Class 11: Multivariate and multi-layer hierarchical models

Andrew Parnell
andrew.parnell@mu.ie

Maynooth University
National University of Ireland Maynooth

Learning outcomes:

- Understand how to add in multiple layers to a hierarchical model
- Follow a detailed example of building a model
- Be able to fit multivariate models in `rstan`

Some new terminology

- Most of the models we have covered so far contain only one hidden or *latent* set of parameters
- For example, the data y may depend on a parameter β, which itself depends on a parameter θ. θ is given a prior distribution
- We say that the data are at the 'top level', the parameter β is a *latent parameter* at the second level, and the hyper-parameter θ is also a latent parameter at the third level
- We say that the prior distribution on β is *conditional* on θ, whilst the prior distribution (if it just involves numbers) is a *marginal prior distribution*

What is a multi-layer model?

- A multi-layer model is one where have many (usually more than 2 or 3) layers of parameters conditional on each other
- It's very straightforward to add in these extra layers in Stan
- The question is whether they are necessary or not, and how much the data can tell us about them

A simple example

- We will work through the `earnings` example, extending it to produce a much more complex model
- Going back to the linear regression version, we had:

```
'
...
y ~ normal(intercept + slope * x, residual_sd);
...
intercept ~ normal(0, 100);
slope ~ normal(0, 100);
residual_sd ~ cauchy(0, 10);
'
```

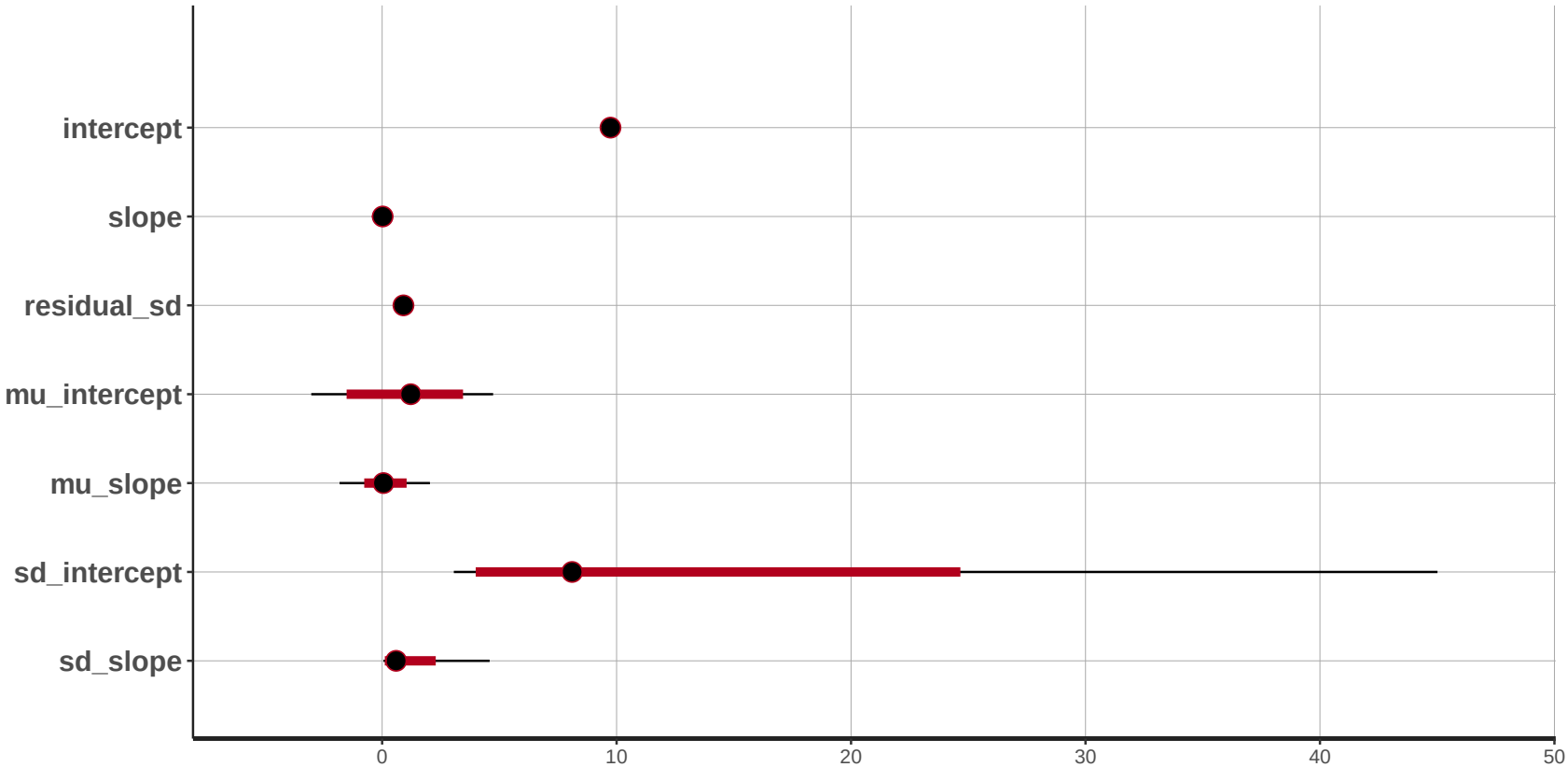
Adding extra layers

```
stan_code = '
data {
  int N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real intercept;
  real slope;
  real<lower=0> residual_sd;
  real mu_intercept;
  real mu_slope;
  real sd_intercept;
  real sd_slope;
}
model {
  // Likelihood
  y ~ normal(intercept + slope * x, residual_sd);
  // Priors
  intercept ~ normal(mu_intercept, sd_intercept);
  slope ~ normal(mu_slope, sd_slope);
  residual_sd ~ cauchy(0, 10);
  mu_intercept ~ normal(0, 2);
  mu_slope ~ normal(0, 2);
  sd_intercept ~ cauchy(0, 1);
  sd_slope ~ cauchy(0, 1);
}
'
```

What information is there about these extra parameters?

```
## ci_level: 0.8 (80% intervals)

## outer_level: 0.95 (95% intervals)
```



How many layers should I add?

