Lathyrus ms2: selection on reaction norms for flowering time

Models with all data performed with brms

Alicia Valdés

Contents

Read data and check ns

```
datadef<-read.csv("data/datadef.csv")
head(datadef)
```

```
##   year id_nr    id fcode      FFD n_fl n_fr totseed intactseed shoot_vol period
## 1 1989     1 old_1     1       NA    6    3       8          6 1418.6000    old
## 2 1990     1 old_1     0       NA    0    0       0          0  523.2000    old
## 3 1991     1 old_1     1 59.91181   23    3      12         12 1915.4000    old
## 4 1992     1 old_1     1 55.66944   19    2       6          1 1460.1917    old
## 5 1993     1 old_1     1       NA   NA    0       0          0  879.6493    old
## 6 1994     1 old_1     1 59.18403   14    1       3          3 1338.6727    old
##   n_years_fl_fitness n_years_study   mean_4      cmean_4
## 1                  5             8 5.236667 -0.228207783
## 2                  5             8 7.195000  1.730125551
## 3                  5             8 5.245000 -0.219874449
## 4                  5             8 3.828333 -1.636541116
## 5                  5             8 5.461667 -0.003207783
## 6                  5             8 6.418333  0.953458884
```

Number of individuals in each period:

```
length(with(subset(datadef,period=="old"),unique(id)))
```

```
## [1] 607
```

```
length(with(subset(datadef,period=="new"),unique(id)))
```

```
## [1] 230
```

Number of observations in each period:

```
nrow(subset(datadef,period=="old"))
```

```
## [1] 4606
```

```
nrow(subset(datadef,period=="new"))
```

```
## [1] 2231
```

Number of cases with FFD in each period:

```
nrow(subset(datadef,period=="old"&!is.na(FFD)))
```

```
## [1] 1467
```

```
nrow(subset(datadef,period=="new"&!is.na(FFD)))
```

```
## [1] 1011
```

Data preparation

```
datadef_total<-datadef %>%
  group_by(id)%>%
  # Calculate mean fitness per year of study
  # and mean fitness per flowering event
  summarise(mean_fitness_study=sum(intactseed,na.rm=T)/mean(n_years_study),
            mean_fitness_fl=sum(intactseed,na.rm=T)/mean(n_years_fl_fitness))%>%
   arrange(.,id) # Order by id

with(datadef_total,cor(mean_fitness_study,mean_fitness_fl))  # Highly corr (0.87)
```

```
## [1] 0.8660685
```

```
# Calculate mean shoot volume for each id using values of shoot volume
# for all ids/years (including flowering and non-flowering years)

shoot_vol_all_means<-datadef[c(1,3,10)]%>%
  group_by(id)%>%
  summarise(shoot_vol_mean=mean(shoot_vol,na.rm=T))
# Mean of all available values

# Join shoot volume data
datadef_total<-datadef_total%>%left_join(shoot_vol_all_means)%>%
  left_join(unique(datadef[c(2,3,11)]))
head(datadef_total)
```

```
## # A tibble: 6 x 6
##   id      mean_fitness_study mean_fitness_fl shoot_vol_mean id_nr period
##   <chr>                <dbl>           <dbl>          <dbl> <int> <chr>
## 1 new_1                 0               0             1830.     1 new
## 2 new_10               14.3            15.7           9794.    10 new
## 3 new_100               3.89            5.83          1959.   100 new
## 4 new_101               2.25            3.00          1195.   101 new
## 5 new_102               5.61            6.73          3269.   102 new
## 6 new_103               3.60            4.32          1694.   103 new
```

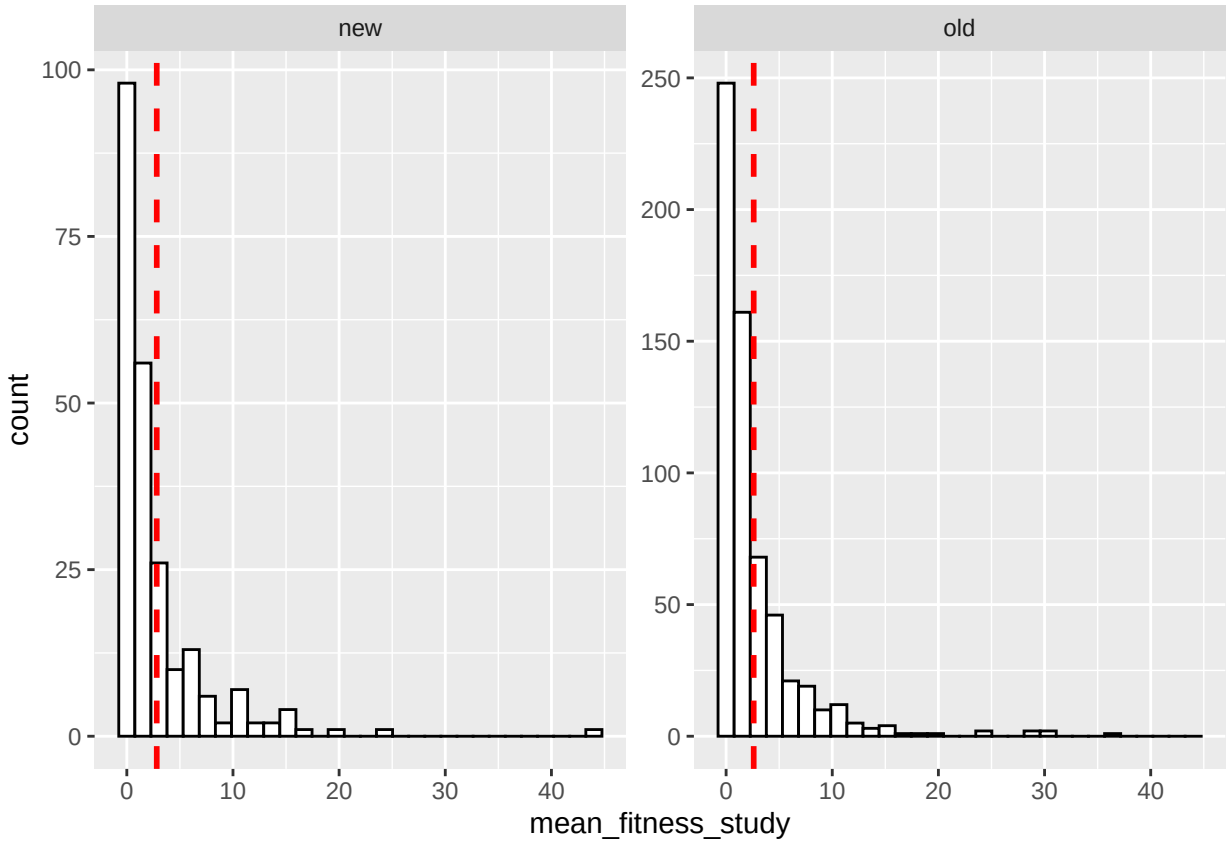
```
nrow(subset(datadef_total,is.na(shoot_vol_mean)))
```

```
## [1] 46
```

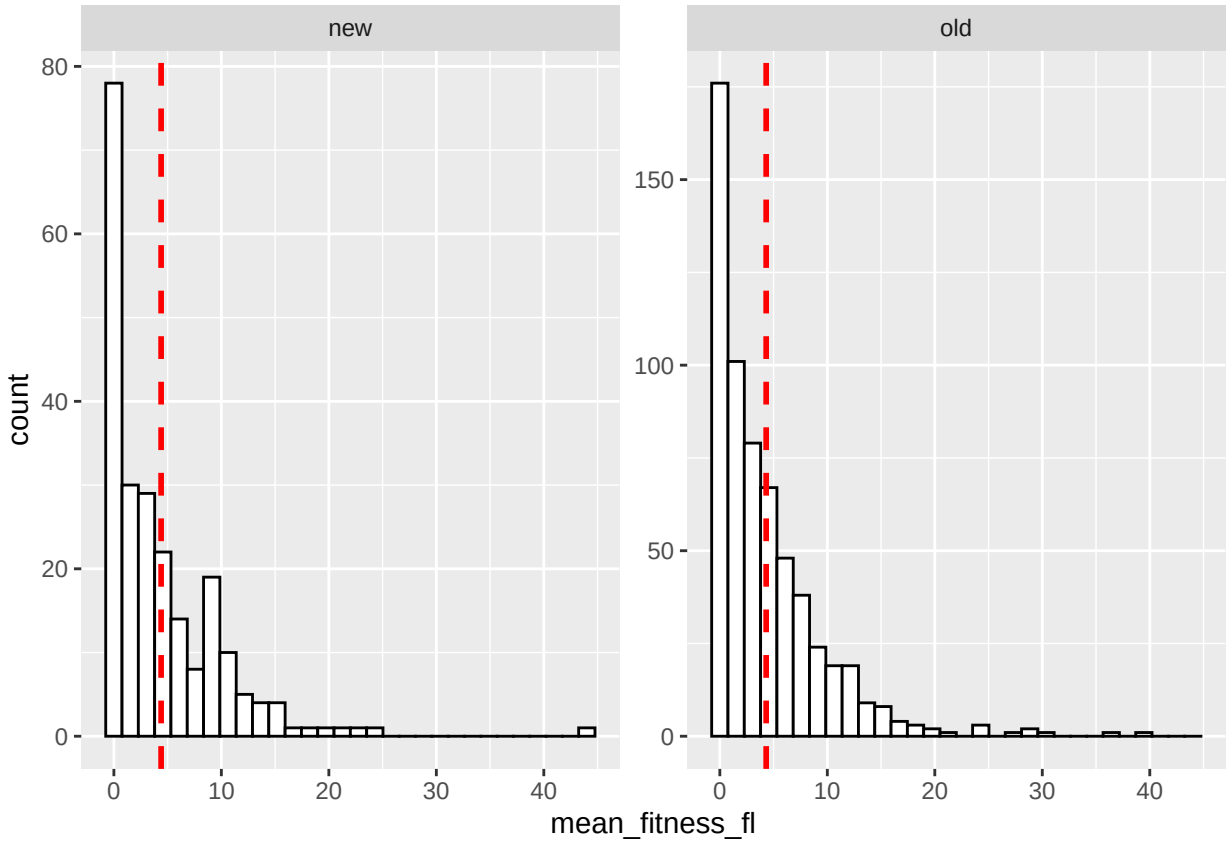
```
# 46 ids with no info on shoot volume

# Add first_yr to total data +
# Year column is only relevant for FFD, but is set to first_yr for fitness values
datadef_total$first_yr<-ifelse(grepl("old",as.character(datadef_total$id)),
                               1987,2006)

# Using sqrt of mean shoot volume over all years when available, centered
datadef_total<-datadef_total%>%
  mutate(shoot_vol_mean_sqrt=sqrt(shoot_vol_mean),
         cn_shoot_vol_mean_sqrt=scale(shoot_vol_mean_sqrt,center=T,scale=F))
```

Compare distributions of mean fitness per year of study and mean fitness per flowering event between old and new periods:

```
ggplot(datadef_total,aes(x=mean_fitness_study))+
  geom_histogram(colour="black",fill="white",position="dodge")+
  facet_wrap(~period,scales="free_y")+
  geom_vline(data=plyr::ddply(datadef_total,"period",summarise,
                        mean_fitness_study.mean=mean(mean_fitness_study)),
             aes(xintercept=mean_fitness_study.mean),
             linetype="dashed", size=1, colour="red")
```




```
ggplot(datadef_total,aes(x=mean_fitness_fl))+
  geom_histogram(colour="black",fill="white",position="dodge")+
  facet_wrap(~period,scales="free_y")+
  geom_vline(data=plyr::ddply(datadef_total,"period",summarise,
                        mean_fitness_fl.mean=mean(mean_fitness_fl)),
             aes(xintercept=mean_fitness_fl.mean),
             linetype="dashed", size=1, colour="red")
```



Distributions and means of the two mean fitness measures are similar among the two periods.

