Class 9: Bayesian Hierarchical Models

Andrew Parnell
andrew.parnell@mu.ie

Learning outcomes:

- Start fitting hierarchical GLMs in `rstanarm`
- Know some of the different versions of Hierarchical GLMs
- Be able to expand and summarise fitted models

From LMs to HGLMs

- The Bayesian analogue of a mixed model is a *hierarchical model*.
- It's called a hierarchical model because the prior distributions come in layers, they depend on other parameters
- Within this framework, we can borrow the ideas from the previous class to create hierarchical GLMs
- We will go through four examples: binomial-logit, Poisson, robust regression, and ordinal regression

Example 1: binomial-logit

- Earlier we met the Binomial-logit model for binary data:

$$y_i \sim Bin(1, p_i), logit(p_i) = \alpha + \beta(x_i - \bar{x})$$

Here $logit(p_i)$ is the link function equal to $\log\left(\frac{p_i}{1-p_i}\right)$ and transforms the bounded probabilities into an unbounded space

- If we have non-binary data we just change the likelihood:

$$y_i \sim Bin(N_i, p_i), logit(p_i) = \alpha + \beta(x_i - \bar{x})$$

- In a hierarchical version of this model, we vary the *latent parameters* α and β and give them prior distributions

The swiss willow tit data

```r
swt = read.csv('../data/swt.csv')
head(swt)
```

```
##   rep.1 rep.2 rep.3 c.2 c.3 elev forest dur.1 day.2 day.3 length alt
## 1     0     0     0   0   0  420      3   240    58    73    6.2 Low
## 2     0     0     0   0   0  450     21   160    39    62    5.1 Low
## 3     0     0     0   0   0 1050     32   120    47    74    4.3 Med
## 4     0     0     0   0   0 1110     35   180    44    71    5.4 Med
## 5     0     0     0   0   0  510      2   210    56    73    3.6 Low
## 6     0     0     0   0   0  630     60   150    56    73    6.1 Low
```

A hierarchical model

- Suppose we want to fit a model on the sum $y_i =$ `rep.1 + rep.2 + rep.3`:

$$y_i \sim Bin(N_i, p_i), logit(p_i) = \alpha_{\text{altitude}_i} + \beta_{\text{altitude}_i}(x_i - \bar{x})$$

where x_i is the percentage of forest cover

- What prior distributions should we use for α and β?
- Useful side note: A value of 10 on the logit scale leads to a probability of about 1, and a value of -10 leads to a probability of about 0 (you can test this by typing `inv.logit(10)`) so I wouldn't expect the value of $logit(p_i)$ to ever get much bigger than 10 or smaller than -10
- I have no idea whether we are more likely to find these birds in high percentage forest or low, so I'm happy to think that β might be around zero, and be positive or negative. Forest cover ranges from 0 to 100 so that suggests that β is every likely to be bigger than 0.1 or smaller than -0.1. Perhaps $\beta \sim N(0, 0.1^2)$ is a good prior
- It looks to me like the intercept is very unlikely to be outside the range (-10, 10) so perhaps $\alpha \sim N(0, 5^2)$ is appropriate

rstanarm code

```
mod_1 = stan_glmer(cbind(y, N) ~  (forest | alt),
                   data = swt,
                   family = binomial(link = 'logit'),
                   prior = normal(0, 0.1),
                   prior_intercept = normal(0, 5))
```

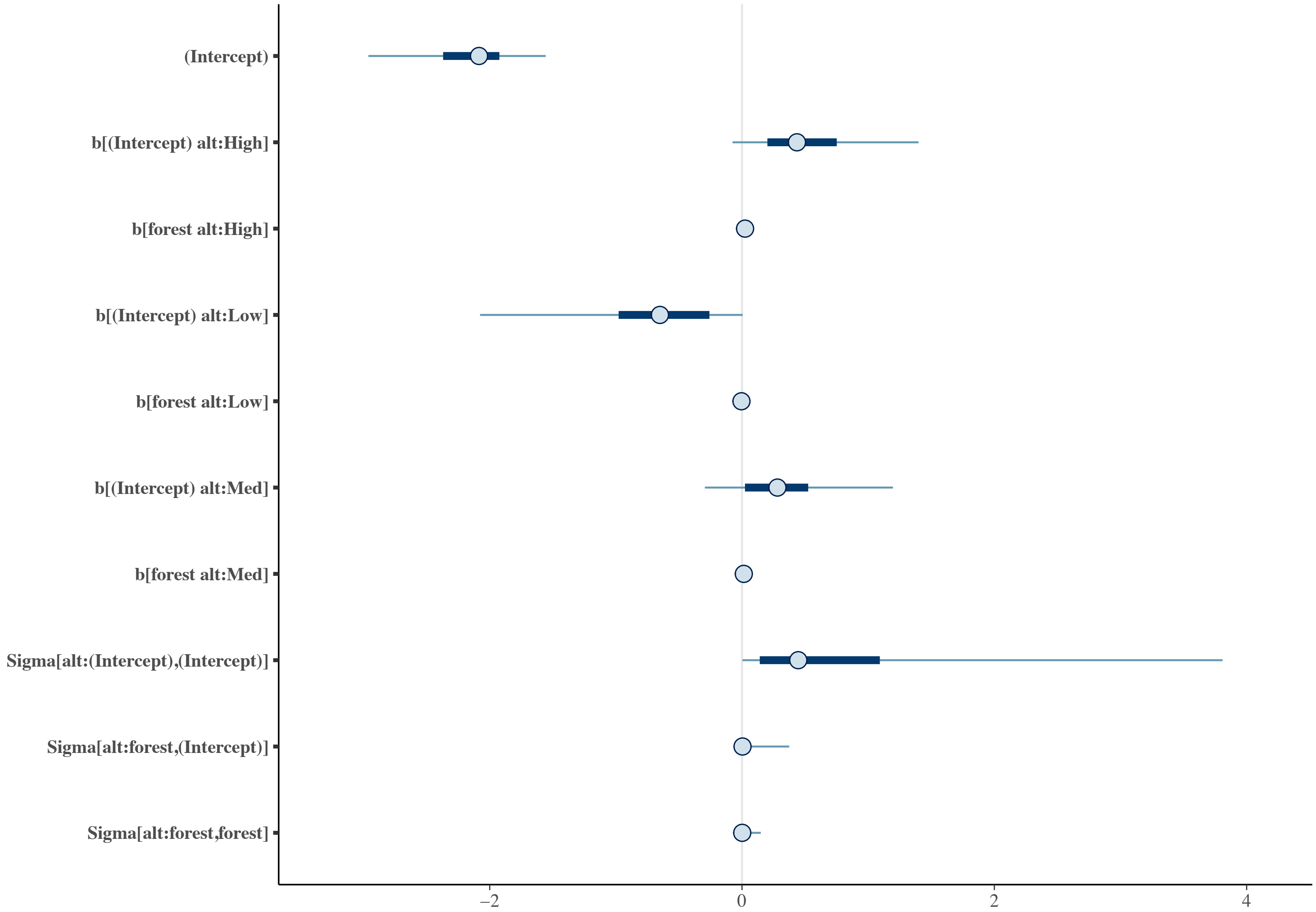
Model summary 1

```
round(as.data.frame(summary(mod_1)), 2)
```

