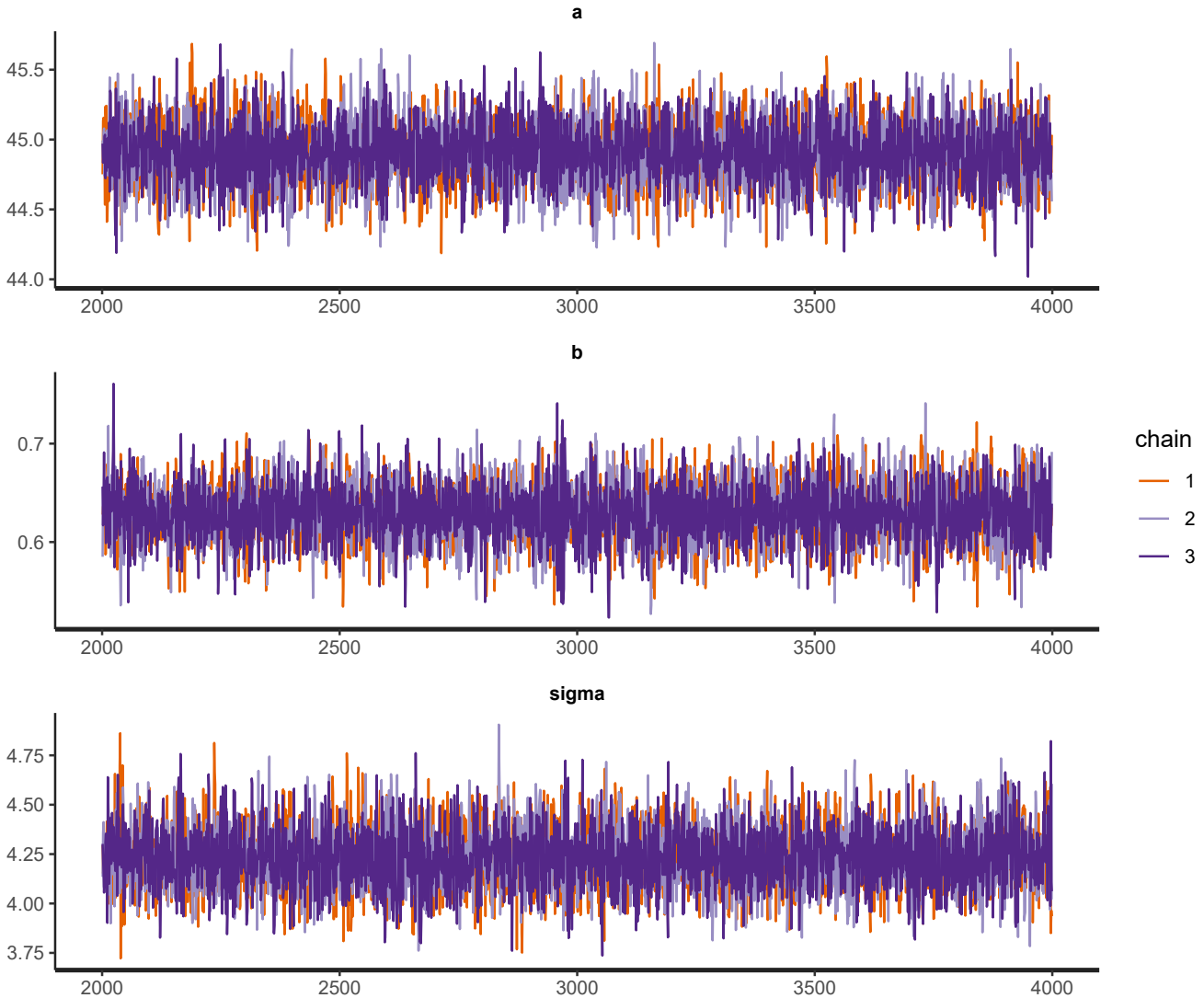
Compare model to data (ii)



Compare model to data (iii)