Univariate models

FFD with random effect of year only

```
my.cores <- detectCores()
```

```
univar.FFD_yearonly.all.brm<-brm(formula=FFD~cmean_4+(1|year),data=datadef,
                warmup = 1000,iter = 4000,thin=2,chains=4,
                # 4 chains, each with 4000 iterations
                inits = "random",seed = 12345,cores = my.cores)
# Total of 6000 post-warmup samples
```

```
summary(univar.FFD_yearonly.all.brm)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: FFD ~ cmean_4 + (1 | year)
##    Data: datadef (Number of observations: 2478)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
```

```
##
## Group-Level Effects:
## ~year (Number of levels: 22)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     5.16      0.83     3.80     7.08 1.00     1393     2951
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    58.52      1.14    56.21    60.69 1.00     1759     2343
## cmean_4      -2.46      0.83    -4.19    -0.85 1.00     1812     2804
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     4.73      0.07     4.60     4.87 1.00     4389     4219
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

FFD with random effects of year and individual-intercept

```
univar.FFD.all.brm<-brm(formula=FFD~cmean_4+(1|year)+(1|id),data=datadef,
                warmup = 1000,iter = 4000,thin=2,chains=4,
                # 4 chains, each with 4000 iterations
                inits = "random",seed = 12345,cores = my.cores)
# Total of 6000 post-warmup samples
```

```
summary(univar.FFD.all.brm)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: FFD ~ cmean_4 + (1 | year) + (1 | id)
##    Data: datadef (Number of observations: 2478)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
##
## Group-Level Effects:
## ~id (Number of levels: 837)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     1.60      0.14     1.32     1.88 1.00     3477     4522
##
## ~year (Number of levels: 22)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     5.13      0.85     3.76     7.01 1.00     2252     3898
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    58.56      1.11    56.44    60.77 1.00     1559     2547
## cmean_4      -2.43      0.81    -4.00    -0.86 1.00     1964     3541
##
## Family Specific Parameters:
```

```
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     4.45      0.07     4.31     4.59 1.00     4320     5245
