Logistic Regression
(Dichotomous predicted variable)

Tim Frasier

Goals and General Idea

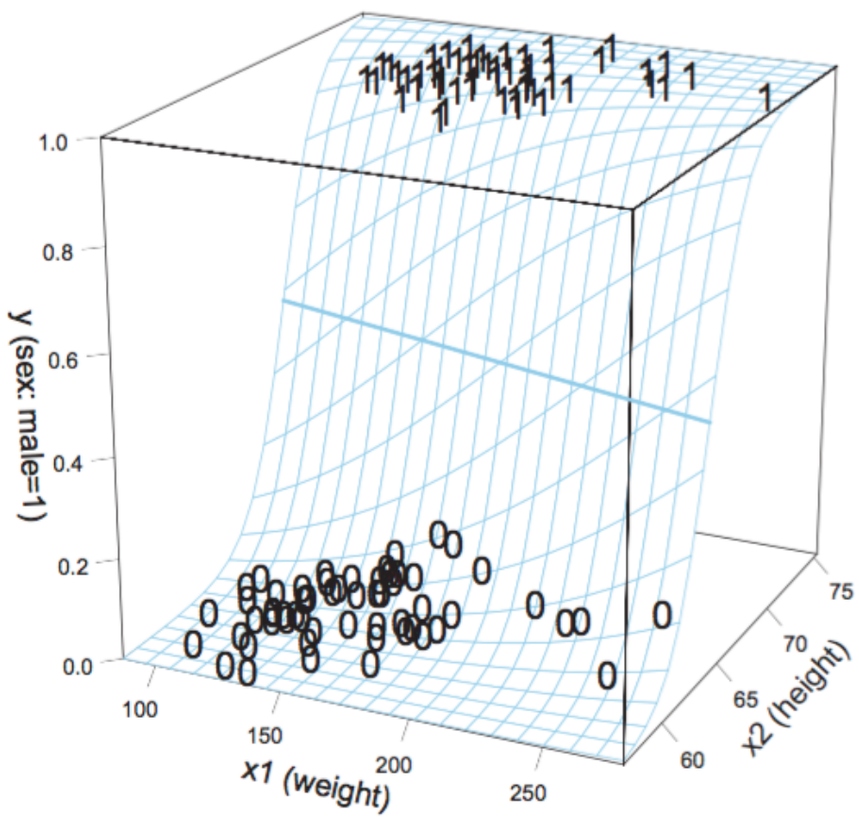
Goals

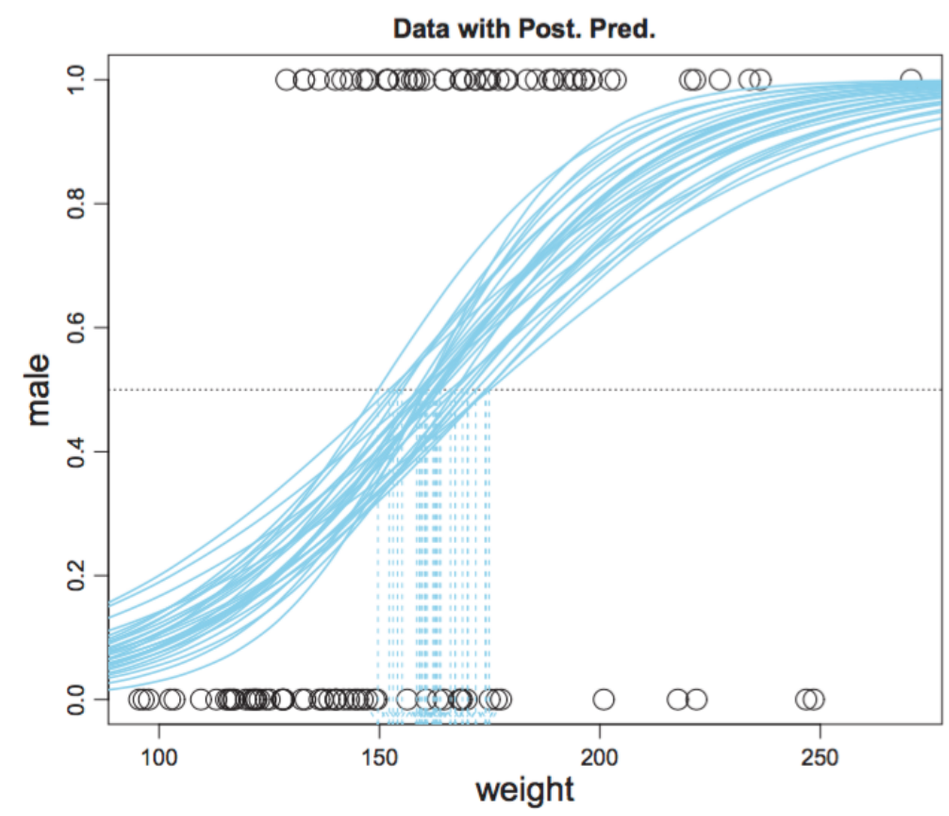
When would we use this type of analysis?

- When the predicted variable can be either a 1 or a 0
  - Probability of a patient being cured (or not) based on age and drug dosage
  - Is a species expected to inhabit (or not) a particular area based on the ecological conditions
  - etc.

Approach

- Probability of being male (1) or female (0) based on height & weight
- Is considerable overlap for both predictor variables

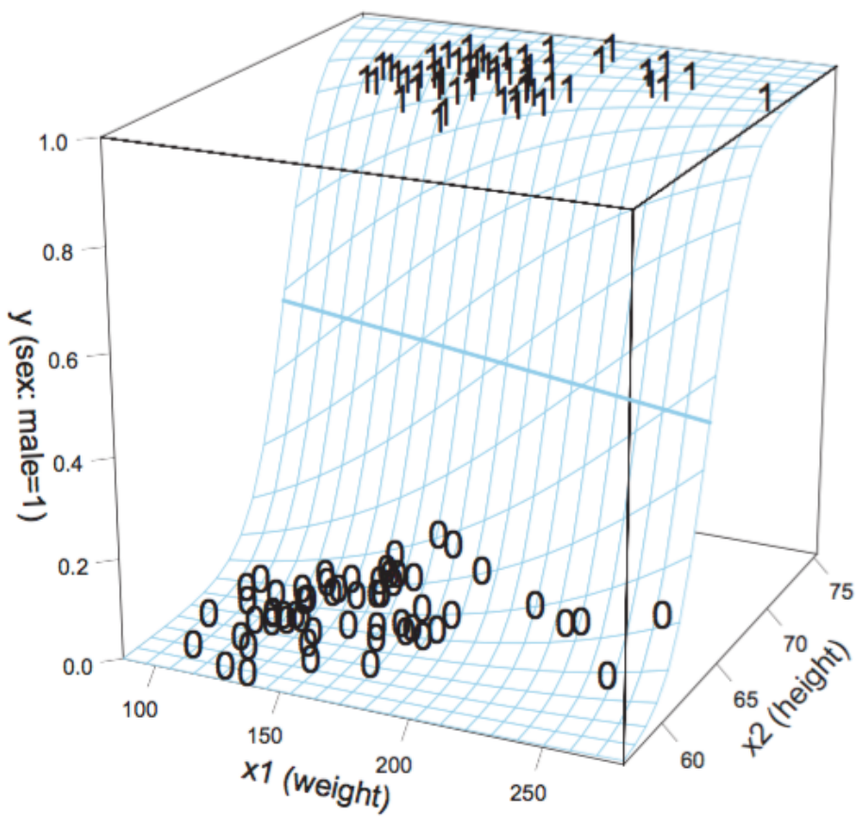


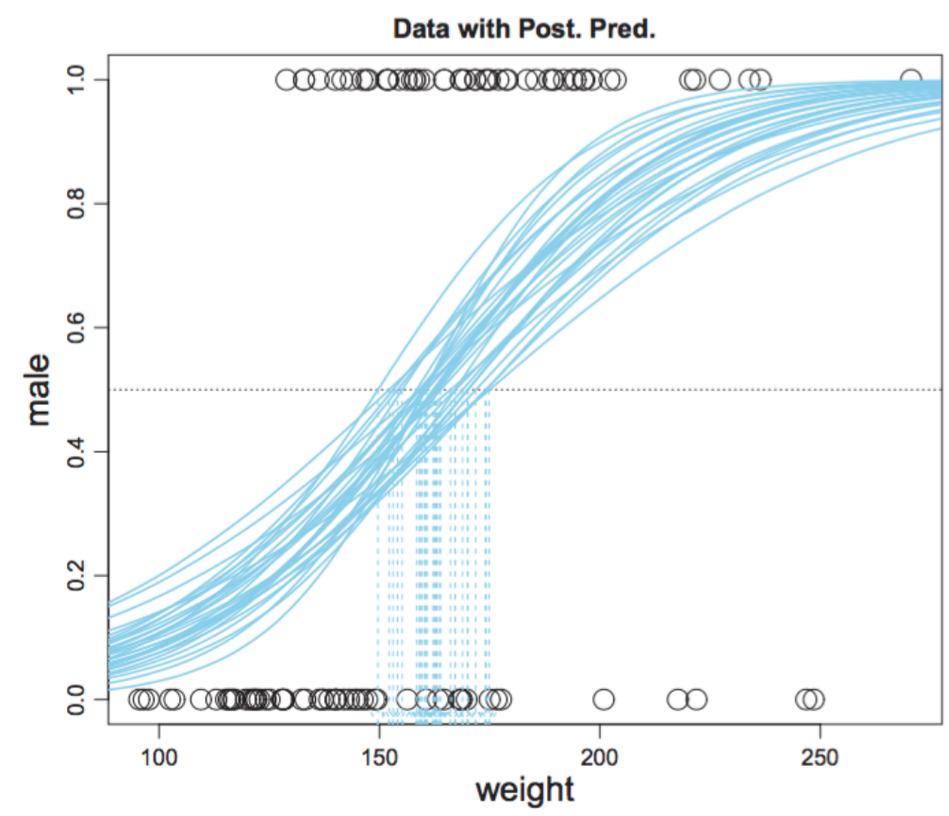


From Kruschke (2015) p. 623 & 627.