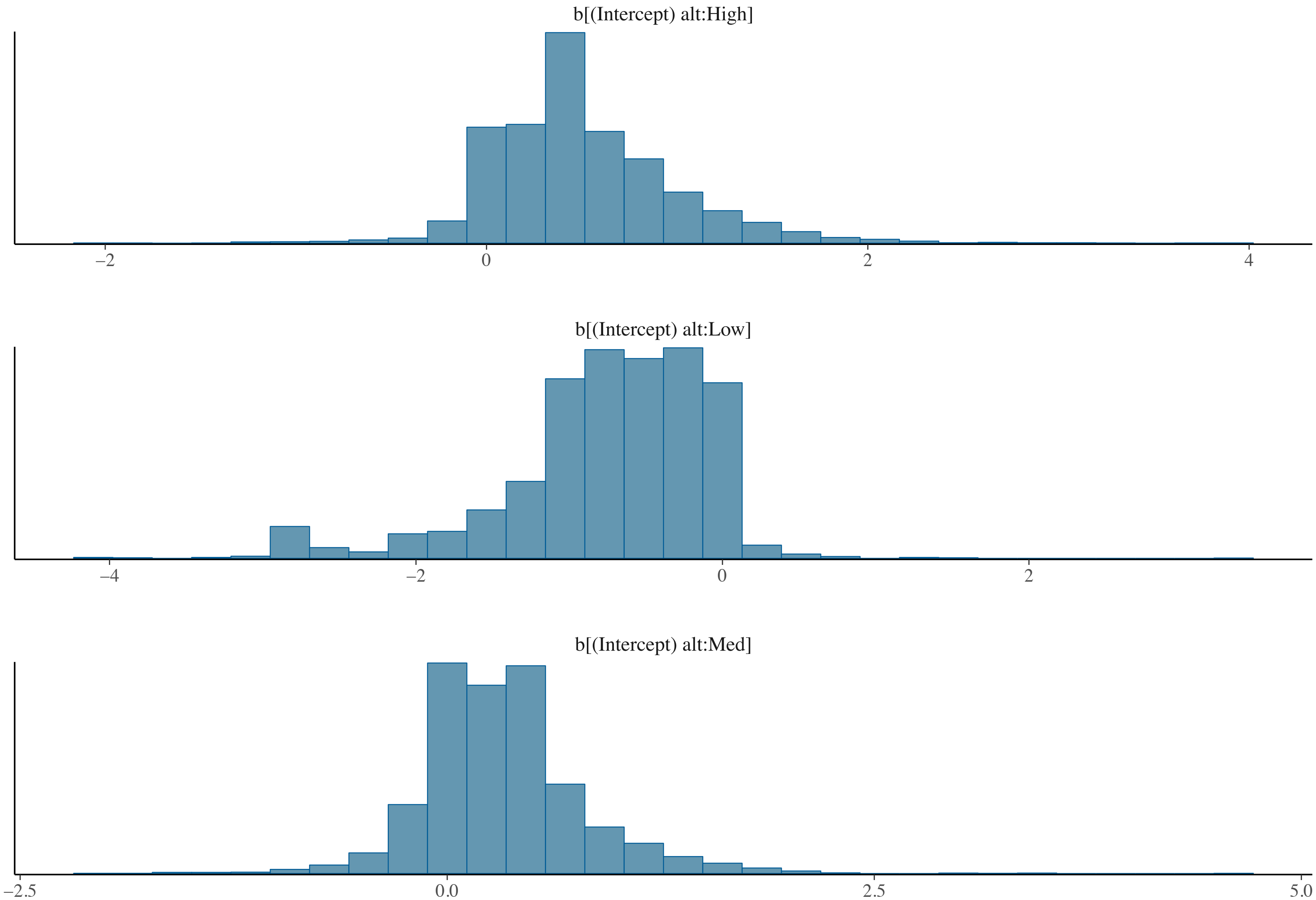
```
##                                    mean mcse   sd    2.5%     25%
## (Intercept)                       -2.16 0.01 0.45   -3.19   -2.37
## b[(Intercept) alt:High]            0.51 0.01 0.48   -0.23    0.20
## b[forest alt:High]                 0.02 0.00 0.00    0.01    0.02
## b[(Intercept) alt:Low]            -0.74 0.07 0.67   -2.73   -0.98
## b[forest alt:Low]                  0.00 0.00 0.01   -0.03   -0.01
## b[(Intercept) alt:Med]             0.32 0.01 0.48   -0.48    0.02
## b[forest alt:Med]                  0.01 0.00 0.01    0.00    0.01
## Sigma[alt:(Intercept),(Intercept)] 0.98 0.24 1.87    0.00    0.14
## Sigma[alt:forest,(Intercept)]      0.03 0.04 0.11   -0.12    0.00
## Sigma[alt:forest,forest]           0.03 0.01 0.06    0.00    0.00
## mean_PPD                           0.77 0.00 0.07    0.64    0.73
## log-posterior                   -215.54 0.17 3.34 -222.75 -217.68
##                                     50%     75%   97.5% n_eff Rhat
## (Intercept)                       -2.09   -1.92   -1.34   994 1.01
## b[(Intercept) alt:High]            0.44    0.75    1.65  1061 1.01
## b[forest alt:High]                 0.02    0.03    0.03  1124 1.00
## b[(Intercept) alt:Low]            -0.65   -0.26    0.06    88 1.04
## b[forest alt:Low]                  0.00    0.00    0.02   175 1.02
## b[(Intercept) alt:Med]             0.28    0.52    1.45  1023 1.01
## b[forest alt:Med]                  0.01    0.02    0.02   808 1.01
## Sigma[alt:(Intercept),(Intercept)] 0.45    1.09    8.31    61 1.07
## Sigma[alt:forest,(Intercept)]      0.00    0.02    0.40     9 1.64
## Sigma[alt:forest,forest]           0.00    0.01    0.29    24 1.22
## mean_PPD                           0.77    0.81    0.90  2243 1.00
## log-posterior                   -215.35 -213.27 -209.51   379 1.01
```

