Stan Tutorial

STA 540 Case Study

Bo Liu (October 6, 2021)

Outline for Section 1

1. Introduction to Stan
   - 1.1 Getting started
   - 1.2 Structure of a Stan program
   - 1.3 Integration with R
2. Basic modeling
   - 2.1 Ordinary Linear Regression
   - 2.2 Generalized Linear Regression
   - 2.3 Hierarchical Models
3. Advanced topics
   - 3.1 Log probability `lp`
   - 3.2 Latent variable models
   - 3.3 Auxiliary sampling

Loading Stan in R

```
library(rstan)

## Loading required package: StanHeaders
## rstan (Version 2.21.1, GitRev: 2e1f913d3ca3)
## For execution on a local, multicore CPU with
excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs,
we recommend calling
## rstan_options(auto_write = TRUE)
##
## Attaching package: 'rstan'
## The following object is masked from
'package:tidyr':
```

Stan program structure

A stan program involves the following blocks.

- functions
- data
- transformed data
- parameters
- transformed parameters
- model
- generated quantities

The blocks should be arranged in this given order.

Program block: `data`

The `data` block is for the declaration of variables that are read in as data.

Note that the data is only read in once.

Grammar:

```
[basic type]<range?> identifier[size];
```

```
data {
  int<lower=0> J;
  real y[J];
  real<lower=0> sigma[J];
}
```

Program block: `parameters`

The variables declared in the `parameters` program block correspond directly to the variables being sampled by Stan.

The parameters defined in this block must be continuous.

Grammar:

```
[basic type]<range?> identifier[size];
```

```
parameters {
  real mu;
  real<lower=0> tau;
  real eta[J];
}
```

Program block: `model`

The model program block consists of optional variable declarations followed by statements. The variables in the model block are local variables and are not written as part of the output.

Grammar:

```
identifier ~ distribution(args);
```

```
model {
  eta ~ normal(0, 1);
  y ~ normal(theta, sigma);
}
```

Try it out in R

Stan code: `8schools.stan`

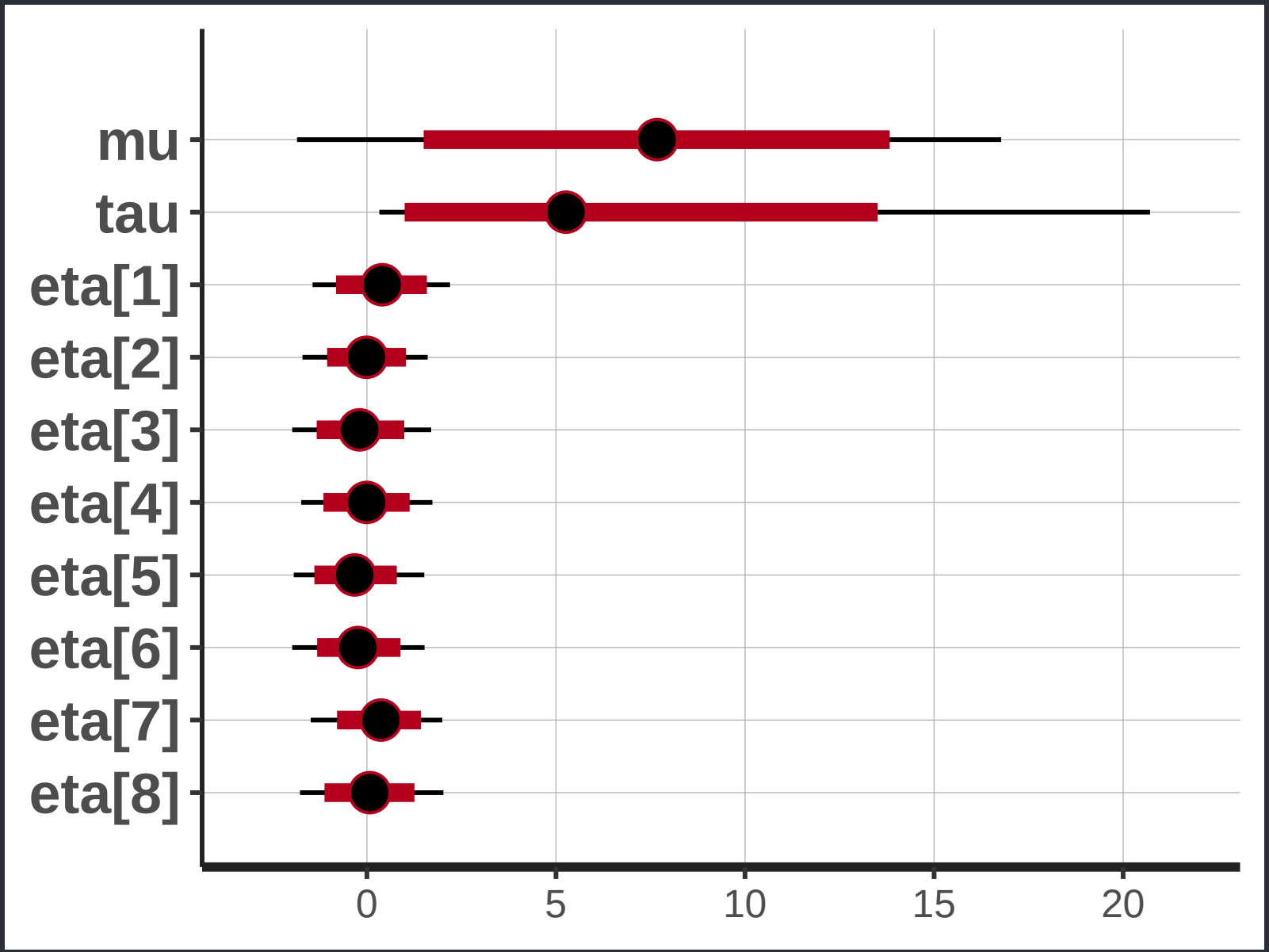
```
data {
  int<lower=0> J;
  // number of schools
  real y[J];
  // estimated treatment
       effects
  real<lower=0> sigma[J];
  // s.e. of effect estimates
}
parameters {
  real mu;
  real<lower=0> tau;
  real eta[J];
}
transformed parameters {
  real theta[J];
  for (j in 1:J)
    theta[j] = mu + tau * eta[j
          ];
}
model {
  eta ~ normal(0, 1);
  y ~ normal(theta, sigma);
}
```

