Class 3: Hierarchical regression models

Andrew Parnell
andrew.parnell@ucd.ie

Learning outcomes

- Be able to build a model by thinking about the data generating process
- Be able to choose prior distributions for regression models
- Understand how to fit a hierarchical regression model in JAGS/Stan
- Be able to compare and check the fit of regression models

Thinking about models as methods for generating data

So far we have:

- Thought about setting up models by choosing parameter values then simulating from the likelihood to get pseudo-data that should hopefully look like our real data
- We can separate this into two parts: simulating pseudo-data *before* we have seen the real data, and simulating pseudo-data *after* we have seen the real data
- The first is a way of getting good prior distributions
- The latter is our posterior predictive check of model fit

Simulating psuedo-data before the experiment

- Let's think again about the earnings data where we want to estimate log(earnings) from people's height in cm using a linear regression model with height is mean centered, i.e.

$$\log(\text{earnings}) \sim N(\alpha + \beta \times (\text{height} - \text{mean}(\text{height})), \sigma^2)$$

- Before we have seen the data, we might know that we're going to collect data on about 100 people in the range of 150 to 190 cm:

```
N = 100
x = sort(runif(N, 150, 190))
```

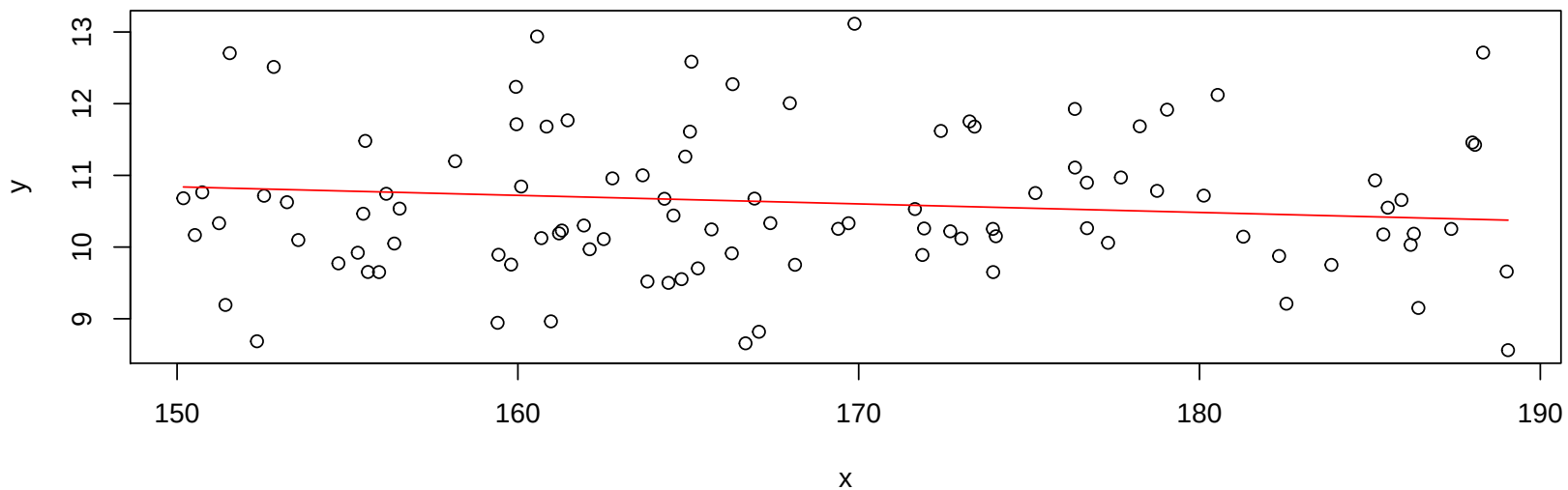
- Let's guess that the earnings are likely to be in the range of about 20k to 200k per year, so on the log scale this is from about 10 to 12

Simulating psuedo-data continued

- Before seeing the data I have no idea about whether height affects earnings, so I'd certainly be willing to consider a value of 0 for the slope. If there was a strong positive effect then people of about 190cm would be earning about 12 (on the log scale) and people of about 150cm would be earning the least. This gives a slope of about 0.05, but the opposite might also be true. To stay conservative let's a suggest a prior on the slop of \$N(0, 0.1^2)
- For the intercept we can guess that it is unlikely to be beyond the range (10, 12) so let's use $N(11, 2^2)$.
- The residual standard deviation is unlikely to be much bigger than the range of salaries so let's use $U(0, 5)$

A simulated data set

```
dat = read.csv('../data/earnings.csv')
alpha = rnorm(1, mean = 10, sd = 2)
beta = rnorm(1, mean = 0, sd = 0.1)
sigma = runif(1, 0, 5)
y = rnorm(N, alpha + beta * (x - mean(x)), sd = sigma)
plot(x, y)
lines(x, alpha + beta * (x - mean(x)), col = 'red')
```



