BDA project report

Amanda Aarnio, Anni Niskanen, Antti Huttunen

2022-12-01

Contents

1 Introduction

Can the grade point averages (GPAs) of first-year college students be predicted based on high school GPA and other factors. This report presents a solution for this problem by applying two statiscal models to predict the first-year college GPA. The models are constracted and compared according to Bayesian data analysis concepts.

```
library(cmdstanr)
library(bayesplot)
library(gridExtra)
library(loo)
library(Stat2Data)

data("FirstYearGPA")
```

2 Data and Problem

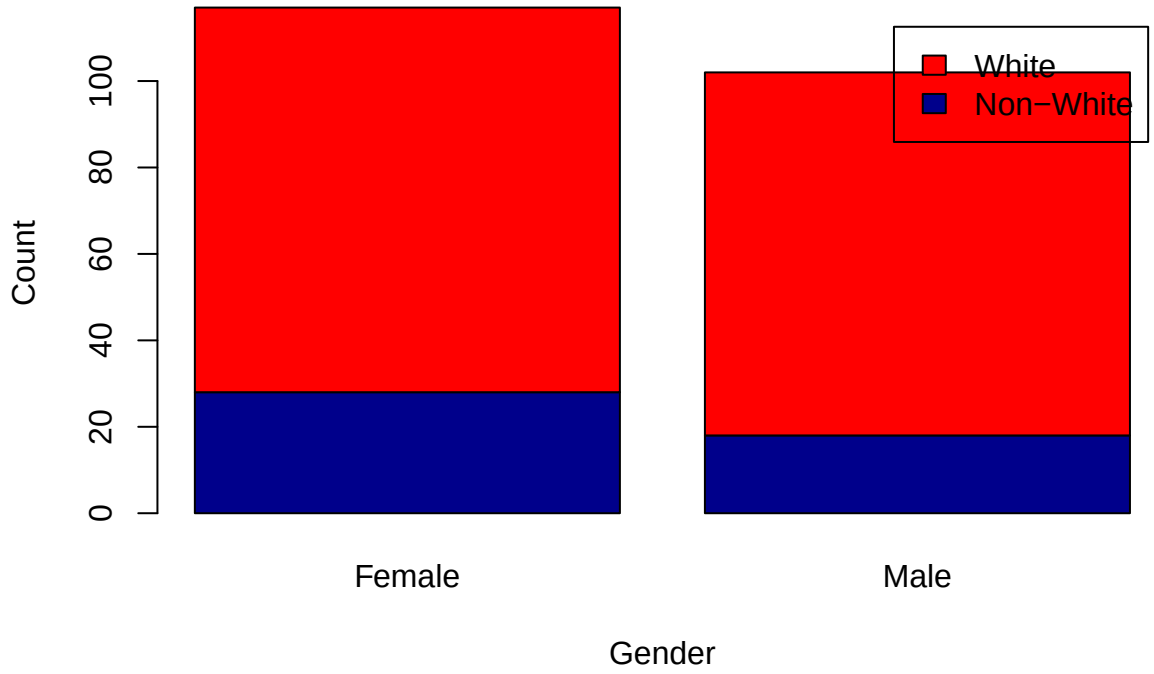
Description of the data and the analysis problem. Provide information where the data was obtained,

The data contains information from a sample of 219 first year students at a midwestern college in 1996. The data has 10 variables:

```
GPA           First-year college GPA on a 0.0 to 4.0 scale
HSGPA         High school GPA on a 0.0 to 4.0 scale
SATV          Verbal/critical reading SAT score
SATM          Math SAT score
Male          1= male, 0= female
HU            Number of credit hours earned in humanities courses in high school
SS            Number of credit hours earned in social science courses in high school
FirstGen      1= student is the first in her or his family to attend college, 0=otherwise
White         1= white students, 0= others
CollegeBound  1=attended a high school where >=50% students intended to go on to college, 0=otherwise
```

The data can be obtained from https://vincentarelbundock.github.io/Rdatasets/doc/Stat2Data/FirstYearGPA.html. Our models use all numerical values (HSGPA, SATV, SATM, HU and SS) to predict the first-year college GPA (GPA). In addition, the hierarchical model uses categorical variables Male and White to group the data into four different groups and predicts the GPA with group-level predictors.

```
counts <- table(White=FirstYearGPA$White, Male=FirstYearGPA$Male)
barplot(counts,
  xlab="Gender", col=c("darkblue","red"),
  legend = c('Non-White', 'White'), ylab = 'Count', names=c('Female','Male'))
```



3 Models

In this project, two different models are utilised: pooled and hierarchical model. Both are are covered in following sections. The mathematical notations, Stan implementations, and stan model runs are included in the sections. The both of the models follow the linear Gaussian model whose expected values are constructed using linear function with variables found in data (HSGPA, SATV, SATM, HU, and SS) and parameters α, β_1, β_2, β_3, β_4, and β_5.

3.1 Pooled model

In pooled model, all the expected values are constructed using the common parameters α and βs with weakly informative priors. All the GPAs have common σ.

Mathematical notation

$$
\begin{aligned}
GPA_i &\sim N(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 \cdot HSGPA_i + \beta_2 \cdot SATV_i + \beta_3 \cdot SATM_i + \beta_4 \cdot HU_i + \beta_5 \cdot SS_i \\
\sigma &\sim N(0, 10) \\
\alpha &\sim N(0, 100) \\
\beta_k &\sim N(0, 100)
\end{aligned}
$$

Stan code

```
writeLines(readLines("pooled.stan"))
```

```
## // Pooled model.
## // Variables: HSGPA, SATM, SATV, HU, SS
##
## data {
##   int<lower=0> N;
##   matrix[N,5] x;
##   vector[N] y;
##
##   real musigma;
##   real sigmasigma;
## }
##
## parameters {
##   real alpha;
##   vector[5] betas;
##   real<lower=0> sigma;
## }
##
## transformed parameters {
##   vector[N] mu;
##   mu = alpha + betas[1]*x[,1] + betas[2]*x[,2] + betas[3]*x[,3]
##        + betas[4]*x[,4] + betas[5]*x[,5];
## }
##
## model {
##   // priors
##   alpha ~ normal(0, musigma);
##   betas ~ normal(0, musigma);
##   sigma ~ normal(0, sigmasigma);
```

```
##
##   // likelihood
##   y ~ normal(mu, sigma);
## }
##
## generated quantities {
##   vector[N] ypred;
##   vector[N] log_lik;
##
##   // Generate predictive distributions for GPA
##   for (i in 1:N)
##     ypred[i] = normal_rng(mu[i], sigma);
##
##   // log likelihoods
##   for (i in 1:N)
##     log_lik[i] = normal_lpdf(y[i] | mu[i], sigma);
## }
```

Running the model

```
data_pooled <- list(N = nrow(FirstYearGPA),
                    x = subset(FirstYearGPA, select = c('HSGPA', 'SATV', 'SATM','HU','SS')),
                    y = FirstYearGPA$GPA,
                    musigma = 100,
                    sigmasigma = 10)
```

```
mod_pooled <- cmdstan_model("pooled.stan")
fit_pooled <- mod_pooled$sample(data_pooled, refresh = 0, seed = 03091900)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 16.6 seconds.
## Chain 2 finished in 16.9 seconds.
## Chain 3 finished in 18.3 seconds.
## Chain 4 finished in 16.1 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 17.0 seconds.
## Total execution time: 68.4 seconds.
```

3.2 Hierarchical model

In Hierarchical model, students have been divided in four different groups: white males, white females, non white males, and non white females. All students in all the groups have own parameters αs and βs which are constructed using common hyperparameters for all the different groups.

Mathematical notation

$$
\begin{aligned}
GPA_{ij} &\sim N(\mu_{ij}, \sigma) \\
\mu_{ij} &= \alpha_j + \beta_{1j} \cdot HSGPA_i + \beta_{2j} \cdot SATV_i + \beta_{3j} \cdot SATM_i + \beta_{4j} \cdot HU_i + \beta_{5j} \cdot SS_i \\
\sigma &\sim N(0, 10) \\
\alpha_j &\sim N(\mu_\alpha, \sigma_\alpha) \\
\beta_{1j} &\sim N(\mu_{\beta_1}, \sigma_{\beta_1}) \\
\beta_{2j} &\sim N(\mu_{\beta_2}, \sigma_{\beta_2}) \\
\beta_{3j} &\sim N(\mu_{\beta_3}, \sigma_{\beta_3}) \\
\beta_{4j} &\sim N(\mu_{\beta_4}, \sigma_{\beta_4}) \\
\beta_{5j} &\sim N(\mu_{\beta_5}, \sigma_{\beta_5}) \\
\mu_\alpha &\sim N(0, 100) \\
\sigma_\alpha &\sim N(0, 10) \\
\mu_{\beta_k} &\sim N(0, 100) \\
\sigma_{\beta_k} &\sim N(0, 10)
\end{aligned}
$$

