homework 10

Homework 10: IRT model

1

In this homework I fit the 2PL.

a. Simulate fake data and check that the model recovers the parameters. Feel free to simplify the model as necessary.

```
# Load R packages
library(rstan)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.18.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
library(ggplot2)
print_file <- function(file) {
  cat(paste(readLines(file), "\n", sep=""), sep="")
}
```

```
# Set paramters for the simulated data
set.seed(1234)
I <- 20                # number of items
J <- 1000              # number of responses
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(n = I, mu = mu, Sigma = 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))
```

```
# Fit model to simulated data
IRT = stan_model("IRT.stan")
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on '/
## Users/yi/Desktop/study/subjects/bayesian-data-analysis/homework/homework
## 15/IRT.stan'
```

```
fake_IRT <- sampling(IRT,data=data_list)
```

```
# model test
library(bayesplot)
```

```
## This is bayesplot version 1.6.0
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
```

```
##    * Does _not_ affect other ggplot2 plots
```

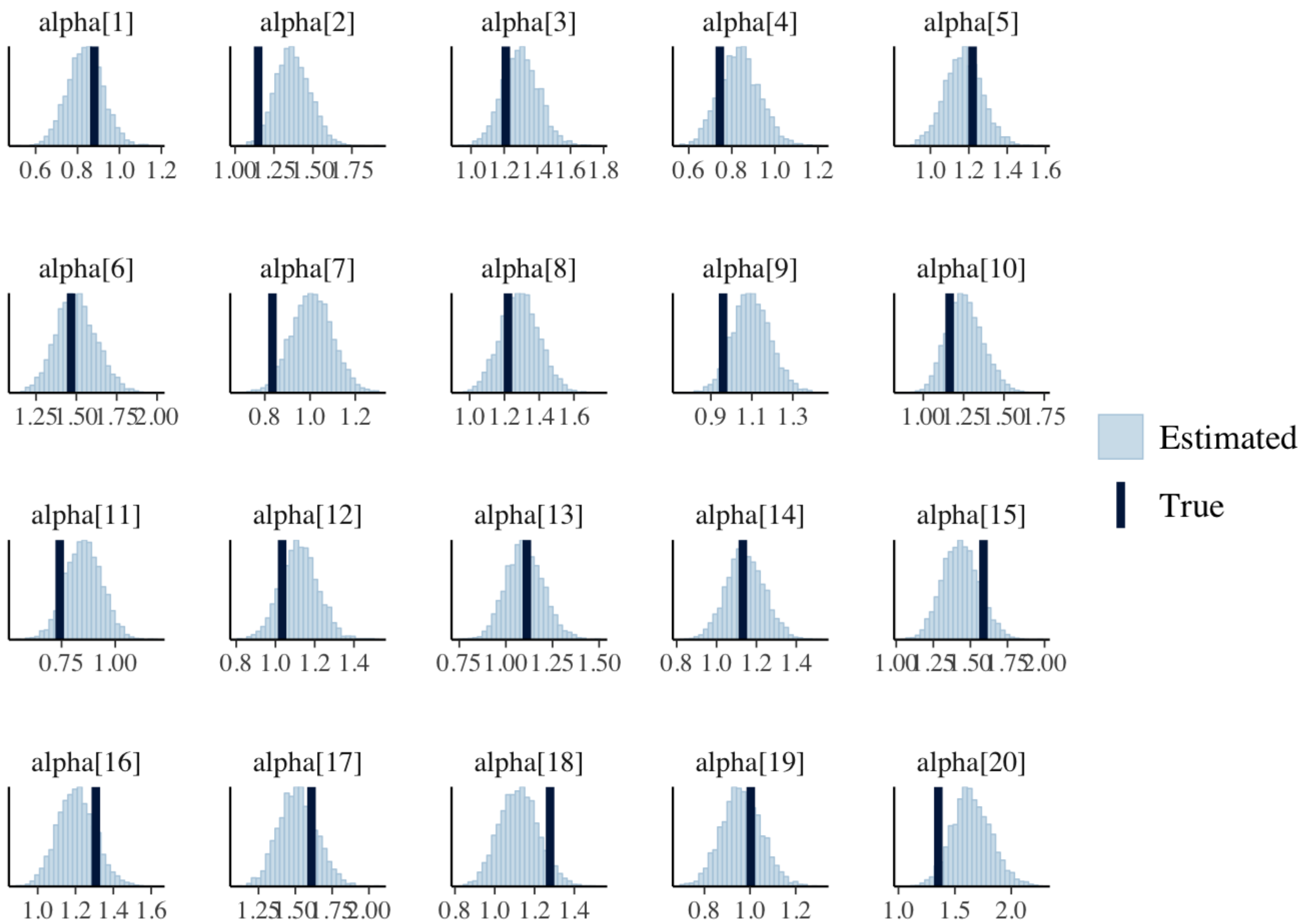
```
##    * See ?bayesplot_theme_set for details on theme setting
```

```
posterior_alpha <- as.matrix(fake_IRT, pars = c('alpha'))
head(posterior_alpha)
```

```
##           parameters
## iterations  alpha[1] alpha[2] alpha[3]  alpha[4] alpha[5] alpha[6]
##       [1,] 0.8242957 1.613327 1.424335 0.6724541 1.234423 1.408473
##       [2,] 0.7717985 1.363289 1.400940 0.7338133 1.158134 1.598503
##       [3,] 0.8455605 1.306456 1.195125 0.8831016 1.106948 1.419934
##       [4,] 0.7955181 1.271752 1.206127 0.9048248 1.191852 1.333402
##       [5,] 0.8032647 1.486208 1.361135 0.8838822 1.265209 1.467136
##       [6,] 0.8443718 1.385928 1.322220 0.7570462 1.208784 1.405698
##           parameters
## iterations  alpha[7] alpha[8]  alpha[9] alpha[10] alpha[11] alpha[12]
##       [1,] 0.9008229 1.250073 1.0093305 1.1685969 0.7859595 0.9109610
##       [2,] 1.0147610 1.239297 1.0595536 0.9353319 0.9769691 0.9231468
##       [3,] 0.9083034 1.533176 0.9544471 1.3783175 0.8992974 1.0392498
##       [4,] 1.0143731 1.528333 0.9487974 1.2680769 0.8621976 0.9468456
##       [5,] 1.0117092 1.237041 1.1483498 1.2027381 0.9410875 1.3467374
##       [6,] 1.0486704 1.362020 1.1812916 1.2387379 1.0332940 1.3499110
##           parameters
## iterations alpha[13] alpha[14] alpha[15] alpha[16] alpha[17] alpha[18]
##       [1,]  1.024120  1.117100  1.432212  1.299396  1.504053  1.064938
##       [2,]  1.053457  1.143184  1.430326  1.139096  1.545625  1.070320
##       [3,]  1.325101  1.138503  1.409349  1.343055  1.364930  1.184027
##       [4,]  1.179943  1.230240  1.474051  1.226147  1.525973  1.186424
##       [5,]  1.175216  1.097464  1.471681  1.149002  1.499581  1.129984
##       [6,]  1.237216  1.093368  1.566035  1.140321  1.470869  1.188178
##           parameters
## iterations alpha[19] alpha[20]
##       [1,] 0.9628372  1.674171
##       [2,] 0.8966865  1.411311
##       [3,] 0.9682262  1.711622
##       [4,] 0.8876810  1.922787
##       [5,] 1.0915007  1.533869
##       [6,] 1.0112454  1.660168
```

```
true_alpha <- c(alpha)
mcmc_recover_hist(posterior_alpha, true = true_alpha)
```

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

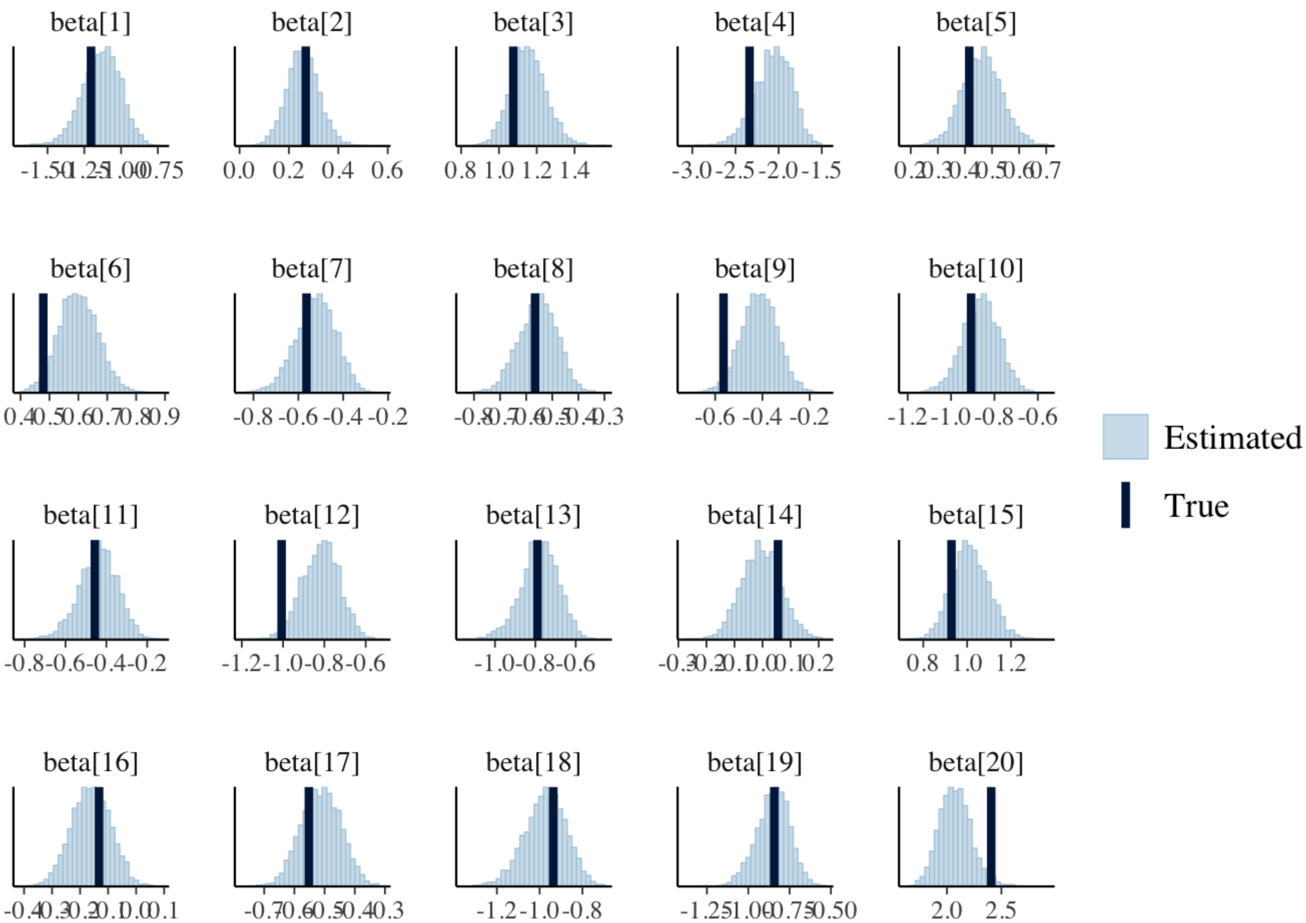



```
# model test
posterior_beta <- as.matrix(fake_IRT, pars = c('beta'))
head(posterior_beta)
```

```
##           parameters
## iterations    beta[1]   beta[2]  beta[3]   beta[4]   beta[5]   beta[6]
##       [1,] -1.0747403 0.2764037 1.156681 -2.410512 0.5375348 0.6887222
##       [2,] -1.0949269 0.2877234 1.156301 -2.122604 0.4590911 0.6967038
##       [3,] -1.1231568 0.3602539 1.209284 -1.870905 0.4920451 0.6908050
##       [4,] -1.0621579 0.3350946 1.339495 -1.886931 0.5566543 0.6284345
##       [5,] -0.9095598 0.2773544 1.166920 -2.022915 0.4201265 0.7080175
##       [6,] -1.0766909 0.2998581 1.051807 -2.249284 0.4708894 0.6690584
##           parameters
## iterations    beta[7]    beta[8]    beta[9]   beta[10]   beta[11]
##       [1,] -0.5654547 -0.4715072 -0.3268491 -0.8261498 -0.3209381
##       [2,] -0.3724661 -0.4996897 -0.2602371 -1.0809529 -0.2901944
##       [3,] -0.5247365 -0.4318078 -0.4380658 -0.5693846 -0.4234454
##       [4,] -0.3758421 -0.4602482 -0.3754820 -0.7373449 -0.4212615
##       [5,] -0.5157223 -0.4144501 -0.3521367 -0.9450449 -0.2984245
##       [6,] -0.5944380 -0.4621827 -0.3089052 -0.8349957 -0.4453349
##           parameters
## iterations   beta[12]   beta[13]     beta[14]  beta[15]    beta[16]
##       [1,] -0.9364730 -0.7146686  0.115171736 1.1551342 -0.11360323
##       [2,] -0.9031479 -0.7893872  0.028486380 1.0547701 -0.06766223
##       [3,] -0.7667380 -0.6553190  0.049498673 0.9565877 -0.13283700
##       [4,] -0.8019649 -0.6437900 -0.001068566 1.0868324 -0.16696878
##       [5,] -0.6122038 -0.7510494  0.102562362 1.0244325 -0.03784640
##       [6,] -0.6019074 -0.7305482  0.085407506 1.0305373 -0.16608471
##           parameters
## iterations   beta[17]   beta[18]   beta[19] beta[20]
##       [1,] -0.5538578 -0.9298062 -0.7321173 2.165123
##       [2,] -0.3989190 -1.0097051 -0.8406378 2.392901
##       [3,] -0.5049367 -0.8662441 -0.8623246 2.007222
##       [4,] -0.4993479 -0.8625280 -0.7773088 1.931860
##       [5,] -0.3841699 -0.9151974 -0.7365530 2.224972
##       [6,] -0.4569253 -0.8622123 -0.7787064 2.019179
```

```
true_beta <- c(beta)
mcmc_recover_hist(posterior_beta, true = true_beta)
```