Approach

- Can estimate the probability of being 1 (or 0) given height and weight
- Will use a logistic transform of a linear combination of the predictor variables



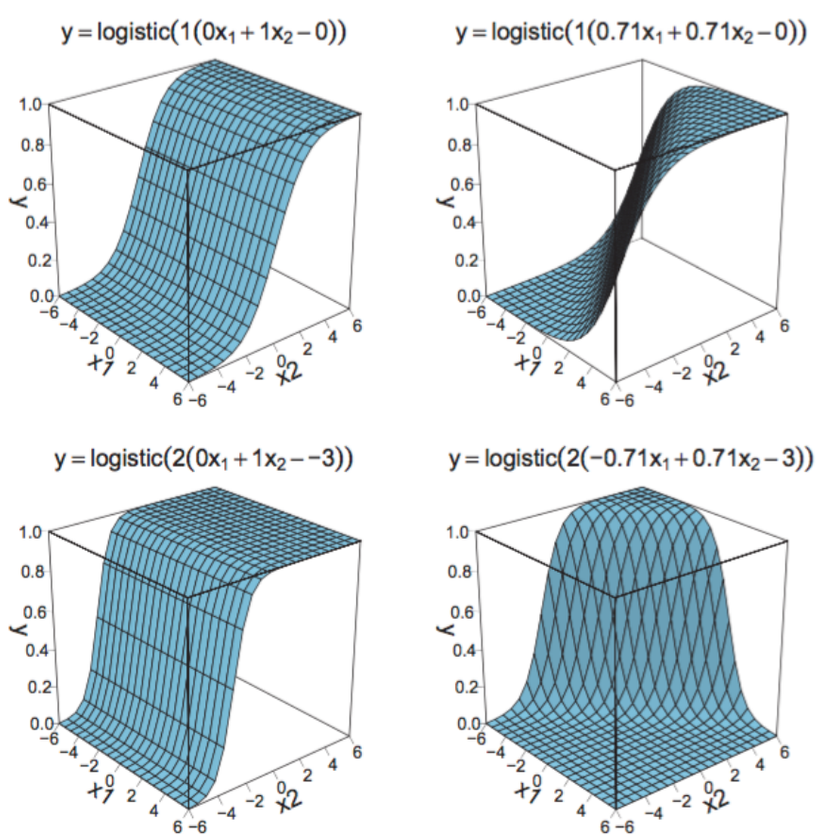


From Kruschke (2015) p. 623 & 627.

Approach

- With multiple linear regression, we calculate E[*y*] as a linear combination of predictors

- With multiple logistic regression, the linear combination is transformed by a sigmoidal (logistic) function ("squashes" values to fall between 0 and 1 - to represent at what predictor values the predicted variable switches from 0 to 1)



From Kruschke (2015) p. 438

Approach

$$mu = \mathrm{sig}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots)$$

Approach

$$mu = \mathrm{sig}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots)$$

- Sigmoidal function is another name for the logistic function

$$\mathrm{sig}(x) = 1 / (1 + \exp(-x))$$

Approach

$$mu = \mathrm{sig}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots)$$

- Sigmoidal function is another name for the logistic function

$$\mathrm{sig}(x) = 1 / (1 + \exp(-x))$$

- Can also be written as a logit function

$$\mathrm{logit}(mu) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$$

Approach

Let's explore this a bit to get a feel for it

- First, write a function that will calculate this for us

```
logistic = function(x) {
   1 / (1 + exp(-x))
 }
```

Approach

Let's explore this a bit to get a feel for it

- Then, try a few positive numbers to see what happens

```
x = 0
logistic(x)

[1] 0.5
```

```
x = 1
logistic(x)

[1] 0.7310586
```

```
x = 2
logistic(x)

[1] 0.8807971
```

Approach

Let's explore this a bit to get a feel for it

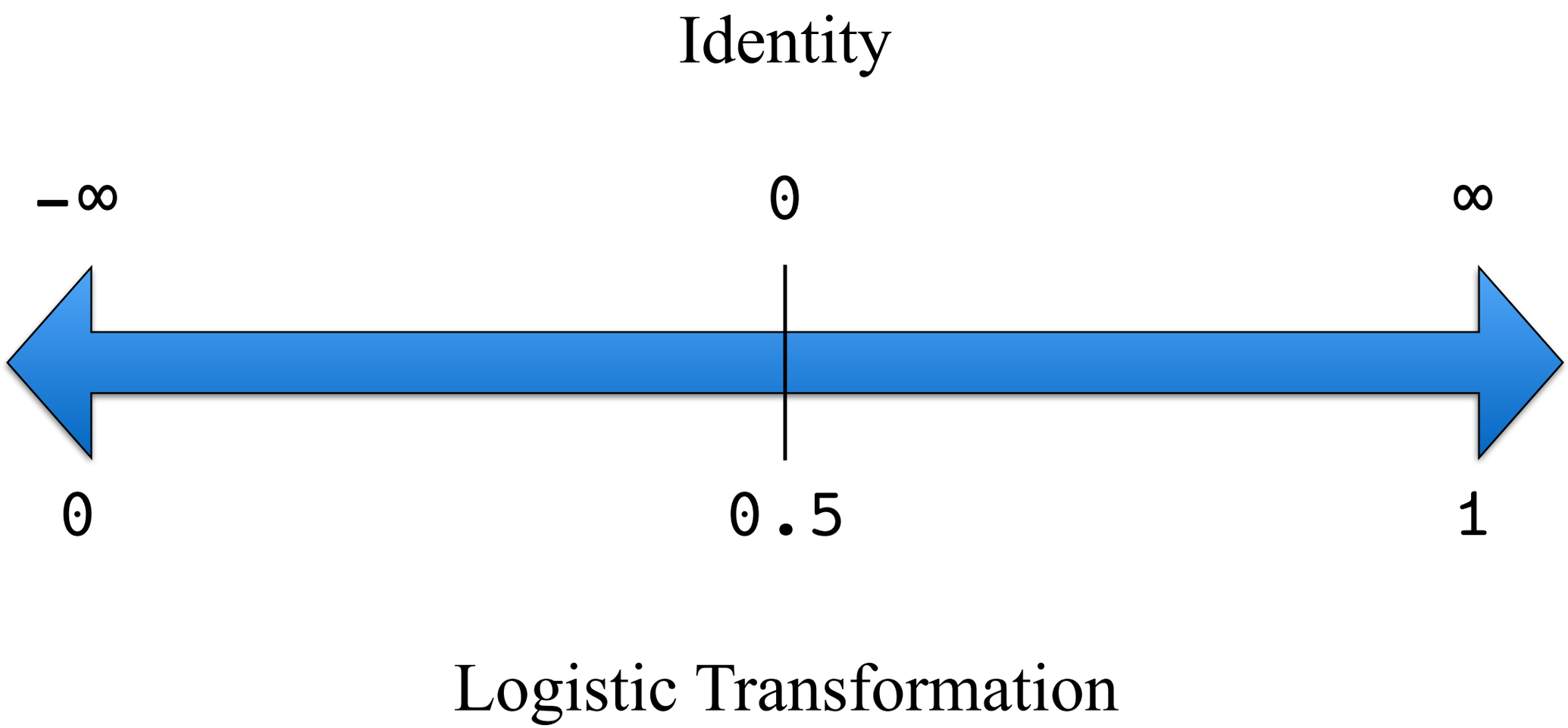
- Then, try a few negative numbers to see what happens

```
x = -1
logistic(x)

[1] 0.2689414
```

```
x = -2
logistic(x)

[1] 0.1192029
```

Approach

Let's explore this a bit to get a feel for it



Approach

Let's explore this a bit to get a feel for it

- Generate some x-values

```
x = rnorm(n = 100, mean = 0, sd = 5)
```

- Generate y-values based on identity of x-values

```
y1 = x
```

- Generate logistic transformation of x

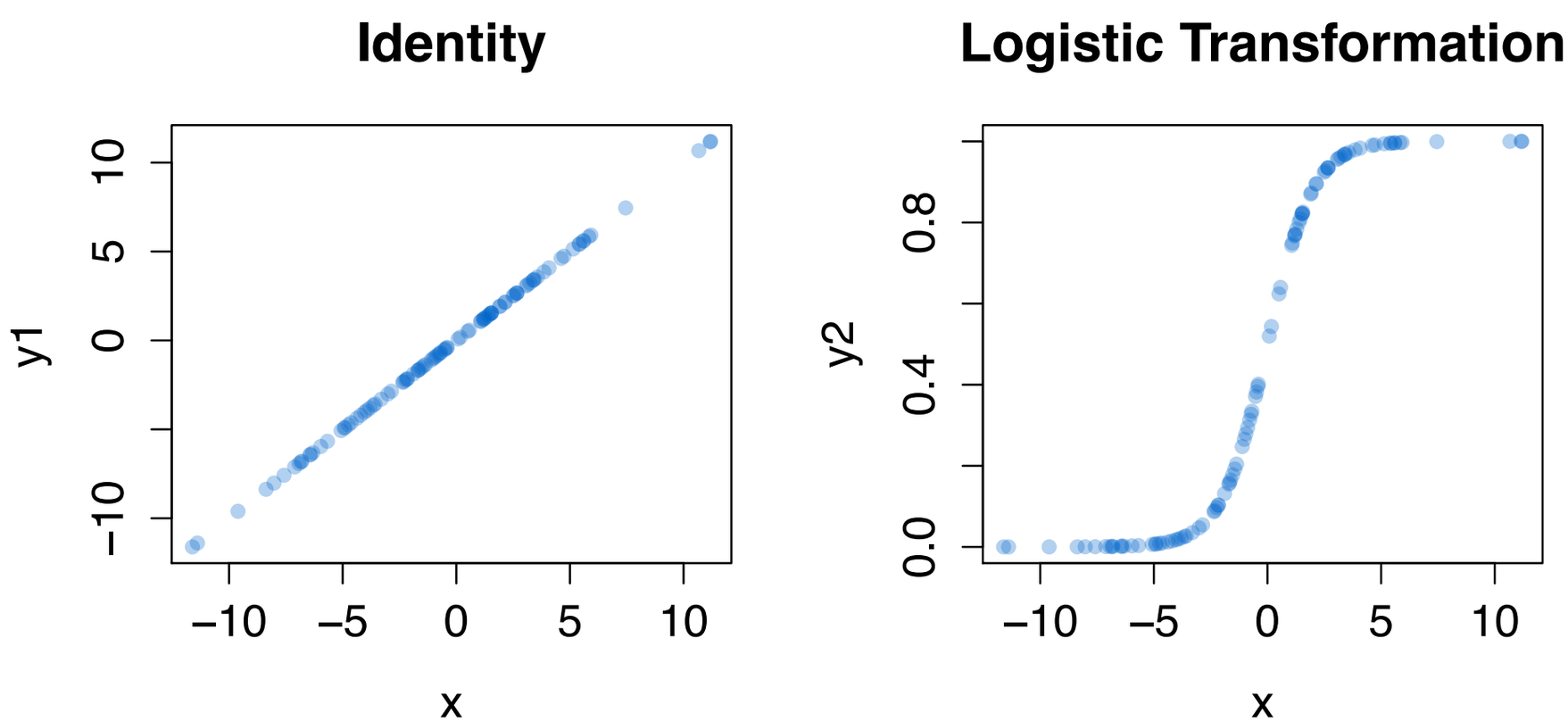
```
y2 = logistic(x)
```

Approach

Let's explore this a bit to get a feel for it

- Let's plot then and see the difference

```
par(mfrow = c(1, 2))
plot(x, y1, pch = 16, main = "Identity", col = rgb(0, 0, 1, 0.3))
plot(x, y2, pch = 16, main = "Logistic Transformation", col = rgb(0, 0, 1,
    0.3))
```