Stan code

```
writeLines(readLines("hierarchical.stan"))
```

```
## // Hierarchical model.
## // Betas in following order: HSGPA, SATM, SATV, HU, SS
## // Alpha: intercept
## data {
##   int<lower=0> N1;
##   int<lower=0> N2;
##   int<lower=0> N3;
##   int<lower=0> N4;
##   matrix[N1,5] x1;
##   matrix[N2,5] x2;
##   matrix[N3,5] x3;
##   matrix[N4,5] x4;
##   vector[N1] y1;
##   vector[N2] y2;
##   vector[N3] y3;
##   vector[N4] y4;
##
##   real musigma;
##   real sigmasigma;
## }
##
## parameters {
##   // parameters
##   real alpha1;
##   real alpha2;
##   real alpha3;
##   real alpha4;
##   vector[5] betas1;
##   vector[5] betas2;
##   vector[5] betas3;
##   vector[5] betas4;
##   real<lower=0> sigma;
##
```

```
##   // hyperparameters
##   real pmualpha;
##   real<lower=0> psalpha;
##   vector[5] pmubetas;
##   vector<lower=0>[5] psbetas;
## }
##
## transformed parameters {
##   vector[N1] mu1 = alpha1 + betas1[1]*x1[,1] + betas1[2]*x1[,2] + betas1[3]*x1[,3]
##                    + betas1[4]*x1[,4] + betas1[5]*x1[,5];
##   vector[N2] mu2 = alpha2 + betas2[1]*x2[,1] + betas2[2]*x2[,2] + betas2[3]*x2[,3]
##                    + betas2[4]*x2[,4] + betas2[5]*x2[,5];
##   vector[N3] mu3 = alpha3 + betas3[1]*x3[,1] + betas3[2]*x3[,2] + betas3[3]*x3[,3]
##                    + betas3[4]*x3[,4] + betas3[5]*x3[,5];
##   vector[N4] mu4 = alpha4 + betas4[1]*x4[,1] + betas4[2]*x4[,2] + betas4[3]*x4[,3]
##                    + betas4[4]*x4[,4] + betas4[5]*x4[,5];
## }
##
## model {
##   // hyperpriors
##   pmualpha ~ normal(0, musigma);
##   psalpha ~ normal(0, sigmasigma);
##   for (i in 1:5){
##     pmubetas[i] ~ normal(0, musigma);
##     psbetas[i] ~ normal(0, sigmasigma);
##   }
##
##   // priors
##   alpha1 ~ normal(pmualpha, psalpha);
##   alpha2 ~ normal(pmualpha, psalpha);
##   alpha3 ~ normal(pmualpha, psalpha);
##   alpha4 ~ normal(pmualpha, psalpha);
##   betas1 ~ normal(pmubetas, psbetas);
##   betas2 ~ normal(pmubetas, psbetas);
##   betas3 ~ normal(pmubetas, psbetas);
##   betas4 ~ normal(pmubetas, psbetas);
##   sigma ~ normal(0, sigmasigma);
##
##   // likelihoods
##   y1 ~ normal(mu1, sigma);
##   y2 ~ normal(mu2, sigma);
##   y3 ~ normal(mu3, sigma);
##   y4 ~ normal(mu4, sigma);
## }
##
## generated quantities{
##   vector[N1] ypred1;
##   vector[N2] ypred2;
##   vector[N3] ypred3;
##   vector[N4] ypred4;
##   vector[N1+N2+N3+N4] log_lik;
##
##   // Generate predictive distributions for GPA
##   for (i in 1:N1)
```

```
##     ypred1[i] = normal_rng(mu1[i], sigma);
##   for (i in 1:N2)
##     ypred2[i] = normal_rng(mu2[i], sigma);
##   for (i in 1:N3)
##     ypred3[i] = normal_rng(mu3[i], sigma);
##   for (i in 1:N4)
##     ypred4[i] = normal_rng(mu4[i], sigma);
##
##   // log likelihoods
##   for (i in 1:N1)
##     log_lik[i] = normal_lpdf(y1[i] | mu1[i], sigma);
##   for (i in 1:N2)
##     log_lik[N1+i] = normal_lpdf(y2[i] | mu2[i], sigma);
##   for (i in 1:N3)
##     log_lik[N1+N2+i] = normal_lpdf(y3[i] | mu3[i], sigma);
##   for (i in 1:N4)
##     log_lik[N1+N2+N3+i] = normal_lpdf(y4[i] | mu4[i], sigma);
## }
```

**Running the model

```
male_white <- FirstYearGPA[FirstYearGPA$Male==1 & FirstYearGPA$White==1,]
male_non_white <- FirstYearGPA[FirstYearGPA$Male==1 & FirstYearGPA$White==0,]
female_white <- FirstYearGPA[FirstYearGPA$Male==0 & FirstYearGPA$White==1,]
female_non_white <- FirstYearGPA[FirstYearGPA$Male==0 & FirstYearGPA$White==0,]

data_hierarchical <- list(N1 = nrow(male_white),
                          N2 = nrow(male_non_white),
                          N3 = nrow(female_white),
                          N4 = nrow(female_non_white),
                          x1 = subset(male_white, select = c('HSGPA', 'SATV', 'SATM','HU','SS')),
                          x2 = subset(male_non_white, select = c('HSGPA', 'SATV', 'SATM','HU','SS')),
                          x3 = subset(female_white, select = c('HSGPA', 'SATV', 'SATM','HU','SS')),
                          x4 = subset(female_non_white, select = c('HSGPA', 'SATV', 'SATM','HU','SS')),
                          y1 = male_white$GPA,
                          y2 = male_non_white$GPA,
                          y3 = female_white$GPA,
                          y4 = female_non_white$GPA,
                          musigma = 100,
                          sigmasigma = 10)
```

```
mod_hierarchical <- cmdstan_model("hierarchical.stan")
fit_hierarchical <-mod_hierarchical$sample(data_hierarchical, refresh = 0,
                                           seed = 03091900)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 32.1 seconds.
## Chain 2 finished in 29.9 seconds.
## Chain 3 finished in 33.9 seconds.
## Chain 4 finished in 35.5 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 32.8 seconds.
## Total execution time: 131.8 seconds.
```

```
## Warning: 218 of 4000 (5.0%) transitions ended with a divergence.
## See https://mc-stan.org/misc/warnings for details.
```

```
## Warning: 4 of 4000 (0.0%) transitions hit the maximum treedepth limit of 10.
## See https://mc-stan.org/misc/warnings for details.
```

Weakly informative normal priors were used in both models. N(0,10) was the prior distribution for σ in both models. In the pooled model, N(0,100) was the prior for both α and all β_k. The hierarchical model had similar priors: N(0,100) for all means μ_α and μ_{β_k}, and N(0,10) for all standard deviations σ_α and σ_{β_k}. We thought these priors reasonable for multiple reasons. Firstly, they all center on 0, which was thought wisest as we have no information whether the intercept α or the slopes β_k should be positive or negative. Secondly, the standard deviations, 100 for the means and 10 for the standard deviations, were considered large enough to produce wide enough (but not too wide) prior distributions.

As can be seen from the above code lines, default values were used for running the MCMC chains. That is, 4 chains of 2000 iterations were run, and the first 1000 iterations of each chain were considered warm-up.

4 Analysis and Results

4.1 Converge diagnostics

4.1.1 Pooled model

```
summary_pooled <- fit_pooled$summary()
sum(summary_pooled$rhat > 1.01)
```

```
## [1] 0
```

```
sum(summary_pooled$rhat < 0.99)
```

```
## [1] 0
```

```
mean(fit_pooled$summary()$ess_tail)
```

```
## [1] 3171.745
```

```
sum(fit_pooled$summary()$ess_tail<400)
```

```
## [1] 0
```

```
mean(fit_pooled$summary()$ess_bulk)
```

```
## [1] 3207.508
```

```
sum(fit_pooled$summary()$ess_bulk<400)
```

```
## [1] 0
```

None of the $\hat{R}$-values for the pooled model exceeds 1.01 which indicates that there are no problems with the convergence of the set of simulated chains. The effective sample size (ESS) is over 400 (recommended threshold with four parallel chains, 100 per chain) for all of the ess_bulk and ess_tail values which indicates that the $\hat{R}$-estimate can be trusted to make decisions about convergence and the quality of the chains. In addition, the model does not observe divergence.

4.1.2 Hierarchical model

```
sum(fit_hierarchical$summary()$rhat > 1.01)
```

```
## [1] 0
```

```
sum(fit_hierarchical$summary()$rhat < 0.99)
```

```
## [1] 0
```

```
mean(fit_hierarchical$summary()$ess_tail)
```

```
## [1] 2984.828
```

```
sum(fit_hierarchical$summary()$ess_tail<400)
```

```
## [1] 2
```

```
mean(fit_hierarchical$summary()$ess_bulk)
```

```
## [1] 2725.984
```

```
sum(fit_hierarchical$summary()$ess_bulk<400)
```

```
## [1] 4
```

For the hierarchical model a few $\hat{R}$-values exceed 1.01 and a few ESS-values are below 400. The model gives divergence of approximately 5%. These facts can indicate that there might be some problems with the convergence. On the other hand, these values do not differ significantly from the desired values the probability of having problems with convergence is small.