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

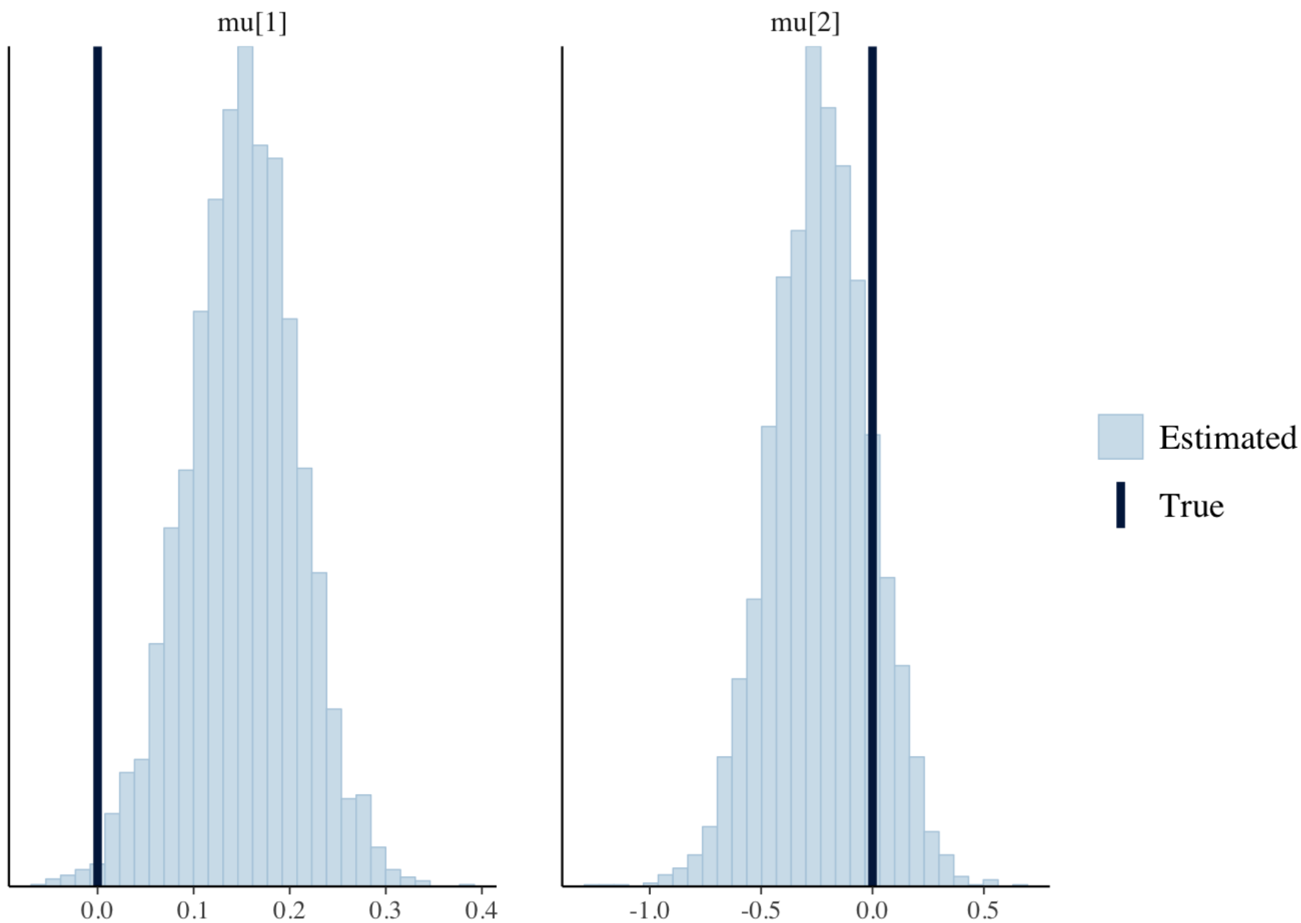



```
# model test
posterior_mu <- as.matrix(fake_IRT, pars = c('mu'))
head(posterior_mu)
```

```
##             parameters
## iterations      mu[1]       mu[2]
##       [1,] 0.1206949 -0.23037438
##       [2,] 0.1019927 -0.21076691
##       [3,] 0.1554032 -0.14692162
##       [4,] 0.1694209 -0.32599790
##       [5,] 0.2207804 -0.22239505
##       [6,] 0.2349292 -0.07286787
```

```
true_mu <- c(mu)
mcmc_recover_hist(posterior_mu, true = true_mu)
```

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

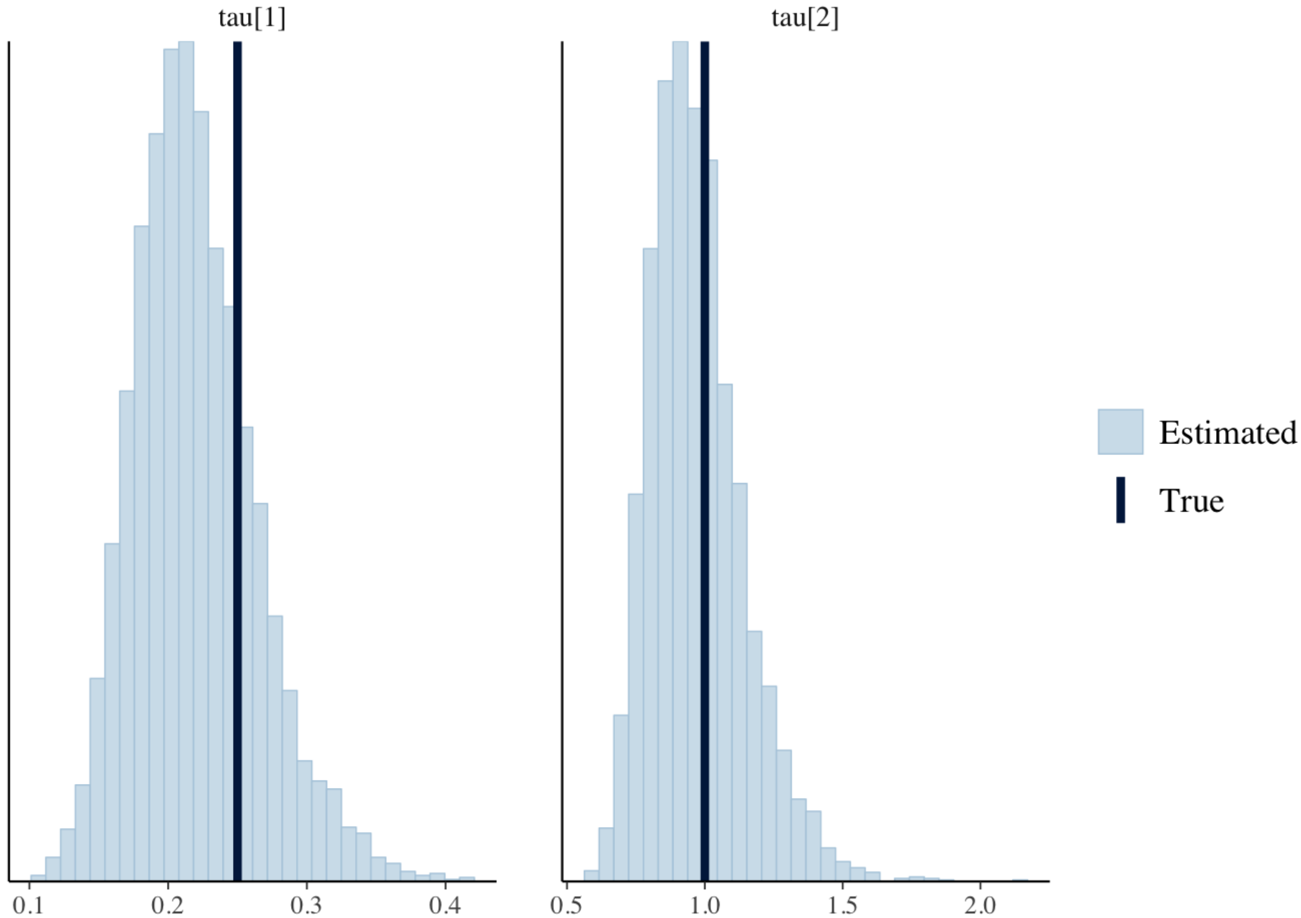



```
# model test
posterior_tau <- as.matrix(fake_IRT, pars = c('tau'))
head(posterior_tau)
```