Approach

- Predicted value then becomes the mean for a Bernoulli distribution

$$y = \text{bernoulli}(mu)$$

$$mu = \text{sig}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots)$$

Bernoulli Distribution

- Can be either 1 or 0
- Is 1 with probability *mu*

Approach

- In Stan:

$$y = \text{bernoulli\_logit}(mu)$$

$$mu = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$$

The Data

Load Libraries & Functions

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

Data

- Modified height and weight data from before (but also including sex now)

```
htwt = read.table(HtWt2.csv", header = TRUE, sep = ",")
```

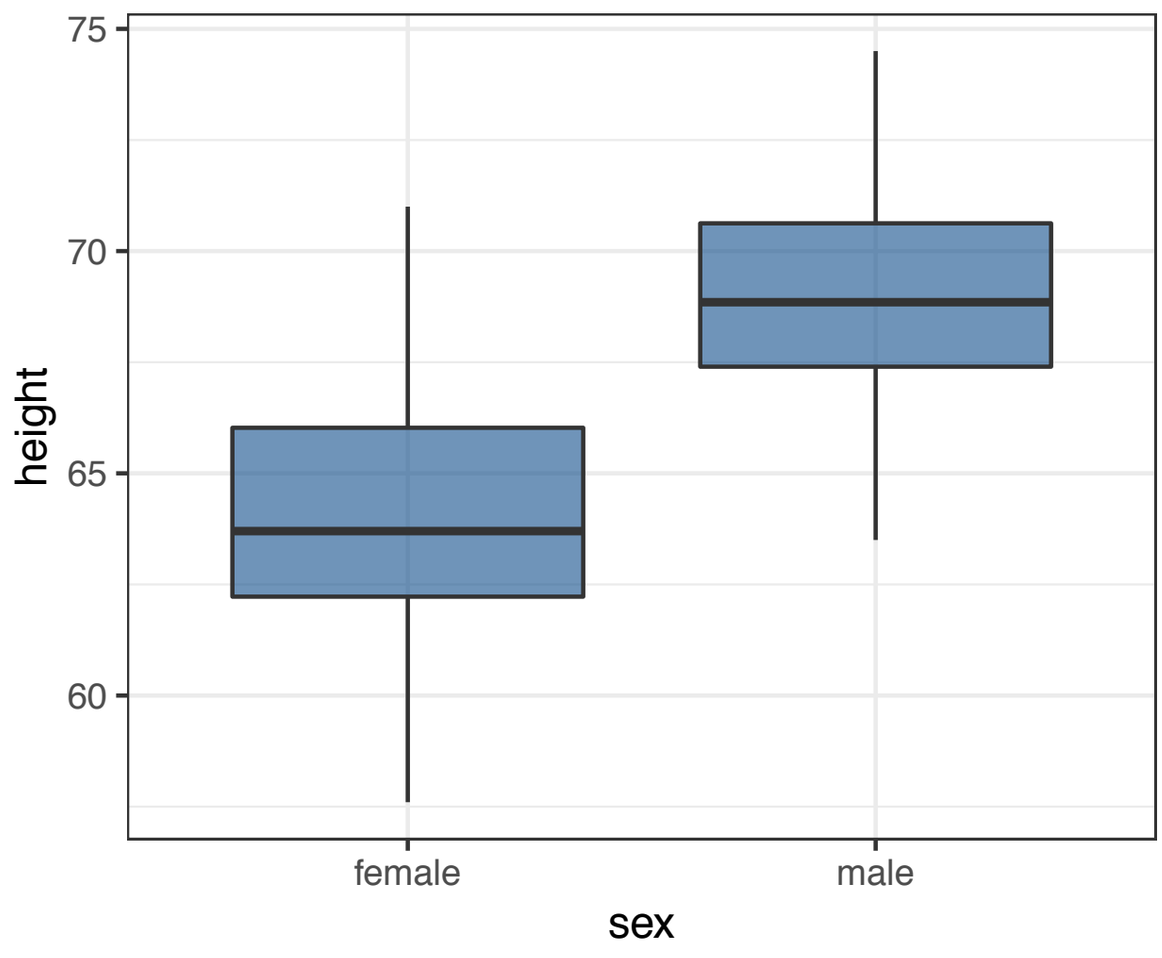
htwt

	male	height	weight
1	0	63.2	168.7
2	0	68.7	169.8
3	0	64.8	176.6
4	0	67.9	246.8
5	1	68.9	151.6
6	1	67.8	158.0
7	1	68.2	168.6
8	0	64.8	137.2
9	1	64.3	177.0
10	0	64.7	128.0
11	1	66.9	168.4
12	1	66.9	136.2
13	1	67.1	160.3
14	1	70.2	233.9
15	1	67.4	171.7

Data

- Take a look at the data as box plots
  - Height vs sex

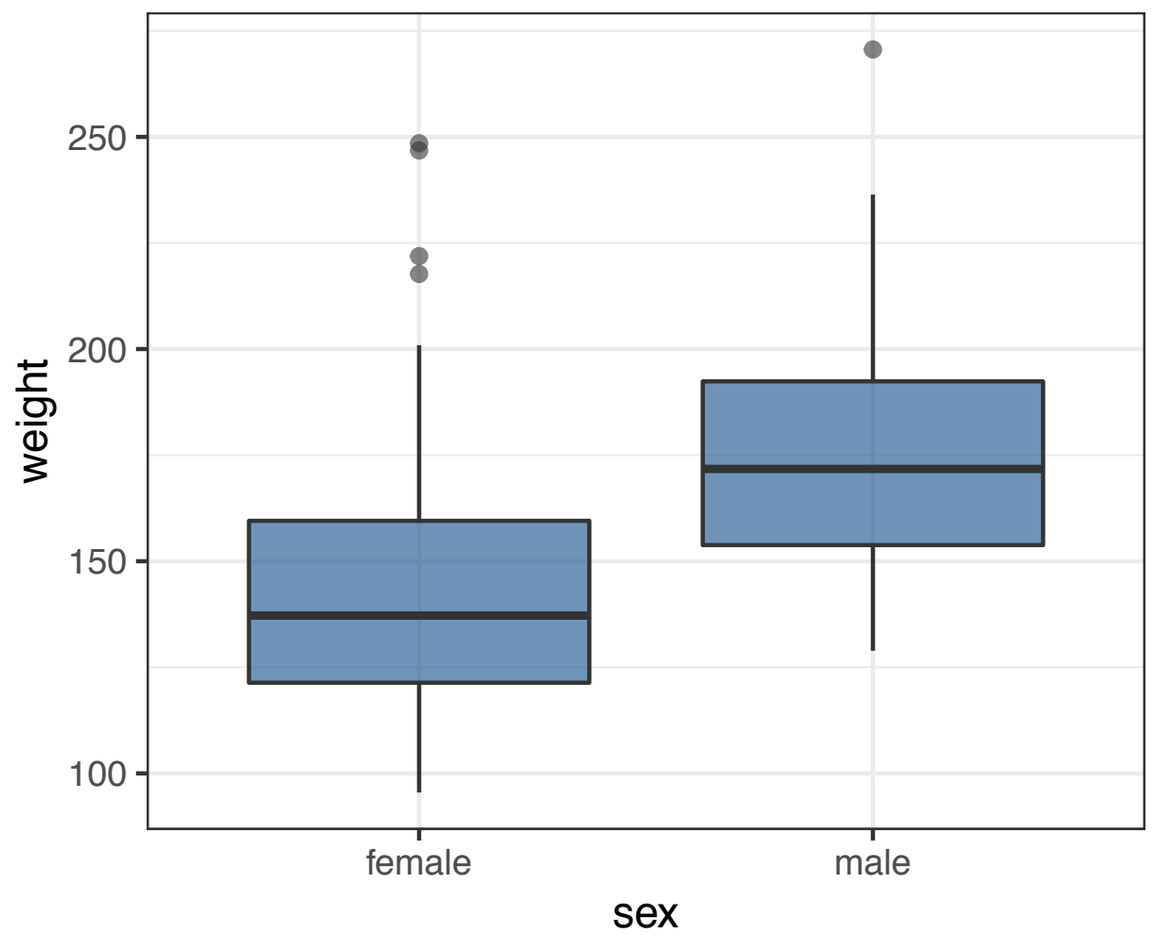
```
ggplot(htwt) +
  theme_bw() +
  geom_boxplot(aes(x = sex, y = height), fill = "dodgerblue4", alpha =
   0.6)
```



Data

- Take a look at the data as box plots
  - Weight vs sex

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



Bayesian Approach

Organize the Data

```
y = as.numeric(htwt$sex)  # female will be 1 and male will be 2
y = y - 1                 # female will be 0 and male will be 1 (logistic...
                          # ...regression requires predicted variables to...
                          # ...be 1 or 0)
N = length(y)
```

Organize the Data

```
#--- height ---#
height = htwt$height
heightMean = mean(height)
heightSD = sd(height)
zheight = (height - heightMean) / heightSD

#--- Weight ---#
weight = htwt$weight
weightMean = mean(weight)
weightSD = sd(weight)
zweight = (weight - weightMean) / weightSD
```

Make Data List For Stan

```
dataList = list(
  y = y,
  N = N,
  height = zheight,
  weight = zweight
)
```

Define the Model

$$y = \text{bernoulli\_logit}(mu)$$

$$mu = \beta_0 + \beta_1 height + \beta_2 weight$$

Define the Model

$$y = \text{bernoulli\_logit}(mu)$$

$$mu = \beta_0 + \beta_1 height + \beta_2 weight$$

The “black box” into which we can put any equations that we have dealt with before (or more)

Define the Model

- The **data** block

```
modelstring = "
  data {
    int N;                  // Sample size
    int y[N];               // Indices of 0s and 1s (predicted variable)
    vector[N] height;       // Vector of height data
    vector[N] weight;       // Vector of weight data
  }
```

Define the Model

- The **parameters** block

```
parameters {
    real b0;
    real b1;     // Effect of height on prob. of sex
    real b2;     // Effect of weight on prob. of sex
  }
```