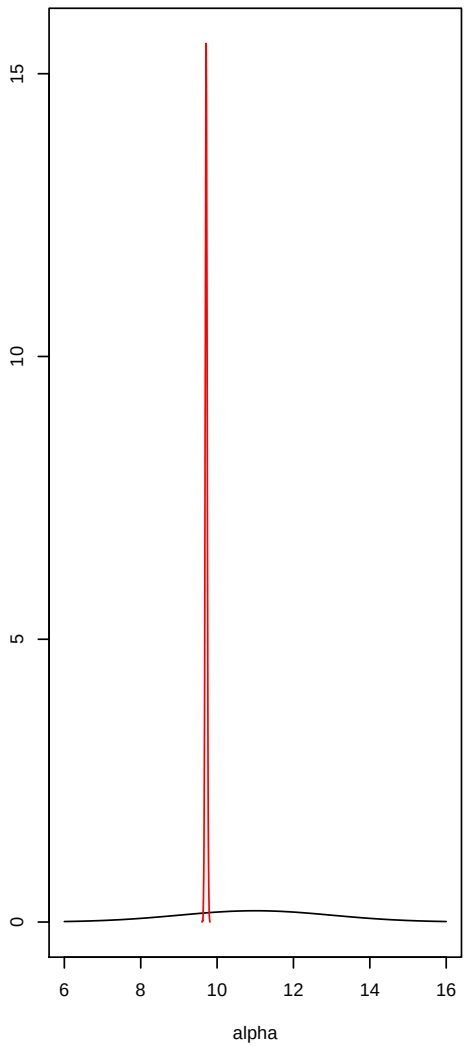
Now fit the model

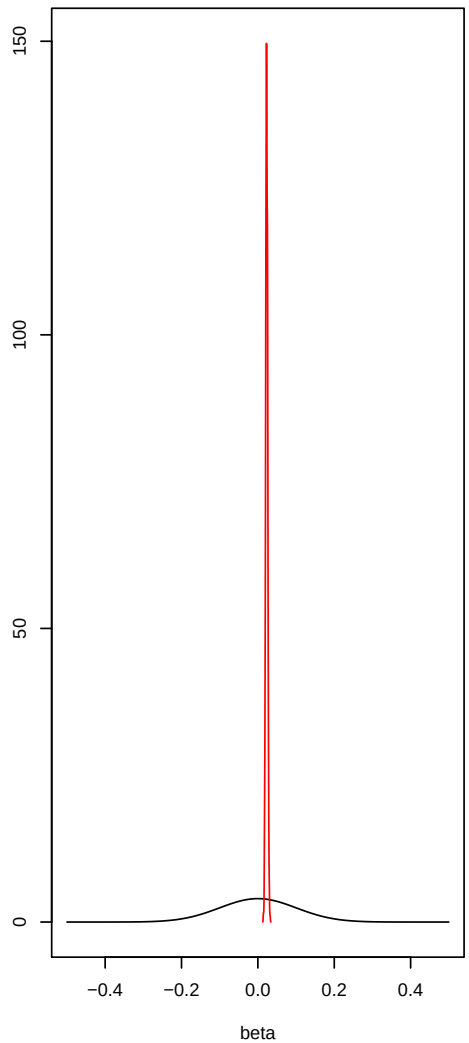
```
library(R2jags)
jags_code = '
model{
  # Likelihood
  for(i in 1:N) {
    y[i] ~ dnorm(alpha + beta*(x[i] - mean(x)), sigma^-2)
    y_sim[i] ~ dnorm(alpha + beta*(x[i] - mean(x)), sigma^-
  }
  # Priors
  alpha ~ dnorm(11, 2^-2)
  beta ~ dnorm(0, 0.1^-2)
  sigma ~ dunif(0, 5)
}
'
```

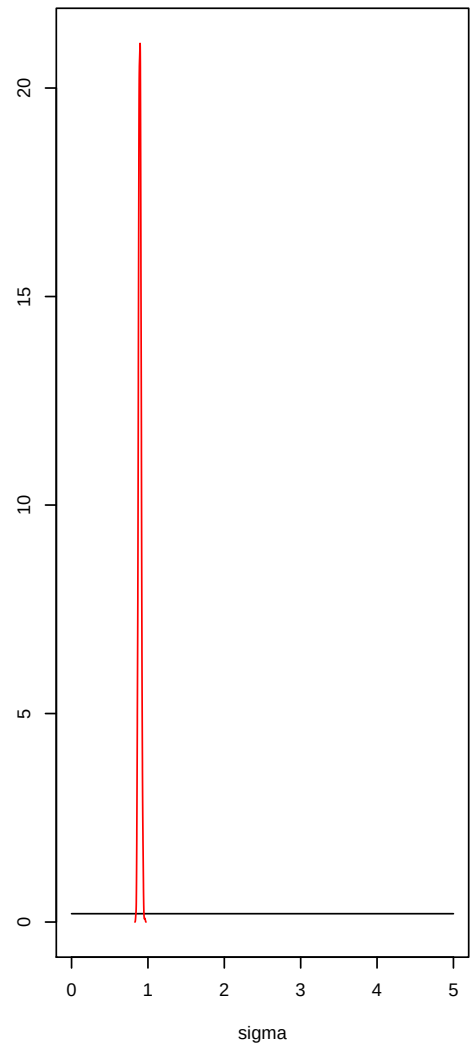
```
## module glm loaded

## Compiling model graph
```

