Multiple Regression:
Mixed Predictor Types

Tim Frasier

Multiple Regression
Mixed predictor types

- Not really any different than single-parameter models... just more parameters
  - Basic structure and logic are all the same

Load Libraries

```
library(ggplot2)
library(rstan)
options(mc.cores = parallel::detectCores())
source("plotPost.R")
```

Load Libraries

```
library(ggplot2)
library(rstan)
options(mc.cores = parallel::detectCores())
source("plotPost.R")
```

How to get Stan to run in parallel (should always use)

The Data

Data

- Same milk data as before

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

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

Data

Our predicted variable: percentage of the brain that is neocortex

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

Data

`clade` will be a nominal predictor variable

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

Data

Milk characteristics will be some of our metric predictor variables

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

Data

`mass` will be another metric predicted variable

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

Data

Lots of missing data, will need to get rid of these.

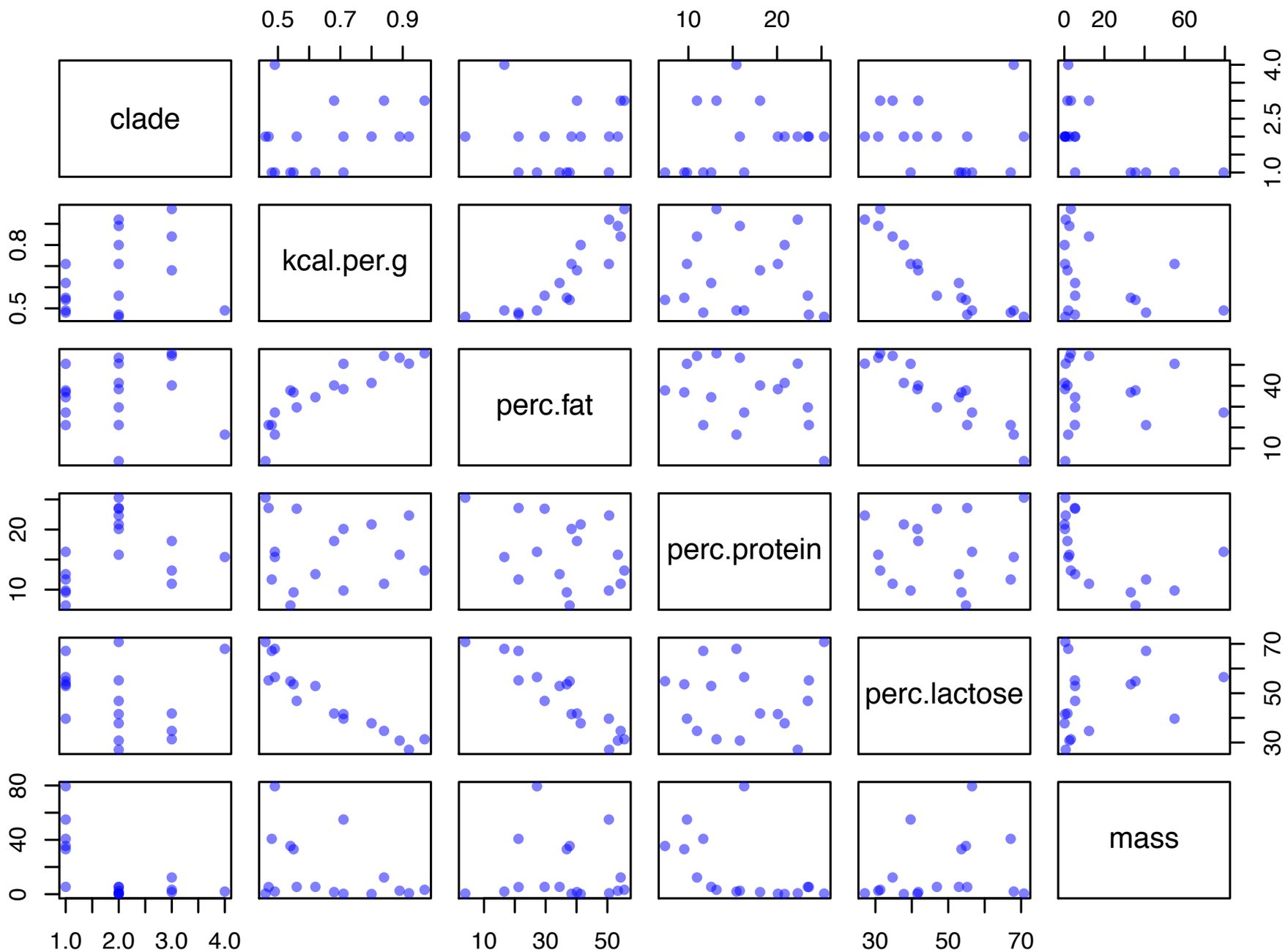
	clade	species	kcal.per.g	perc.fat	perc.protein	perc.lactose	mass	neocortex.perc
1	Strepsirrhine	Eulemur fulvus	0.49	16.60	15.42	67.98	1.95	55.16
2	Strepsirrhine	E macaco	0.51	19.27	16.91	63.82	2.09	NA
3	Strepsirrhine	E mongoz	0.46	14.11	16.85	69.04	2.51	NA
4	Strepsirrhine	E rubriventer	0.48	14.91	13.18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

Data

```
milkFull = milk[complete.cases(milk), ]
```

Data

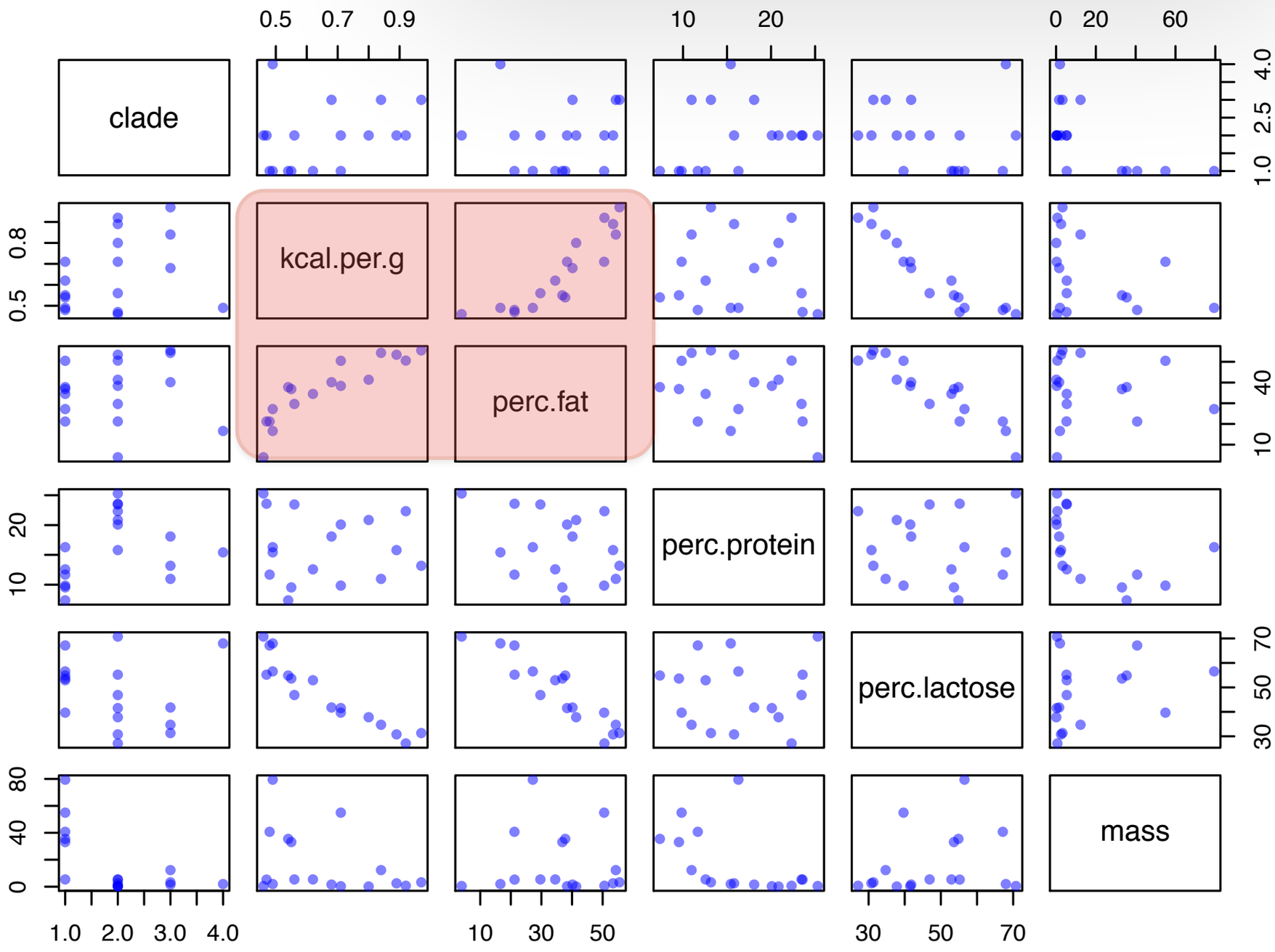
```
pairs(milkFull[, c(1, 3:7)], pch = 16, col = rgb(0, 0, 1, 0.5))
```



Data

```
pairs(milkFull[, c(1, 3:7)],
```

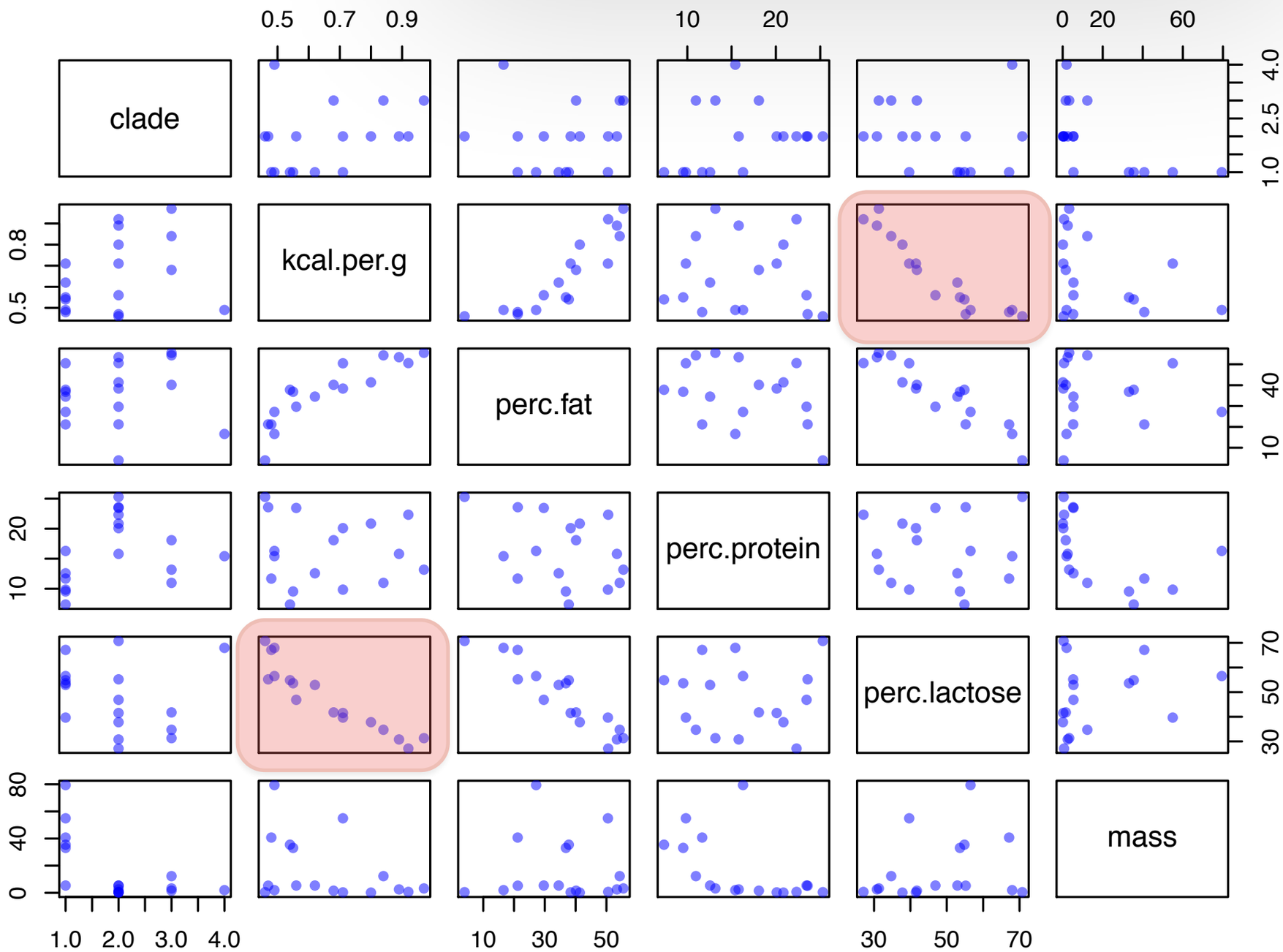
Positive correlation between `kcal.per.g` and `perc.fat`



Data

```
pairs(milkFull[, c(1, 3:7)],
```

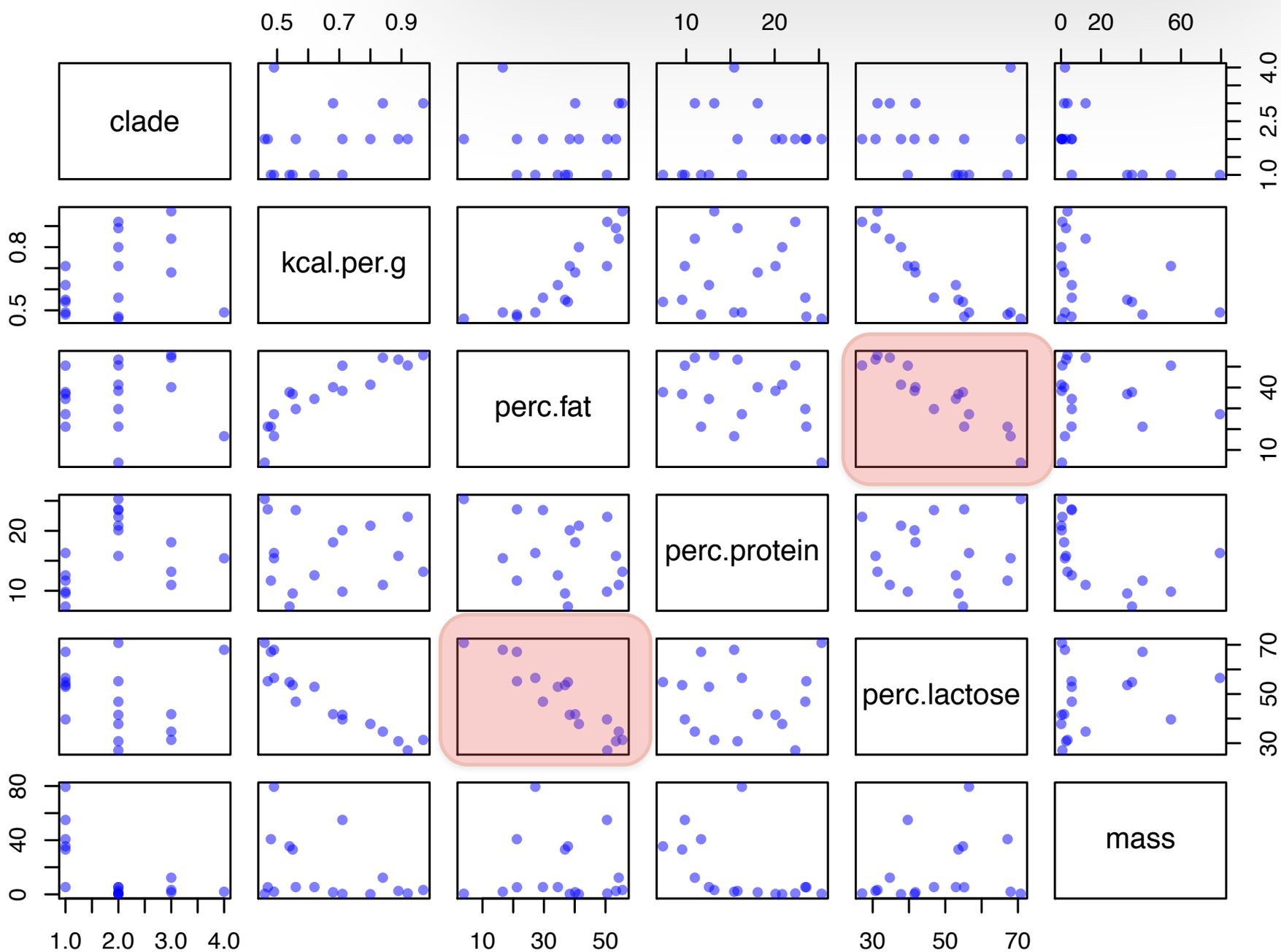
Negative correlation between `kcal.per.g` and `perc.lactose`