- We could go on adding layers here if we wanted to, but it's not clear what benefit it would have
- For example, do we really need to know the standard deviation of the mean of the intercept? The answer to this will depend on the questions of interest, and the amount and type of prior information available
- A general rule is, in the absence of any strong prior information, to add one extra layer of parameters beyond the latent parameters of key interest

Stan and indexing

- When writing out the model with differing slopes and intercepts for each ethnic group, the hierarchical formulation ties them together through the prior distributions
- The likelihood is written as:

```
y[i] ~ normal(alpha[eth[i]] +
              beta[eth[i]]*(x[i] - mean(x)),
                sigma)
```

  which in maths can be written as:

$$y_i \sim N(\alpha_{\text{eth}_i} + \beta_{\text{eth}_i} x_i, \sigma^2)$$

  where eth_i takes the values 1, 2, 3, or 4
- Remember, y_i is the log-earnings of individual i where $i = 1, \ldots, N$

Re-writing the model

- Commonly you'll see y here re-defined as y_{ij} where $j = 1, .., 4$ represents ethnicity, and $i = 1, \ldots, N_j$ is the number of individuals with ethnicity j
- The likelihood can then be written as:

$$y_{ij} \sim N(\alpha_j + \beta_j x_i, \sigma^2)$$

- Note that this is exactly the same model, just re-written slightly differently. In fact, this latter model is much harder to write out in Stan code

Fixed vs random effect models

- Thinking about this model in more detail

$$y_{ij} \sim N(\alpha_j + \beta_j x_i, \sigma^2)$$

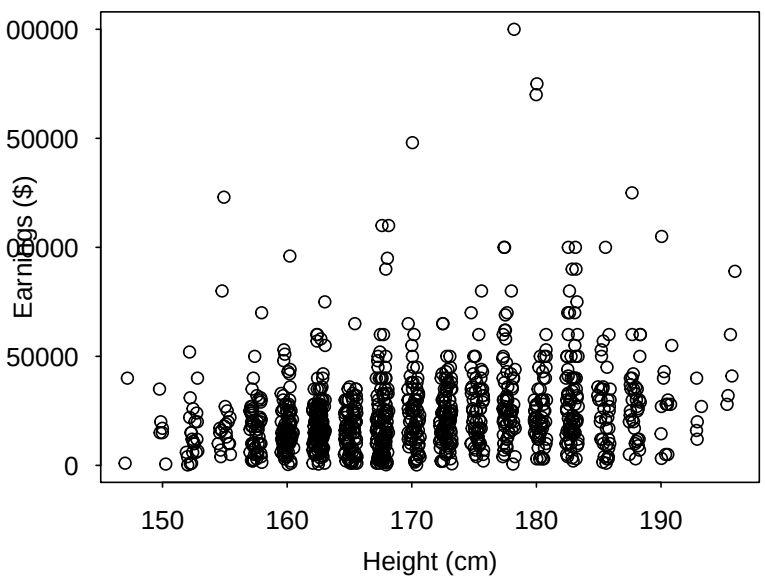
- If the α_j and β_j parameters are all given independent prior distributions, e.g. $\alpha_j \sim N(0, 100)$ or similar, then this is considered a *fixed effects* model
- If the α_j and β_j are given prior distributions that tie the values together, e.g. $\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2)$, then this is often called a *random effects* model
- (In fact, nobody can agree on what a fixed or random effects model actually is)
- If you have multiple different types of priors then this is called a *mixed effects* model

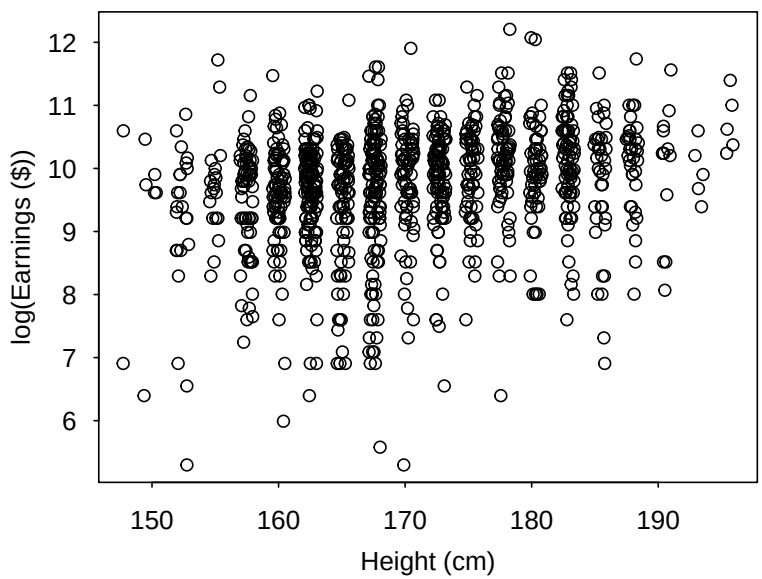
Mixed effects vs hierarchical models

- The hierarchical models we have been studying all use the *random effects* approach wherever possible
- The big advantage of using this approach is that we get to *borrow* strength between the different groups (here `eth`, but it could be anything)
- Whenever we have a categorical covariate we should always be putting a constraining/tying prior distribution on them, and looking at how the effects vary between the groups
- This is strongly linked to the idea of *partial pooling* which we will meet later in the course

Example: multi-layer earnings data

- We will now go through and build a much more complicated model for the earnings data, taken from the Gelman and Hill book, using only weak priors
- We can generate data from these models (either using the prior or the posterior), and we can draw a DAG
- Our goal is to explore the factors which explain earnings. We have variables on height, age, and ethnicity (but not sex!).
- If we first plot the data