Model summary 2

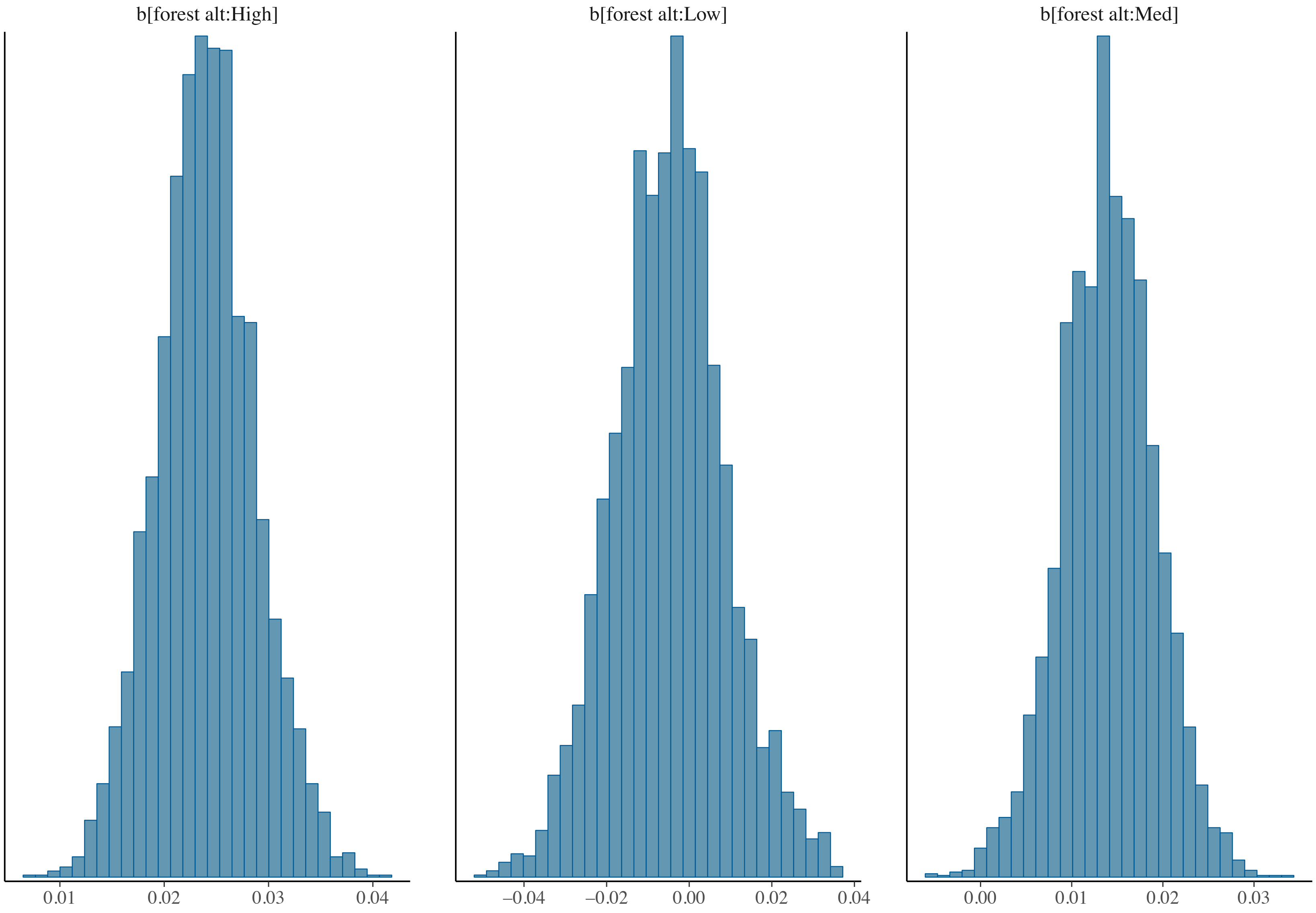
```r
plot(mod_1)
```



Model fit - intercepts

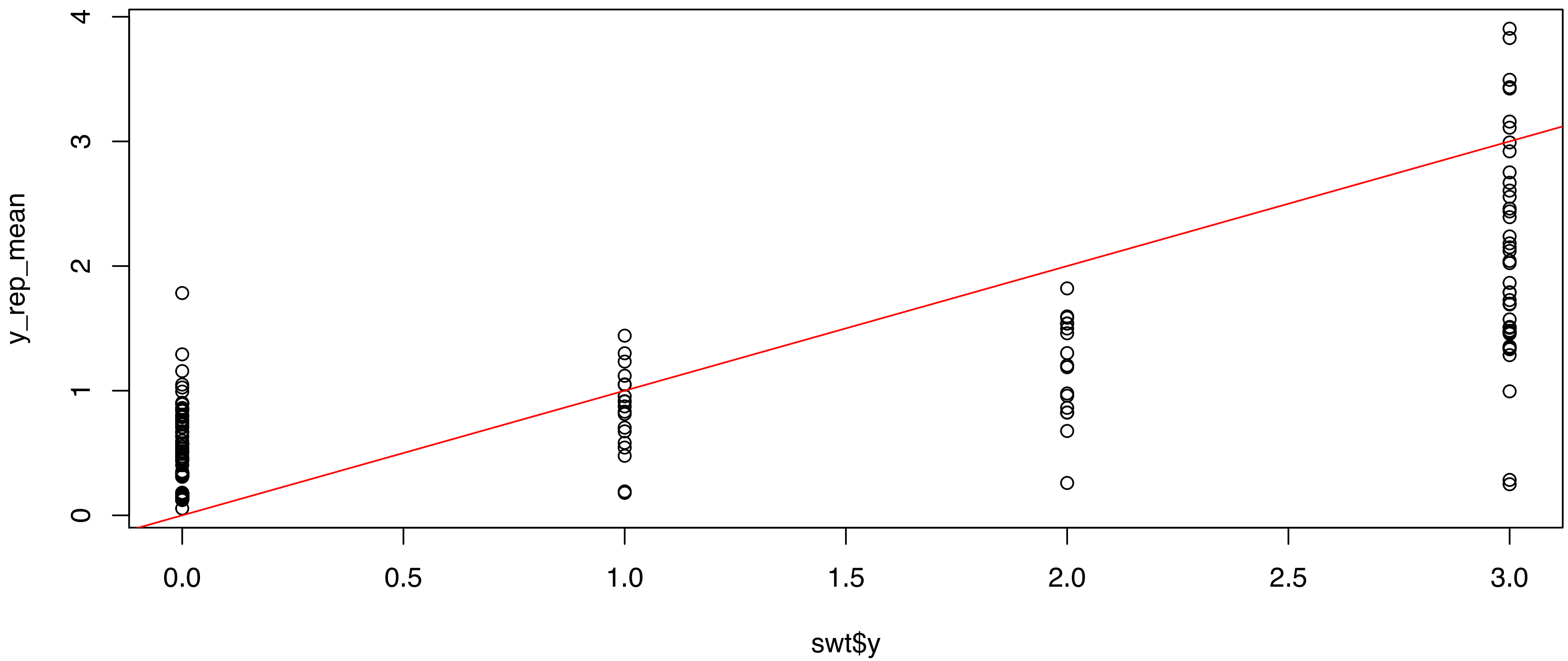


Model fit - Slopes

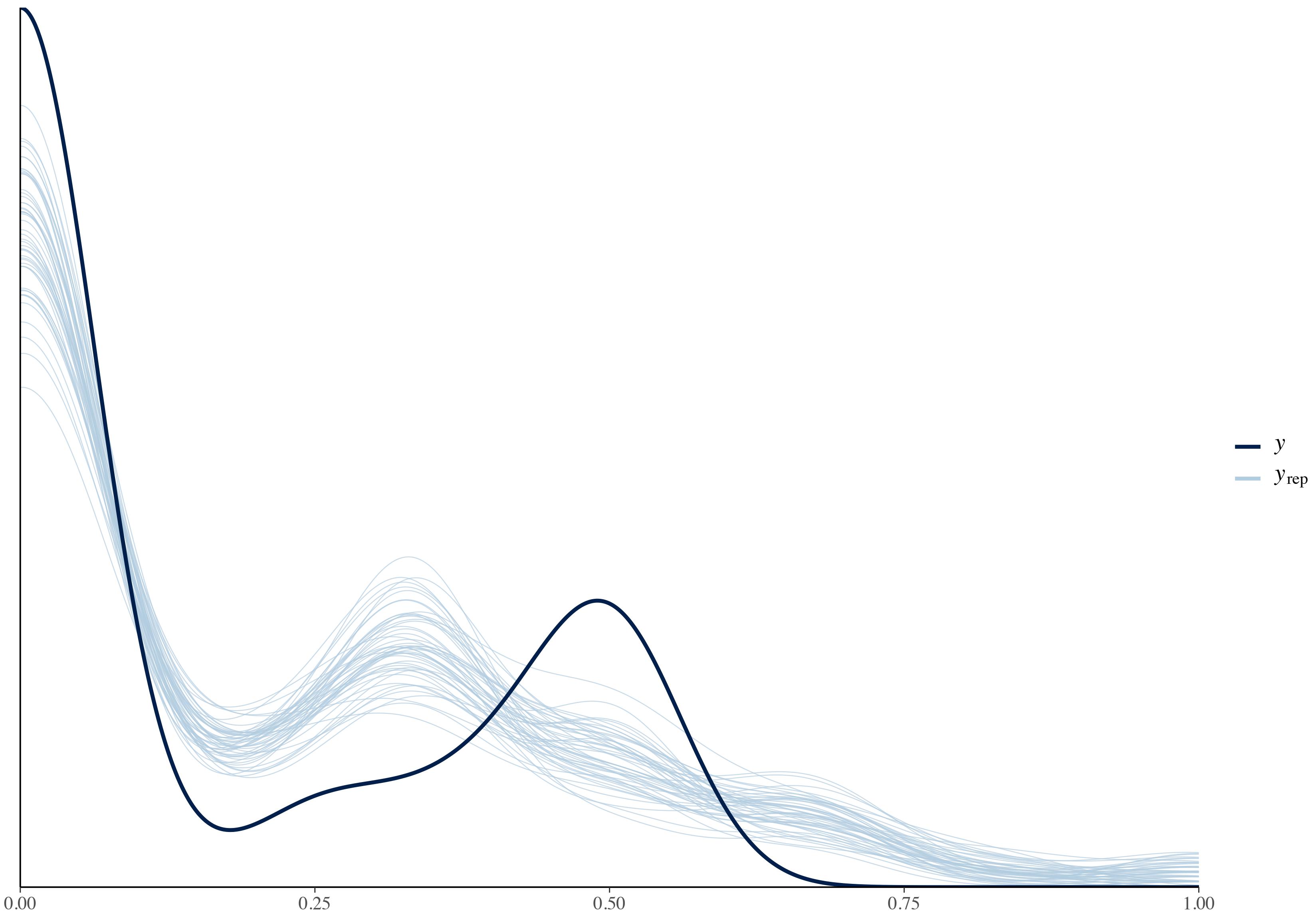


Model fit - posterior predictive check

```
y_rep = posterior_predict(mod_1)
y_rep_mean = apply(y_rep, 2, 'mean')
plot(swt$y, y_rep_mean)
abline(a = 0, b = 1, col = 'red')
```