Data

```
pairs(milkFull[, c(1, 3:7)],
```

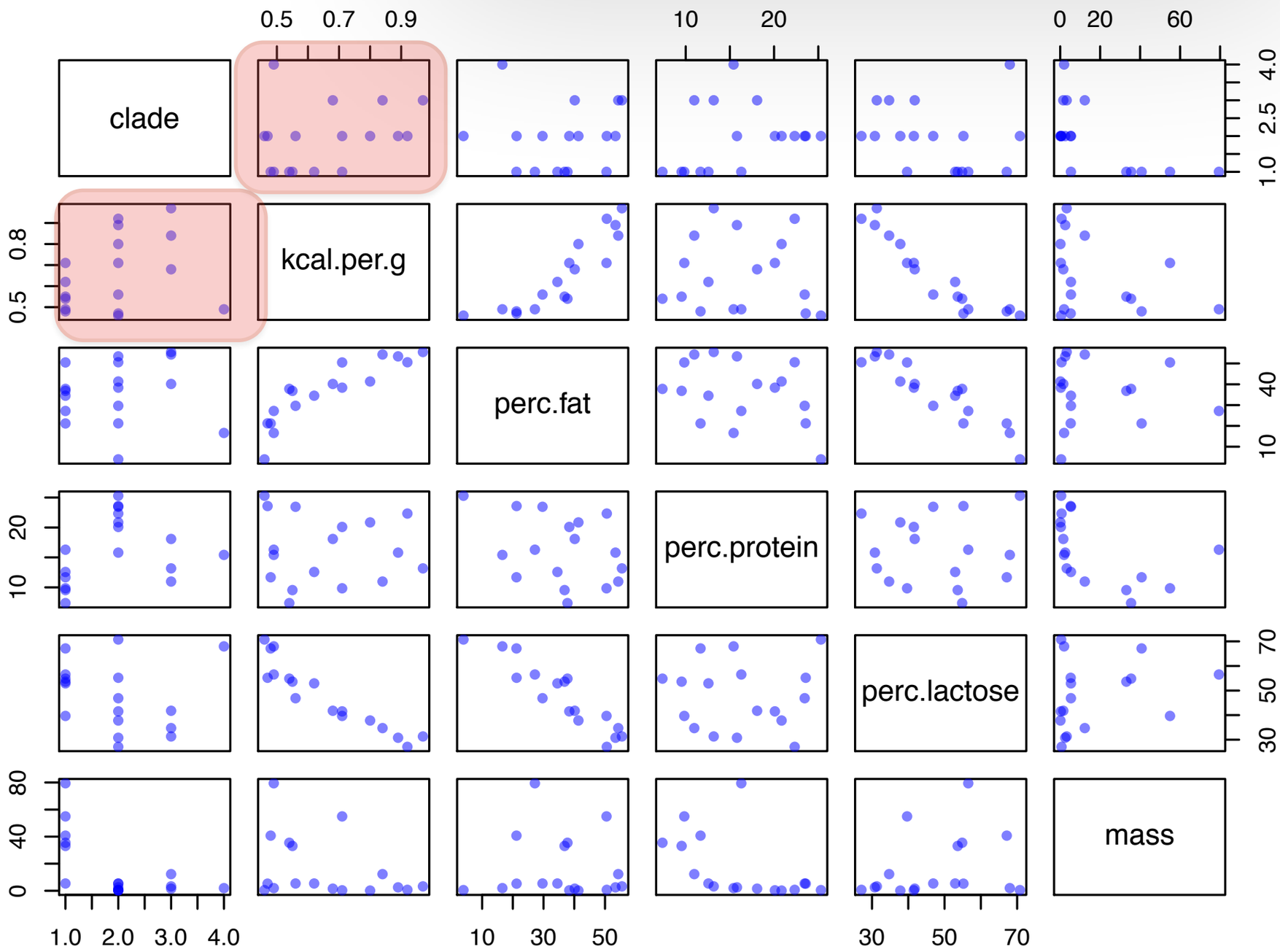
Negative correlation between `perc.fat` and `perc.lactose`



Data

```
pairs(milkFull[, c(1, 3:7)],
```

Positive correlation between `clade` and `kcal.per.g`



Data

- Check correlations

```
cor(milkFull[, 3:7])
```

Data

- Check correlations

```
cor(milkFull[, 3:7])
```

Can only calculate correlation among metric variables

Data

- Check correlations

```
cor(milkFull[, 3:7])

              kcal.per.g    perc.fat perc.protein perc.lactose        mass
kcal.per.g    1.00000000  0.88227809 -0.057561572 -0.931304232 -0.35963028
perc.fat      0.88227809  1.00000000 -0.382816282 -0.924695458 -0.04645955
perc.protein -0.05756157 -0.38281628  1.000000000  0.002281431 -0.51182926
perc.lactose -0.93130423 -0.92469546  0.002281431  1.000000000  0.26121492
mass         -0.35963028 -0.04645955 -0.511829260  0.261214918  1.00000000
```

Data

- Check correlations

```
cor(milkFull[, 3:7])

               kcal.per.g    perc.fat perc.protein perc.lactose        mass
kcal.per.g     1.00000000  0.88227809 -0.057561572 -0.931304232 -0.35963028
perc.fat       0.88227809  1.00000000 -0.382816282 -0.924695458 -0.04645955
perc.protein  -0.05756157 -0.38281628  1.000000000  0.002281431 -0.51182926
perc.lactose  -0.93130423 -0.92469546  0.002281431  1.000000000  0.26121492
mass          -0.35963028 -0.04645955 -0.511829260  0.261214918  1.00000000
```

`kcal.per.g` and `perc.fat` highly correlated

Data

- Check correlations

```
cor(milkFull[, 3:7])

               kcal.per.g    perc.fat perc.protein perc.lactose        mass
kcal.per.g     1.00000000  0.88227809 -0.057561572 -0.931304232 -0.35963028
perc.fat       0.88227809  1.00000000 -0.382816282 -0.924695458 -0.04645955
perc.protein  -0.05756157 -0.38281628  1.000000000  0.002281431 -0.51182926
perc.lactose  -0.93130423 -0.92469546  0.002281431  1.000000000  0.26121492
mass          -0.35963028 -0.04645955 -0.511829260  0.261214918  1.00000000
```

`perc.lactose` highly correlated with `kcal.per.g` and `perc.fat`

Data

- Will run five models:
  1. Full model
  2. Leaving out `perc.lactose`
  3. Leaving out `perc.lactose` and `kcal.per.g`
  4. Leaving out `perc.lactose` and `perc.fat`
  5. Leaving out `kcal.per.g` and `perc.fat` (but not perc. lactose)

Organize the Data

```
#--- Neocortex ---#
y = milkFull$neocortex.perc
yMean = mean(y)
ySD = sd(y)
zy = (y - yMean) / ySD
N = length(y)
```

```r
#--- kcal.per.g ---#
x1 = milkFull$kcal.per.g
x1Mean = mean(x1)
x1SD = sd(x1)
zx1 = (x1 - x1Mean) / x1SD

#--- perc.fat ---#
x2 = milkFull$perc.fat
x2Mean = mean(x2)
x2SD = sd(x2)
zx2 = (x2 - x2Mean) / x2SD

#--- perc.protein ---#
x3 = milkFull$perc.protein
x3Mean = mean(x3)
x3SD = sd(x3)
zx3 = (x3 - x3Mean) / x3SD

#--- perc.lactose ---#
x4 = milkFull$perc.lactose
x4Mean = mean(x4)
x4SD = sd(x4)
zx4 = (x4 - x4Mean) / x4SD

#--- mass ---#
x5 = milkFull$mass
x5Mean = mean(x5)
x5SD = sd(x5)
zx5 = (x5 - x5Mean) / x5SD
```

Organize the Data

```
#--- clade ---#
x6 = as.numeric(milkFull$clade)
x6Names = levels(milkFull$clade)
nx6Levels = length(unique(milkFull$clade))
```

Prepare the Data for Stan

```
dataList = list(
  y = zy,
  x1 = zx1,
  x2 = zx2,
  x3 = zx3,
  x4 = zx4,
  x5 = zx5,
  x6 = x6,
  N = N,
  nx6Levels = nx6Levels
)
```

Build the Model

$$y = normal(mu, sigma)$$

$$mu = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 [\mathrm{x}_{[6]}]$$

Build the Model

- The **data** block

```
modelstring = "
  data {
    int N;            // Sample size
    int nx6Levels;    // Number of clade types (levels in our nominal variable)
    vector[N] y;      // Vector of neocortex percentages
    vector[N] x1;     // Vector of kcal.per.g data
    vector[N] x2;     // Vector of perc.fat data
    vector[N] x3;     // Vector of perc.protein data
    vector[N] x4;     // Vector of perc.lactose data
    vector[N] x5;     // Vector of mass data
    int x6[N];        // Vector of indicators of which clade each sample is in
  }
```

Build the Model

$$y = normal(mu, sigma)$$

$$mu = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 [\mathrm{x}_{[6]}]$$

- The **parameters** block

```
parameters {
  real b0;
  real b1;                    // Coefficient for effect of kcal.per.g
  real b2;                      // Coefficient for effect of perc.fat
  real b3;                      // Coefficient for effect of perc.protein
  real b4;                      // Coefficient for effect of perc.lactose
  real b5;                      // Coefficient for effect of mass
  real b6[nx6Levels];           // Coefficients for effects of being in each clade
  real<lower=0> sigma;          // Coefficients for overall sd
  real cladeMean;               // Mean effect across all clades
  real<lower=0> cladeMeanSD;    // sd for mean effect across all clades
}
```

Build the Model

$$y = normal(mu, sigma)$$

$$mu = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6[x_{[6]}]$$

- The **parameters** block

Will make the `clade` part hierarchical, and these will be the hyperprior parameters

```
parameters {
  real b0;
  real b1;                                                    g
  real b2;                                                    t
  real b3;                                                    otein
  real b4;                    // Coefficient for effect of perc.lactose
  real b5;                    // Coefficient for effect of mass
  real b6[nx6Levels];         // Coefficients for effects of being in each clade
  real<lower=0> sigma;        // Coefficients for overall sd
  real cladeMean;             // Mean effect across all clades
  real<lower=0> cladeMeanSD;  // sd for mean effect across all clades
}
```

Build the Model

- The **model** block

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

    // Likelihood
    for (i in 1:N) {
      mu[i] = b0 + b1*x1[i] + b2*x2[i] + b3*x3[i] + b4*x4[i] + b5*x5[i] + b6[x6[i]];
      y[i] ~ normal(mu[i], sigma);
    }

    // Priors
    b0 ~ normal(0, 1);
    b1 ~ normal(0, 1);
    b2 ~ normal(0, 1);
    b3 ~ normal(0, 1);
    b4 ~ normal(0, 1);
    b5 ~ normal(0, 1);
    sigma ~ cauchy(0, 1);

    for (j in 1:nx6Levels) {
      b6[j] ~ normal(cladeMean, cladeMeanSD);
    }

    // Hyperpriors
    cladeMean   ~ normal(0, 1);
    cladeMeanSD ~ normal(1, 1);
  }
```

Build the Model

- The **generated quantities** block

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

  // WAIC Variable Definitions
  vector[N] log_lik;
  vector[N] mu_waic;

  // For Posterior Predictive Calculations
  for (i in 1:N) {
    mu_pred[i] = b0 + b1*x1[i] + b2*x2[i] + b3*x3[i] + b4*x4[i] + b5*x5[i] + b6[x6[i]];
    y_pred[i] = normal_rng(mu_pred[i], sigma);
  }

  // For WAIC Calculatons
  for (i in 1:N) {
    mu_waic[i] = b0 + b1*x1[i] + b2*x2[i] + b3*x3[i] + b4*x4[i] + b5*x5[i] + b6[x6[i]];
    log_lik[i] = normal_lpdf(y[i] | mu_waic[i], sigma);
  }
}
"
writeLines(modelstring, con = “model_Full.stan”)
```

Run the Model

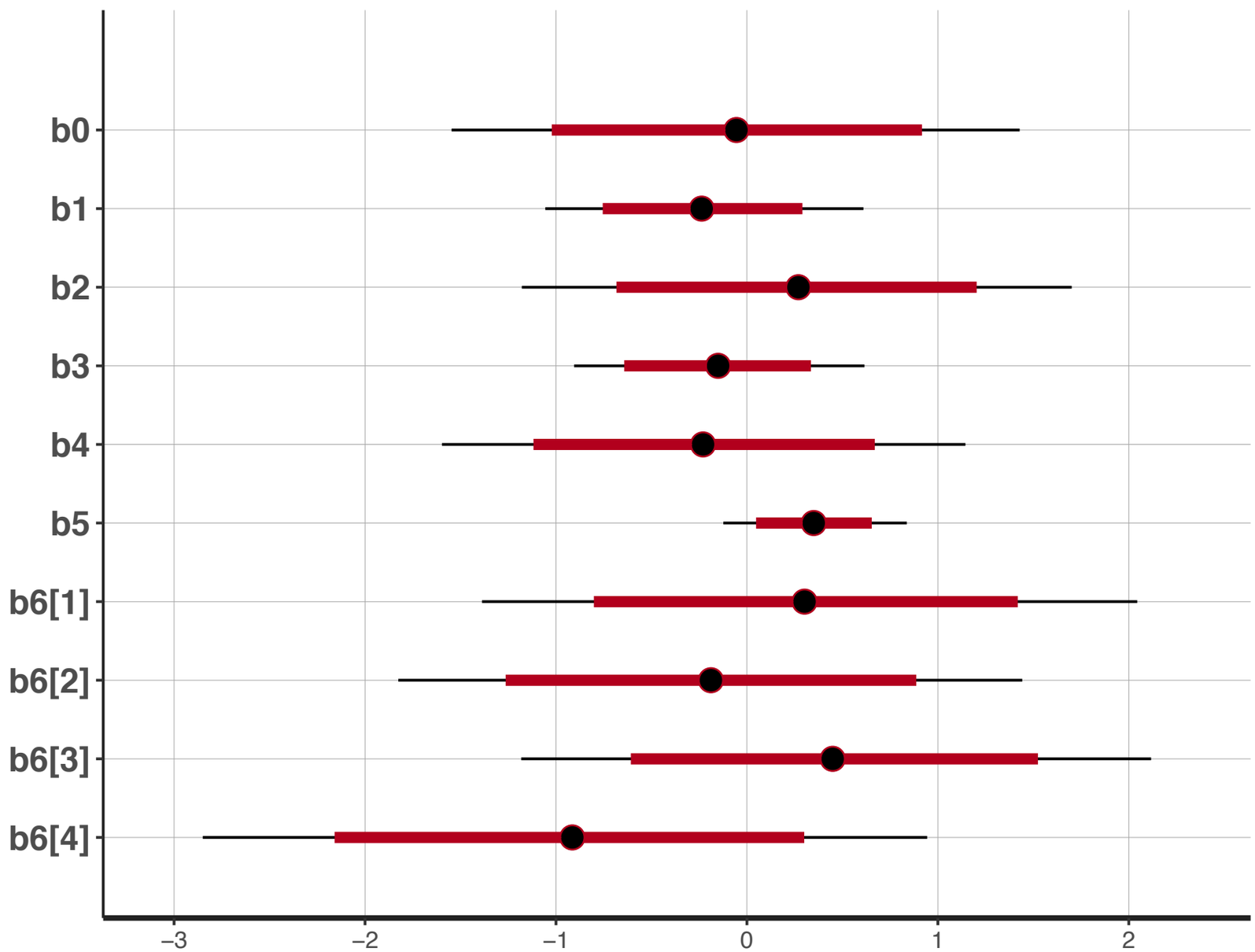
```
model_Full = stan(file = "model_Full.stan",
                  data = dataList,
                  pars = c("b0", "b1", "b2", "b3", "b4", "b5", "b6", “sigma",
                           “y_pred", "log_lik"),
                  warmup = 2000,
                  iter = 12000,
                  chains = 3)
```

Check MCMC Performance

```
print(model_Full)

Inference for Stan model: model_Full.
3 chains, each with iter=12000; warmup=2000; thin=1;
post-warmup draws per chain=10000, total post-warmup draws=30000.

             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
b0          -0.05    0.01 0.76 -1.55 -0.56 -0.05  0.46  1.43  8121    1
b1          -0.23    0.00 0.42 -1.06 -0.51 -0.24  0.04  0.61 22276    1
b2           0.27    0.01 0.74 -1.18 -0.23  0.27  0.76  1.70 14346    1
b3          -0.15    0.00 0.39 -0.91 -0.40 -0.15  0.10  0.62 16263    1
b4          -0.23    0.01 0.70 -1.60 -0.70 -0.23  0.24  1.14 14760    1
b5           0.35    0.00 0.24 -0.12  0.19  0.35  0.50  0.84 16559    1
b6[1]        0.31    0.01 0.87 -1.39 -0.29  0.30  0.89  2.04  9376    1
b6[2]       -0.19    0.01 0.84 -1.83 -0.75 -0.19  0.37  1.44  9386    1
b6[3]        0.46    0.01 0.84 -1.18 -0.11  0.45  1.02  2.12  8539    1
b6[4]       -0.92    0.01 0.96 -2.85 -1.57 -0.91 -0.26  0.94  9741    1
sigma        0.59    0.00 0.15  0.38  0.49  0.57  0.67  0.95 13871    1
y_pred[1]   -1.63    0.01 0.85 -3.25 -2.19 -1.67 -1.11  0.13 26018    1
y_pred[2]   -0.75    0.00 0.70 -2.16 -1.19 -0.75 -0.31  0.65 28985    1
...
```

Preliminary Evaluation of Parameters

```
stan_plot(model_Full, par = c("b0", "b1", "b2", "b3", "b4", "b5", "b6"))
```



How Well Does This Model Perform?