Transformations

- From the left-hand plot there seem to be quite a few extreme observations, and there's a possibility that the relationship between height and earnings is non-linear
- The right-hand plot seems to have stabilised most of the extreme observations, and perhaps linearity is more appropriate
- Notice that a linear model implies:

$$y_i = \alpha + \beta x_i + \epsilon_i$$

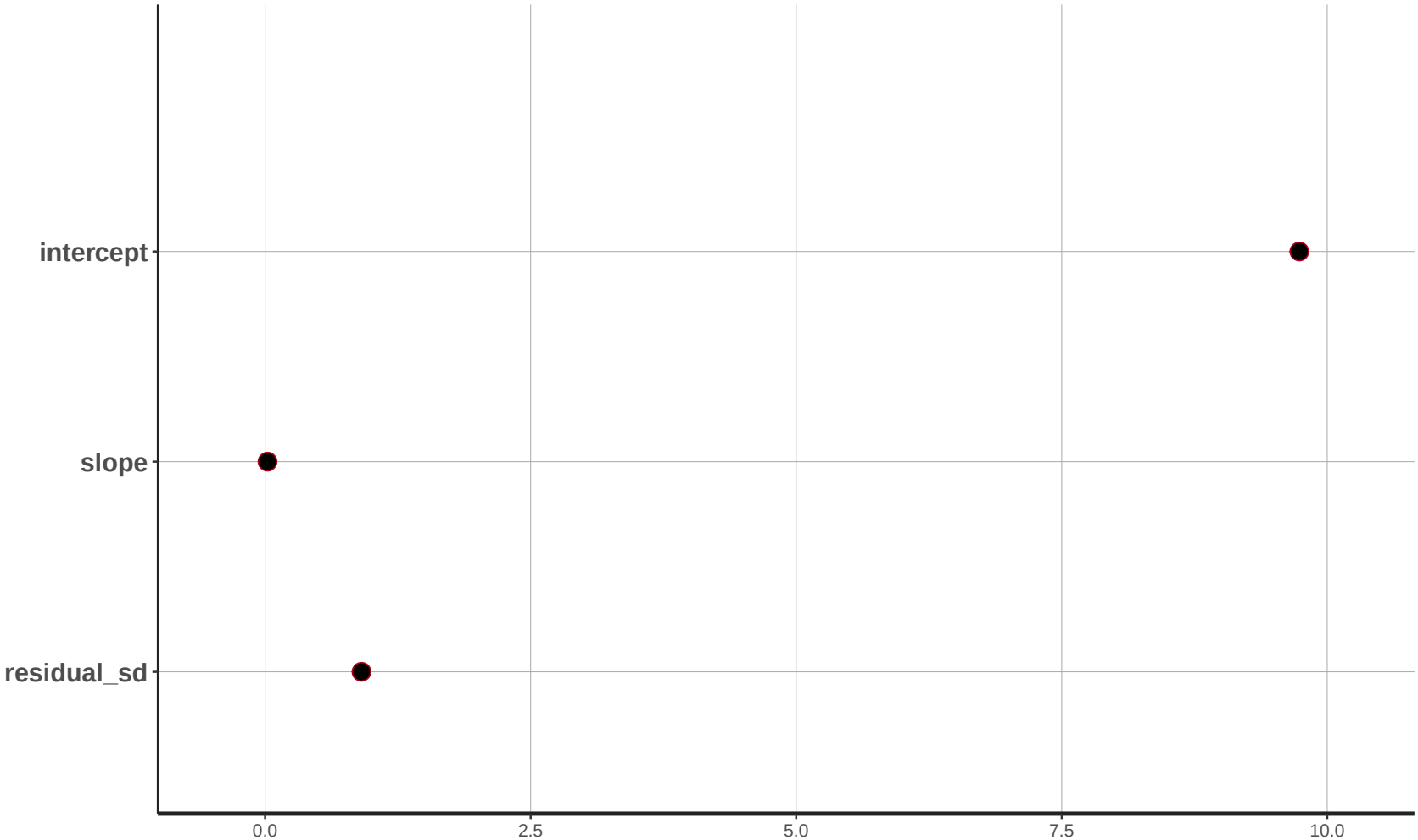
whilst the log-linear model implies:

$$y_i = \exp(\alpha + \beta x_i + \epsilon_i) = e^{\alpha} \times e^{\beta x_i} \times e_i^{\epsilon}$$

so the coefficients, once exponentiated, have multiplicative effects that are relatively easy to interpret

Fitting the first model

- If we fit a model with just height (mean centered) we get the following Stan output



Interpreting the parameters

- These parameters are directly interpretable:

```
stan_run_2_summ = summary(stan_run_2)
round(stan_run_2_summ$summary, 2)
```

```
##                mean se_mean   sd    2.5%     25%     50%     75%   97.5%
## intercept      9.74    0.00 0.03    9.68    9.72    9.74    9.76    9.79
## slope          0.02    0.00 0.00    0.02    0.02    0.02    0.02    0.03
## residual_sd    0.91    0.00 0.02    0.87    0.90    0.91    0.92    0.95
## lp__        -426.23    0.03 1.24 -429.31 -426.80 -425.91 -425.32 -424.83
##                n_eff Rhat
## intercept    2219.39    1
## slope        4000.00    1
## residual_sd  2755.38    1
## lp__         1914.48    1
```

- The mean of the log earnings at the mean height is about 9.737, which is about 17k on the original scale
- We can also get e.g. a 95% confidence interval directly from the output
- For every extra cm so you gain 0.0225 on the log scale, i.e. an 2.28% gain in income
- From the posterior of σ, we can guess that about 68% of predictions will be within 0.908 on the log scale or within a factor of about 2.48 of the prediction
- Interpretation for the intercept would have been harder had we not mean-centered the height variable