Model fit - posterior predictive check 2



Type 2: Poisson HGLMs

- For a Poisson distribution there is no upper bound on the number of counts
- We just change the likelihood (to Poisson) and the link function (to log):

$$y_i \sim Po(\lambda_i), \log(\lambda_i) = \alpha + \beta(x_i - \bar{x}))$$

- We can now add our hierarchical layers into α and β, or...
- Another way we can add an extra layer is by giving $\log(\lambda_i)$ a probability distribution rather than setting it to a value
- This is a way of introducing *over-dispersion*, i.e. saying that the data are more variable than that expected by a standard Poisson distribution with our existing covariates

An over-dispersed model

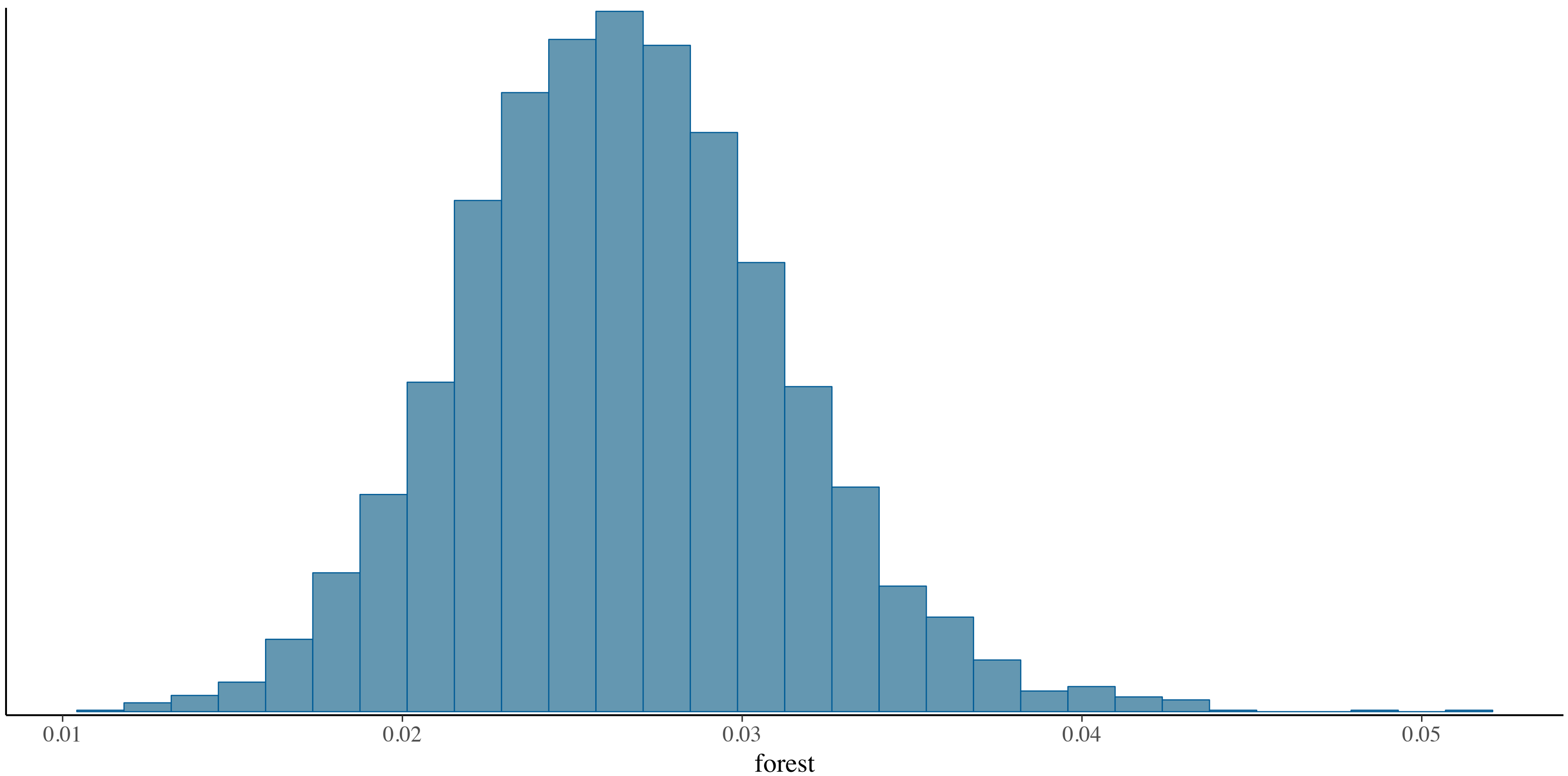
- The over-dispersed model looks like:

$$y_i \sim Po(\lambda_i), \log(\lambda_i) \sim N(\alpha + \beta(x_i - \bar{x}), \sigma^2)$$

  where σ is the over-dispersion parameter
- We now need to estimate prior distributions for α, β, and σ
- We will use the SWT data again, but pretend that we didn't know that they had gone out N times looking for the birds

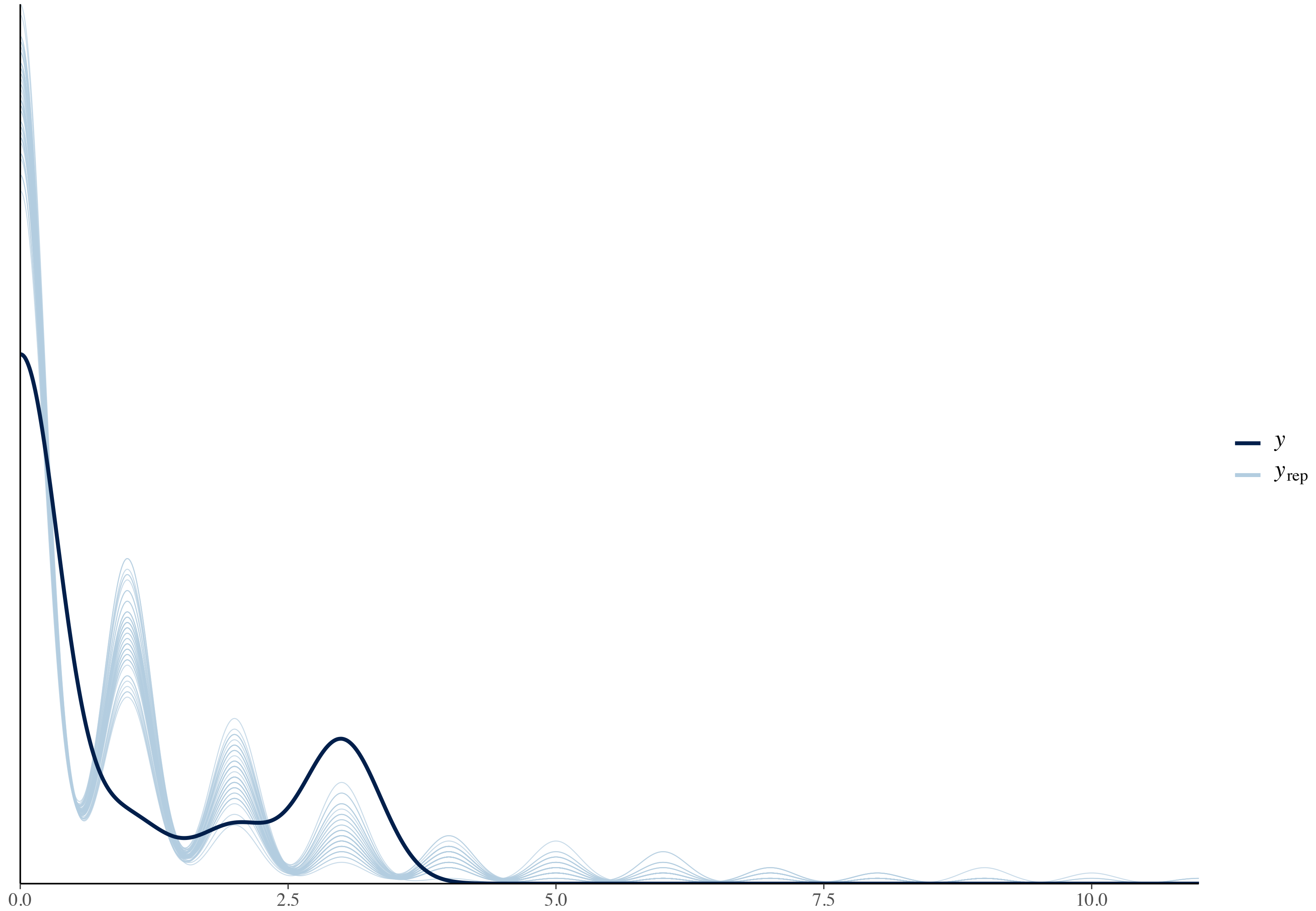
rstanarm code for OD Poisson

```
swt$obs <- 1:nrow(swt)
mod_2 = stan_glmer(y ~ forest + (1 | obs),
                   family = poisson, data = swt)
mcmc_hist(as.data.frame(mod_2), regex_pars = 'forest')
```



