Hierarchical two-parameter logistic item response model

Daniel C. Furr

April 21, 2016

Table of contents

1 Model

1.1 Overview

The two-parameter logistic model (2PL) is an item response theory model that includes parameters for both the difficulty and discrimination of items. A hierarchical extension, presented here, models these item parameter pairs as correlated draws from a bivariate normal distribution. This model is similar to the hierarchical three-parameter logistic model proposed by Glas and van der Linden (2003).

$$\mathrm{logit}[\Pr(y_{ij} = 1 | \theta_j, \alpha_i, \beta_i)] = \alpha_i(\theta_j - \beta_i)$$

$$\log \alpha_i, \beta_i \sim \mathrm{MVN}(\mu_1, \mu_2, \Sigma)$$

$$\theta_p \sim \mathrm{N}(0, 1)$$

Variables:

- $i = 1 \ldots I$ indexes items
- $j = 1 \ldots J$ indexes persons

- $y_{ij} \in \{0, 1\}$ is the response of person j to item i

Parameters:

- α_i is the discrimination for item i
- β_i is the difficulty for item i
- θ_j is the ability for person j
- μ_1 is the mean for $\log \alpha_i$
- μ_2 is the mean for β_i
- Σ is the covariance matrix for $\log \alpha_i$ and β_i

Priors:

- $\mu_1 \sim \mathrm{N}(0, 1)$ is a weakly informative prior for the mean of the log discrimination parameters.
- $\mu_2 \sim \mathrm{N}(0, 25)$ is a weakly informative prior for the mean of the difficulty parameters.
- Let $\tau_1^2 = \Sigma_{1,1}$ be the variance of the log discrimination parameters. Then $\tau \sim \mathrm{Exp}(.1)$ is a weakly informative prior for the standard deviation.
- Let $\tau_2^2 = \Sigma_{2,2}$ be the variance of the difficulty parameters. Then $\tau_2 \sim \mathrm{Exp}(.1)$ is a weakly informative prior for the standard deviation.
- A weakly informative prior is placed on the covariance, $\Sigma_{1,2} = \Sigma_{2,1}$, shrinking the posterior towards zero. This is described in more detail in the next section.

1.2 **Stan** program

The **Stan** program for the model is provided below. It differs from the notation above in that it is written in terms of the Cholesky decomposition of the correlation matrix for better efficiency. This matrix is named `L_Omega` ; `Omega` because it is the correlation rather than covariance matrix, and `L` because it is a Cholesky decomposition. The vector `tau` contains the standard deviations that would be the (square root of the) diagonal of the covariance matrix. (The standard deviations are invariant whether or not the the Cholesky decomposition is used.) In the `model` block, `L_Omega` is converted to its covariance equivalent `L_Sigma` , and item parameter pairs `xi[i]` are sampled from `L_Sigma` . The first element of vector `xi[i]` is the log discrimination for item `i` , and the second is the difficulty for item `i` .

Weakly informative normal priors are placed on `mu` , and weakly informative truncated normal priors are placed on `tau` . The prior placed on `L_Omega` using `lkj_corr_cholesky()` is weakly informative and slightly favors a correlation of zero. See the **Stan** manual for details.

While more efficient, parameters `L_Omega` and `xi` are difficult to interpret. To alleviate this inconvenience, item parameters `alpha` and `beta` are derived from `xi` in the `transformed parameters` block. (There is some redundancy, as `beta[i]` and `xi[i,2]` are equal.) Also, `L_Omega` is converted to a standard correlation matrix, `Omega` , in the `generated quantities` block.

```
data {
  int<lower=1> I;               // # items
  int<lower=1> J;               // # persons
  int<lower=1> N;               // # observations
  int<lower=1, upper=I> ii[N];  // item for n
  int<lower=1, upper=J> jj[N];  // person for n
  int<lower=0, upper=1> y[N];   // correctness for n
}
parameters {
  vector[J] theta;              // abilities
  vector[2] xi[I];              // alpha/beta pair vectors
  vector[2] mu;                 // vector for alpha/beta means
  vector<lower=0>[2] tau;       // vector for alpha/beta residual sds
  cholesky_factor_corr[2] L_Omega;
}
transformed parameters {
  vector[I] alpha;
  vector[I] beta;
  for (i in 1:I) {
    alpha[i] <- exp(xi[i,1]);
    beta[i] <- xi[i,2];
  }
}
model {
  matrix[2,2] L_Sigma;
  L_Sigma <- diag_pre_multiply(tau, L_Omega);
  for (i in 1:I)
    xi[i] ~ multi_normal_cholesky(mu, L_Sigma);
  theta ~ normal(0, 1);
  L_Omega ~ lkj_corr_cholesky(4);
  mu[1] ~ normal(0,1);
  tau[1] ~ exponential(.1);
  mu[2] ~ normal(0,5);
  tau[2] ~ exponential(.1);
  y ~ bernoulli_logit(alpha[ii] .* (theta[jj] - beta[ii]));
}
generated quantities {
  corr_matrix[2] Omega;
  Omega <- multiply_lower_tri_self_transpose(L_Omega);
}
```

2 Simulation

First, the necessary **R** packages are loaded.

```
# Load R packages
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
library(ggplot2)
```

The **R** code that follows simulates a dataset conforming to the model. The **Stan** model will be evaluated in terms of its ability to recover the generating values of the parameters when fit to this dataset.

