Notes about homework

- Please submit 2 files (at least):
    1. Code
    2. Brief explanation, justification, and discussion

I'm interested in how your are thinking through things and interpreting them, not just in if you can get results!!!

- Your code should be different from that in class

Metric Predicted Variable With One Nominal Predictor Variable

Tim Frasier

Goals & General Idea

Goals

When would we use this type of analysis?

- When we want to know the effect of being in a group (based on one time of variable) on some metric predictor variable
- Very common type of data set
  - Comparison between two groups (*t*-test)
  - Comparison among several groups (ANOVA)
  - etc.

General Idea

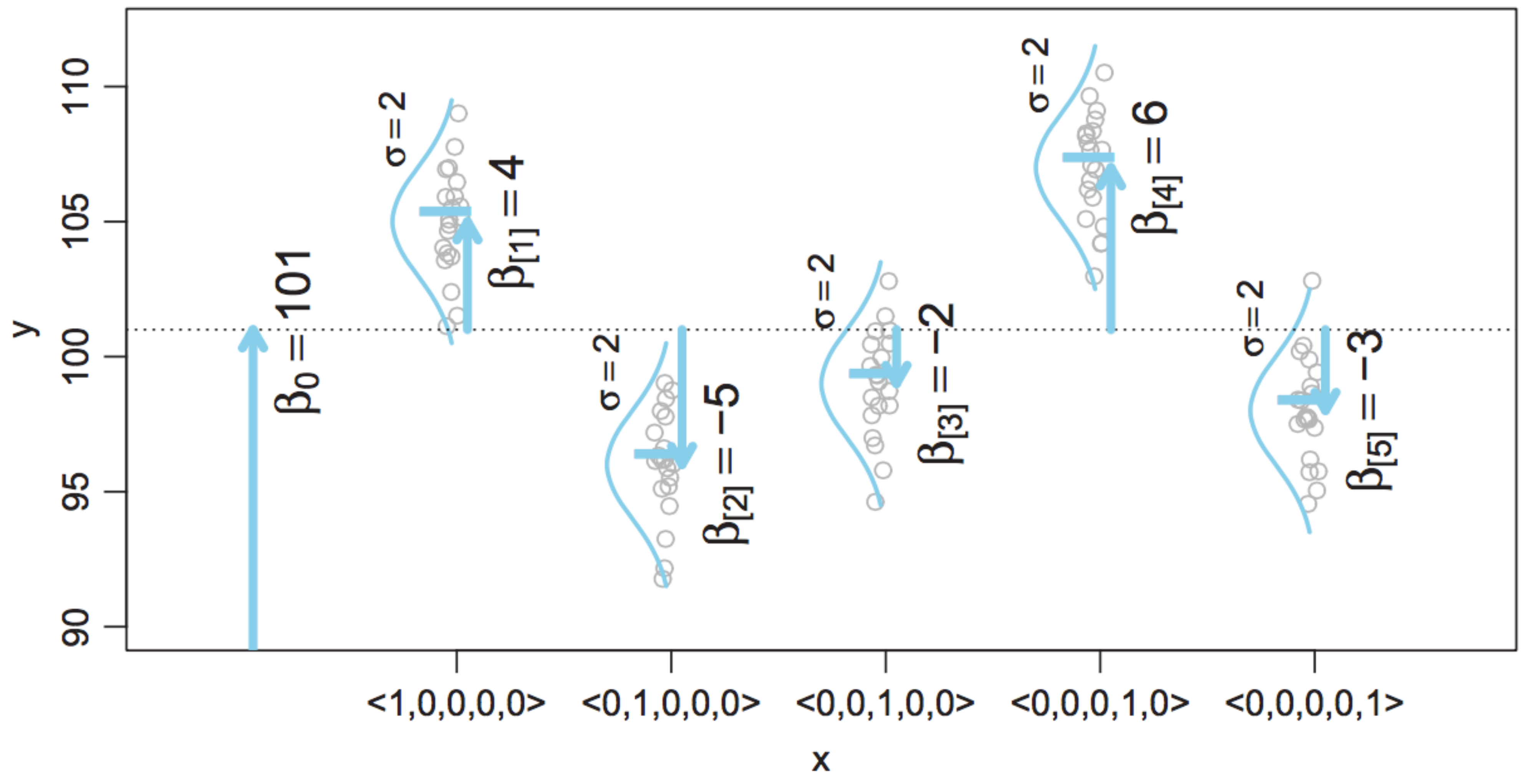
- Trying to quantify the relationship between two different sets of data
  - One (`y`) is the metric response (or predicted) variable
  - The other (`x`) is the nominal predictor variable that represents the categories in which measurements, samples, individuals can belong

Equation

- Are actually several ways to parameterize these types of models

Equation: First Parameterization

$$y = \beta_0 + \sum \beta_{[j]} x_{[j]} \quad \text{or} \quad y = \beta_0 + \overrightarrow{\beta_j} \cdot \overrightarrow{x_j}$$



Kruschke (2015) p. 555

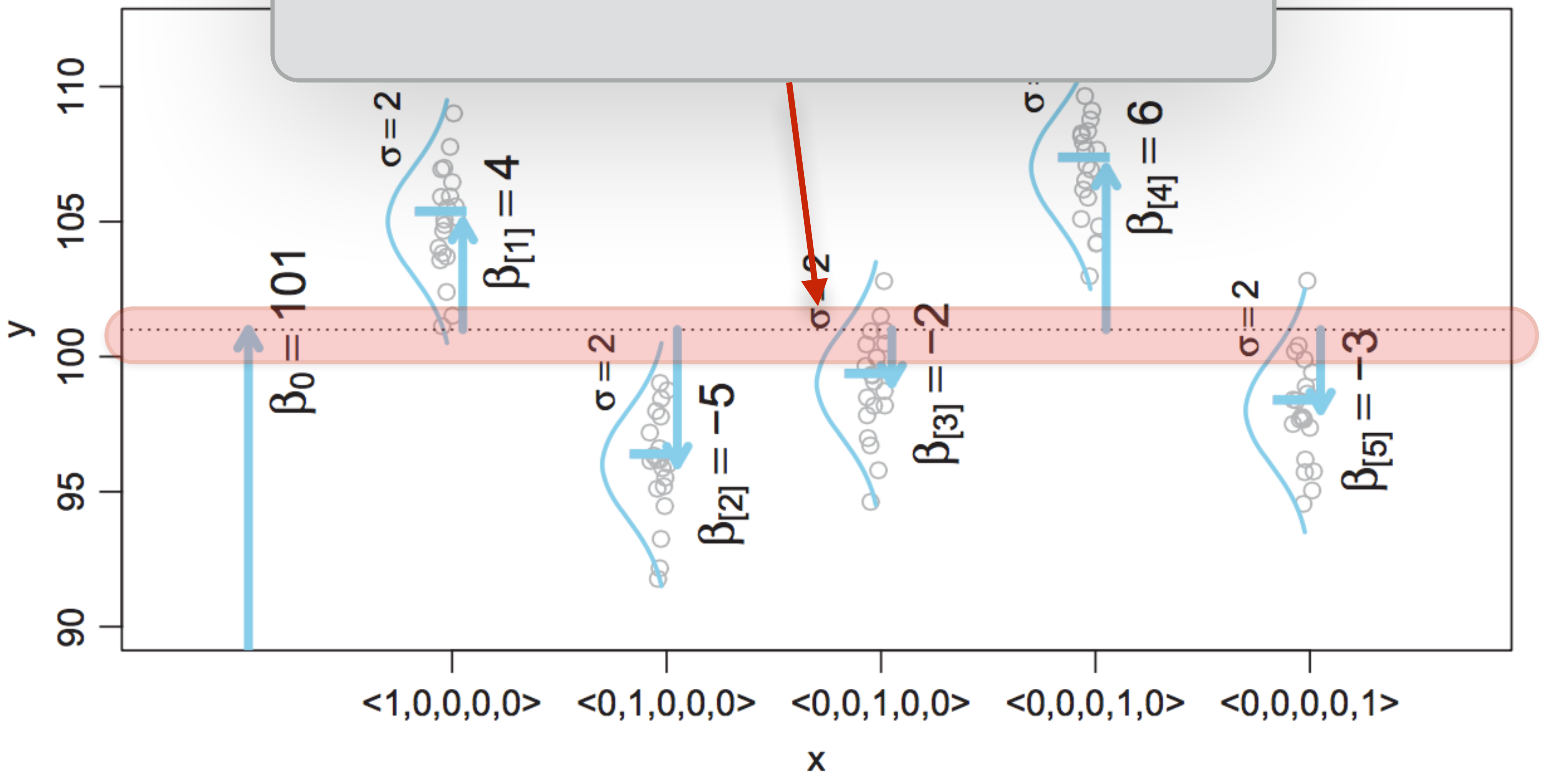
Equation: First Parameterization

$$y = \boxed{\beta_0} + \sum \beta_{[j]} x_{[j]}$$

or

$$y = \boxed{\beta_0} + \vec{\beta_j} \cdot \vec{x_j}$$

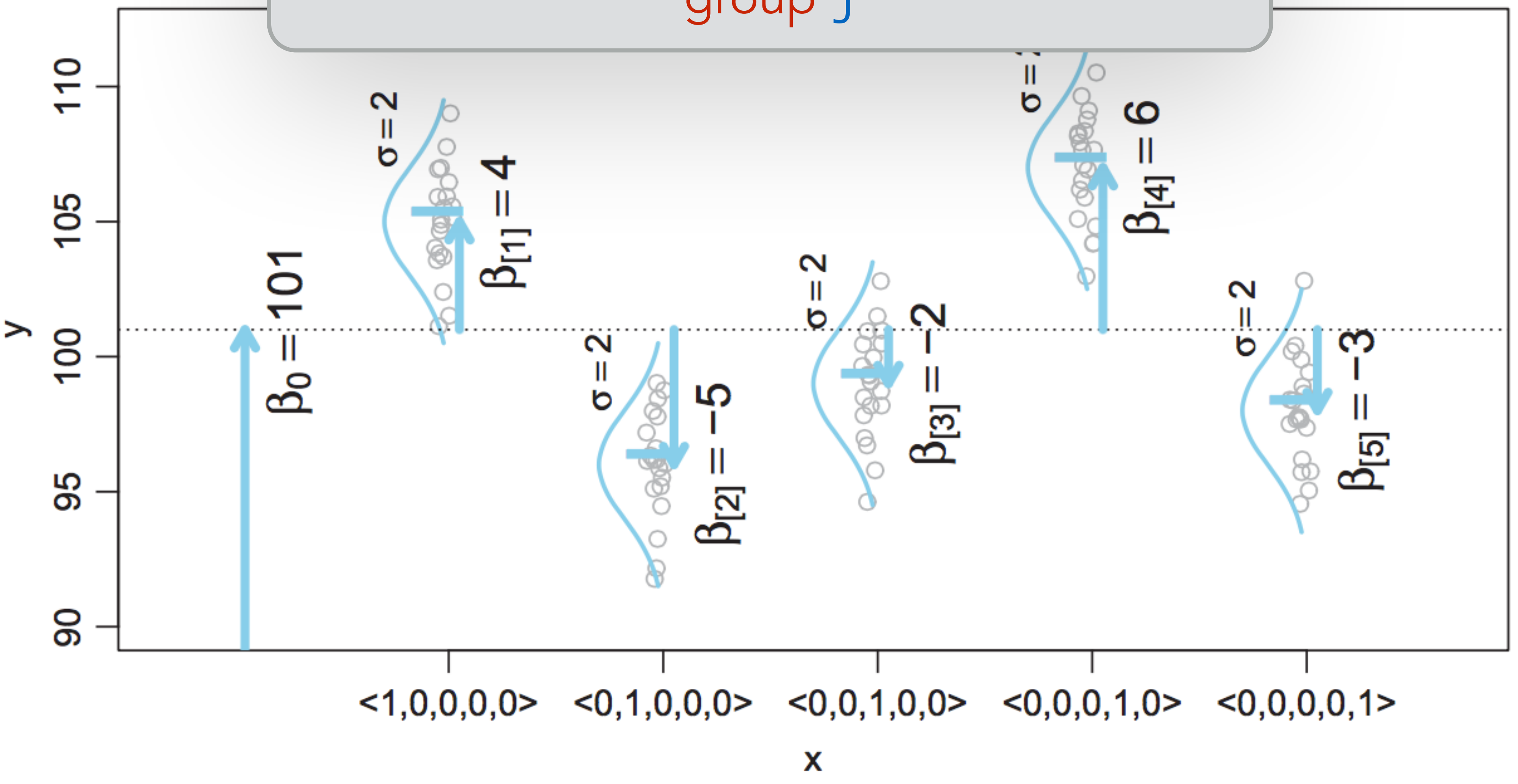
Mean value of y, across all groupings



Equation: First Parameterization

$y = \beta_0 + \sum \boxed{\beta_{[j]}} x_{[j]}$ or $y = \beta_0 + \boxed{\vec{\beta}_j} \cdot \vec{x}_j$

Degree to which values are deflected above or below mean value, based on being in group j



Kruschke (2015) p. 555

Equation: Second Parameterization

$$y = \beta_{[j]} x_{[j]}$$

$$y = \vec{\beta_j} \cdot \vec{x_j}$$

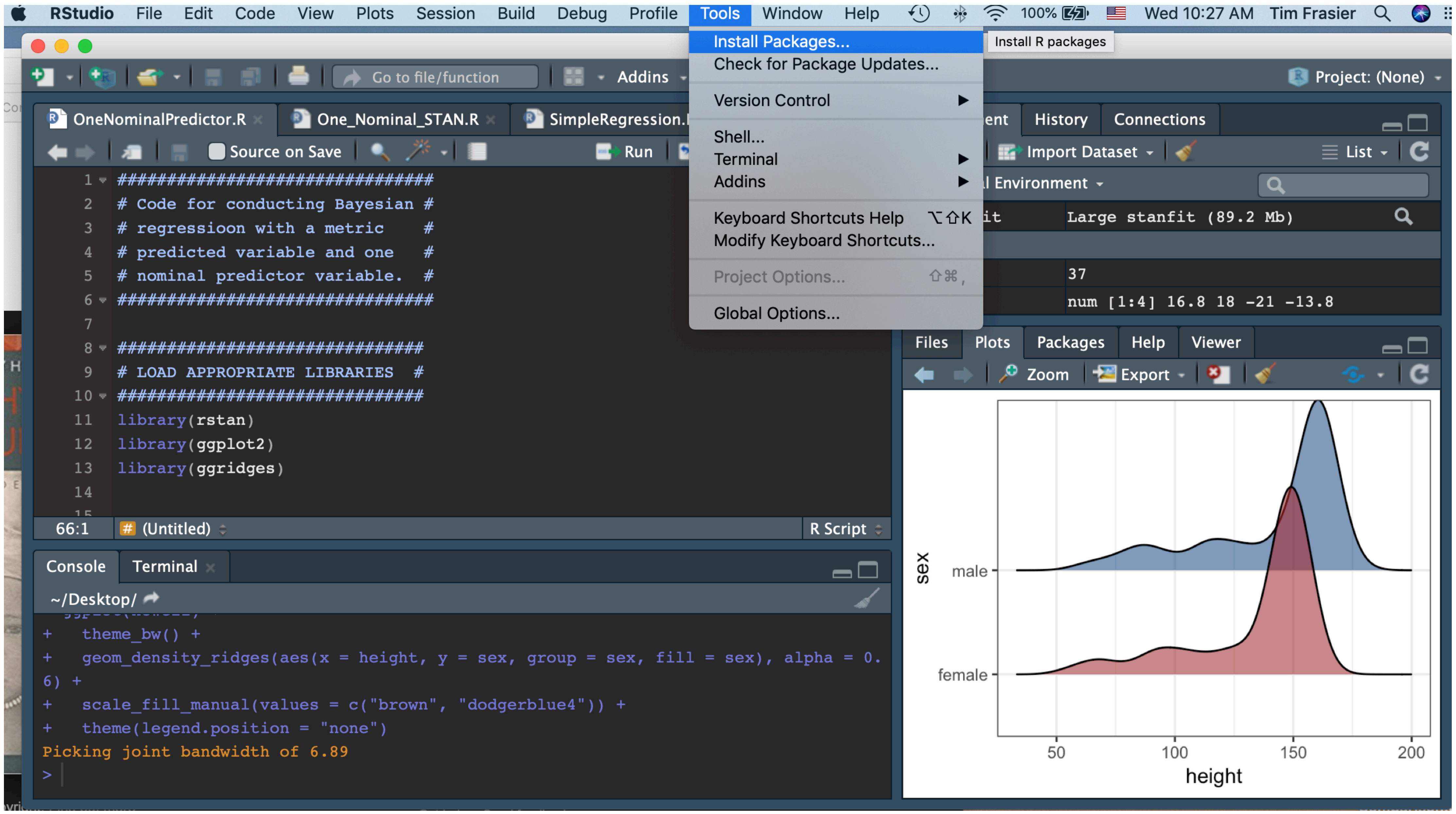
Equation: Second Parameterization

$$y = \boxed{\beta_{[j]}} x[j] \qquad y = \boxed{\vec{\beta_j}} \cdot \vec{x_j}$$

Just estimate a different value for each of `j` groups

Preparing R

- Install the `ggridges` package



```
################################
# Code for conducting Bayesian #
# regressioon with a metric    #
# predicted variable and one   #
# nominal predictor variable.  #
################################

###############################
# LOAD APPROPRIATE LIBRARIES  #
###############################
library(rstan)
library(ggplot2)
library(ggridges)
```

```
Large stanfit (89.2 Mb)
37
num [1:4] 16.8 18 -21 -13.8
```

```
+   theme_bw() +
+   geom_density_ridges(aes(x = height, y = sex, group = sex, fill = sex), alpha = 0.
6) +
+   scale_fill_manual(values = c("brown", "dodgerblue4")) +
+   theme(legend.position = "none")
Picking joint bandwidth of 6.89
>
```

Preparing R

- Load the appropriate libraries and code

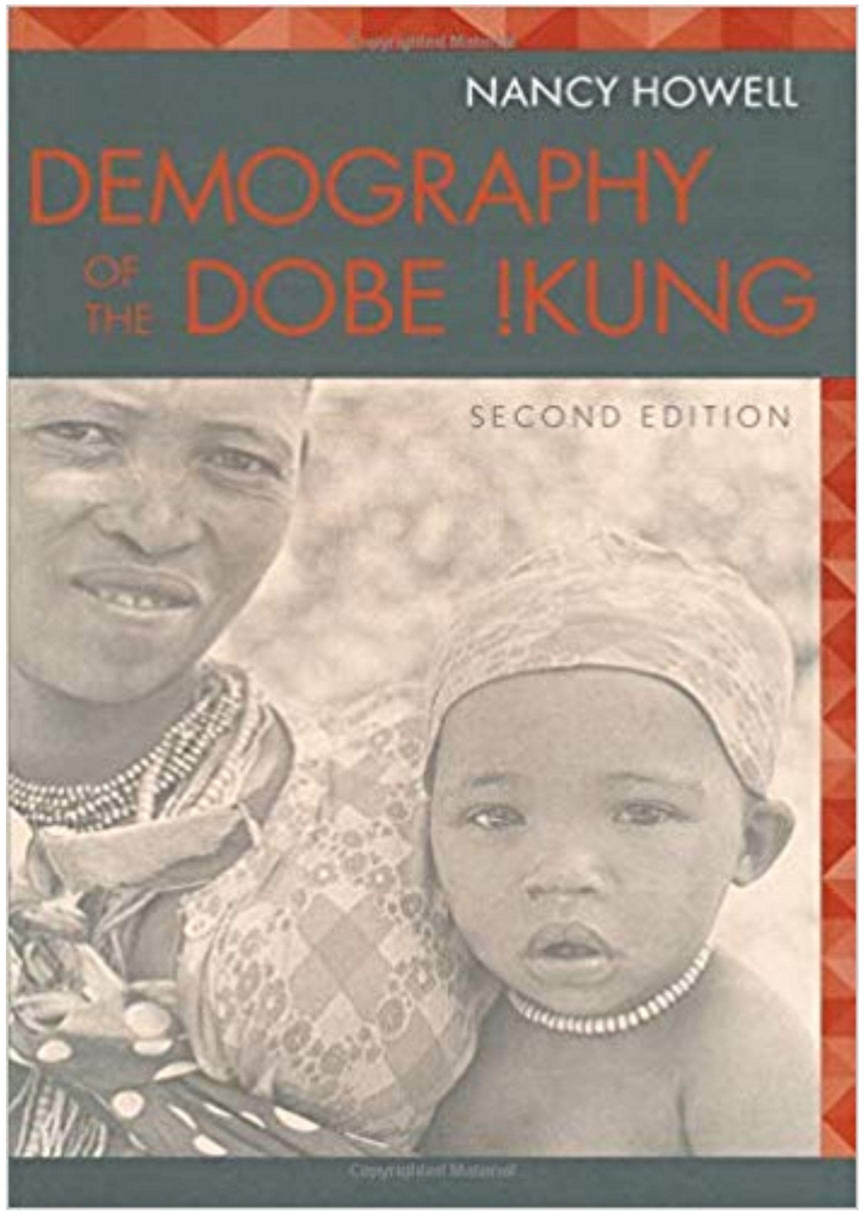
```
library(rstan)
library(ggplot2)
library(ggridges)
source("plotPost.R")
```