Plot the priors and posteriors

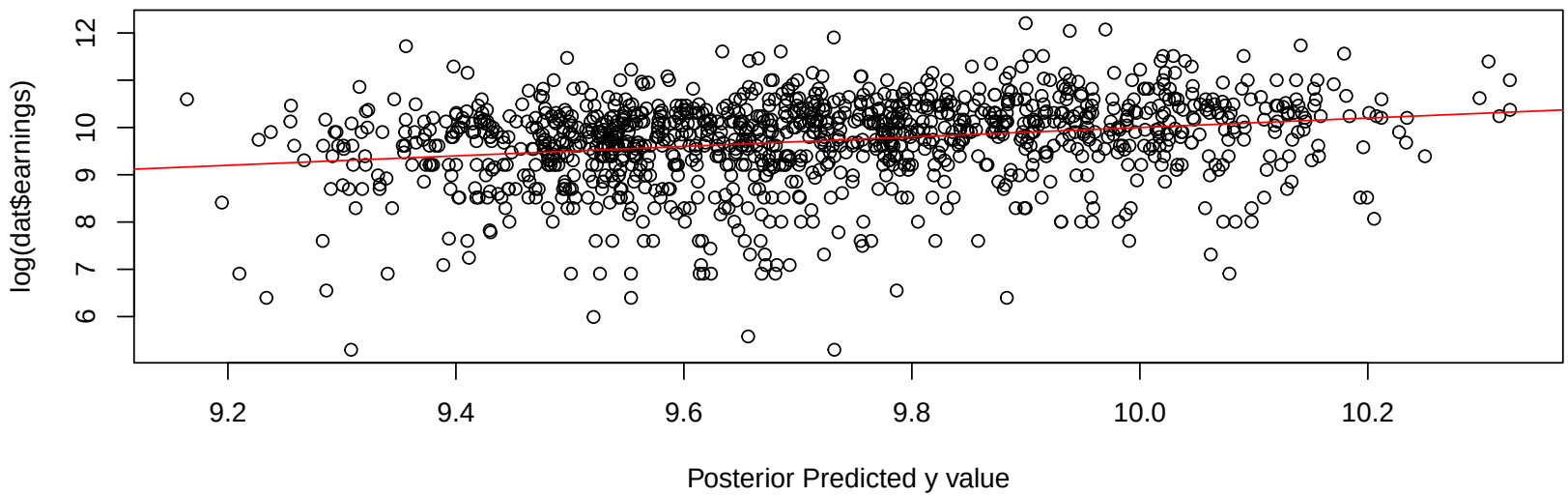






Plot the posterior predictive

```
pred = pars$y_sim
y_sim_summary = apply(pred, 2, 'quantile',
                      probs = c(0.05, 0.5, 0.95))
plot(y_sim_summary[2,], log(dat$earnings),
     xlab = 'Posterior Predicted y value')
abline(a=0, b=1, col = 'red')
```



From standard to hierarchical linear models

- Suppose now we want to see if there was a race effect on the relationship between height and earnings
- Some possible choices:
  - Divide the data up into different races and fit the model separately
  - Add race in as a covariate, this will lead to a *varying intercepts* model
  - Add race in as an interaction, this will lead to a *varying slopes* model
- An example of the varying slopes model is:

$$y_i \sim N(\alpha + \beta_{\text{race}_i}(x_i - \bar{x}), \sigma^2)$$

Writing a varying intercepts slope model in JAGS/Stan

- One way is to simply include it as an extra covariate:

```
log_earn[i] ~ dnorm(alpha + beta1 * height[i] +
                      beta2 * race_1[i]
                        + beta3 * race_2[i] + ...,
                        sigma^-2)
```

- A neater way is to include it as part of the definition of the slope:

```
log_earn[i] ~ dnorm(alpha + beta[race[i]] * height[i],
                      sigma^-2)
```

- This means that instead of there being a single `beta` value there are multiple different values, according to when `race[i]` is value 1, 2, 3, or 4
- We can easily extend to varying intercepts and slopes:

```
log_earn[i] ~ dnorm(alpha[race[i]] +
                      beta[race[i]] * height[i],
                        sigma^-2)
```