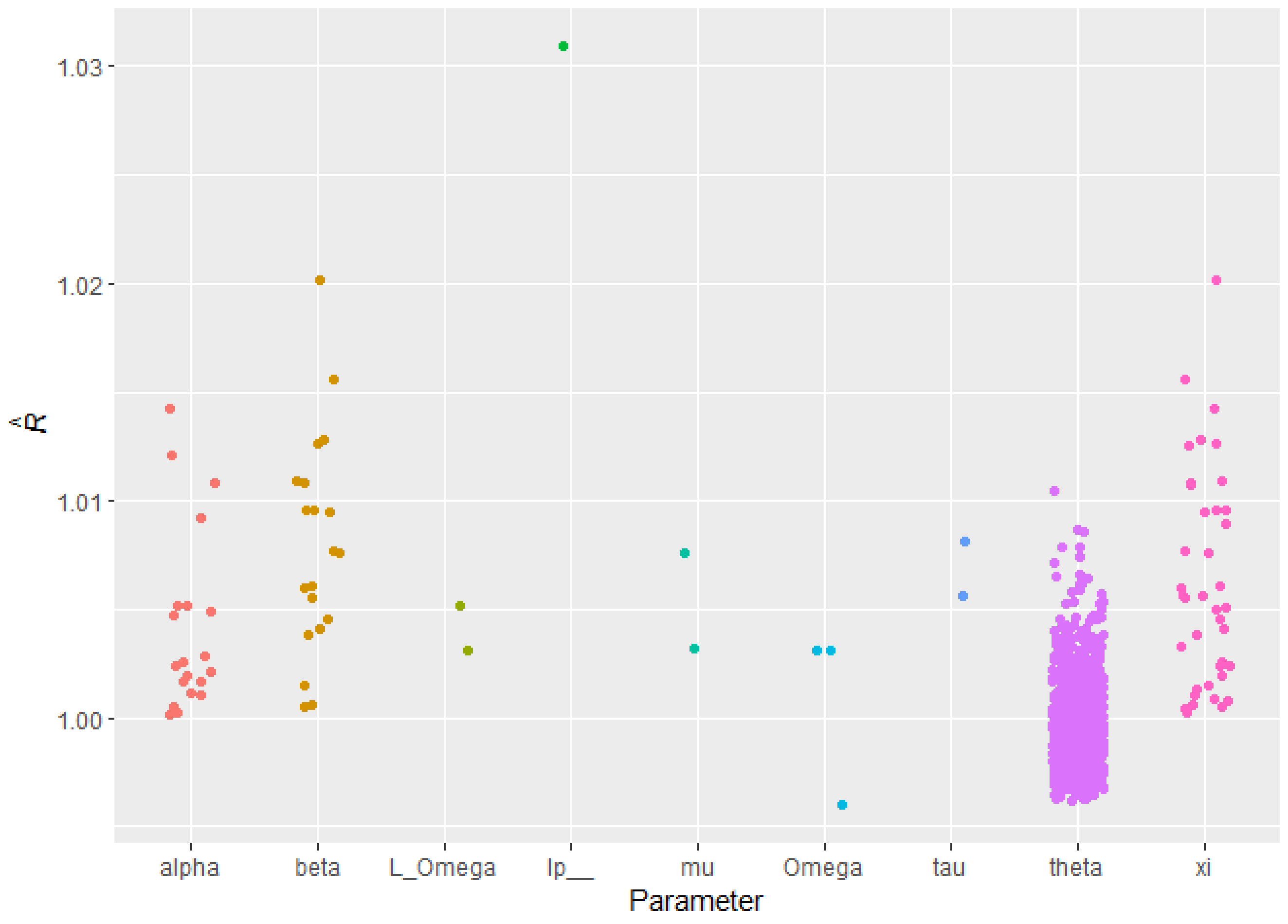
```
# Set paramters for the simulated data
I <- 20
J <- 1000
mu <- c(0, 0)
tau <- c(0.25, 1)
Omega <- matrix(c(1, 0.3, 0.3, 1), ncol = 2)

# Calculate or sample remaining paramters
Sigma <- tau %*% t(tau) * Omega
xi <- MASS::mvrnorm(I, c(0, 0), Sigma)
alpha <- exp(mu[1] + as.vector(xi[, 1]))
beta <- as.vector(mu[2] + xi[, 2])
theta <- rnorm(J, mean = 0, sd = 1)

# Assemble data and simulate response
data_list <- list(I = I, J = J, N = I * J, ii = rep(1:I, times = J), jj = rep(1:J,
    each = I))
eta <- alpha[data_list$ii] * (theta[data_list$jj] - beta[data_list$ii])
data_list$y <- as.numeric(boot::inv.logit(eta) > runif(data_list$N))
```