Define the Model

- The **model** block

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

    // Likelihood
    for (i in 1:N) {
      mu[i] = b0 + (b1 * height[i]) + (b2 * weight[i]);
      y[i] ~ bernoulli_logit(mu[i]);
    }

    // Priors
    b0 ~ normal(0, 1);
    b1 ~ normal(0, 1);
    b2 ~ normal(0, 1);
  }
```

Define the Model

- The **generated quantities** block

```
generated quantities {
  // Definitions
  vector[N] mu_pred;
  vector[N] y_pred;

  for (i in 1:N) {
    mu_pred[i] = b0 + (b1 * height[i]) + (b2 * weight[i]);
    y_pred[i] = bernoulli_logit_rng(mu_pred[i]);
  }
}
"
writeLines(modelstring, con = "model.stan")
```

Run the Model

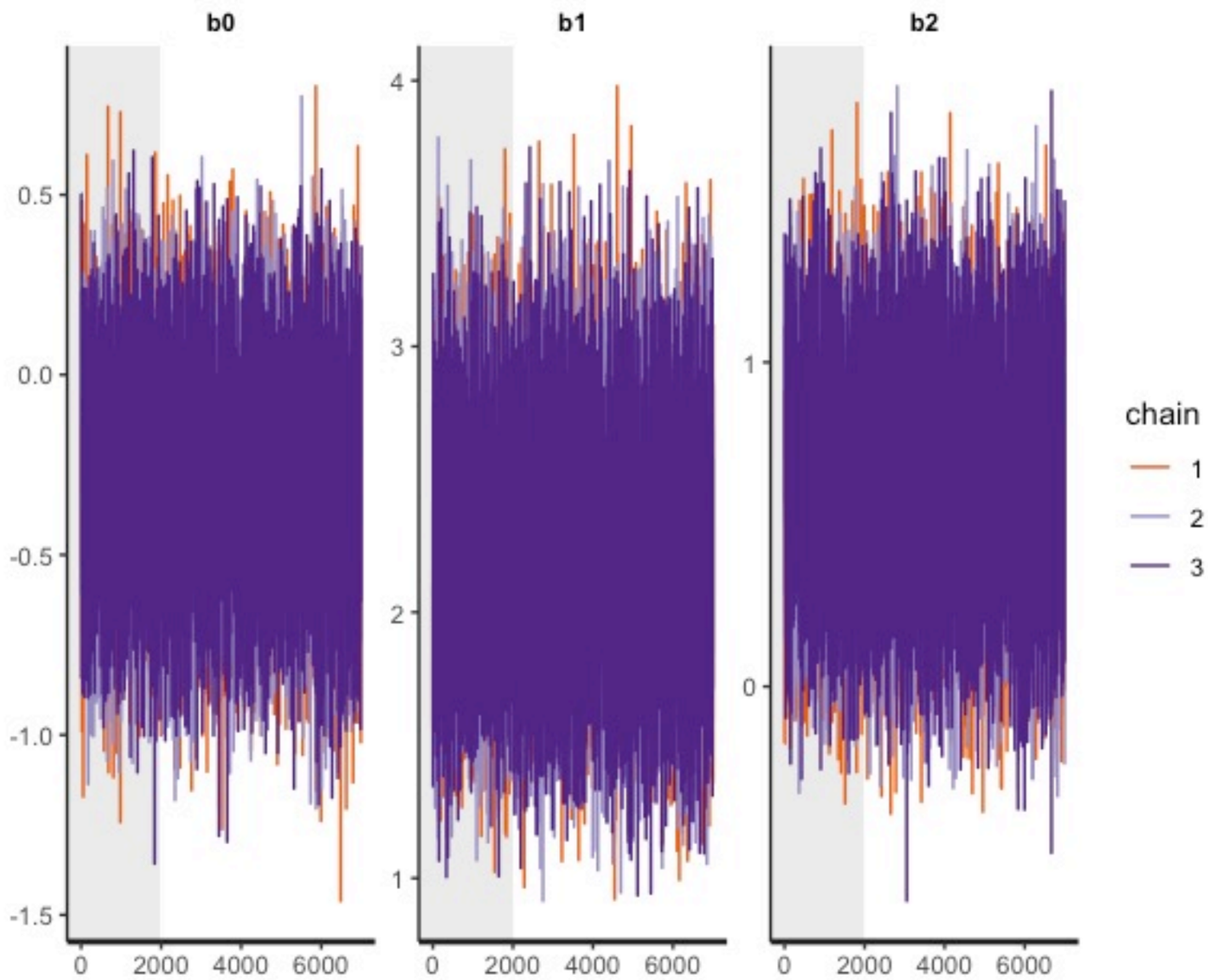
```
stanFit = stan(file = "model.stan",
               data = dataList,
               pars = c("b0", "b1", “b2”, “y_pred”),
               warmup = 2000,
               iter = 7000,
               chains = 3)
```

Evaluate MCMC Performance

```
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
b0    -0.28    0.00 0.27  -0.81  -0.46  -0.28  -0.10   0.25 13230    1
b1     2.22    0.00 0.41   1.45   1.94   2.21   2.49   3.07 13011    1
b2     0.63    0.00 0.31   0.04   0.41   0.62   0.84   1.25 13107    1
lp__ -42.28    0.01 1.25 -45.54 -42.86 -41.97 -41.36 -40.86  7278    1
```

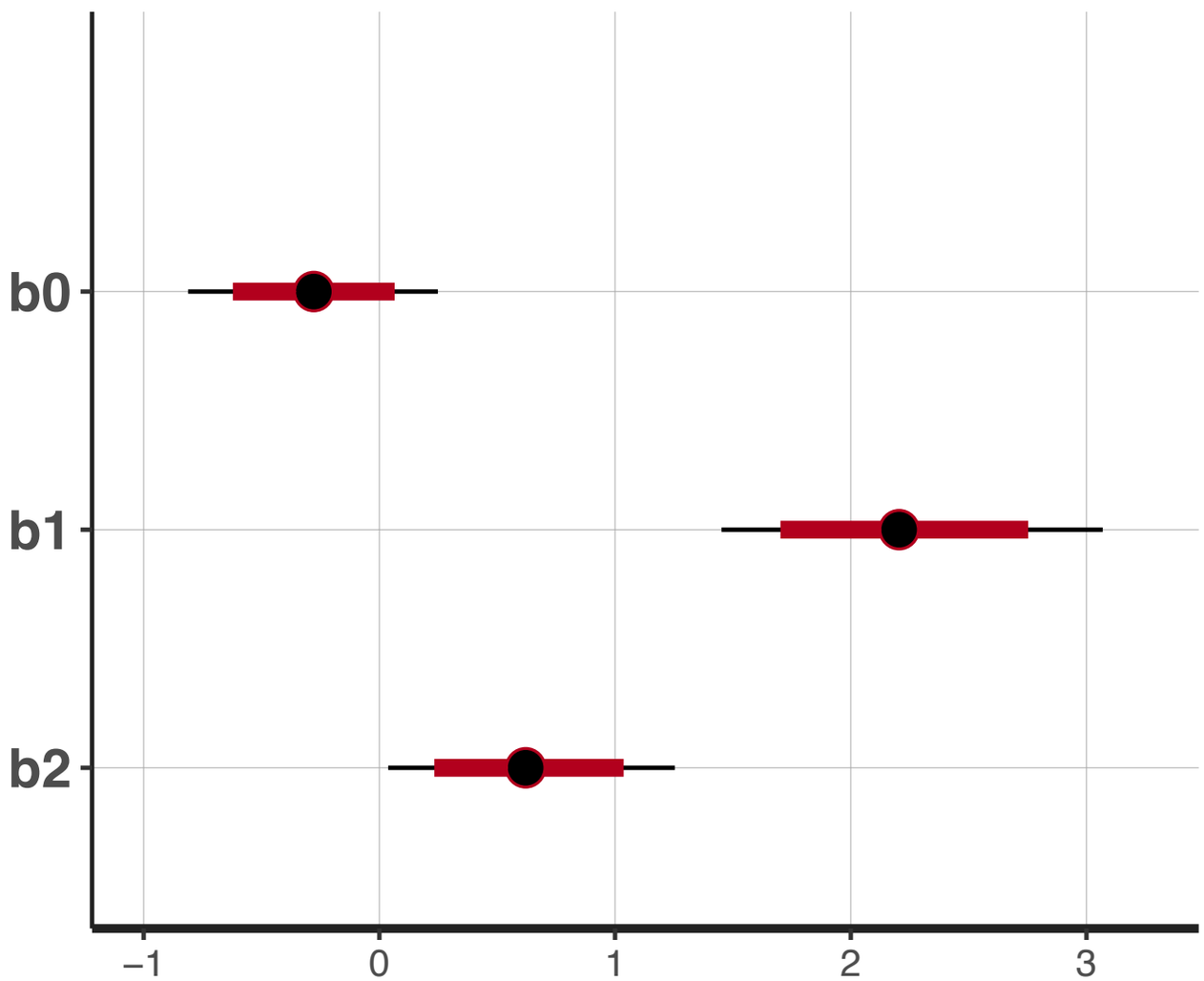
Evaluate MCMC Performance

```
stan_trace(stanFit, pars = c("b0", "b1", "b2"), inc_warmup = TRUE)
```



Evaluate Results

```
stan_plot(stanFit, par = c("b0", "b1", "b2"))
```



Evaluate Results

- Extract the data