4.2 Posterior predictive checks

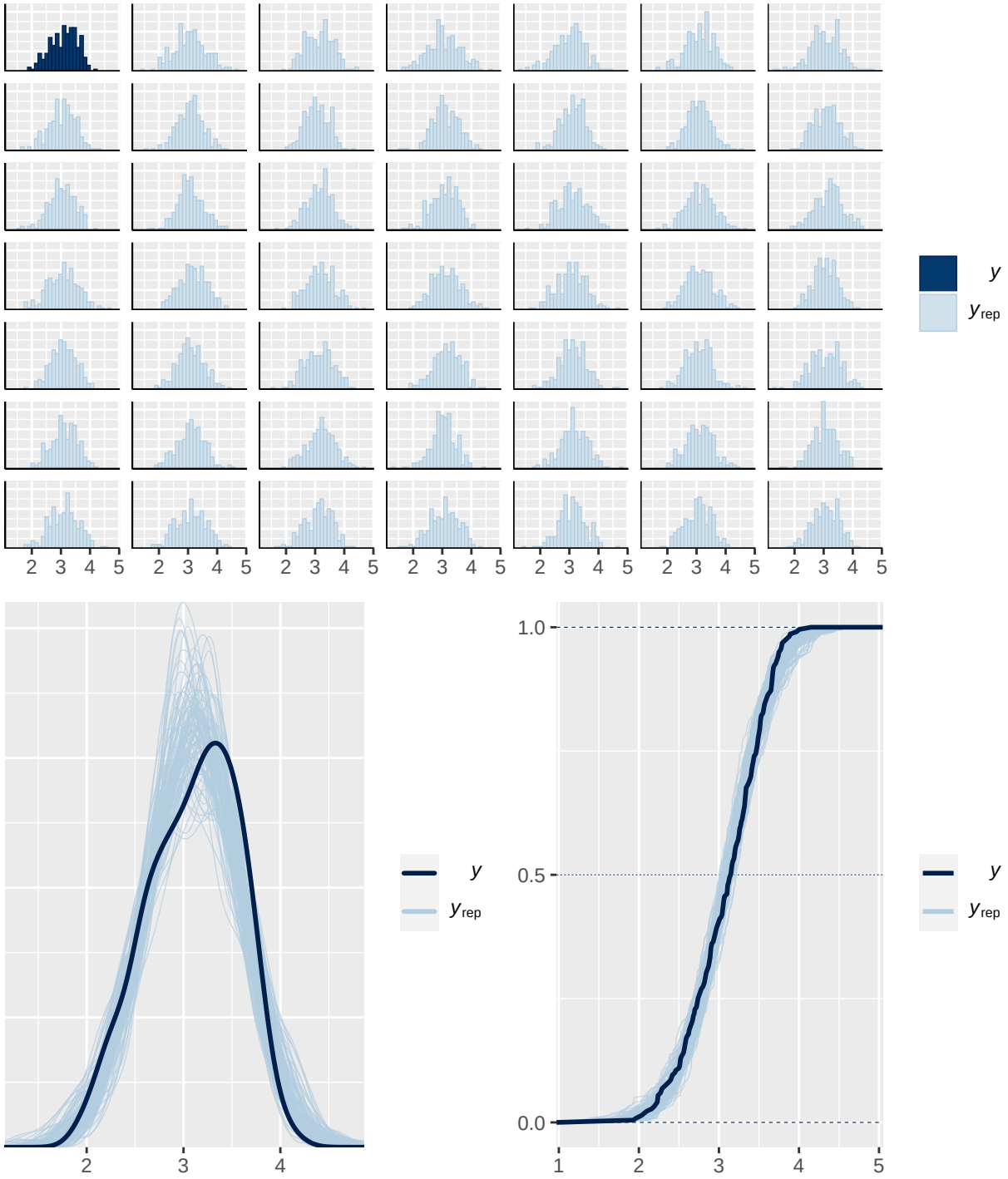
48 histograms of replications of the data (created based on the posterior predictive distributions) are shown for all the models. These replication histograms are compared to histogram showing the distribution of the original data to investigate, how good predictions fitted model can make i.e., it is check how good the model is. In addition, there is over layed distributions and empirical cumulative distribution (ecdf) functions of 100 replications in the two with the original distribution and ecdf.

4.2.1 Pooled model

```
y <- FirstYearGPA$GPA
ypred <- fit_pooled$draws("ypred", format = "matrix")

grid.arrange( ppc_hist(y, ypred[1:48,]),
              ppc_dens_overlay(y, ypred[1:100,]),
              ppc_ecdf_overlay(y, ypred[1:100,]),
              layout_matrix = matrix(c(1,2,1,3), nrow = 2), nrow=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

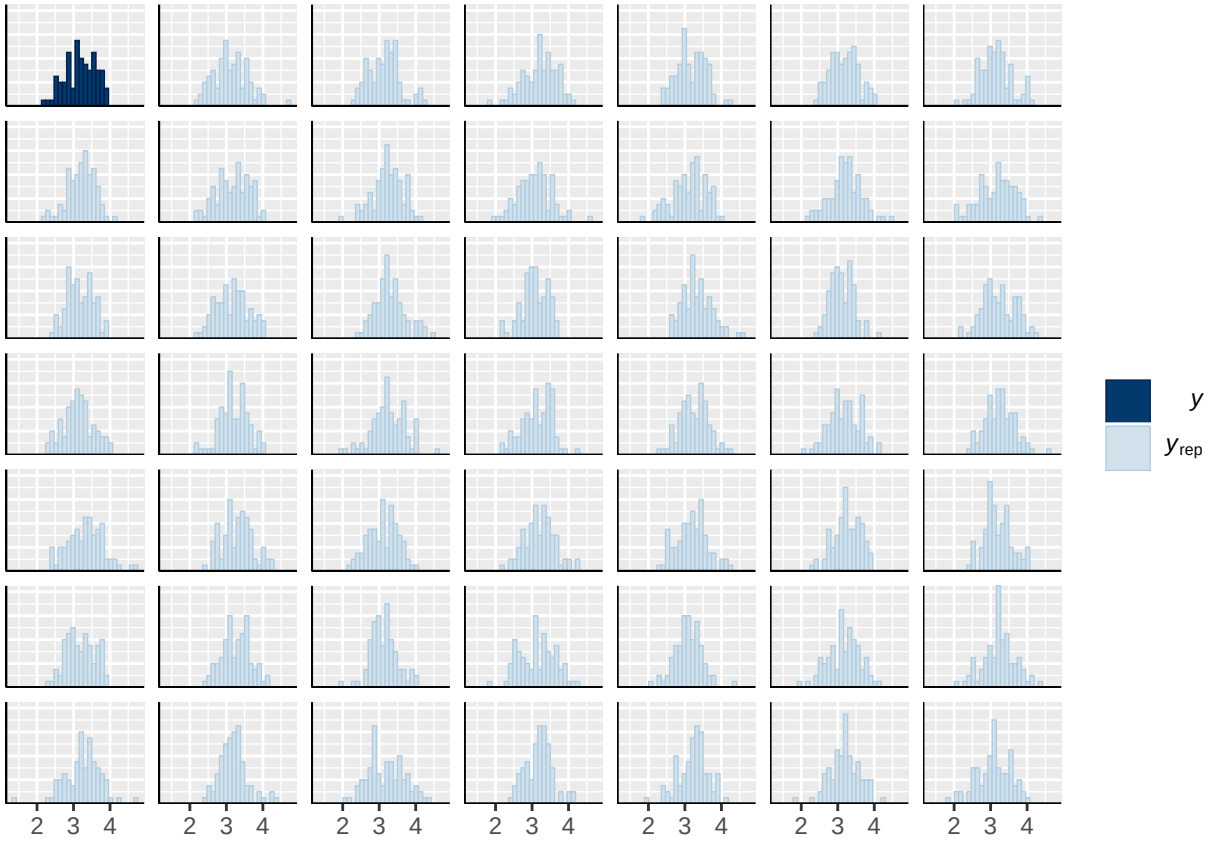


4.2.2 Hierarchical model

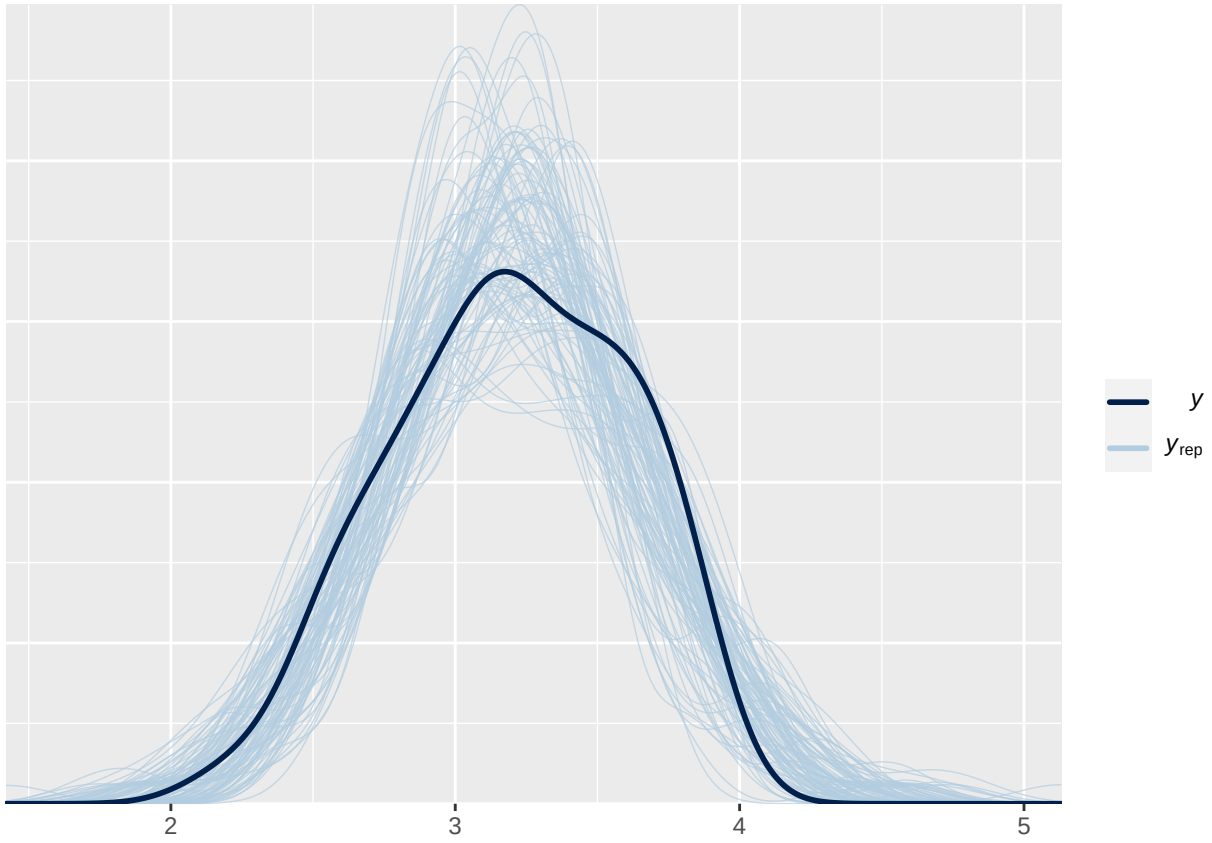
```
y1 <- data_hierarchical$y1
ypred1 <- fit_hierarchical$draws("ypred1", format = "matrix")