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Random regression for FFD, including random effects of individual slopes and covariance between intercept and slope

```
univar.FFD_RR.all.brm<-brm(formula=FFD~cmean_4+(1|year)+(cmean_4|id),
                           data=datadef,
                           warmup = 1000,iter = 4000,thin=2,chains=4,
                           inits = "random",seed = 12345,cores = my.cores,
                           sample_prior="yes")
# Total of 6000 post-warmup samples
```

```
summary(univar.FFD_RR.all.brm)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: FFD ~ cmean_4 + (1 | year) + (cmean_4 | id)
##    Data: datadef (Number of observations: 2478)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
##
## Group-Level Effects:
## ~id (Number of levels: 837)
##                         Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)               1.55      0.15     1.26     1.84 1.00     3341
## sd(cmean_4)                 0.78      0.13     0.53     1.05 1.00     2943
## cor(Intercept,cmean_4)      0.80      0.13     0.49     0.99 1.00     1667
##                         Tail_ESS
## sd(Intercept)               4633
## sd(cmean_4)                 4808
## cor(Intercept,cmean_4)      3220
##
## ~year (Number of levels: 22)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     5.16      0.86     3.83     7.21 1.00     2589     3971
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    58.59      1.12    56.49    60.86 1.00     1513     2749
## cmean_4      -2.37      0.83    -4.05    -0.74 1.00     2067     3294
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     4.37      0.07     4.22     4.51 1.00     3543     4353
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
```

```
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Compare models

```
loo_comp<-loo_compare(univar.FFD_yearonly.all.brm,univar.FFD.all.brm,
                      univar.FFD_RR.all.brm, criterion="loo")
```

```
loo_comp
```

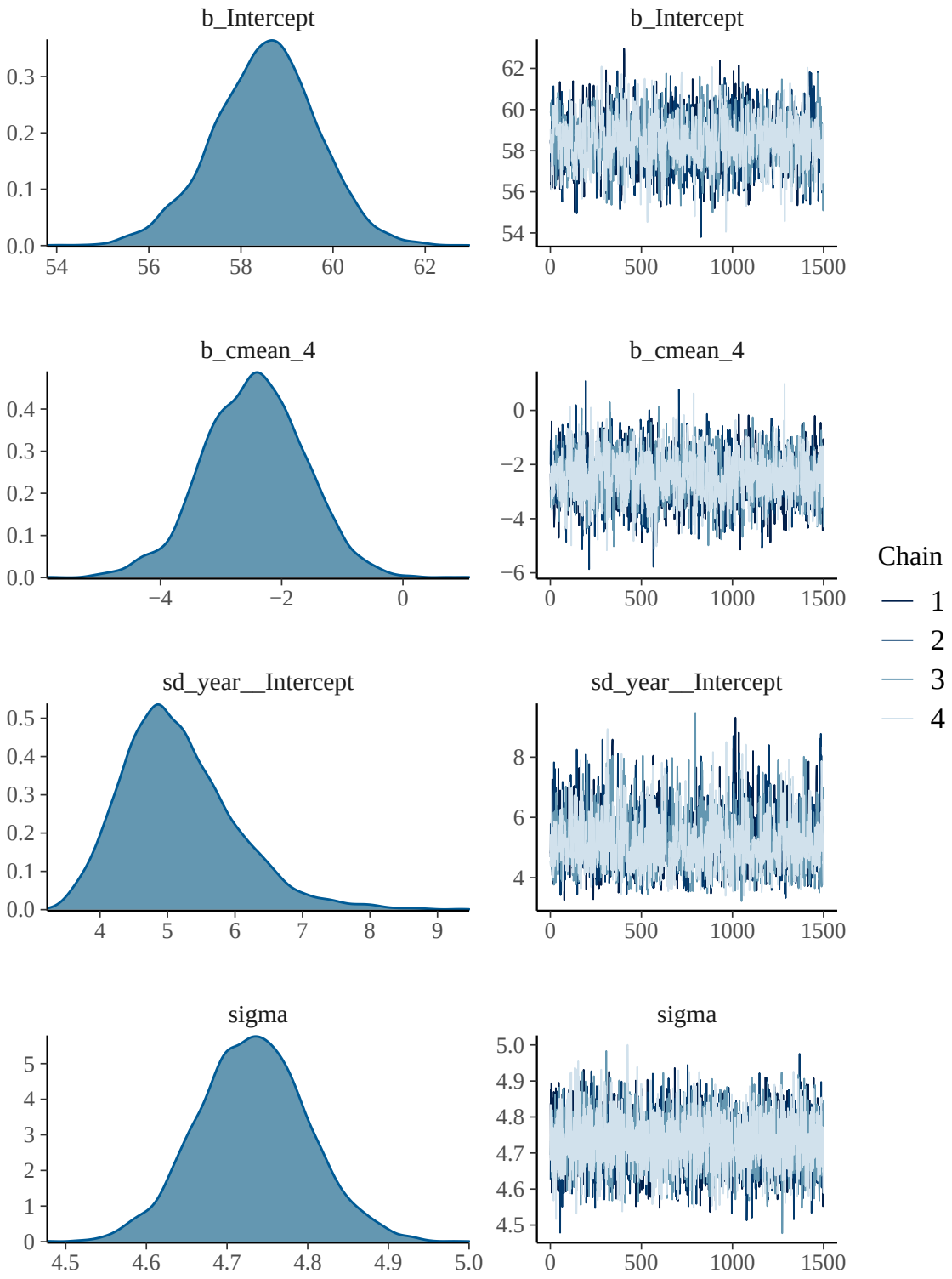
```
##                              elpd_diff se_diff
## univar.FFD_RR.all.brm          0.0       0.0
## univar.FFD.all.brm           -13.2       8.1
## univar.FFD_yearonly.all.brm -58.9      13.4
```

Random regression model is better

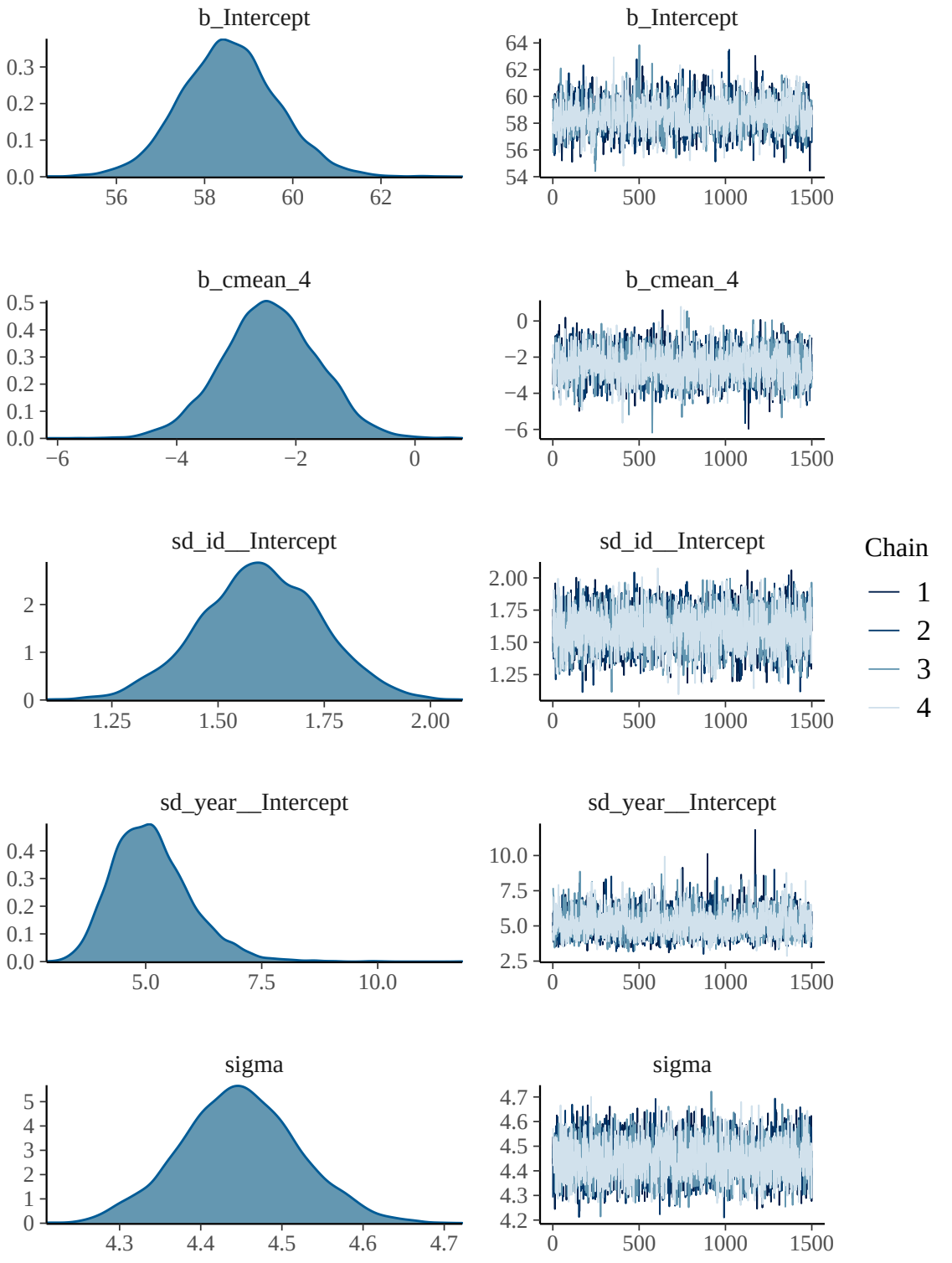
Model evaluation

From https://biol609.github.io/lectures/23c_brms_prediction.html#24_evaluating_brms_models

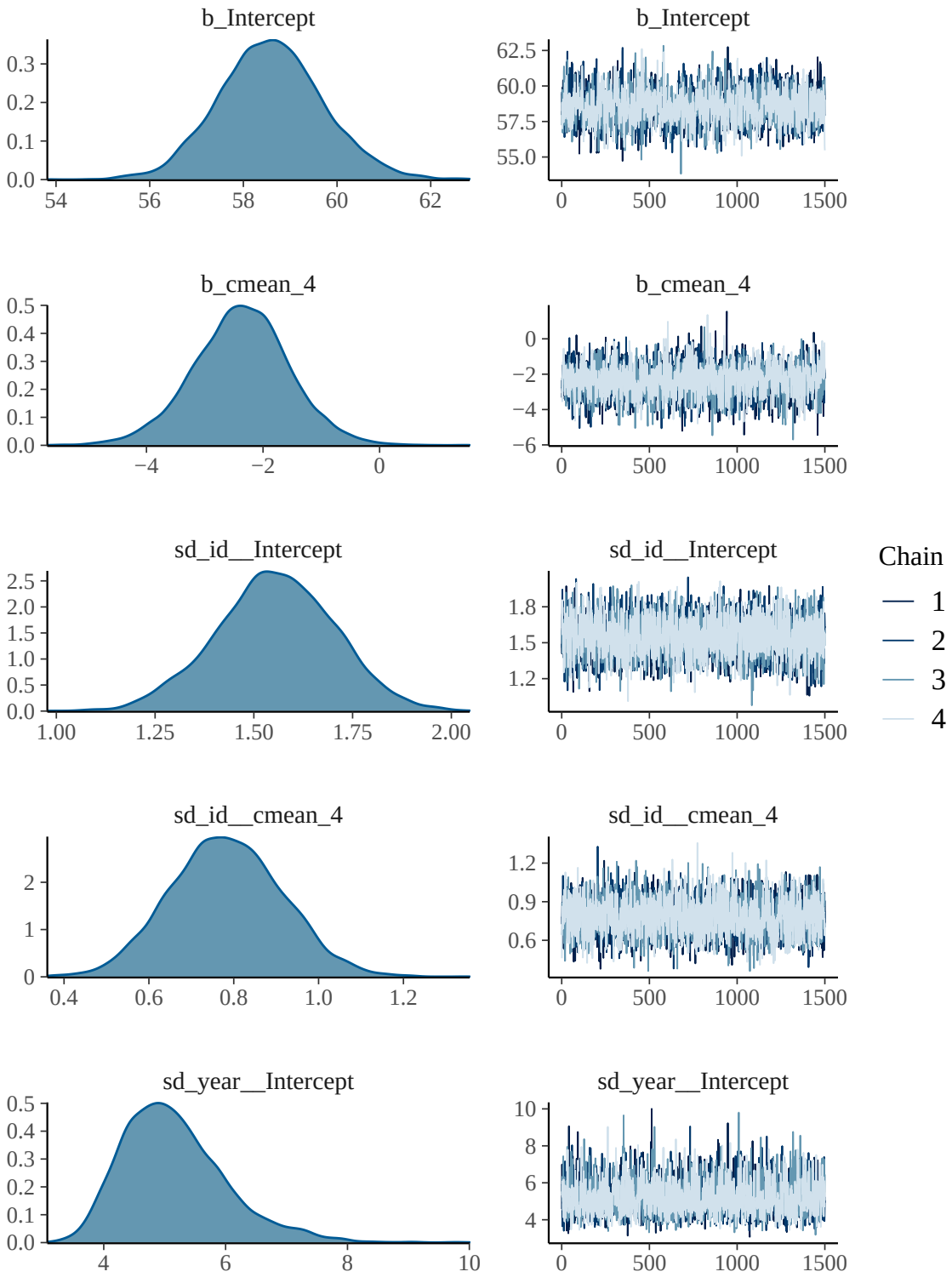
```
plot(univar.FFD_yearonly.all.brm)
```

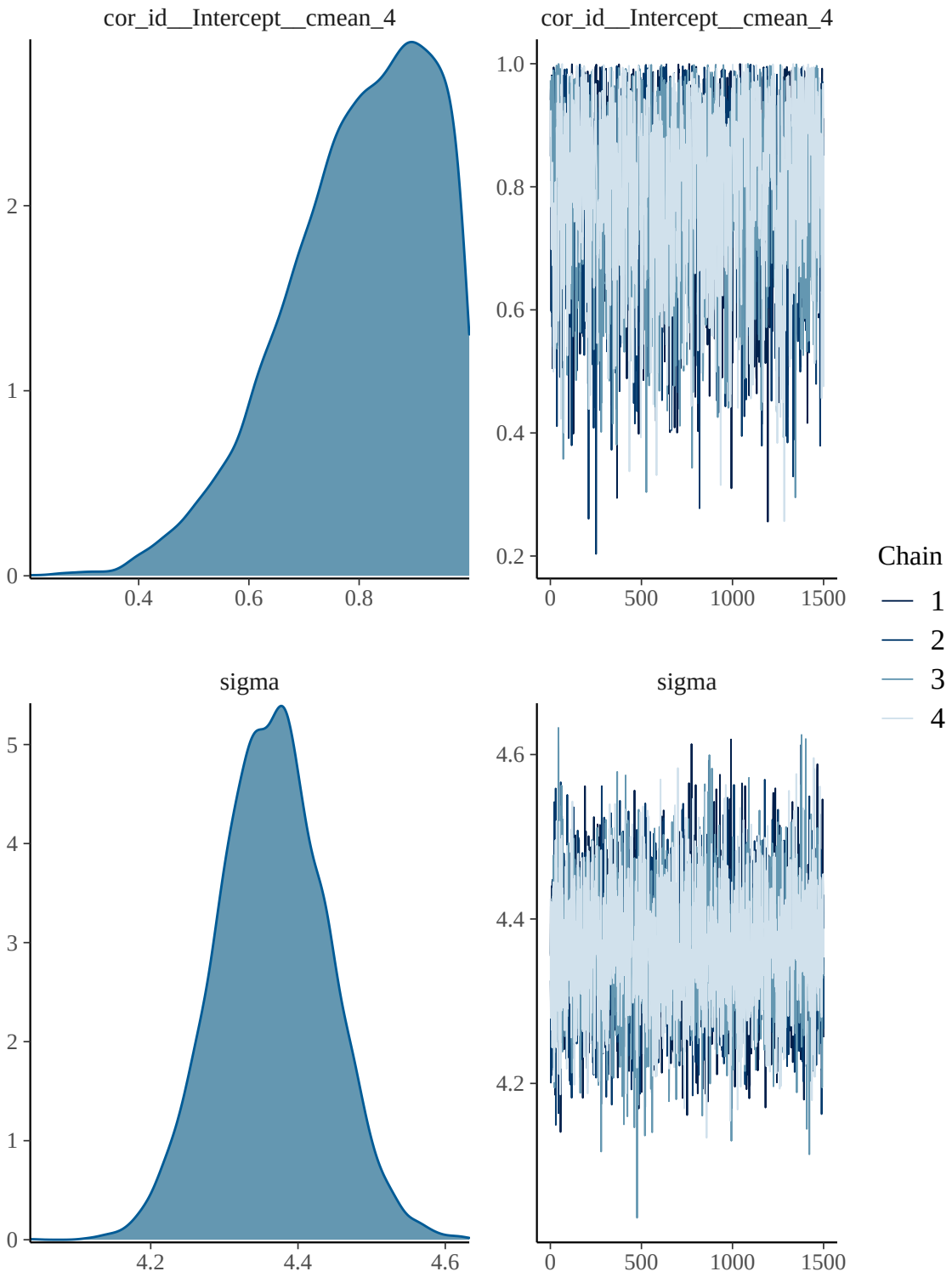