Posterior predictive check

```
pp_check(mod_2)
```



Notes about OD Poisson model

- The way to think about OD models is via the data generating process. Draw a DAG and think about how these processes might arise
- We could compare this model to one without over dispersion via the PPC (or if time, cross validation).
- In general, the parameter values (i.e. the intercepts and slopes) tend to be more uncertain when you add in over dispersion
- Also in the data set is a variable called `dur` which represents how long they spent looking for the birds. This could be added in as an offset via the likelihood

Type 4: Ordinal data HGLMs

- Often we have a response variable which is ordinal, e.g. disagree, neutral, agree, etc
- There are lots of different (and complicated) ways to model such data
- Perhaps the easiest is to think of it as a hierarchical model with ‘cut-points’ on a latent linear regression

An ordinal model example

- Suppose $y_i = \{\text{disagree, neutral, agree}\}$ and we make it dependent on a latent continuous variable z_i, so that :

$$y_i = \begin{cases} \text{agree} & \text{if } z_i > 0.5 \\ \text{neutral} & \text{if } -0.5 < z_i \leq 0.5 \\ \text{disagree} & \text{if } z_i \leq -0.5 \end{cases}$$

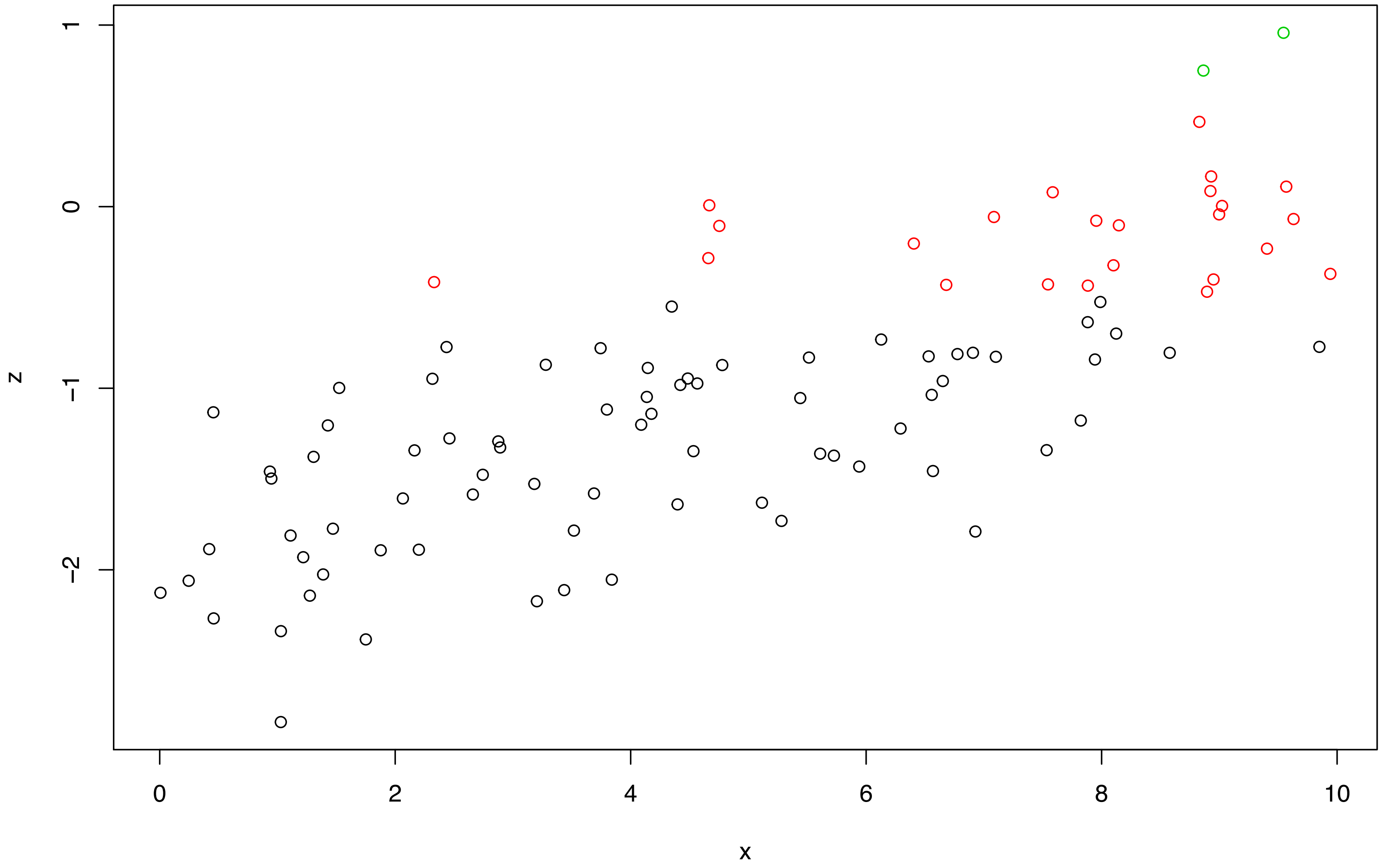
- We then give z_i a prior distribution, e.g. $N(\beta_0 + \beta_1 x_i, \sigma^2)$