The Data

Data

- Data on height, weight, age, and sex of the Dobe area !Kung San people (Howell 2000, 2010)

- We'll just look at height as it relates to sex for now



Howell N (2000) *Demography of the Dobe !Kung*. Aldine de Gruyter, New York
Howell N (2010) *Life Histories of the Dobe !Kung: Food, Fatness, and Well-being over the Life-span*. Origins of Human Behavior and Culture. University of California Press.
Data distributed as part of the "`rethinking`" R package

Data

- Read the data into R and examine

```
howell = read.table("Howell.csv", header = TRUE, sep = “,")


str(howell)

'data.frame':  544 obs. of  4 variables:
 $ height: num  152 140 137 157 145 ...
 $ weight: num  47.8 36.5 31.9 53 41.3 ...
 $ age   : num  63 63 65 41 51 35 32 27 19 54 ...
 $ sex   : Factor w/ 2 levels "female","male": 2 1 1 2 1 2 1 2 1 2 ...
```

Data

- Read the data into R and examine

```
howell = read.table("Howell.csv", h

str(howell)

'data.frame':  544 obs. of  4 variables:
 $ height: num  152 140 137 157 145 ...
 $ weight: num  47.8 36.5 31.9 53 41.3 ...
 $ age   : num  63 63 65 41 51 35 32 27 19 54 ...
 $ sex   : Factor w/ 2 levels "female","male": 2 1 1 2 1 2 1 2 1 2 ...
```

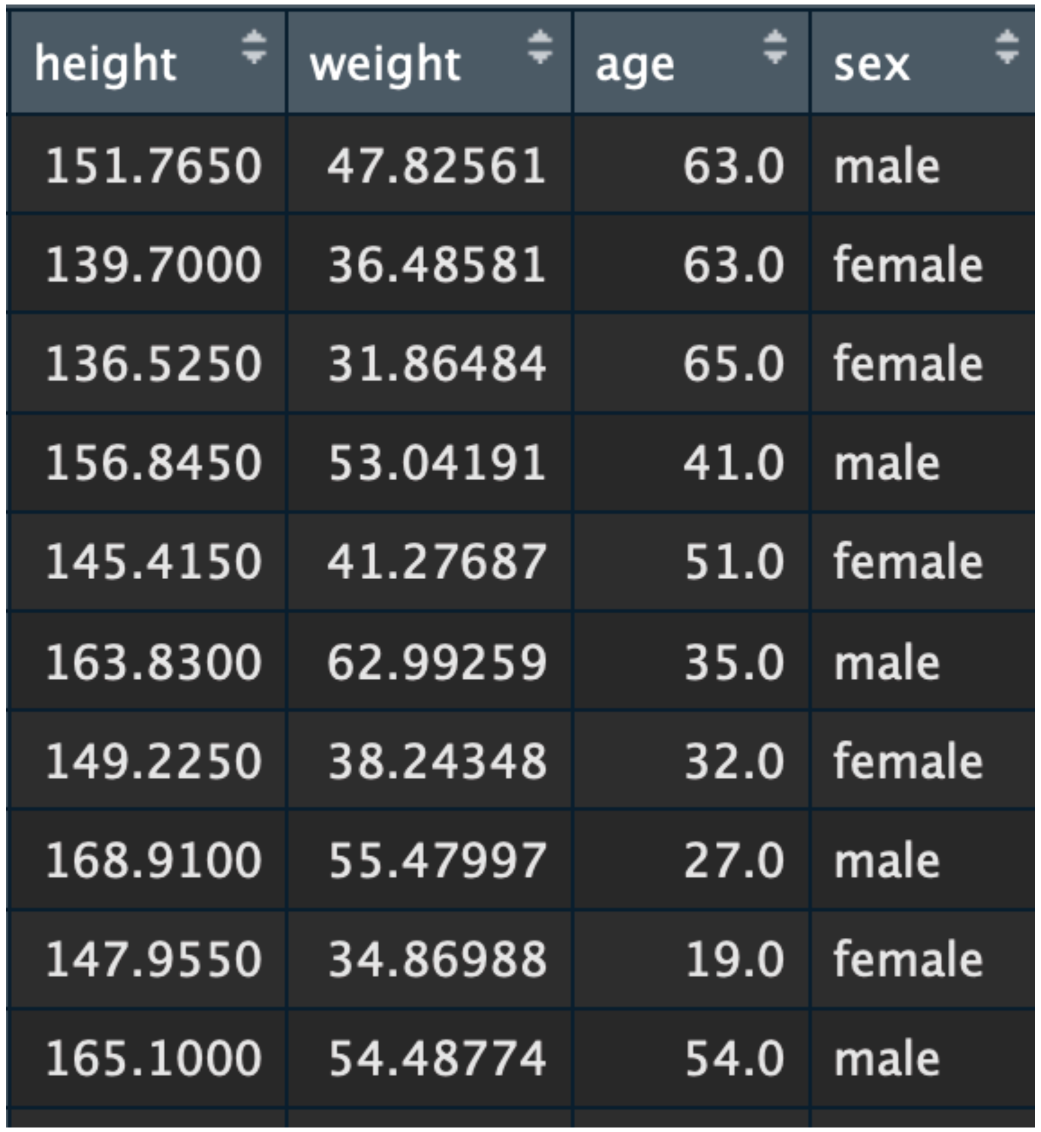
Important!!! Categorical variables must always be factors!

Data

- Read the data into R and examine

```
summary(howell)

     height           weight            age             sex     
 Min.   : 53.98   Min.   : 4.252   Min.   : 0.00   female:287  
 1st Qu.:125.09   1st Qu.:22.008   1st Qu.:12.00   male  :257  
 Median :148.59   Median :40.058   Median :27.00               
 Mean   :138.26   Mean   :35.611   Mean   :29.34               
 3rd Qu.:157.48   3rd Qu.:47.209   3rd Qu.:43.00               
 Max.   :179.07   Max.   :62.993   Max.   :88.00               
```

Data

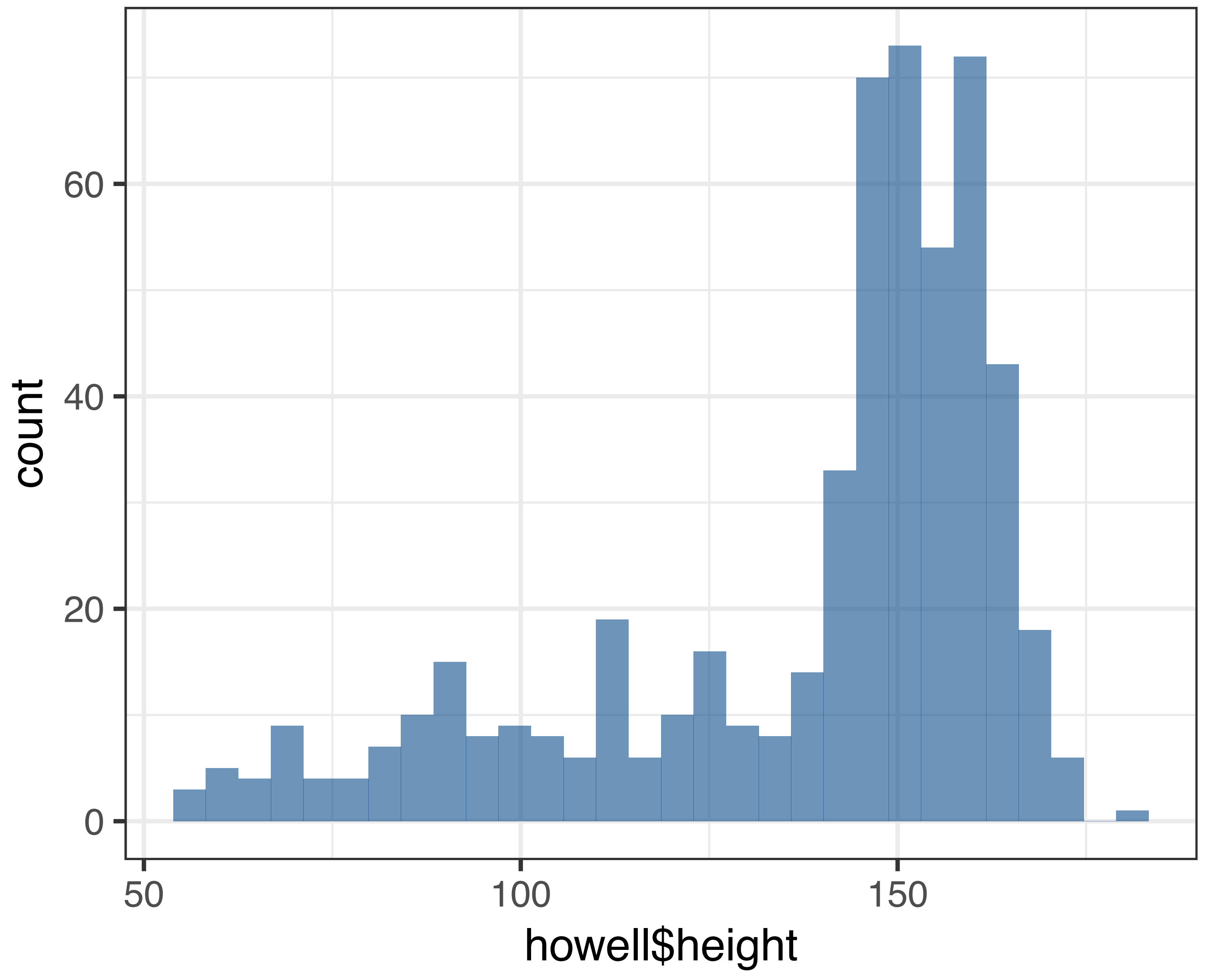


height	weight	age	sex
151.7650	47.82561	63.0	male
139.7000	36.48581	63.0	female
136.5250	31.86484	65.0	female
156.8450	53.04191	41.0	male
145.4150	41.27687	51.0	female
163.8300	62.99259	35.0	male
149.2250	38.24348	32.0	female
168.9100	55.47997	27.0	male
147.9550	34.86988	19.0	female
165.1000	54.48774	54.0	male

Plot the Data

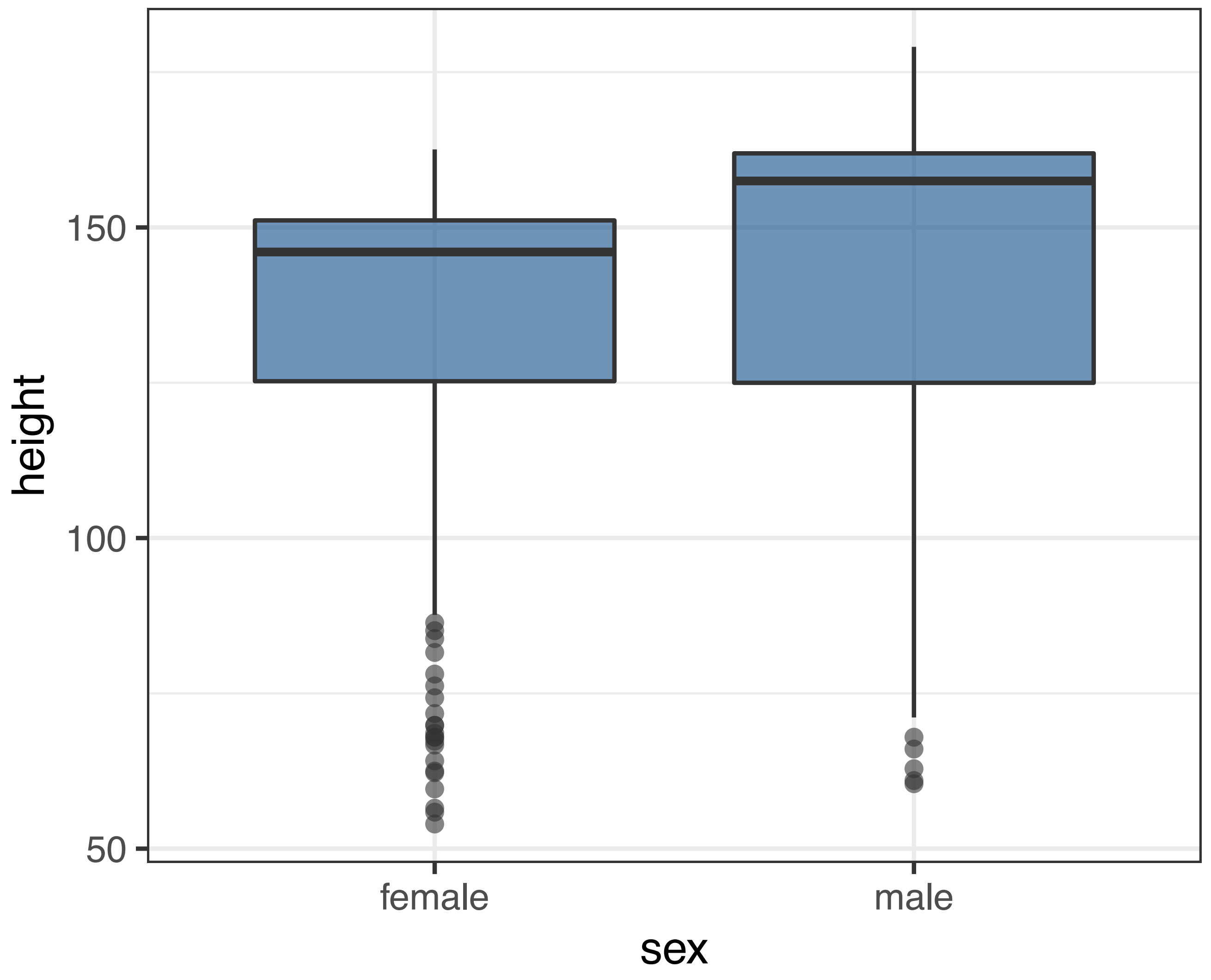
histogram

```
ggplot(howell) +
  theme_bw() +
  geom_histogram(aes(x = howell$height), fill = "dodgerblue4",
    alpha = 0.6)
```



Plot the Data
box plot

```
ggplot(howell) +
  theme_bw() +
  geom_boxplot(aes(x = sex, y = height), fill = "dodgerblue4",
    alpha = 0.6)
```