Prior distributions for varying intercepts and slopes

- Let's think about our different choices for prior distributions on these varying parameters
- Remember we now have four values `alpha[1]`, `alpha[2]`, ... and `beta[1]`, `beta[2]`, ... i.e. intercepts and slopes depending on race
- One choice is to give totally independent prior distributions, e.g. $\alpha_1 \sim N(11, 2^2), \alpha_2 \sim N(11, 2^2), ..., \beta_1 \sim N(0, 0.1^2), \beta_2 \sim N(0, 0.1^2)$,...
- However, the above can fall down when there are very small numbers in some of the groups
- A better choice is to put a prior distribution on the alpha and beta values that *ties them together* and add some extra parameters. For example, $\alpha_{\text{race}_i} \sim N(\mu_\alpha, \sigma_\alpha^2)$, and $\beta_{\text{race}_i} \sim N(\mu_\beta, \sigma_\beta^2)$
- Under this model we can *borrow strength* between the different races, and put extra prior distributions on the new parameters $\mu_\alpha, \mu_\beta, \sigma_\alpha, \sigma_\beta$

Our first hierarchical model!

```
jags_code = '
model{
  # Likelihood
  for(i in 1:N) {
    y[i] ~ dnorm(alpha[race[i]] +
                 beta[race[i]]*(x[i] - mean(x)),
                   sigma^-2)
  }
  # Priors
  for(j in 1:N_race) {
    alpha[j] ~ dnorm(mu_alpha, sigma_alpha^-2)
    beta[j] ~ dnorm(mu_beta, sigma_beta^-2)
  }
  mu_alpha ~ dnorm(11, 2^-2)
  mu_beta ~ dnorm(0, 0.1^-2)
  sigma ~ dunif(0, 5)
  sigma_alpha ~ dunif(0, 2)
  sigma_beta ~ dunif(0, 2)
}
'
```