- Extract predicted values

```
mcmcChains = as.data.frame(model_Full)
chainLength = length(mcmcChains[, 1])

zypred = matrix(0, ncol = N, nrow = chainLength)
for (i in 1:N) {
  zypred[, i] = mcmcChains[, paste("y_pred[", i, "]", sep = "")]
}
```

How Well Does This Model Perform?

- Calculate mean and HDIs

```
#--- Mean expected value for record ---#
ypredMean = apply(zypred, 2, mean)

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

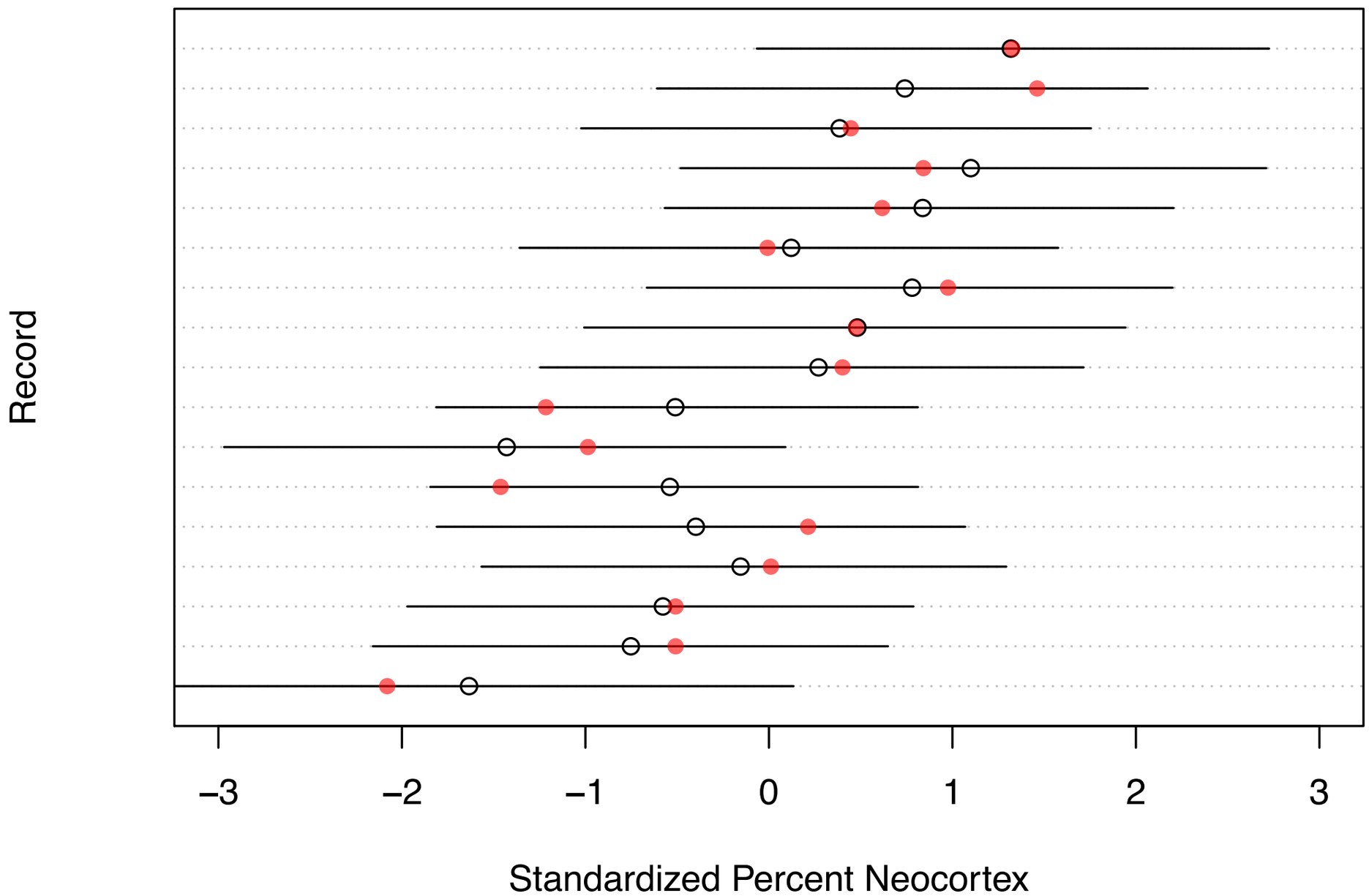
How Well Does This Model Perform?

- Plot predicted vs observed

```
#--- Plot mean predicted values ---#
record = 1:N
dotchart(ypredMean, xlim = c(-3, 3), xlab = "Standardized Percent Neocortex",
   ylab = "Record")

#--- Add HDIs ---#
segments(x0 = ypredLow, y0 = record, x1 = ypredHigh, y1 = record)

#--- Add observed values ---#
points(x = zy, y = record, pch = 16, col = rgb(1, 0, 0, 0.6))
```

How Well Does This Model Perform?

- Plot predicted vs observed



How Well Does This Model Perform?

- Get WAIC values

```
loglik_Full = extract_log_lik(model_Full)
waic_Full = waic(loglik_Full)
```

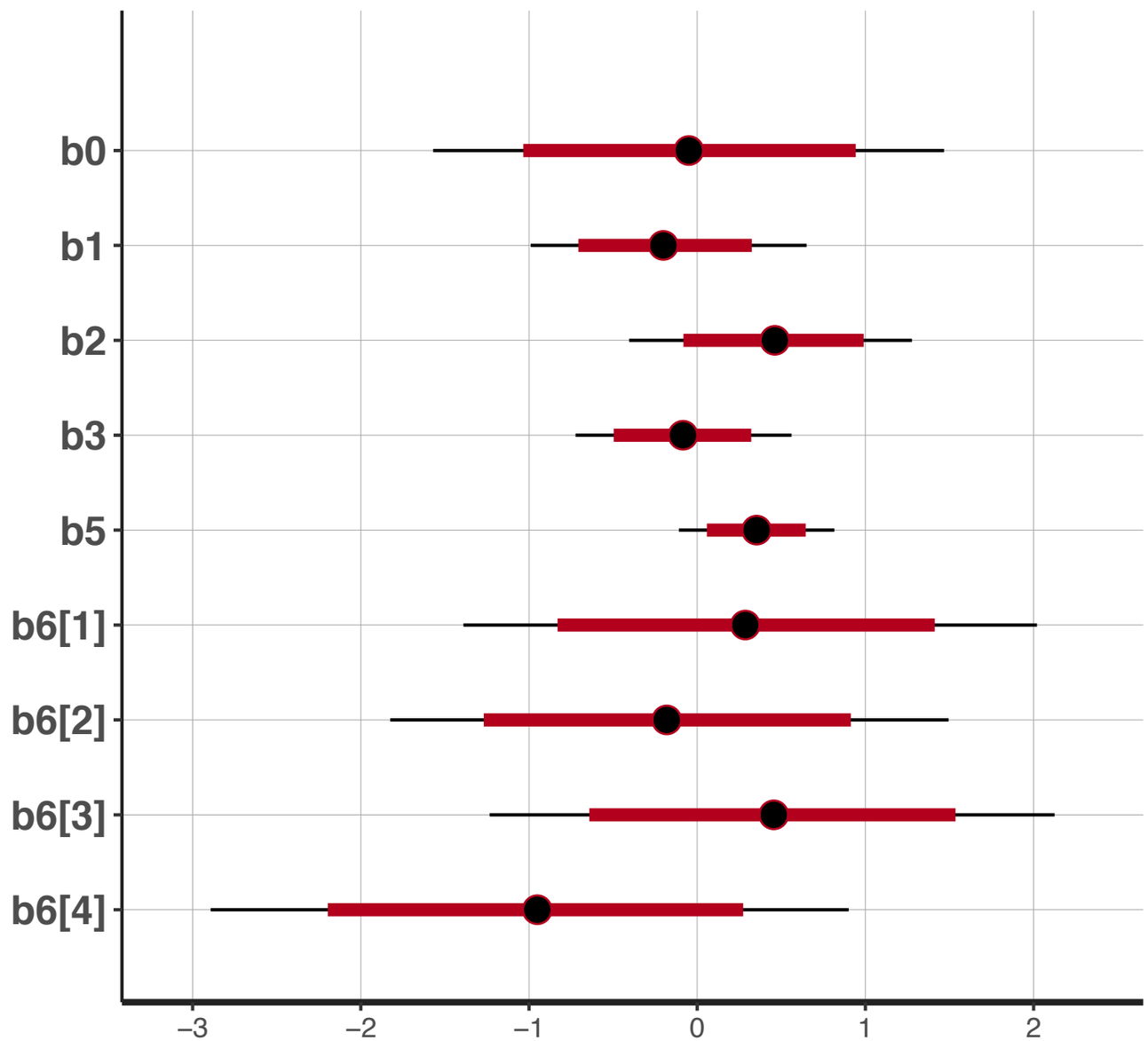
Model 2:
Removing `perc.lactose`

Check MCMC Performance

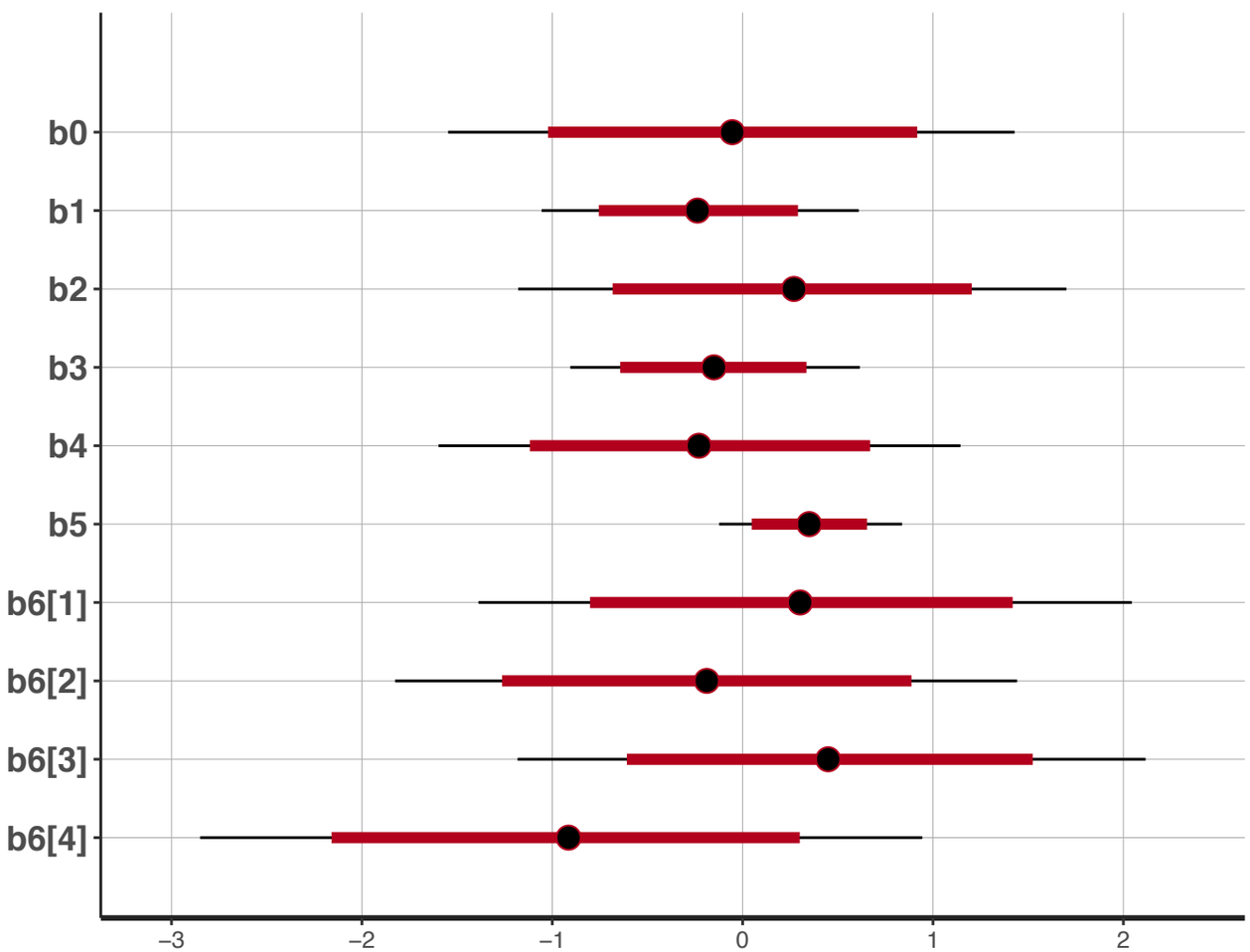
```
print(model_2)

Inference for Stan model: model_Full.
3 chains, each with iter=12000; warmup=2000; thin=1;
post-warmup draws per chain=10000, total post-warmup draws=30000.