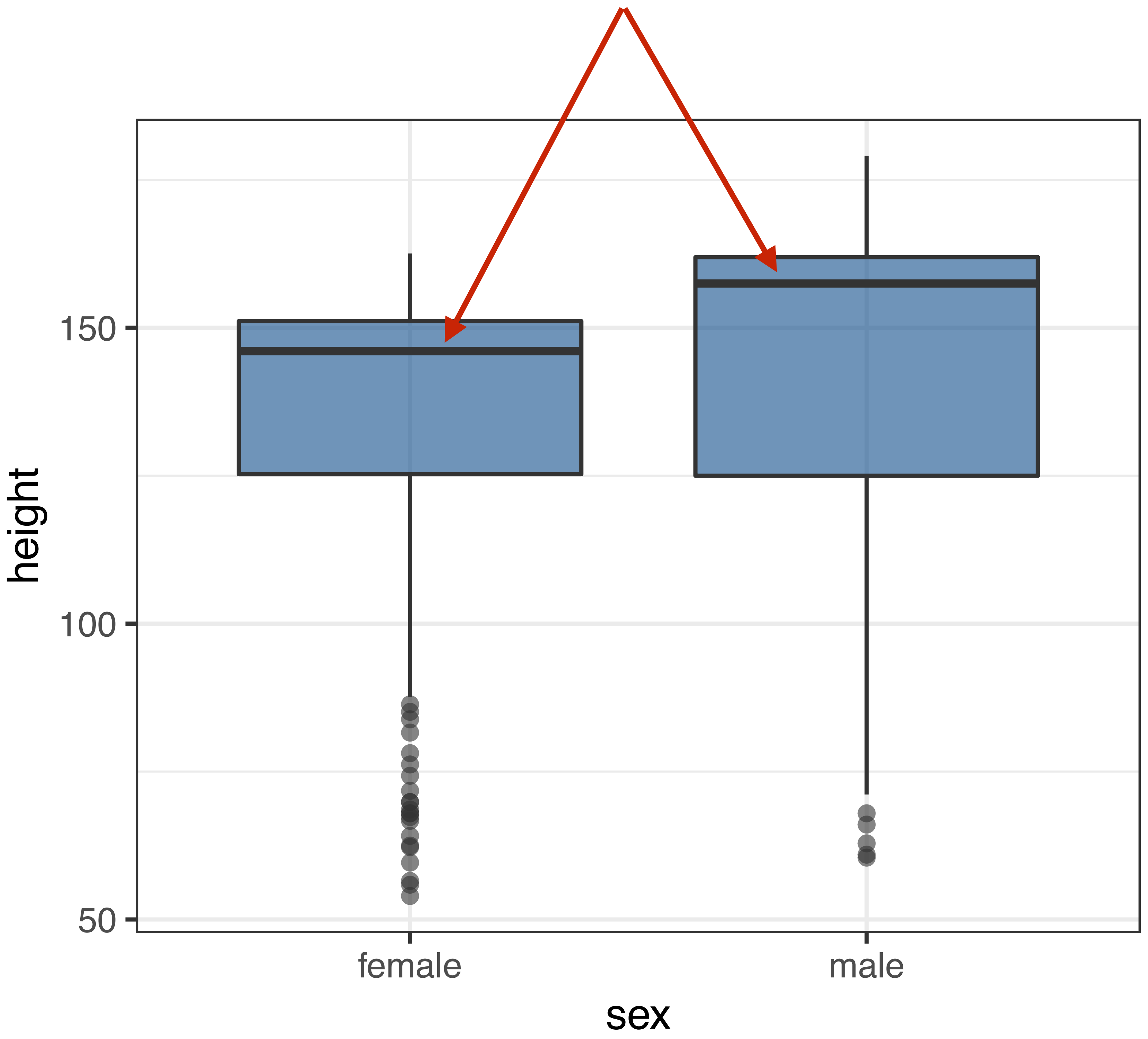


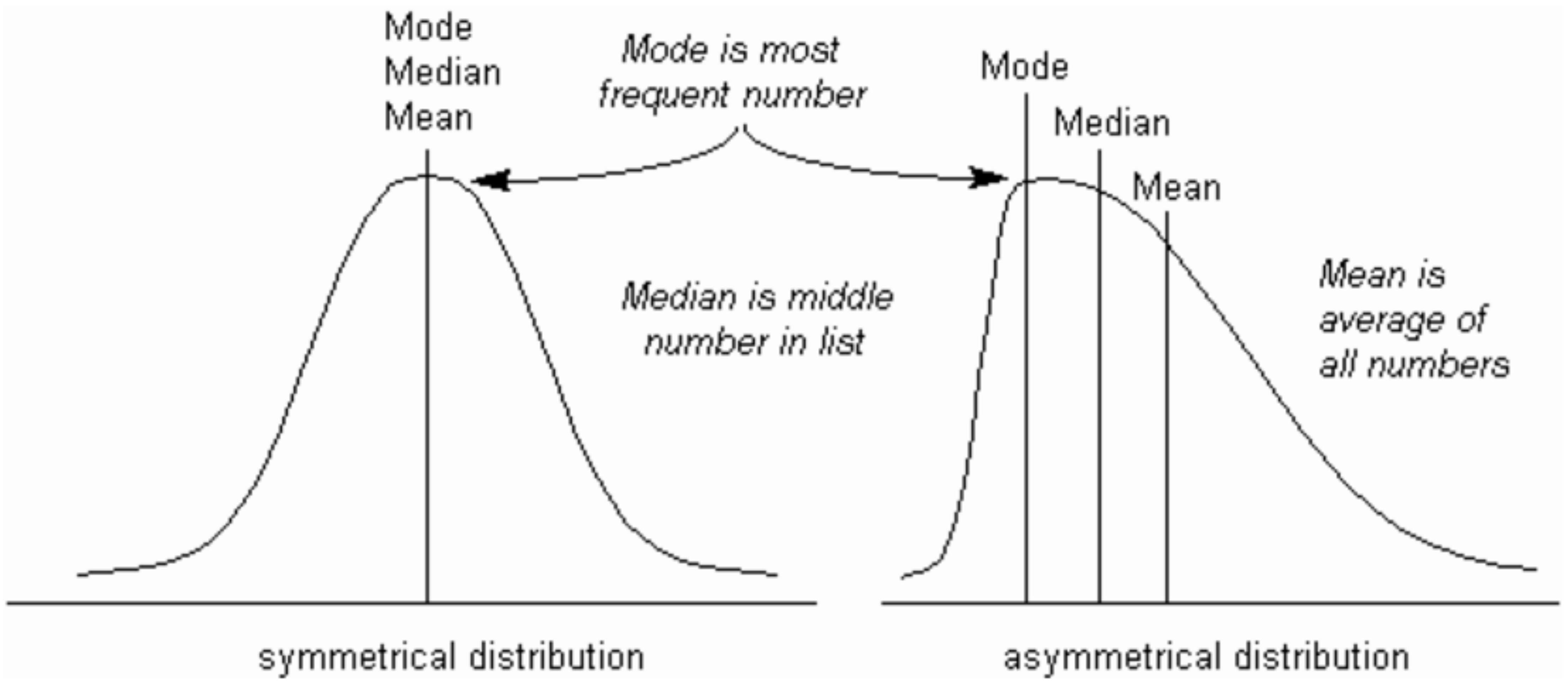
Plot the Data

box plot

median

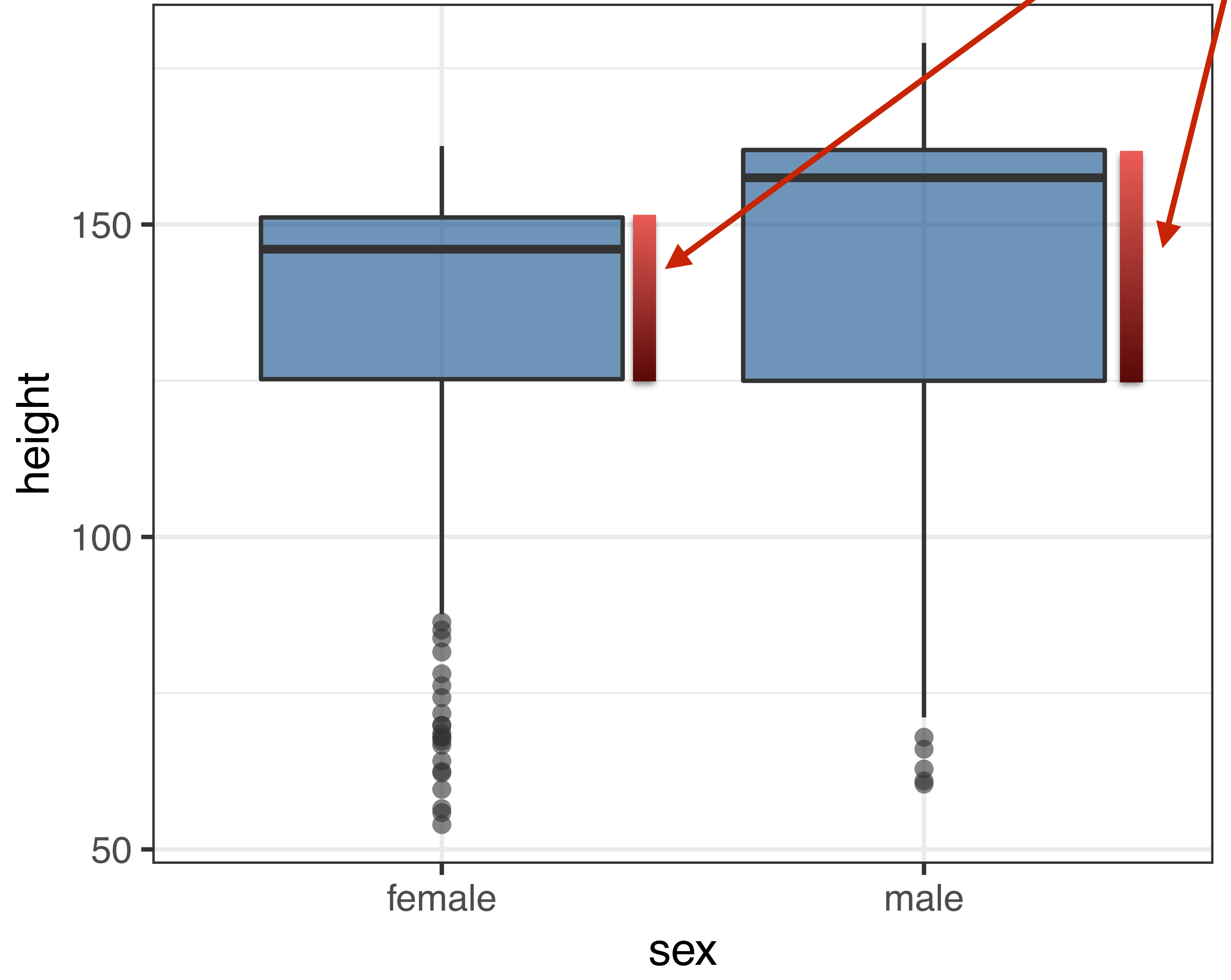
a number is equally likely to fall above or below it




Mode
Median
Mean
Mode is most frequent number
Mode
Median
Mean
Median is middle number in list
Mean is average of all numbers
symmetrical distribution
asymmetrical distribution

Plot the Data
box plot

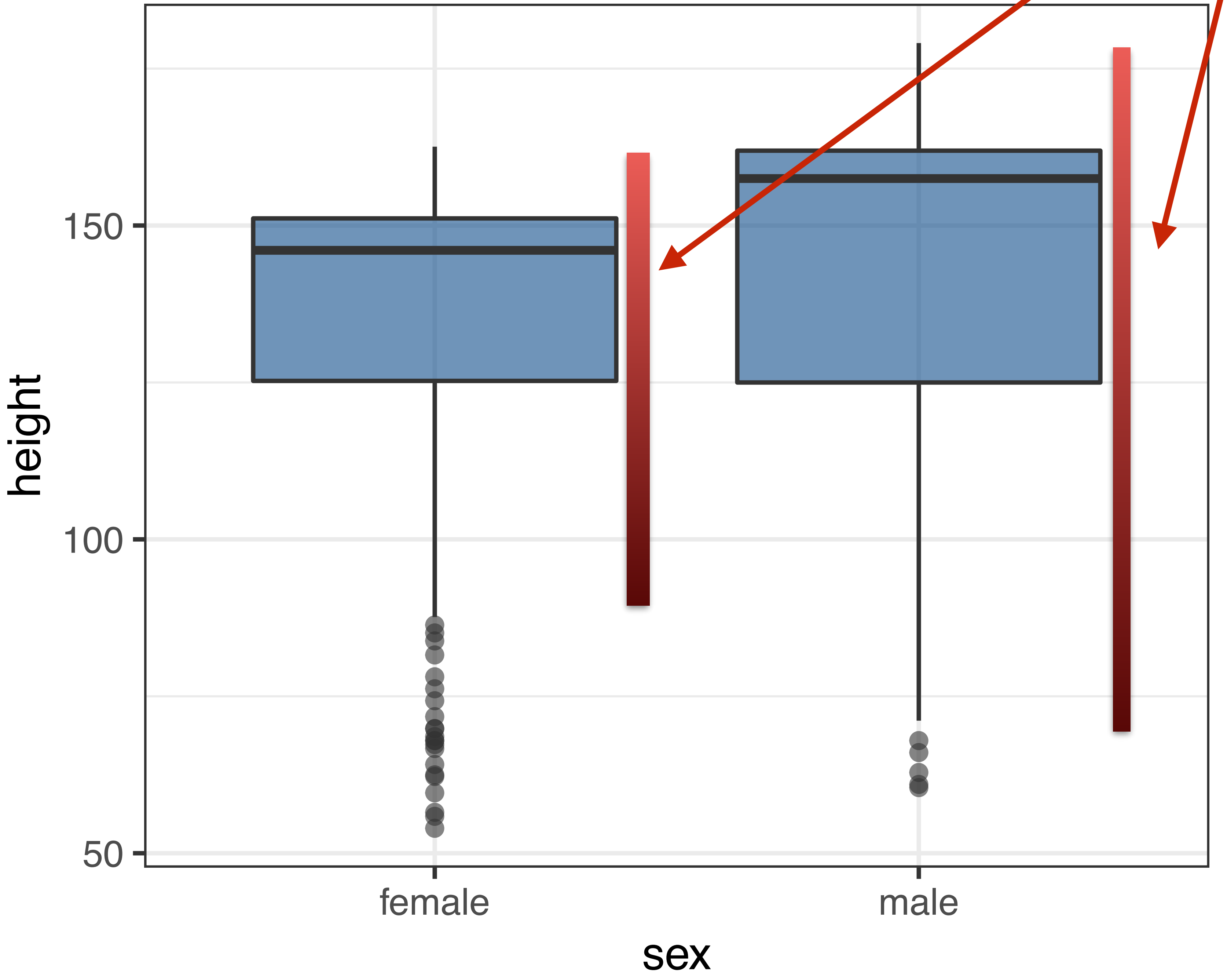
25th and 75th percentiles (1st and 3rd quartiles)



Plot the Data

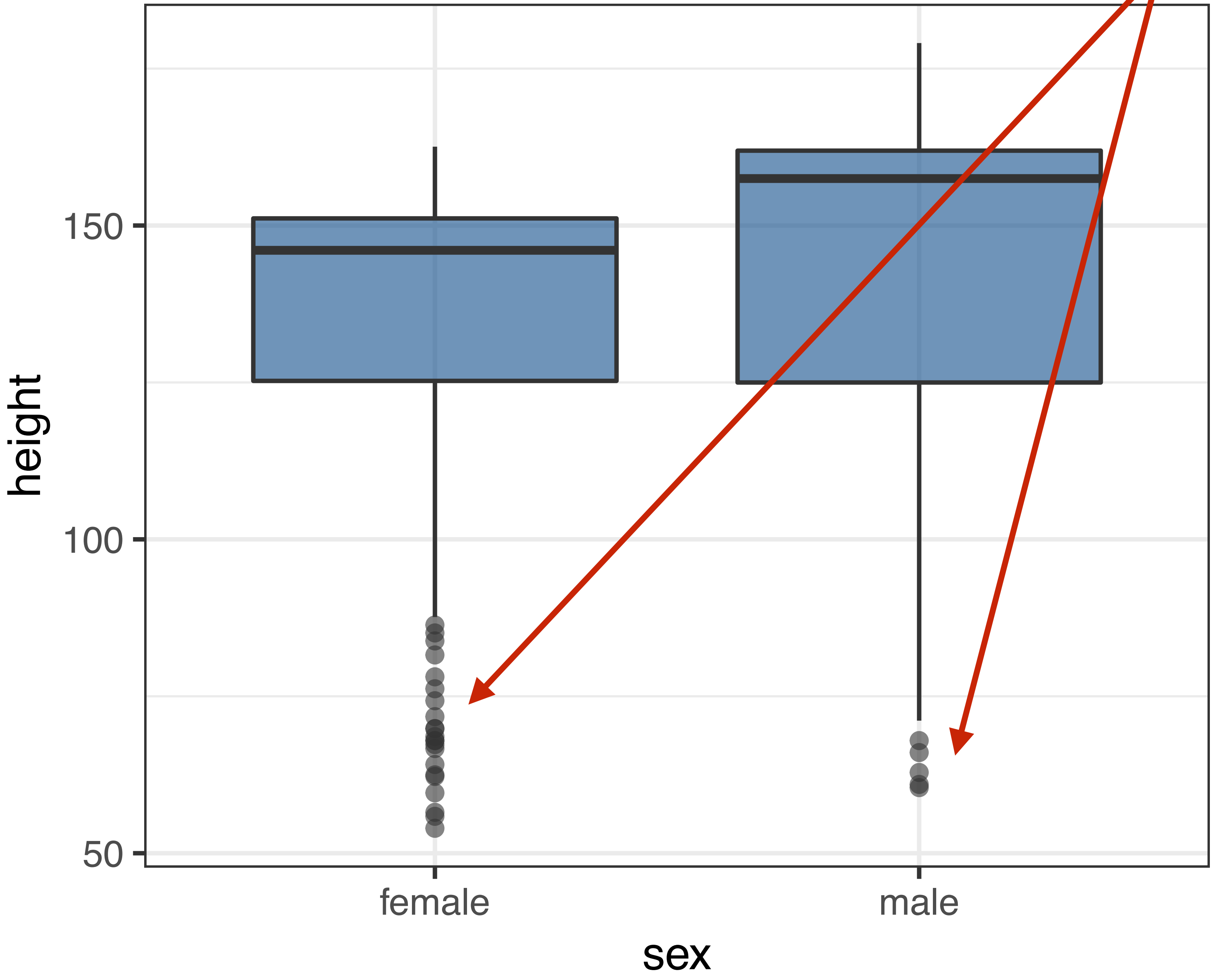
box plot

1.5X inter-quartile range (distance between 1st and 3rd quartile)



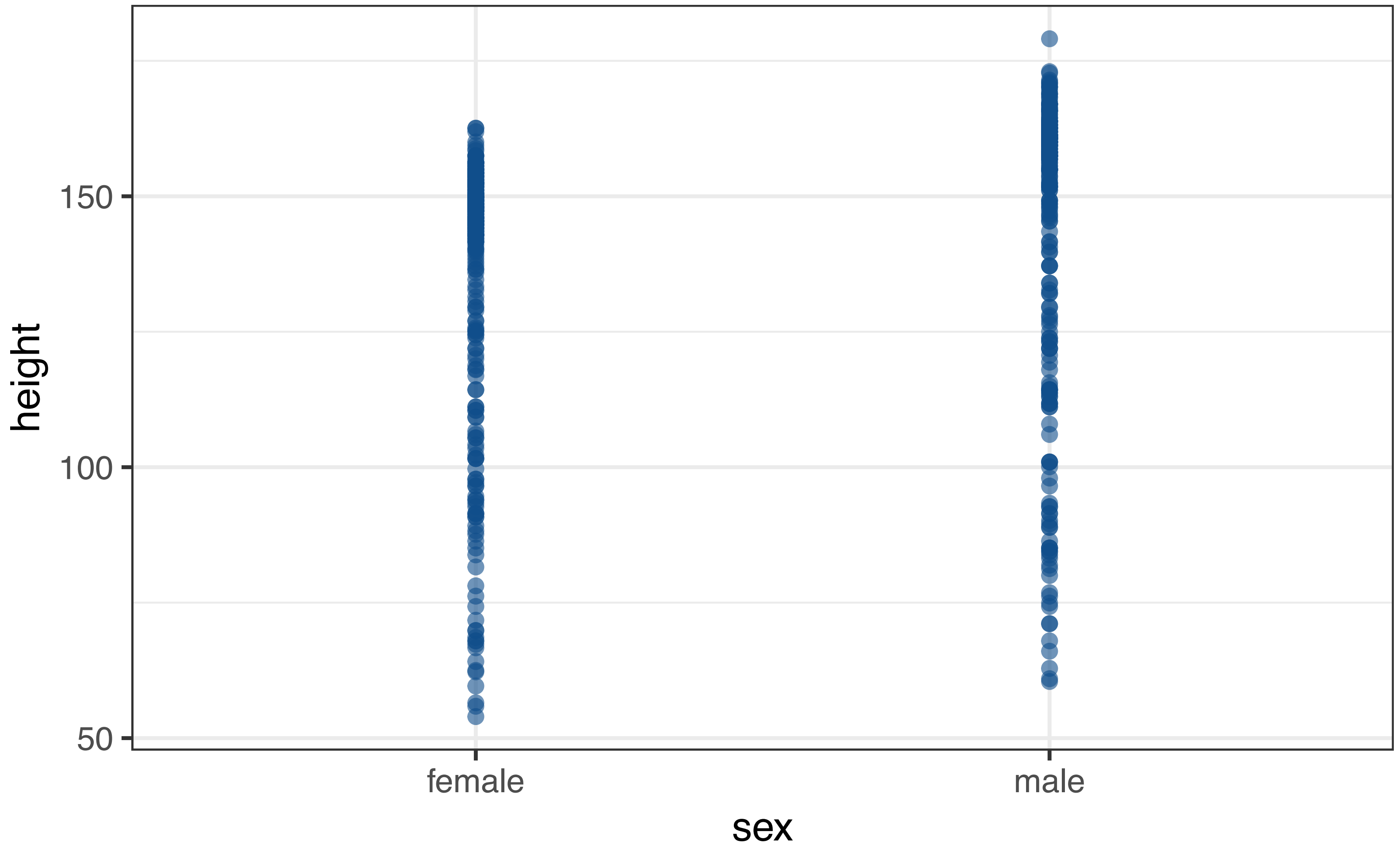
Plot the Data
box plot

“outliers”
All points that fall outside the 1.5X inter-quartile range



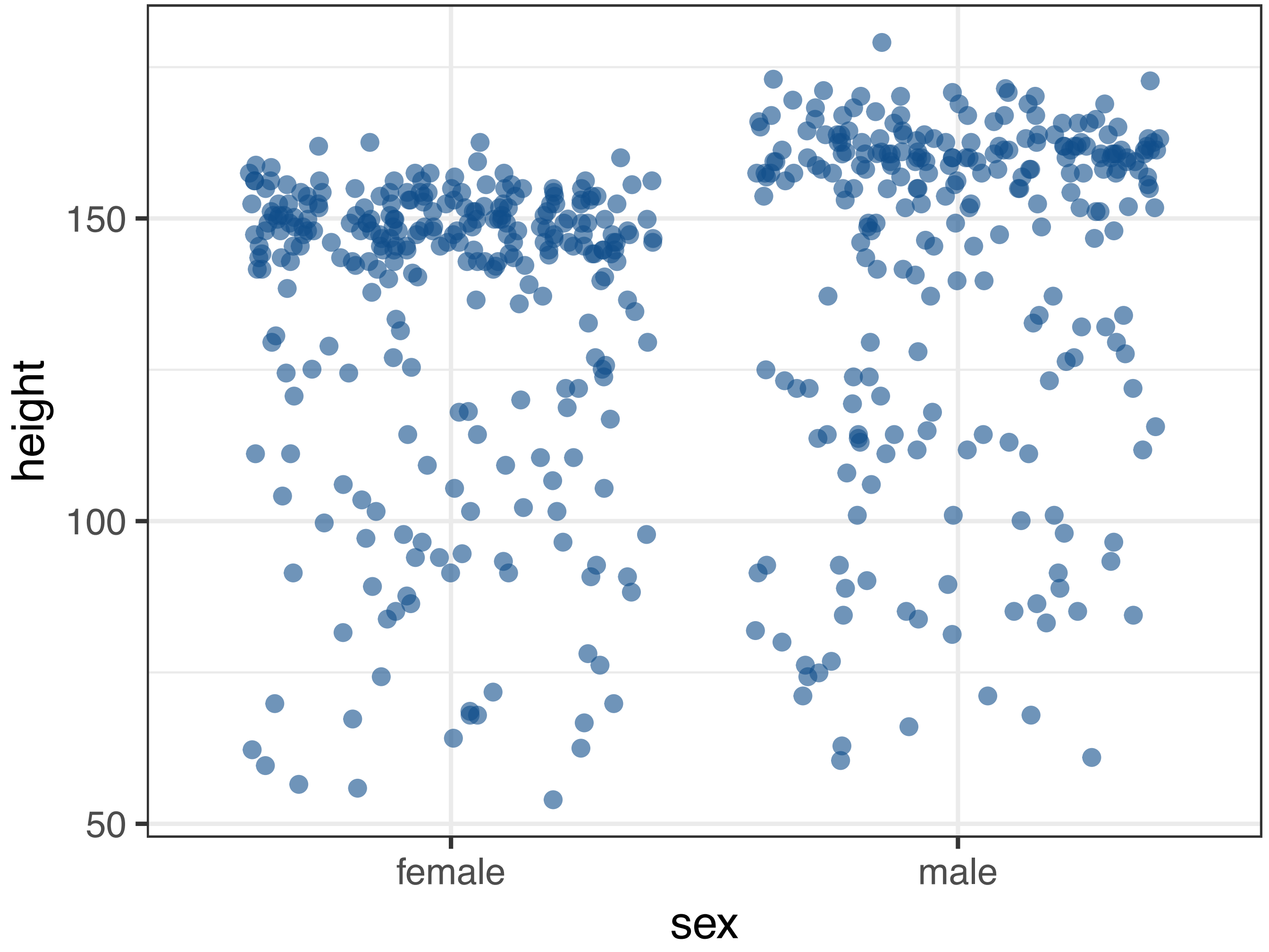
Plot the Data
point plot

```
ggplot(howell) +
  theme_bw() +
  geom_point(aes(x = sex, y = height), colour = "dodgerblue4",
    alpha = 0.6)
```