```
plot(univar.FFD.all.brm)
```




```
plot(univar.FFD_RR.all.brm)
```


b_Intercept
b_Intercept
b_cmean_4
b_cmean_4
sd_id__Intercept
sd_id__Intercept
sd_id__cmean_4
sd_id__cmean_4
sd_year__Intercept
sd_year__Intercept
Chain
1
2
3
4

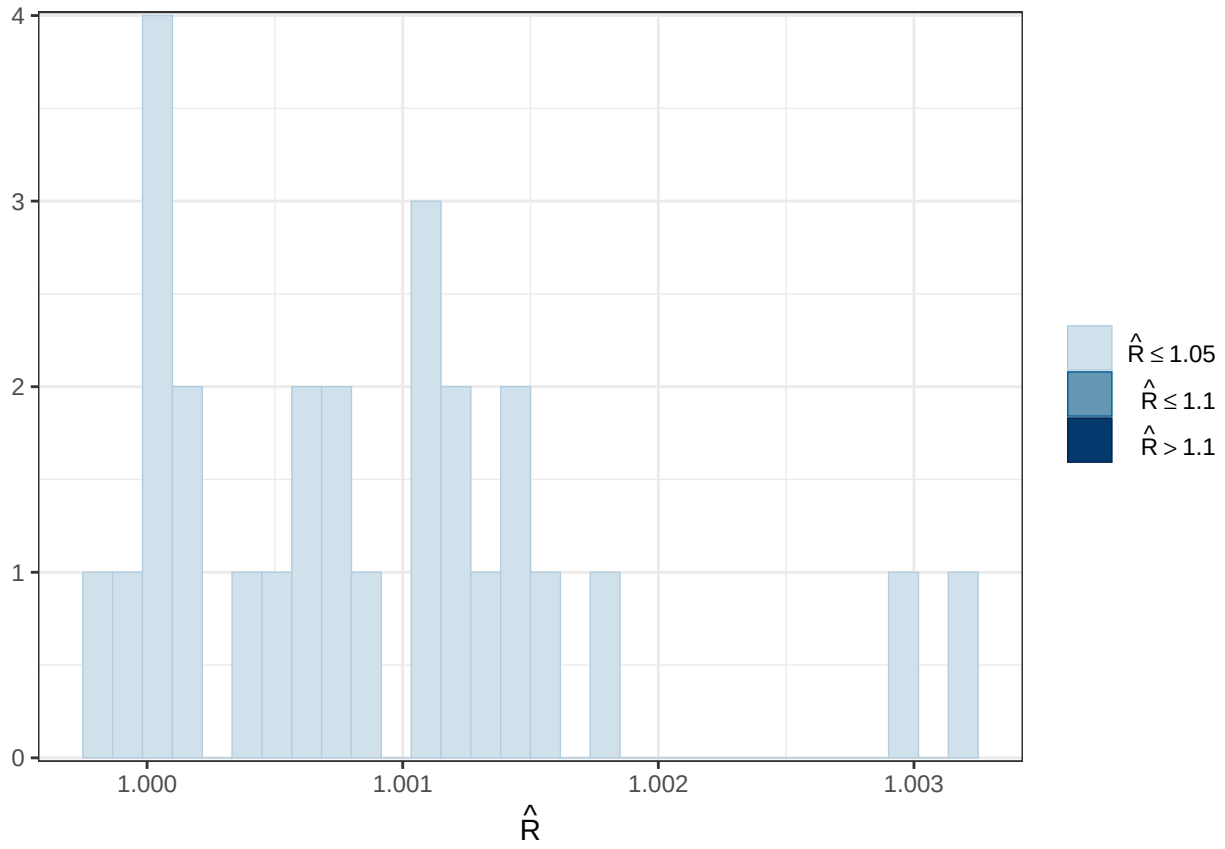


Rhat (potential scale reduction statistic): monitors whether a chain has converged to the equilibrium distribution by comparing its behavior to other randomly initialized chains.

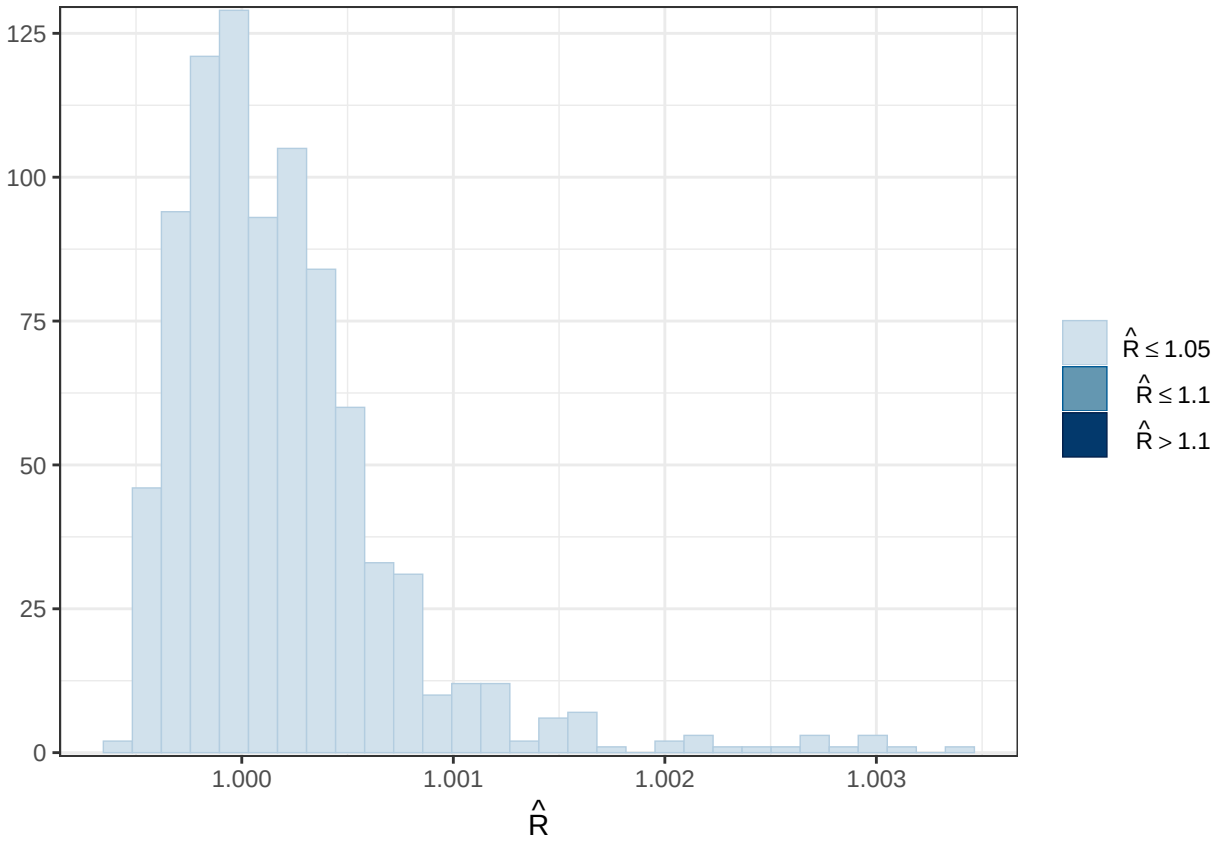
- light: below 1.05 (good)
- mid: between 1.05 and 1.1 (ok)

- dark: above 1.1 (too high)

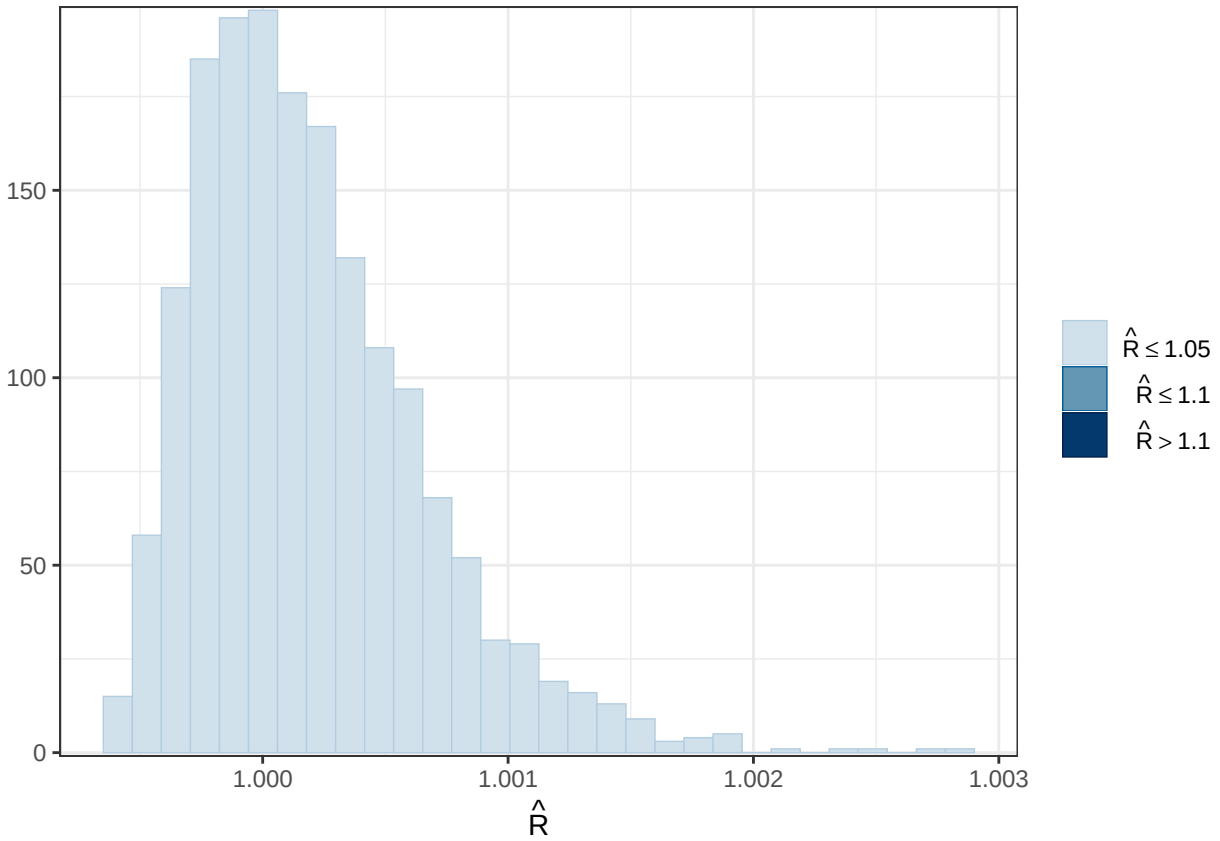
```
mcmc_rhat_hist(rhat(univar.FFD_yearonly.all.brm))+theme_bw()
```