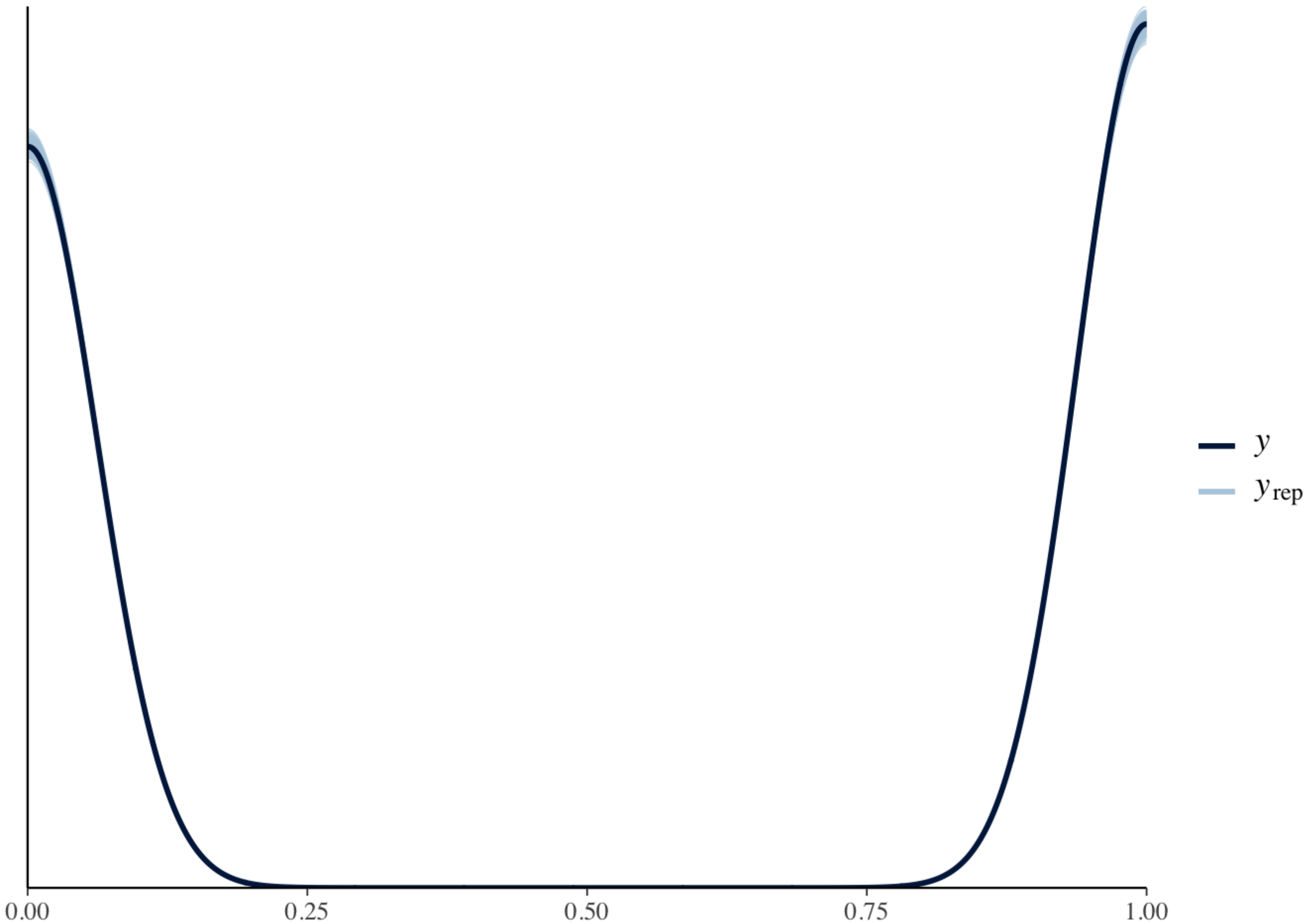
```
##           parameters
## iterations    tau[1]    tau[2]
##       [1,] 0.2215442 0.7495497
##       [2,] 0.2133678 0.9408208
##       [3,] 0.2107596 1.0394565
##       [4,] 0.2205099 0.9124711
##       [5,] 0.1515492 0.8423531
##       [6,] 0.1658208 0.7972126
```

```
true_tau <- c(tau)
mcmc_recover_hist(posterior_tau, true = true_tau)
```

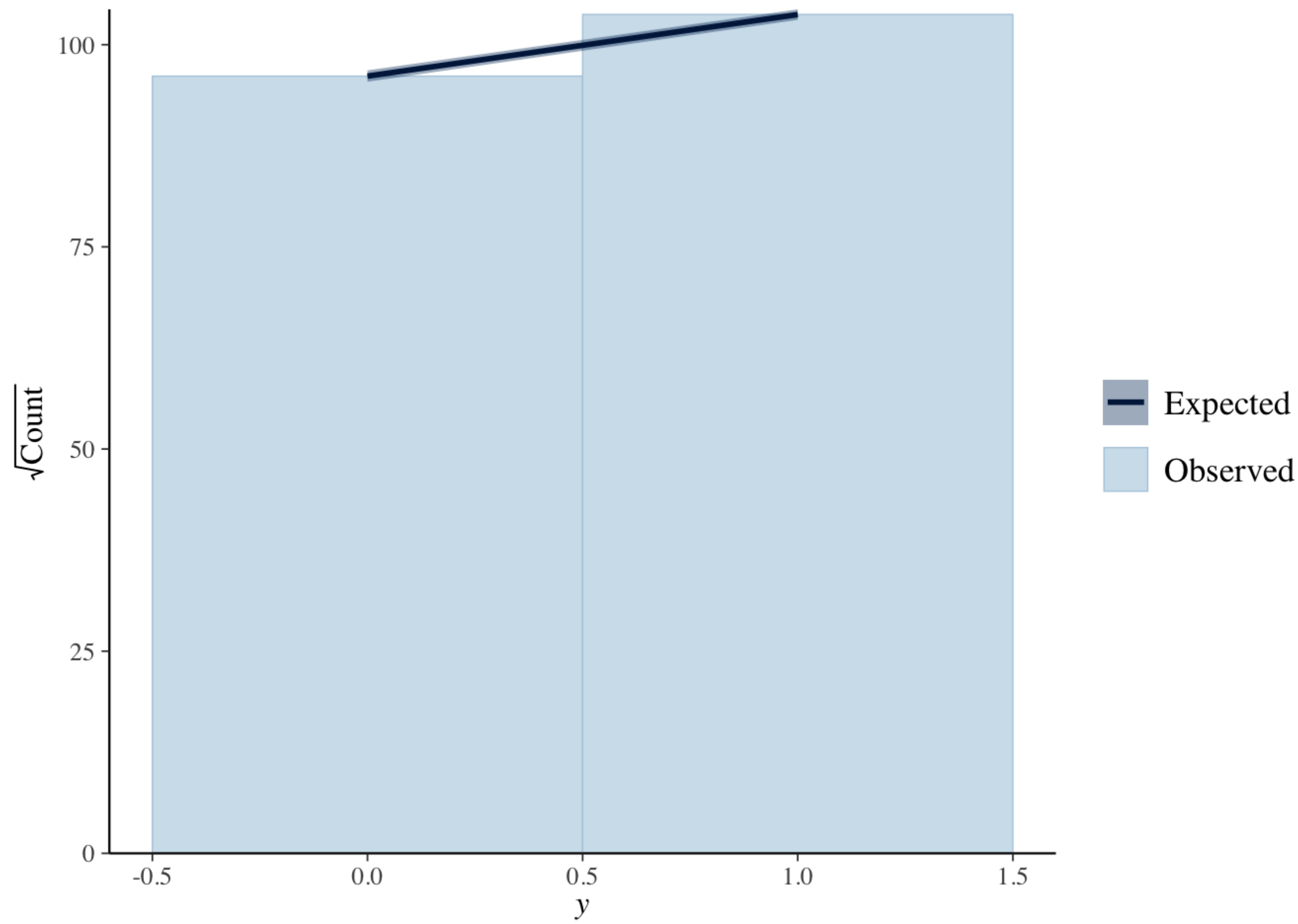
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
y_rep <- as.matrix(fake_IRT, pars = "y_rep")
ppc_dens_overlay(y = data_list$y, yrep = y_rep[1:200, ])
```




```
ppc_rootogram(data_list$y, yrep = y_rep)
```



In summary，the not all parameter fit the model well.

However, as we can see the predictive check and rootogram al very well. Thus, we can say the model is acceptable.

b. In two or three sentences, discuss the strengths and weaknesses of the model. How might the model be expanded?

1. Once θ is specified, the scale of β and α is identified. Thus, there is no need to set the prior for θ. Otherwise, this would leed to a biased estimation even the prior infromation is very week.

2.

a. Fit the model to the real data and perform model checking and/or validation (Chapters 6 and 7 of BDA).