```
mcmcChains = as.data.frame(stanFit)

zb0 = mcmcChains[, "b0"]
zb1 = mcmcChains[, "b1"]
zb2 = mcmcChains[, "b2"]
```

Evaluate Results

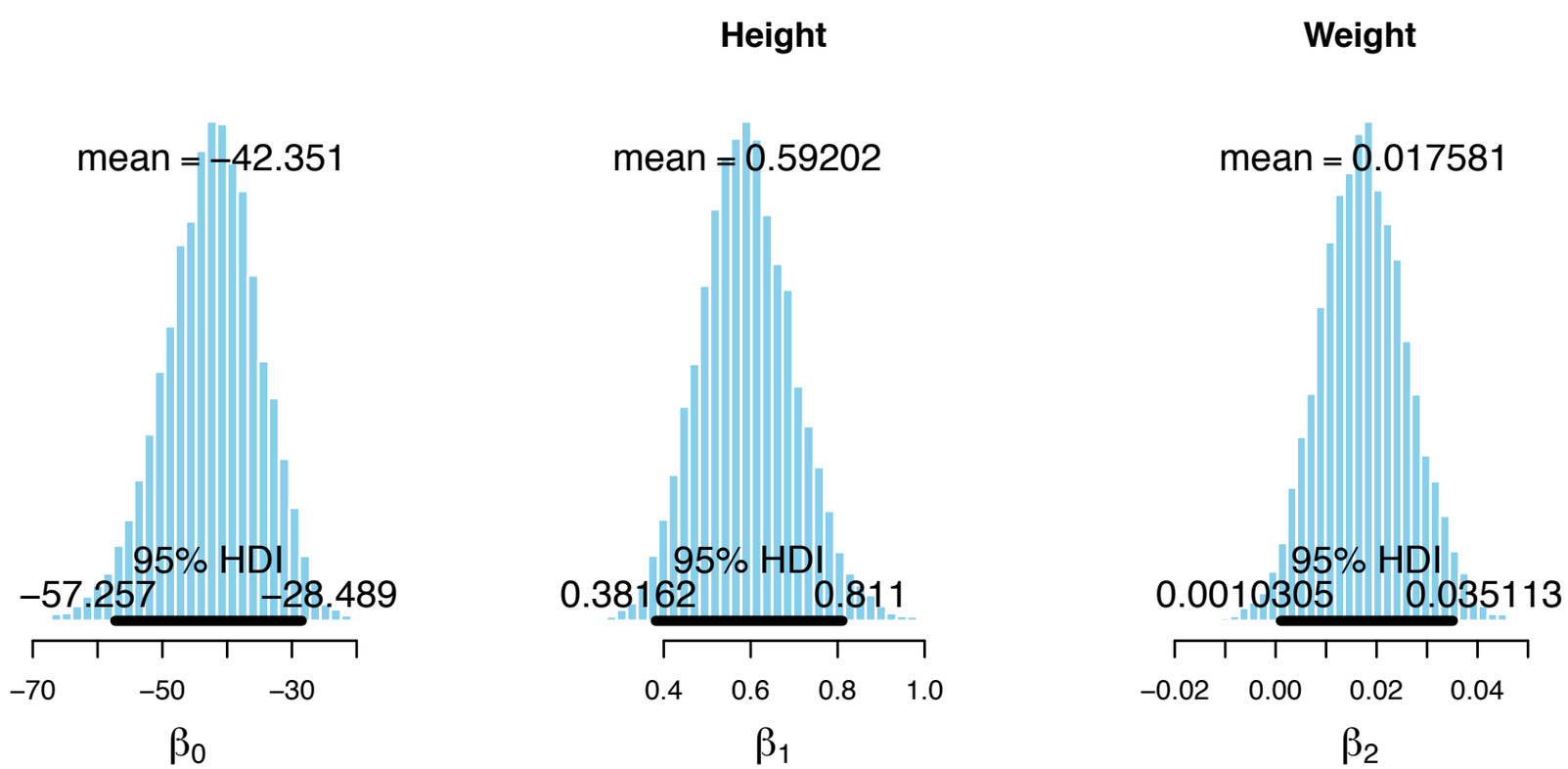
- Convert to original scale

```
b1 = zb1 / heightSD
b2 = zb2 / weightSD
b0 = zb0 - (((zb1 * heightMean) / heightSD) + ((zb2 * weightMean) / weightSD))
```

Evaluate Results

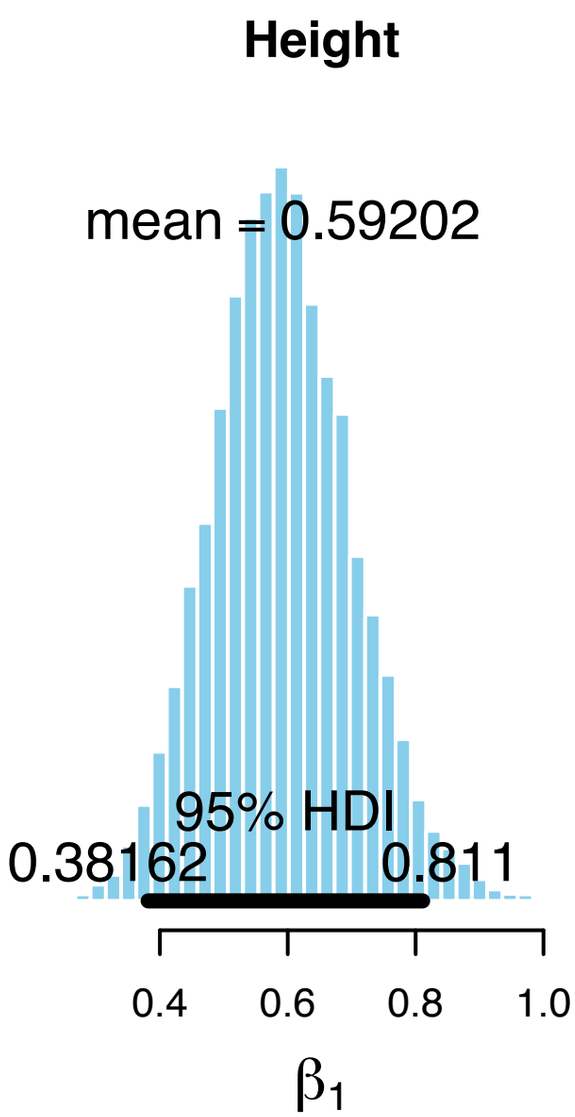
- Plot posterior distributions

```
par(mfrow = c(1, 3))
histInfo = plotPost(b0, xlab = bquote(beta[0]))
histInfo = plotPost(b1, xlab = bquote(beta[1]), main = "Height")
histInfo = plotPost(b2, xlab = bquote(beta[2]), main = "Weight")
```



View Posteriors

Interpreting Posteriors



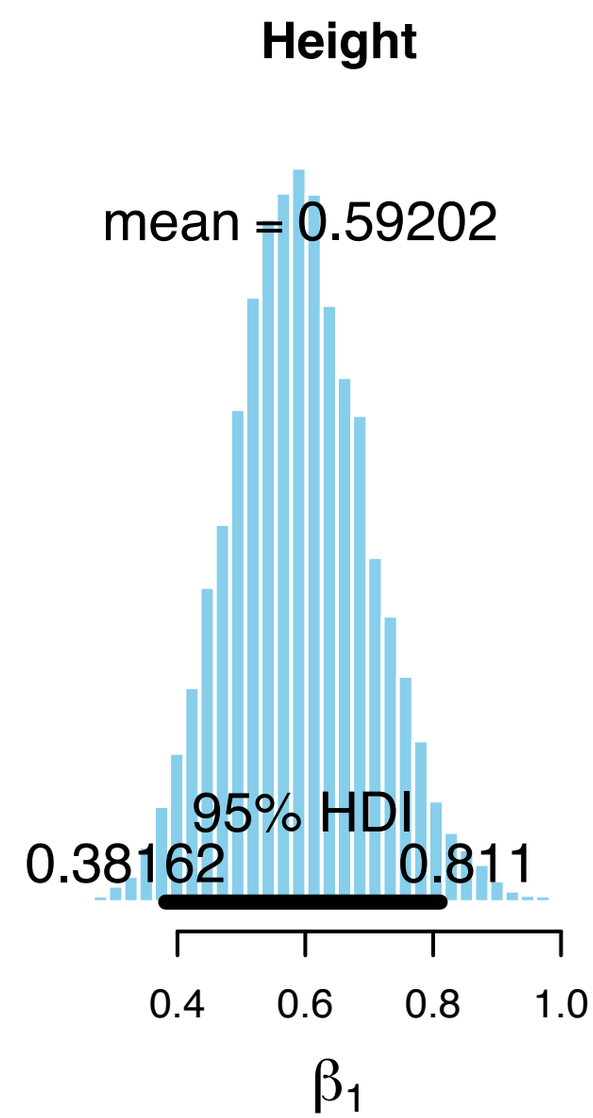
what does this mean?

Interpreting Posteriors

- Coefficients of logistic regression tell us about the log odds

$$\text{logit}(mu) = \log \frac{p(y=1)}{p(y=0)}$$

- When x_i increases by 1 unit, the log odds increase β_i units



- When height increases by 1 inch, the log odds of being a male increases by ~0.592

How Well Does the Model Fit the Data?

Assessing Model Fit
Single parameters (height)

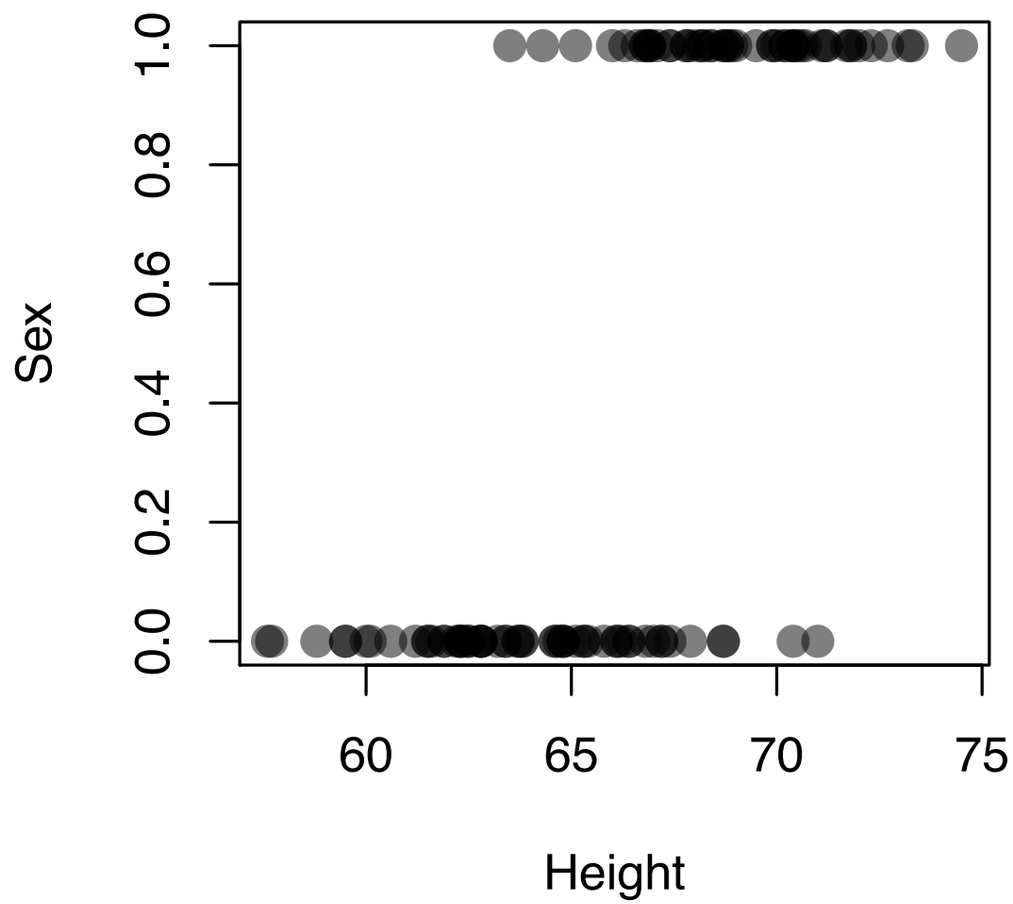
- Can ask, "How does probability of being male change as a function of height, for someone of average weight*"?
  - Evaluate effects of one parameter, while holding the others constant

*Graph will differ if you use different weights, but this will give you a general idea of fit

Assessing Model Fit
Single parameters (height)

- First, plot raw data of sex values versus height data

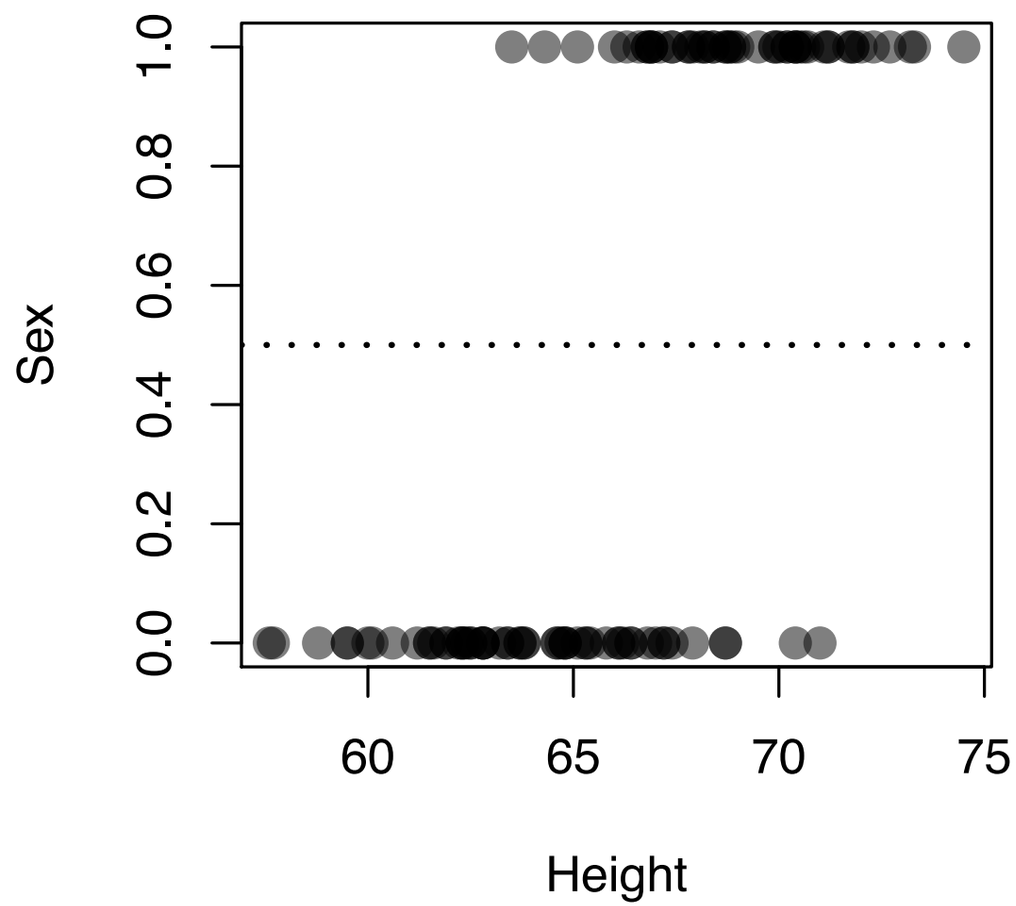
```
plot(height, y, xlab = "Height", ylab = "Sex", pch = 16, cex = 1.5,
     col = rgb(0, 0, 0, 0.5))
```



Assessing Model Fit
Single parameters (height)

- Draw a dashed line showing probability of 0.5