The simulated data consists of 20 items and 1000 persons. The log discriminations have mean 0 and standard deviation 0.25. The difficulties have mean 0 and standard deviation 1. The correlation between the log discrimination residuals and difficulty residuals is 0.3. The simulated dataset is fit with **Stan**.

```
# Fit model to simulated data
sim_fit <- stan(file = "hierarchical_2pl.stan", data = data_list, chains = 4,
    iter = 500)
```

Before interpreting the results, it is necessary to check that the chains have converged. **Stan** provides the $\hat{R}$ statistic for the model parameters and log posterior. These are provided in the following figure. All values for $\hat{R}$ should be less than 1.1.

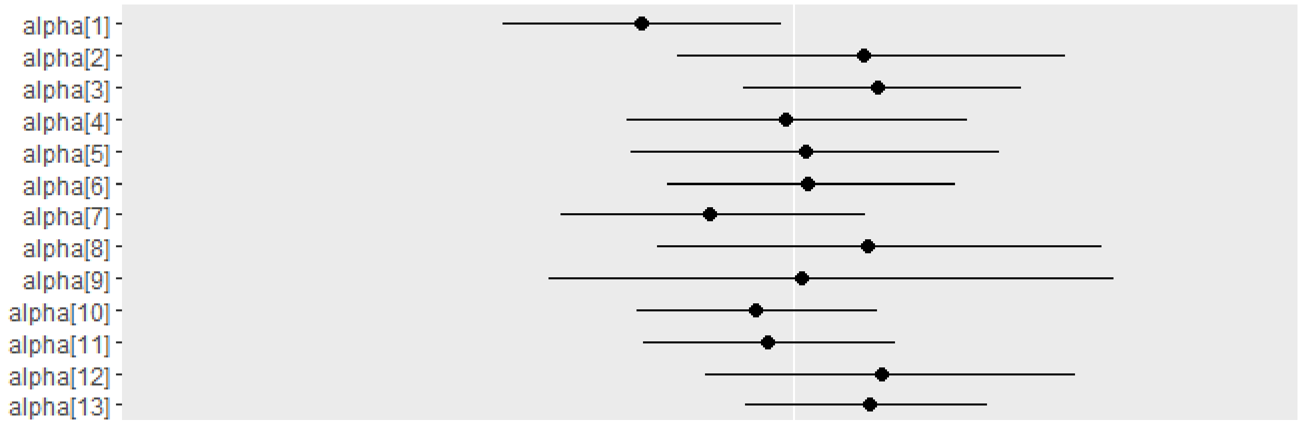
```
sim_summary <- as.data.frame(summary(sim_fit)[[1]])
sim_summary$Parameter <- as.factor(gsub("\\[.*]", "", rownames(sim_summary)))
ggplot(sim_summary) + aes(x = Parameter, y = Rhat, color = Parameter) + geom_jitter(height = 0,
    width = 0.5, show.legend = FALSE) + ylab(expression(hat(italic(R))))
```

