

Class 9: Bayesian Hierarchical Models

Andrew Parnell
andrew.parnell@mu.ie



**Maynooth
University**
National University
of Ireland Maynooth

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: earnings data (again!)

- Very easy to convert quickly from `lmer` to `stan_lmer`:

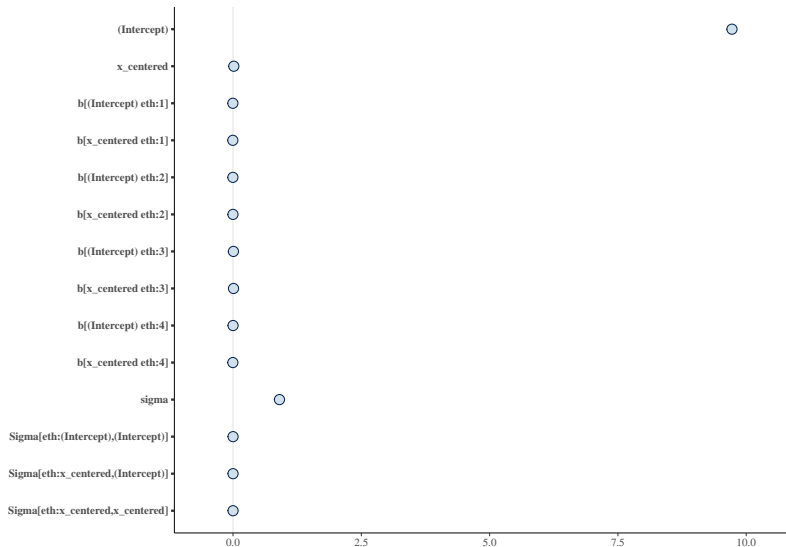
```
dat = read.csv('data/earnings.csv')
mod_1 = stan_lmer(y ~ x_centered + (x_centered | eth), data = dat)

round(posterior_interval(mod_1), 3)
```

```
##                                5%   95%
## (Intercept)                   9.636 9.782
## x_centered                    -0.009 0.031
## b[(Intercept) eth:1]          -0.081 0.040
## b[x_centered eth:1]           -0.025 0.016
## b[(Intercept) eth:2]          -0.105 0.040
## b[x_centered eth:2]           -0.022 0.022
## b[(Intercept) eth:3]          -0.025 0.107
## b[x_centered eth:3]           -0.005 0.035
## b[(Intercept) eth:4]          -0.068 0.075
## b[x_centered eth:4]           -0.033 0.017
## sigma                        0.873 0.938
## Sigma[eth:(Intercept),(Intercept)] 0.000 0.016
## Sigma[eth:x_centered,(Intercept)] -0.001 0.002
## Sigma[eth:x_centered,x_centered]   0.000 0.002
```

Earnings model output

```
plot(mod_1)
```



Example 2: binomial-logit

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

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

Here $\text{logit}(p_i)$ 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:

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

- ▶ 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

```
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 $y_i = \text{rep.1} + \text{rep.2} + \text{rep.3}$:

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

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 $\text{logit}(p_i)$ 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_2 = stan_glmer(cbind(y, N) ~ forest + (forest | alt),  
  data = swt,  
  family = binomial(link = 'logit'),  
  prior = normal(0, 0.1),  
  prior_intercept = normal(0, 5))
```

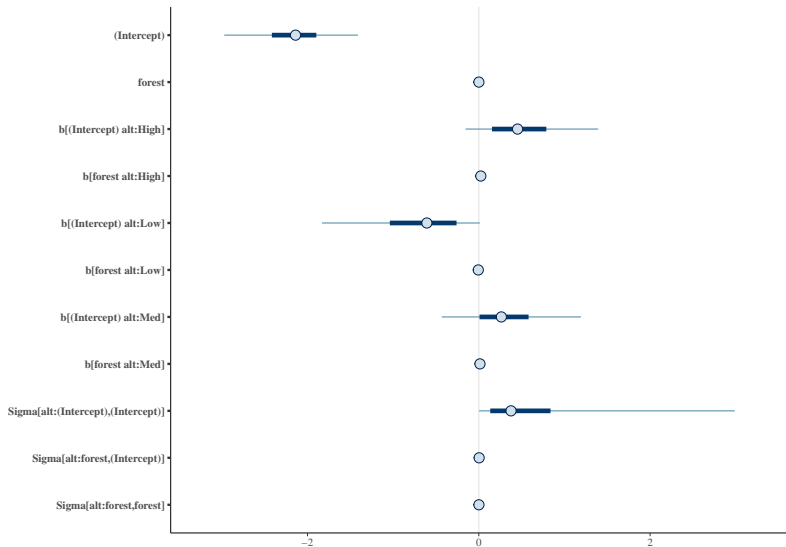
Model summary 1

```
posterior_interval(mod_2)
```