```
# Use data and scoring function from the mirt package
library(mirt)
```

```
## Loading required package: stats4
```

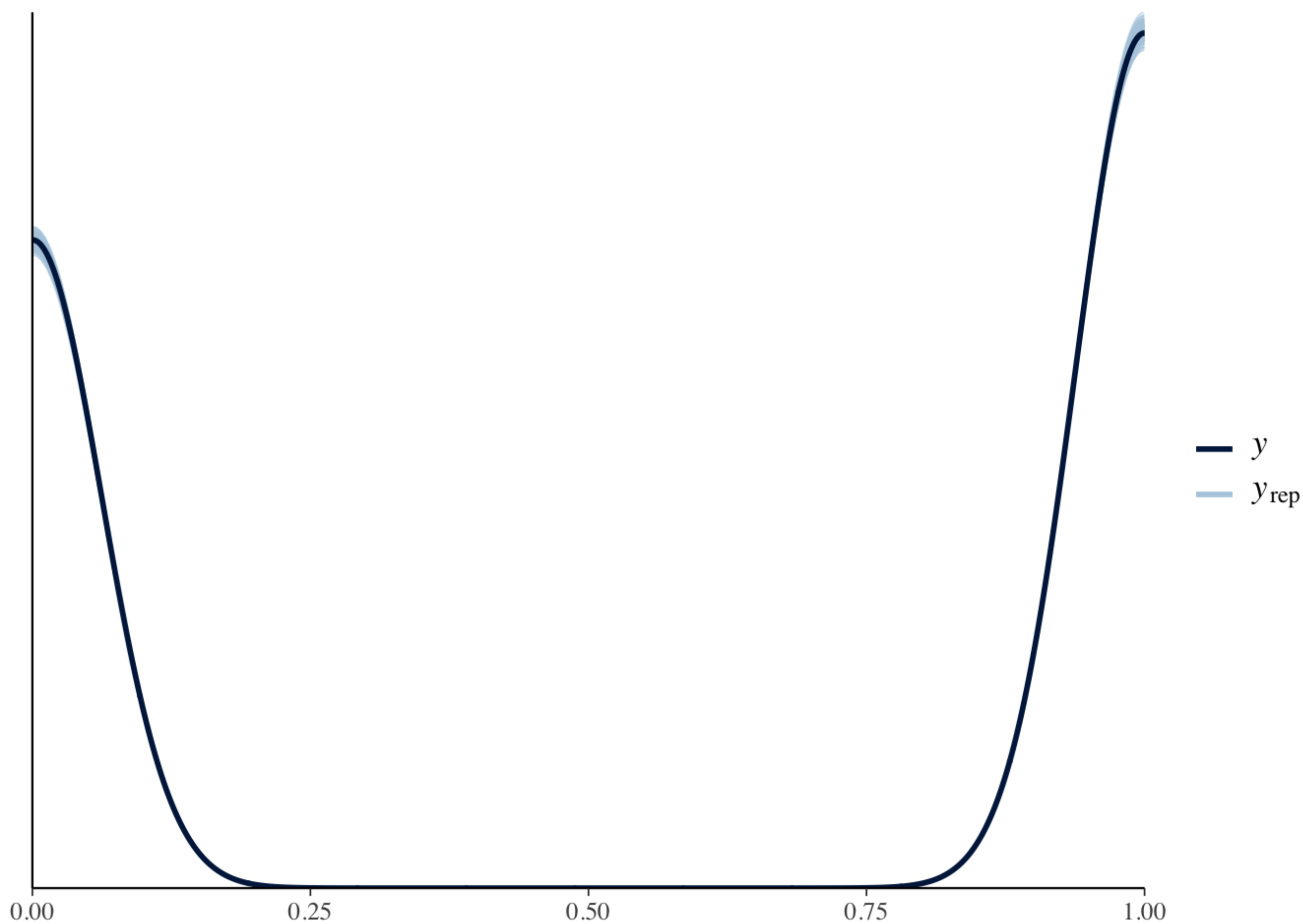
```
## Loading required package: lattice
```

```
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))
# 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))
model <- sampling(IRT,data=sat_list)
print(model,pars=c('alpha','beta'))
```

```
## Inference for Stan model: IRT.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha[1]    0.79    0.00 0.11  0.57  0.71  0.79  0.87  1.03  4089    1
## alpha[2]    1.45    0.00 0.17  1.14  1.33  1.44  1.56  1.80  3760    1
## alpha[3]    1.04    0.00 0.13  0.80  0.95  1.04  1.13  1.31  4798    1
## alpha[4]    0.59    0.00 0.10  0.41  0.52  0.59  0.65  0.78  4569    1
## alpha[5]    0.97    0.00 0.12  0.74  0.89  0.97  1.05  1.23  4595    1
## alpha[6]    1.10    0.00 0.15  0.82  1.00  1.10  1.20  1.41  4716    1
## alpha[7]    1.00    0.00 0.14  0.74  0.90  1.00  1.10  1.29  4236    1
## alpha[8]    0.69    0.00 0.11  0.49  0.62  0.69  0.77  0.92  4665    1
## alpha[9]    0.74    0.00 0.13  0.50  0.65  0.73  0.82  1.00  4283    1
## alpha[10]   0.99    0.00 0.13  0.76  0.90  0.99  1.07  1.26  5102    1
## alpha[11]   1.70    0.01 0.37  1.06  1.44  1.67  1.92  2.53  4334    1
## alpha[12]   0.28    0.00 0.07  0.14  0.23  0.28  0.33  0.43  3741    1
## alpha[13]   1.08    0.00 0.14  0.82  0.99  1.08  1.17  1.37  4081    1
## alpha[14]   1.03    0.00 0.14  0.76  0.93  1.03  1.12  1.32  3737    1
## alpha[15]   1.27    0.00 0.18  0.94  1.14  1.26  1.38  1.64  3176    1
## alpha[16]   0.72    0.00 0.10  0.52  0.64  0.71  0.78  0.93  4412    1
## alpha[17]   1.52    0.00 0.29  0.99  1.31  1.50  1.71  2.14  4374    1
## alpha[18]   1.64    0.00 0.18  1.31  1.52  1.63  1.75  2.01  4366    1
## alpha[19]   0.83    0.00 0.11  0.61  0.75  0.83  0.90  1.06  5713    1
## alpha[20]   1.47    0.00 0.22  1.05  1.32  1.45  1.60  1.93  3655    1
## alpha[21]   0.84    0.00 0.14  0.57  0.74  0.83  0.93  1.14  3793    1
## alpha[22]   1.47    0.00 0.25  1.03  1.30  1.45  1.63  2.01  4223    1
## alpha[23]   0.64    0.00 0.10  0.45  0.57  0.63  0.71  0.84  5169    1
## alpha[24]   1.17    0.00 0.16  0.88  1.06  1.17  1.27  1.50  3733    1
## alpha[25]   0.75    0.00 0.11  0.55  0.68  0.75  0.83  0.97  4200    1
## alpha[26]   1.48    0.00 0.16  1.18  1.37  1.48  1.59  1.82  3679    1
## alpha[27]   1.81    0.00 0.26  1.34  1.63  1.80  1.97  2.36  3892    1
## alpha[28]   1.04    0.00 0.13  0.80  0.96  1.04  1.13  1.30  5017    1
## alpha[29]   0.82    0.00 0.11  0.60  0.74  0.82  0.90  1.05  4281    1
## alpha[30]   0.43    0.00 0.09  0.28  0.37  0.43  0.49  0.61  4156    1
## alpha[31]   2.14    0.01 0.30  1.60  1.92  2.13  2.33  2.78  3136    1
## alpha[32]   0.38    0.00 0.07  0.24  0.32  0.37  0.42  0.53  4766    1
## beta[1]     1.34    0.00 0.21  0.98  1.19  1.31  1.46  1.81  3045    1
## beta[2]    -0.30    0.00 0.08 -0.46 -0.35 -0.30 -0.24 -0.14  3103    1
## beta[3]     1.09    0.00 0.15  0.83  0.99  1.09  1.19  1.41  3479    1
## beta[4]     0.92    0.00 0.21  0.57  0.77  0.90  1.05  1.39  4214    1
## beta[5]    -0.63    0.00 0.12 -0.87 -0.70 -0.62 -0.55 -0.42  3799    1
## beta[6]     1.85    0.00 0.22  1.48  1.70  1.83  1.98  2.33  3780    1
## beta[7]    -1.40    0.00 0.19 -1.81 -1.51 -1.38 -1.27 -1.08  3289    1
## beta[8]     2.21    0.01 0.35  1.64  1.96  2.16  2.40  2.99  3585    1
## beta[9]    -3.07    0.01 0.50 -4.23 -3.36 -3.01 -2.71 -2.26  3417    1
## beta[10]    0.36    0.00 0.11  0.15  0.29  0.36  0.44  0.59  3915    1
## beta[11]   -3.16    0.01 0.49 -4.34 -3.41 -3.09 -2.80 -2.40  3220    1
## beta[12]    1.37    0.01 0.54  0.58  0.99  1.27  1.63  2.64  2337    1
## beta[13]   -0.79    0.00 0.12 -1.06 -0.87 -0.78 -0.71 -0.58  3242    1
## beta[14]   -1.16    0.00 0.16 -1.50 -1.25 -1.14 -1.05 -0.88  3547    1
## beta[15]   -1.53    0.00 0.18 -1.93 -1.64 -1.51 -1.40 -1.23  2965    1
## beta[16]    0.54    0.00 0.15  0.27  0.43  0.53  0.64  0.87  4088    1
```

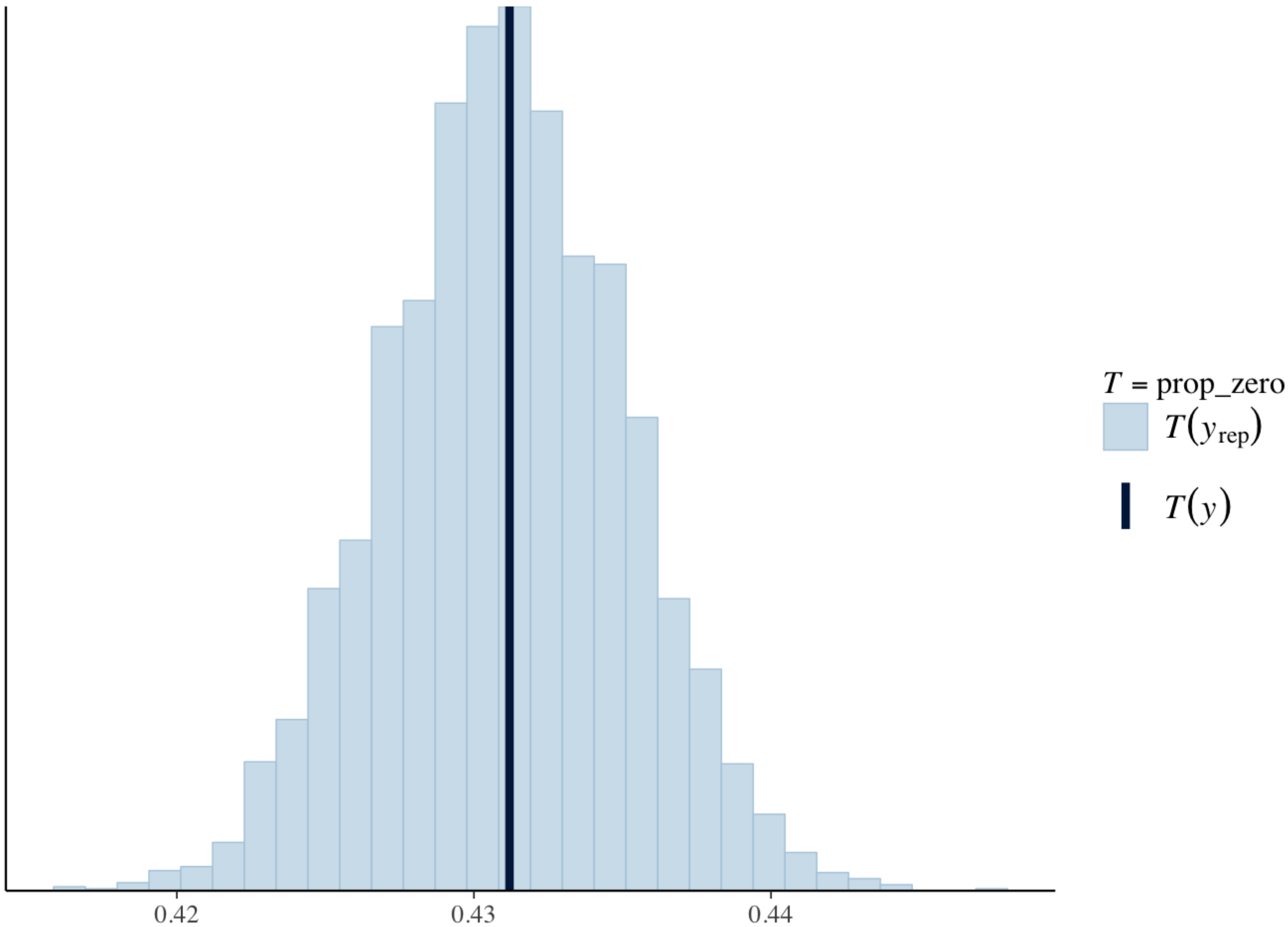
```
## beta[17]  -2.79    0.01 0.40 -3.70 -3.02 -2.74 -2.51 -2.17  3506    1
## beta[18]   0.51    0.00 0.08  0.36  0.46  0.51  0.57  0.69  3004    1
## beta[19]  -0.29    0.00 0.12 -0.54 -0.36 -0.28 -0.21 -0.06  3899    1
## beta[20]  -1.77    0.00 0.20 -2.22 -1.89 -1.75 -1.63 -1.44  2656    1
## beta[21]  -3.20    0.01 0.50 -4.35 -3.48 -3.13 -2.84 -2.40  2765    1
## beta[22]  -2.37    0.01 0.30 -3.02 -2.54 -2.34 -2.16 -1.88  3051    1
## beta[23]   1.36    0.00 0.25  0.95  1.18  1.33  1.50  1.91  4263    1
## beta[24]  -1.09    0.00 0.14 -1.39 -1.17 -1.08 -0.99 -0.84  2948    1
## beta[25]   0.76    0.00 0.16  0.48  0.65  0.74  0.85  1.11  4329    1
## beta[26]   0.11    0.00 0.08 -0.03  0.06  0.11  0.17  0.27  3025    1
## beta[27]  -1.51    0.00 0.14 -1.81 -1.60 -1.49 -1.41 -1.25  2590    1
## beta[28]  -0.17    0.00 0.10 -0.36 -0.23 -0.17 -0.10  0.02  3825    1
## beta[29]   0.92    0.00 0.16  0.63  0.81  0.91  1.02  1.27  3125    1
## beta[30]   0.61    0.00 0.25  0.20  0.44  0.59  0.76  1.17  4295    1
## beta[31]  -1.25    0.00 0.11 -1.48 -1.31 -1.24 -1.17 -1.05  2503    1
## beta[32]   4.56    0.02 0.93  3.13  3.91  4.42  5.09  6.76  3088    1
##
## Samples were drawn using NUTS(diag_e) at Sun Nov 11 18:49:32 2018.
## 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).
```