Plot the Data
jittered point plot

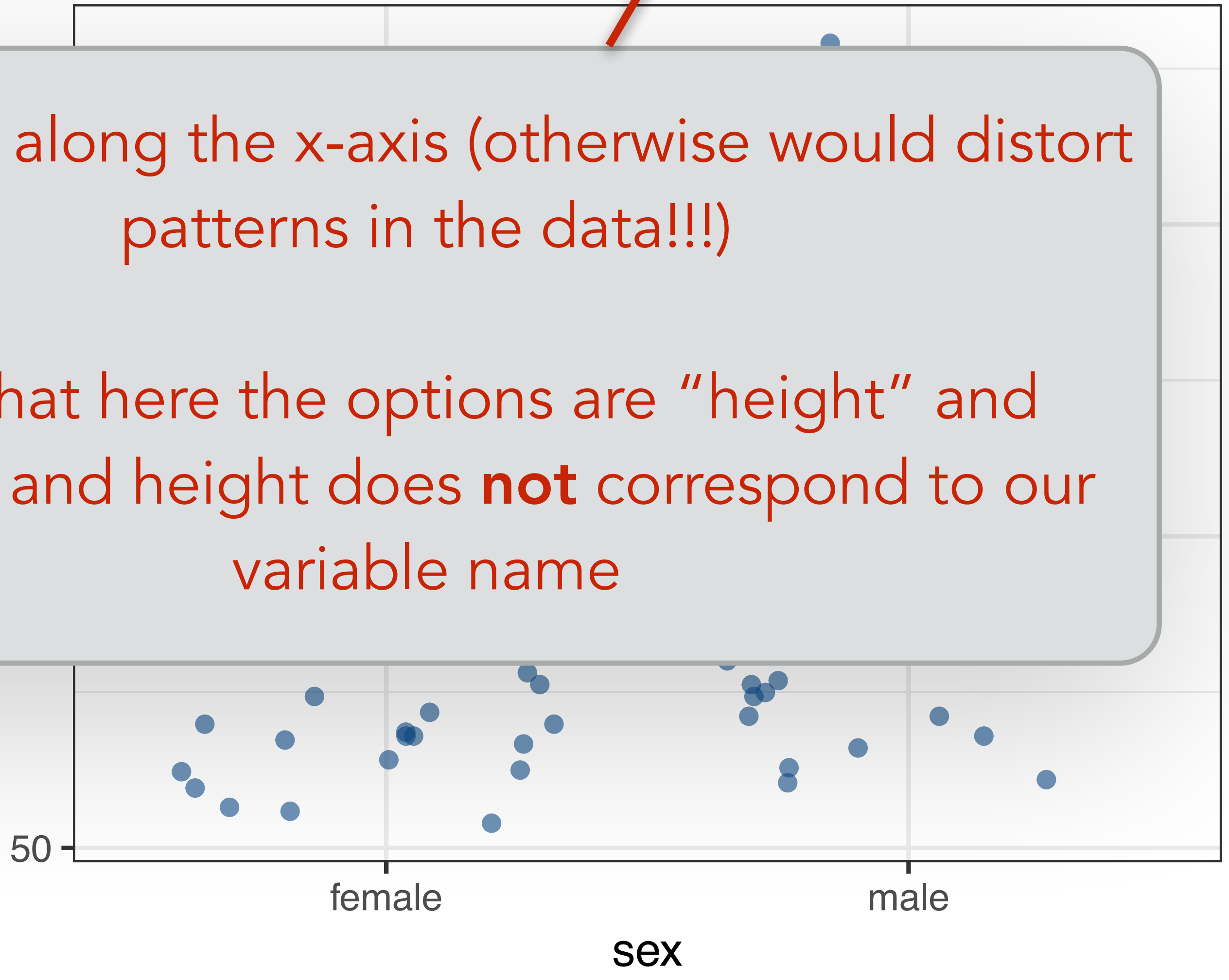
```
ggplot(howell) +
  theme_bw() +
  geom_jitter(aes(x = sex, y = height), height = 0, colour = "dodgerblue4",
   alpha = 0.6)
```



Plot the Data

jittered point plot

```
ggplot(howell) +
  theme_bw() +
  geom_jitter(aes(x = sex, y = height), height = 0, colour = "dodgerblue4",
   alpha = 0.6)
```



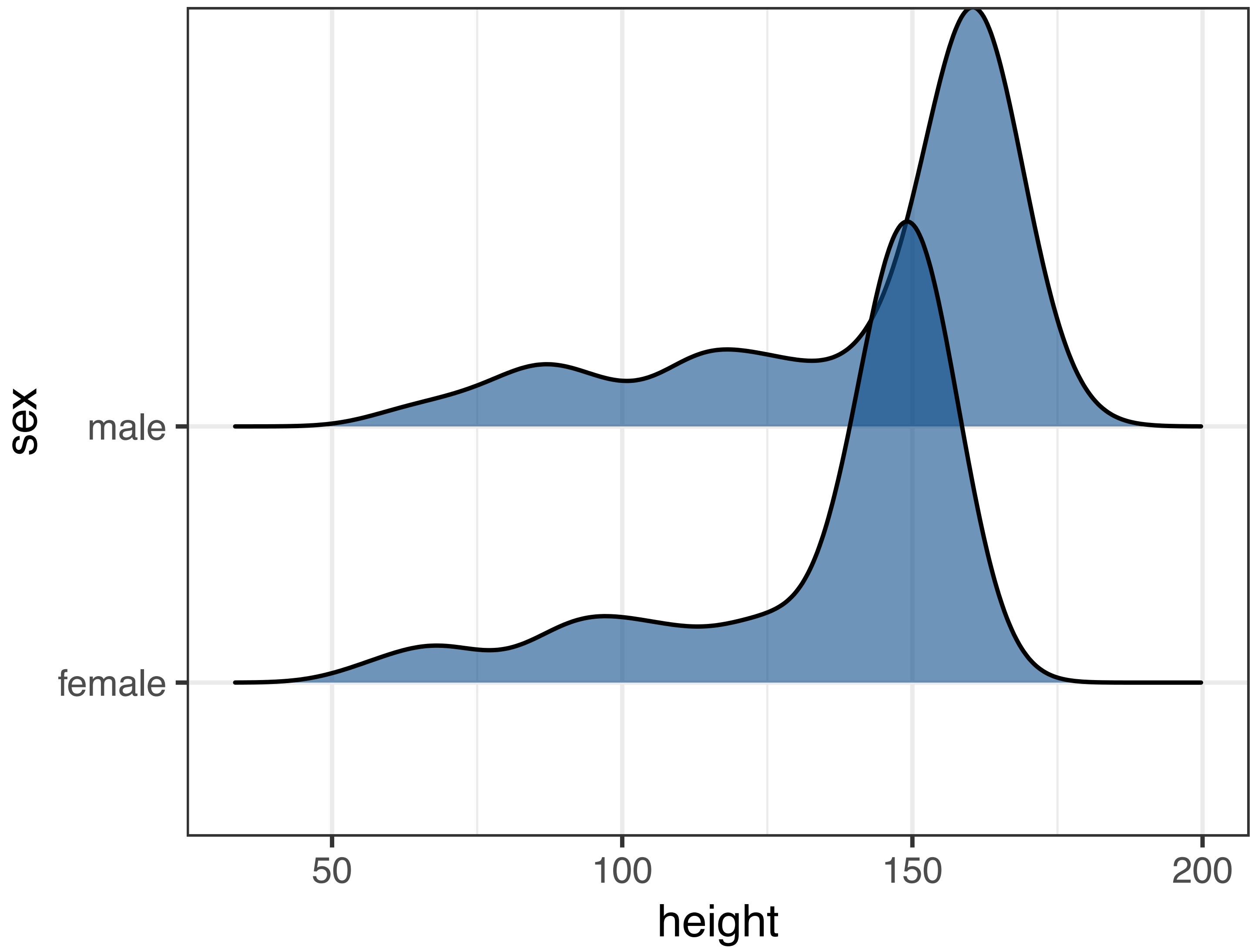
Only jitter along the x-axis (otherwise would distort patterns in the data!!!)

Note that here the options are "height" and "width", and height does **not** correspond to our variable name

Plot the Data

ridges (from the `ggridges` package)

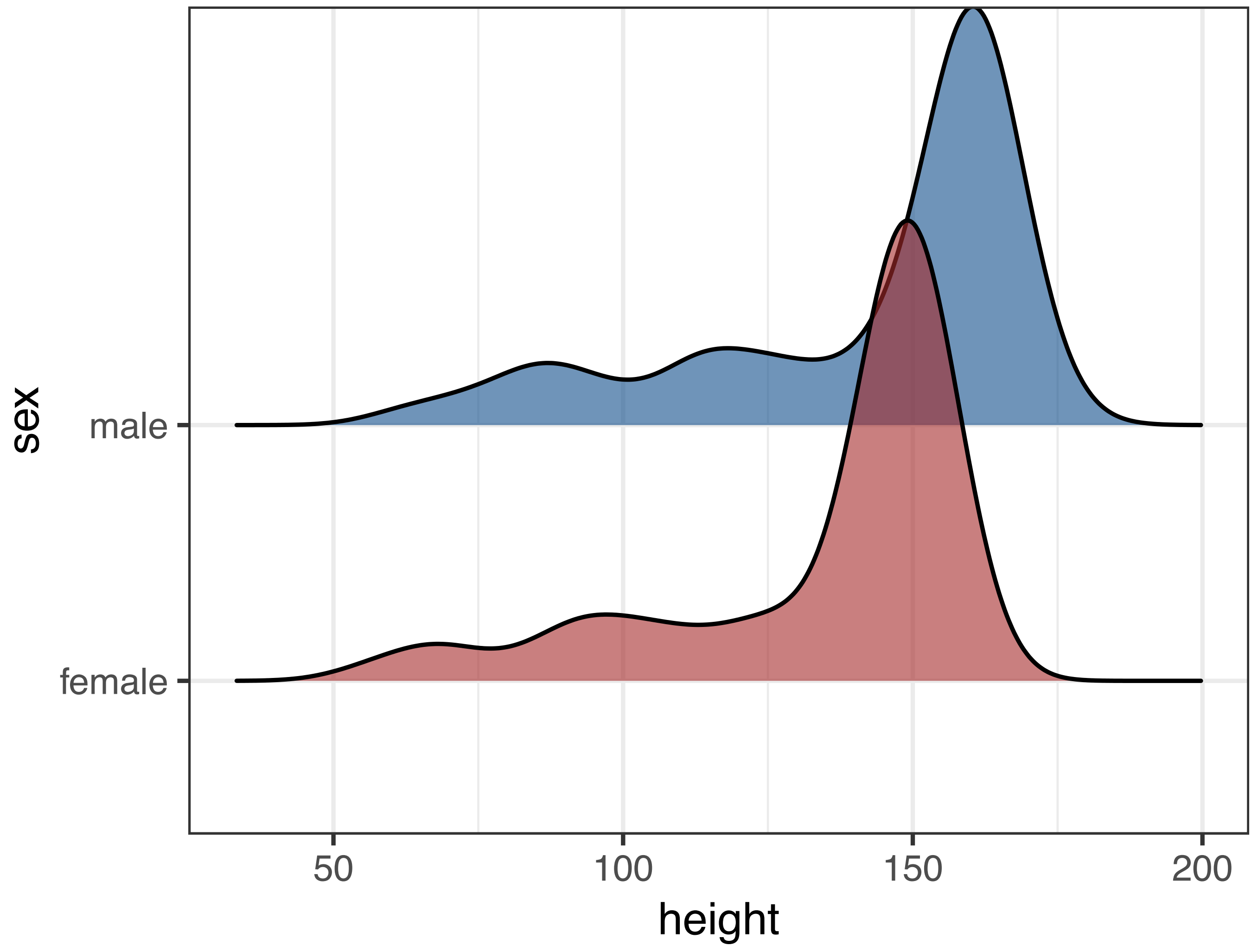
```
ggplot(howell) +
  theme_bw() +
  geom_density_ridges(aes(x = height, y = sex, group = sex),
    fill = "dodgerblue4", alpha = 0.6)
```



Plot the Data

ridges (from the `ggridges` package)

```
ggplot(howell) +
  theme_bw() +
  geom_density_ridges(aes(x = height, y = sex, group = sex, fill = sex),
   alpha = 0.6) +
  scale_fill_manual(values = c("brown", "dodgerblue4")) +
  theme(legend.position = "none")
```



Frequentist Approach

Frequentist Approach

- Often when comparing two groups, a *t*-test is used
- Data must be organized as two vectors: one for each sex

```
yFemale = howell[howell$sex == "female", 1]
yMale = howell[howell$sex == "male", 1]
```

Frequentist Approach

t-test

```
t.test(yFemale, yMale)

    Welch Two Sample t-test

data:  yFemale and yMale
t = -3.254, df = 517.65, p-value = 0.001212
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -12.334008  -3.047511
sample estimates:
mean of x mean of y 
 134.6303  142.3210 
```

Frequentist Approach
linear regression

```
model = lm(howell$height ~ howell$sex)


summary(model)

Call:
lm(formula = howell$height ~ howell$sex)

Residuals:
   Min      1Q  Median      3Q     Max 
-81.87 -12.73  12.65  18.33  36.75 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)     134.630      1.615  83.366  < 2e-16 ***
howell$sexmale    7.691      2.350   3.273  0.00113 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 27.36 on 542 degrees of freedom
Multiple R-squared:  0.01938, Adjusted R-squared:  0.01758 
F-statistic: 10.71 on 1 and 542 DF,  p-value: 0.001131
```

Frequentist Approach
linear regression

```
model = lm(howell$height ~ howell$sex)


summary(model)

Call:
lm(formula = howell$height ~ howell$sex)

Residuals:
   Min      1Q  Median      3Q     Max
-81.87 -12.73   12.65   18.33   36.75

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    134.630      1.615  83.366  < 2e-16 ***
howell$sexmale   7.691      2.350   3.273  0.00113 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 27.36 on 542 degrees of freedom
Multiple R-squared:  0.01938, Adjusted R-squared:  0.01758
F-statistic: 10.71 on 1 and 542 DF,  p-value: 0.001131
```

Mean height for females

Frequentist Approach

linear regression

```
model = lm(howell$height


summary(model)

Call:
lm(formula = howell$hei

Residuals:
   Min     1Q Median     3Q    Max
-81.87 -12.73  12.65  18.33  36.75

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     134.630      1.615  83.366  < 2e-16 ***
howell$sexmale    7.691      2.350   3.273  0.00113 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 27.36 on 542 degrees of freedom
Multiple R-squared:  0.01938, Adjusted R-squared:  0.01758
F-statistic: 10.71 on 1 and 542 DF,  p-value: 0.001131
```

Difference in mean height between males and females

`134.630 + 7.691 = 142.321`

Bayesian Approach

Analyses With Stan (or any MCMC process)

1. Prepare data for Stan
2. Build/define model
3. Run model
4. Assess MCMC process
5. Tentatively evaluate results
6. Conduct posterior predictive checks
7. Accept results or go back to step 2 to refine model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

“The standardized zy values come from j normal distributions, each with their own mean ($\text{mu}_{[j]}$), but with the same standard deviation (sigma).” In this case, the coefficient being estimated (β_1) represents the mean of each of the j levels of our nominal variable.

Data: How many "levels" there are in our nominal variable

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

Data: the standardized height values

Data: which "level" of our nominal variable each data point is in

Data: How many "levels" there are in our nominal variable

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$

$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

Data: the standardized height values

Parameter: the s.d. for each normal distribution for the *j* "levels" in our nominal variable. Assuming they all have the same here.

Parameter: the mean for the normal distribution for each of *j* "levels" in our nominal variable (we will have 2 in this case)

Data: which "level" of our nominal variable each data point is in

1. Prepare data for Stan

Prepare and Standardize the Metric Data

```
y = howell$height
N = length(y)

yMean = mean(y)
ySD = sd(y)
zy = (y - yMean) / ySD
```

Prepare the Nominal Data

```
x = as.numeric(howell$sex)
x

 [1] 2 1 1 2 1 2 1 2 1 2 1 2 1 1 1 2 2 1 2 1 1 2 1 2 1 2 1 1 2 1 2 2 2 1 1
     1 1 1 1 2 2 2 2 1
[45] 1 1 2 1 1 2 1 2 2 2 1 1 2 1 1 1 1 2 1 2 1 1 1 2 2 1 2 1 2 1 1 2 1 2 2
     1 1 1 1 2 2 1 2 2
...
```

Prepare the Nominal Data

```
xNames = levels(howell$sex)
xNames

[1] "female" "male"
```

Prepare the Nominal Data

```
nxLevels = length(xNames)
nxLevels

[1] 2
```

Prepare the Data As A List For Stan

```
dataList = list (
  y = zy,
  N = N,
  x = x,
  nxLevels = nxLevels
)
```

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

2. Build/Define the Model

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **data** block

```
data {
    int N;
    int nxLevels;
    vector[N] y;
    int x[N];    // Note that indices like this can't be vectors
  }
```

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **parameters** block

```
  parameters {
      real b1[nxLevels];      // A different b1 coefficient for each level in the variable
      real<lower=0> sigma;   // A single sigma value
    }
```

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **model** block

```
model {
    // Definitions
    vector[N] mu;

    // Likelihood
    for (i in 1:N) {
      mu[i] = b1[x[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    for (j in 1:nxLevels) {
      b1[j] ~ normal(0, 1);
    }

    sigma ~ cauchy(1, 1);
  }
```

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **model** block

```
model {
    // Definitions
    vector[N] mu;

    // Likelihood
    for (i in 1:N) {
      mu[i] = b1[x[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    for (j in 1:nxLevels) {
      b1[j] ~ normal(0, 1);
    }

    sigma ~ cauchy(1, 1);
  }
```

Define a vector for holding the mu values for each individual/sample

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **model** block

```
model {
    // Definitions
    vector[N] mu;

    // Likelihood
    for (i in 1:N) {
      mu[i] = b1[x[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    for (j in 1:nxLevels) {
      b1[j] ~ normal(0, 1);
    }

    sigma ~ cauchy(1, 1);
  }
```

Because we have to refer to specific locations, need to nest likelihood in a `for` loop.

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **model** block

```
model {
    // Definitions
    vector[N] mu;

    // Likelihood
    for (i in 1:N) {
      mu[i] = b1[x[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    for (j in 1:nxLevels) {
      b1[j] ~ normal(0, 1);
    }

    sigma ~ cauchy(1, 1);
  }
```

Likelihood is the same as above.

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **model** block

```
model {
    // Definitions
    vector[N] mu;

    // Likelihood
    for (i in 1:N) {
      mu[i] = b1[x[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    for (j in 1:nxLevels) {
      b1[j] ~ normal(0, 1);
    }

    sigma ~ cauchy(1, 1);
  }
```

How we tell Stan that the x values are indices rather than actual values.

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **model** block

```
model {
    // Definitions
    vector[N] mu;

    // Likelihood
    for (i in 1:N) {
      mu[i] = b1[x[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    for (j in 1:nxLevels) {
      b1[j] ~ normal(0, 1);
    }

    sigma ~ cauchy(1, 1);
  }
```

The mu for each individual/ sample will be based on what "level" it is in

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **model** block

```
model {
    // Definitions
    vector[N] mu;

    // Likelihood
    for (i in 1:N) {
      mu[i] = b1[x[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    for (j in 1:nxLevels) {
      b1[j] ~ normal(0, 1);
    }

    sigma ~ cauchy(1, 1);
  }
```

Need a prior for each b1 parameter.

2. Build/Define The Model

$$zy \sim \text{normal}(\text{mu}_{[j]}, \text{sigma})$$
$$\text{mu}_{[j]} = \beta_1[x_{[j]}]$$

- The **model** block

```
model {
    // Definitions
    vector[N] mu;

    // Likelihood
    for (i in 1:N) {
      mu[i] = b1[x[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    for (j in 1:nxLevels) {
      b1[j] ~ normal(0, 1);
    }

    sigma ~ cauchy(1, 1);
  }
```

Prior for sigma the same as before.