Try it out in R

R code:

```
set.seed(0)
schools_dat <- list(
  J = 8,
  y = c(28, 8, -3, 7, -1, 1, 18, 12),
  sigma = c(15, 10, 16, 11, 9, 11, 10, 18)
)
fit <- stan(
  file = "8schools.stan", data = schools_dat,
  iter = 1000, warmup = 100, chains = 4,
  [verbose = F, refresh = 0]
)
```

Try it out in R



Pre-compiling stan code

If the same model is to be fit on multiple datasets, it would be more efficient to pre-compile the stan model.
We can separate the compilation process from the sampling process.

- Compilation:

```
stan_m <- stan_model(file = ...)
```

- Sampling:

```
fit <- sampling(stan_m, data = ...)
```

Pre-compiling stan code

If you are using RMarkdown, you can also add a stan chunk directly.

````
```{stan output.var = "model"}
  // some stan code
```
````

Behind the scenes, the RMarkdown chunk calls

```
model <- stan_model(model_code = the_code_text).
```

Therefore, you can do sampling in the next ordinary R chunk.

````
```{r}
  sampling(model, data = ...)
```
````

Outline for Section 2

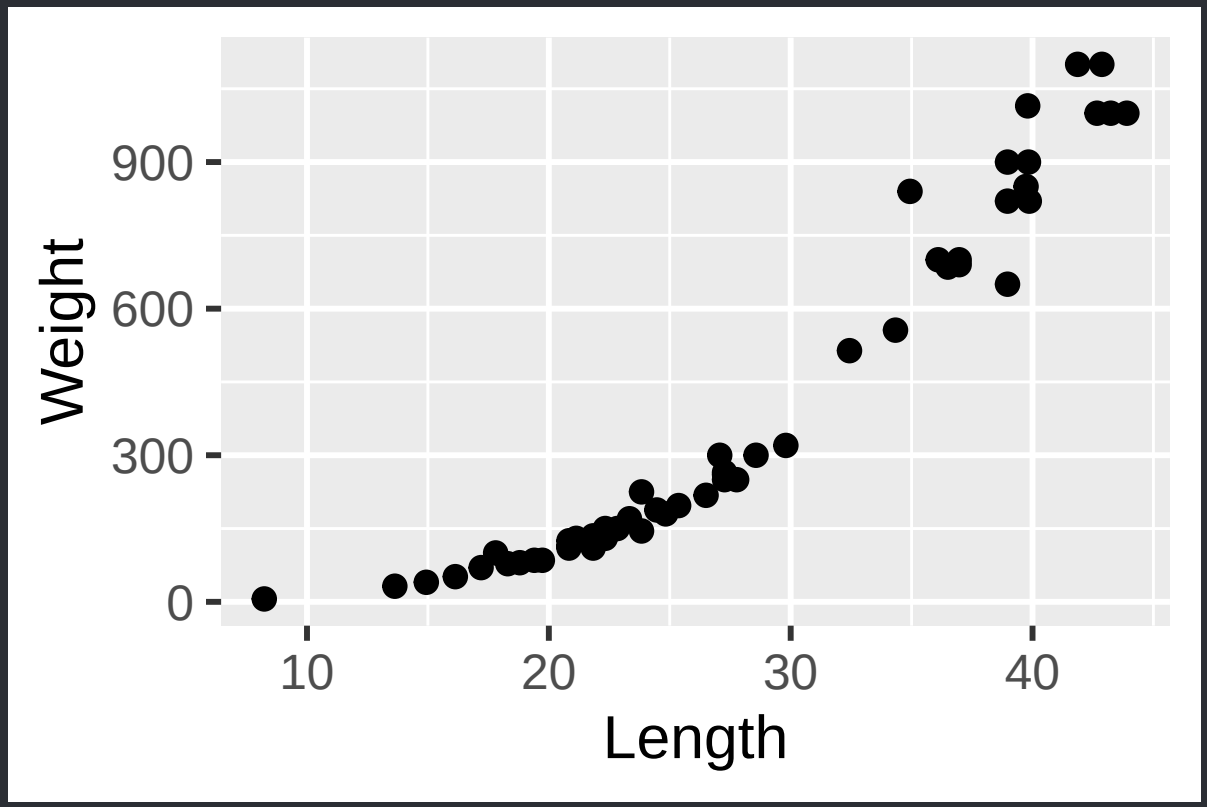
1. Introduction to Stan
   - 1.1 Getting started
   - 1.2 Structure of a Stan program
   - 1.3 Integration with R
2. Basic modeling
   - 2.1 Ordinary Linear Regression
   - 2.2 Generalized Linear Regression
   - 2.3 Hierarchical Models
3. Advanced topics
   - 3.1 Log probability `lp`
   - 3.2 Latent variable models
   - 3.3 Auxiliary sampling

Ordinary Linear Regression

The file `Fish.csv` contains the weight and body measurements of fish from seven species. Suppose we are interested in building a linear model to explain the association between weight and body measurements for perches.