ppc_hist(y1, ypred1[1:48,])
```

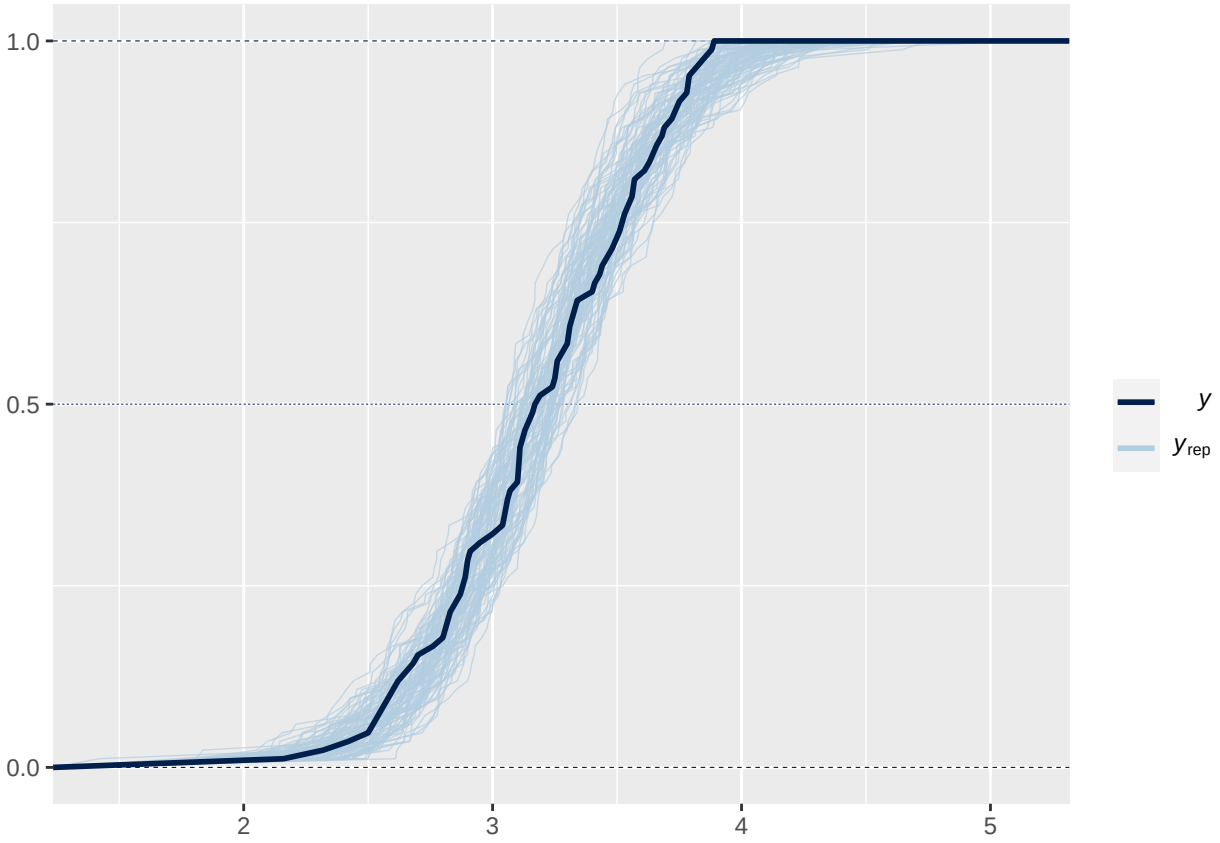
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ppc_dens_overlay(y1, ypred1[1:100,])
```



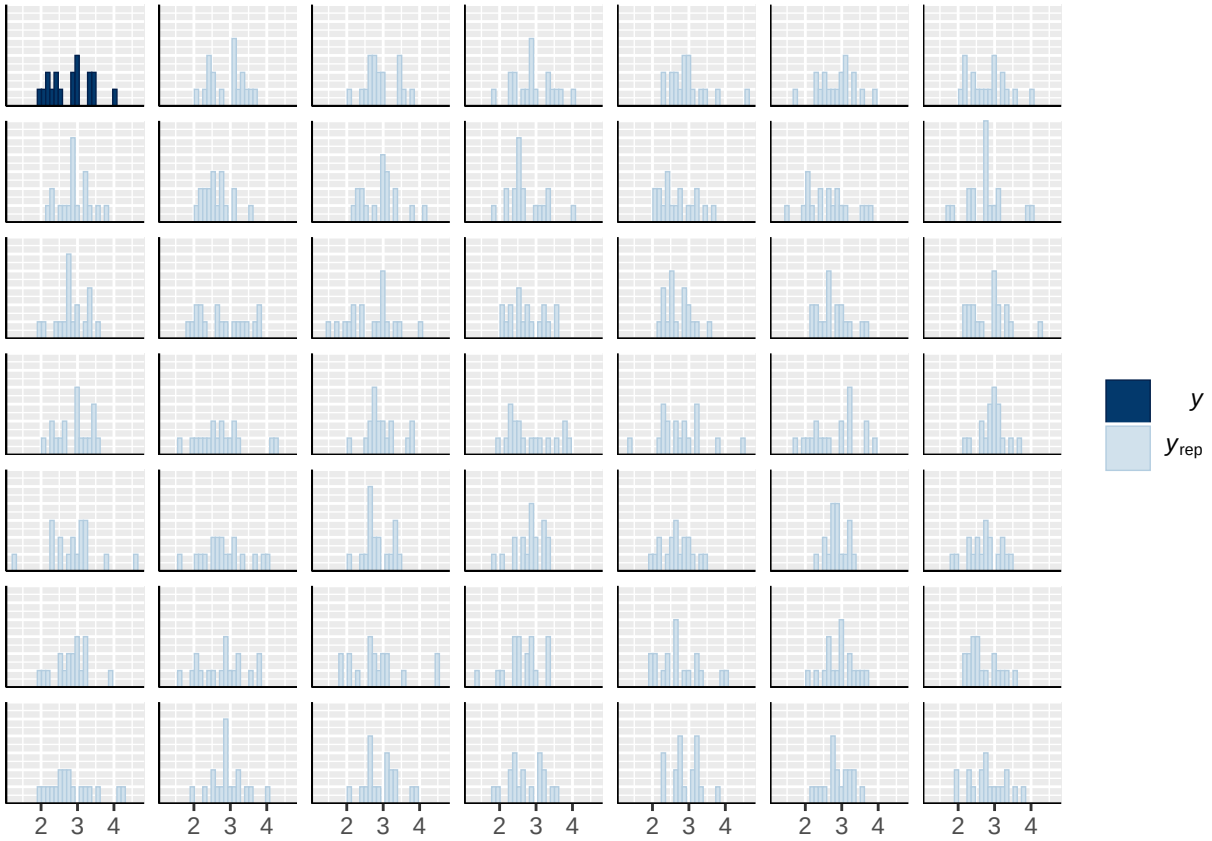

```
ppc_ecdf_overlay(y1, ypred1[1:100,])
```