2. Build/Define The Model

- The **generated quantities** block

```
generated quantities {
    vector[N] y_pred;

    for (i in 1:N) {
      y_pred[i] = normal_rng(b1[x[i]], sigma);
    }
  }
```

```
modelString = "
  data {
    int N;
    int nxLevels;
    vector[N] y;
    int x[N];    // Note that indices like this can't be vectors
  }

  parameters {
    real b1[nxLevels];    // A different b1 coefficient for each level in the variable
    real<lower=0> sigma;  // A single sigma value
  }

  model {
    // Definitions
    vector[N] mu;

    // Likelihood
    for (i in 1:N) {
      mu[i] = b1[x[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    for (j in 1:nxLevels) {
      b1[j] ~ normal(0, 1);
    }

    sigma ~ cauchy(1, 1);
  }

  generated quantities {
    vector[N] y_pred;

    for (i in 1:N) {
      y_pred[i] = normal_rng(b1[x[i]], sigma);
    }
  }
"
writeLines(modelString, con="model.stan")
```

3. Run the Model

3. Run The Model

```
stanFit <- stan(file = "model.stan",
                data = dataList,
                pars = c("b1", "sigma", "y_pred"),
                warmup = 2000,
                iter = 7000,
                chains = 3)
```

4. Assess Performance of MCMC Process

4. Assess MCMC Process

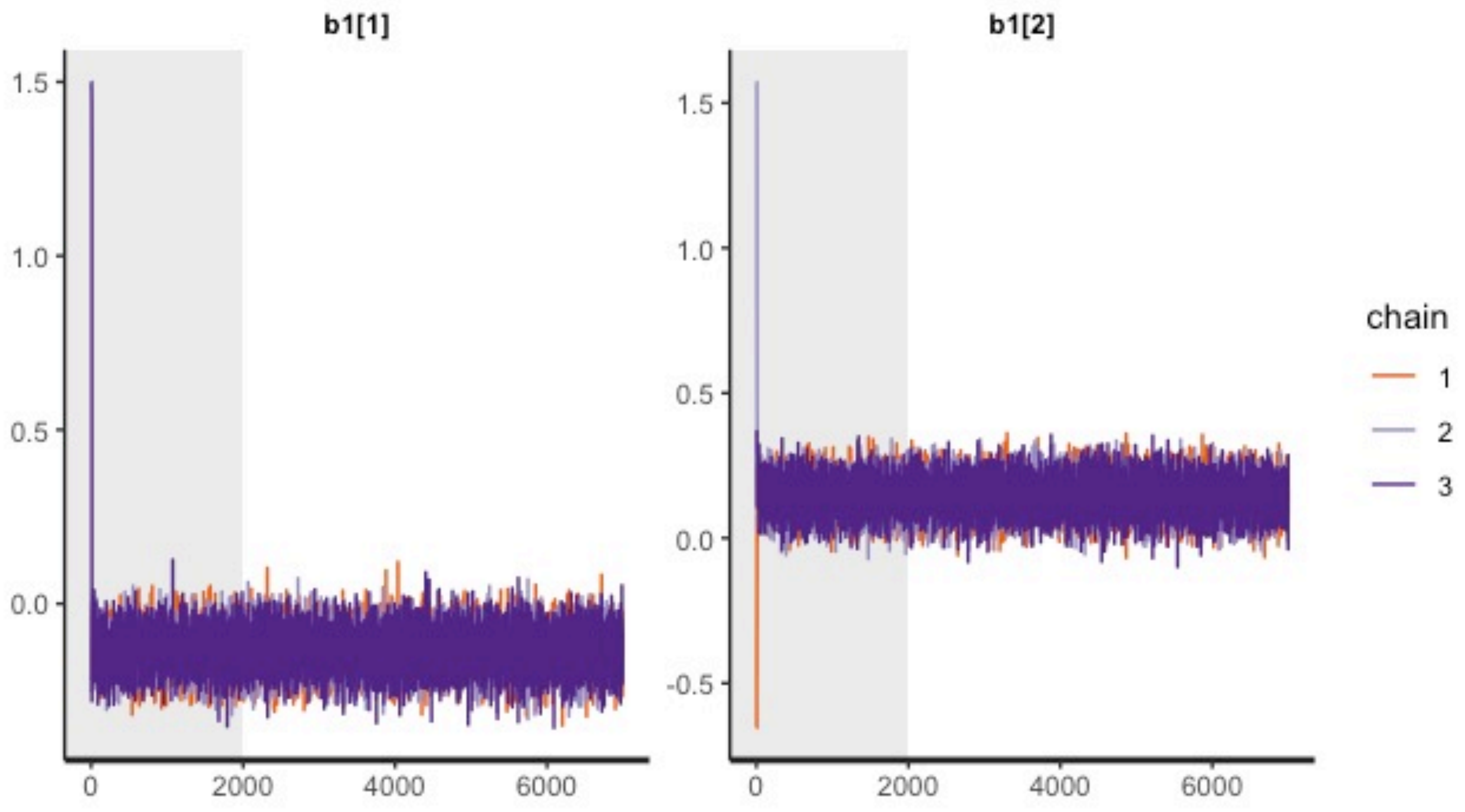
- Check Rhat and effective sample size stats

```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=7000; warmup=2000; thin=1;
post-warmup draws per chain=5000, total post-warmup draws=15000.

               mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
b1[1]         -0.13    0.00 0.06   -0.25   -0.17   -0.13   -0.09   -0.02 14209    1
b1[2]          0.15    0.00 0.06    0.03    0.10    0.15    0.19    0.27 13291    1
sigma          0.99    0.00 0.03    0.94    0.97    0.99    1.01    1.05 13966    1
y_pred[1]      0.14    0.01 1.00   -1.82   -0.54    0.14    0.80    2.09 14833    1
...
```

4. Assess MCMC Process

- Check trace plots

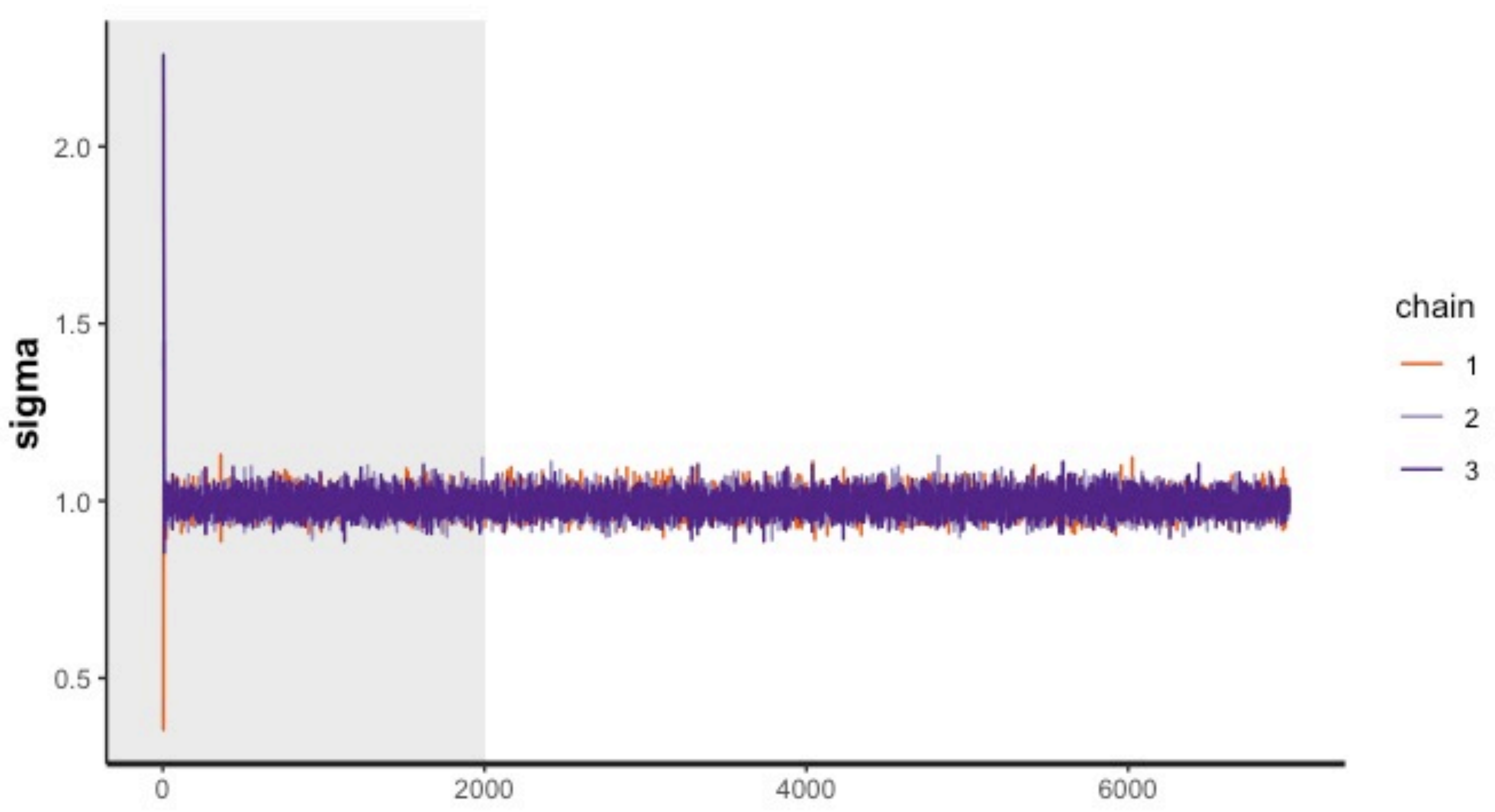
```
stan_trace(stanFit, pars = "b1", inc_warmup = TRUE)
```



4. Assess MCMC Process

- Check trace plots

```
stan_trace(stanFit, pars = "sigma", inc_warmup = TRUE)
```



5. Tentatively Interpret Results

5. Tentatively Interpret Results

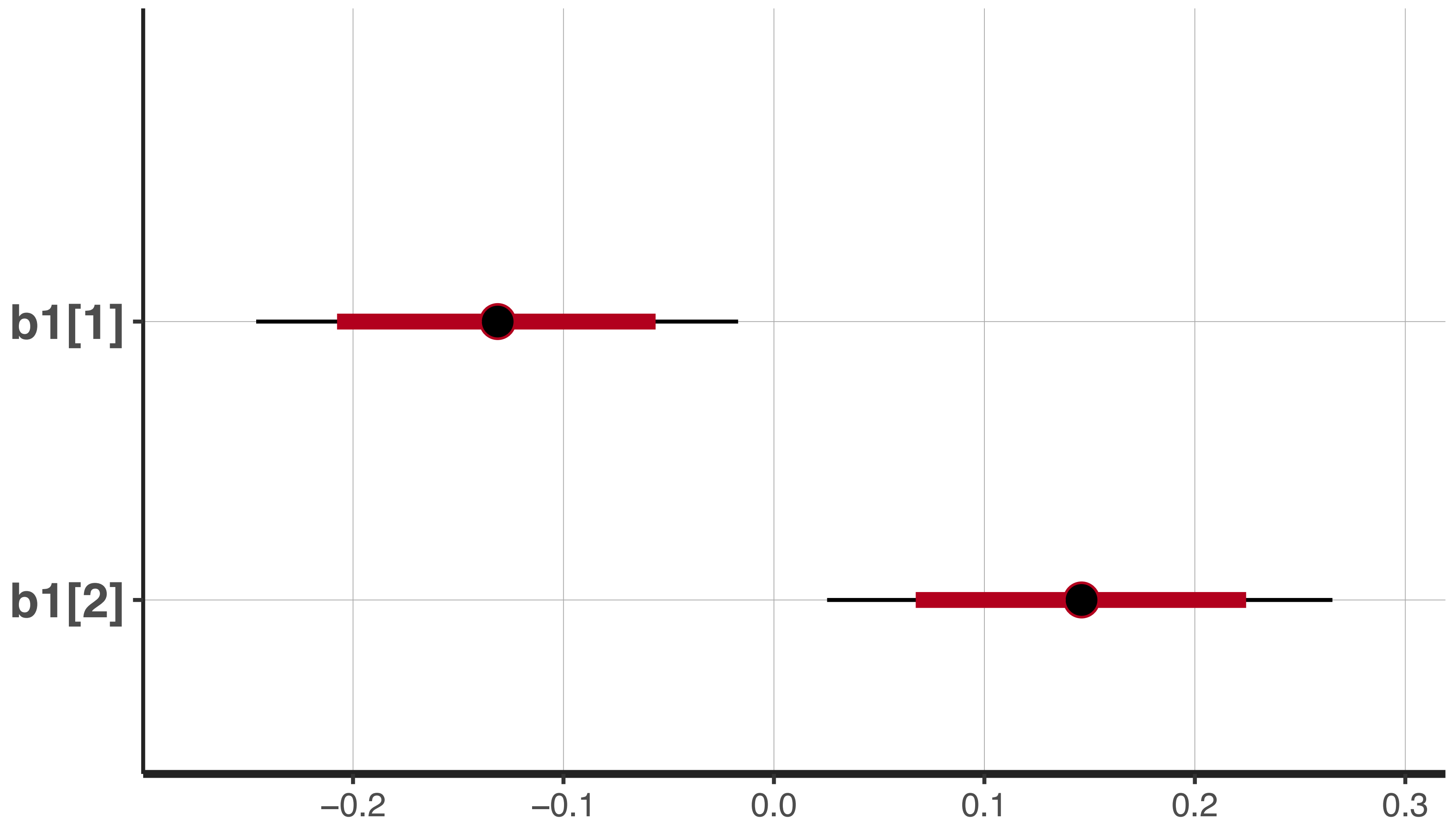
View stats

```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=7000; warmup=2000; thin=1;
post-warmup draws per chain=5000, total post-warmup draws=15000.

              mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
b1[1]        -0.13    0.00 0.06  -0.25  -0.17  -0.13  -0.09  -0.02 14209    1
b1[2]         0.15    0.00 0.06   0.03   0.10   0.15   0.19   0.27 13291    1
sigma         0.99    0.00 0.03   0.94   0.97   0.99   1.01   1.05 13966    1
y_pred[1]     0.14    0.01 1.00  -1.82  -0.54   0.14   0.80   2.09 14833    1
...
```

5. Tentatively Interpret Results

Plot with `rstan` functions

```
stan_plot(stanFit, par = "b1")
```

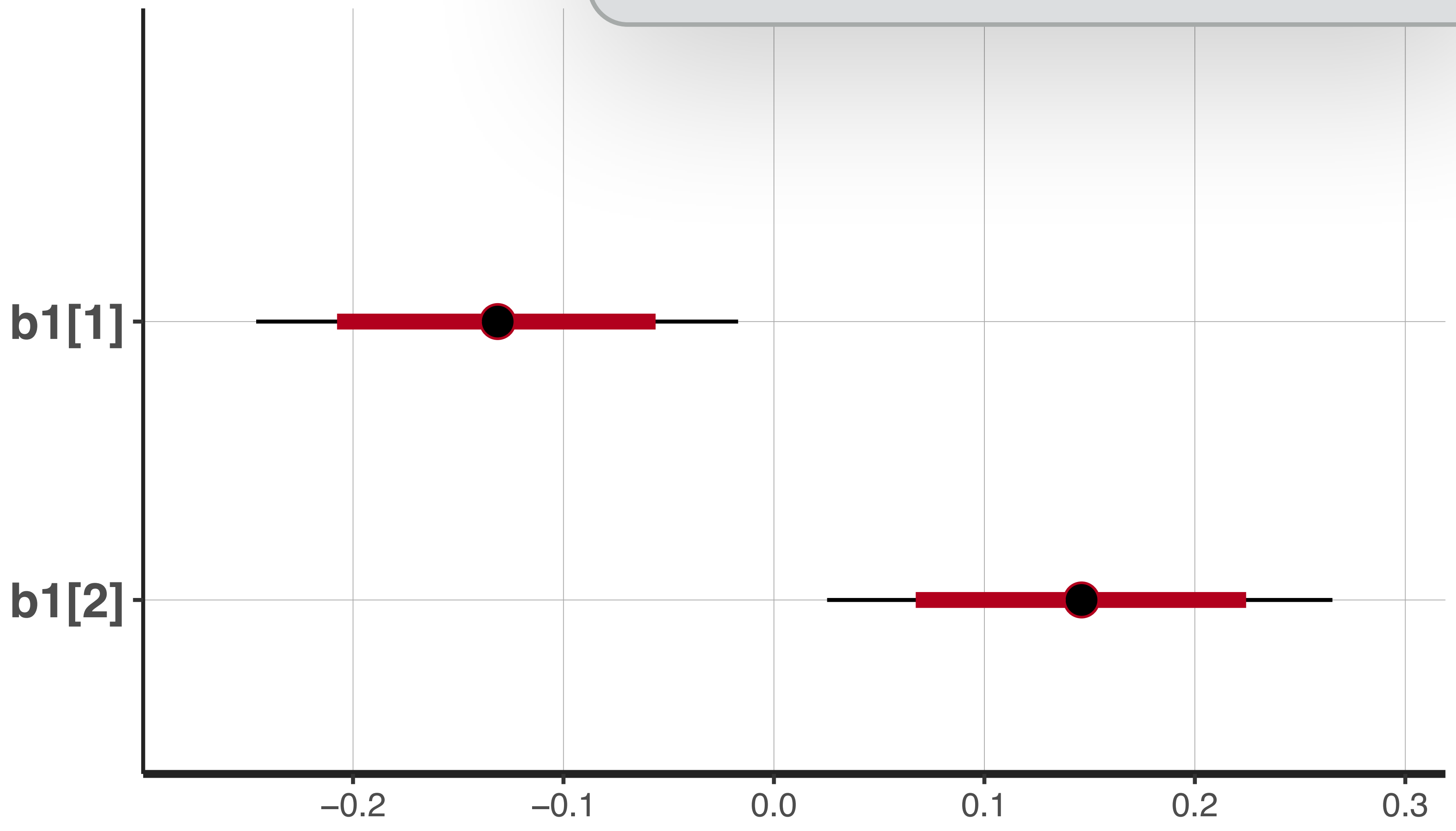


5. Tentatively Interpret Results

Plot with `rstan` functions

```
stan_plot(stanFit, par = "b1")
```

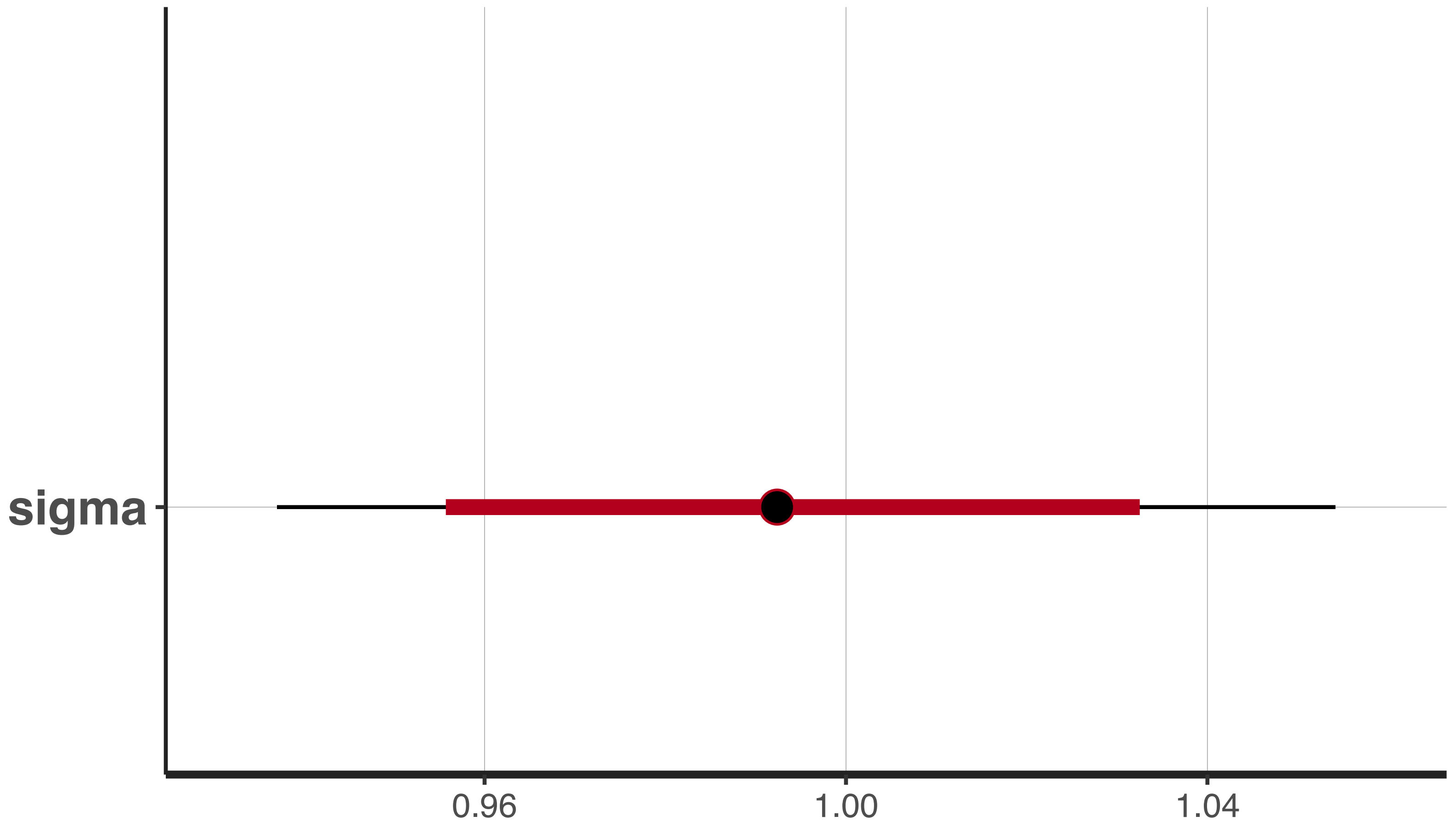
Interpretation: Females are shorter than males, with no overlap in 95% HDI of estimated mean values.