```
y_rep <- as.matrix(model, pars = "y_rep")
ppc_dens_overlay(y = sat_list$y, y_rep[1:200,])
```



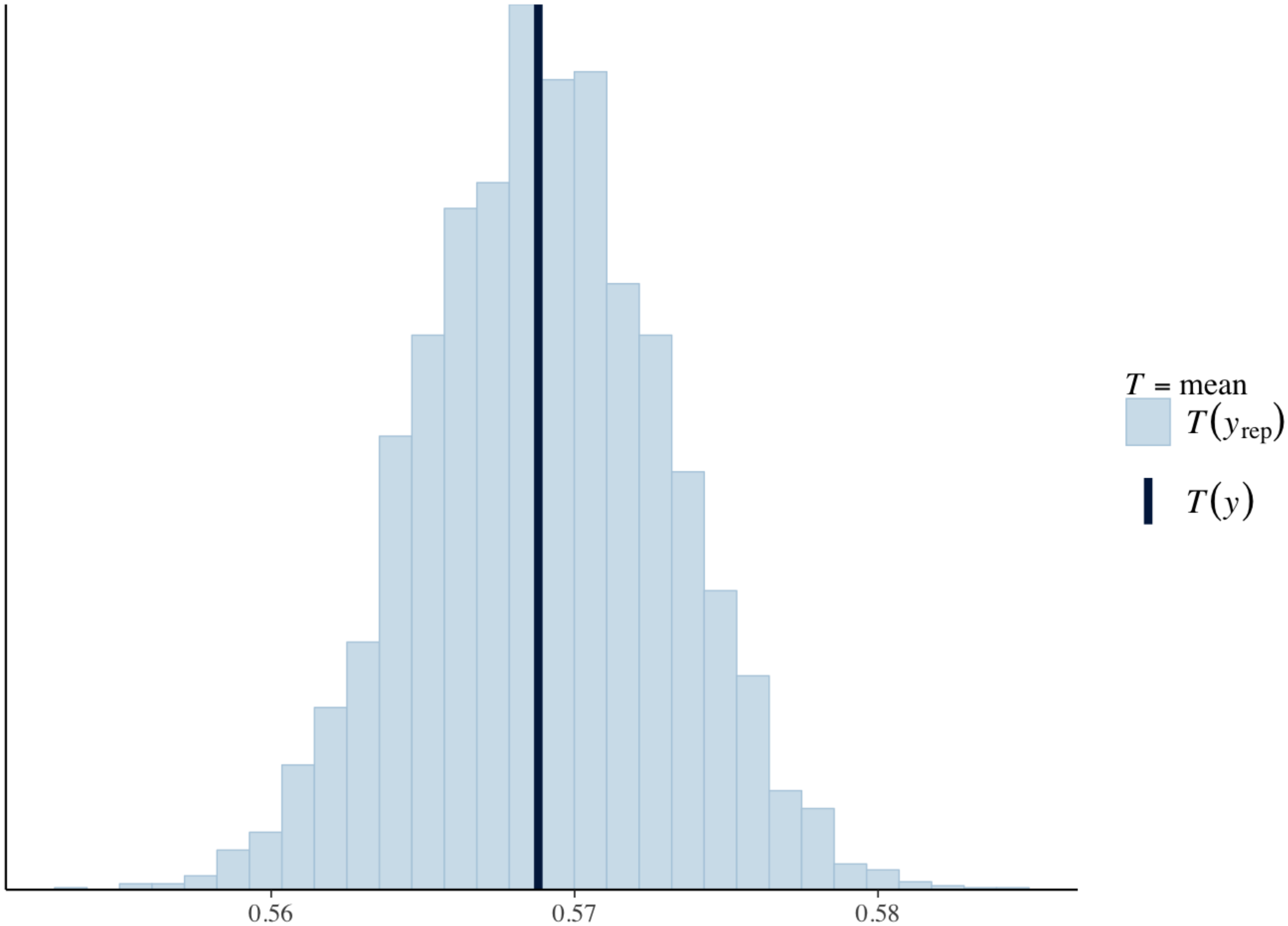
```r
prop_zero <- function(x) mean(x == 0)
ppc_stat(y = sat_list$y, yrep = y_rep, stat = "prop_zero")
```

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



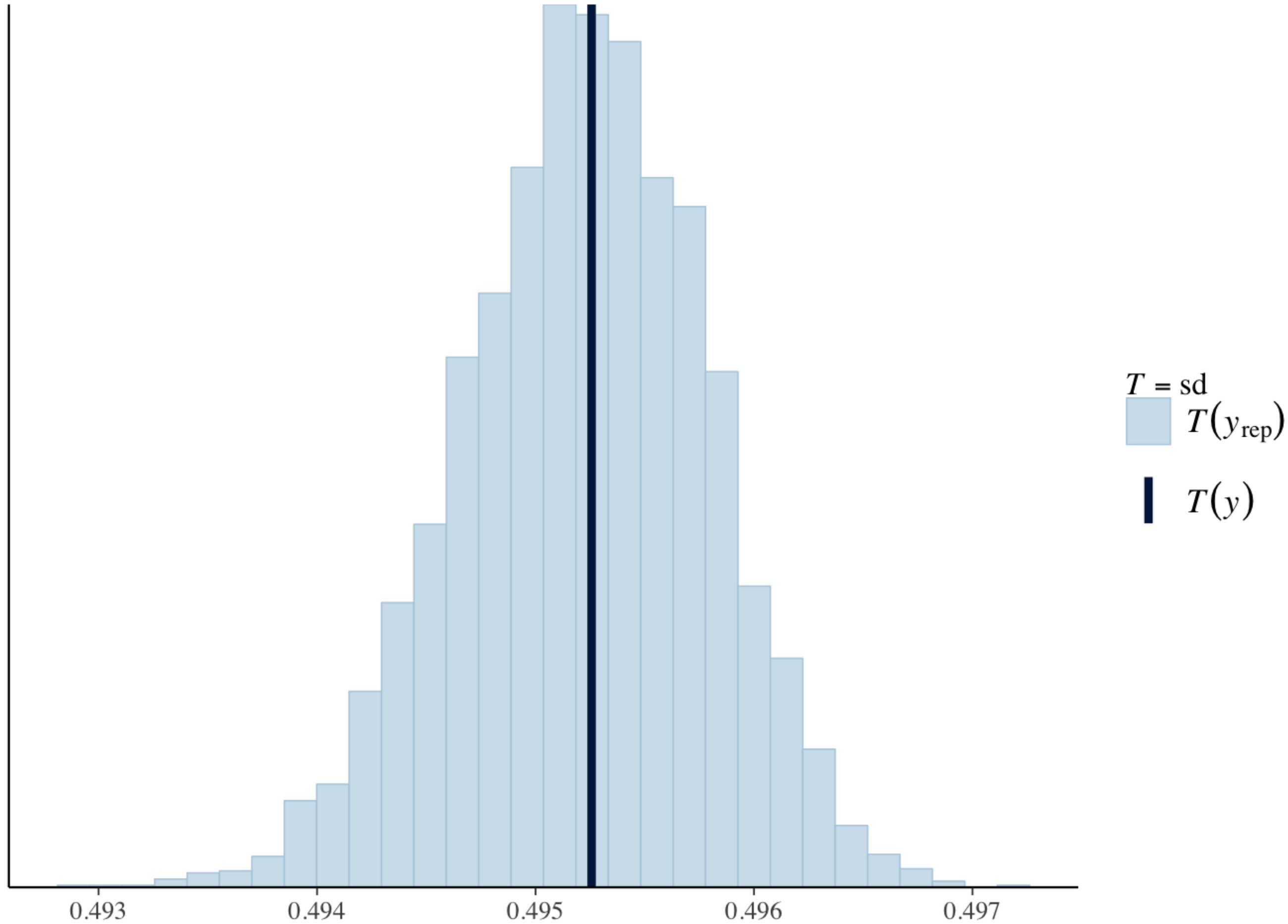