Improving the model

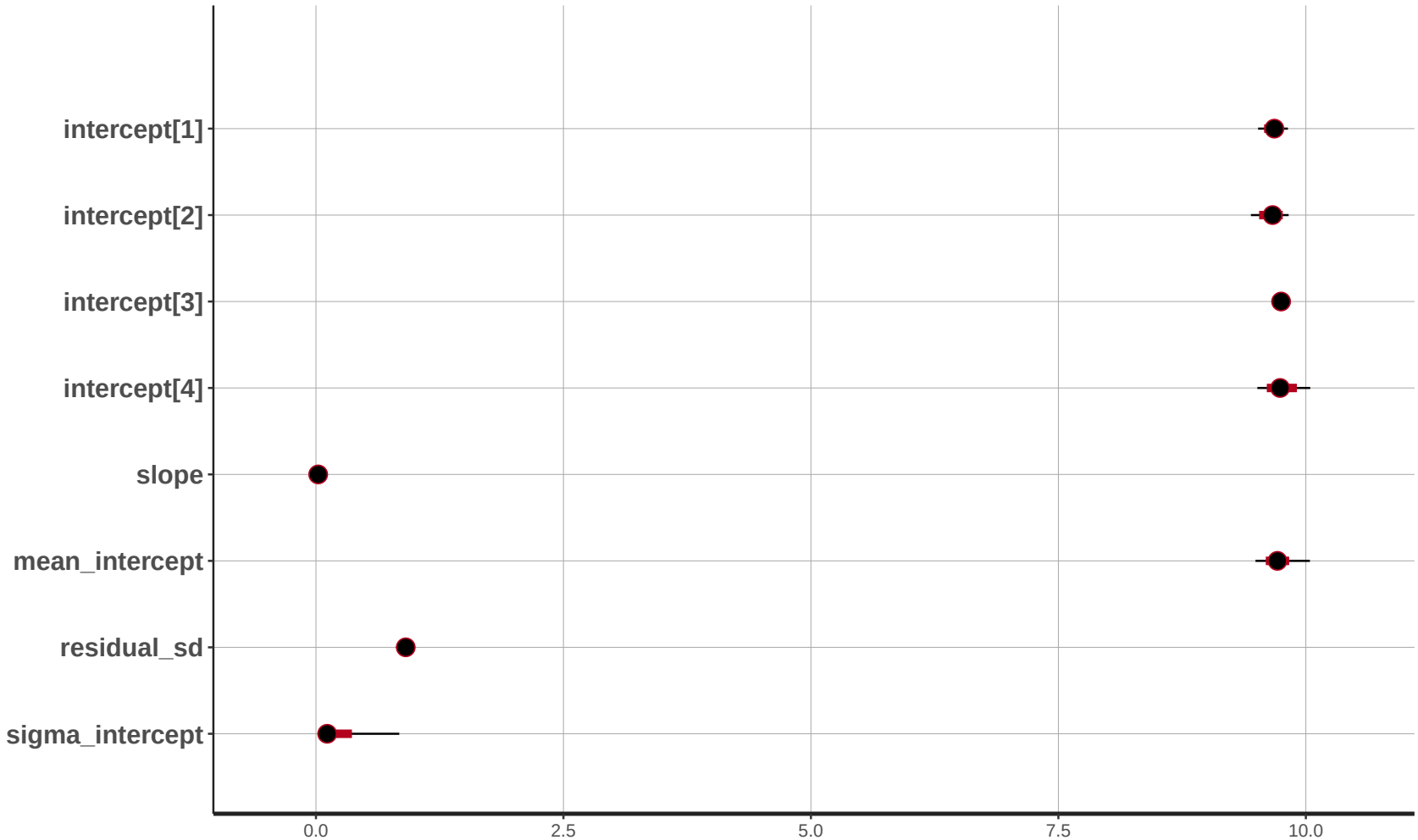
- Now suppose we fit a model with a random intercept for ethnicity

```
stan_code_3 = '
data {
  int N;
  int N_eth;
  vector[N] x;
  vector[N] y;
  int eth[N];
}
parameters {
  real intercept[N_eth];
  real slope;
  real mean_intercept;
  real<lower=0> residual_sd;
  real<lower=0> sigma_intercept;
}
model {
  // Likelihood
  for (i in 1:N) {
    y[i] ~ normal(intercept[eth[i]] + slope * x[i], residual_sd);
  }
  // Priors
  slope ~ normal(0, 0.1);
  for (j in 1:N_eth) {
    intercept[j] ~ normal(mean_intercept, sigma_intercept);
  }
  mean_intercept ~ normal(11, 2);
  sigma_intercept ~ cauchy(0, 10);
  residual_sd ~ cauchy(0, 10);
}
'
```

Random intercept model fits

```
## ci_level: 0.8 (80% intervals)

## outer_level: 0.95 (95% intervals)
```



Interpreting the output

```
stan_run_3_summ = summary(stan_run_3)
round(stan_run_3_summ$summary, 2)
```

```
##                     mean se_mean   sd    2.5%     25%     50%     75%
## intercept[1]        9.68    0.00 0.08    9.52    9.63    9.68    9.73
## intercept[2]        9.65    0.00 0.10    9.45    9.60    9.66    9.72
## intercept[3]        9.75    0.00 0.03    9.69    9.73    9.75    9.77
## intercept[4]        9.75    0.00 0.13    9.51    9.67    9.74    9.82
## slope               0.02    0.00 0.00    0.02    0.02    0.02    0.02
## mean_intercept      9.73    0.01 0.18    9.49    9.66    9.71    9.76
## residual_sd         0.91    0.00 0.02    0.87    0.89    0.91    0.92
## sigma_intercept     0.19    0.01 0.31    0.03    0.07    0.11    0.21
## lp__             -421.11    0.12 2.88 -427.80 -422.75 -420.84 -419.12
##                    97.5%   n_eff Rhat
## intercept[1]        9.82 2118.16 1.00
## intercept[2]        9.83 1601.60 1.00
## intercept[3]        9.81 2052.98 1.00
## intercept[4]       10.04 1988.57 1.00
## slope               0.03 4000.00 1.00
## mean_intercept     10.04  411.58 1.01
## residual_sd         0.95 1509.05 1.00
## sigma_intercept     0.84  640.29 1.01
## lp__             -416.30  551.07 1.01
```

- The parameters α and β haven't changed much in the mean
- The 95% confidence interval for α has increased
- We also have estimates for each ethnicity intercept, none of these have a strong effect away from the mean