5. Tentatively Interpret Results

Plot with `rstan` functions

```
stan_plot(stanFit, par = “sigma")
```



5. Tentatively Interpret Results

Make custom plots (base R)

- Extract the data and parse out components

```
mcmcChains = as.data.frame(stanFit)
zsigma = mcmcChains$sigma
```

5. Tentatively Interpret Results

Make custom plots (base R)

- Extract the data and parse out components

```
chainLength = length(zsigma)

zb1 = matrix(0, ncol = nxLevels, nrow = chainLength)

for (i in 1:nxLevels) {
  zb1[, i] = mcmcChains[, paste("b1[", i, "]", sep = "")]
}
```

5. Tentatively Interpret Results

Make custom plots (base R)

- Extract the data and parse out components

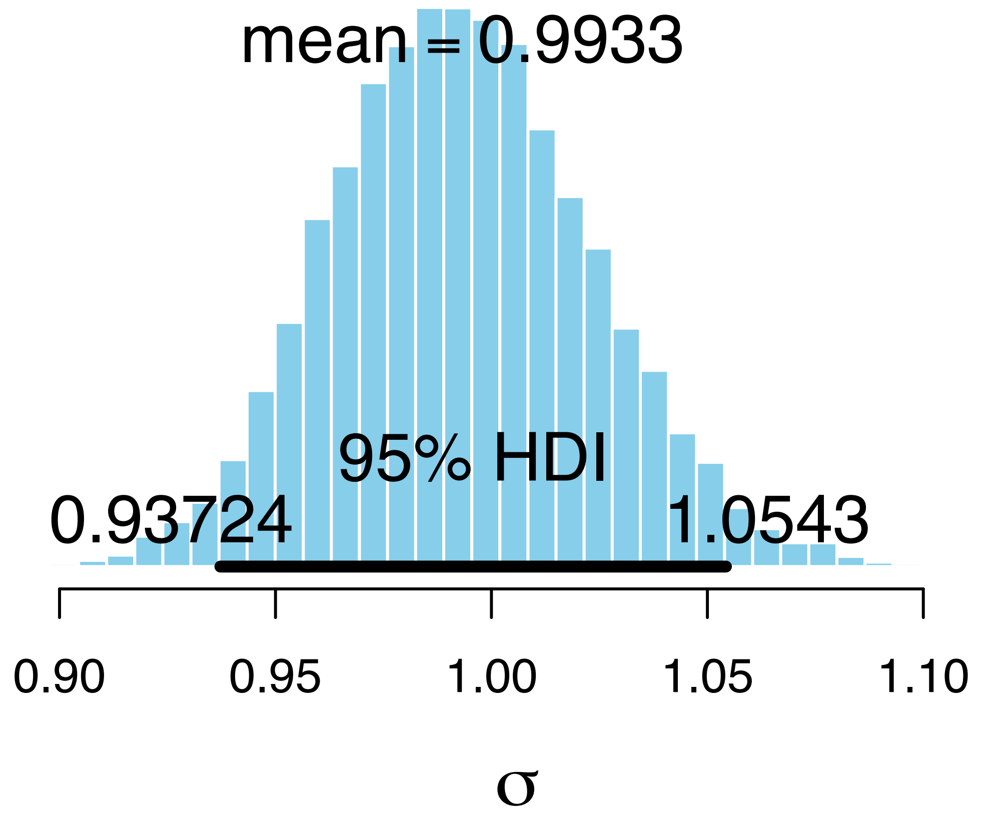
```
ypred = matrix(0, ncol = N, nrow = chainLength)

for (i in 1:N) {
  ypred[, i] = mcmcChains[, paste("y_pred[", i, "]", sep = "")]
}
```

5. Tentatively Interpret Results

Make custom plots (base R)

- Plot using Kruschke's functions

```
histInfo = plotPost(zsigma, xlab = bquote(sigma))
```

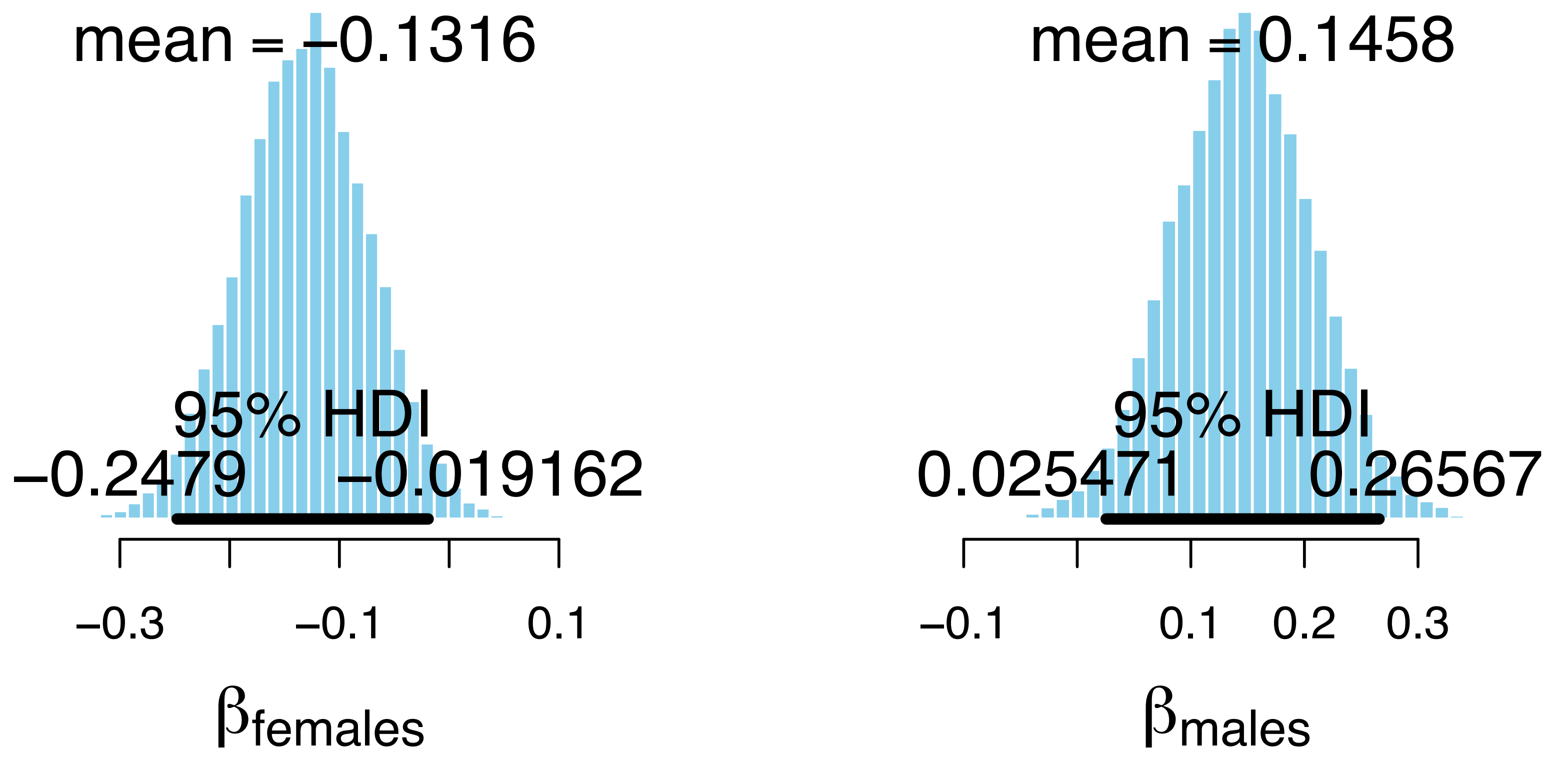


5. Tentatively Interpret Results

Make custom plots (base R)

- Plot using Kruschke's functions

```
par(mfrow = c(1, 2))
histInfo = plotPost(zb1[, 1], xlab = bquote(beta[females]))
histInfo = plotPost(zb1[, 2], xlab = bquote(beta[males]))
```



5. Tentatively Interpret Results

Plot with ggridges

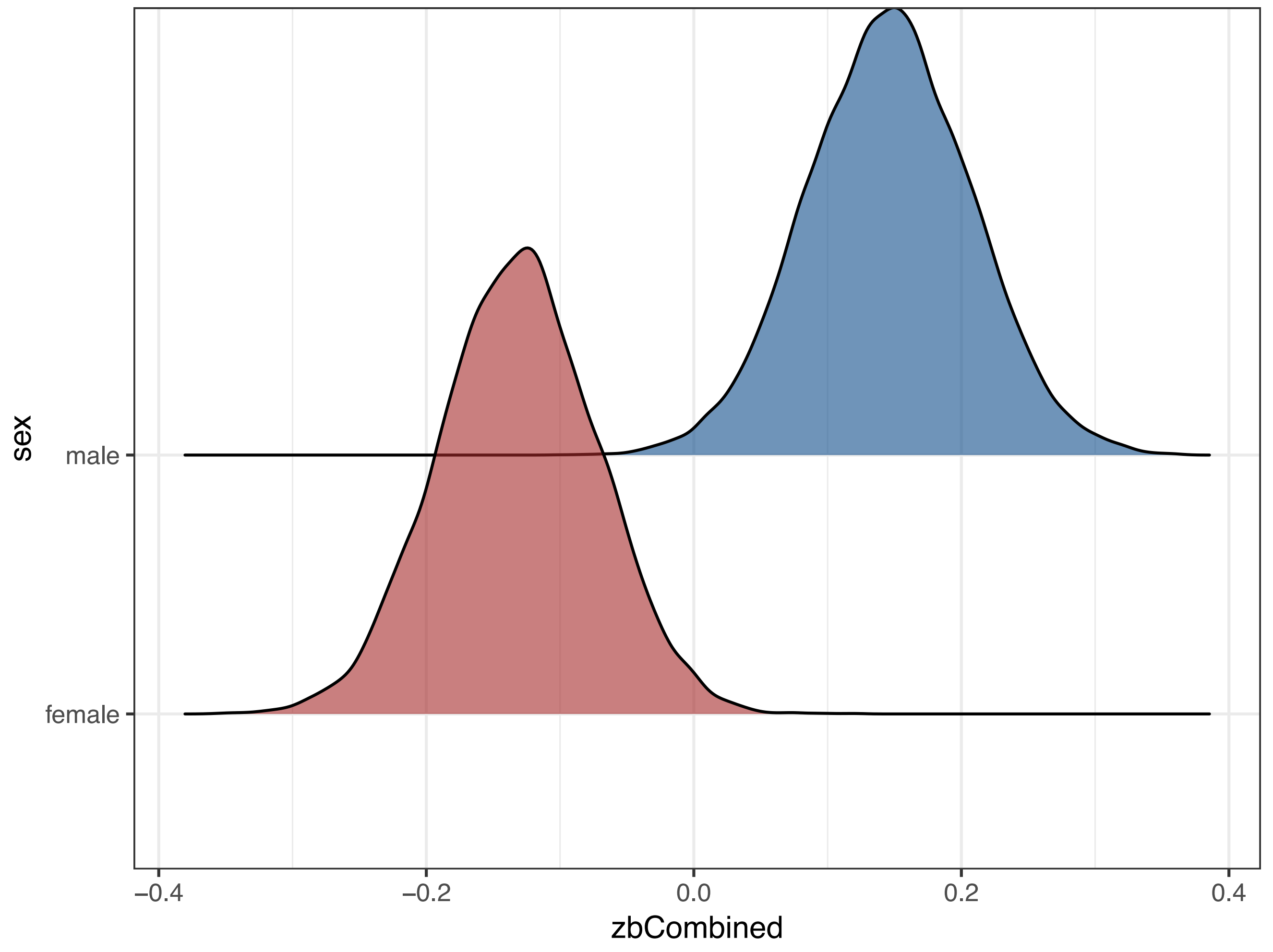
- Organize the data

```
zbCombined = c(zb1[, 1], zb1[, 2])

sex = c(rep("female", times = length(zb1[, 1])), rep("male",
  times = length(zb1[, 1])))

combined = data.frame(zbCombined, sex)
```

5. Tentatively Interpret Results

Plot with ggridges

```
ggplot(combined) +
  theme_bw() +
  geom_density_ridges(aes(x = zbCombined, y = sex, group = sex,
    fill = sex), alpha = 0.6) +
  scale_fill_manual(values = c("brown", "dodgerblue4")) +
  theme(legend.position = "none")
```

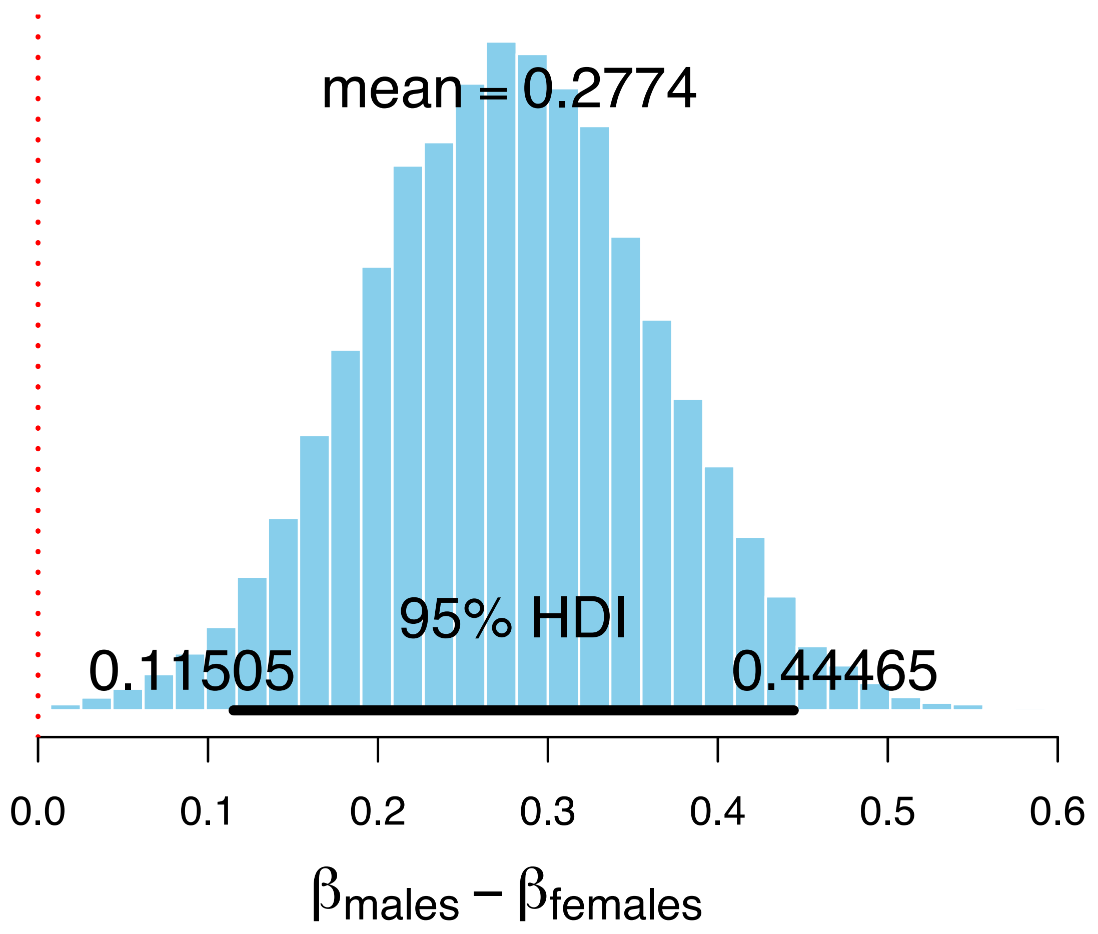


- Since posterior probability distributions are true probabilities, can manipulate them as you would any probability

- For differences between males and females, just subtract one from the other

5. Tentatively Interpret Results

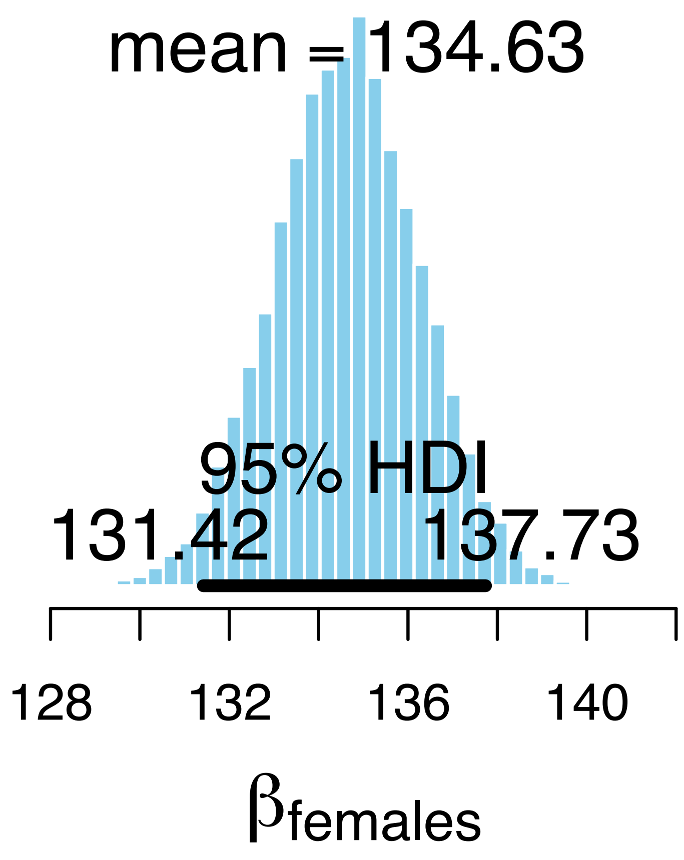
```
difference = zb1[, 2] - zb1[, 1]

par(mfrow = c(1, 1))
histInfo = plotPost(difference, xlab = bquote(beta[males] -
  beta[females]))
abline(v = 0, lwd = 2, lty = 3, col = "red")
```

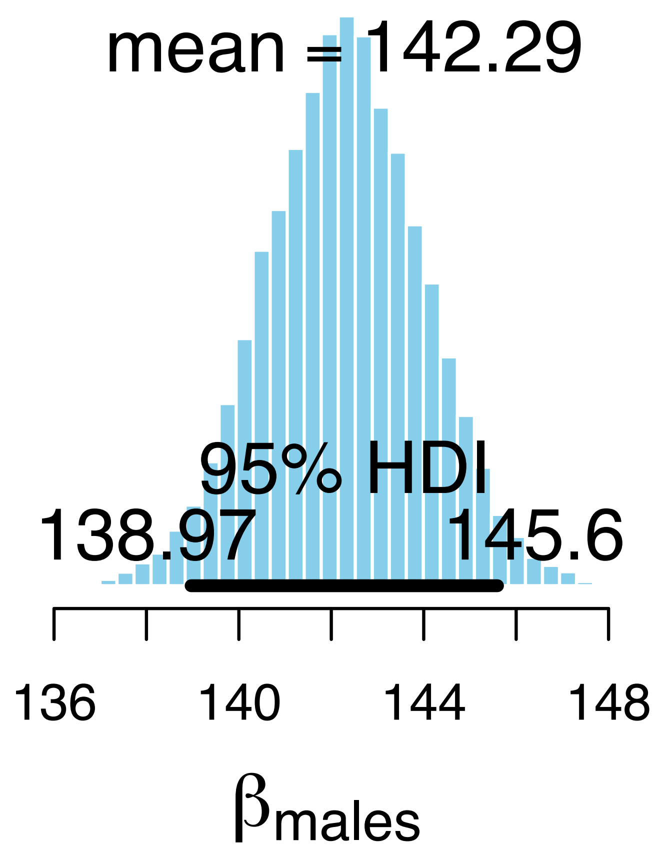


5. Tentatively Interpret Results

Convert back to original scale

```
sigma = zsigma * ySD
b1 = yMean + (zb1 * ySD)

par(mfrow = c(1, 2))
histInfo = plotPost(b1[, 1], xlab = bquote(beta[females]))
histInfo = plotPost(b1[, 2], xlab = bquote(beta[males]))
```