```r
ppc_stat(y = sat_list$y, yrep = y_rep, stat = "mean")
```

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

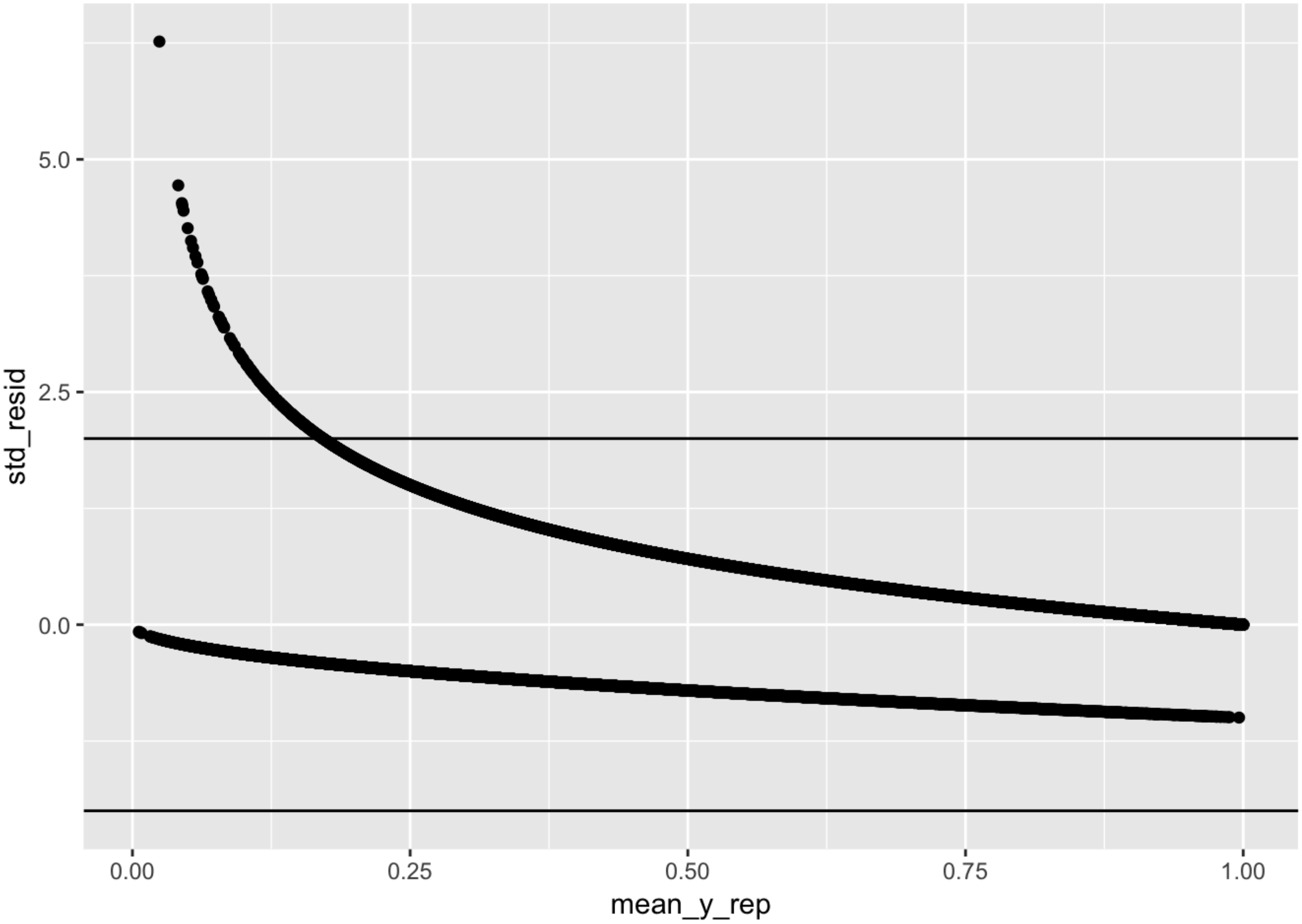



```
ppc_stat(y = sat_list$y, yrep = y_rep, stat = "sd")
```

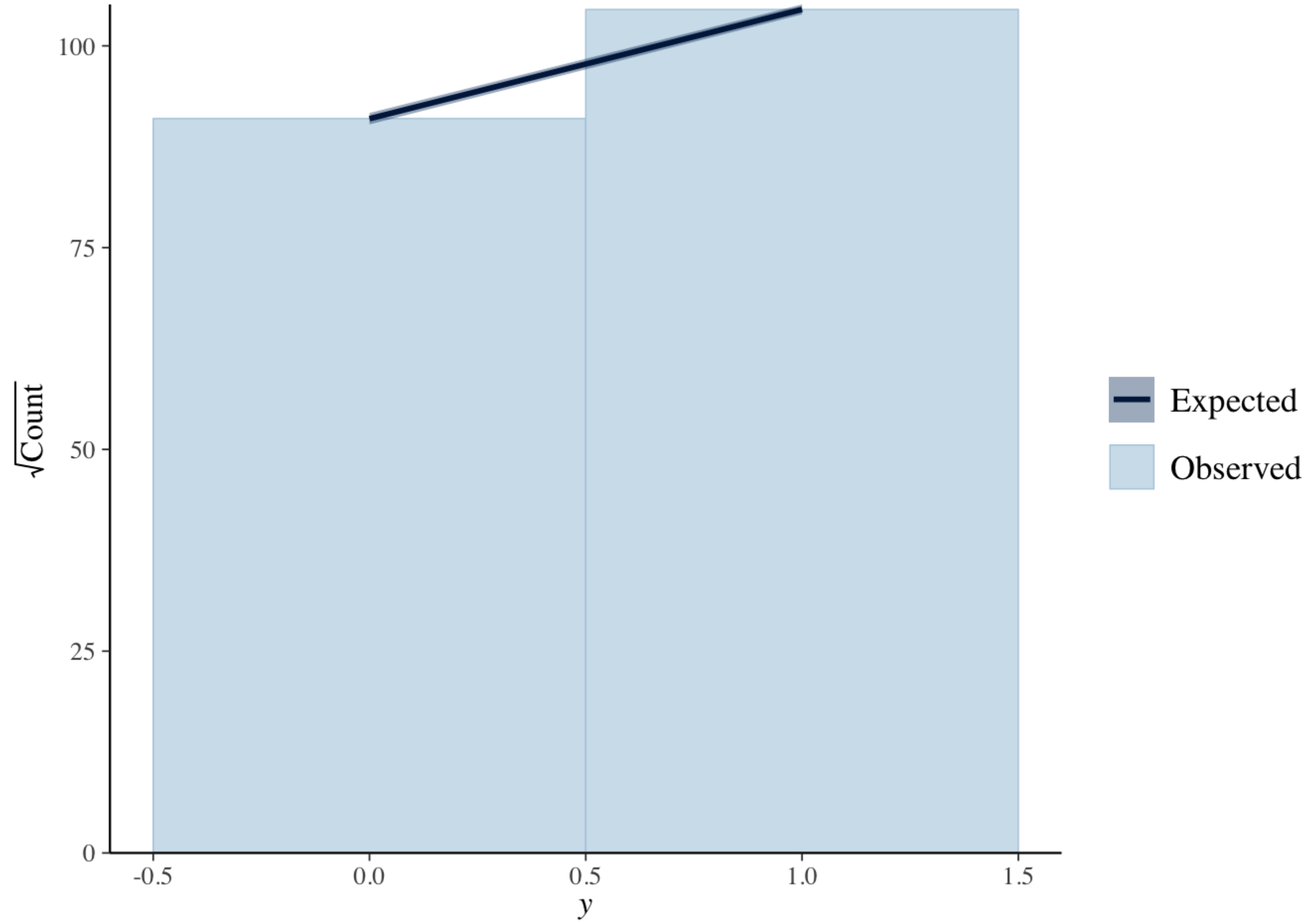
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
mean_y_rep <- colMeans(y_rep)
std_resid <- (sat_list$y - mean_y_rep) / sqrt(mean_y_rep)
qplot(mean_y_rep, std_resid) + hline_at(2) + hline_at(-2)
```




```
ppc_rootogram(sat_list$y, yrep = y_rep)
```



b. Expand the model as discussed in 1.b./class and interpret the results.

```
print_file("yi.stan")
```

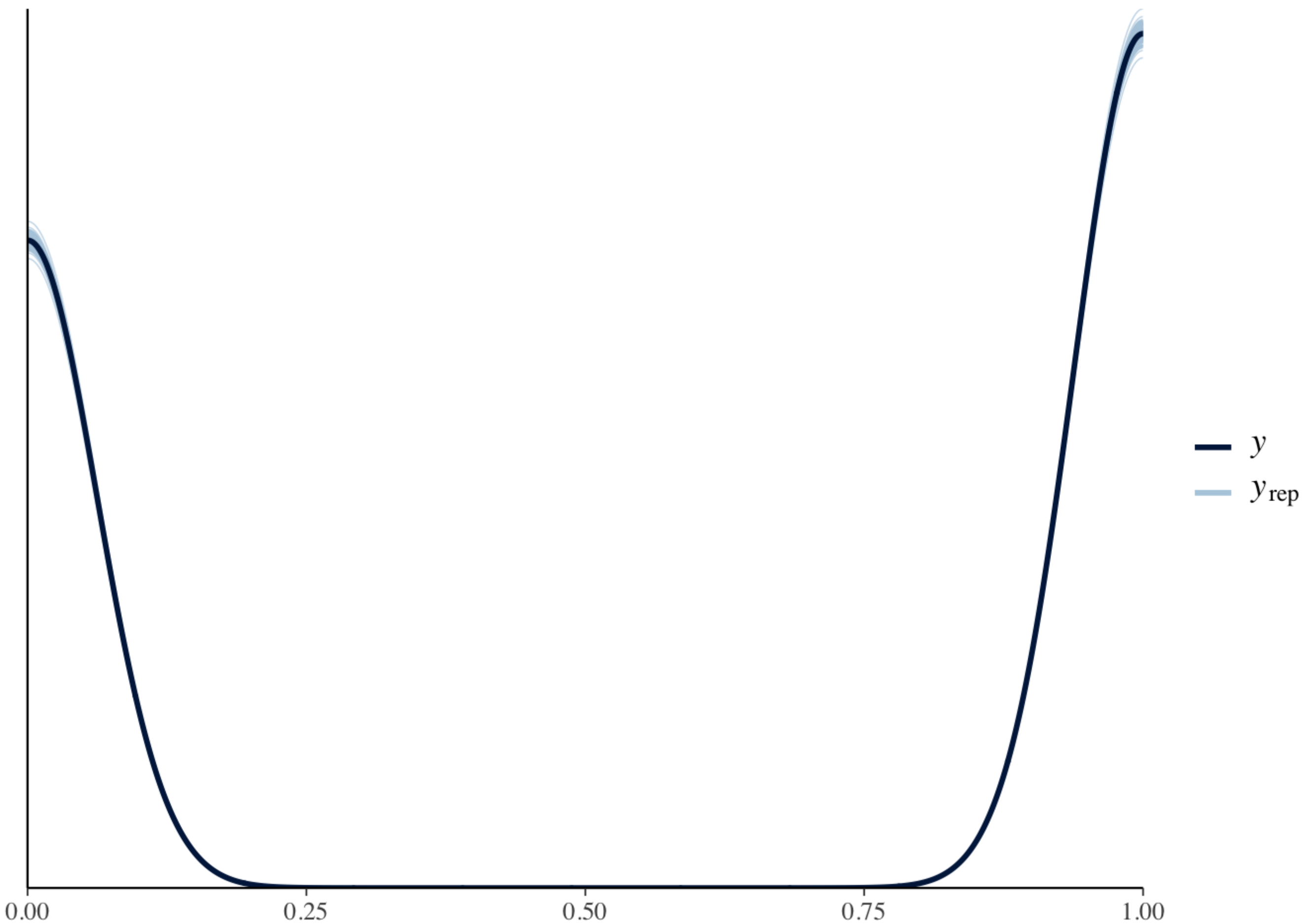
```
## data {
##   int<lower=1> I;                 // # of items
##   int<lower=1> J;                 // # of response
##   int<lower=1> N;                 // # observations
##   int<lower=1, upper=I> ii[N];  // item for n
##   int<lower=1, upper=J> jj[N];  // person for n
##   int<lower=0, upper=1> y[N];   // correctness for n
## }
## parameters {
##   vector[J] theta;              // abilities for response j
##   vector[2] xi[I];              // alpha/beta pair vectors
##   vector[2] mu;                 // vector for means of log alpha / beta
##   vector<lower=0>[2] tau;       // vector for alpha/beta residual sds
##   cholesky_factor_corr[2] L_Omega;
## }
## transformed parameters {
##   vector[I] alpha;              // discrimination for item i
##   vector[I] beta;               // difficulty for itme i
##   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 ~ cauchy(0, 1);
##   L_Omega ~ lkj_corr_cholesky(4);
##   mu[1] ~ cauchy(0,1);
##   tau[1] ~ exponential(.1);
##   mu[2] ~ cauchy(0,5);
##   tau[2] ~ exponential(.1);
##   y ~ bernoulli_logit(alpha[ii] .* (theta[jj] - beta[ii]));
## }
## generated quantities {
##   corr_matrix[2] Omega;
##   int<lower=0, upper=1> y_rep[N];
##   Omega = multiply_lower_tri_self_transpose(L_Omega);
##   for (n in 1:N){
##      y_rep[n] = bernoulli_logit_rng(alpha[ii[n]] * (theta[jj[n]] - beta[ii[n]]));
##   }
## }
```

```
IRT_extend = stan_model("yi.stan")
```

```
## hash mismatch so recompiling; make sure Stan code ends with a blank line
```