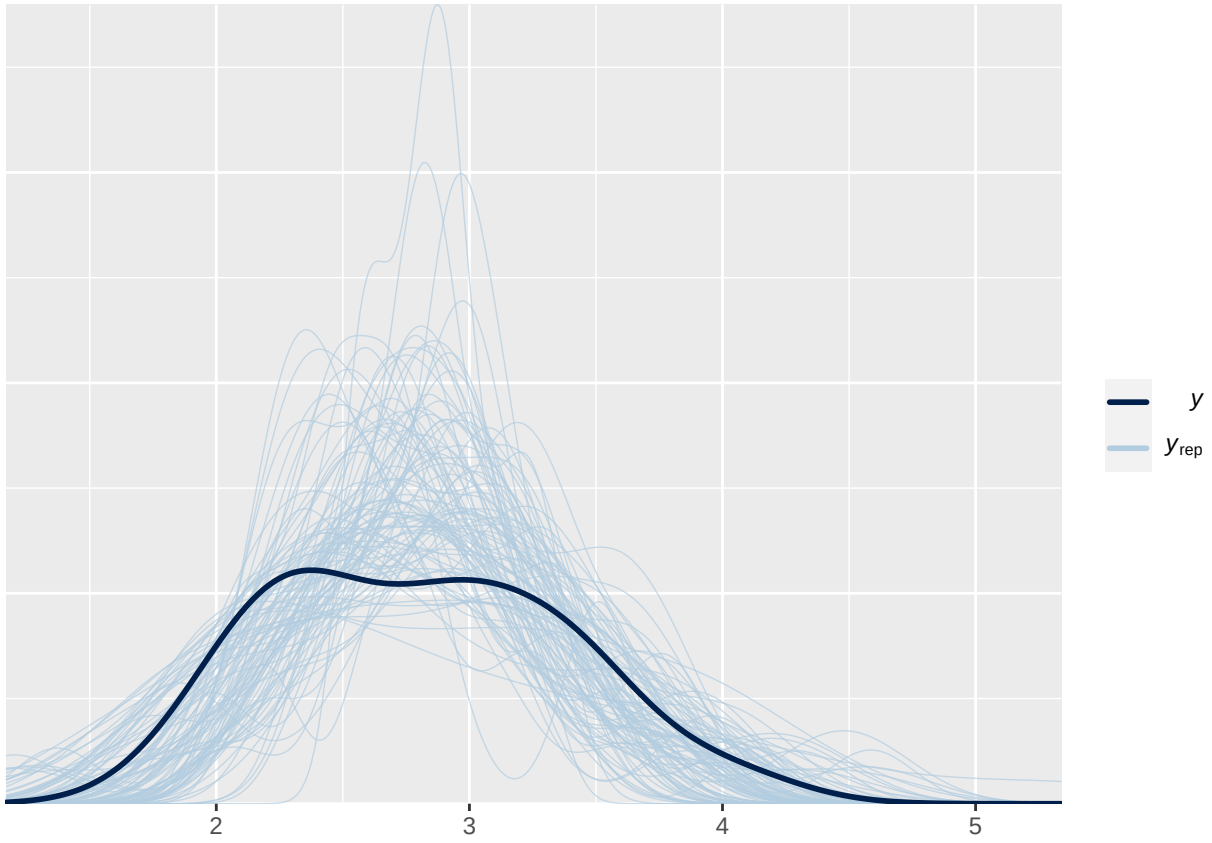

```
y2 <- data_hierarchical$y2
ypred2 <- fit_hierarchical$draws("ypred2", format = "matrix")

ppc_hist(y2, ypred2[1:48,])
```

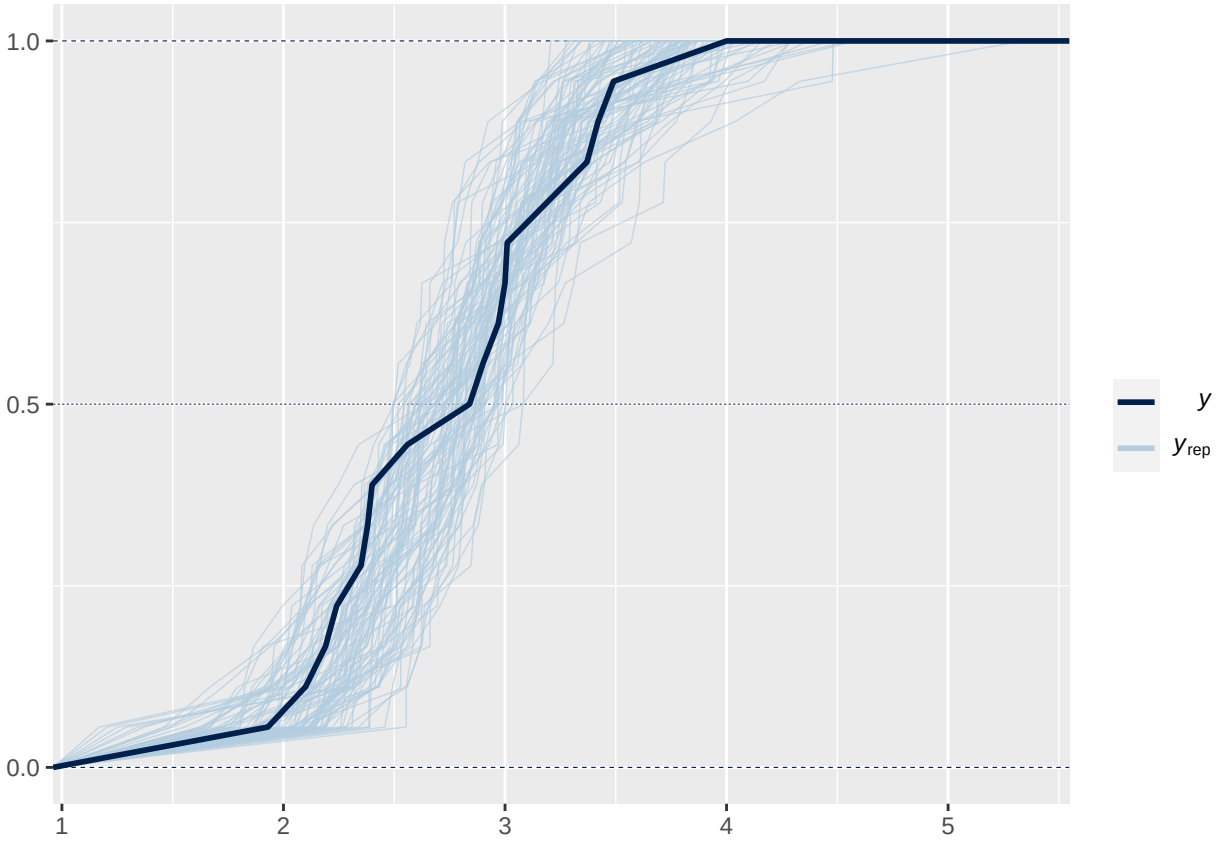
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ppc_dens_overlay(y2, ypred2[1:100,])
```



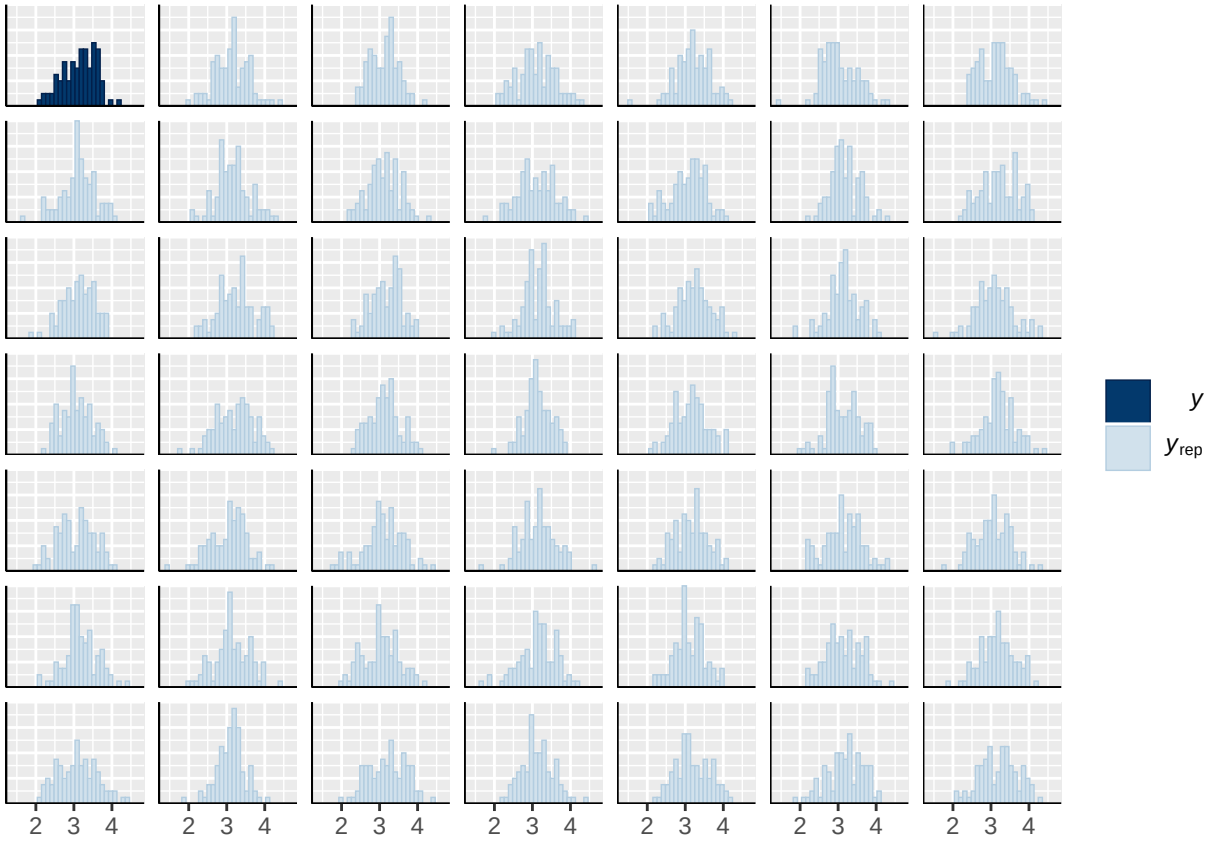