Model with varying slopes and intercepts

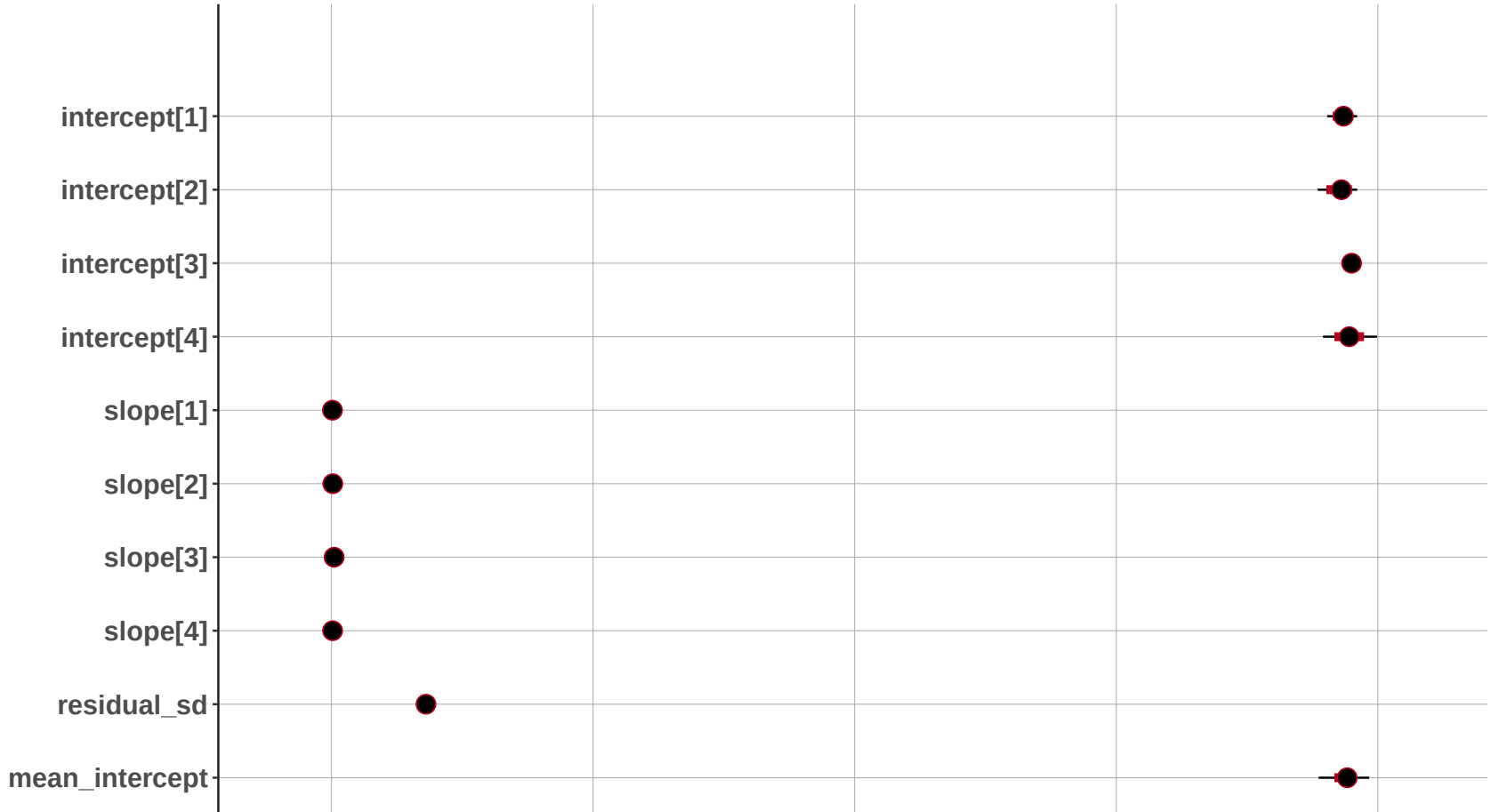
```
stan_code_4 = '
data {
  int N;
  int N_eth;
  vector[N] x;
  vector[N] y;
  int eth[N];
}
parameters {
  real intercept[N_eth];
  real slope[N_eth];
  real<lower=0> residual_sd;
  real mean_intercept;
  real<lower=0> sigma_intercept;
  real mean_slope;
  real<lower=0> sigma_slope;
}
model {
  // Likelihood
  for (i in 1:N) {
    y[i] ~ normal(intercept[eth[i]] + slope[eth[i]] * x[i], residual_sd);
  }
  // Priors
  for (j in 1:N_eth) {
    intercept[j] ~ normal(mean_intercept, sigma_intercept);
    slope[j] ~ normal(mean_slope, sigma_slope);
  }
  mean_intercept ~ normal(11, 2);
  sigma_intercept ~ cauchy(0, 10);
  mean_slope ~ normal(0, 1);
  sigma_slope ~ cauchy(0, 10);
  residual_sd ~ cauchy(0, 10);
}
'
```

Random intercept/slope model fits

```
## 'pars' not specified. Showing first 10 parameters by def

## ci_level: 0.8 (80% intervals)

## outer_level: 0.95 (95% intervals)
```



Random intercept/slopes interpretation

```
stan_run_4_summ = summary(stan_run_4)
round(stan_run_4_summ$summary, 2)
```

```
##                      mean se_mean   sd     2.5%      25%      50%      75%
## intercept[1]         9.67    0.01 0.08     9.52     9.61     9.67     9.72
## intercept[2]         9.64    0.00 0.10     9.42     9.58     9.65     9.71
## intercept[3]         9.75    0.00 0.03     9.69     9.73     9.75     9.77
## intercept[4]         9.72    0.00 0.12     9.48     9.66     9.73     9.78
## slope[1]             0.01    0.00 0.01    -0.01     0.00     0.01     0.02
## slope[2]             0.01    0.00 0.01    -0.01     0.01     0.01     0.02
## slope[3]             0.03    0.00 0.00     0.02     0.02     0.03     0.03
## slope[4]             0.01    0.00 0.02    -0.03     0.00     0.01     0.02
## residual_sd          0.90    0.00 0.02     0.87     0.89     0.90     0.92
## mean_intercept       9.70    0.01 0.14     9.43     9.65     9.71     9.75
## sigma_intercept      0.17    0.01 0.27     0.02     0.06     0.11     0.19
## mean_slope           0.01    0.00 0.02    -0.02     0.01     0.02     0.02
## sigma_slope          0.02    0.00 0.03     0.00     0.01     0.02     0.03
## lp__              -407.26    0.29 3.93  -415.50  -409.89  -406.98  -404.32
##                     97.5%   n_eff Rhat
## intercept[1]         9.80  186.83 1.02
## intercept[2]         9.80  764.30 1.00
## intercept[3]         9.81  239.54 1.02
## intercept[4]         9.99 1685.87 1.00
## slope[1]             0.02   98.33 1.03
## slope[2]             0.03  152.04 1.02
## slope[3]             0.03 4000.00 1.00
## slope[4]             0.04  349.76 1.01
## residual_sd          0.94  155.30 1.03
## mean_intercept       9.92  614.83 1.01
## sigma_intercept      0.68  501.78 1.00
## mean_slope           0.04  685.83 1.01
## sigma_slope          0.08  742.87 1.01
## lp__              -400.39  178.26 1.01
```