Simulating some example data

```
N = 100
alpha = -1
beta = 0.2
sigma = 0.51
set.seed(123)
x = runif(N, 0, 10)
cuts = c(-0.5, 0.5)
z = rnorm(N, alpha + beta * (x - mean(x)), sigma)
y = findInterval(z, cuts)
dat = data.frame(y = as.factor(y),
                 x = x)
```

Simulated data - plot

```
plot(x, z, col = y + 1)
```



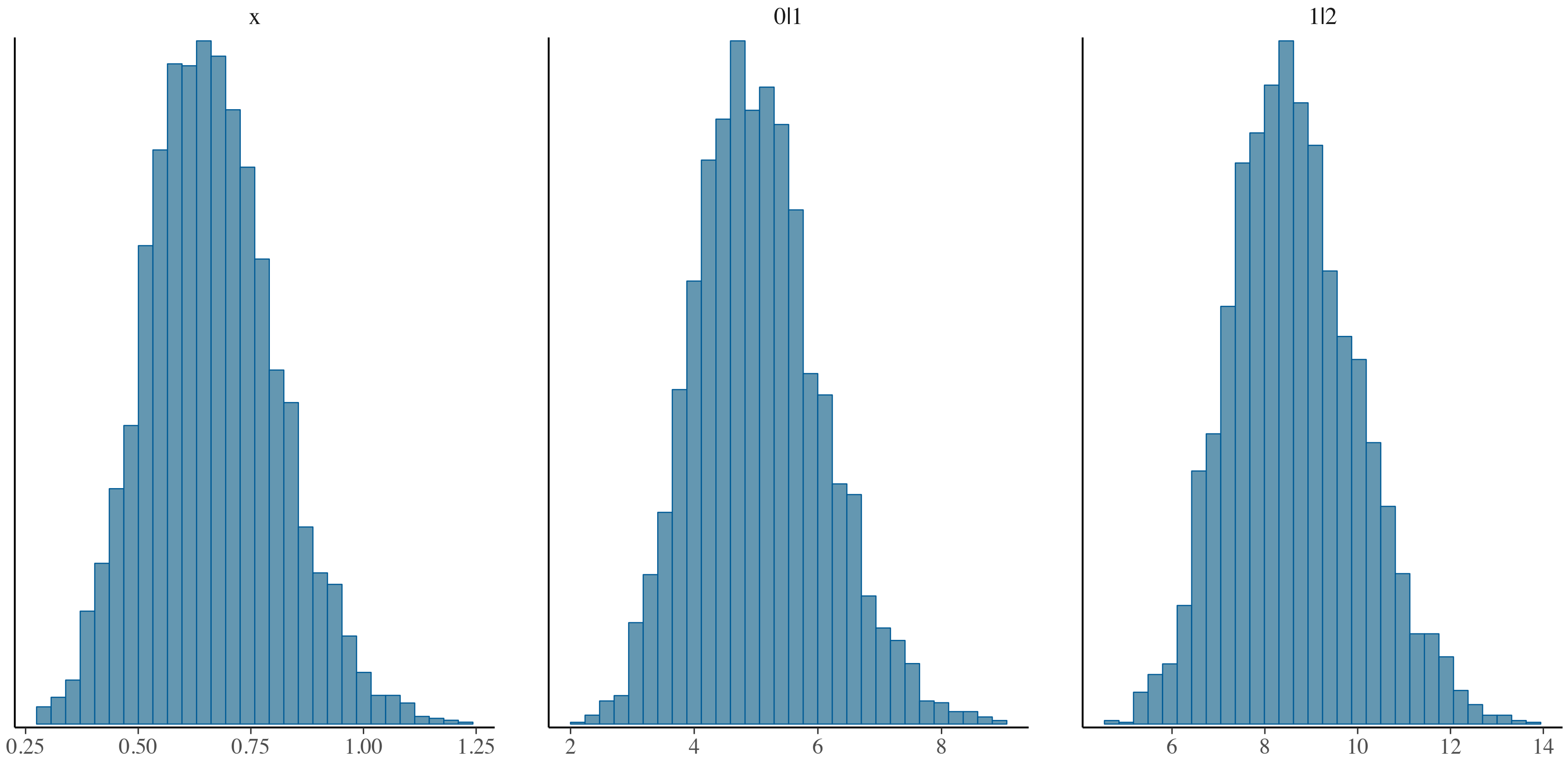
Fitting in rstanarm

```
mod_3 <- stan_polr(y ~ x,
                   data = dat,
                   prior = R2(0.25, 'mean'),
                   prior_counts = dirichlet(1))
```

Output

```
mcmc_hist(as.data.frame(mod_3))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwi
```



Summary

- We have now seen a number of different types of hierarchical GLM
- Many of the ideas of hierarchical linear models transfer over, but we can explore richer behaviour with hierarchical GLMs
- These have all used the normal, binomial or Poisson distribution at the top level, and have allowed for over-dispersion, robustness, and ordinal data, to name just three