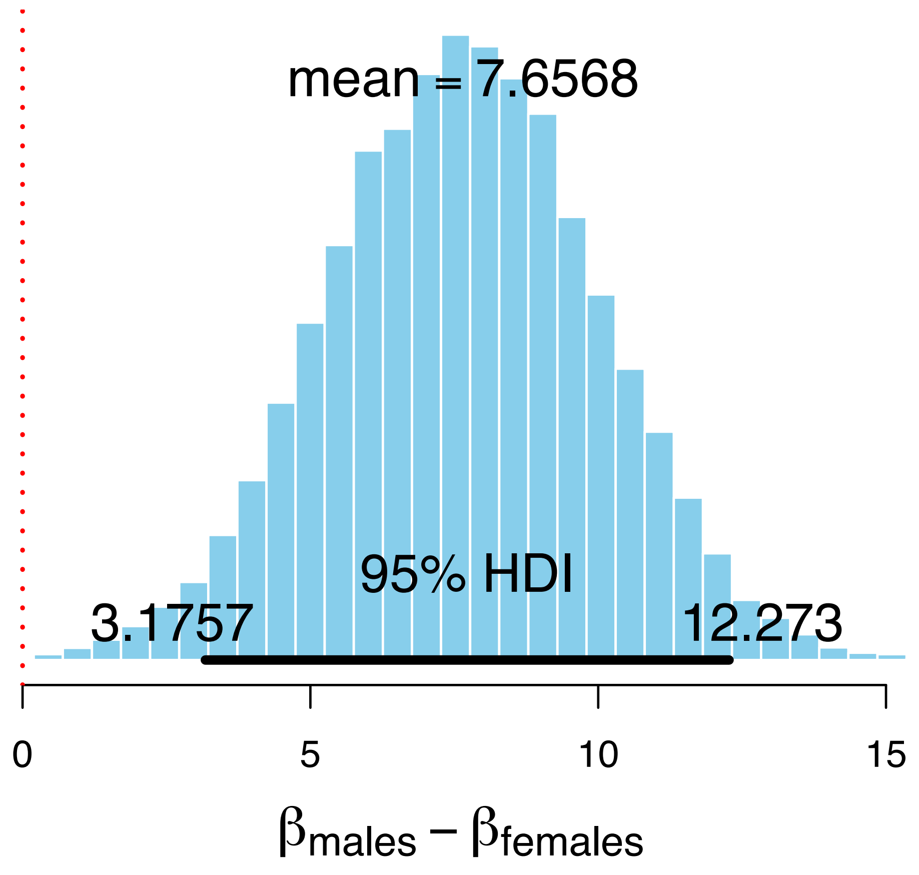




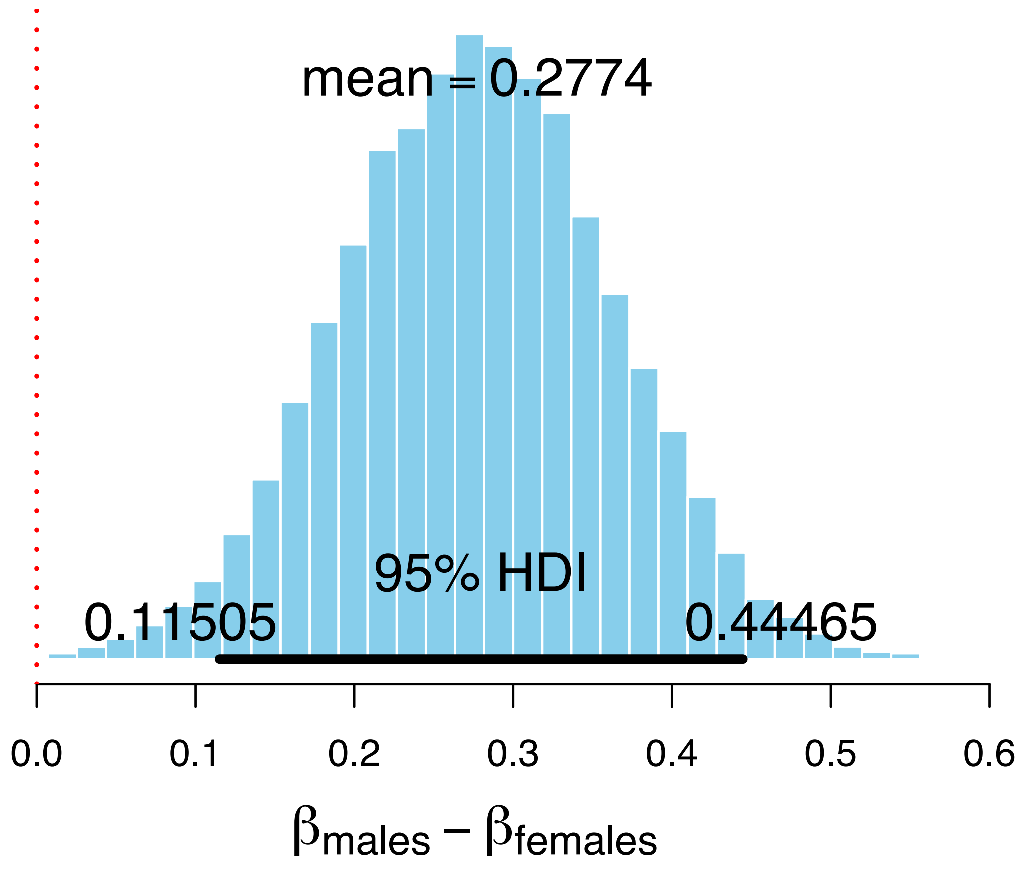
5. Tentatively Interpret Results

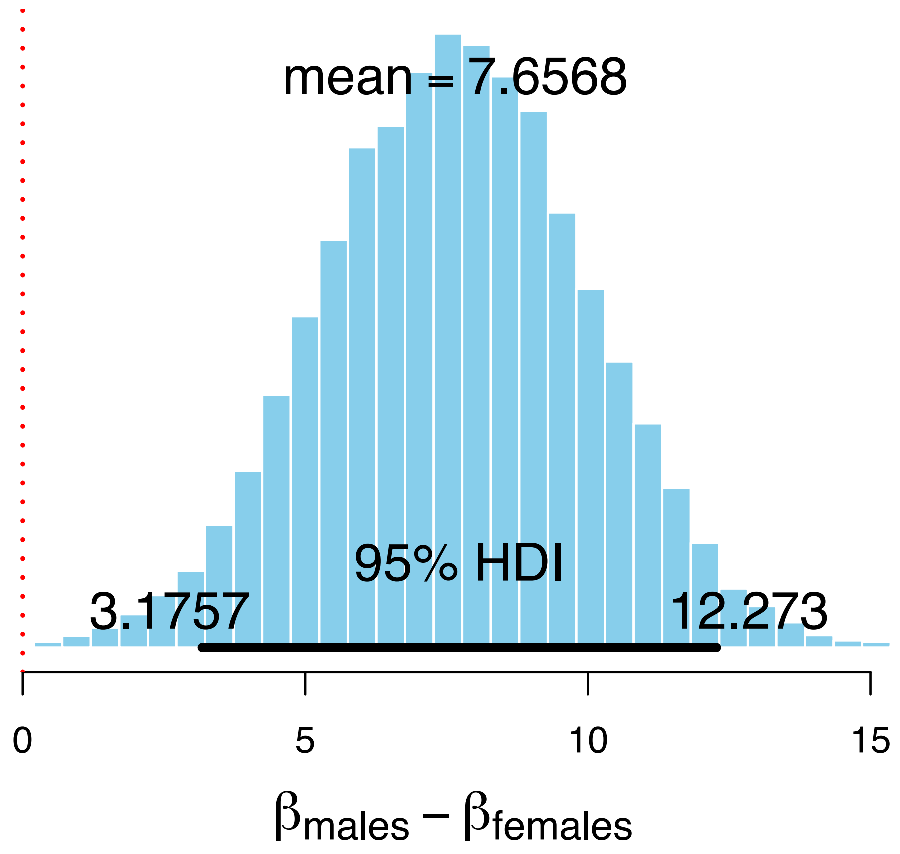
Calculate and plot the difference

```
difference = b1[, 2] - b1[, 1]

par(mfrow = c(1, 1))
histInfo = plotPost(difference, xlab = bquote(beta[males] -
  beta[females]))
abline(v = 0, lwd = 2, lty = 3, col = "red")
```



Note that the interpretation of the data is the same, regardless of scale. Only difference is...scale.





6. Conduct Posterior Predictive Checks:

6. Posterior Predictive Checks

- Calculate the mean, 95% high, and 95% low expected values

```
predMean = apply(ypred, 2, mean)
predLow = apply(ypred, 2, quantile, probs = 0.025)
predHigh = apply(ypred, 2, quantile, probs = 0.975)
```

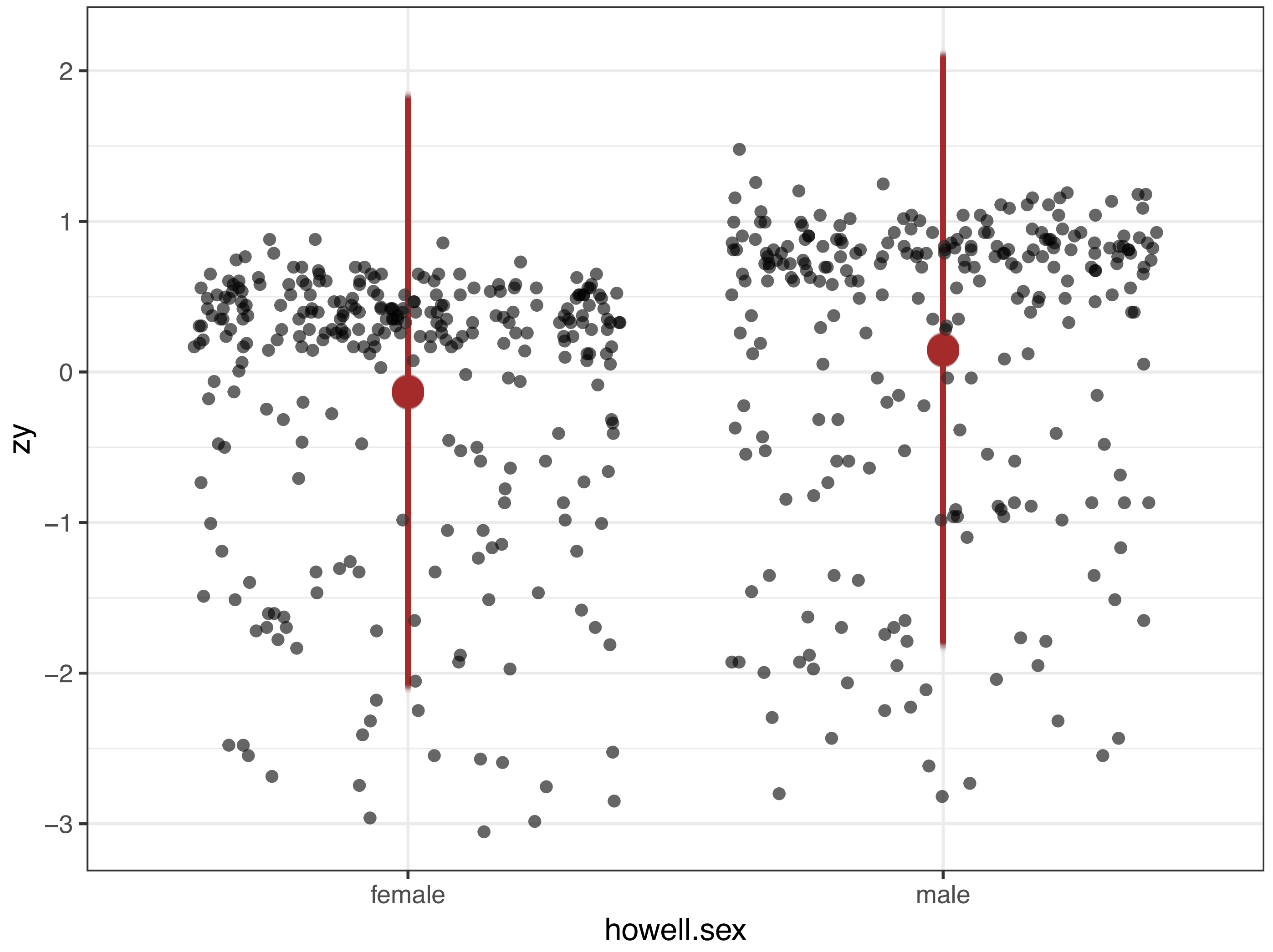
- Combine with the *standardized* height values, and sex data

```
howellCombined = data.frame(zy, howell$sex, predMean, predLow, predHigh)
```

6. Posterior Predictive Checks

- Plot observed and predicted values

```
ggplot(howellCombined) +
  theme_bw() +
  geom_jitter(aes(x = howell.sex, y = zy), height = 0, alpha = 0.6) +
  geom_pointrange(aes(x = howell.sex, y = predMean, ymin = predLow,
    ymax = predHigh), size = 1, colour = "brown", alpha = 0.007)
```



Questions?

- This model works regardless of how many “levels” there are in our nominal predicted variable!!!

Data

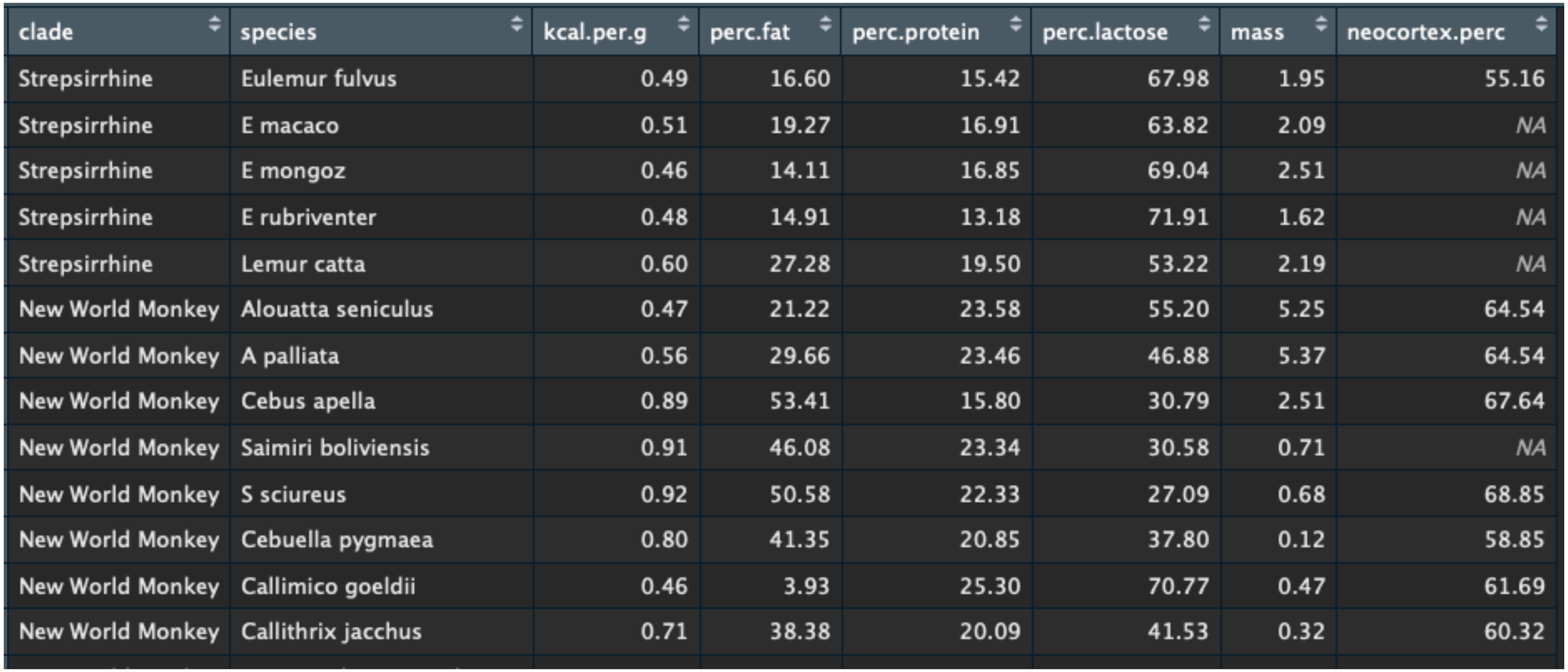
- Milk composition and brain characteristics for a range of primates. From Hinde and Milligan (2011)

Hinde K, Milligan LM (2011) Primate milk synthesis: Proximate mechanisms and ultimate perspectives. *Evolutionary Anthropology* 20: 9-23.
Data distributed as part of the "`rethinking`" R package

Data

- Read data into R

```
milk = read.table(“milk.csv”, header = TRUE, sep = “,”)
```

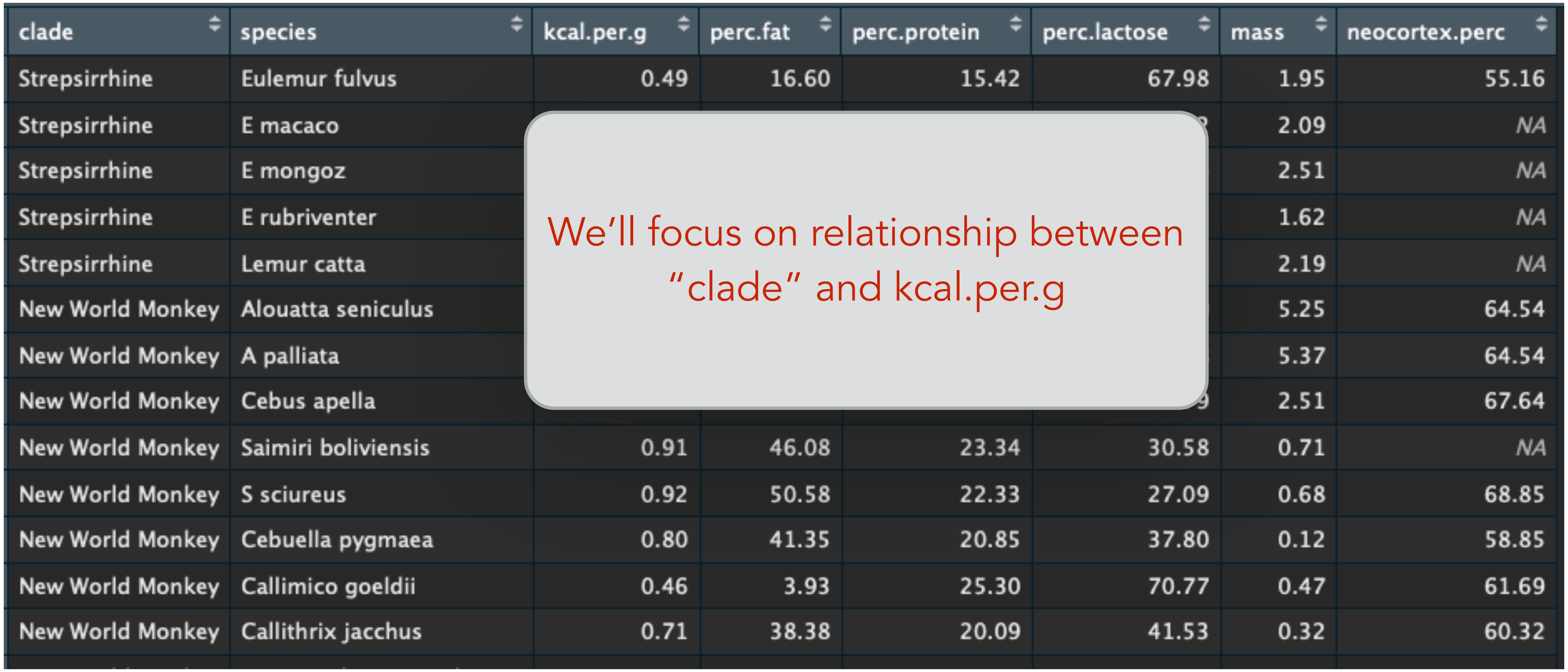


clade	species	kcal.per.g	perc.fat	perc.protein	perc.lactose	mass	neocortex.perc
Strepsirrhine	Eulemur fulvus	0.49	16.60	15.42	67.98	1.95	55.16
Strepsirrhine	E macaco	0.51	19.27	16.91	63.82	2.09	*NA*
Strepsirrhine	E mongoz	0.46	14.11	16.85	69.04	2.51	*NA*
Strepsirrhine	E rubriventer	0.48	14.91	13.18	71.91	1.62	*NA*
Strepsirrhine	Lemur catta	0.60	27.28	19.50	53.22	2.19	*NA*
New World Monkey	Alouatta seniculus	0.47	21.22	23.58	55.20	5.25	64.54
New World Monkey	A palliata	0.56	29.66	23.46	46.88	5.37	64.54
New World Monkey	Cebus apella	0.89	53.41	15.80	30.79	2.51	67.64
New World Monkey	Saimiri boliviensis	0.91	46.08	23.34	30.58	0.71	*NA*
New World Monkey	S sciureus	0.92	50.58	22.33	27.09	0.68	68.85
New World Monkey	Cebuella pygmaea	0.80	41.35	20.85	37.80	0.12	58.85
New World Monkey	Callimico goeldii	0.46	3.93	25.30	70.77	0.47	61.69
New World Monkey	Callithrix jacchus	0.71	38.38	20.09	41.53	0.32	60.32