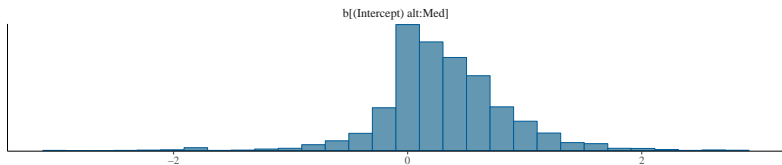
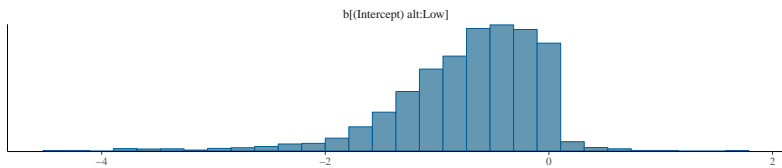
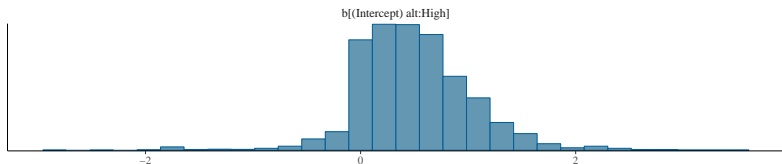
##	5%	95%
## (Intercept)	-2.9716628359	-1.410061400
## forest	-0.0053591880	0.006660342
## b[(Intercept) alt:High]	-0.1546953851	1.393642353
## b[forest alt:High]	0.0132429998	0.033927229
## b[(Intercept) alt:Low]	-1.8313389184	0.011825971
## b[forest alt:Low]	-0.0278474865	0.015678492
## b[(Intercept) alt:Med]	-0.4328987953	1.192305997
## b[forest alt:Med]	0.0024746665	0.024134954
## Sigma[alt:(Intercept),(Intercept)]	0.0033074897	2.987954932
## Sigma[alt:forest,(Intercept)]	-0.0316645266	0.056452610
## Sigma[alt:forest,forest]	0.0001374629	0.025760649