             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
b0          -0.05    0.01 0.76 -1.55 -0.56 -0.05  0.46  1.43  8121    1
b1          -0.23    0.00 0.42 -1.06 -0.51 -0.24  0.04  0.61 22276    1
b2           0.27    0.01 0.74 -1.18 -0.23  0.27  0.76  1.70 14346    1
b3          -0.15    0.00 0.39 -0.91 -0.40 -0.15  0.10  0.62 16263    1
b4          -0.23    0.01 0.70 -1.60 -0.70 -0.23  0.24  1.14 14760    1
b5           0.35    0.00 0.24 -0.12  0.19  0.35  0.50  0.84 16559    1
b6[1]        0.31    0.01 0.87 -1.39 -0.29  0.30  0.89  2.04  9376    1
b6[2]       -0.19    0.01 0.84 -1.83 -0.75 -0.19  0.37  1.44  9386    1
b6[3]        0.46    0.01 0.84 -1.18 -0.11  0.45  1.02  2.12  8539    1
b6[4]       -0.92    0.01 0.96 -2.85 -1.57 -0.91 -0.26  0.94  9741    1
sigma        0.59    0.00 0.15  0.38  0.49  0.57  0.67  0.95 13871    1
y_pred[1]   -1.63    0.01 0.85 -3.25 -2.19 -1.67 -1.11  0.13 26018    1
y_pred[2]   -0.75    0.00 0.70 -2.16 -1.19 -0.75 -0.31  0.65 28985    1
...
```

Preliminary Evaluation of Parameters

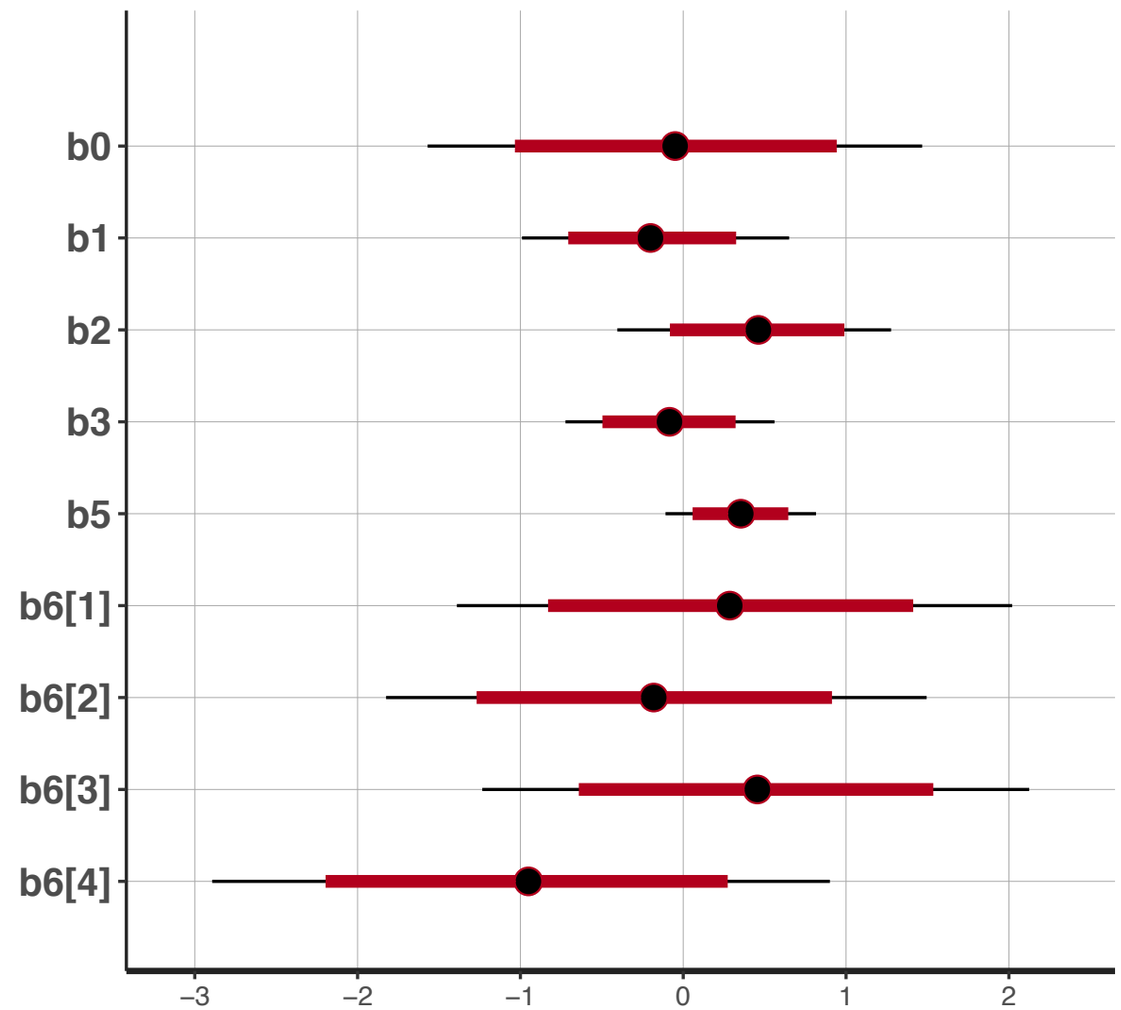
```
stan_plot(model_2, par = c("b0", "b1", "b2", "b3", "b5", "b6"))
```



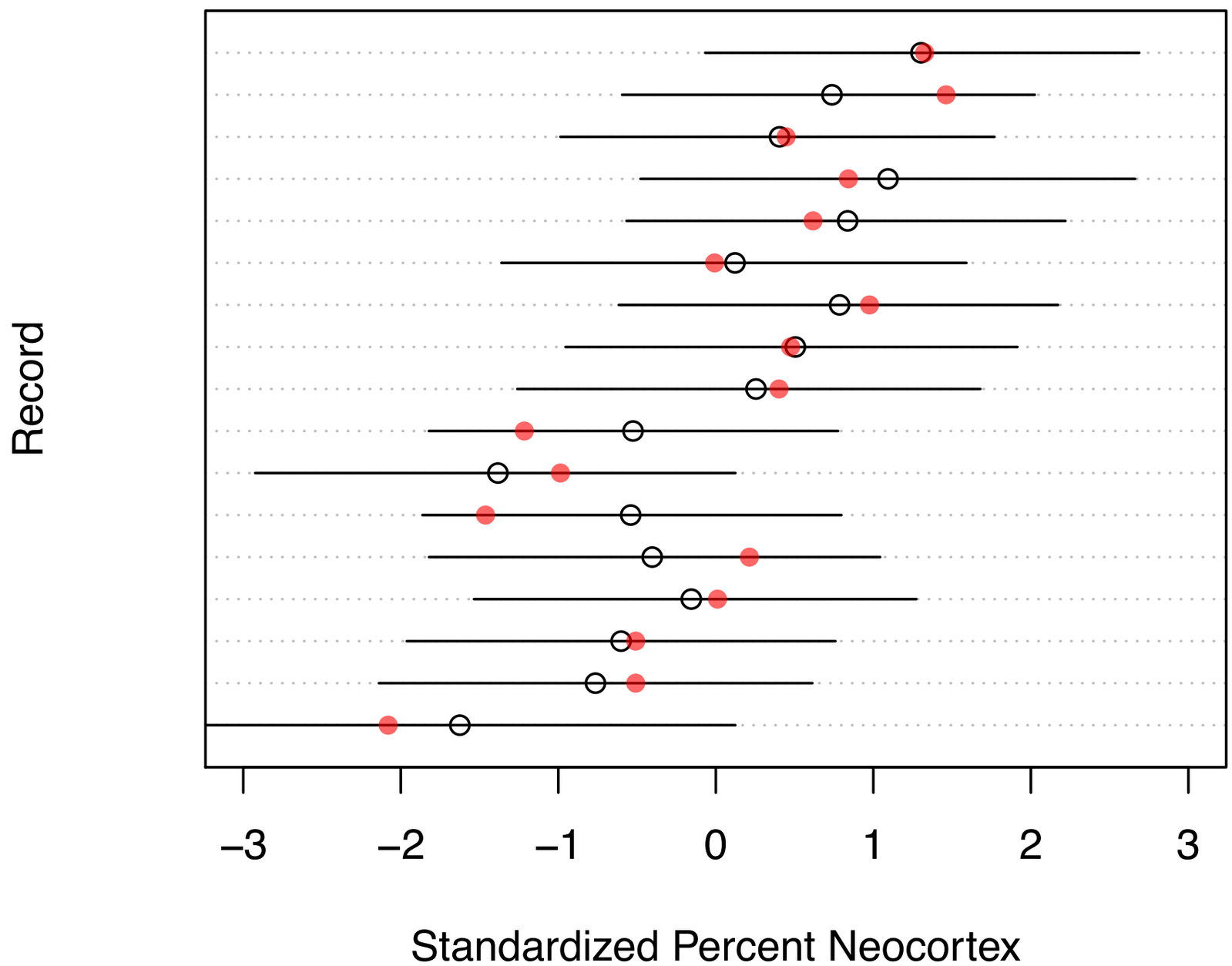
Full Model



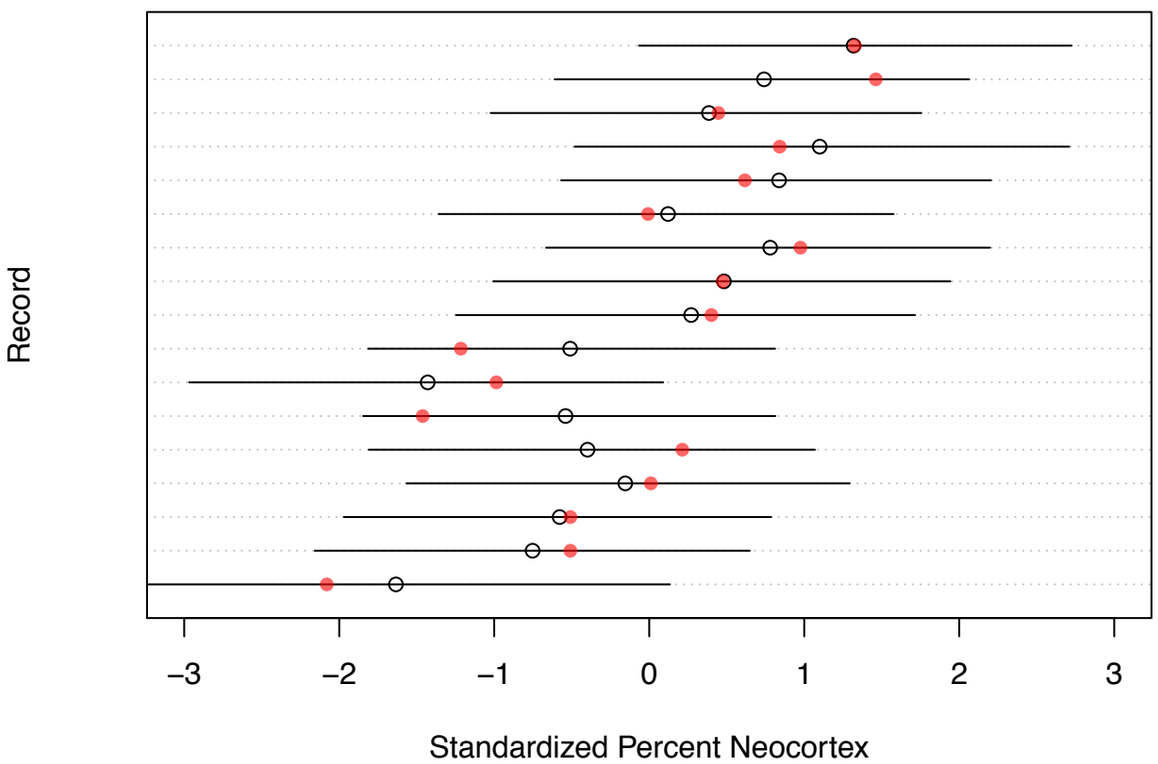
Model 2

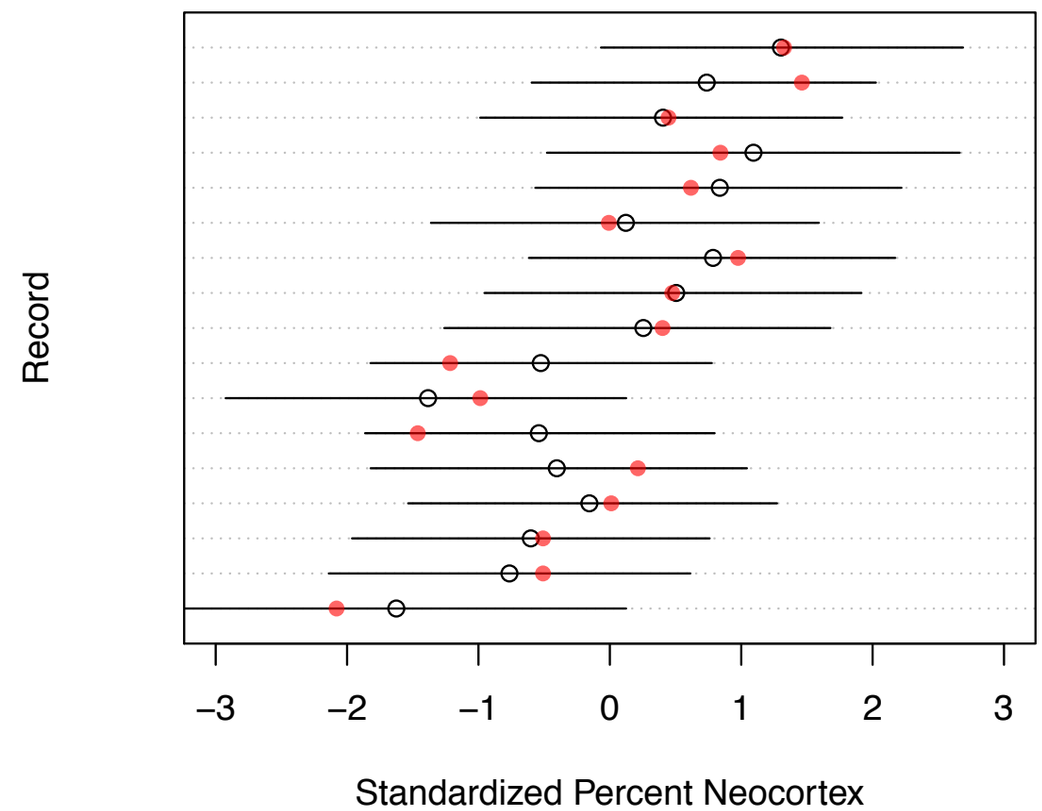


How Well Does This Model Perform?



How Well Does This Model Perform?





How Well Does This Model Perform?

- Get WAIC values

```
loglik_2 = extract_log_lik(model_2)
waic_2 = waic(loglik_2)
```

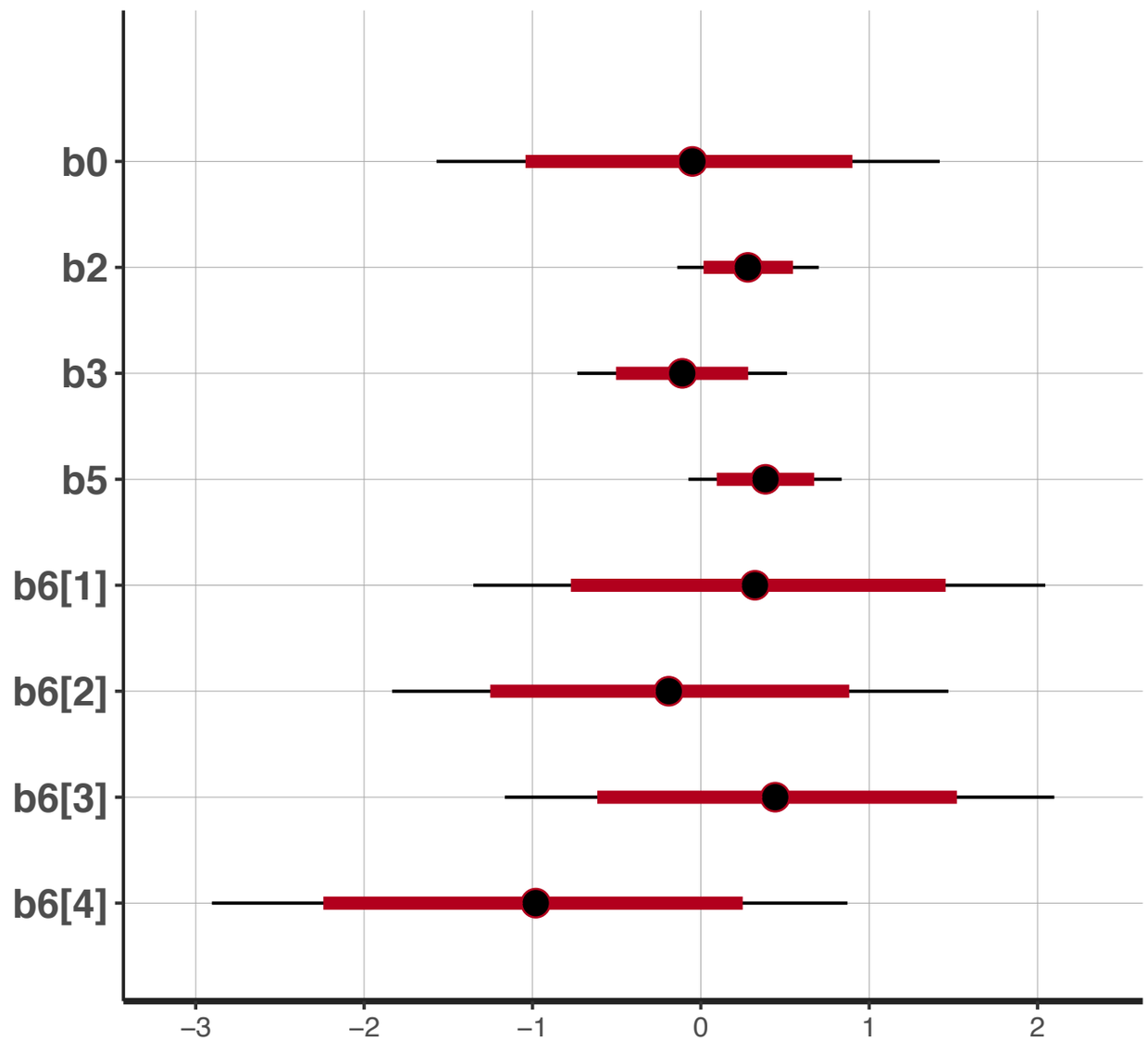
Model 3:
Removing `perc.lactose` and and `kcal.per.g`

Check MCMC Performance

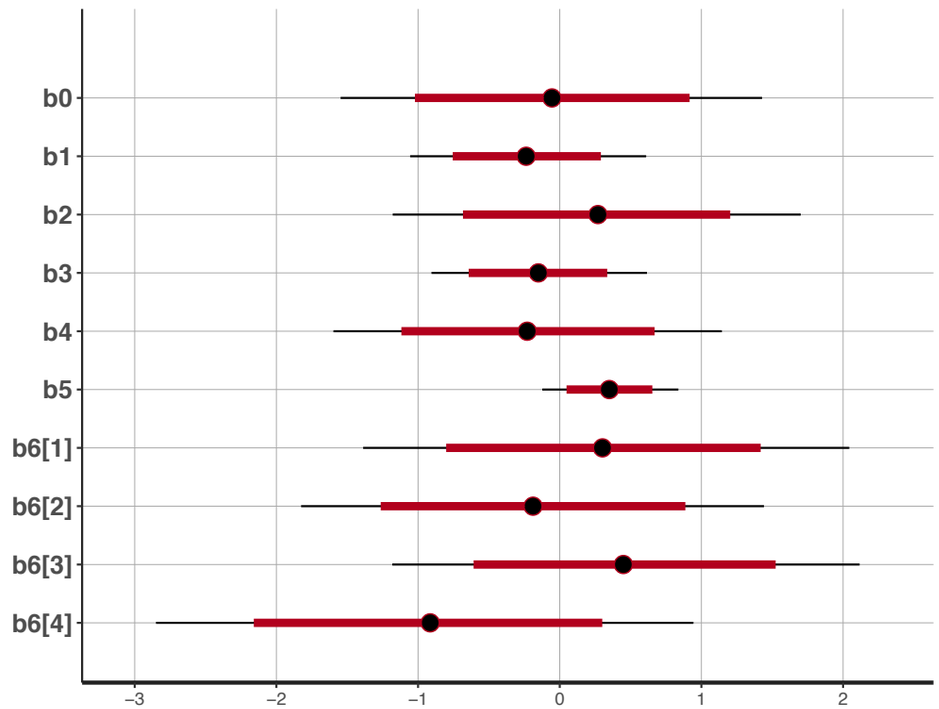
```
print(model_3)