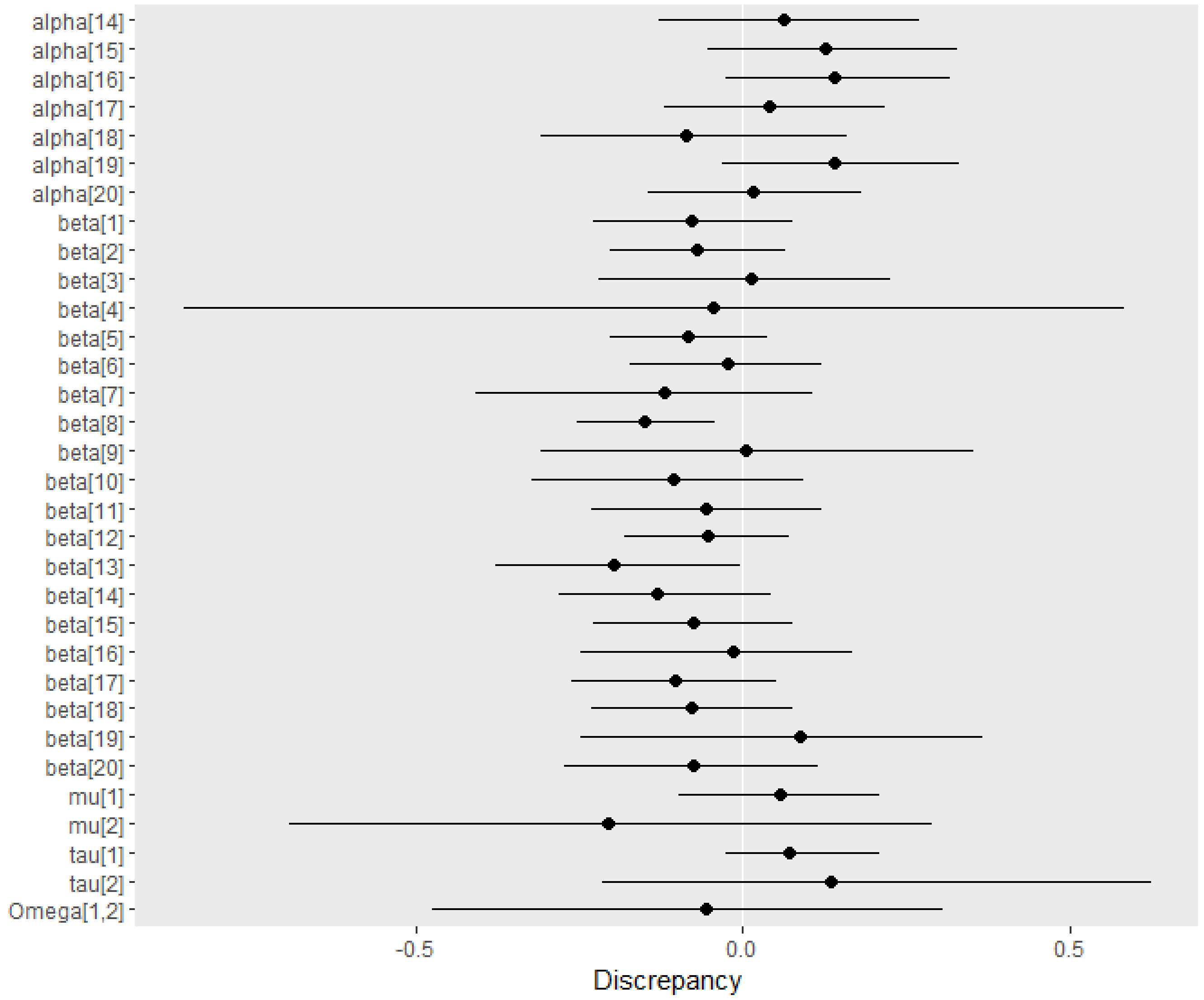


Convergence statistics ($\hat{R}$) by parameter for the simulation. All values should be less than 1.1 to infer convergence.

The **Stan** model is evaluated in terms of its ability to recover the generating values of the parameters. The R code below prepares a plot in which the points indicate the difference between the posterior means and generating values for the parameters of main interest. This difference is referred to as discrepancy. The lines indicate the 95% posterior intervals for the difference. Ideally, (nearly) all the 95% posterior intervals would include zero.

```
# Make vector of wanted parameter names
wanted_pars <- c(paste0("alpha[", 1:I, "]"), paste0("beta[", 1:I, "]"), c("mu[1]",
    "mu[2]", "tau[1]", "tau[2]", "Omega[1,2]"))

# Get estimated and generating values for wanted parameters
generating_values = c(alpha, beta, mu, tau, Omega[1, 2])
estimated_values <- sim_summary[wanted_pars, c("mean", "2.5%", "97.5%")]

# Assesmble a data frame to pass to ggplot()
sim_df <- data.frame(parameter = factor(wanted_pars, rev(wanted_pars)), row.names = NULL)
sim_df$middle <- estimated_values[, "mean"] - generating_values
sim_df$lower <- estimated_values[, "2.5%"] - generating_values
sim_df$upper <- estimated_values[, "97.5%"] - generating_values

# Plot the discrepancy
ggplot(sim_df) + aes(x = parameter, y = middle, ymin = lower, ymax = upper) +
    scale_x_discrete() + geom_abline(intercept = 0, slope = 0, color = "white") +
    geom_linerange() + geom_point(size = 2) + labs(y = "Discrepancy", x = NULL) +
    theme(panel.grid = element_blank()) + coord_flip()
```





Discrepancies between estimated and generating parameters. Points indicate the difference between the posterior means and generating values for a parameter, and horizontal lines indicate 95% posterior intervals for the difference. Most of the discrepancies are about zero, indicating that **Stan** successfully recovers the true parameters.