Introducing age

- Let's fit an even more complicated model with intercepts and slopes varying by ethnicity and age group
- Age is divided up into three groups 1: 18-34, 2: 35-49, and 3: 50-64
- We want to know whether the degree to which height affects earnings for different ethnic/age group combinations

Stan model

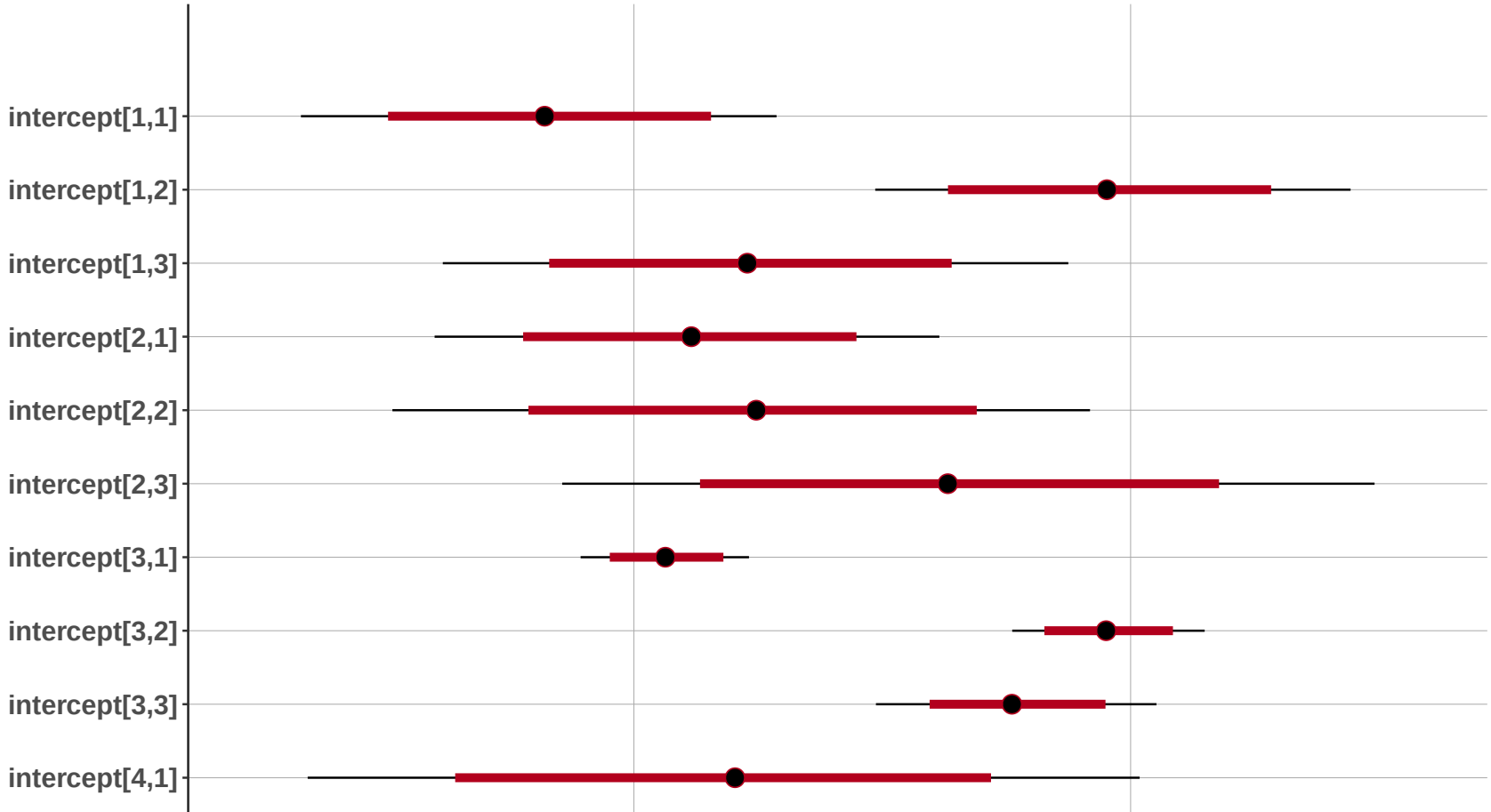
```
stan_code_5 = '
data {
  int N;
  int N_eth;
  int N_age;
  vector[N] x;
  vector[N] y;
  int eth[N];
  int age[N];
}
parameters {
  matrix[N_eth, N_age] intercept;
  matrix[N_eth, N_age] slope;
  real<lower=0> residual_sd;
  real mean_intercept;
  real<lower=0> sigma_intercept;
  real mean_slope;
  real<lower=0> sigma_slope;
}
model {
  // Likelihood
  for (i in 1:N) {
    y[i] ~ normal(intercept[eth[i], age[i]] + slope[eth[i], age[i]] * x[i], residual_sd);
  }
  // Priors
  for (j in 1:N_eth) {
    for (k in 1:N_age) {
      intercept[j,k] ~ normal(mean_intercept, sigma_intercept);
      slope[j,k] ~ normal(mean_slope, sigma_slope);
    }
  }
  mean_intercept ~ normal(11, 2);
  sigma_intercept ~ cauchy(0, 10);
  mean_slope ~ normal(0, 1);
  sigma_slope ~ cauchy(0, 10);
  residual_sd ~ cauchy(0, 10);
```