Chain diagnostics



Chain diagnostics

It is good practice to run multiple chains for a given model.

```
> print(fit)
Inference for Stan model: stan_model1-202301111007-1-55e7c6.
3 chains, each with iter=4000; warmup=2000; thin=1;
post-warmup draws per chain=2000, total post-warmup draws=6000.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
a       44.90    0.00 0.23   44.45   44.75   44.90   45.06   45.35  5789    1
b        0.63    0.00 0.03    0.57    0.61    0.63    0.65    0.69  5247    1
sigma    4.23    0.00 0.16    3.93    4.12    4.23    4.34    4.56  6117    1
lp__  -728.37    0.03 1.27 -731.72 -728.94 -728.02 -727.44 -726.92  2281    1

Samples were drawn using NUTS(diag_e) at Wed Jan 11 10:07:36 2023.
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).
```

- $\mathrm{n}_{\mathrm{eff}}$: For each parameter, this is the equivalent number of *uncorrelated* samples obtained
- Rhat is a ratio of variance (within-chain variance and total variance, across all the chains). Rhat $\rightarrow 1$ as total variance shrinks to average variance within chains.
- Stan may sometimes warn you about *divergent transitions*. This means simulation has diverged from the true path.

Another regression model

Let's look at a logistic regression model. We'll use data collected from a meta-analysis of clinical trials for a antimalarial therapy (see Mumtaz et al., Malaria Journal [2020]). I've included a minimal dataset as an .RDS file

```
> d2 <- readRDS('minimal_data.rds')
> str(d2)
'data.frame':   82 obs. of  3 variables:
 $ total_failures: int  1 3 4 1 3 1 2 1 4 1 ...
 $ LPF           : int  0 3 4 0 3 1 0 1 4 1 ...
 $ LCF           : int  1 0 0 1 0 0 2 0 0 0 ...
> head(d2)
  total_failures LPF LCF
1              1   0   1
2              3   3   0
3              4   4   0
4              1   0   1
5              3   3   0
6              1   1   0
```

Each row is a clinical trial, indicating how many treatment failures occured, and whether they were symptomatic (**L**ate **C**linical **F**ailure) or not (**L**ate **P**arasitological **F**ailure).

A logistic regression model

What proportion (p) of patients experienced symptomatic treatment failure? Here I've written N_i to indicate the total number of treatment failures in trial i.

$$
\begin{aligned}
LCF_i &\sim \text{Binomial}(N_i, p_i) \\
\text{logit}(p_i) &= \alpha \\
\alpha &\sim \text{Normal}(0, 2)
\end{aligned}
$$

A logistic regression model

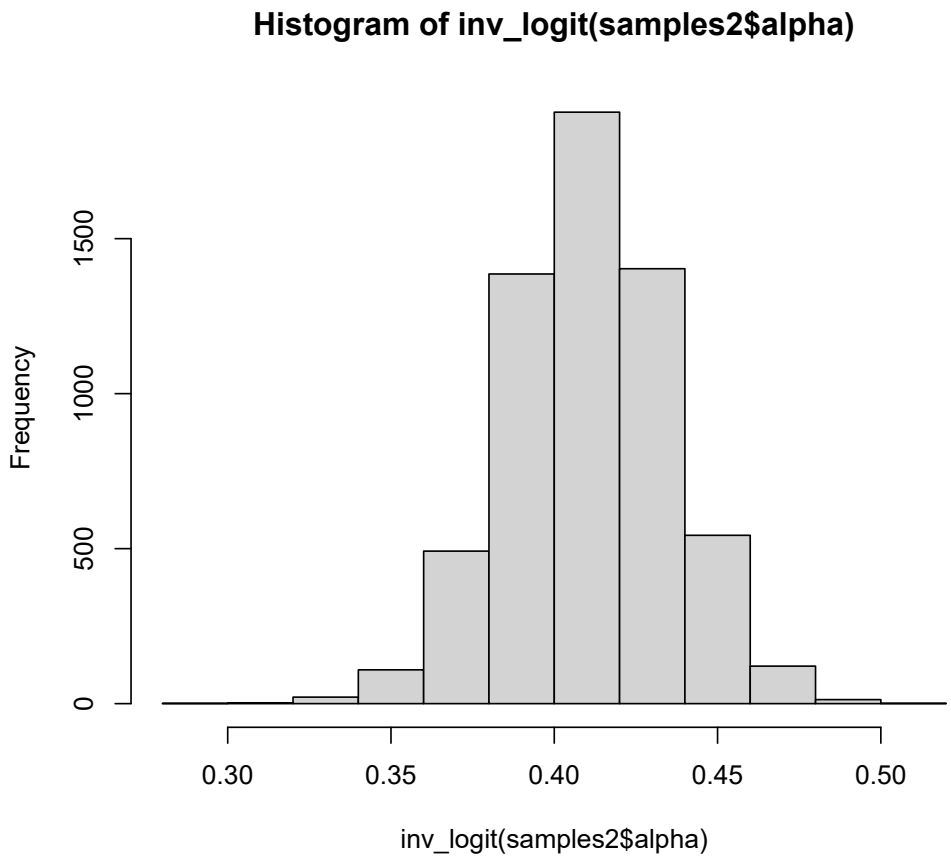
```
data {
  int<lower=0> N;
  array[N] int<lower=0> LCF;
  array[N] int<lower=0> tf;
}
parameters {
  real alpha;
}
model {
  real p;
  alpha ~ normal(0,2);
  p = inv_logit(alpha);
  LCF ~ binomial(tf, p);
}