Model summary 2

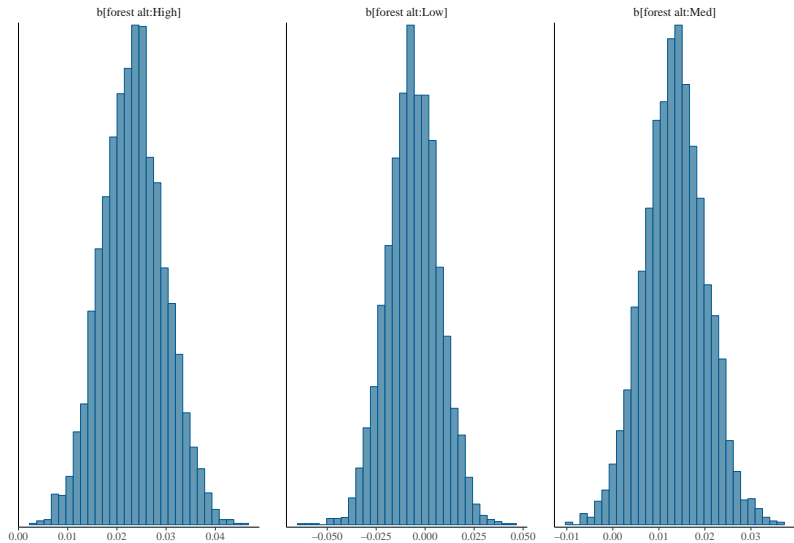
```
plot(mod_2)
```



Model fit - intercepts

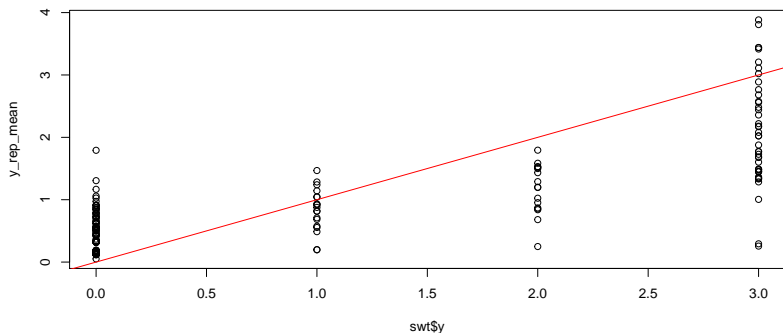


Model fit - Slopes

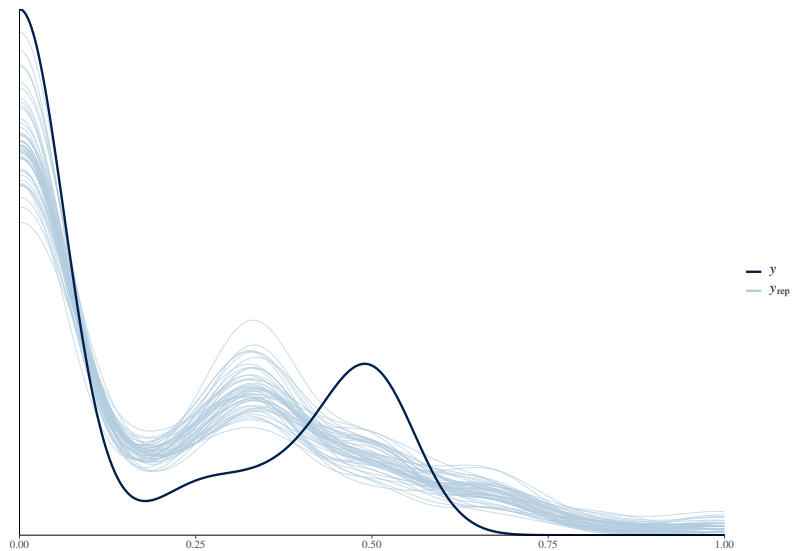


Model fit - posterior predictive check

```
y_rep = posterior_predict(mod_2)
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:

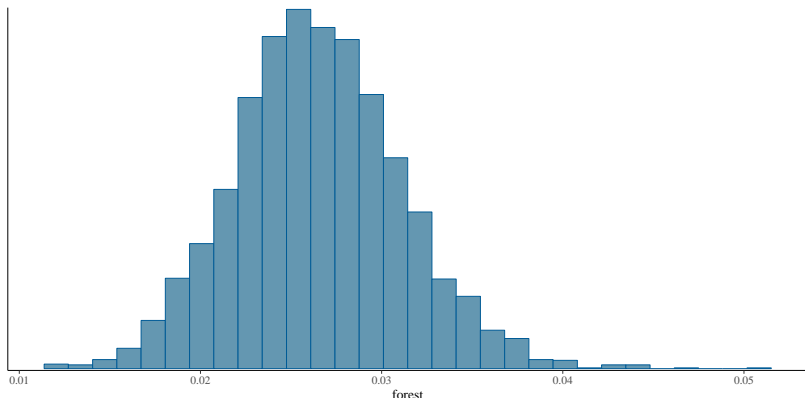
$$y_i \sim Po(\lambda_i), \log(\lambda_i) \sim N(\alpha + \beta(x_i - \bar{x}), \sigma^2)$$

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

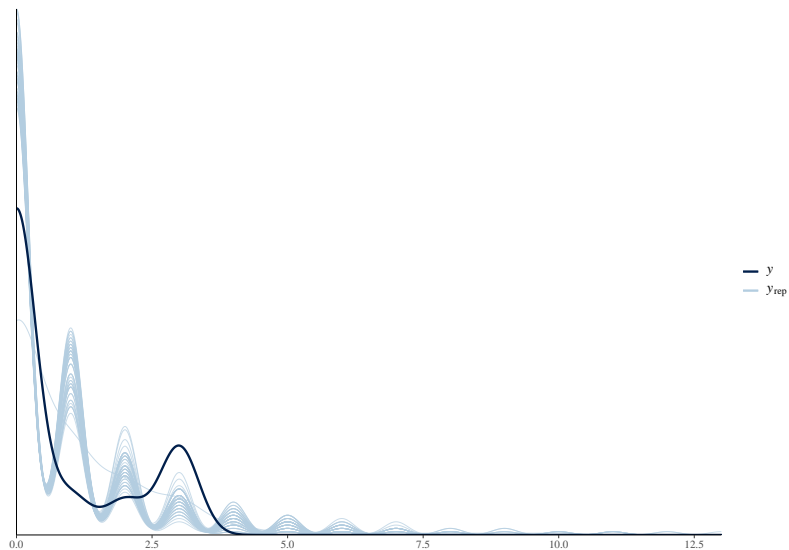
rstanarm code for OD Poisson