3 Example application

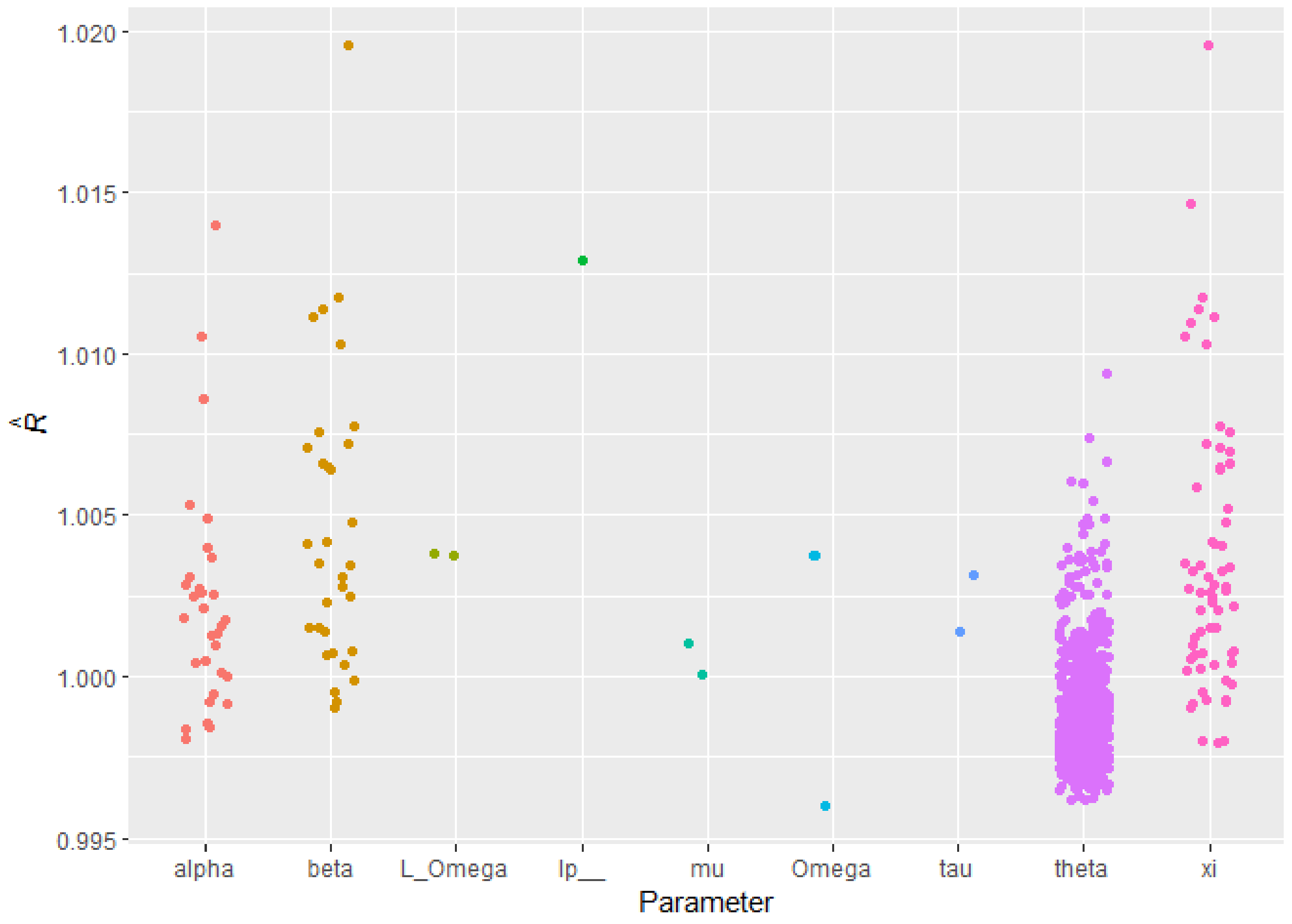
```
# Use data and scoring function from the mirt package
library(mirt)
sat <- key2binary(SAT12, key = c(1, 4, 5, 2, 3, 1, 2, 1, 3, 1, 2, 4, 2, 1, 5,
    3, 4, 4, 1, 4, 3, 3, 4, 1, 3, 5, 1, 3, 1, 5, 4, 5))
```

The example data (Wood et al. 2003) are from a grade 12 science assessment. 600 students responded to 32 dichotomously scored items. Non-responses were scored as incorrect, so the data contain no missing values. The scored response matrix is converted to list form and fit with **Stan**.

```
# Assemble data list and fit model
sat_list <- list(I = ncol(sat), J = nrow(sat), N = length(sat), ii = rep(1:ncol(sat),
    each = nrow(sat)), jj = rep(1:nrow(sat), times = ncol(sat)), y = as.vector(sat))
sat_fit <- stan(file = "hierarchical_2pl.stan", data = sat_list, chains = 4,
    iter = 500)
```

As discussed above, convergence of the chains is assessed for every parameter, and also the log posterior, using $\hat{R}$.

```
ex_summary <- as.data.frame(summary(sat_fit)[[1]])
ex_summary$Parameter <- as.factor(gsub("\\[.*]", "", rownames(ex_summary)))
ggplot(ex_summary) + aes(x = Parameter, y = Rhat, color = Parameter) + geom_jitter(height = 0,
    width = 0.5, show.legend = FALSE) + ylab(expression(hat(italic(R))))
```



Convergence statistics ($\hat{R}$) by parameter for the SAT data. All values should be less than 1.1 to infer convergence.