```
abline(h = 0.5, lty = "dotted", lwd = 2)
```



Assessing Model Fit
Single parameters (height)

- Generate 30 steps of the posteriors, spread throughout the chain

```
xNew = floor(seq(1, chainLength, length = 30))
```

- Find the maximum difference between the smallest and largest values of the predictor variable

```
xRange = max(height) - min(height)
```

Assessing Model Fit
Single parameters (height)

- Generate 200 x values to be used for plot
  - Range from min value - (`0.1 * xRange`) to max value + (`0.1 * xRange`)
  - So that lines span width of plot window, rather than stopping right at end points

```
xComb = seq(min(height) - 0.1 * xRange, max(height) + 0.1 * xRange, length = 200)
```

Assessing Model Fit
Single parameters (height)

- Draw lines

```
for (i in xNew) {
    lines(xComb , 1 / (1 + exp(-(b0[i] + (b1[i] * xComb) + (b2[i] *
          mean(weight))))), lwd = 1.5, col = rgb(0, 0, 1, 0.3))
}
```

Assessing Model Fit
Single parameters (height)

- Draw lines

```
for (i in xNew) {
    lines(xComb , 1 / (1 + exp(-(b0[i] + (b1[i] * xComb) + (b2[i] *
          mean(weight))))), lwd = 1.5, col = rgb(0, 0, 1, 0.3))
}
```

The x-values for our lines (200 of ’em)

Assessing Model Fit

Single parameters (height)

- Draw lines

```
for (i in xNew) {
    lines(xComb , 1 / (1 + exp(-(b0[i] + (b1[i] * xComb) + (b2[i] *
          mean(weight))))), lwd = 1.5, col = rgb(0, 0, 1, 0.3))
}
```

The y-values, which are the logistic transformation of...

Assessing Model Fit
Single parameters (height)

- Draw lines

```
for (i in xNew) {
    lines(xComb , 1 / (1 + exp(-(b0[i] + (b1[i] * xComb) + (b2[i] *
          mean(weight))))), lwd = 1.5, col = rgb(0, 0, 1, 0.3))
}
```

The “intercept” plus...

Assessing Model Fit
Single parameters (height)

- Draw lines

```
for (i in xNew) {
    lines(xComb , 1 / (1 + exp(-(b0[i] + (b1[i] * xComb) + (b2[i] *
          mean(weight))))), lwd = 1.5, col = rgb(0, 0, 1, 0.3))
}
```

The β_1 coefficient times the x-values plus ...

Assessing Model Fit
Single parameters (height)

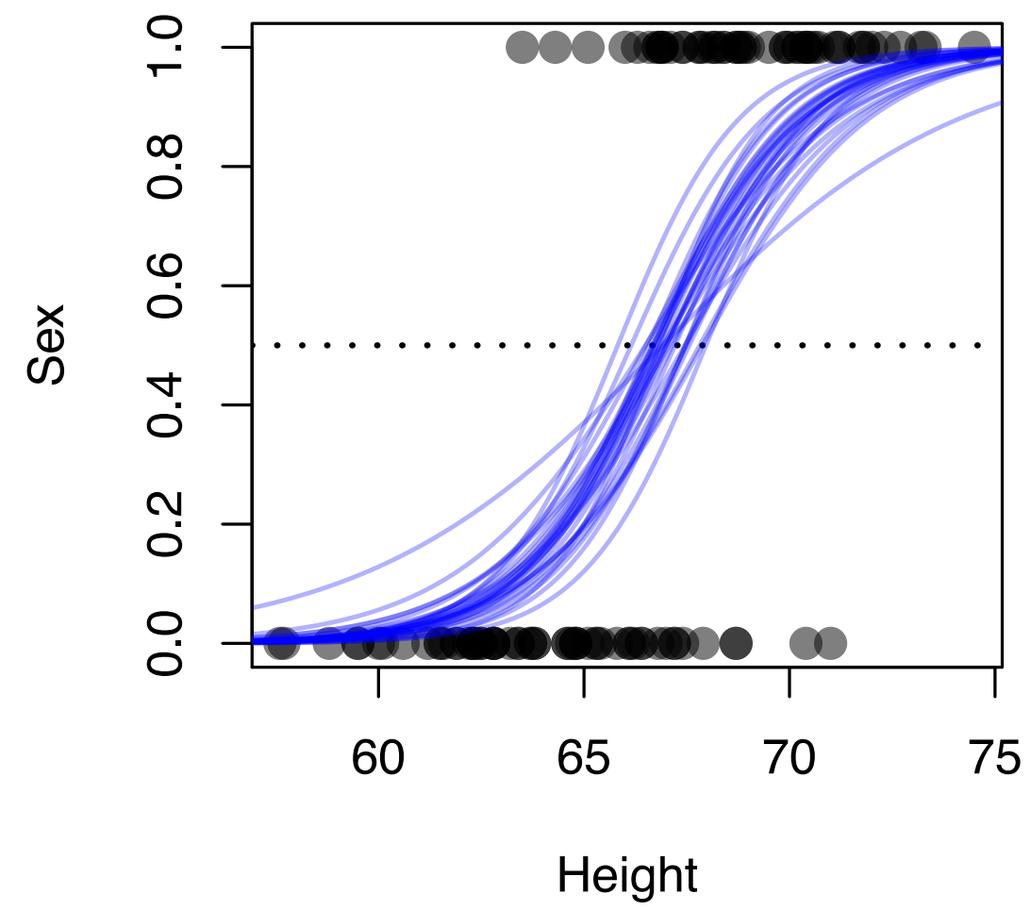
- Draw lines