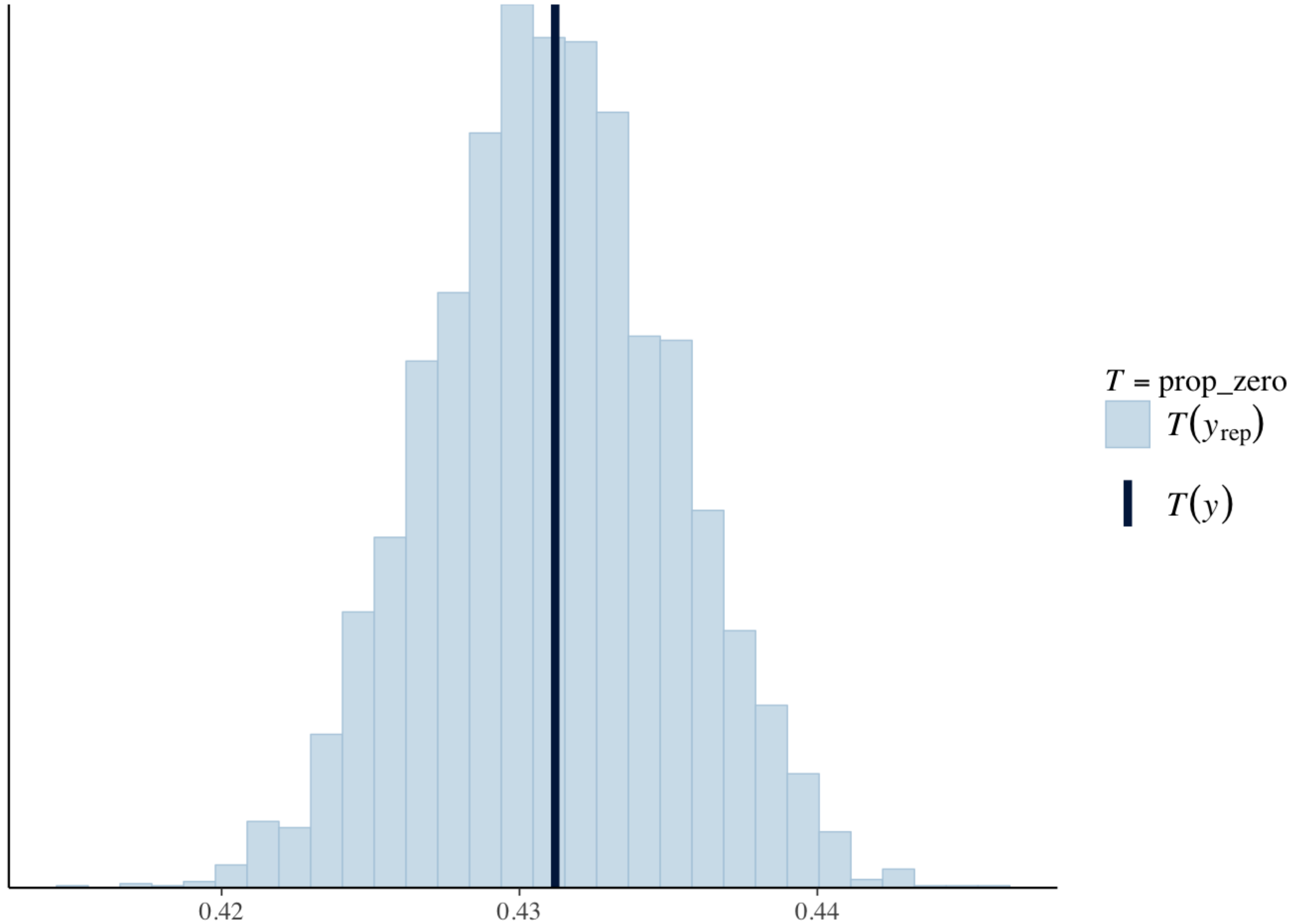
```
model_extend <- sampling(IRT_extend,data=sat_list )
```

```
y_rep <- as.matrix(model_extend, pars = "y_rep")
ppc_dens_overlay(y = sat_list$y, y_rep[1:200,])
```



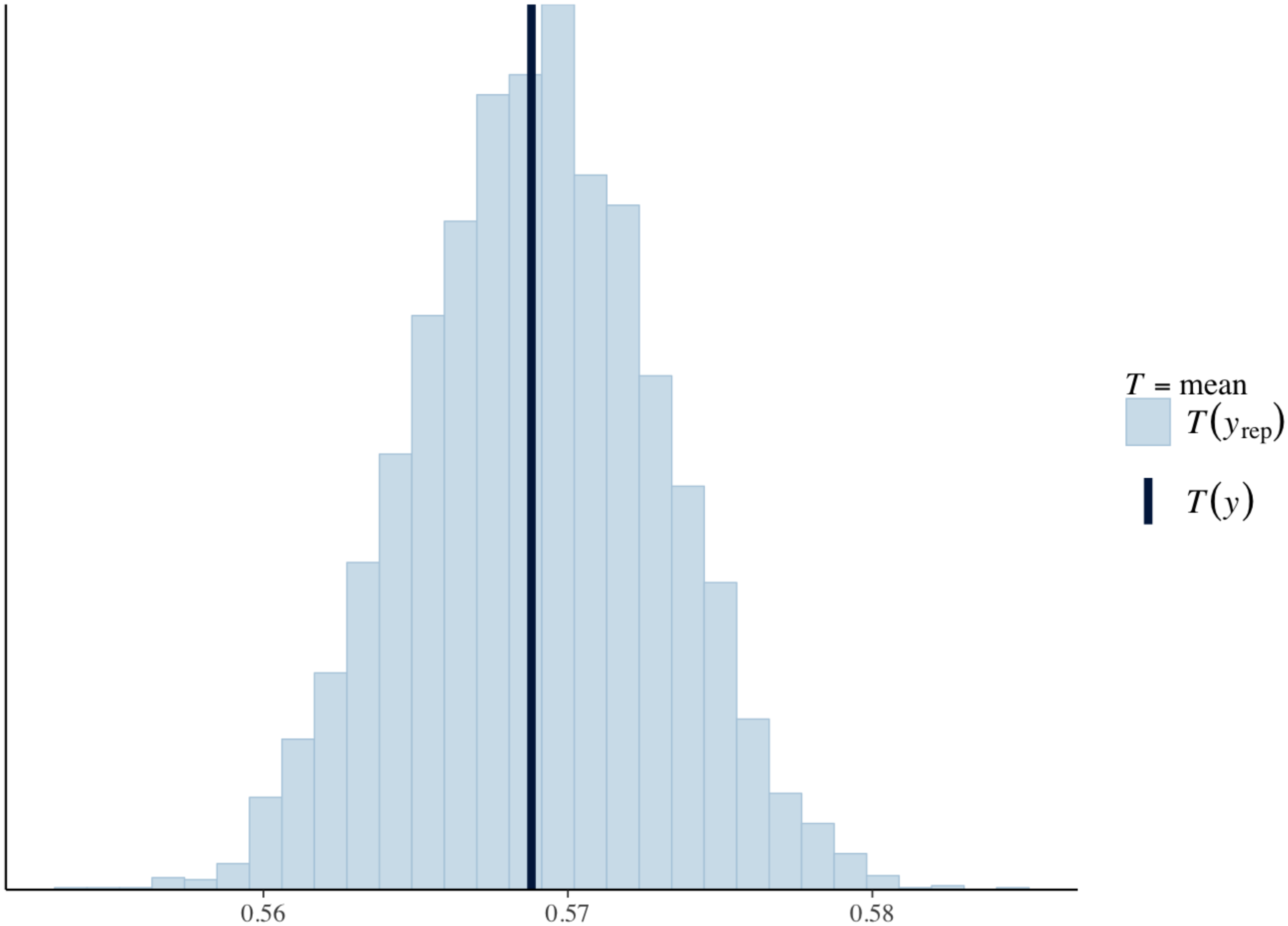

```
prop_zero <- function(x) mean(x == 0)
ppc_stat(y = sat_list$y, yrep = y_rep, stat = "prop_zero")
```