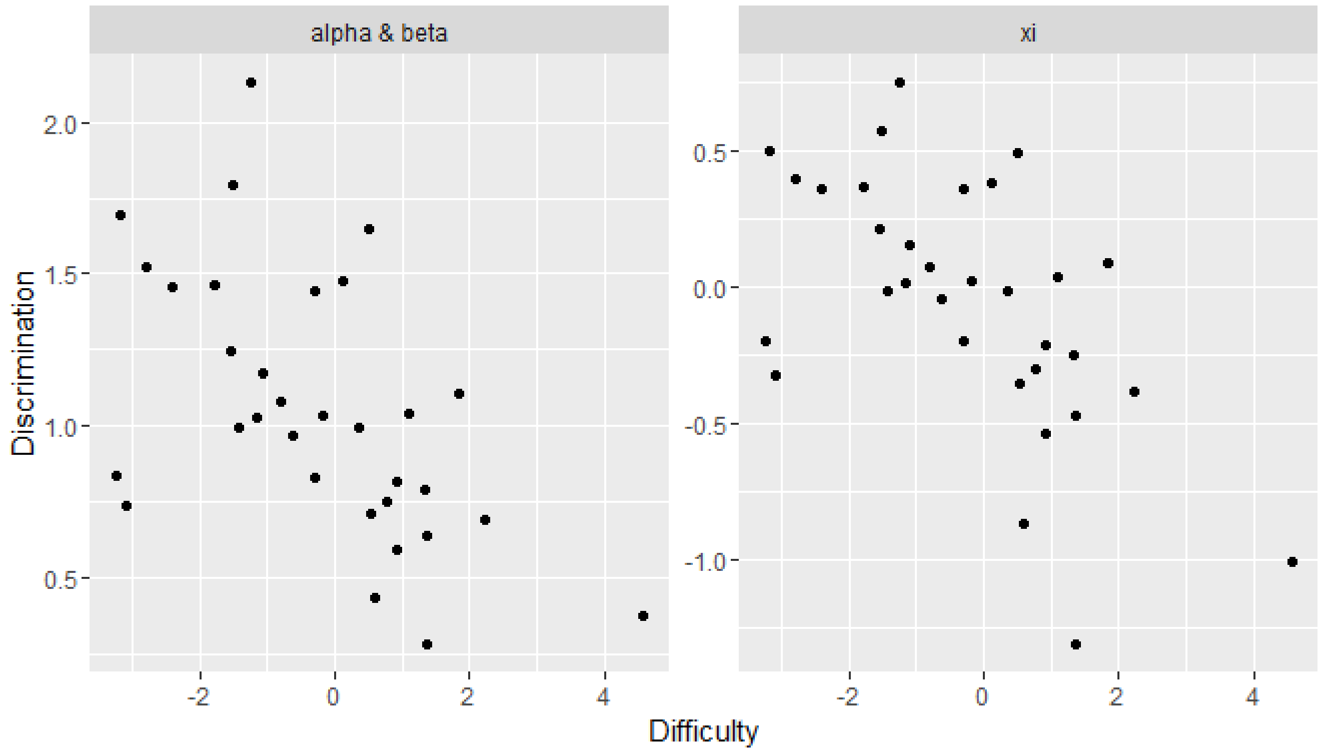
Next we view summaries of the parameter posteriors.

```
# View table of parameter posteriors
print(sat_fit, pars = c("alpha", "beta", "mu", "tau", "Omega[1,2]"))
```

```
## Inference for Stan model: hierarchical_2pl.
## 4 chains, each with iter=500; warmup=250; thin=1; 
## post-warmup draws per chain=250, total post-warmup draws=1000.
## 
##              mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha[1]     0.79    0.00 0.12  0.57  0.71  0.78  0.86  1.03   605 1.00
## alpha[2]     1.44    0.01 0.17  1.13  1.33  1.44  1.56  1.77  1000 1.00
## alpha[3]     1.04    0.00 0.13  0.81  0.95  1.04  1.12  1.31   717 1.00
## alpha[4]     0.59    0.00 0.10  0.40  0.52  0.59  0.66  0.80  1000 1.00
## alpha[5]     0.97    0.00 0.13  0.74  0.88  0.96  1.05  1.25  1000 1.01
## alpha[6]     1.11    0.01 0.15  0.83  1.00  1.11  1.20  1.40   640 1.00
## alpha[7]     0.99    0.01 0.14  0.74  0.90  0.99  1.09  1.28   574 1.00
## alpha[8]     0.69    0.00 0.11  0.48  0.62  0.69  0.76  0.91   600 1.00
## alpha[9]     0.73    0.00 0.12  0.51  0.65  0.73  0.81  1.00   663 1.00
## alpha[10]    0.99    0.00 0.12  0.77  0.91  0.98  1.07  1.25  1000 1.00
## alpha[11]    1.70    0.01 0.39  1.03  1.43  1.64  1.91  2.59   799 1.00
## alpha[12]    0.28    0.00 0.07  0.15  0.23  0.28  0.32  0.43   767 1.00
## alpha[13]    1.08    0.01 0.14  0.83  0.99  1.08  1.17  1.35   700 1.00
## alpha[14]    1.02    0.01 0.14  0.77  0.93  1.02  1.11  1.32   652 1.00
## alpha[15]    1.25    0.01 0.18  0.93  1.12  1.24  1.36  1.62   467 1.01
## alpha[16]    0.71    0.00 0.10  0.53  0.64  0.70  0.77  0.92  1000 1.00
## alpha[17]    1.52    0.01 0.30  1.00  1.32  1.49  1.72  2.15   638 1.00
## alpha[18]    1.65    0.01 0.18  1.33  1.52  1.64  1.77  2.04   548 1.00
## alpha[19]    0.83    0.00 0.11  0.61  0.75  0.82  0.90  1.06  1000 1.00
## alpha[20]    1.46    0.01 0.22  1.08  1.32  1.44  1.59  1.94   493 1.00
## alpha[21]    0.83    0.01 0.15  0.57  0.72  0.83  0.94  1.14   732 1.00
## alpha[22]    1.46    0.01 0.27  0.98  1.27  1.44  1.62  2.03   440 1.01
## alpha[23]    0.63    0.00 0.10  0.45  0.56  0.63  0.70  0.85  1000 1.00
## alpha[24]    1.17    0.01 0.15  0.90  1.06  1.17  1.27  1.51   728 1.00
## alpha[25]    0.75    0.00 0.11  0.54  0.67  0.74  0.83  0.97  1000 1.00
## alpha[26]    1.48    0.01 0.16  1.20  1.36  1.47  1.59  1.79   599 1.00
## alpha[27]    1.79    0.01 0.27  1.31  1.60  1.78  1.98  2.35   400 1.01
## alpha[28]    1.03    0.00 0.13  0.80  0.95  1.03  1.12  1.31  1000 1.00
## alpha[29]    0.82    0.00 0.11  0.60  0.74  0.81  0.89  1.05   689 1.00
## alpha[30]    0.43    0.00 0.08  0.26  0.37  0.43  0.49  0.60   717 1.00
## alpha[31]    2.13    0.02 0.30  1.58  1.93  2.11  2.31  2.75   347 1.00
## alpha[32]    0.37    0.00 0.07  0.25  0.32  0.37  0.42  0.53   415 1.00
## beta[1]      1.34    0.01 0.22  0.97  1.20  1.32  1.46  1.84   598 1.00
## beta[2]     -0.31    0.00 0.08 -0.47 -0.36 -0.30 -0.25 -0.15   354 1.01
## beta[3]      1.09    0.01 0.15  0.82  0.99  1.08  1.18  