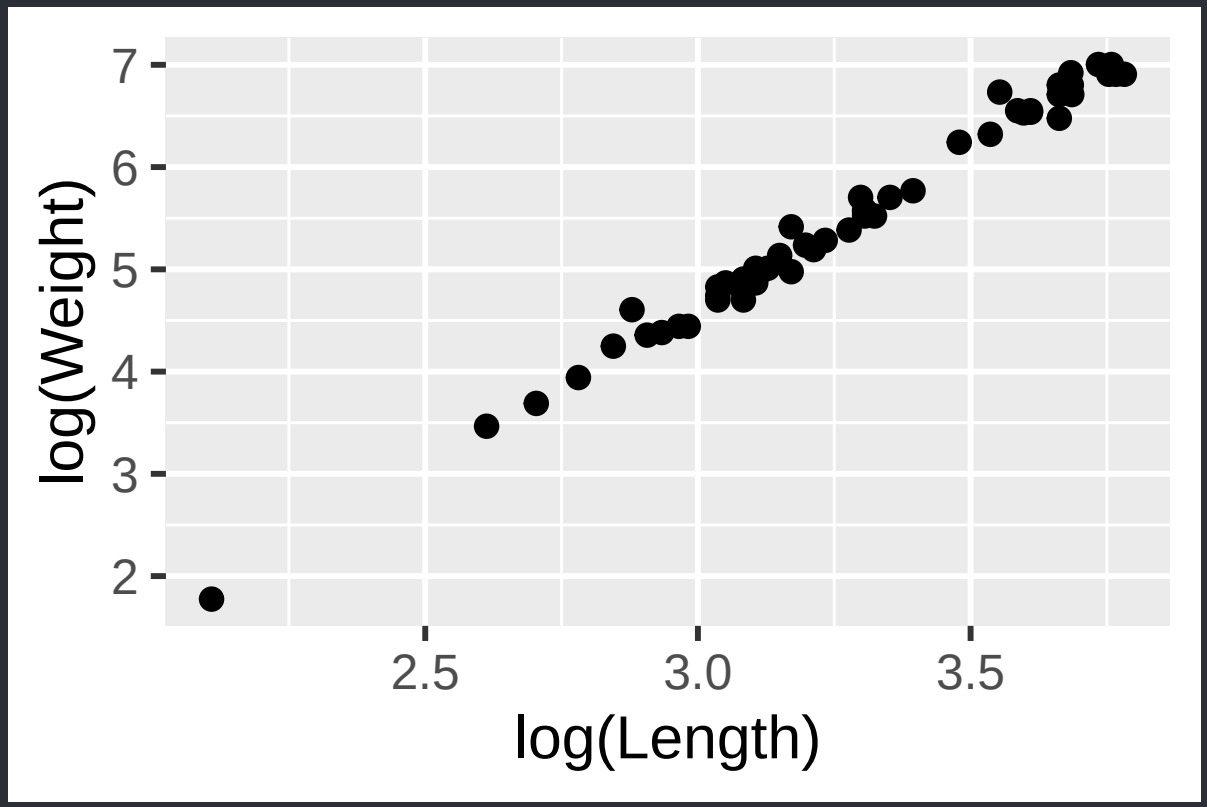
Ordinary Linear Regression

For now, we only use one predictor: the average of three length measurements.



Ordinary Linear Regression

It looks like a log transform should be appropriate.



Thus, we use the following model.

$$\log(y_i) \sim \mathcal{N}(\alpha + \beta \log(x_i), \sigma^2).$$

Ordinary Linear Regression

Stan model:

```
data {
  int<lower=0> N;
  real y[N];
  real x[N];
}

parameters {
  real alpha;
  real beta;
  real<lower=0> sigma_sq;
}
```

```
model {
  alpha ~ normal(0, 1);
  beta ~ normal(0, 1);
  sigma_sq ~ inv_gamma(0.1,
      0.1);
  for (n in 1:N) {
    y[n] ~ normal(
      alpha + beta * x[n],
      sqrt(sigma_sq)
    );
  }
}
```

Any improvements?