```
mcmc_rhat_hist(rhat(univar.FFD.all.brm))+theme_bw()
```




```
mcmc_rhat_hist(rhat(univar.FFD_RR.all.brm))+theme_bw()
```

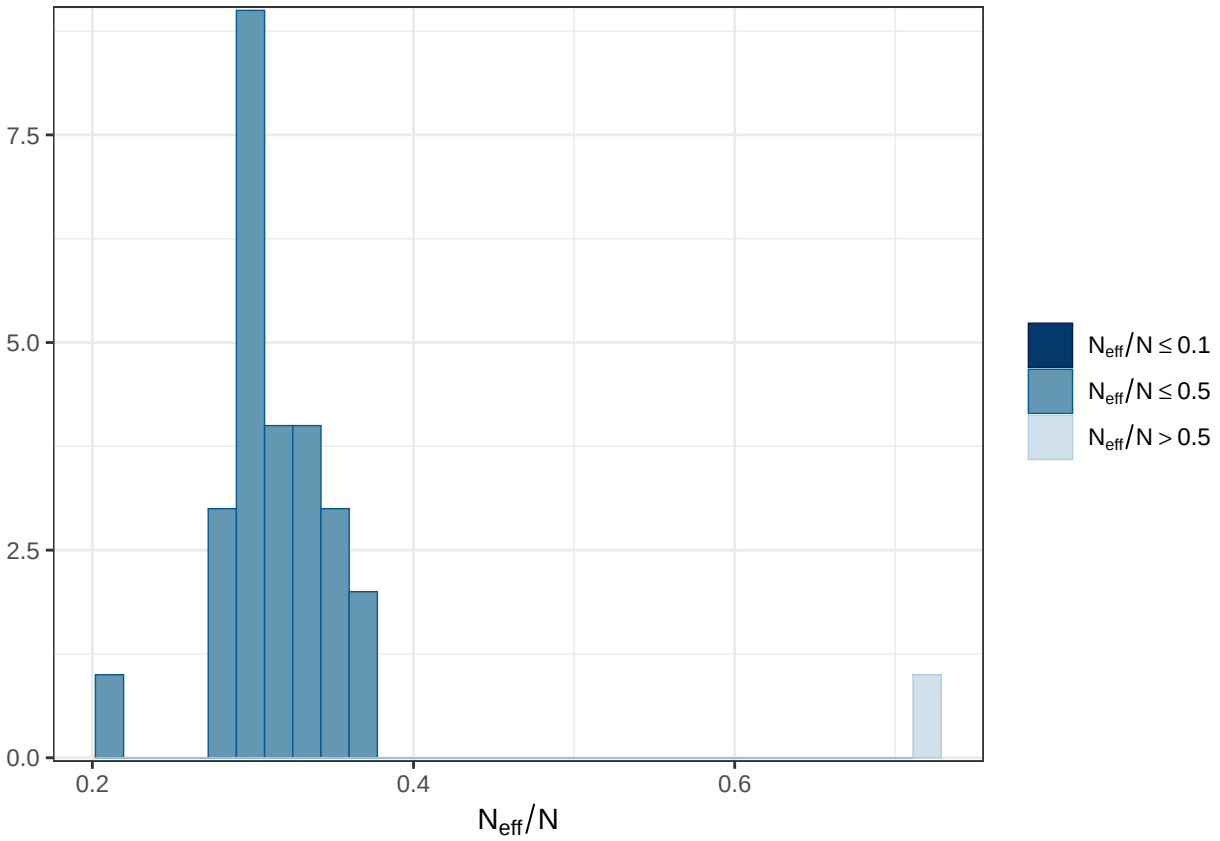


Effective sample size: an estimate of the number of independent draws from the posterior distribution of the estimand of interest.

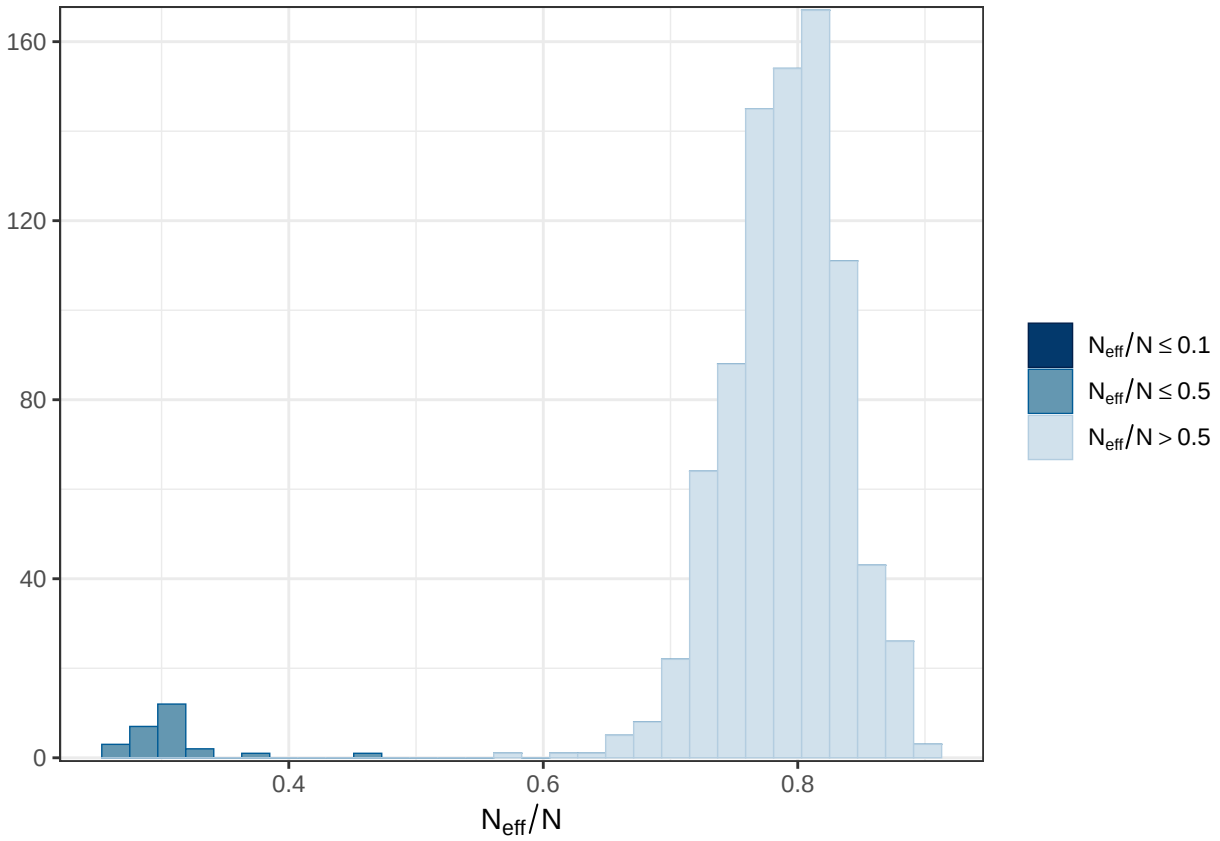
- light: between 0.5 and 1 (high)
- mid: between 0.1 and 0.5 (good)
- dark: below 0.1 (low)

We should worry about any values less than 0.1.

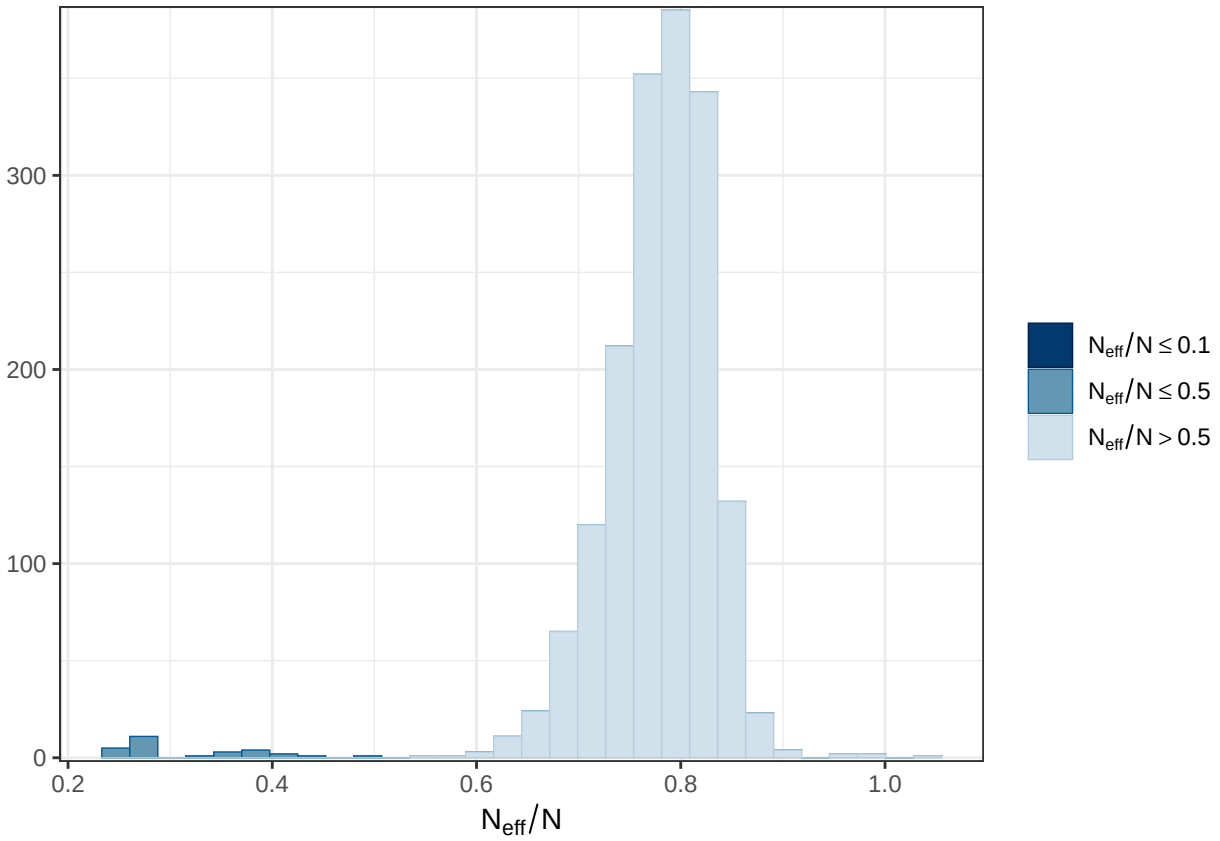
```
mcmc_neff_hist(neff_ratio(univar.FFD_yearonly.all.brm))+theme_bw()
```




```
mcmc_neff_hist(neff_ratio(univar.FFD.all.brm))+theme_bw()
```




```
mcmc_neff_hist(neff_ratio(univar.FFD_RR.all.brm))+theme_bw()
```



Autocorrelation: Takes very long / gets stuck, see later.

```
# mcmc_acf(univar.FFD_RR.all.brm,lags=20)
```

Assessing fit:

Posterior predictive checks: Compares observed data to simulated data from the posterior predictive distribution (if a model is a good fit, we should be able to use it to generate data that resemble the data that we observed).

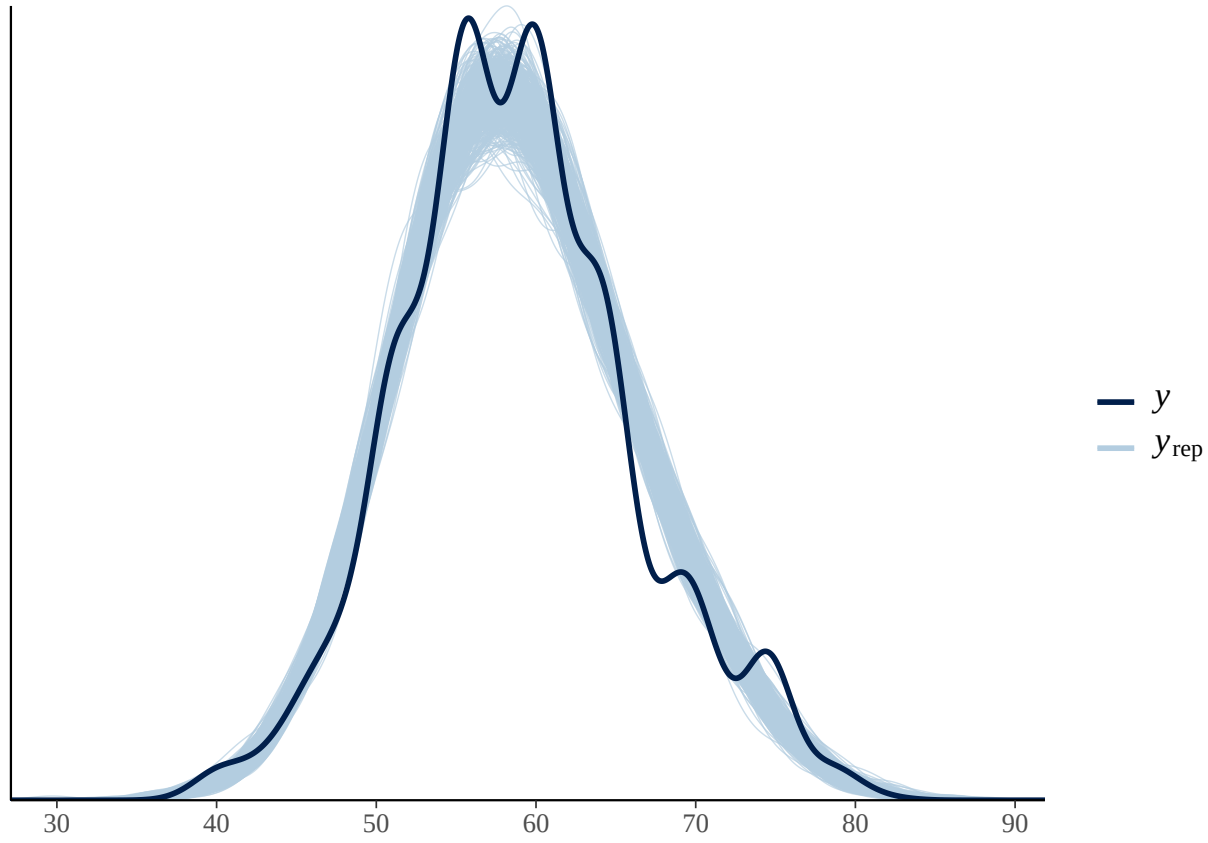
https://cran.r-project.org/web/packages/bayesplot/vignettes/graphical-ppcs.html#:~:text=MCMC%20diagnostics%20vigne
,Graphical%20posterior%20predictive%20checks%20(PPCs),Gabry%20et%20al%2C%202019).