Random age/ethnicity model fits

```
## 'pars' not specified. Showing first 10 parameters by def

## ci_level: 0.8 (80% intervals)

## outer_level: 0.95 (95% intervals)
```

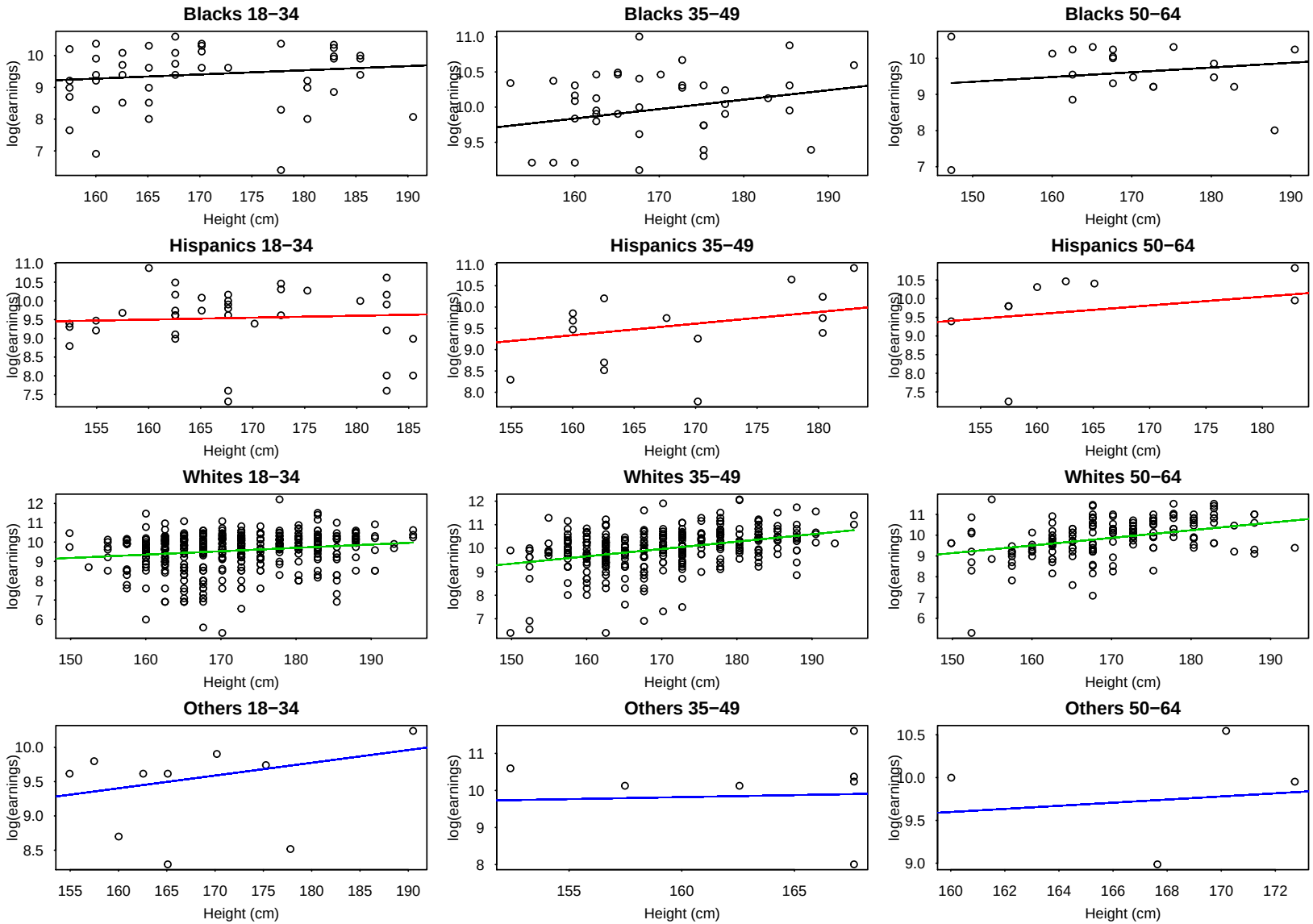


Random intercept/slopes interpretation

```
stan_run_5_summ = summary(stan_run_5)
round(stan_run_5_summ$summary, 2)
```

```
##                    mean se_mean   sd    2.5%     25%     50%     75%
## intercept[1,1]     9.41    0.00 0.13    9.16    9.33    9.41    9.50
## intercept[1,2]     9.98    0.00 0.12    9.74    9.89    9.98   10.06
## intercept[1,3]     9.62    0.00 0.16    9.31    9.51    9.61    9.73
## intercept[2,1]     9.55    0.00 0.13    9.30    9.47    9.56    9.64
## intercept[2,2]     9.62    0.00 0.18    9.26    9.51    9.62    9.74
## intercept[2,3]     9.83    0.00 0.21    9.43    9.68    9.82    9.97
## intercept[3,1]     9.53    0.00 0.04    9.45    9.50    9.53    9.56
## intercept[3,2]     9.98    0.00 0.05    9.88    9.94    9.98   10.01
## intercept[3,3]     9.88    0.00 0.07    9.74    9.84    9.88    9.93
## intercept[4,1]     9.60    0.00 0.21    9.17    9.45    9.60    9.73
## intercept[4,2]     9.93    0.00 0.24    9.49    9.77    9.92   10.09
## intercept[4,3]     9.79    0.00 0.25    9.29    9.62    9.78    9.95
## slope[1,1]         0.01    0.00 0.01   -0.01    0.01    0.01    0.02
## slope[1,2]         0.01    0.00 0.01   -0.01    0.01    0.01    0.02
## slope[1,3]         0.01    0.00 0.01   -0.01    0.01    0.01    0.02
## slope[2,1]         0.01    0.00 0.01   -0.02    0.00    0.01    0.01
## slope[2,2]         0.03    0.00 0.01    0.00    0.02    0.03    0.04
## slope[2,3]         0.02    0.00 0.01    0.00    0.01    0.02    0.03
## slope[3,1]         0.02    0.00 0.00    0.01    0.01    0.02    0.02
## slope[3,2]         0.03    0.00 0.00    0.02    0.03    0.03    0.03
## slope[3,3]         0.04    0.00 0.01    0.02    0.03    0.04    0.04
## slope[4,1]         0.02    0.00 0.01   -0.01    0.01    0.02    0.03
## slope[4,2]         0.01    0.00 0.02   -0.03    0.00    0.01    0.02
## slope[4,3]         0.02    0.00 0.02   -0.02    0.01    0.02    0.03
## residual_sd        0.87    0.00 0.02    0.84    0.86    0.87    0.89
## mean_intercept     9.73    0.00 0.10    9.53    9.66    9.72    9.79
## sigma_intercept    0.28    0.00 0.09    0.15    0.21    0.27    0.33
## mean_slope         0.02    0.00 0.01    0.00    0.01    0.02    0.02
## sigma_slope        0.02    0.00 0.01    0.00    0.01    0.01    0.02
## lp__            -336.04    0.31 5.38 -347.29 -339.45 -335.78 -332.44
```