1.42   683 1.00
## beta[4]     0.92    0.01 0.22  0.57  0.76  0.90  1.05  1.38  1000 1.01
## beta[5]    -0.64    0.01 0.12 -0.89 -0.71 -0.63 -0.56 -0.41   399 1.01
## beta[6]     1.84    0.01 0.21  1.49  1.69  1.82  1.98  2.30   594 1.00
## beta[7]    -1.42    0.01 0.19 -1.85 -1.52 -1.41 -1.30 -1.10   517 1.00
## beta[8]     2.21    0.02 0.36  1.66  1.96  2.17  2.41  3.12   470 1.00
## beta[9]    -3.08    0.02 0.47 -4.11 -3.36 -3.02 -2.75 -2.30   499 1.00
## beta[10]    0.36    0.00 0.10  0.18  0.29  0.35  0.42  0.57  1000 1.01
## beta[11]   -3.19    0.02 0.49 -4.36 -3.46 -3.12 -2.85 -2.38   668 1.00
## beta[12]    1.36    0.02 0.49  0.64  1.01  1.27  1.59  2.60   625 1.00
## beta[13]   -0.79    0.01 0.12 -1.03 -0.86 -0.78 -0.71 -0.58   472 1.00
## beta[14]   -1.16    0.01 0.15 -1.50 -1.25 -1.15 -1.05 -0.91   666 1.00
## beta[15]   -1.55    0.01 0.18 -1.92 -1.66 -1.53 -1.42 -1.23   397 1.02
## beta[16]    0.54    0.00 0.15  0.26  0.43  0.53  0.64  0.83  1000 1.00
## beta[17]   -2.80    0.02 0.44 -3.74 -3.04 -2.74 -2.50 -2.12   495 1.00
## beta[18]    0.51    0.00 0.08  0.35  0.45  0.50  0.56  0.68   479 1.01
## beta[19]   -0.29    0.00 0.11 -0.50 -0.36 -0.29 -0.22 -0.08  1000 1.00
## beta[20]   -1.78    0.01 0.19 -2.19 -1.89 -1.76 -1.64 -1.43   434 1.01
## beta[21]   -3.23    0.02 0.53 -4.49 -3.55 -3.14 -2.84 -2.39   619 1.00
## beta[22]   -2.39    0.02 0.33 -3.10 -2.57 -2.37 -2.17 -1.85   425 1.01
## beta[23]    1.36    0.01 0.26  0.95  1.20  1.34  1.51  1.94   725 1.00
## beta[24]   -1.09    0.00 0.13 -1.38 -1.17 -1.08 -0.99 -0.84   716 1.00
## beta[25]    0.77    0.01 0.16  0.49  0.65  0.75  0.87  1.11   721 1.01
## beta[26]    0.11    0.00 0.08 -0.03  0.06  0.11  0.16  0.26   576 1.00
## beta[27]   -1.52    0.01 0.15 -1.86 -1.62 -1.51 -1.41 -1.26   289 1.01
## beta[28]   -0.17    0.00 0.10 -0.38 -0.23 -0.17 -0.11  0.02  1000 1.00
## beta[29]    0.93    0.01 0.16  0.66  0.81  0.91  1.04  1.27   557 1.00
## beta[30]    0.60    0.01 0.25  0.17  0.44  0.57  0.73  1.19   555 1.01
## beta[31]   -1.26    0.01 0.11 -1.50 -1.32 -1.26 -1.18 -1.06   226 1.01
## beta[32]    4.57    0.04 0.92  3.17  3.90  4.43  5.16  6.43   606 1.00
## mu[1]      -0.05    0.00 0.10 -0.25 -0.11 -0.05  0.02  0.13  1000 1.00
## mu[2]      -0.25    0.01 0.34 -0.91 -0.48 -0.26 -0.02  0.39  1000 1.00
## tau[1]      0.50    0.00 0.08  0.38  0.45  0.50  0.55  0.68   708 1.00
## tau[2]      1.83    0.01 0.28  1.37  1.64  1.80  1.99  2.44   523 1.00
## Omega[1,2] -0.44    0.00 0.15 -0.70 -0.56 -0.45 -0.34 -0.12  1000 1.00
## 
## Samples were drawn using NUTS(diag_e) at Tue Apr 19 16:52:58 2016.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

To visualize the correlation between item parameters, the values of `alpha[i]` and `beta[i]` may be plotted against one another. This is presented in the left side of the figure below. Because the correlation is actually between `xi[i,1]` (which is equal to `beta[i]`) and `xi[i,2]` (which is `alpha[i]` on the log scale), a scatter plot pf `xi[i,1]` versus `xi[i,2]` is given on the right side.