```

```
dat2 <- list(tf = d2$total_failures, LCF = d2$LCF, N = length(d2$LCF))
fit2 = stan('stan_model3.stan', data = dat2, iter = 4000, chains = 3)
print(fit2)
```

A logistic regression model

Note that Stan has returned posterior samples for α. Generally speaking, we are more interested in the results for the probability, p. We can convert using the inverse of logistic function.



The Rethinking package

A simple way to get used to Stan is through the `rethinking` package, which accompanies Richard McElreath's textbook *Statistical Rethinking*. This provides a simpler interface to Stan, whilst still fitting the same models in the same way. I've included some code to demonstrate how to rewrite the Stan models we've already seen, using the `ulam()` function.

Specify model in Stan

```
data {
  int<lower=0> N;
  vector[N] H;
  vector[N] W;
  real Hbar;
}
parameters {
  real a;
  real b;
  real<lower=0> sigma;
}
model {
  a ~ normal(0,5);
  b ~ normal(0,5); //lognormal(0,1);
  sigma ~ exponential(1);
  W ~ normal(a + b*(H - Hbar), sigma);
  //equivalently
  //for (i in 1:N) {
  //  W[i] ~ normal(a + b*(H[i] - Hbar), sigma),
  //}
}
```

Specify model in Rethinking

```
library(ggplot2)
library(rstan)
library(rethinking)

#'dat' should be loaded loaded in the tutorial script

m1 <- ulam(
  alist(
    W ~ dnorm(mu, sigma),
    mu <- alpha + beta*(H-Hbar),
    alpha ~ dnorm(0,5),
    beta ~ dnorm(0,5),
    sigma ~ dexp(1)
  ),
  data = dat, chains = 2, cores = 2, iter = 4000)
stancode(m1)
precis(m1)
```

Rethinking also includes useful functions for analysing the fitted model *e.g.* `link()` and `sim()`.

Other things you can do in Stan

Today we've barely scratched the surface. Things we haven't covered include:

- Mixed-effects (hierarchical) models
- Using non-standard probability distributions (e.g. zero-inflated models, or finite mixtures)
- Ordinary differential equation models (but can be slow in my experience)
- Solve algebraic equations by numerical methods
- Gaussian process models
- ...
- Probably lots of other things!

Some resources

- Today we've seen how to manually extract posterior samples from the fitted model and do things with them. There are packages that can help you do this, such as `tidybayes` or `bayesplot`
- If you're looking for a textbook, Richard McElreath's *Statistical Rethinking* is pretty good. His lecture courses are also on Youtube: `https://www.youtube.com/@rmcelreath`
- `The Stan user guide is pretty comprehensive (https://mc-stan.org/docs/stan-users-guide/), there is a forum on this website too.`

Thank you for attending!

Any questions?