Plotting the fitted values



More about this model

- So we now have varying effects - we should also plot the uncertainties in these lines (see practical)
- There are ways to make this model more complicated
  1. We could start partitioning up the nested random effects to have e.g. means associated with just age group or ethnicity
  2. We could start fitting multivariate random effects models ...

Multivariate models

- Often we have more than one variable or parameter that changes simultaneously
- For example, we might be interested in how both earnings and social grade change in response to height. These are likely to be correlated, even after we take into account the effect of height
- Or we might be interested in how both the slope and the intercept change in a particular model
- In either case, the *multivariate normal distribution* helps us achieve this by borrowing strength across the different dimensions

The multivariate normal distribution

- The standard normal distribution we have met has two parameters, a mean and a standard deviation/variance
- The multivariate normal (MVN) distribution has the same, except that the mean is now a *vector* and the variance is now a *matrix* sometimes called the *covariance matrix*
- Each element of the mean represents the mean of each variable. Each diagonal element of the covariance matrix is the variance of that variable, and each off-diagonal element is the covariance between those two variables
- For the 2D MVN distribution, we write:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} v_{11} & v_{12} \\ v_{12} & v_{22} \end{bmatrix}\right)$$

or, for short

$$y \sim MVN(\mu, \Sigma)$$

Learning about the parameters of the MVN

- You can think of the MVN distribution as borrowing strength between variables, just like the hierarchical model borrows strength across categories
- If we want to use the MVN in a Bayesian model we need to be able to put prior distributions on the parameters
- The means μ are easier as we can set independent prior distributions on them, e.g. $\mu_1 \sim N(0, 100)$
- The covariance matrix is more fiddly. Luckily there is a probability distribution called the *Wishart distribution* which works on covariance matrices. Stan also has some special probability distributions for covariance matrices

Returning to the earnings data

- Let's add to our complicated model with intercepts and slopes varying by ethnicity and age group
- We will include a multivariate normal prior on the intercept/slope pairs and look at the interaction between ethnicity and ages
- We are introducing a specific parameter which measures the degree of correlation between the intercept and the slope for each group. This could potentially vary between age group or ethnicity
- We will show this model but we won't run it (too slow and inelegant)

Fitting a multivariate, multi-layer model

```
stan_code_6 = '
data {
  int N;
  int N_eth;
  int N_age;
  vector[N] x;
  vector[N] y;
  int eth[N];
  int age[N];
}
parameters {
  vector[2] beta[N_eth, N_age];
  real<lower=0> residual_sd;
  vector[2] mean_beta;
  cov_matrix[2] Sigma_beta;
}
model {
  // Likelihood
  for (i in 1:N) {
    y[i] ~ normal(beta[eth[i], age[i], 1] + beta[eth[i], age[i], 2] * x[i], residual_sd);
  }
  // Priors
  for (j in 1:N_eth) {
    for (k in 1:N_age) {
      beta[j,k] ~ multi_normal(mean_beta, Sigma_beta);
    }
  }
  for (l in 1:2) {
    mean_beta[l] ~ normal(0, 10);
  }
  residual_sd ~ cauchy(0, 10);
}
'
```

Model comparison with `rstan` models

- We have already seen that you can directly use the `loo` and `waic` functions in the `loo` package to get useful Bayesian model comparison values
- Unfortunately you can't use these directly in any code you write from Stan
- If you do want to use them you need to put some extra lines at the bottom of the R code
- This is because `WAIC` and `loo` require estimates of the log likelihood in order to calculate their values

Regression example

```
stan_code_loo = '
data {
  int N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real intercept;
  real slope;
  real<lower=0> residual_sd;
}
model {
  // Likelihood
  y ~ normal(intercept + slope * x, residual_sd);
  // Priors
  intercept ~ normal(0, 20);
  slope ~ normal(0, 1);
  residual_sd ~ cauchy(0, 10);
}
generated quantities {
  vector[N] log_lik;
  for (i in 1:N) {
    log_lik[i] = normal_lpdf(y[i] | intercept + slope * x[i], residual_sd);
  }
}
'
```

Getting the loo and WAIC

```
stan_run_loo = stan(data = list(N = nrow(earnings),
                                y = earnings$y,
                                x = earnings$x_centered),
                    model_code = stan_code_loo)
library(loo)
log_lik <- extract_log_lik(stan_run_loo)
```

```
loo(log_lik)
```

```
##
## Computed from 4000 by 1059 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo  -1401.4 34.1
## p_loo         4.3  0.5
## looic      2802.8 68.1
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
```

Summary

- We have seen how to create some rich multi-layers models
- We have gone through quite a detailed example
- We have learnt about the multivariate normal distribution for even more complicated hierarchical models
- We have computed the `loo` and `waic` for a simple model