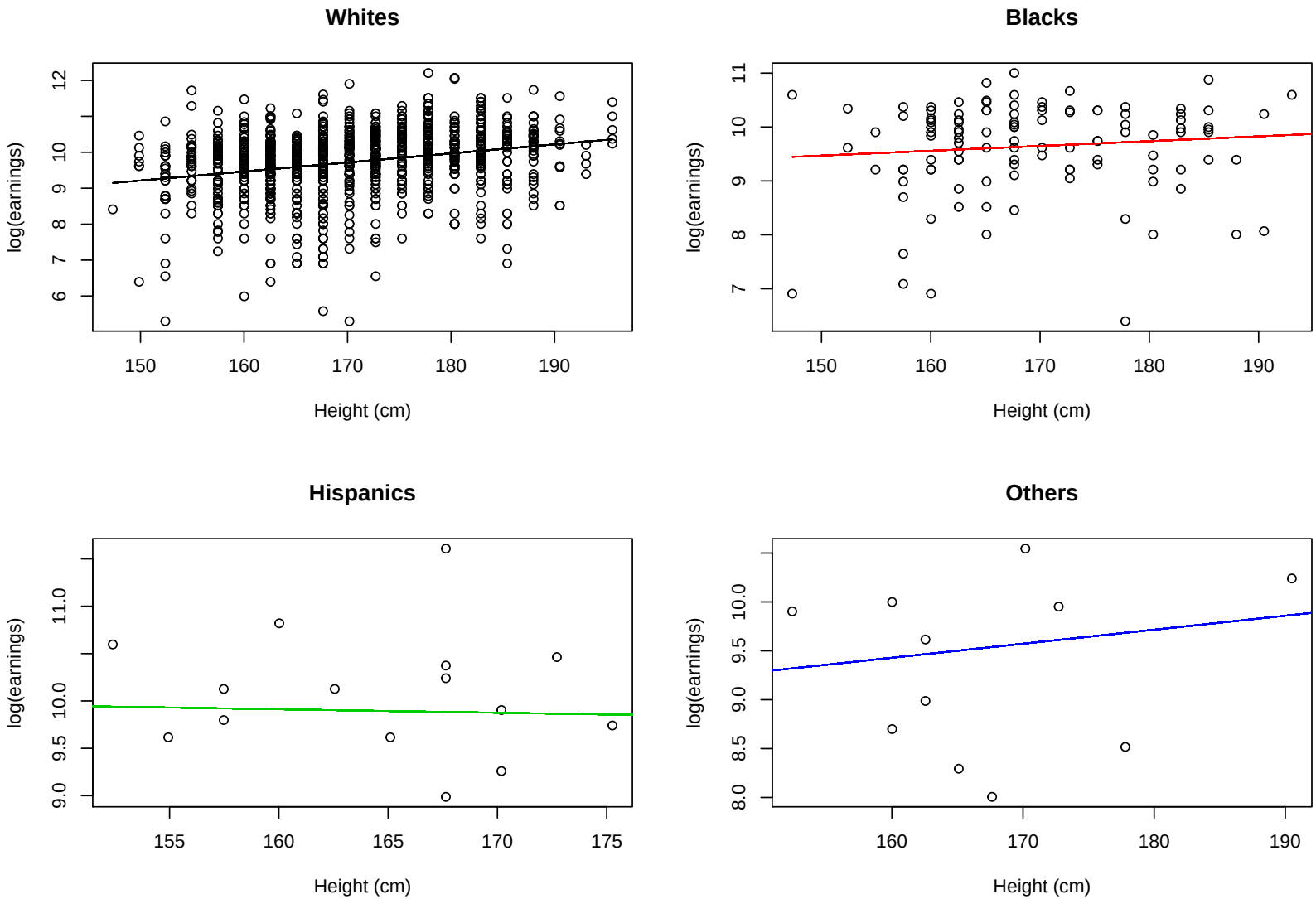
About the hierarhical model

- The parameters μ_α and μ_β have useful interpretations: they are the overall mean intercept and the overall mean slope for all observations removing the effect of race
- The parameters σ_α, and σ_β have useful interpretations too: they are the variability in the intercept and slope between race
- You may have met this type of model in other modelling courses when it is occasionally called *random effects* modelling. More on this later
- It's a good idea to try and draw a DAG for hierarchical models

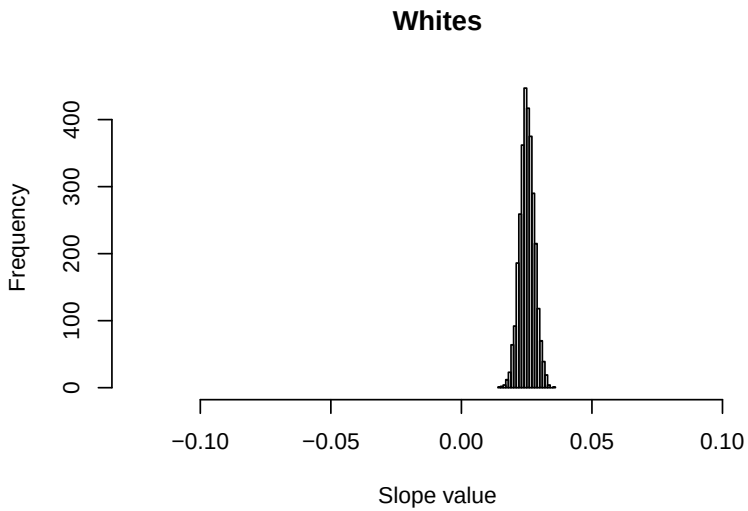
Running the model

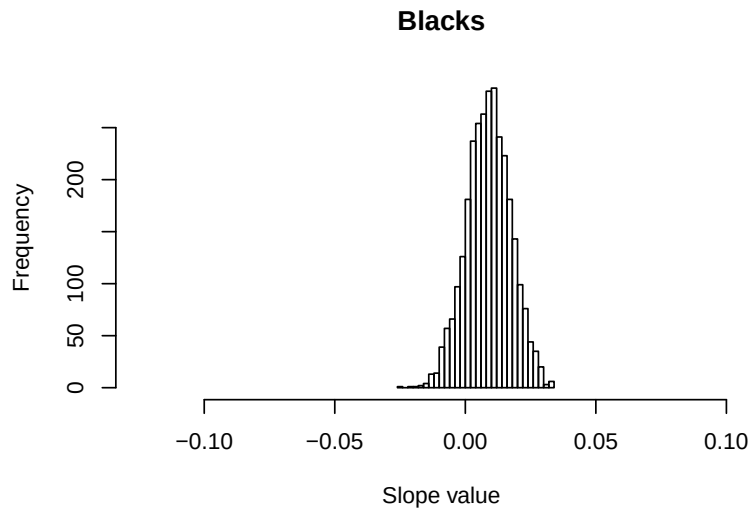
```
jags_run = jags(data = list(N = nrow(dat),
                            y = log(dat$earnings),
                            race = dat$race,
                            N_race = length(unique(dat$race)),
                            x = dat$height_cm),
                parameters.to.save = c('alpha',
                                       'beta',
                                       'sigma',
                                       'mu_alpha',
                                       'mu_beta',
                                       'sigma_alpha',
                                       'sigma_beta'),
                model.file = textConnection(jags_code))
```