```
ppc_ecdf_overlay(y2, ypred2[1:100,])
```



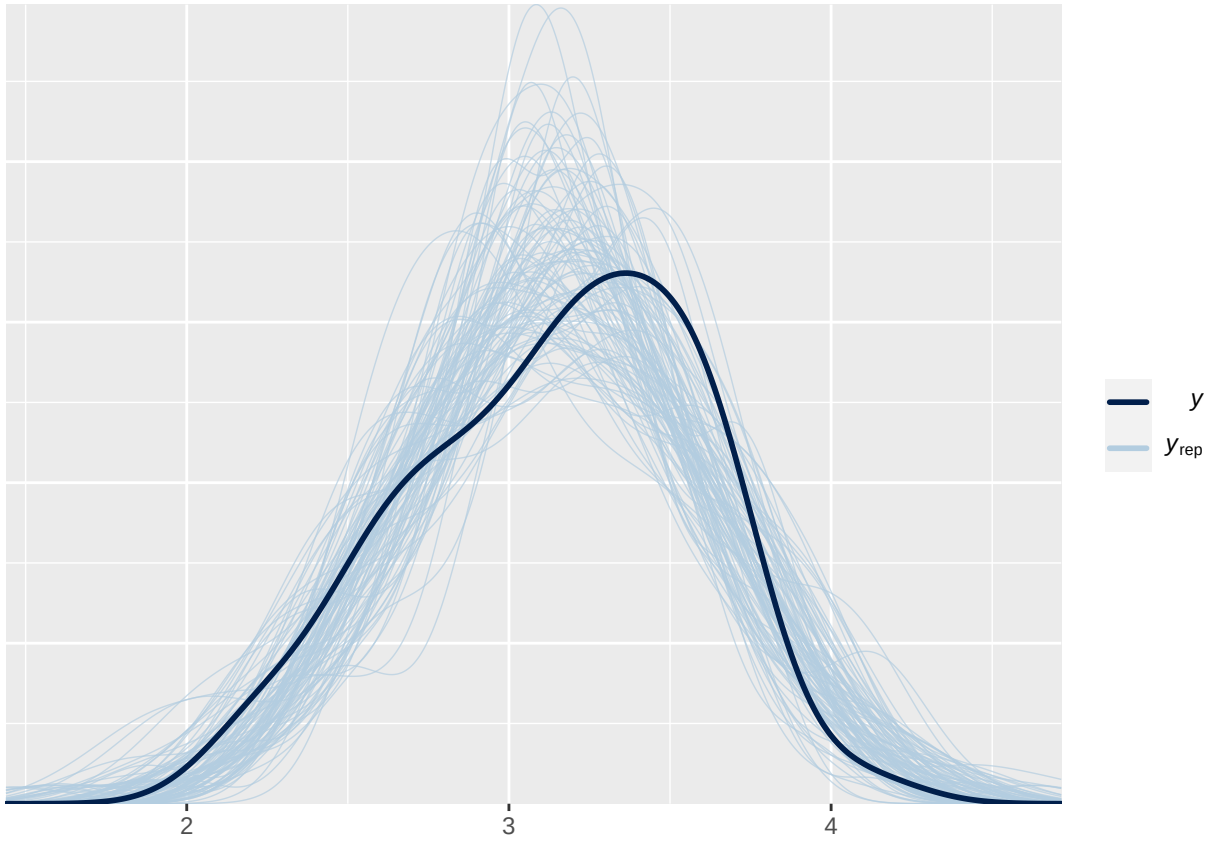

```
y3 <- data_hierarchical$y3
ypred3 <- fit_hierarchical$draws("ypred3", format = "matrix")

ppc_hist(y3, ypred3[1:48,])
```

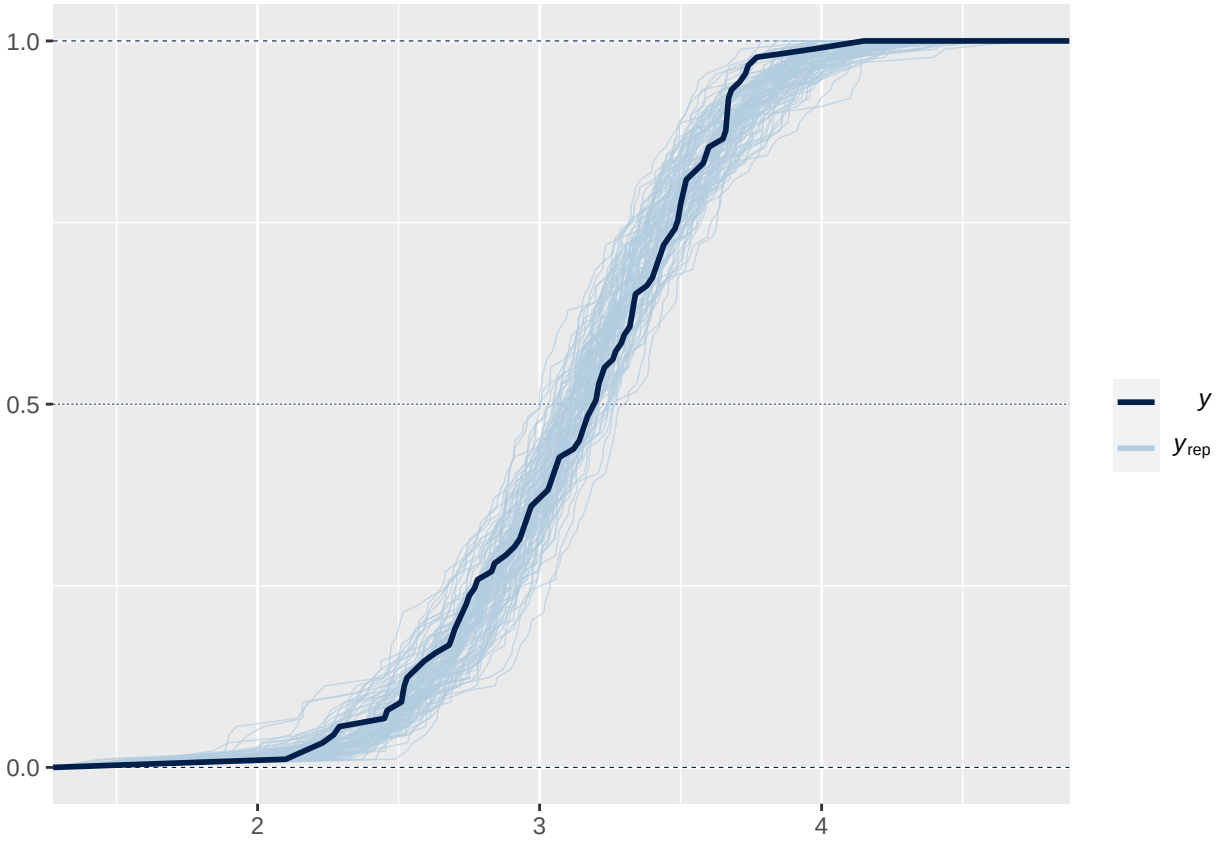
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ppc_dens_overlay(y3, ypred3[1:100,])
```