```
y1<-subset(datadef,!is.na(FFD))$FFD # vector of outcome values
yrep1<-posterior_predict(univar.FFD_yearonly.all.brm, draws = 500)
yrep2<-posterior_predict(univar.FFD.all.brm, draws = 500)
yrep3<-posterior_predict(univar.FFD_RR.all.brm, draws = 500)
# matrices of draws from the posterior predictive distribution
```

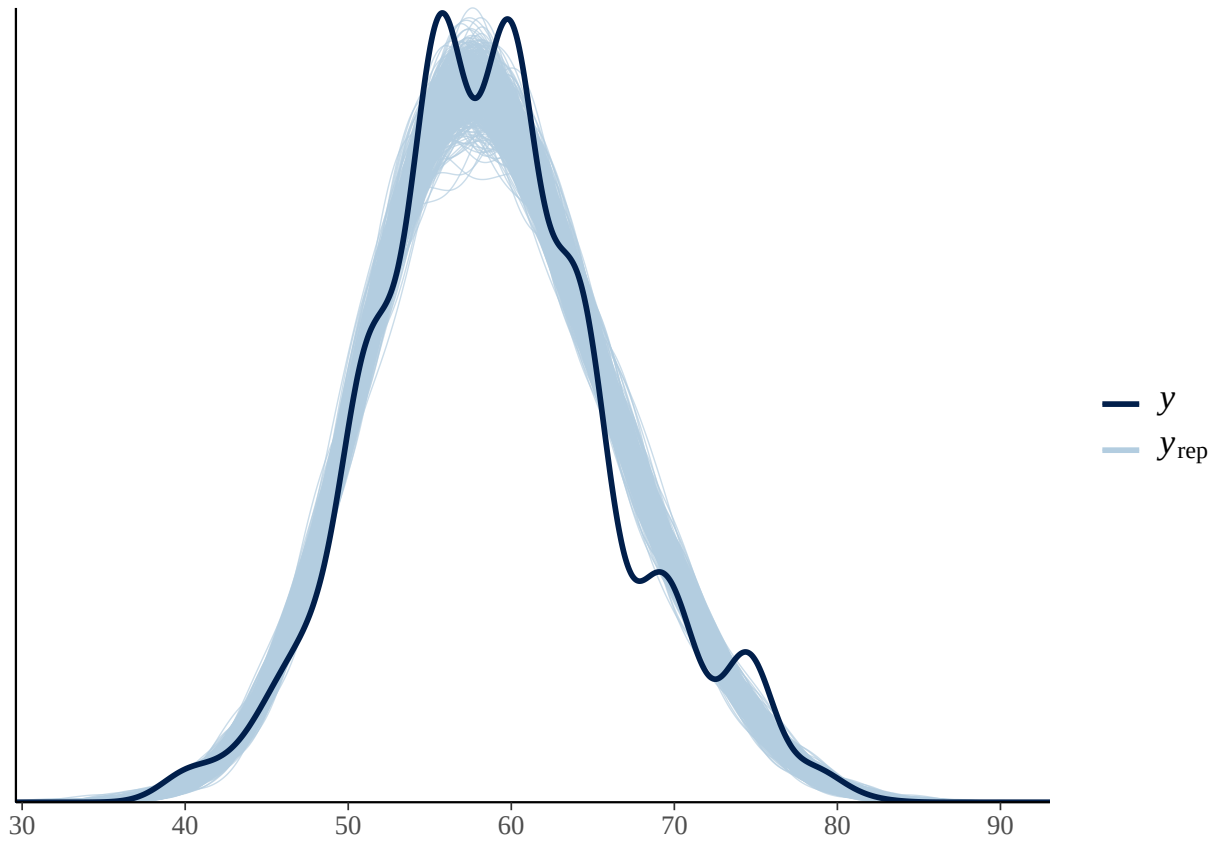
Each row of the matrix is a draw from the posterior predictive distribution, i.e. a vector with one element for each of the data points in y.

Comparison of the distribution of y and the distributions of the simulated datasets in the yrep matrix

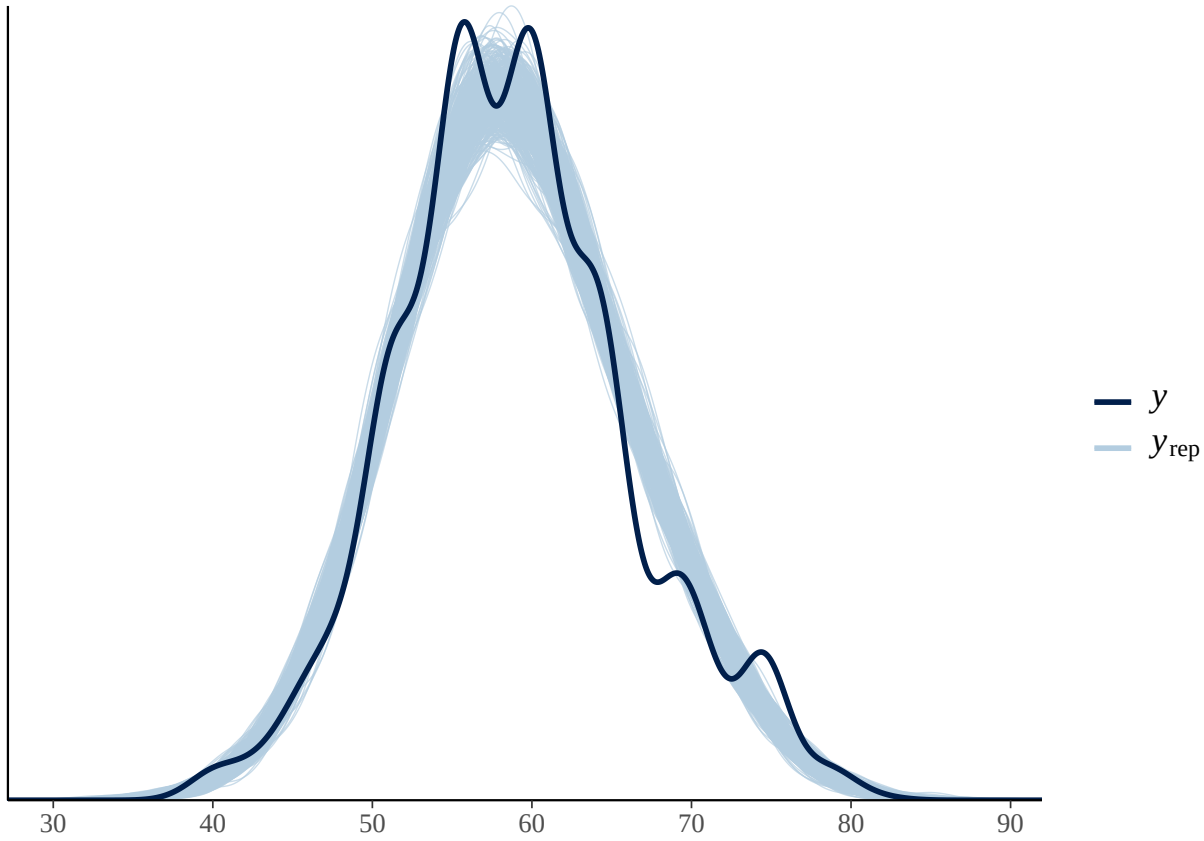
```
ppc_dens_overlay(y1, yrep1[1:500,])
```




```
ppc_dens_overlay(y1, yrep2[1:500,])
```

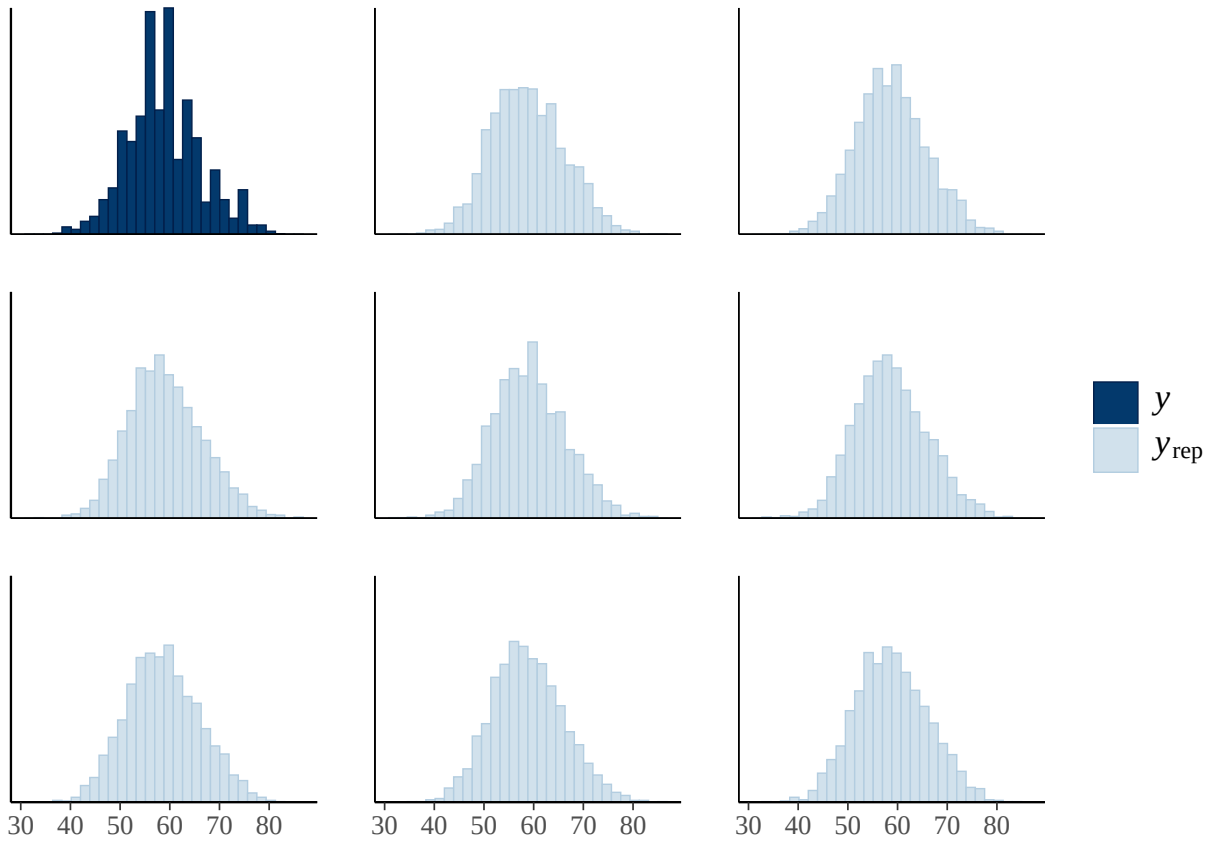



```
ppc_dens_overlay(y1, yrep3[1:500,])
```

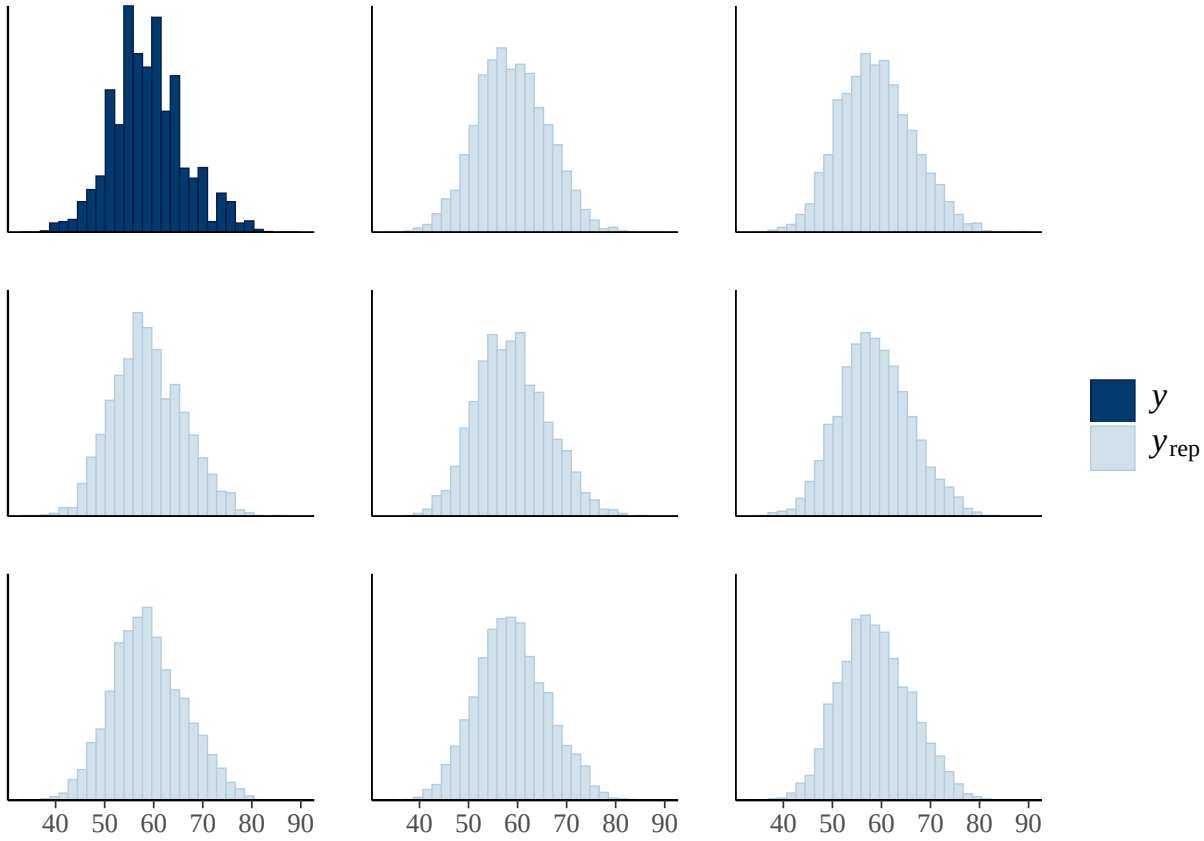


Separate histograms of y and some of the yrep datasets

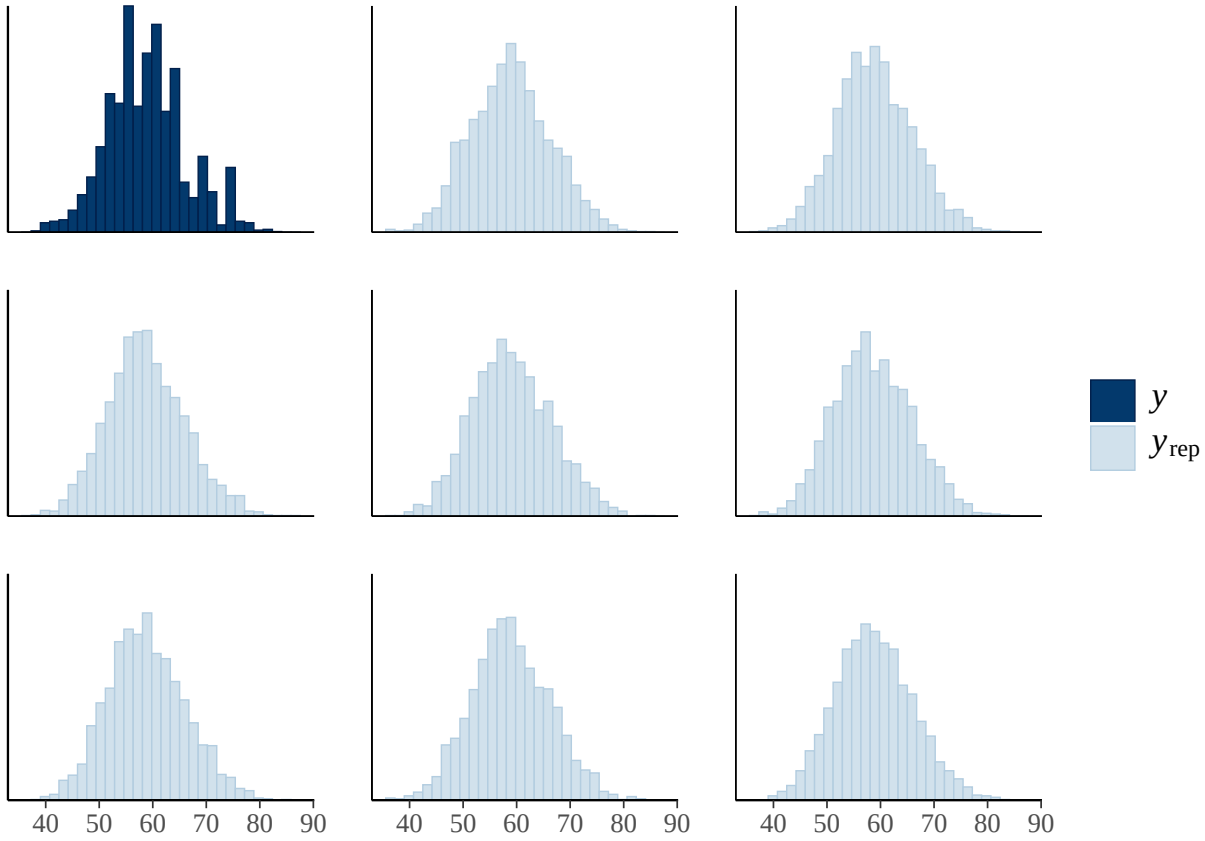
```
ppc_hist(y1, yrep1[1:8, ])
```