Ordinary Linear Regression

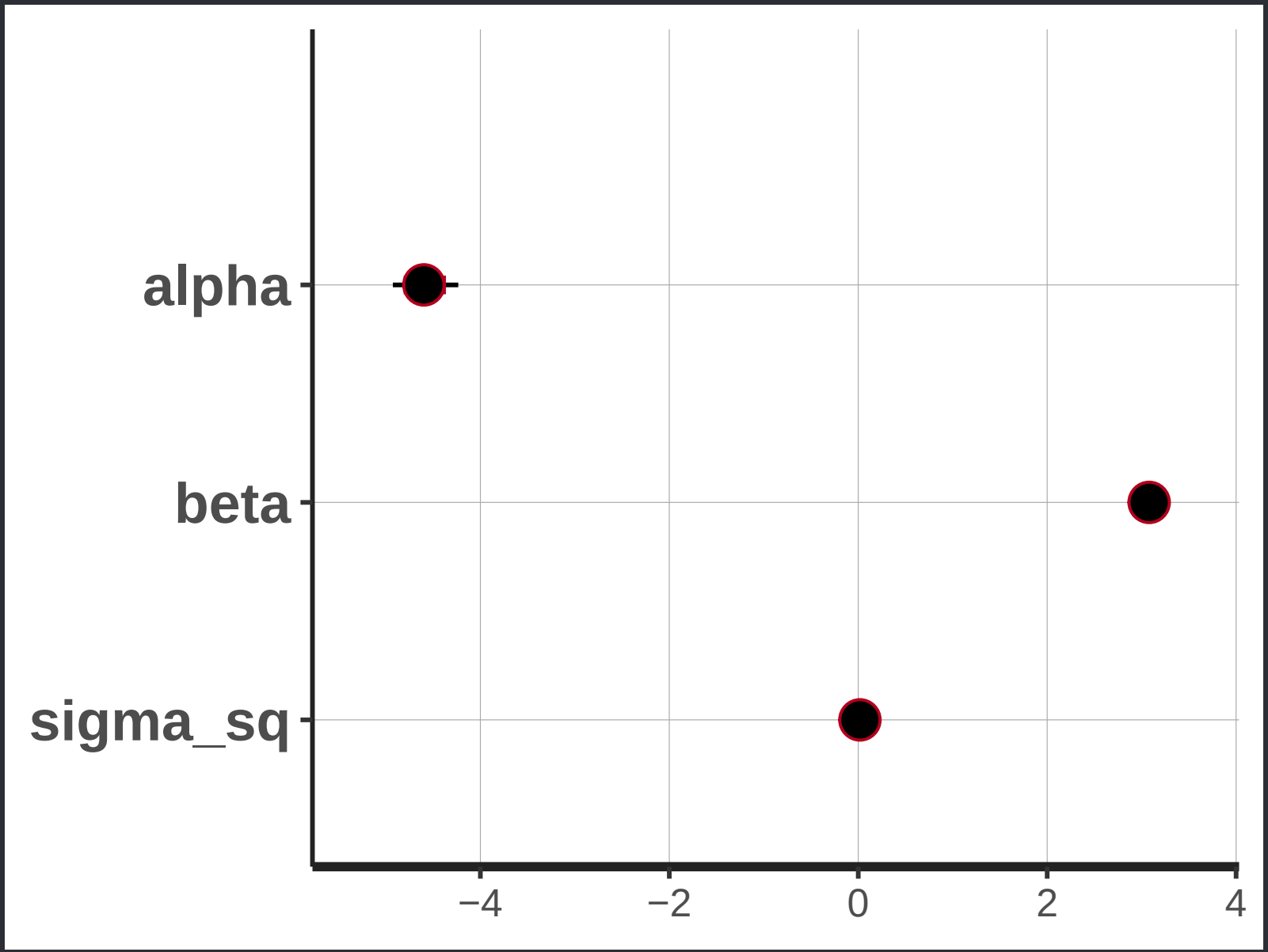
More efficient:

```
data {
  int<lower=0> N;
  real y[N];
  real x[N];
}

parameters {
  real alpha;
  real beta;
  real<lower=0> sigma_sq;
}

model {
  alpha ~ normal(0, 1);
  beta ~ normal(0, 1);
  sigma_sq ~ inv_gamma(0.1,
      0.1);
  y ~ normal(alpha + beta * x,
      sqrt(sigma_sq));
}
```

Ordinary Linear Regression



Ordinary Linear Regression

Now let's use all predictors

```
data {
  int<lower=0> N;
  int<lower=0> P;
  real y[N];
  matrix[N, P] X;
}

parameters {
  real alpha;
  real beta[P];
  real<lower=0> sigma_sq;
}

model {
  alpha ~ N(0, 1);
  beta ~ N(0, 1);
  sigma_sq ~ inv_gamma(0.1,
       0.1);
  y ~ normal(alpha + X * beta,
       sqrt(sigma_sq));
}
```

Generalized Linear Regression

- `Advertisement.csv` contains data of people's gender, age, estimated salary and whether they purchased the advertised product. The aim is to build a Bayesian model to predict the probability of purchase.

Generalized Linear Regression

- `Advertisement.csv` contains data of people's gender, age, estimated salary and whether they purchased the advertised product. The aim is to build a Bayesian model to predict the probability of purchase.
- `awards.csv` contains data of the number of awards some students achieved in college, with the type of program they are in (numbered 1, 2, 3) and their math scores. Considering only the math scores, build a Bayesian model for predicting the number of awards.

Hierarchical Models

Back to the `Fish.csv` data. This time, we would like to consider building a model on weight for all seven species, but only using the average body measurement as predictor.

- Species-specific intercepts
- Species-specific slopes

Data: Y (weight), X (measurement), G (species). Model:

$$Y_{ig} \sim \mathcal{N}(\alpha_g + x_{ig}\beta_g, \sigma^2),$$

where $\alpha_g \sim \mathcal{N}(0, \tau_\alpha^2)$, $\beta_g \sim \mathcal{N}(0, \tau_\beta^2)$.