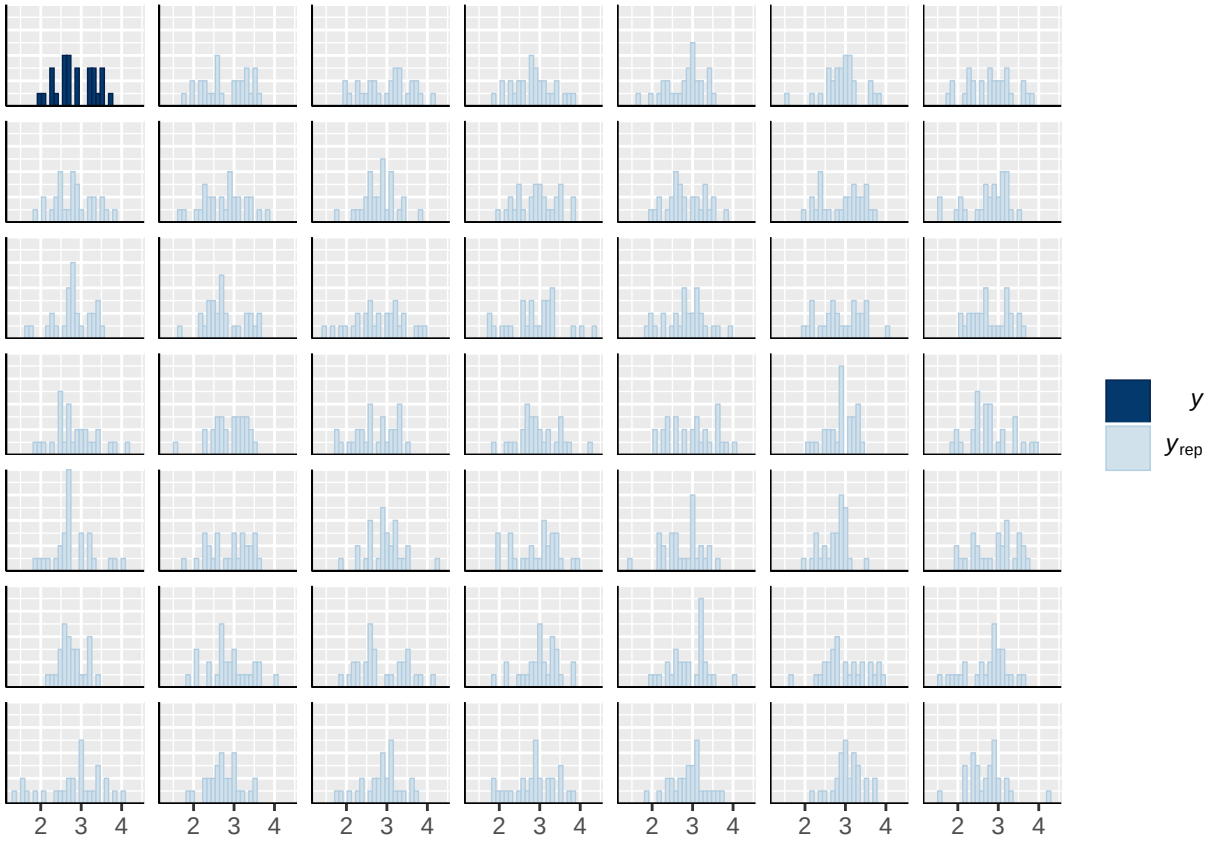

```
ppc_ecdf_overlay(y3, ypred3[1:100,])
```



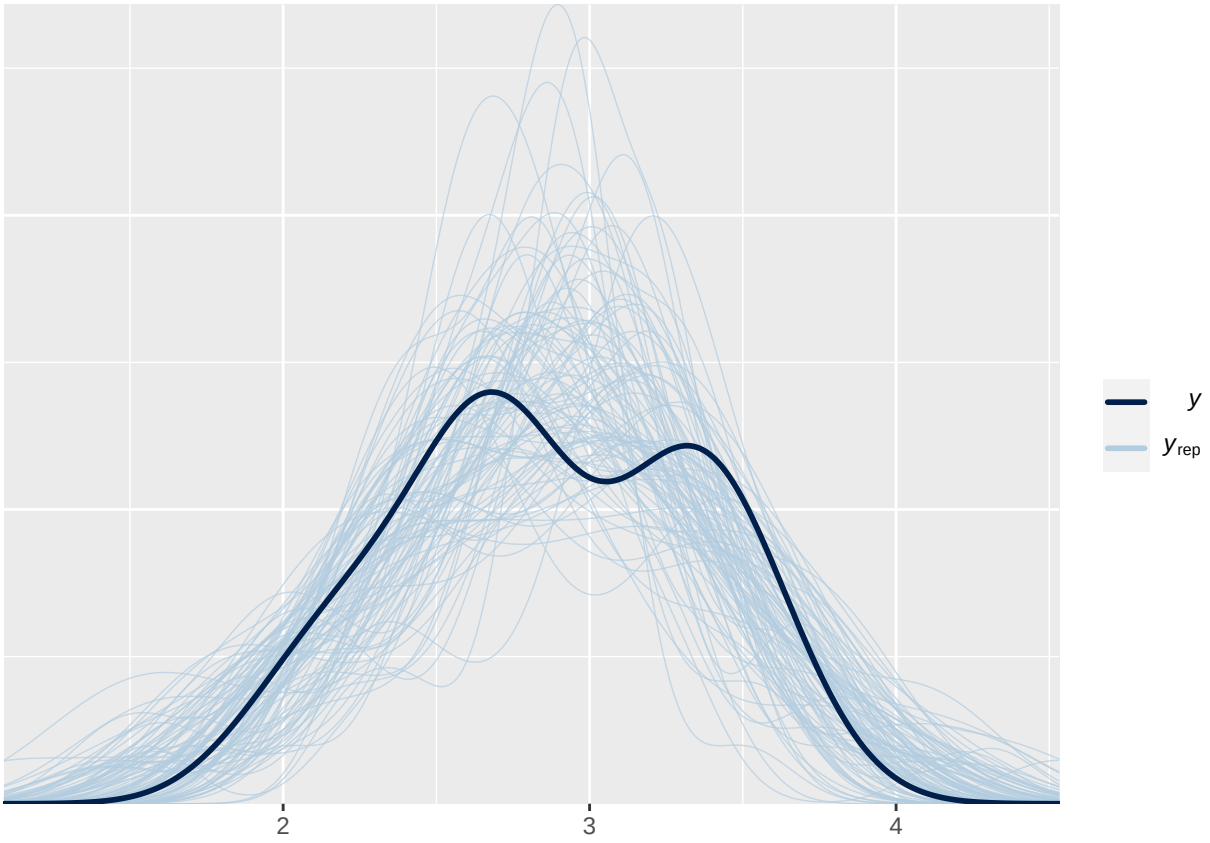

```
y4 <- data_hierarchical$y4
ypred4 <- fit_hierarchical$draws("ypred4", format = "matrix")

ppc_hist(y4, ypred4[1:48,])
```

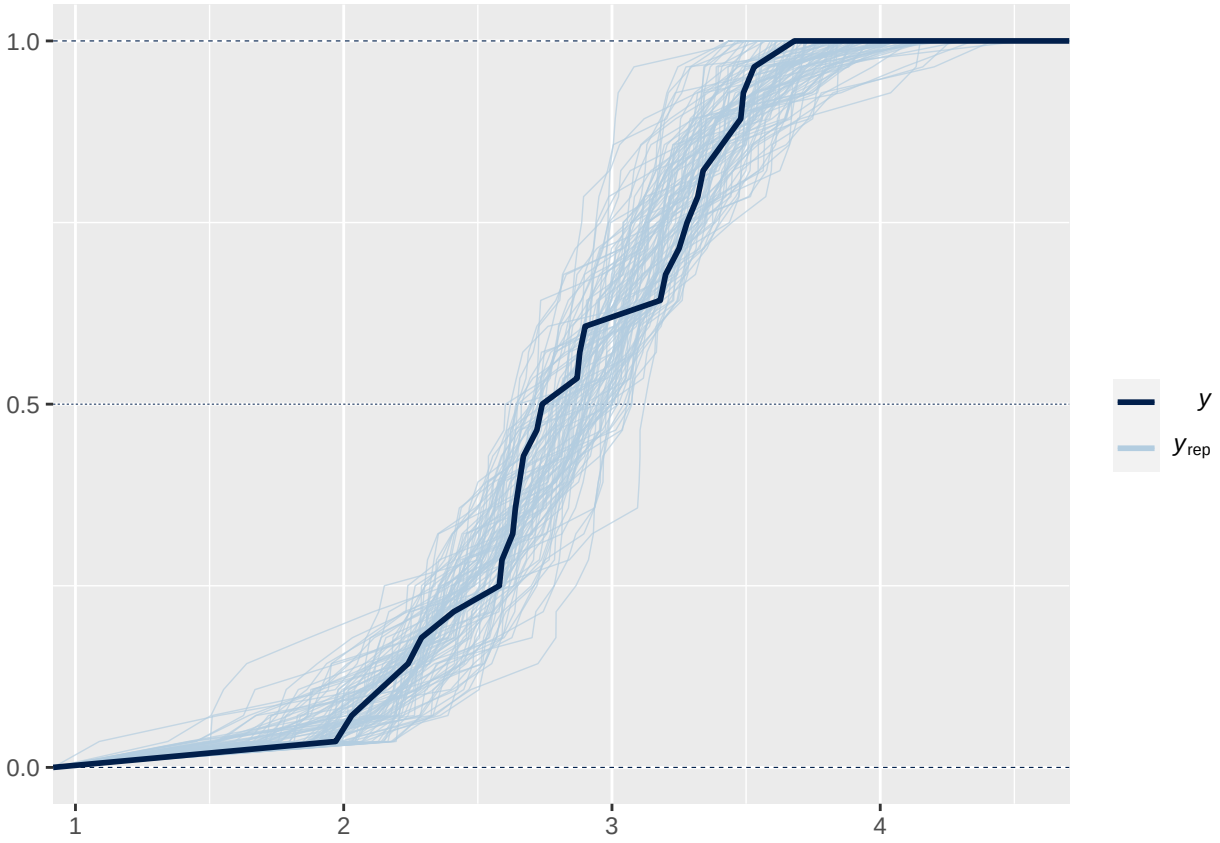
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ppc_dens_overlay(y4, ypred4[1:100,])
```