```
for (i in xNew) {
    lines(xComb , 1 / (1 + exp(-(b0[i] + (b1[i] * xComb) + (b2[i] *
        mean(weight)))), lwd = 1.5, col = rgb(0, 0, 1, 0.3))
}
```

The β_2 coefficient times the mean weight value

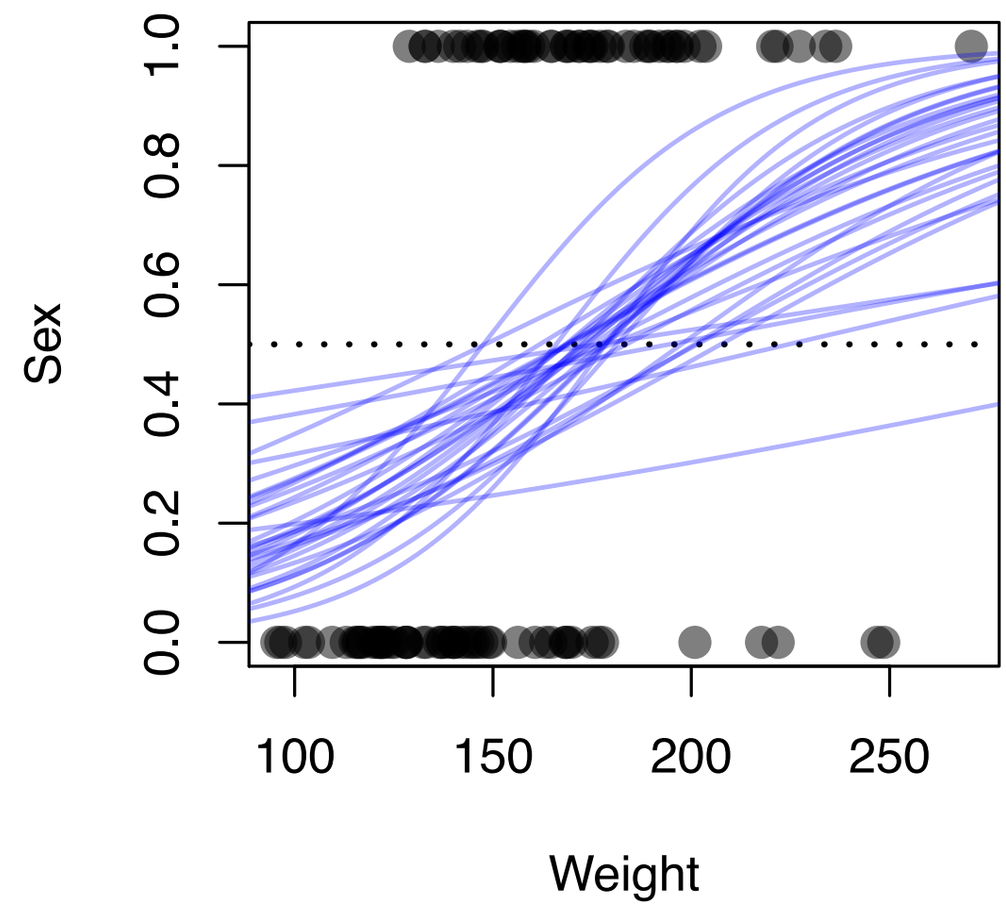
Assessing Model Fit
Single parameters (height)

- Looks good!
- Lines will get tighter if you choose higher weight
- Lines will get looser if you choose lower weight



Assessing Model Fit
Single parameters (weight)

- Much less of a good fit
- Fits with coefficient estimates (height a good predictor, weight not so much)



Posterior Predictive Check

Posterior Predictive Check

- A bit tricky due to the 1 or 0 nature of results
  - HDI bars of limited value

- Will not test predictions on full data set (110 individuals)
  - Will just choose a subset of 20

```
nPred = 20
```

Posterior Predictive Check

- Select 20 rows, evenly space throughout the data

```
newRows = seq(from = 1, to = NROW(htwt), length = nPred)
newRows = round(newRows)

newObs = y[newRows]
```

Posterior Predictive Check

- Get the posterior predictions from our results

```
chainLength = length(mcmcChains[, 1])

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

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

Posterior Predictive Check

- Get mean and HDI values

```
#--- Mean expected values  ---#
ypredMean = apply(ypred, 2, mean)

#--- Upper and lower expected 95% HDI  ---#
ypredLow  = apply(ypred, 2, quantile, probs = 0.025)
ypredHigh = apply(ypred, 2, quantile, probs = 0.975)
```

Posterior Predictive Check

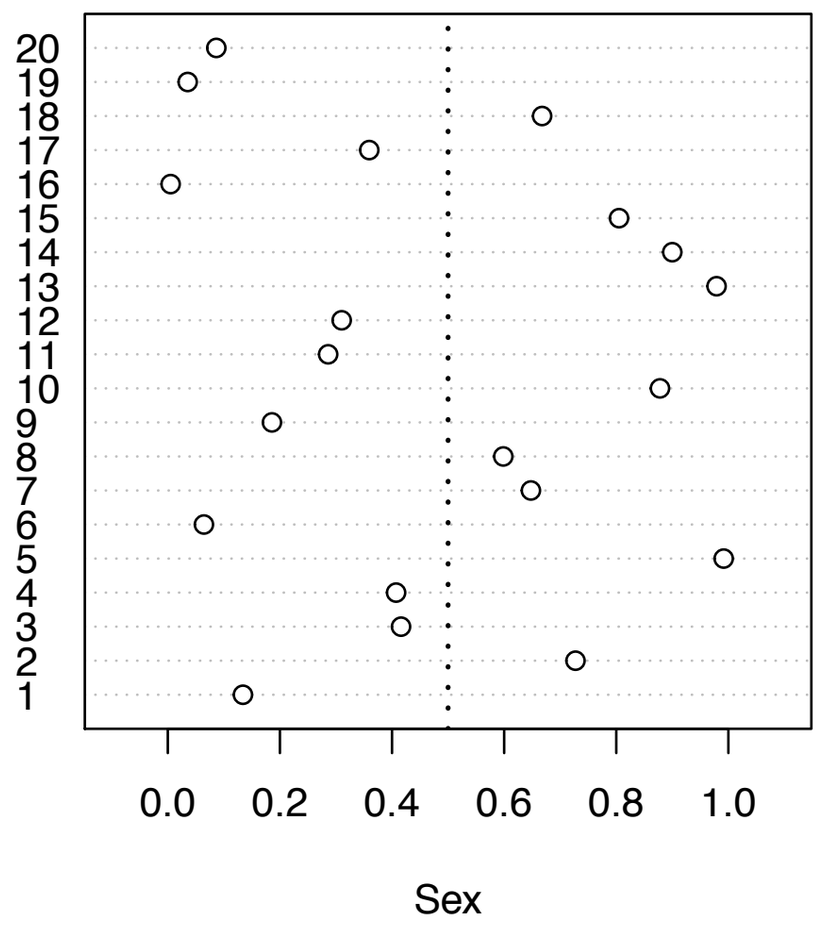
- Take just the subset of these corresponding to the 20 individuals selected from the observed data

```
subypredMean = ypredMean[newRows]
subypredLow = ypredLow[newRows]
subypredHigh = ypredHigh[newRows]
```

Posterior Predictive Check

- Plot the predicted results
  - Will just plot means, HDIs don't plot well here

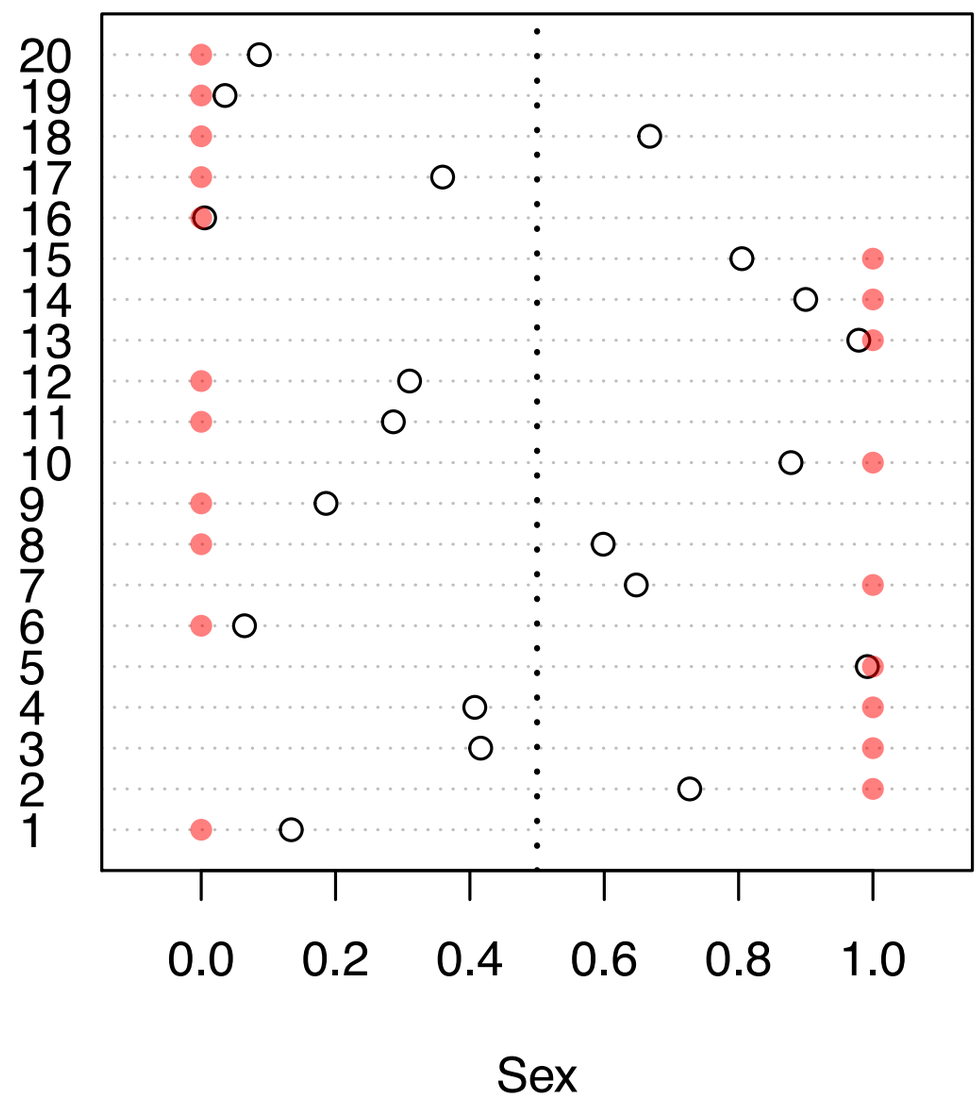
```
dotchart(subypredMean, labels = 1:nPred, xlim = c(-0.1, 1.1), xlab = “Sex")

abline(v = 0.5, lty = "dotted", lwd = 2)
```