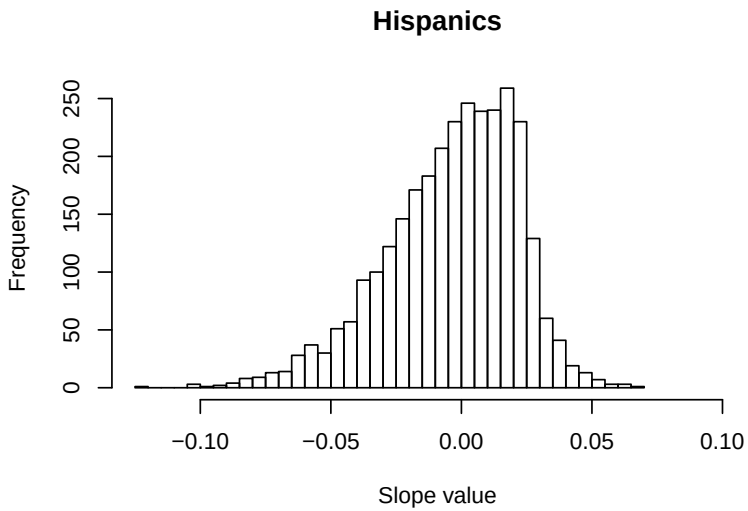
Plotting the output - mean slopes

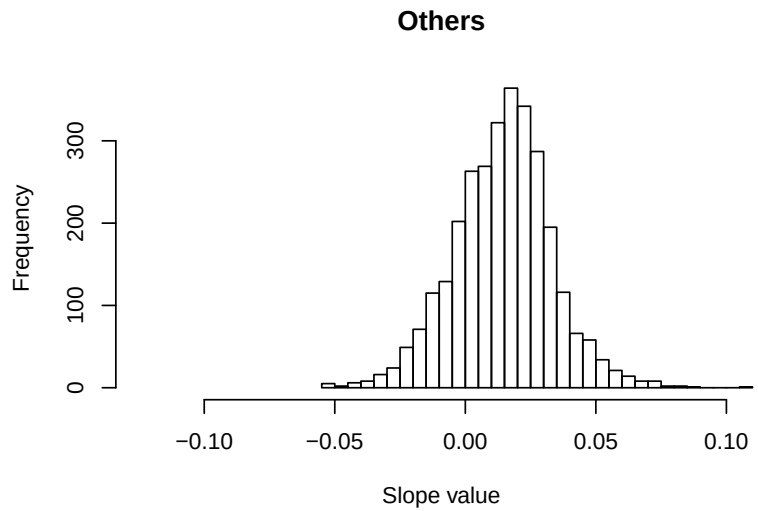


Are the slopes different?









Setting up hierarchical prior distributions

- In general a hierarchical model is created whenever we put prior distributions on prior distributions
- The lower level parameters in the model are often called *hyper-parameters*
- The normal distribution is by far the most common probability distribution, but there are others which can also be useful, some of which we will come to later in the course
- Because the parameters in the higher levels of the hierarchy have fewer values (e.g. only 4 slopes and intercepts) if there are explanatory variables that work on the level of race, we can include them at this level, rather than at the level of the observations. Again, more on this later.

A note on priors for standard deviations

- We set our prior distributions on the standard deviations parameters as Uniform
- However, not all standard deviation parameters are created equal!
- The standard deviation of the residuals, here σ, is estimated from N observations so is quite well defined
- By contrast the standard deviations of alpha and beta, σ_α and σ_β are estimated based on just the four different race groups
- We therefore need to be a bit more careful with prior distributions for these parameters. The usual recommendation is to use a half-t-distribution on these parameters rather than a uniform

Side note: causation vs prediction as goals

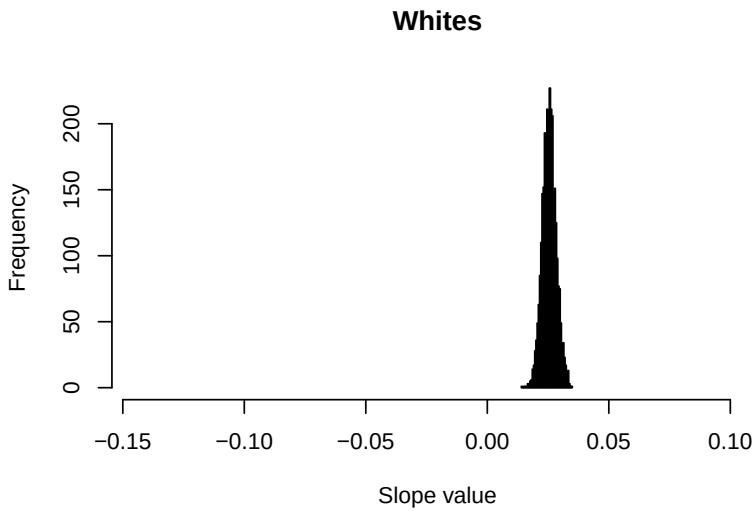
- When fitting models, there are often two competing goals
- One is simply to find the model that best *predicts* the response variable from the explanatory variable. Hierarchical models are good at this because we can allow lots of parameters to vary in complex ways
- The other is to find which explanatory variables *causes* the response variable. This is arguably impossible from a mathematical model but this is often what people want (and mistakenly use p-values for). Hierarchical are good at this too because they stop us from over-estimating effects