```
ppc_ecdf_overlay(y4, ypred4[1:100,])
```



4.3 Prior sensitivity analysis

Original priors: Normal(0, 100) + Normal(0, 10)

Priors to test: Normal(0, 1000) + Normal(0,10) Normal(0, 50) + Normal(0,10) Normal(0, 10) + Normal(0,10) Normal(0, 1) + Normal(0,10) and vice versa Normal (0,100) + Normal (0,100) Normal (0,100) + Normal (0,50) Normal (0,100) + Normal (0,1)

4.3.1 Pooled model

```
# musigmas <- c(100, 1000, 50, 10, 1, 100, 100, 100)
# sigmasigmas <- c(10, 10, 10, 10, 10, 100, 50, 1)
# mus <- matrix(nrow = 6, ncol = length(musigmas))
# sigmas <- matrix(nrow = 6, ncol = length(musigmas))
#
# for (i in 1:length(musigmas)) {
#   data_pooled_sens <- list(N = nrow(FirstYearGPA),
#                            x = subset(FirstYearGPA, select = c('HSGPA', 'SATV',
#                                                                'SATM','HU','SS')),
#                            y = FirstYearGPA$GPA,
#                            musigma = musigmas[i],
#                            sigmasigma = sigmasigmas[i])
#   fit_pooled_sens <- mod_pooled$sample(data_pooled, refresh = 0,
#                                        seed = 03091900)
#   summary_sens <- fit_pooled_sens$summary()
#   mus[,i] <- summary_sens$mean[2:7]
```

```
#   sigmas[,i] <- summary_sens$sd[2:7]
# }
```

```
# dimnames <- list(c("alpha", "beta1", "beta2", "beta3", "beta4", "beta5"),
#                  c("1", "2", "3", "4", "5", "6", "7", "8"))
# dimnames(mus) <- dimnames
# dimnames(sigmas) <- dimnames
#
# as.data.frame(mus)
# as.data.frame(sigmas)
```

Regardless of the used priors, all posteriors for α and β_k are the same. Therefore our pooled model is not sensitive to changes in the priors, as long as wide enough ones are utilised.

Posteriors plotted:

```
# par(mfrow = c(3,2))
# x <- seq(-2, 2, length = 1000)
# y <- dnorm(x, mean = summary_pooled$mean[2], sd = summary_pooled$sd[2])
# plot(x, y, type = "l", xlab = "alpha")
# for (i in 1:5) {
#   x <- seq(-1, 1, length = 1000)
#   y <- dnorm(x, mean = summary_pooled$mean[2+i], sd = summary_pooled$sd[2+i])
#   plot(x, y, type = "l", xlab = paste0("beta", i))
# }
```

4.3.2 Hierarchical model

```
# musigmas <- c(100, 1000, 50, 10, 1, 100, 100, 100)
# sigmasigmas <- c(10, 10, 10, 10, 10, 100, 50, 1)
# mus <- matrix(nrow = 24, ncol = length(musigmas))
# sigmas <- matrix(nrow = 24, ncol = length(musigmas))
#
# for (i in 1:length(musigmas)) {
#   data_hierarchical_sens <- list(N1 = nrow(male_white),
#                                  N2 = nrow(male_non_white),
#                                  N3 = nrow(female_white),
#                                  N4 = nrow(female_non_white),
#                                  x1 = subset(male_white, select = c('HSGPA', 'SATV', 'SATM','HU','SS'
#                                  x2 = subset(male_non_white, select = c('HSGPA', 'SATV', 'SATM','HU',
#                                  x3 = subset(female_white, select = c('HSGPA', 'SATV', 'SATM','HU','S
#                                  x4 = subset(female_non_white, select = c('HSGPA', 'SATV', 'SATM','HU
#                                  y1 = male_white$GPA,
#                                  y2 = male_non_white$GPA,
#                                  y3 = female_white$GPA,
#                                  y4 = female_non_white$GPA,
#                                  musigma = musigmas[i],
#                                  sigmasigma = sigmasigmas[i])
#   fit_hierarchical_sens <- mod_hierarchical$sample(data_hierarchical_sens, refresh = 0,
#                                                    seed = 03091900)
#   summary_sens <- fit_hierarchical_sens$summary()
#   mus[,i] <- summary_sens$mean[2:25]
#   sigmas[,i] <- summary_sens$sd[2:25]
# }
```

```
# dimnames <- list(c("alpha1", "alpha2", "alpha3", "alpha4",
#                    "beta1_1", "beta2_1", "beta3_1", "beta4_1", "beta5_1",
#                    "beta1_2", "beta2_2", "beta3_2", "beta4_2", "beta5_2",
#                    "beta1_3", "beta2_3", "beta3_3", "beta4_3", "beta5_3",
#                    "beta1_4", "beta2_4", "beta3_4", "beta4_4", "beta5_4"),
#                  c("1", "2", "3", "4", "5", "6", "7", "8"))
# dimnames(mus) <- dimnames
# dimnames(sigmas) <- dimnames
#
# as.data.frame(mus)
# as.data.frame(sigmas)
```

```
# # Each group's parameters get its own plots (4 3x2 plots in total)
# for (i in 1:4) {
#   par(mfrow = c(3,2))
#   x <- seq(-2, 2, length = 1000)
#   y <- dnorm(x, mean = summary_hierarchical$mean[1+i], sd = summary_hierarchical$sd[1+i])
#   plot(x, y, type = "l", xlab = "alpha")
#   for (j in 1:5) {
#     x <- seq(-1, 1, length = 1000)
#     y <- dnorm(x, mean = summary_hierarchical$mean[5+(i-1)*5+j], sd = summary_hierarchical$sd[5+(i-1)
#     plot(x, y, type = "l", xlab = paste0("beta", i))
#   }
# }
```

4.4 Model comparison

```
# dimnames <- list(c("alpha", "beta1", "beta2", "beta3", "beta4", "beta5"),
#                  c("pooled", "h. group 1", "h. group 2",
#                    "h. group 3", "h. group 4"))
# post_table <- matrix(nrow = 6, ncol = 5, dimnames = dimnames)
#
# # Fill 1st column
# for (i in 1:6) {
#   mean <- summary_pooled$mean[1+i]
#   sd <- summary_pooled$sd[1+i]
#   post_table[i,1] <- paste0("N(", round(mean,3), ", ", round(sd,3), ")")
# }
# # Fill other columns
# # first alphas
# for (i in 1:4) {
#   mean <- summary_hierarchical$mean[1+i]
#   sd <- summary_hierarchical$sd[1+i]
#   post_table[1, 1+i] <- paste0("N(", round(mean,3), ", ", round(sd,3), ")")
# }
# # then betas
# for (i in 1:4) {
#   for(j in 1:5) {
#     mean <- summary_hierarchical$mean[5+(i-1)*5+j]
#     sd <- summary_hierarchical$sd[5+(i-1)*5+j]
#     post_table[j+1,i+1] <- paste0("N(", round(mean,3), ", ", round(sd,3), ")")
#   }
# }
```

```
#
# as.data.frame(post_table)
```

```
# log_lik_pooled <- fit_pooled$draws("log_lik", format = "matrix")
# log_lik_hierarchical <- fit_hierarchical$draws("log_lik", format = "matrix")
#
# loo_pooled <- loo(log_lik_pooled)
# loo_hierarchical <- loo(log_lik_hierarchical)
#
# loo_compare(list("pooled" = loo_pooled, "hierarchical" = loo_hierarchical))
```

5 Discussion

6 Conclusion

7 Self-reflection