```
ppc_hist(y1, yrep2[1:8, ])
```

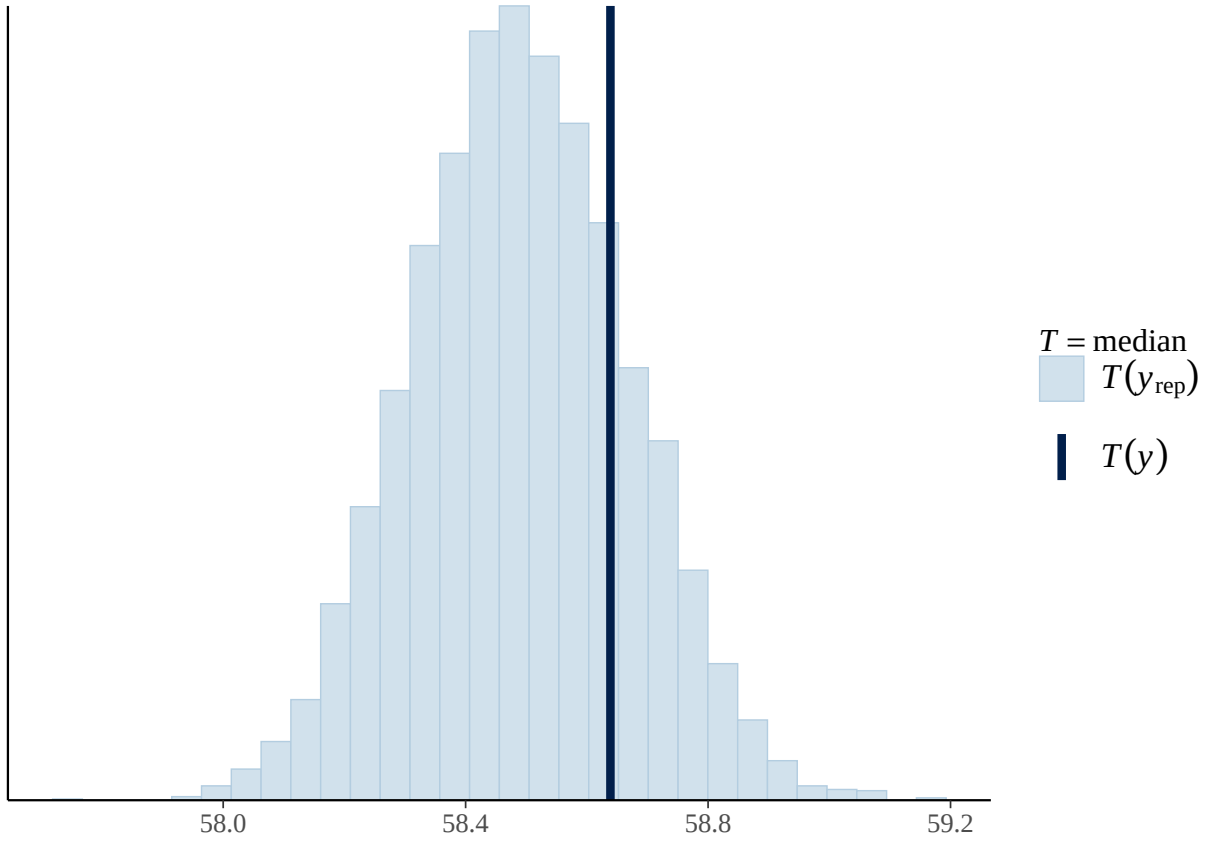



```
ppc_hist(y1, yrep3[1:8, ])
```

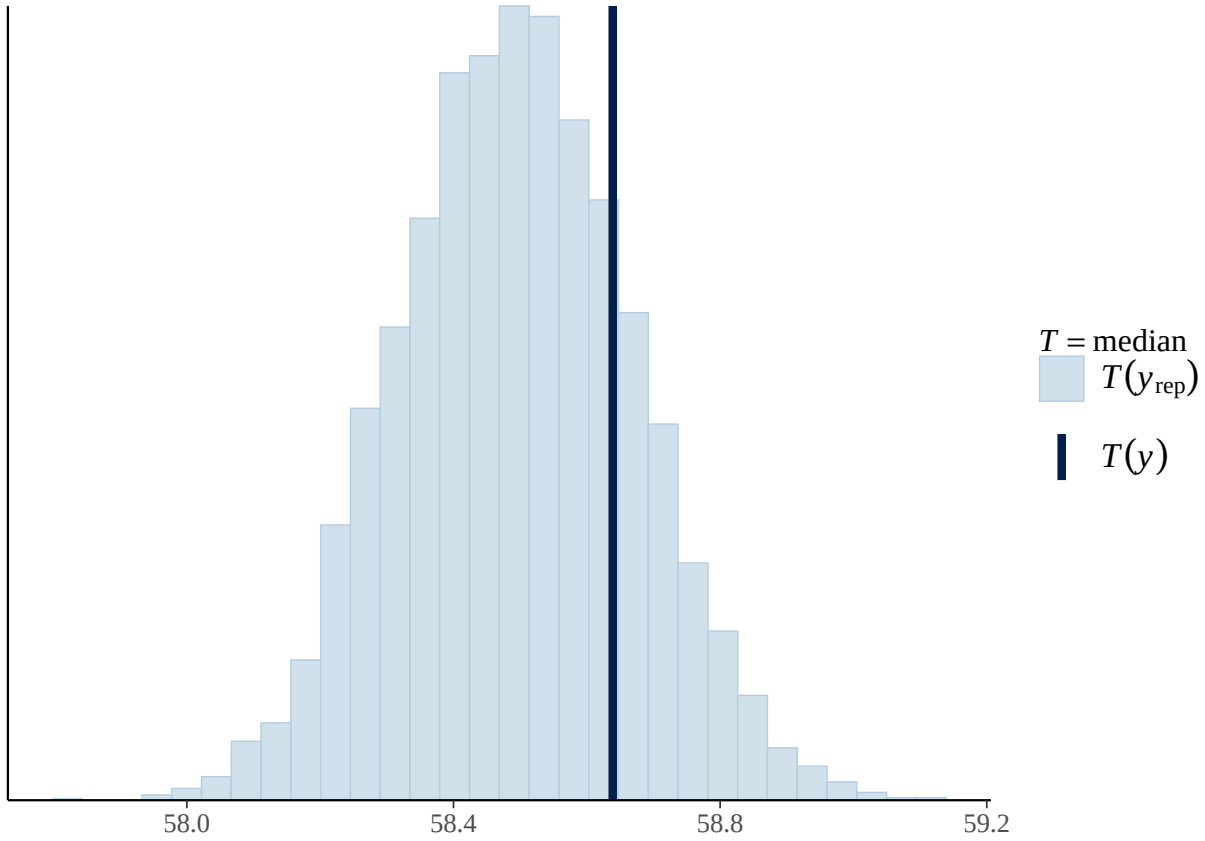


Is this good enough?

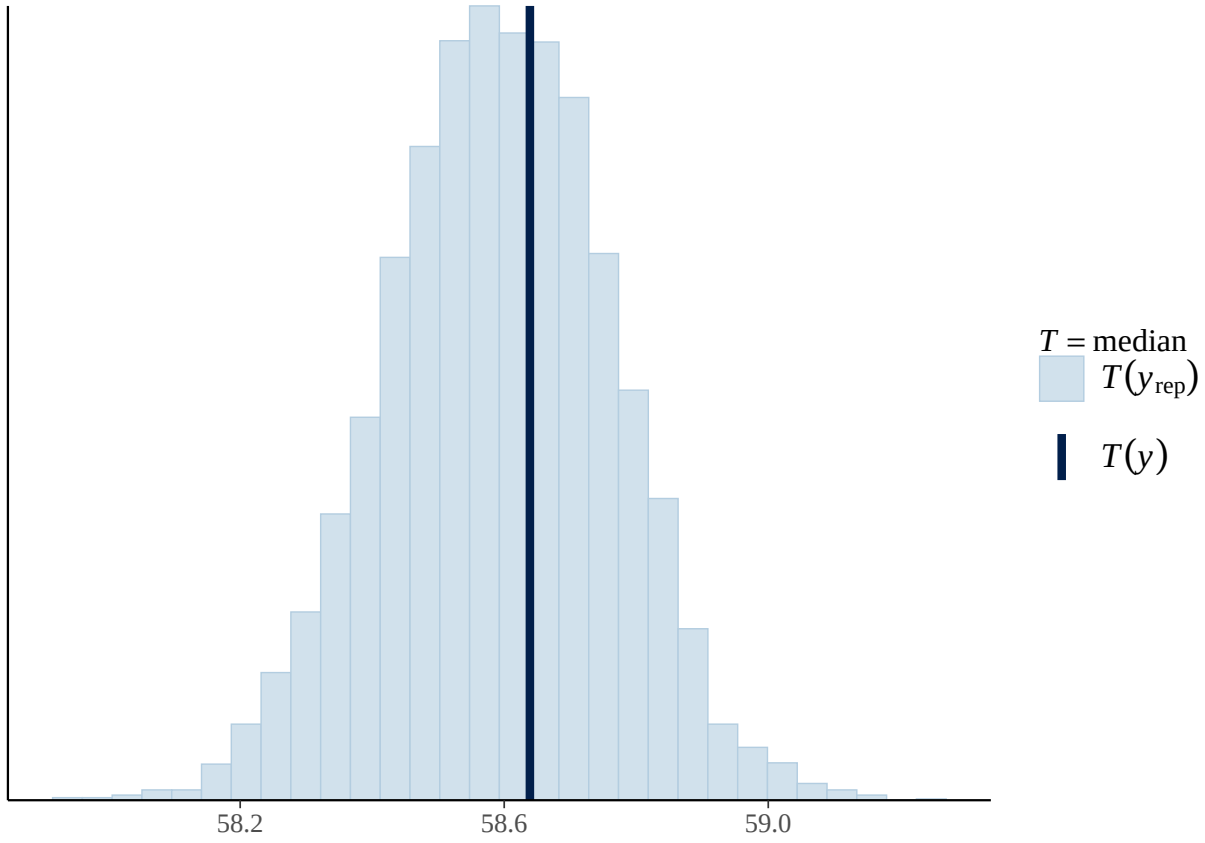
```
ppc_stat(y1,yrep1,stat="median")
```



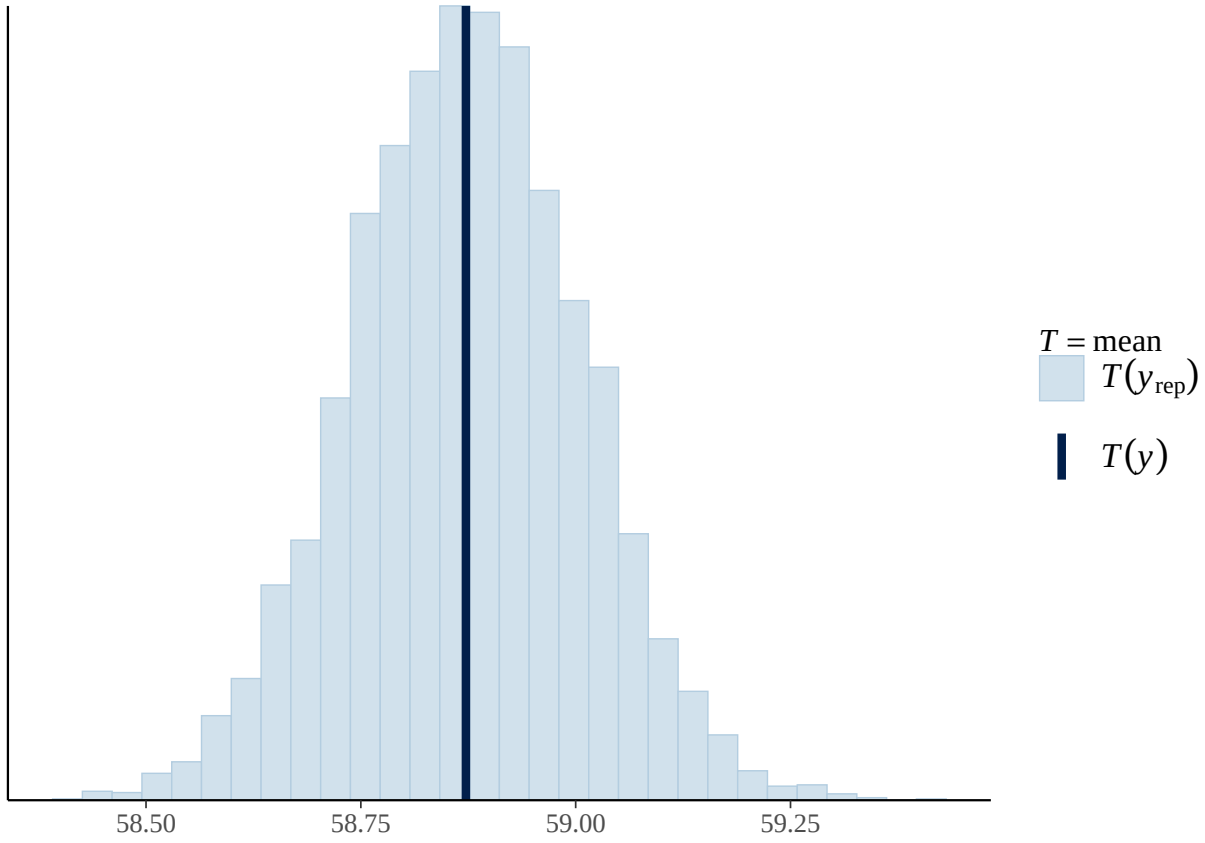