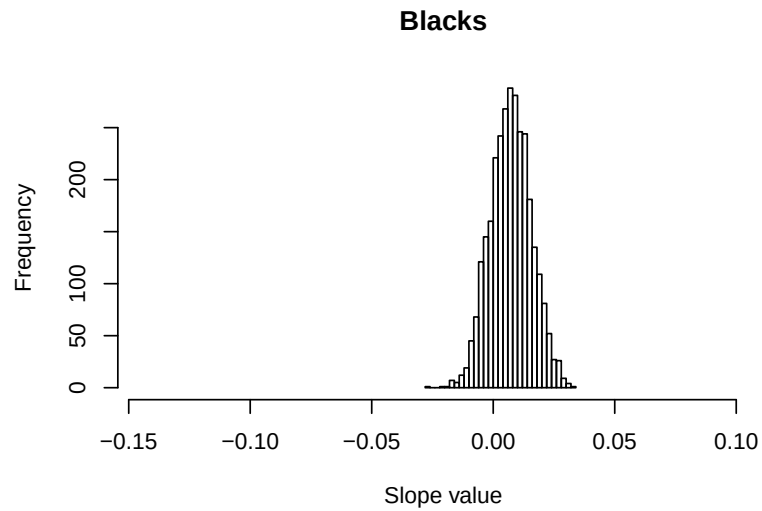
Over-estimating effects

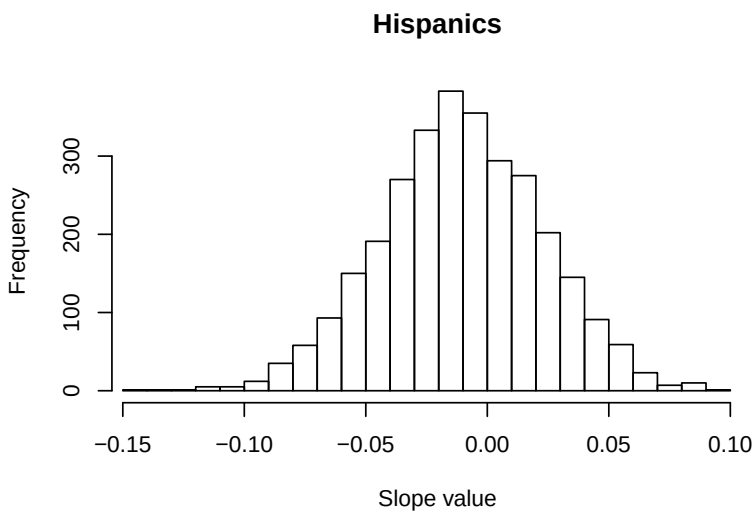
- Suppose we had fitted the independent priors version of the model:

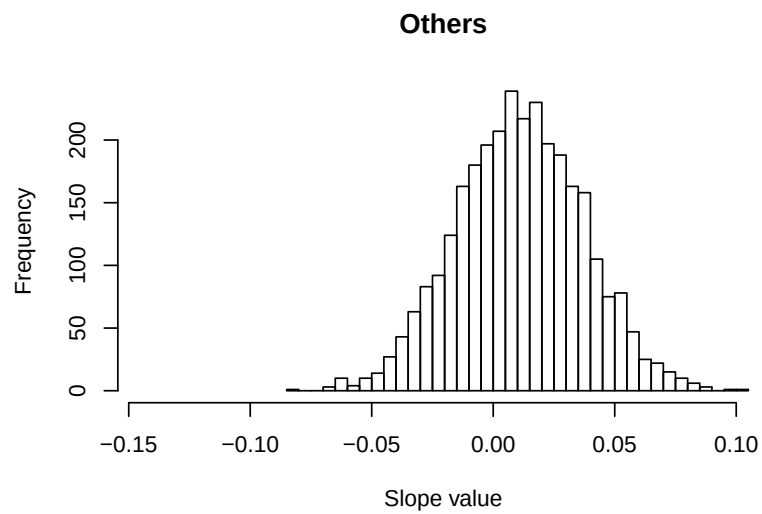
```
jags_code = '
model{
  # Likelihood
  for(i in 1:N) {
    y[i] ~ dnorm(alpha[race[i]] +
                 beta[race[i]]*(x[i] - mean(x)),
                   sigma^-2)
  }
  # Priors
  for(j in 1:N_race) {
    alpha[j] ~ dnorm(11, 10^-2)
    beta[j] ~ dnorm(0, 0.1^-2)
  }
  sigma ~ dunif(0, 5)
}
```

Results


Whites
Frequency
0
50
100
150
200
−0.15
−0.10
−0.05
0.00
0.05
0.10
Slope value


Blacks
Frequency
0
50
100
200
−0.15
−0.10
−0.05
0.00
0.05
0.10
Slope value


Hispanics
Frequency
0
100
200
300
−0.15
−0.10
−0.05
0.00
0.05
0.10
Slope value


Others
Frequency
0
50
100
150
200
−0.15
−0.10
−0.05
0.00
0.05
0.10
Slope value

Model comparison

Model comparison: an introduction

- We can come up with the fanciest model in the world but if it does not meet our desired goals (either prediction or causation) then we cannot publish or use it
- You can broadly split model comparison into two parts: *absolute* model comparison and *relative* model comparison
- In absolute model comparison, we are looking at how well a specific model fits the data at hand
- In relative model comparison, we can only look at how well a set of models performs on the same data with the goal of choosing the best one (or group)

Relative model comparison: model information criteria

- You might have come across these before: Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC)
- The general idea is that the score on the likelihood is a good measure of model fit, except for the fact that more complex models will generally have higher likelihood scores
- If we penalise these scores by some measure of the complexity of the model then we can compare models across complexities
- The usual measure of complexity is some function of the number of parameters
- Because these are relative model comparisons, the best model acording to an IC might still be useless!