Hierarchical Models

```
data {
  int<lower=0> N;
  real Y[N];
  real X[N];
  int<lower=1, upper=7> G[N];
}

parameters {
  real alpha[7];
  real beta[7];
  real<lower=0> sigma;
  real<lower=0> tau_alpha;
  real<lower=0> tau_beta;
}

transformed parameters {
  vector[N] y_hat;
  for (i in 1:N) {
    y_hat[i] = alpha[G[i]] +
        beta[G[i]] * X[i];
  }
}

model {
  sigma ~ inv_gamma(0.1, 0.1);
  tau_alpha ~ inv_gamma(0.1,
      0.1);
  tau_beta ~ inv_gamma(0.1,
      0.1);
  alpha ~ normal(0, tau_alpha);
  beta ~ normal(0, tau_beta);
  y ~ normal(y_hat, sigma);
}
```

Outline for Section 3

1. Introduction to Stan
   - 1.1 Getting started
   - 1.2 Structure of a Stan program
   - 1.3 Integration with R
2. Basic modeling
   - 2.1 Ordinary Linear Regression
   - 2.2 Generalized Linear Regression
   - 2.3 Hierarchical Models
3. Advanced topics
   - 3.1 Log probability `lp`
   - 3.2 Latent variable models
   - 3.3 Auxiliary sampling

Log probability `lp`

- We've used (semi-)conjugate priors in order to sample from posterior distributions with ease.
- For Stan, there is no need to look for conjugacy.
- A posterior distribution is nothing other than its density function (recall that Stan only accepts continuous parameters), whose logarithm is often used instead in practice to avoid numerical issues.
- With this log posterior density function, Stan can sample from the posterior distribution using methods including HMC.

Log probability `lp`

- We have learned that

$$p(\theta \mid y) \propto p(\theta)p(y \mid \theta),$$

  which is equivalent to

$$\log p(\theta \mid y) = \log p(\theta) + \log p(y \mid \theta),$$

  up to an additive constant.
- Therefore, `lp` keeps updating itself with the sampling statements in the `model` block.

Log probability `lp`

- Technically speaking, there is no real need to distinguish between statements for the prior distribution and for the conditional distribution.
- Both kinds of statements have the same effect of adding to `lp` the corresponding log probability.
- The sampling statement is just a short-hand notation for updating `lp` - no real sampling is done in this step.

Log probability `lp`

- The sampling statement

```
alpha ~ N(0, 1);
```

  is equivalent to directly updating `lp` by

```
target += normal_lpdf(alpha | 0, 1);
```

  except that the sampling statement drops the normalizing constant.

Log probability `lp`

Is it okay if one does not write any sampling statements on some parameters? If so, what prior is put on those parameters?