Posterior Predictive Check

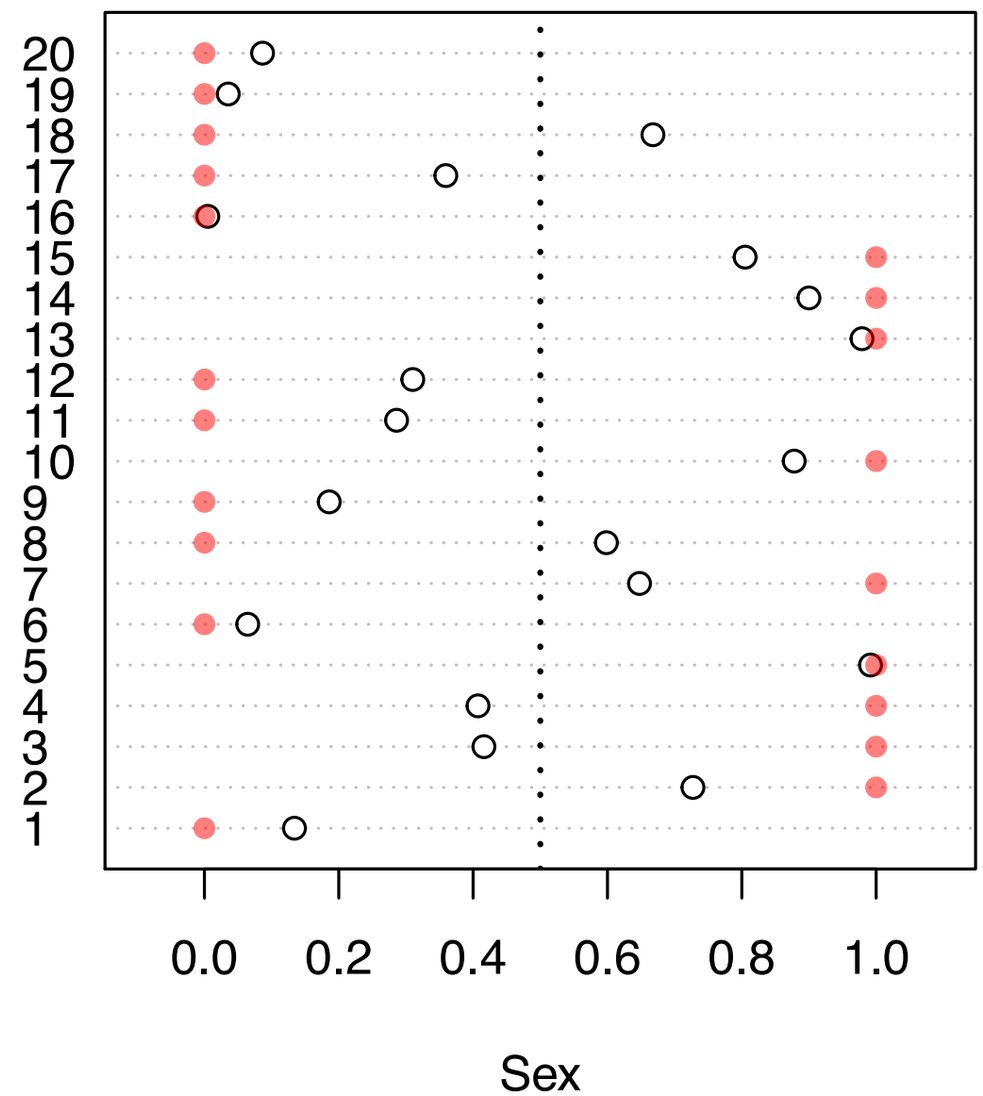
- Add the observed data

```
points(x = newObs, y = 1:nPred, pch = 16, col = rgb(1, 0, 0, 0.5))
```



Posterior Predictive Check

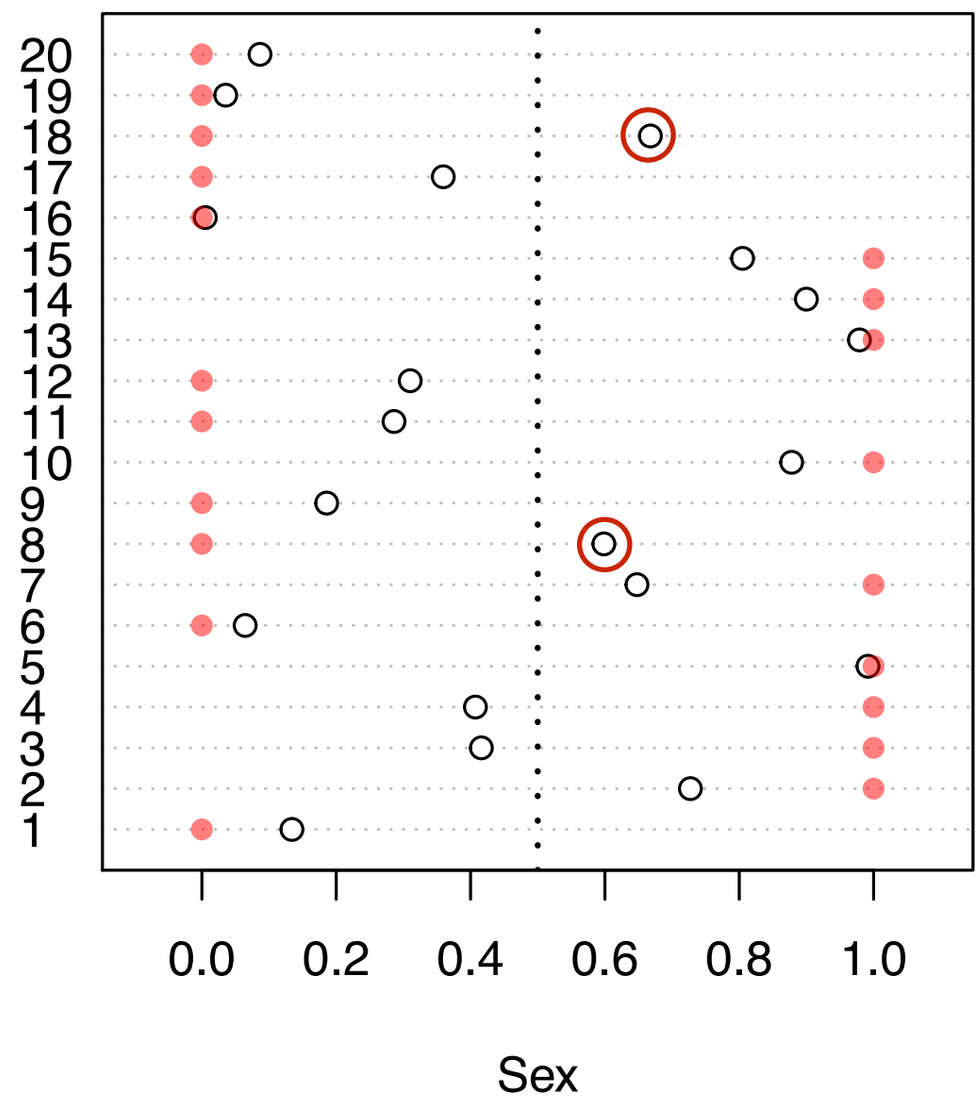
- Not great
- Not too surprising (some important parameters seem to be missing)



Posterior Predictive Check

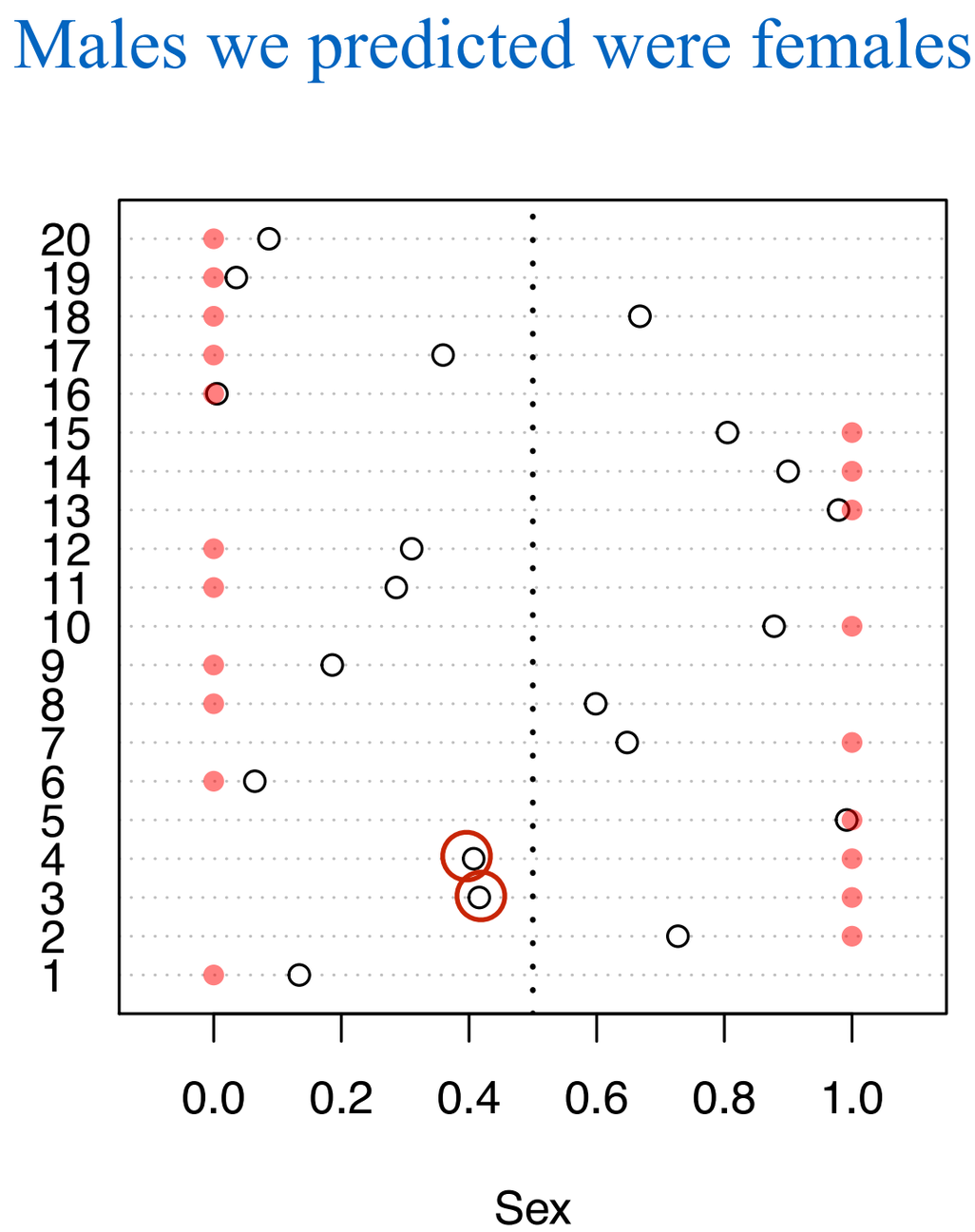
- Not great
- Not too surprising (some important parameters seem to be missing)

Females we predicted were males



Posterior Predictive Check

- Not great
- Not too surprising (some important parameters seem to be missing)



Questions?