Different types of information criteria

- For various historical reasons, people tend to transform the likelihood score into the *deviance*, which is minus twice the log-likelihood score
- They then add a model complexity term onto it
- The two most common ICs are:

$$\text{AIC} : -2 \log L + 2p$$

$$\text{BIC} : -2 \log L + p \log n$$

  where p is the number of parameters and n is the number of observations
- We usually pick the smallest values of these across different models

Information criteria for Hierarchical models

- For Bayesian models it's hard to know which value of L to use, seeing as at each iteration we get a different likelihood score.
- Two specific versions of IC have been developed for these situations
- The first, called the *Deviance Information Criteria* (DIC) is calculated via:

$$\text{DIC} : -2 \log L_{\text{max}} + 2p_D$$

  where p_D is the *effective number of parameters*
- The second called the Widely Applicable Information Criterion (WAIC) which is calculated as:

$$\text{WAIC} : -2 \log L_{\text{max}} + p_{\text{WAIC}}$$

- Here p_{WAIC} is a measure of the variability of the likelihood scores

Which information criterion should I use?

- WAIC and DIC are built for Bayesian hierarchical models
- DIC is included by default in the `R2jags` package
- WAIC is included in the `loo` package which is installed alongside Stan
- WAIC is considered superior as it also provides uncertainties on the values. Most of the others just give a single value
- More generally there is a philosophical argument about whether we ever want to choose a single best model

Absolute model criteria: cross validation

- Cross validation (CV) works by:

1. Removing part of the data,
2. Fitting the model to the remaining part,
3. Predicting the values of the removed part,
4. Comparing the predictions with the true (left-out) values

- It's often fitted repeatedly, as in k-fold CV where the data are divided up into k groups, and each group is left out in turn
- In smaller data sets, people perform leave-one-out cross-validation (LOO-CV)

Example: 5-fold cross-validation

```
jags_code = '
model{
  # Likelihood
  for(i in 1:N) {
    y[i] ~ dnorm(alpha +
                   beta*(x[i] - mean(x)),
                     sigma^-2)
  }
  for(j in 1:N_pred) {
    y_pred[j] ~ dnorm(alpha +
                        beta*(x_pred[j] - mean(x)),
                          sigma^-2)
  }
  alpha ~ dnorm(11, 2^-2)
  beta ~ dnorm(0, 0.1^-2)
  sigma ~ dunif(0, 5)
}
'
```

Example continued

```
folds = sample(rep(1:5, length = nrow(dat)))
rmsep = rep(NA, nrow(dat))
for(i in 1:5) {
  curr_data = subset(dat, folds != i)
  curr_pred = subset(dat, folds == i)
  jags_run = jags(data = list(N = nrow(curr_data),
                              y = log(curr_data$earnings),
                              x = curr_data$height_cm,
                              N_pred = nrow(curr_pred),
                              x_pred = curr_pred$height_cm),
                  parameters.to.save = 'y_pred',
                  model.file = textConnection(jags_code))
  out_pred = jags_run$BUGSoutput$mean$y_pred
  rmsep[folds ==i] = (log(curr_pred$earnings) - out_pred)^2
}
mean(rmsep)
```

Pros and cons of CV

- We might also run the 5-fold CV on the previous slide for different complexity models and see which had the smallest root mean square error of prediction (RMSEP), i.e. use it as a relative criteria
- CV is great because it actually directly measures the performance of the model on real data, based on data the model hasn't seen
- However, it's computationally expensive, and problems occur in hierarchical models if some groups are small, and therefore might get left out of a fold

Other absolute model comparison

- We've already met posterior predictive distributions, which is essentially leave none out CV.
- Another popular one is something called the *Bayes Factor*. This is created by first calculating the posterior distribution of a model given the data, a measure of absolute model fit. The ratios of these can be compared for different models to create a relative model criteria.
- However, Bayes Factors are really hard to calculate and often overly sensitive to irrelevant prior choices

Continuous model expansion

- There are lots of clever ways to set up prior distributions so that a model choice step is part of the model fit itself
- One way is partial pooling, by which we force e.g. varying slope and intercept parameters to the same value (or not)
- Another way is to put shrinkage or selection priors on the parameters in the model, possibly setting them to zero
- More on all of these later in the course

Summary

- A hierarchical model is one where there are extra layers in the model, priors on priors
- For simple regression models, we can having varying slopes and intercepts by group, and borrow strength even when we have few observations for a group
- We always need to be careful with priors, especially on hyper-parameters
- There are lots of good methods for comparing between models, especially WAIC and DIC