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



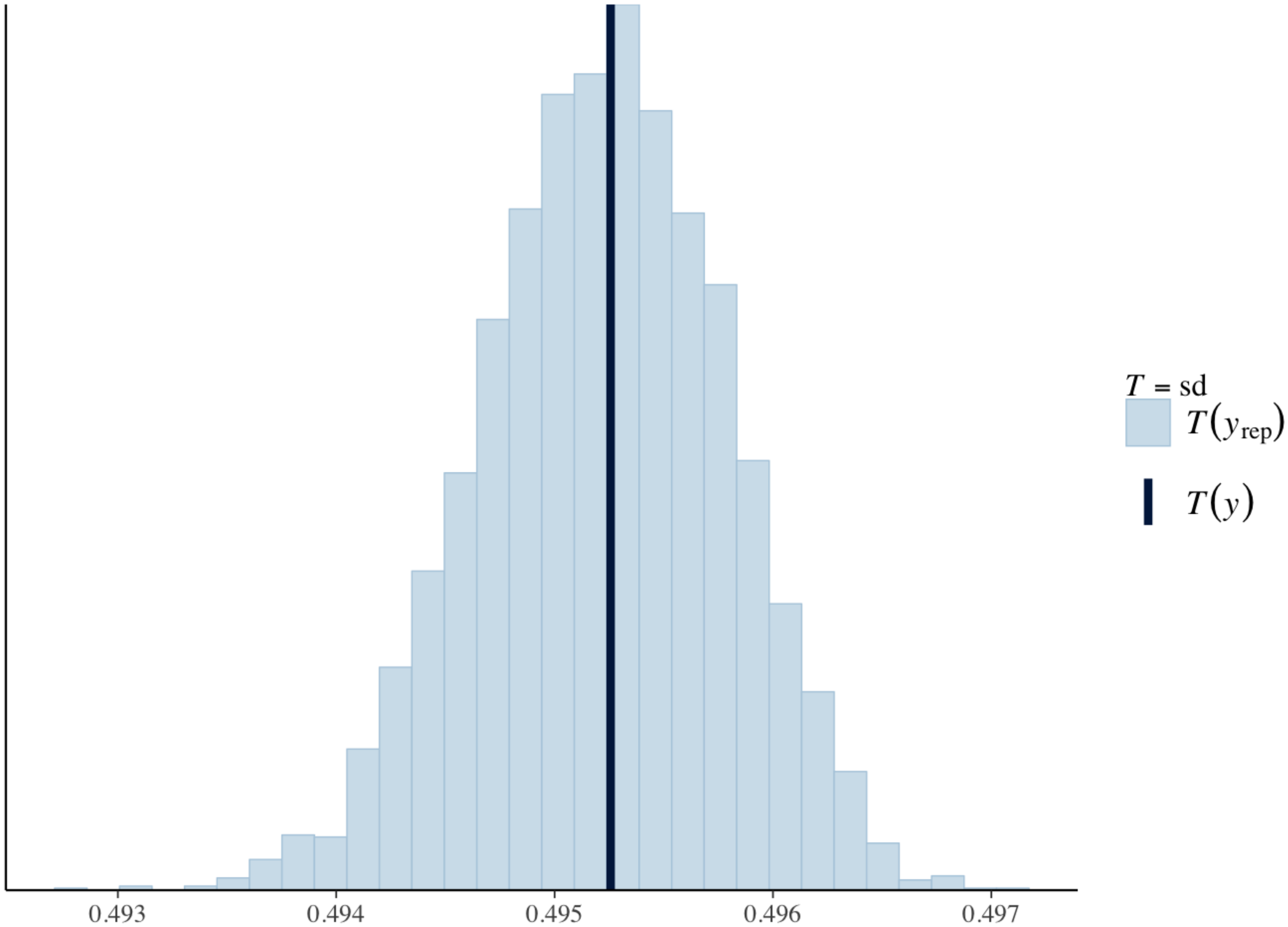

```
ppc_stat(y = sat_list$y, yrep = y_rep, stat = "mean")
```

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

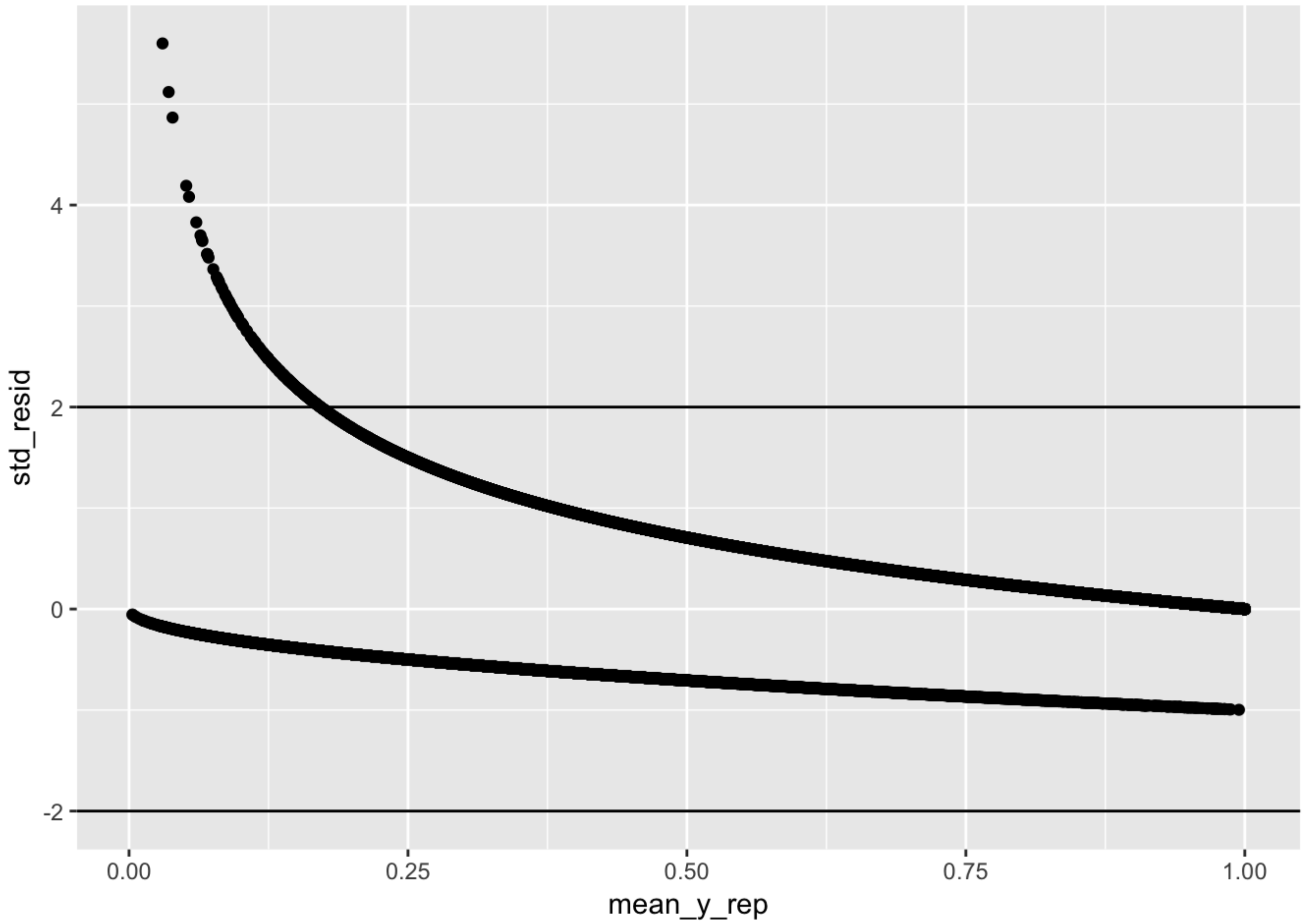



```
ppc_stat(y = sat_list$y, yrep = y_rep, stat = "sd")
```

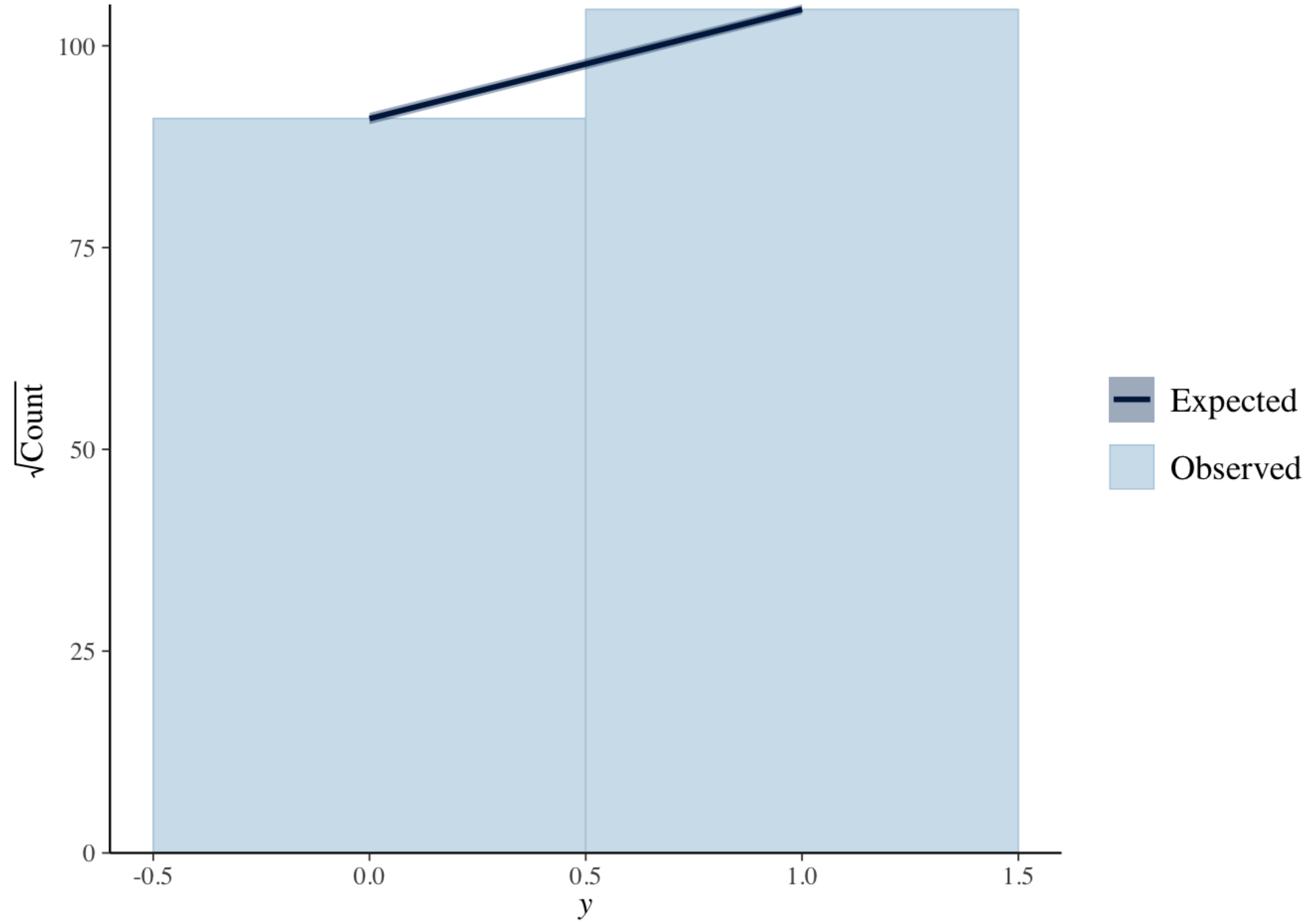
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
mean_y_rep <- colMeans(y_rep)
std_resid <- (sat_list$y - mean_y_rep) / sqrt(mean_y_rep)
qplot(mean_y_rep, std_resid) + hline_at(2) + hline_at(-2)
```




```
ppc_rootogram(sat_list$y, yrep = y_rep)
```



100
75
50
25
0
√Count
-0.5
0.0
0.5
1.0
1.5
y
Expected
Observed