Inference for Stan model: model_3.
3 chains, each with iter=12000; warmup=2000; thin=1;
post-warmup draws per chain=10000, total post-warmup draws=30000.

              mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
b0           -0.06    0.01 0.76 -1.57 -0.57 -0.05  0.45  1.42  6894    1
b2            0.28    0.00 0.21 -0.14  0.15  0.28  0.42  0.70 16395    1
b3           -0.11    0.00 0.31 -0.73 -0.32 -0.11  0.09  0.51 15128    1
b5            0.38    0.00 0.23 -0.07  0.23  0.38  0.53  0.84 17579    1
b6[1]         0.33    0.01 0.87 -1.35 -0.26  0.32  0.91  2.05  7651    1
b6[2]        -0.19    0.01 0.84 -1.83 -0.74 -0.19  0.37  1.47  7948    1
b6[3]         0.45    0.01 0.84 -1.16 -0.13  0.44  1.01  2.10  6991    1
b6[4]        -0.99    0.01 0.97 -2.90 -1.63 -0.98 -0.35  0.87  8215    1
sigma         0.58    0.00 0.14  0.37  0.48  0.56  0.65  0.91 12899    1
y_pred[1]    -1.64    0.01 0.84 -3.25 -2.19 -1.67 -1.12  0.07 16585    1
y_pred[2]    -0.86    0.00 0.65 -2.18 -1.27 -0.86 -0.45  0.44 29661    1
...
```

Preliminary Evaluation of Parameters

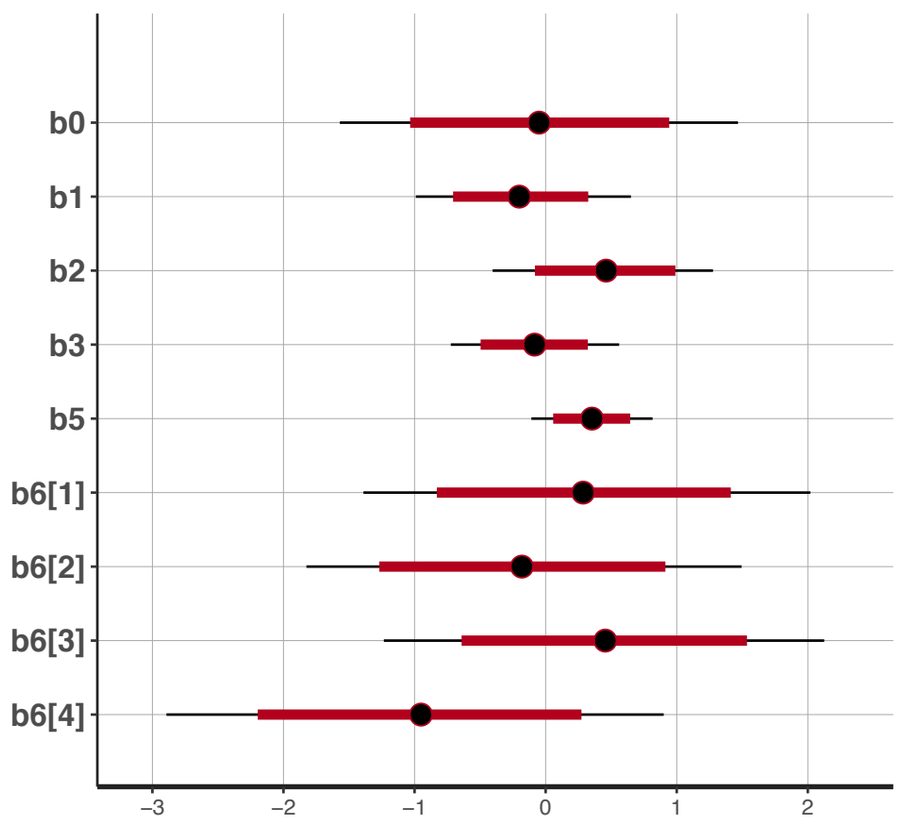
```
stan_plot(model_3, par = c(“b0", "b2", "b3", "b5", "b6"))
```



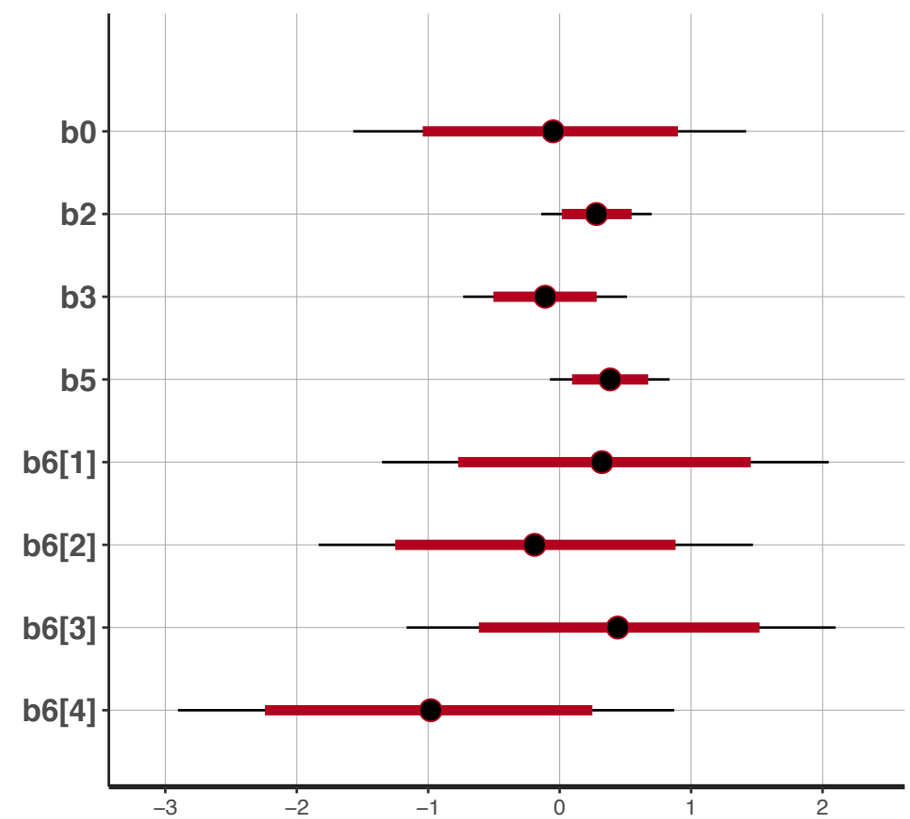
Full Model



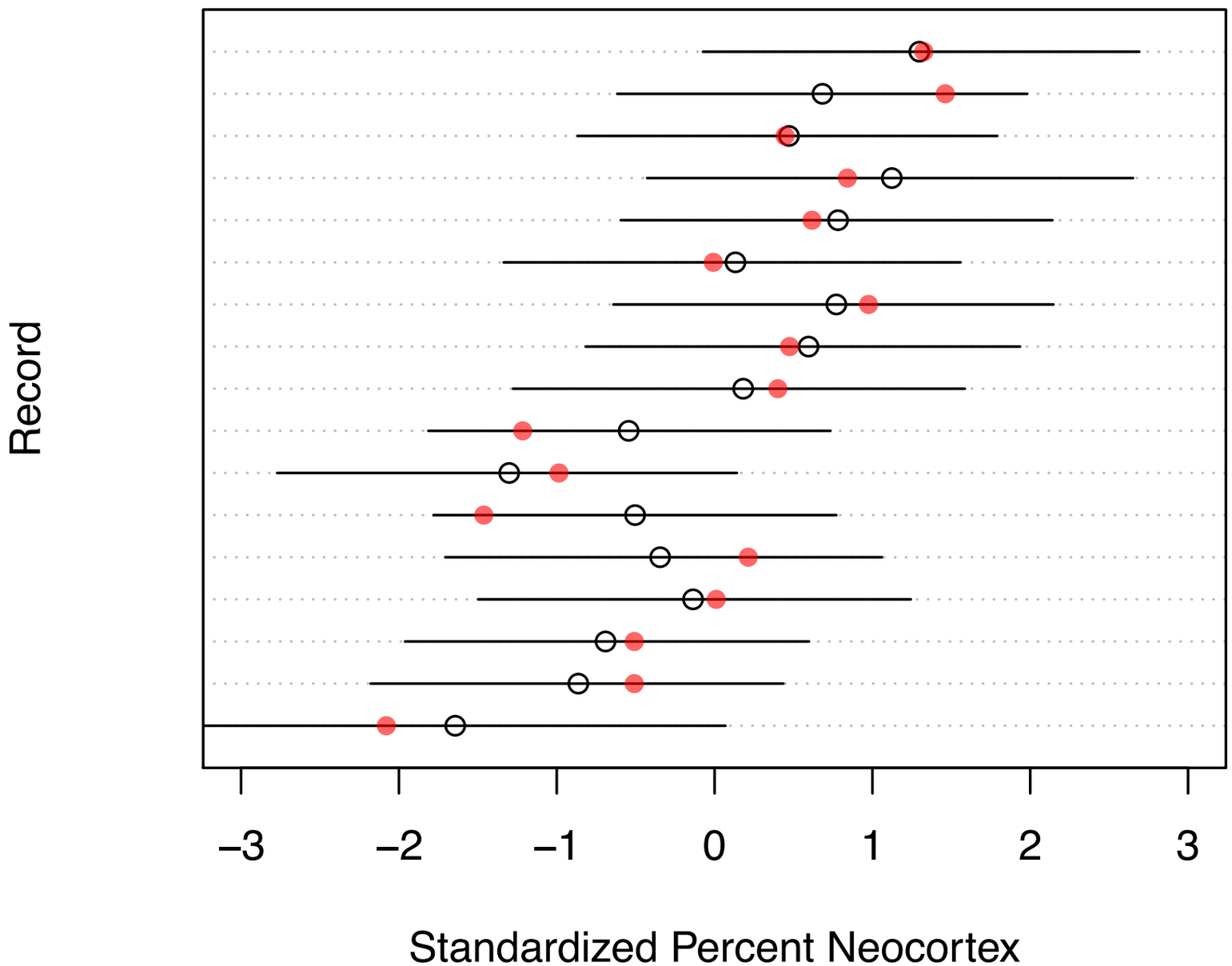
Model 2

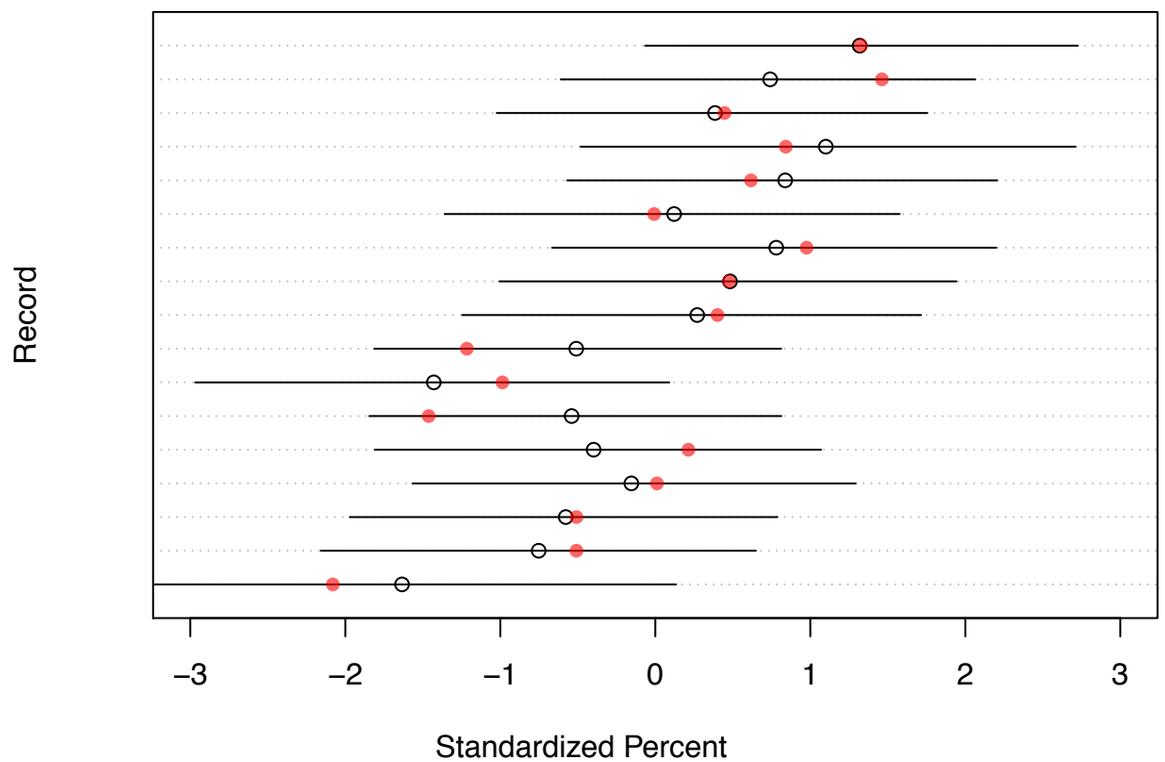


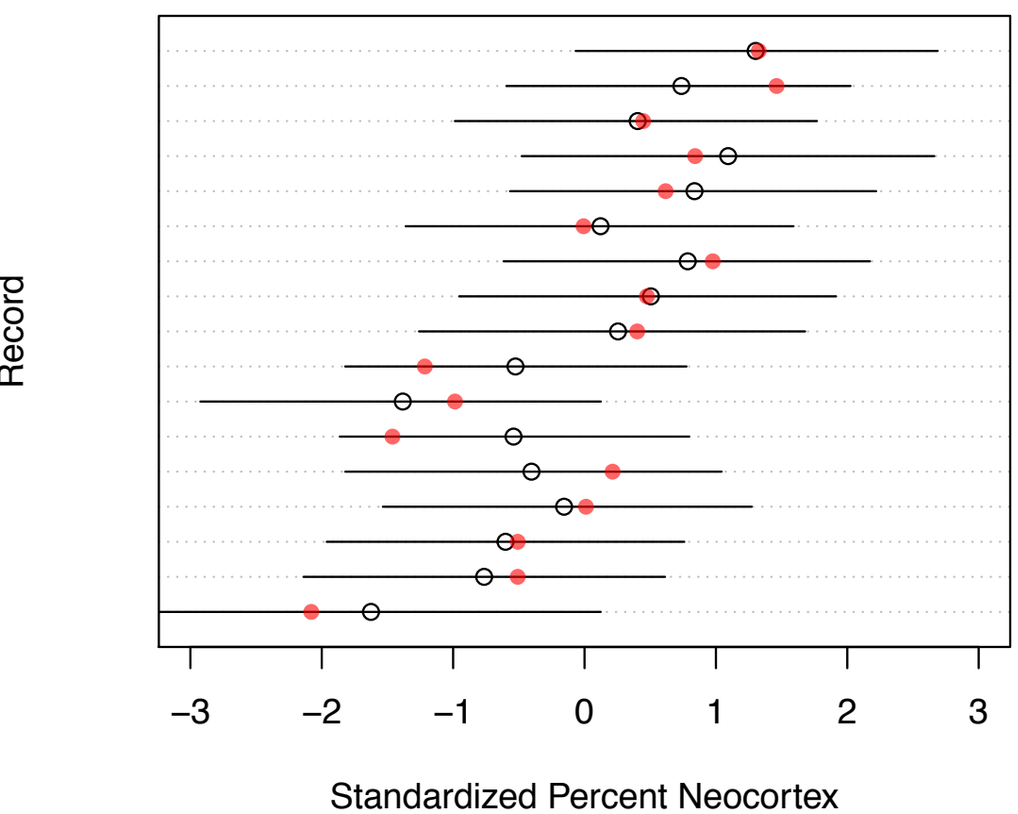
Model 3

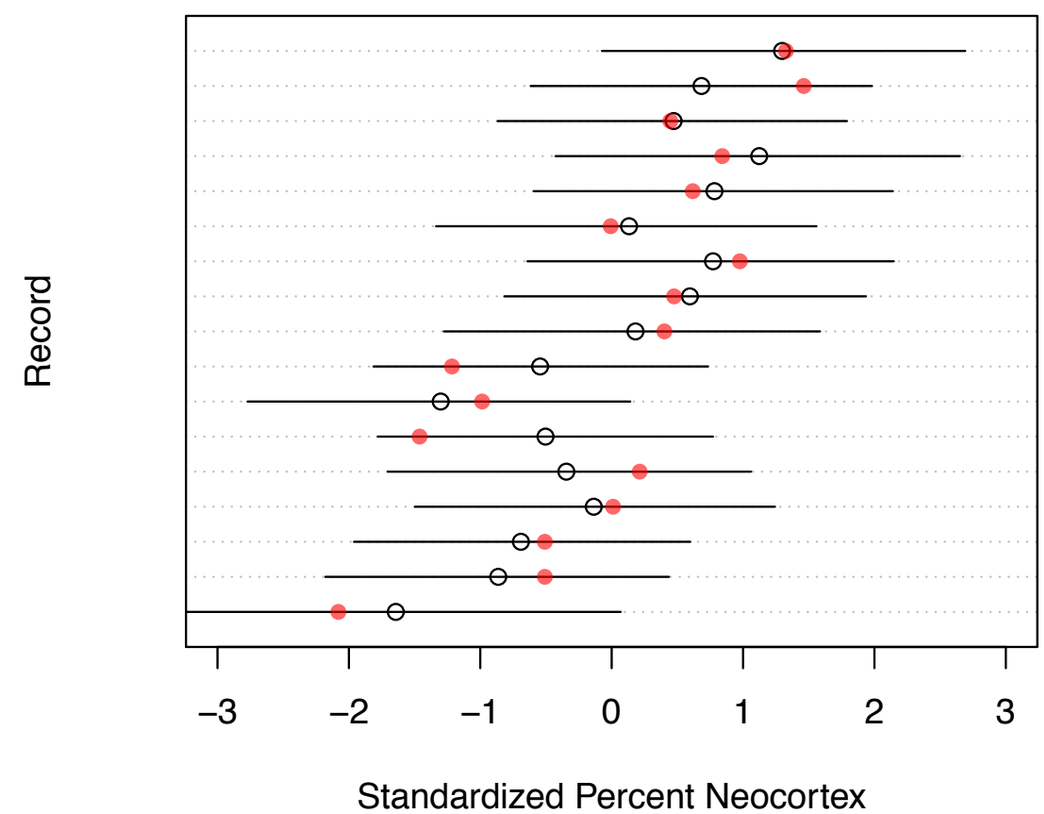


How Well Does This Model Perform?




Record
−3
−2
−1
0
1
2
3
Standardized Percent


Record
−3
−2
−1
0
1
2
3
Standardized Percent Neocortex


Record
−3
−2
−1
0
1
2
3
Standardized Percent Neocortex

How Well Does This Model Perform?

- Get WAIC values and compare