Data

- Read data into R

```
milk = read.table("milk.csv", header = TRUE, sep = ",")
```



clade	species	kcal.per.g	perc.fat	perc.protein	perc.lactose	mass	neocortex.perc
Strepsirrhine	Eulemur fulvus	0.49	16.60	15.42	67.98	1.95	55.16
Strepsirrhine	E macaco	[illegible]	[illegible]	[illegible]	[illegible]	2.09	NA
Strepsirrhine	E mongoz	[illegible]	[illegible]	[illegible]	[illegible]	2.51	NA
Strepsirrhine	E rubriventer	[illegible]	[illegible]	[illegible]	[illegible]	1.62	NA
Strepsirrhine	Lemur catta	[illegible]	[illegible]	[illegible]	[illegible]	2.19	NA
New World Monkey	Alouatta seniculus	[illegible]	[illegible]	[illegible]	[illegible]	5.25	64.54
New World Monkey	A palliata	[illegible]	[illegible]	[illegible]	[illegible]	5.37	64.54
New World Monkey	Cebus apella	[illegible]	[illegible]	[illegible]	[illegible]	2.51	67.64
New World Monkey	Saimiri boliviensis	0.91	46.08	23.34	30.58	0.71	NA
New World Monkey	S sciureus	0.92	50.58	22.33	27.09	0.68	68.85
New World Monkey	Cebuella pygmaea	0.80	41.35	20.85	37.80	0.12	58.85
New World Monkey	Callimico goeldii	0.46	3.93	25.30	70.77	0.47	61.69
New World Monkey	Callithrix jacchus	0.71	38.38	20.09	41.53	0.32	60.32

We'll focus on relationship between "clade" and kcal.per.g

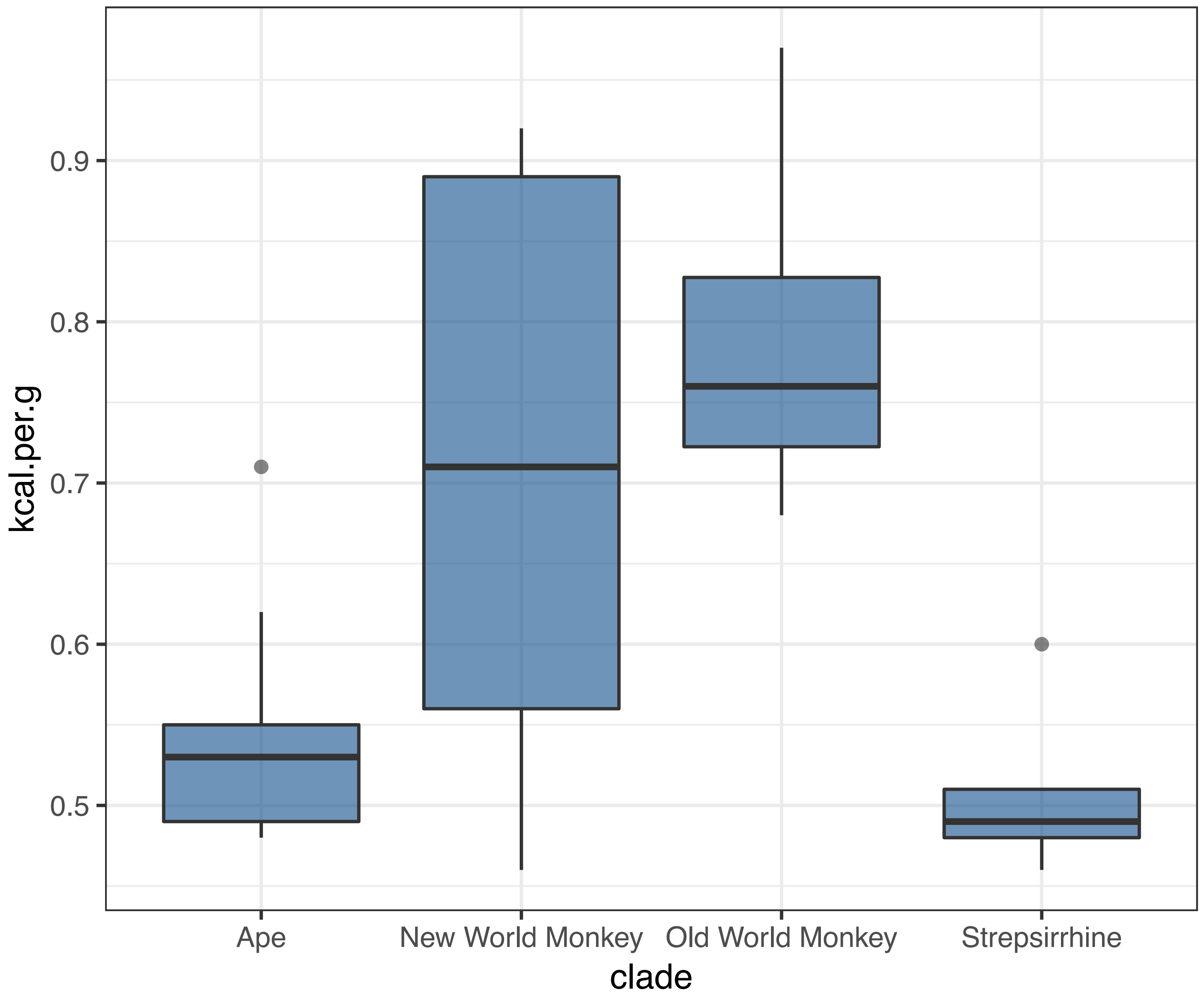
Data

```
summary(milk$clade)

        Ape         New World Monkey    Old World Monkey Strepsirrhine
         9                  9                  6                  5
```

```
summary(milk$kcal.per.g)

    Min.   1st Qu.  Median    Mean   3rd Qu.   Max.
   0.4600  0.4900  0.6000  0.6417  0.7300  0.9700
```

Plot the Data

```
ggplot(milk) +
  theme_bw() +
  geom_boxplot(aes(x = clade, y = kcal.per.g), fill = "dodgerblue4",
    alpha = 0.6)
```



Frequentist Approach
ANOVA

```
anovatest = aov(milk$kcal.per.g ~ milk$clade)


print(model.tables(anovatest))

Tables of effects

 milk$clade
         Ape New World Monkey Old World Monkey Strepsirrhine
    -0.09617          0.07272           0.1466       -0.1337
rep  9.00000          9.00000           6.0000        5.0000
```

Frequentist Approach

ANOVA

```
summary(anovatest)

            Df Sum Sq Mean Sq F value   Pr(>F)    
milk$clade   3 0.3492 0.11640   7.654 0.000858 ***
Residuals   25 0.3802 0.01521                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Frequentist Approach

Linear regression

```
model = lm(milk$kcal.per.g ~ milk$clade)


summary(model)

Call:
lm(formula = milk$kcal.per.g ~ milk$clade)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.254444 -0.058333 -0.005556  0.074444  0.205556 

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 0.54556    0.04111  13.271 8.05e-13 ***
milk$cladeNew World Monkey  0.16889    0.05813   2.905 0.007575 ** 
milk$cladeOld World Monkey  0.24278    0.06500   3.735 0.000975 ***
milk$cladeStrepsirrhine    -0.03756    0.06879  -0.546 0.589920    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1233 on 25 degrees of freedom
Multiple R-squared:  0.4787,	Adjusted R-squared:  0.4162 
F-statistic: 7.654 on 3 and 25 DF,  p-value: 0.0008577
```

Bayesian Approach
Organize the data

- No standardization because `y` ranges from 0 to 1 already

```
y = milk$kcal.per.g
N = length(y)

x = as.numeric(milk$clade)
xNames = levels(milk$clade)
nxLevels = length(xNames)
```

Bayesian Approach
Organize the data

- Create as a list to send to Stan

```
dataList = list (
  y = y,
  N = N,
  x = x,
  nxLevels = nxLevels
)
```

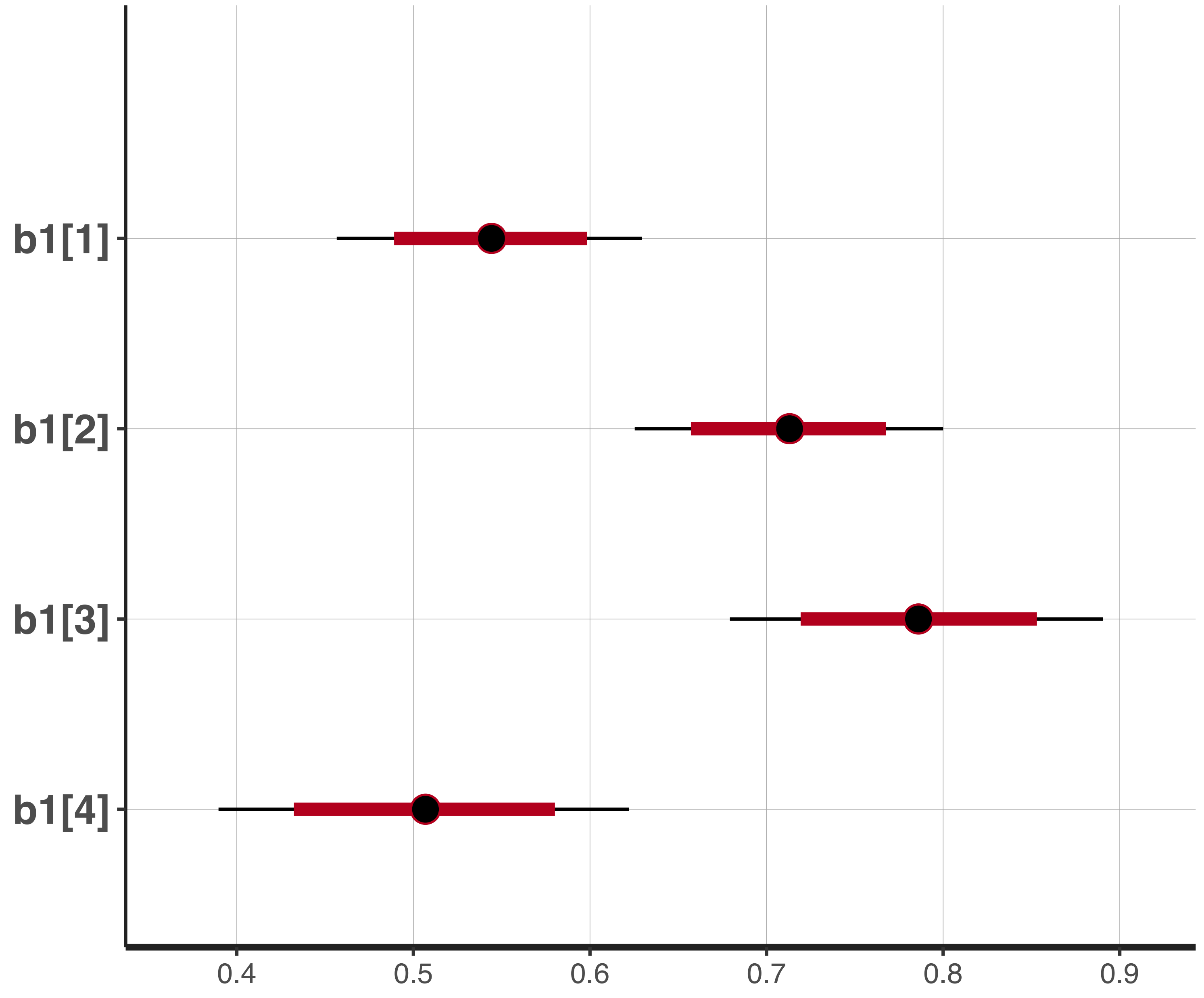
Bayesian Approach
Run model

- No need to re-specify, just run it!

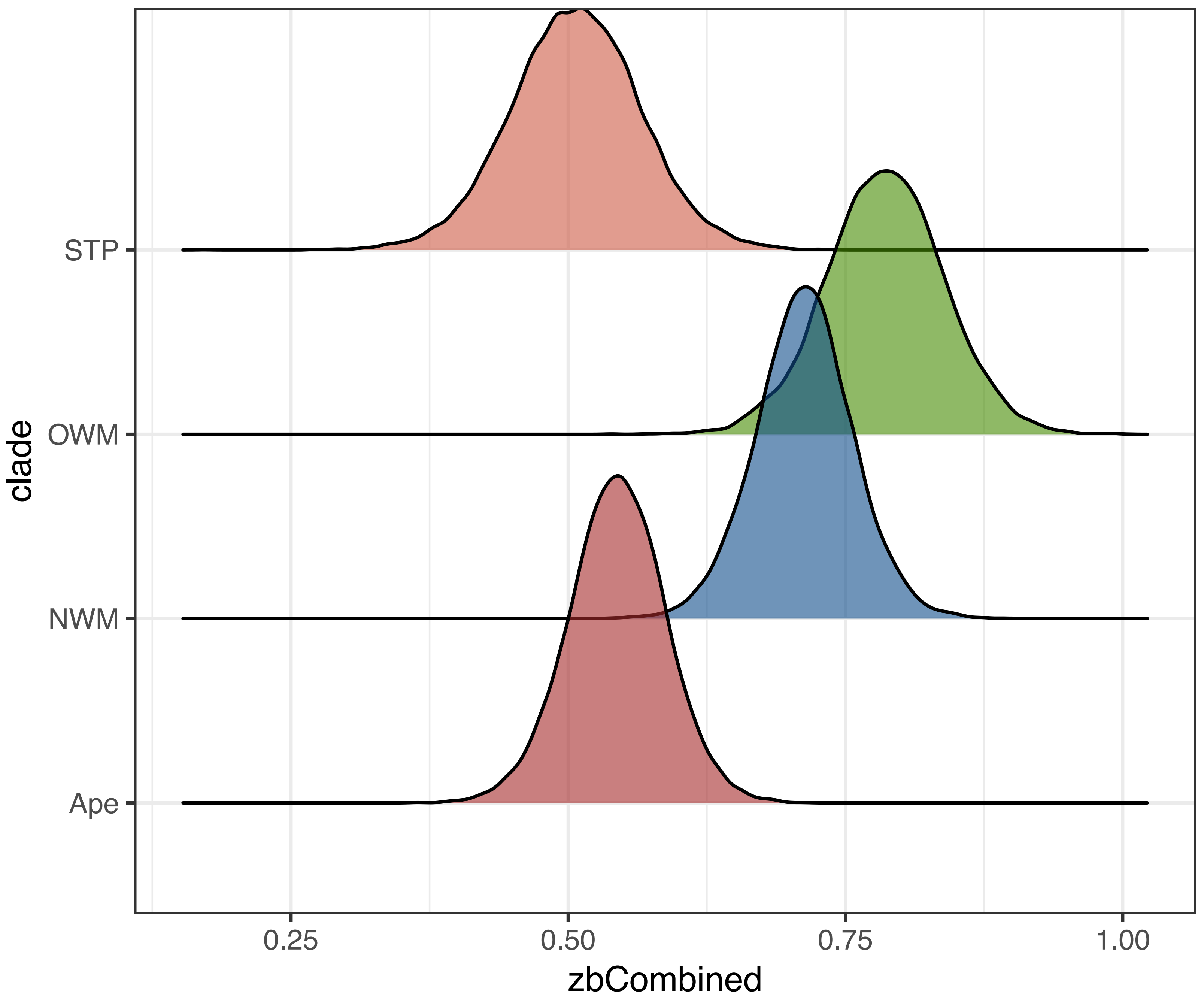
```
stanFit <- stan(file = "model.stan",
                data = dataList,
                pars = c("b1", "sigma", "y_pred"),
                warmup = 2000,
                iter = 7000,
                chains = 3)
```

Bayesian Approach
Check results

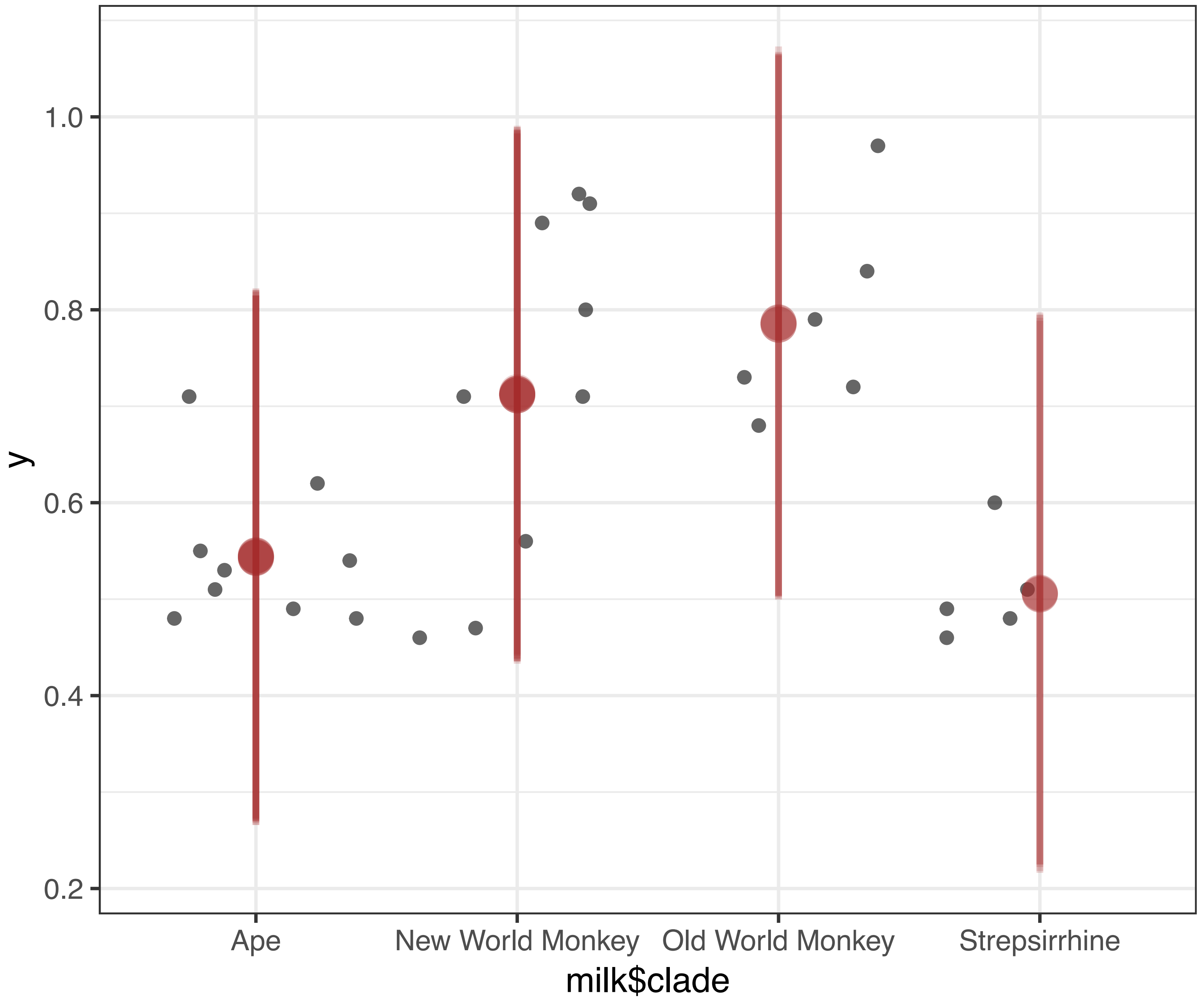
- Will skip MCMC evaluation in class



Bayesian Approach
Check results



Bayesian Approach
Posterior predictive check



- What if we don’t like the assumption of equal s.d. across levels?

```
sd(yFemale)

[1] 25.93023
```

```
sd(yMale)

[1] 28.87132
```

- What if we don't like the assumption of equal s.d. across levels?

```
sd(yFemale)

[1] 25.93023
```

```
sd(yMale)

[1] 28.87132
```

Just change that part of the model!!