```
swt$obs <- 1:nrow(swt)
mod_3 = stan_glmer(y ~ forest + (1 | obs),
  family = poisson, data = swt)
mcmc_hist(as.data.frame(mod_3), regex_pars = 'forest')
```



Posterior predictive check

```
pp_check(mod_3)
```



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 :

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

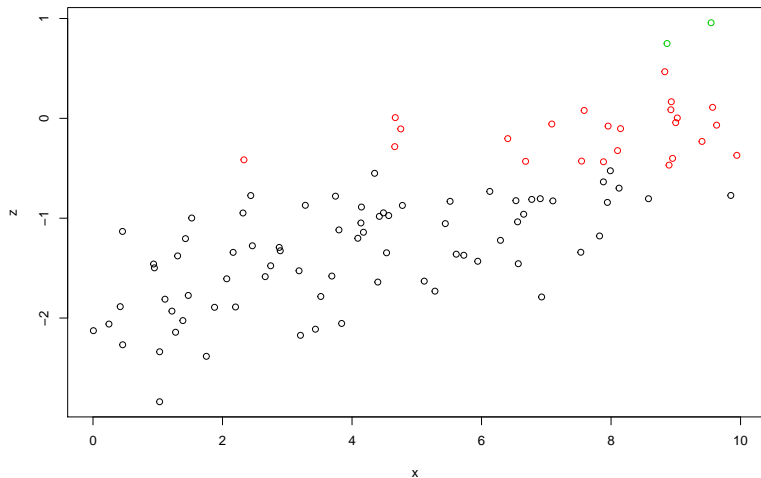
- We then give z_i a prior distribution, e.g. $N(\beta_0 + \beta_1 x_i, \sigma^2)$

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)
```



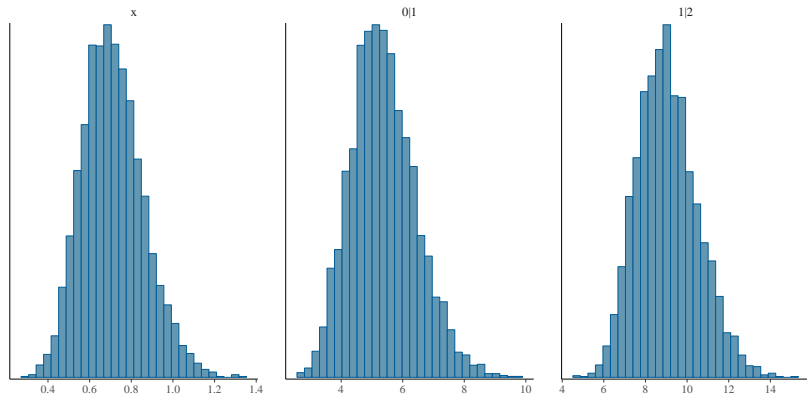
Fitting in rstanarm

```
mod_4 = stan_polr(y ~ x,  
                  data = dat,  
                  prior = R2(0.5, "mean"),  
                  prior_counts = dirichlet(1))
```

Output

```
mcmc_hist(as.data.frame(mod_4))
```

```
## `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