```
loglik_3 = extract_log_lik(model_3)
waic_3 = waic(loglik_3)

compare(waic_Full, waic_2, waic_3)

          elpd_diff elpd_waic se_elpd_waic p_waic se_p_waic waic  se_waic
waic_3      0.0     -17.3       2.3          5.1    1.0      34.6   4.6
waic_2     -0.6     -17.9       2.2          5.5    1.1      35.8   4.4
waic_Full  -0.6     -17.9       2.1          5.5    1.0      35.9   4.2
```

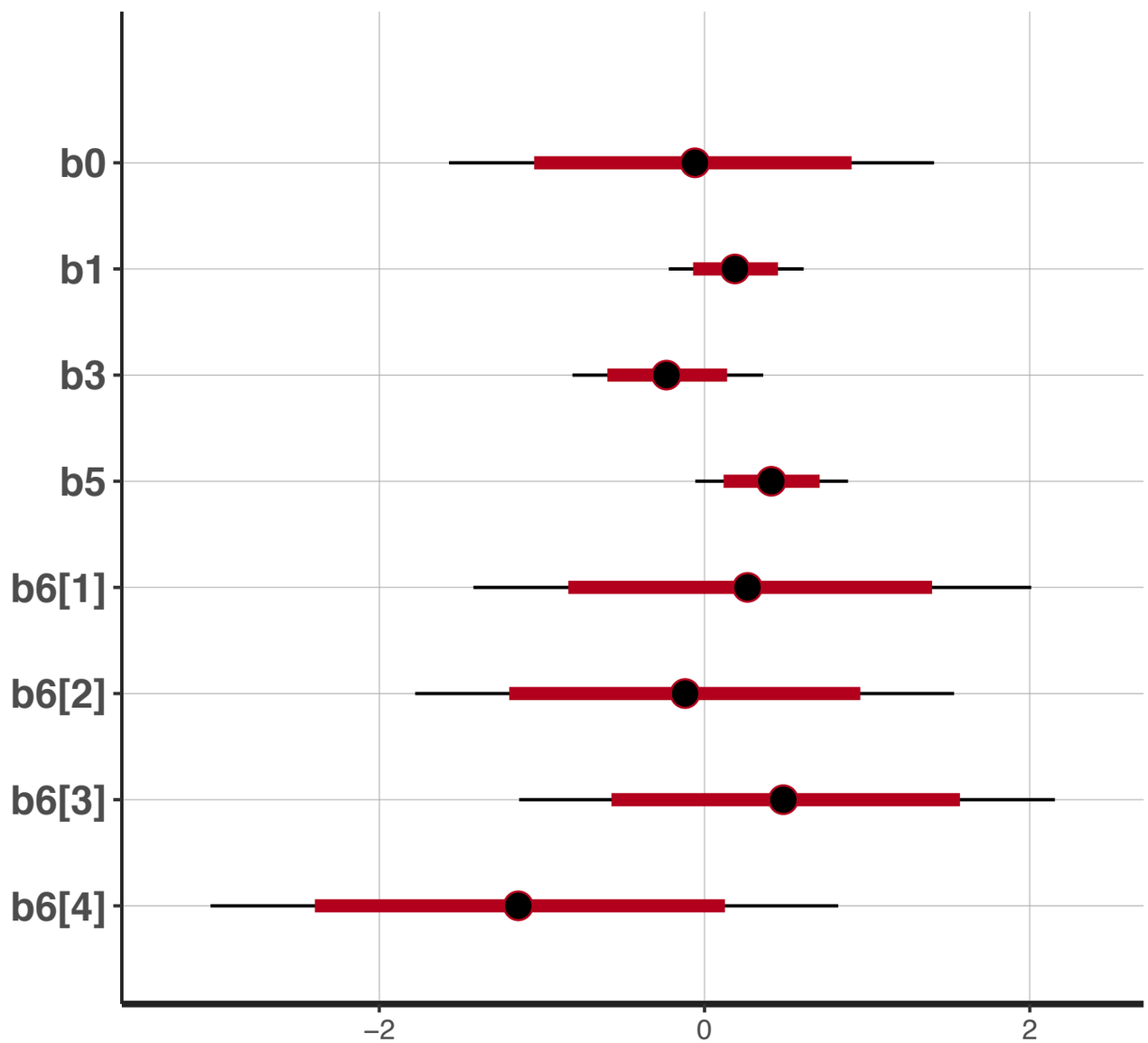
Model 4:
Removing `perc.lactose` and and `perc.fat`

Check MCMC Performance

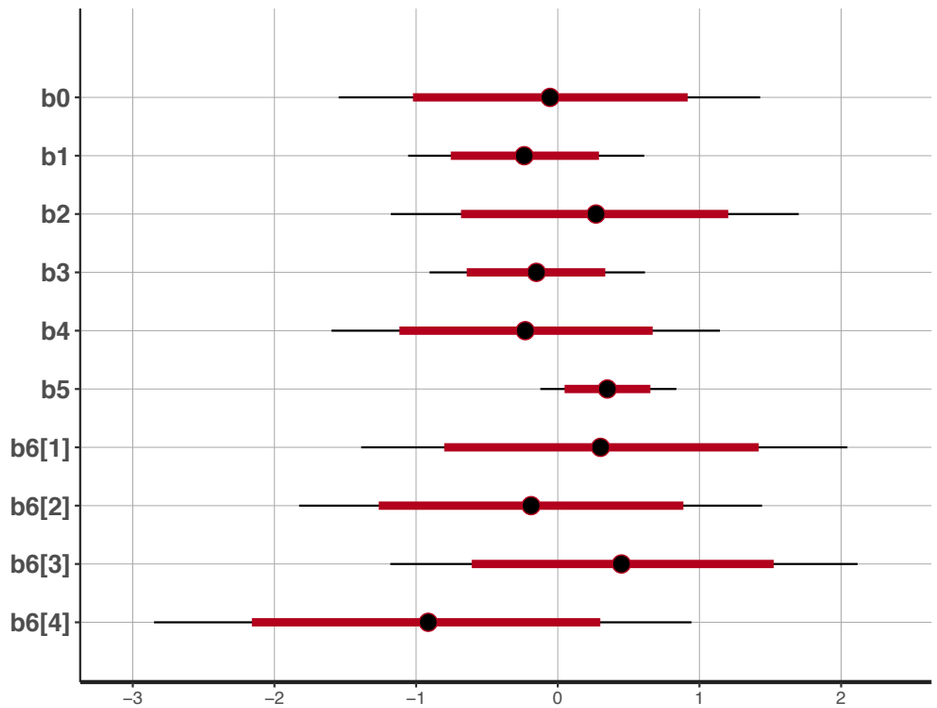
```
print(model_4)

Inference for Stan model: model_4.
3 chains, each with iter=12000; warmup=2000; thin=1;
post-warmup draws per chain=10000, total post-warmup draws=30000.

              mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
b0           -0.06    0.01 0.76 -1.57 -0.57 -0.06  0.45  1.41  7980    1
b1            0.19    0.00 0.21 -0.22  0.05  0.19  0.32  0.61 17521    1
b3           -0.23    0.00 0.29 -0.81 -0.43 -0.23 -0.04  0.36 17174    1
b5            0.41    0.00 0.24 -0.06  0.26  0.41  0.56  0.88 18487    1
b6[1]         0.27    0.01 0.88 -1.42 -0.33  0.26  0.86  2.01  8912    1
b6[2]        -0.12    0.01 0.84 -1.78 -0.69 -0.12  0.44  1.53  8947    1
b6[3]         0.49    0.01 0.84 -1.14 -0.08  0.48  1.05  2.15  8703    1
b6[4]        -1.14    0.01 0.98 -3.04 -1.81 -1.15 -0.48  0.82  9460    1
sigma         0.60    0.00 0.15  0.38  0.50  0.58  0.68  0.94 11897    1
y_pred[1]    -1.61    0.01 0.88 -3.27 -2.19 -1.64 -1.07  0.20 20807    1
y_pred[2]    -0.89    0.00 0.69 -2.29 -1.33 -0.89 -0.45  0.47 28837    1
...
```

Preliminary Evaluation of Parameters

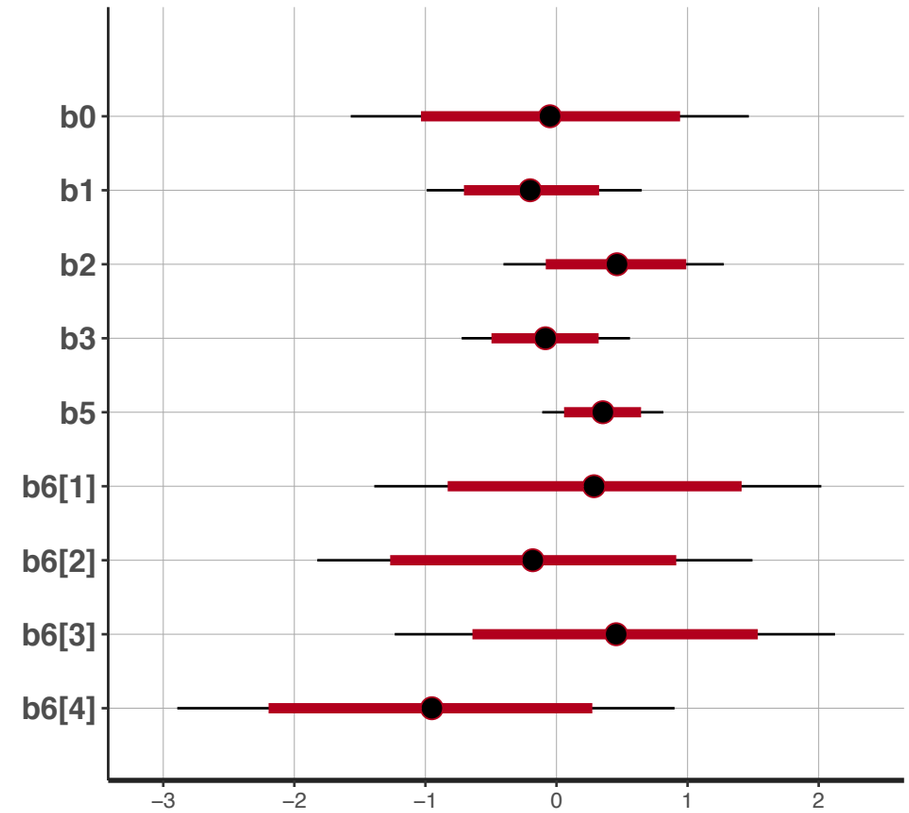
```
stan_plot(model_4, par = c(“b0", “b1", "b3", "b5", "b6"))
```



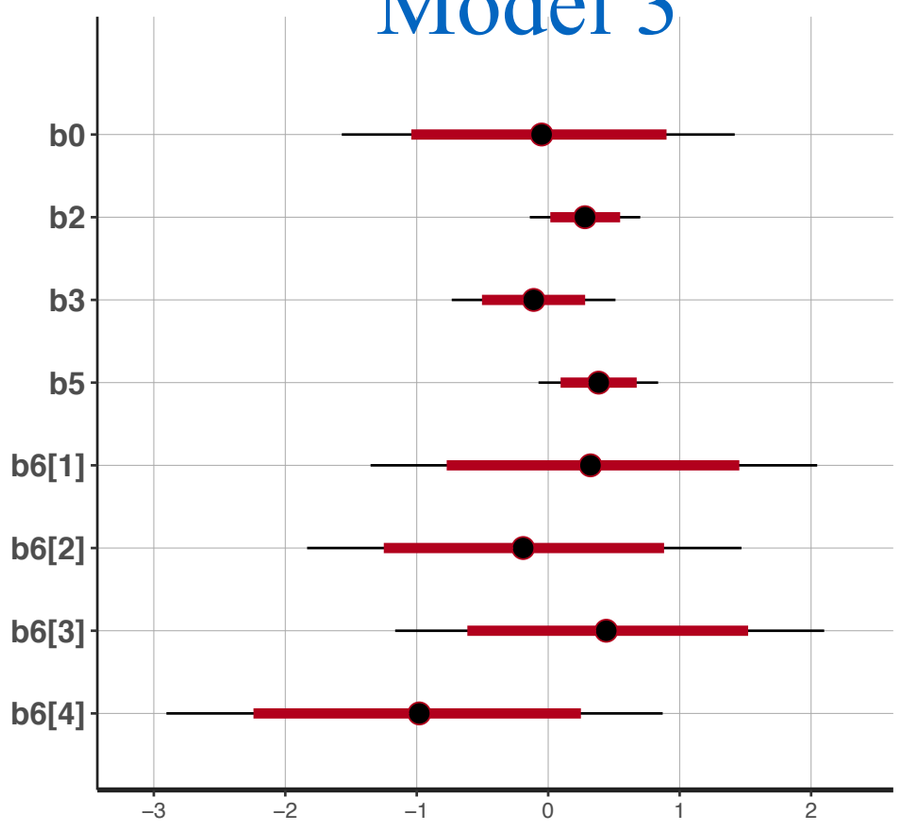
Full Model



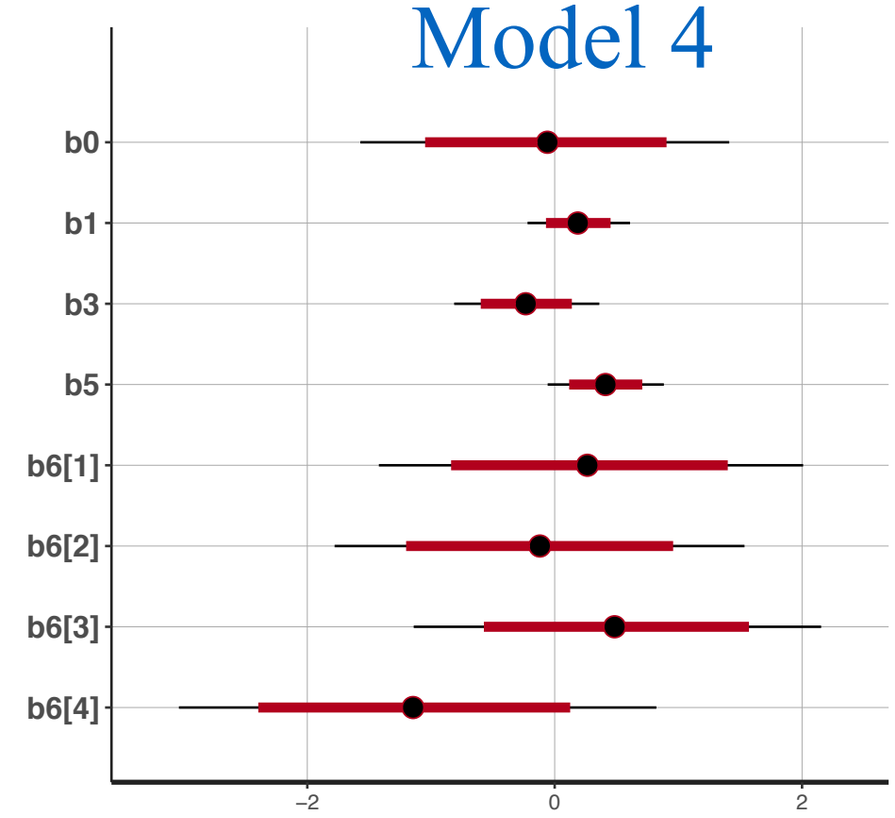
Model 2



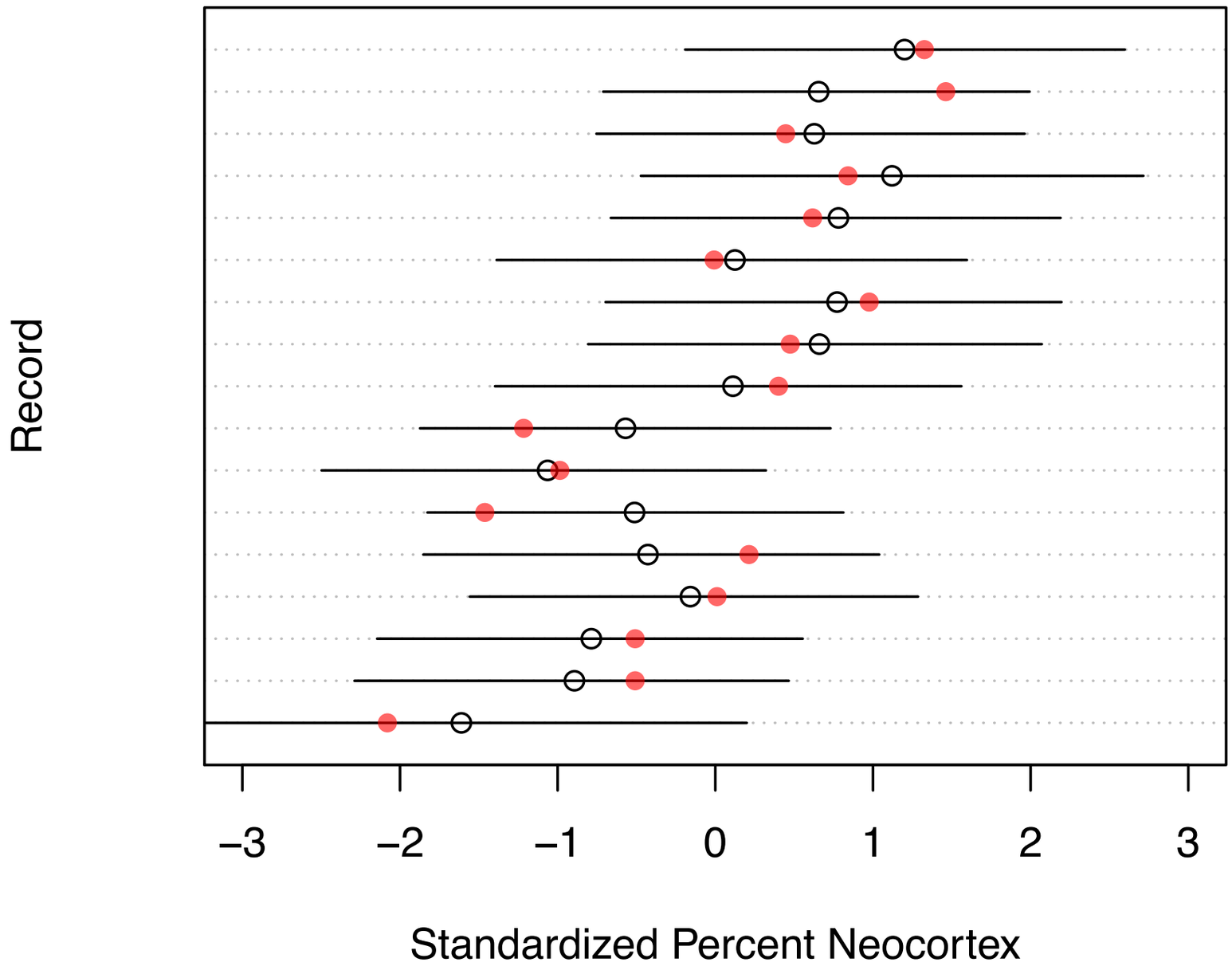
Model 3



Model 4



How Well Does This Model Perform?



How Well Does This Model Perform?

- Get WAIC values and compare

```
logloglik_4 = extract_log_lik(model_4)
waic_4 = waic(loglik_4)

compare(waic_Full, waic_2, waic_3, waic_4)

          elpd_diff elpd_waic se_elpd_waic p_waic se_p_waic waic  se_waic
waic_3      0.0     -17.3       2.3          5.1    1.0      34.6   4.6
waic_2     -0.6     -17.9       2.2          5.5    1.1      35.8   4.4
waic_Full  -0.6     -17.9       2.1          5.5    1.0      35.9   4.2
waic_4     -0.7     -18.0       2.3          5.2    1.1      35.9   4.6
```

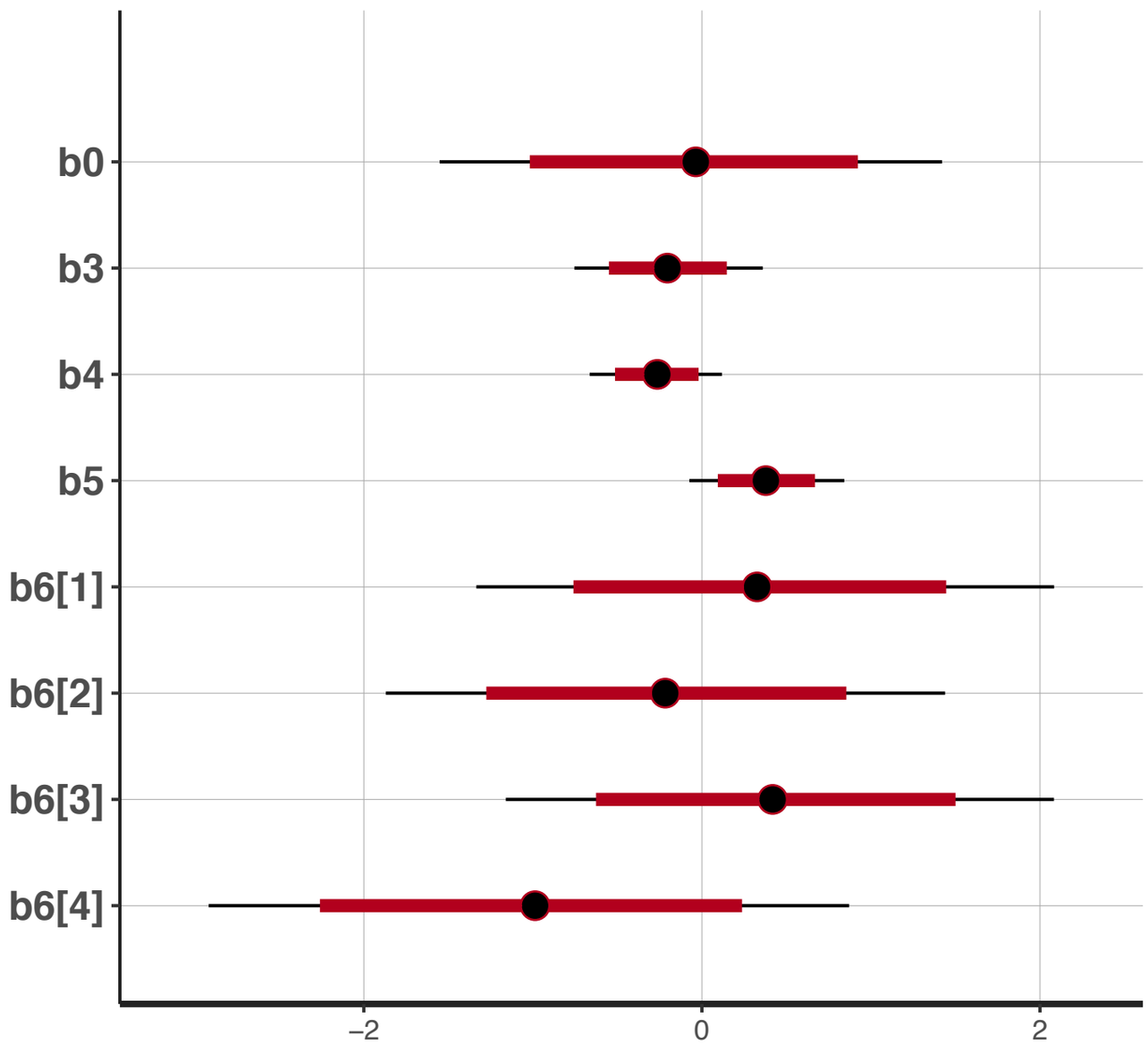
Model 5:
Removing `kcal.per.g` and and `perc.fat`

Check MCMC Performance

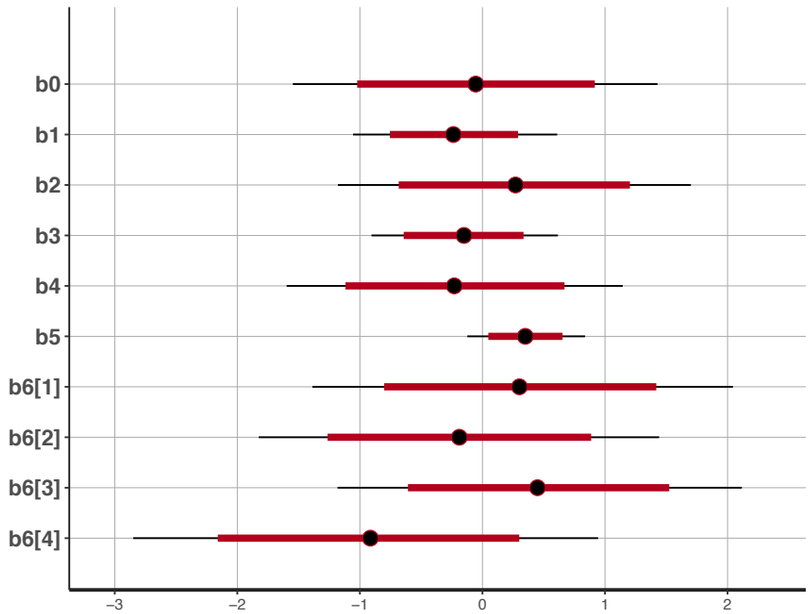
```
print(model_5)

Inference for Stan model: model_5.
3 chains, each with iter=12000; warmup=2000; thin=1;
post-warmup draws per chain=10000, total post-warmup draws=30000.

              mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
b0           -0.04    0.01 0.76 -1.55 -0.55 -0.04  0.47  1.42  6490    1
b3           -0.20    0.00 0.28 -0.75 -0.38 -0.20 -0.02  0.36 18627    1
b4           -0.27    0.00 0.20 -0.66 -0.39 -0.26 -0.14  0.12 16574    1
b5            0.38    0.00 0.23 -0.07  0.23  0.38  0.53  0.84 17486    1
b6[1]         0.33    0.01 0.87 -1.33 -0.26  0.33  0.91  2.08  8073    1
b6[2]        -0.22    0.01 0.84 -1.87 -0.78 -0.22  0.34  1.44  7969    1
b6[3]         0.43    0.01 0.83 -1.16 -0.13  0.42  0.98  2.08  7184    1
b6[4]        -1.00    0.01 0.97 -2.92 -1.65 -0.99 -0.36  0.87  8434    1
sigma         0.58    0.00 0.14  0.37  0.48  0.56  0.65  0.93 10760    1
y_pred[1]    -1.66    0.01 0.84 -3.26 -2.21 -1.68 -1.13  0.09 15503    1
y_pred[2]    -0.86    0.00 0.65 -2.16 -1.27 -0.86 -0.46  0.44 28663    1
...
```

Preliminary Evaluation of Parameters

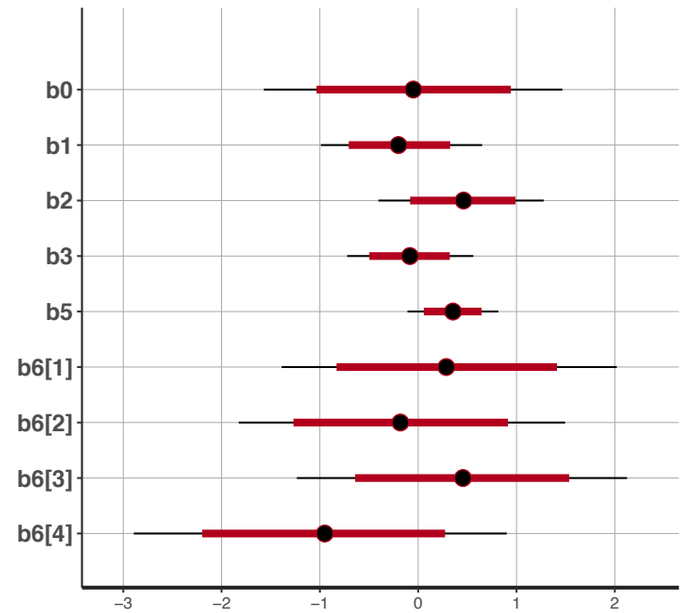
```
stan_plot(model_5, par = c("b0", "b3", "b4", "b5", "b6"))
```



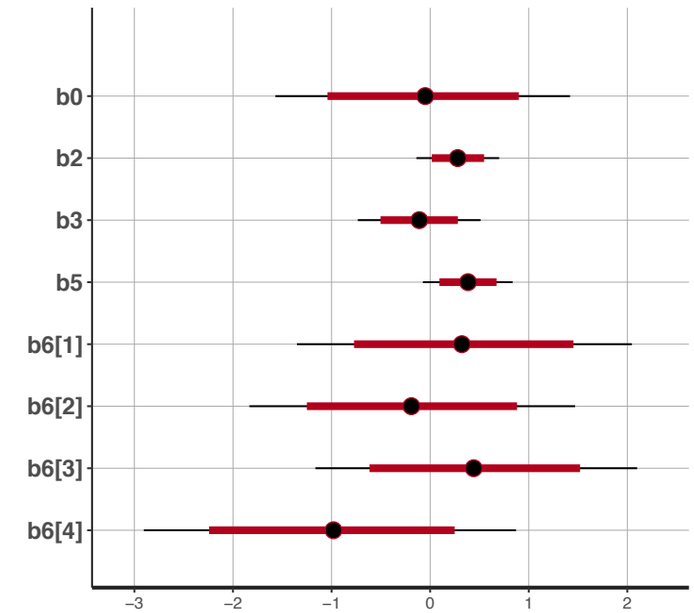
Full Model



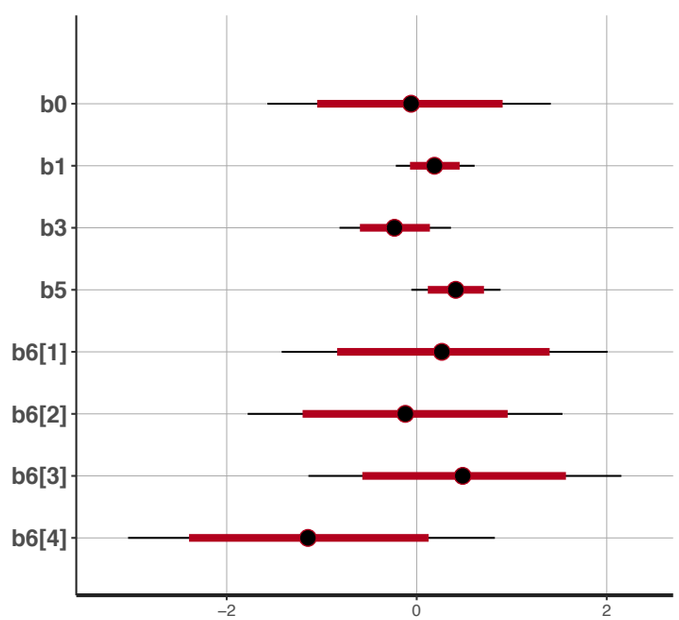
Model 2



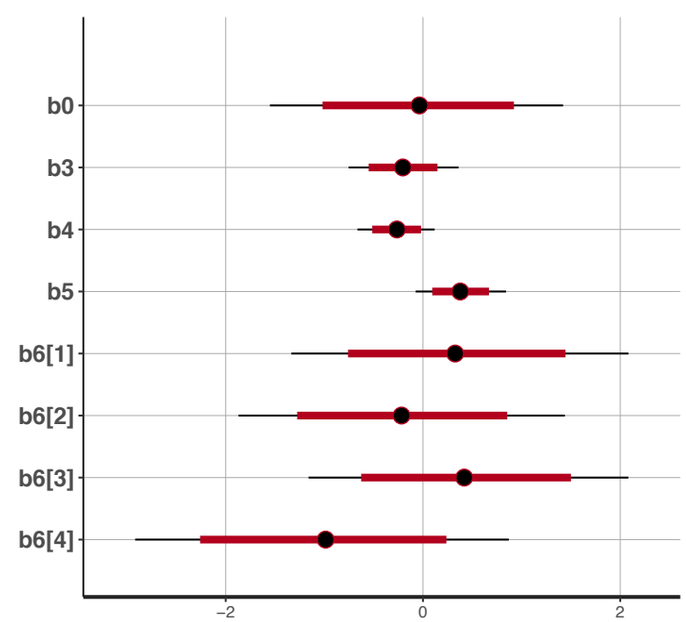
Model 3



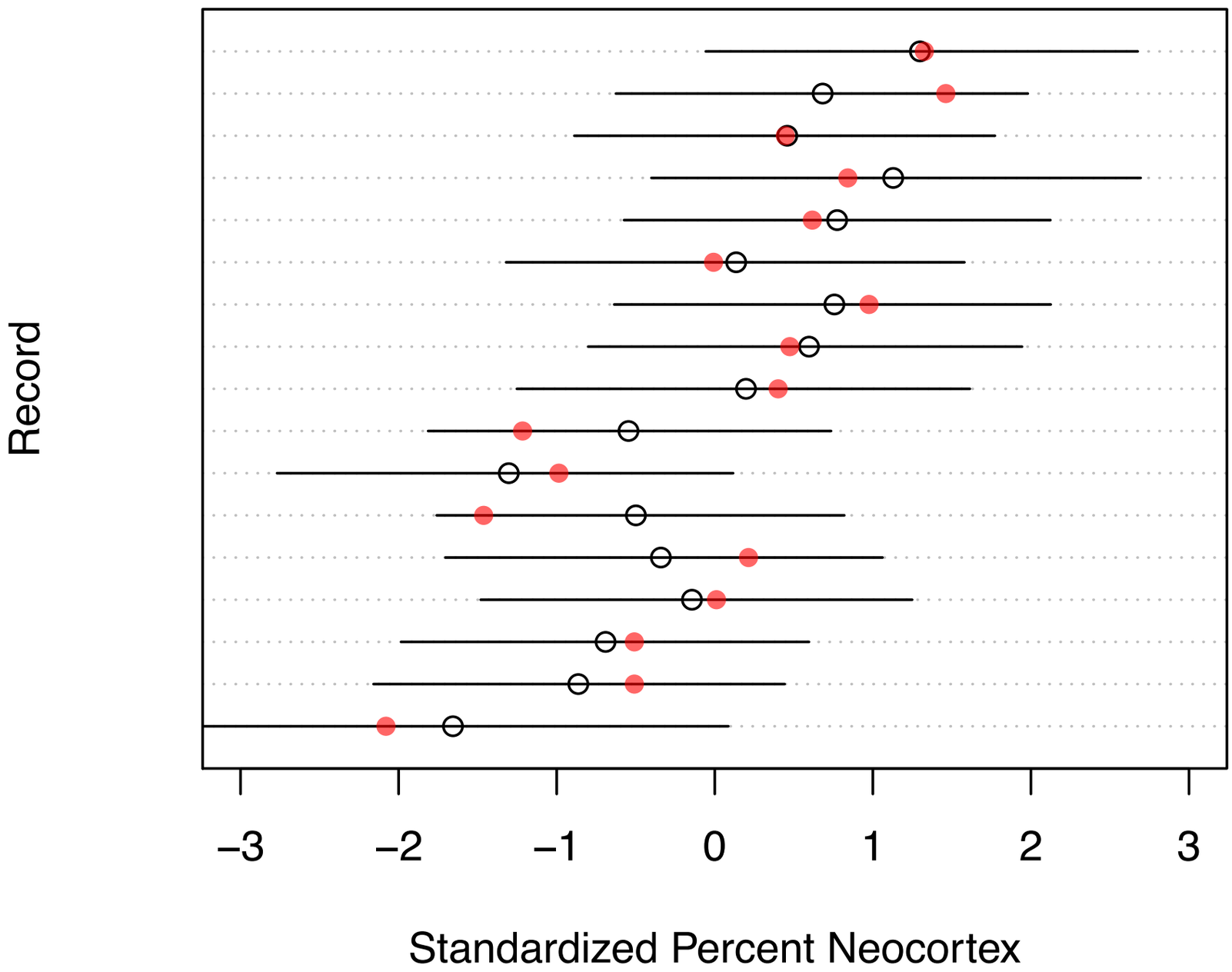
Model 4



Model 5



How Well Does This Model Perform?



How Well Does This Model Perform?

- Get WAIC values and compare

```
loglik_5 = extract_log_lik(model_5)
waic_5 = waic(loglik_5)

compare(waic_Full, waic_2, waic_3, waic_4, waic_5)

          elpd_diff elpd_waic se_elpd_waic p_waic se_p_waic waic  se_waic
waic_5      0.0     -17.3       2.3          5.0    1.0      34.6   4.6
waic_3      0.0     -17.3       2.3          5.1    1.0      34.6   4.6
waic_2     -0.6     -17.9       2.2          5.5    1.1      35.8   4.4
waic_Full  -0.7     -17.9       2.1          5.5    1.0      35.9   4.2
waic_4     -0.7     -18.0       2.3          5.2    1.1      35.9   4.6
```

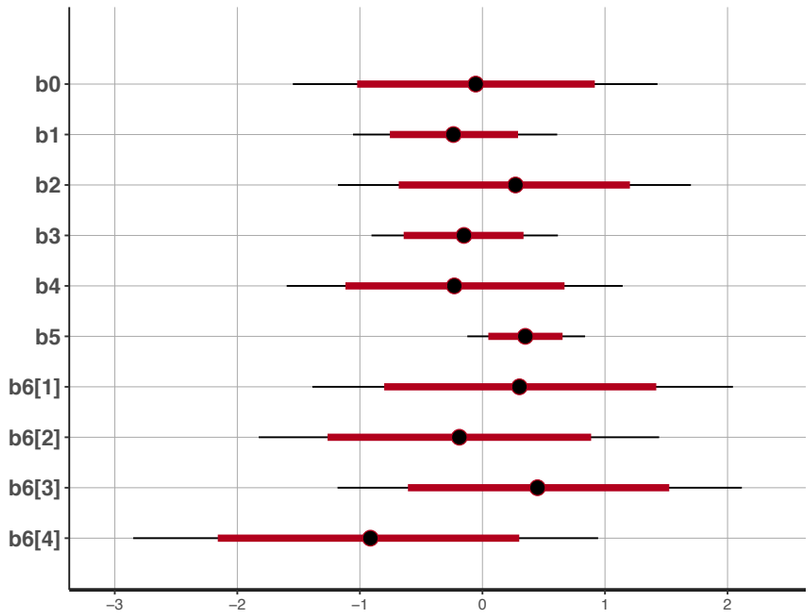
Interpretation

Let’s walk through one parameter

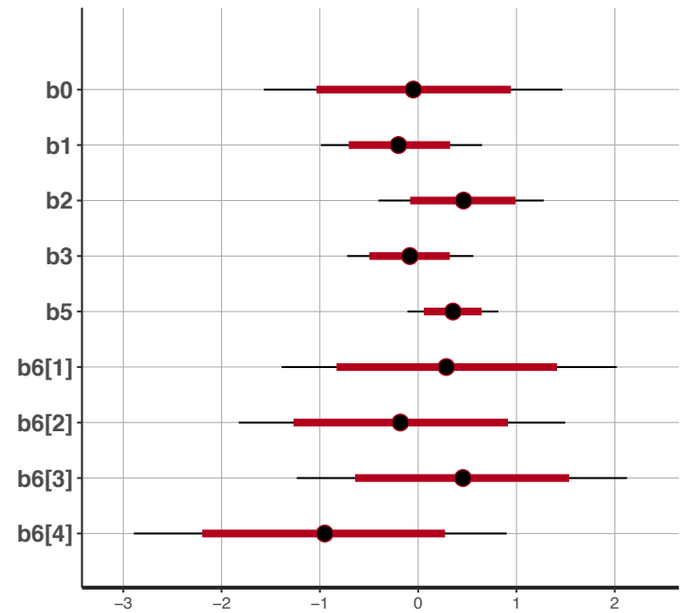
Interpretation

`kcal.per.g` (`x1` and `b1`)

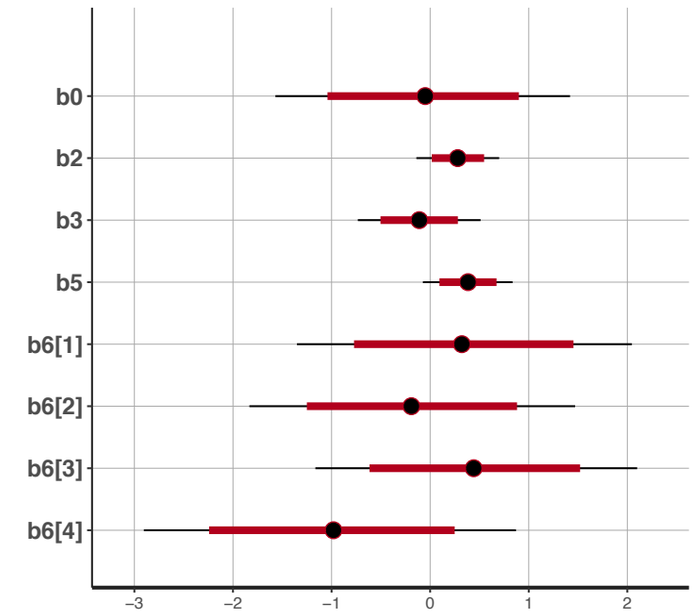
Full Model



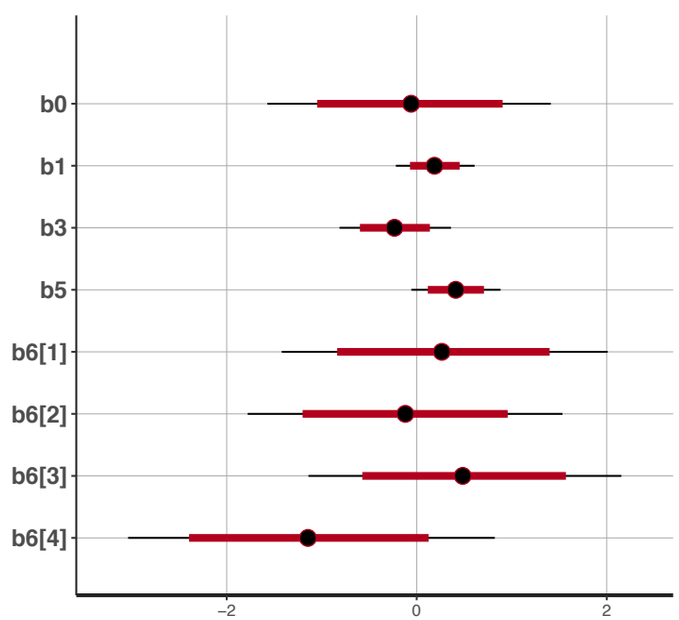
Model 2 (removed perc.lactose)



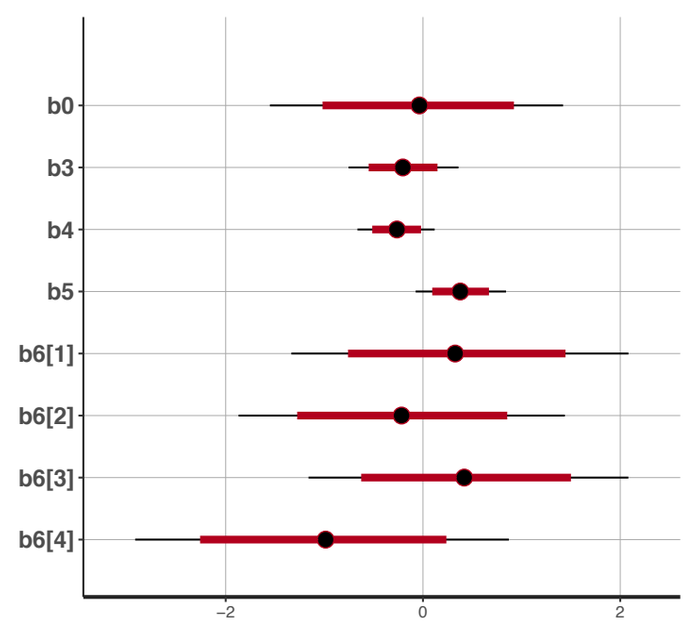
Model 3 (removed perc.lactose & kcal.per.g)



Model 4 (removed perc.lactose and perc.fat)



Model 5 (removed kcal.per.g and perc.fat)



Interpretation

`kcal.per.g` (`x1` and `b1`)

- Seems to have a slight positive effect
  - Masked by correlation with `perc.fat`, less so by correlation with `perc.lactose`

Interpretation

`kcal.per.g` (`x1` and `b1`)

- Seems to have a slight positive effect
  - Masked by correlation with `perc.fat`, less so by correlation with `perc.lactose`
- Is this effect important?
  - Plot predictions at low, mean, and high of other parameters (pick nominals wisely)

Interpretation

`kcal.per.g` (`x1` and `b1`)