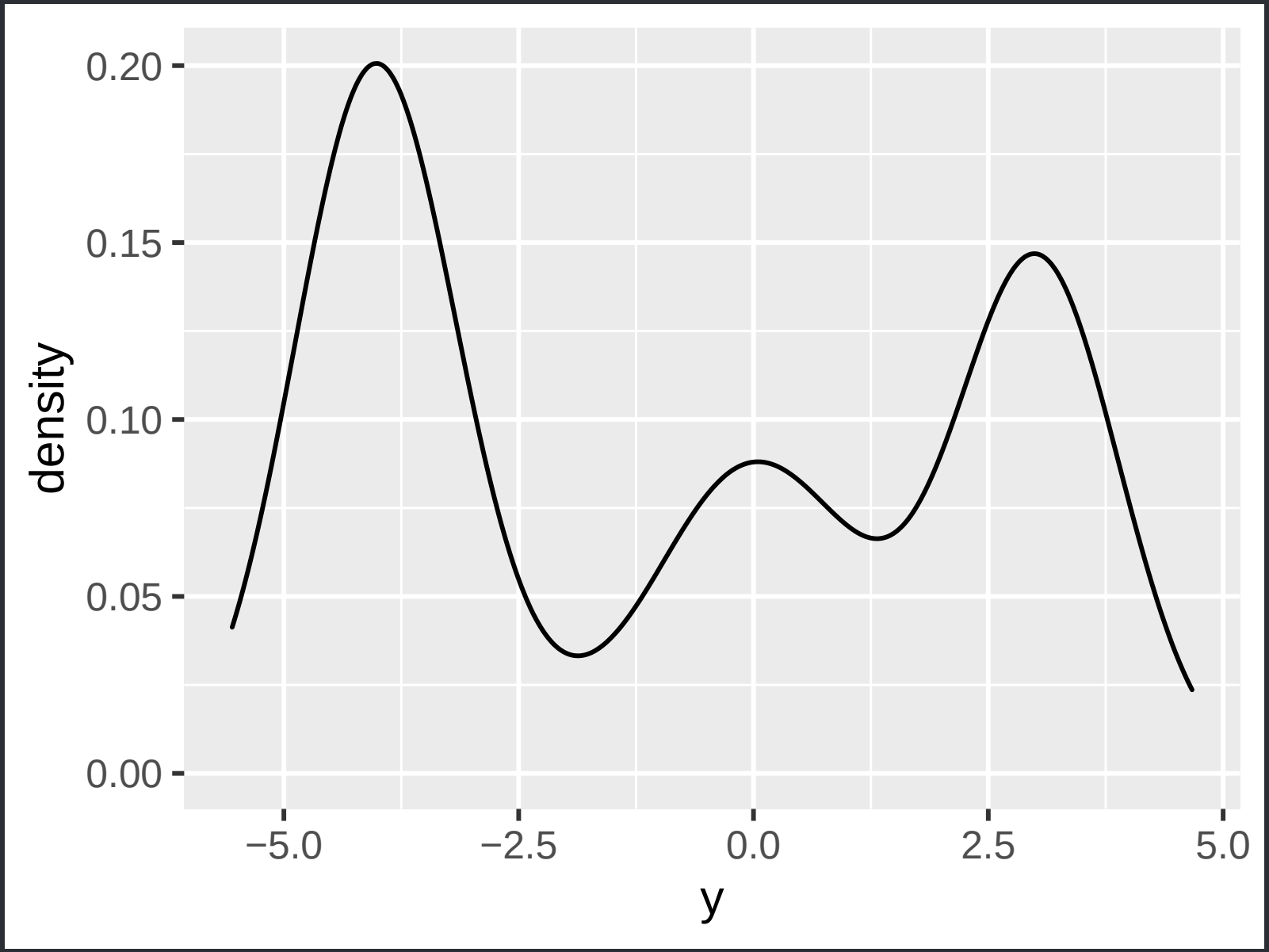
Latent variable models

In this part, we use a simulated dataset where there are three groups of data points. Within group j, the outcome follows a normal distribution

$$Y_i \sim \mathcal{N}(\alpha_j, \sigma_j^2).$$

Unless in a hierarchical model, we do not observe the group each data point belongs to.

Latent variable models



Latent variable models

- Data: the outcome Y.
- Parameters: α_j, σ_j, and the group assignment g_i for data point i.
- Conditional distribution:

$$Y_i \mid \alpha_j, \sigma_j, g_i \sim \mathcal{N}(\alpha_{g_i}, \sigma^2_{g_i}).$$

- Is there a problem to convert it into Stan code?

Latent variable models

- Data: the outcome Y.
- Parameters: α_j, σ_j, and the group assignment g_i for data point i.
- Conditional distribution:

$$Y_i \mid \alpha_j, \sigma_j, g_i \sim \mathcal{N}(\alpha_{g_i}, \sigma^2_{g_i}).$$

- Is there a problem to convert it into Stan code?
- Group assignment g_i is discrete!
- Solution: integrate (sum) out discrete parameters g_i.

Latent variable models

- Suppose that $\Pr(g_i = j) = \pi_j$.
- Then,

$$Y_i \mid \alpha_j, \sigma_j \sim \sum \pi_j \mathcal{N}(\alpha_j, \sigma_j^2).$$

- This is not a known distribution implemented in Stan, so we cannot use sampling statements for this conditional distribution.
- Instead, we have to update `lp` manually.
- How to deal with summation under logarithm? `log-sum-exp`.

$$\texttt{log-sum-exp}(u, v) = \texttt{log}(\texttt{exp}(u) + \texttt{exp}(v)).$$