```
# Assesmble a data frame of item parameter estimates and pass to ggplot
ab_df <- data.frame(Discrimination = ex_summary[paste0("alpha[", 1:sat_list$I,
    "]"), "mean"], Difficulty = ex_summary[paste0("beta[", 1:sat_list$I, "]"),
    "mean"], parameterization = "alpha & beta")
xi_df <- data.frame(Discrimination = ex_summary[paste0("xi[", 1:sat_list$I,
    ",1]"), "mean"], Difficulty = ex_summary[paste0("xi[", 1:sat_list$I, ",2]"),
    "mean"], parameterization = "xi")
full_df <- rbind(ab_df, xi_df)
ggplot(full_df) + aes(x = Difficulty, y = Discrimination) + geom_point() + facet_wrap(~parameterization,
    scales = "free")
```



Discrimination versus difficulty parameters for the SAT data. The lefthand plot shows the pairs in terms of alpha and beta, the usual formulation. The righthand plot shows the pairs in terms of xi, as used by the Stan model.

4 References

Glas, Cees AW, and Wim J van der Linden. 2003. “Computerized Adaptive Testing with Item Cloning.” *Applied Psychological Measurement* 27 (4). Sage Publications: 247–61.

Wood, R., Wilson D. T., Gibbons R. D., Schilling S. G., Muraki E., and R. D. Bock. 2003. *TESTFACT 4 for Windows: Test Scoring, Item Statistics, and Full-Information Item Factor Analysis*. Lincolnwood, IL: Scientific Software International.