Latent variable models

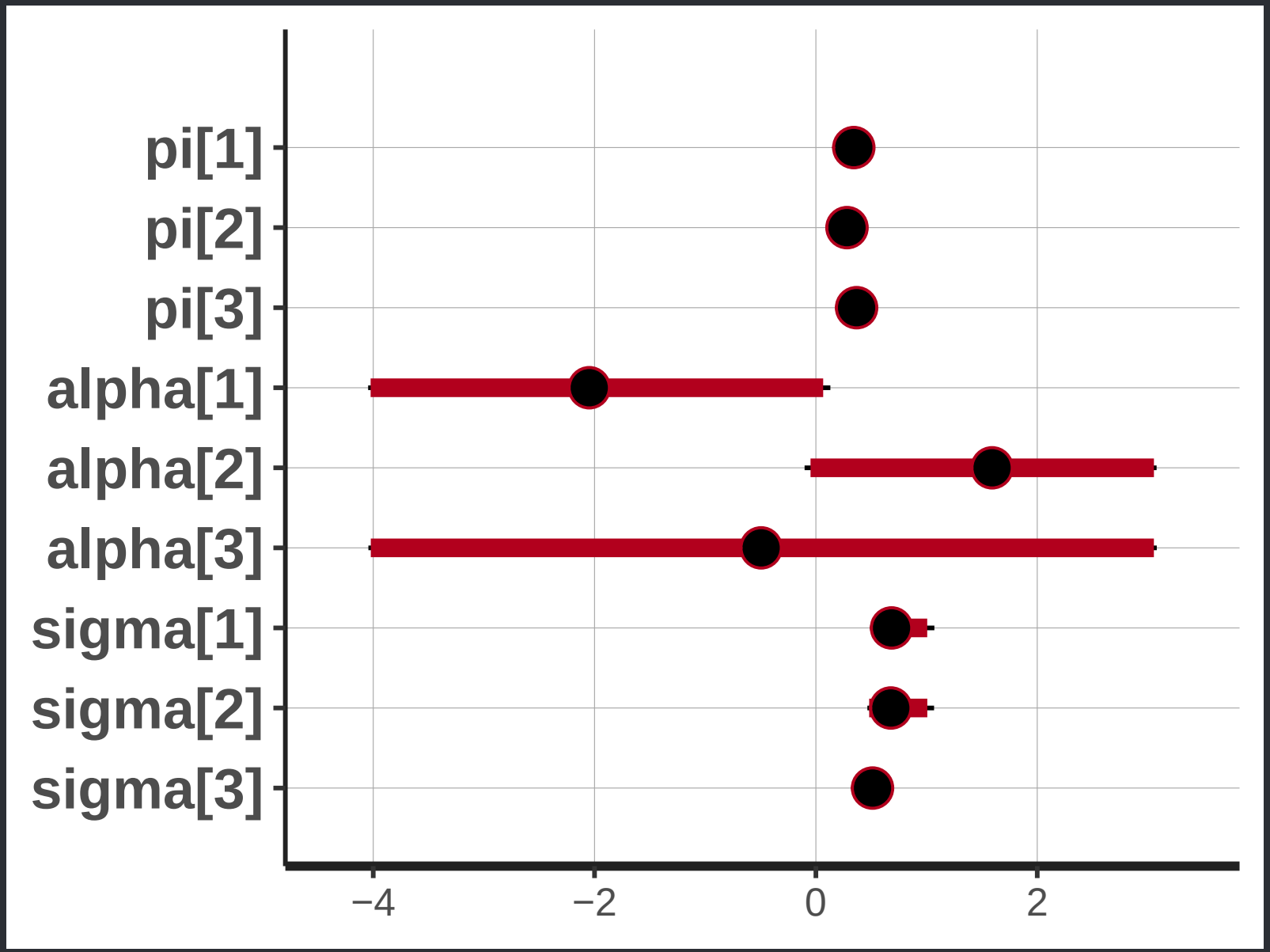
- $\pi = (\pi_1, \pi_2, \pi_3)$ is a parameter such that $\pi_j \in [0, 1]$, $\sum \pi_j = 1$.
- This is called a simplex.

```
simplex[3] pi;
```

- Updating `lp`:

```
vector[3] log_pi = log(pi);
for (i in 1:N) {
  vector[3] lps = log_pi;
  for (k in 1:3) {
    lps[k] += normal_lpdf(Y[i] | alpha[k], sigma[k]);
  }
  target += log_sum_exp(lps);
}
```

Latent variable models



Auxiliary sampling

Sometimes, we might want to generate samples that are not parameters themselves.

- In the latent variable model, we might want to sample which group each point belongs to.
- For predicting the outcome given a new data point, we may want to sample the potential outcome given the posterior samples of parameters.
- To carry out posterior predictive checks, we need to sample data under the sampled parameters.

Auxiliary sampling

In Stan, this can be done within the `generated quantities` block. For example, if one is interested in sampling the group each data point belongs to, one can sample from the multinomial distribution with posterior samples of π.

```
generated quantities {
  int<lower=0> group[N, 3];
  for (i in 1:N) {
    group[i] = multinomial_rng(pi, 1);
  }
}
```