```
ppc_stat(y1,yrep2,stat="median")
```



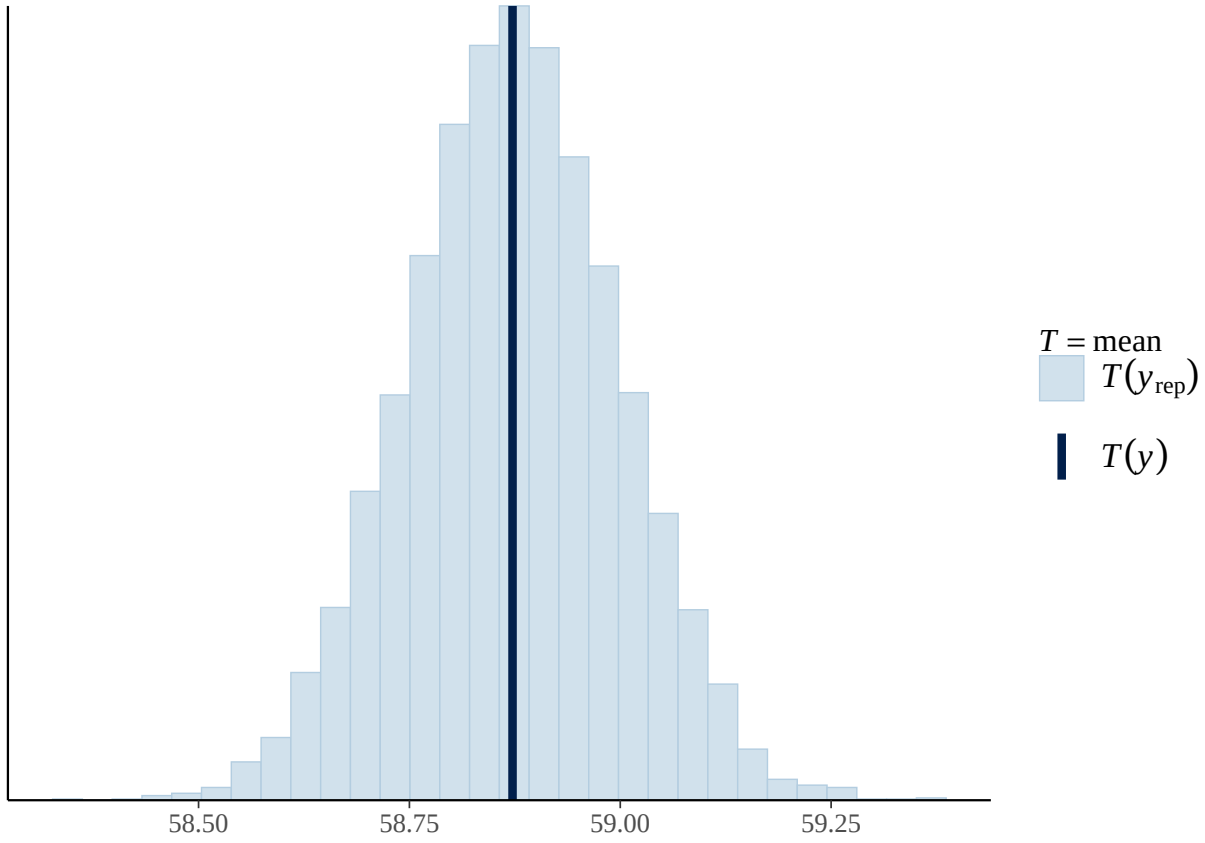

```
ppc_stat(y1,yrep3,stat="median")
```



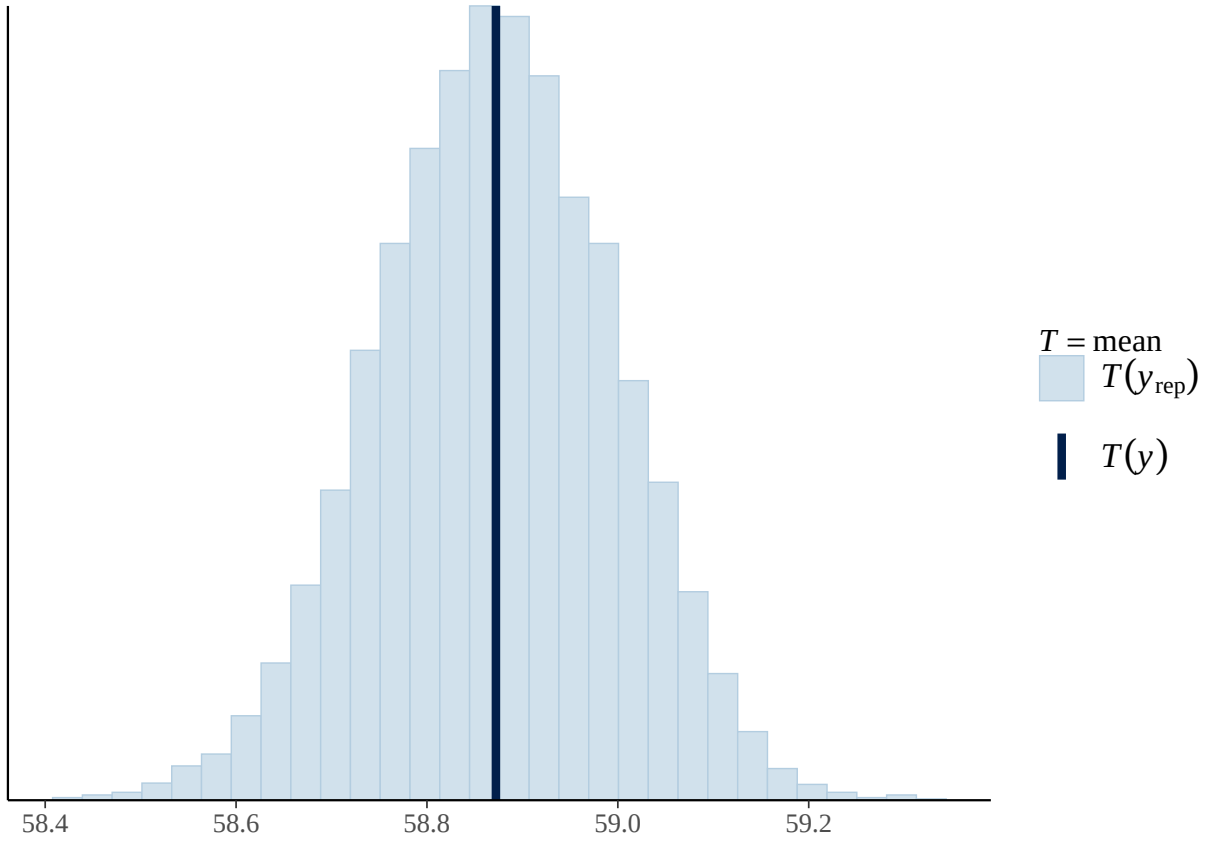

```
ppc_stat(y1,yrep1,stat="mean")
```



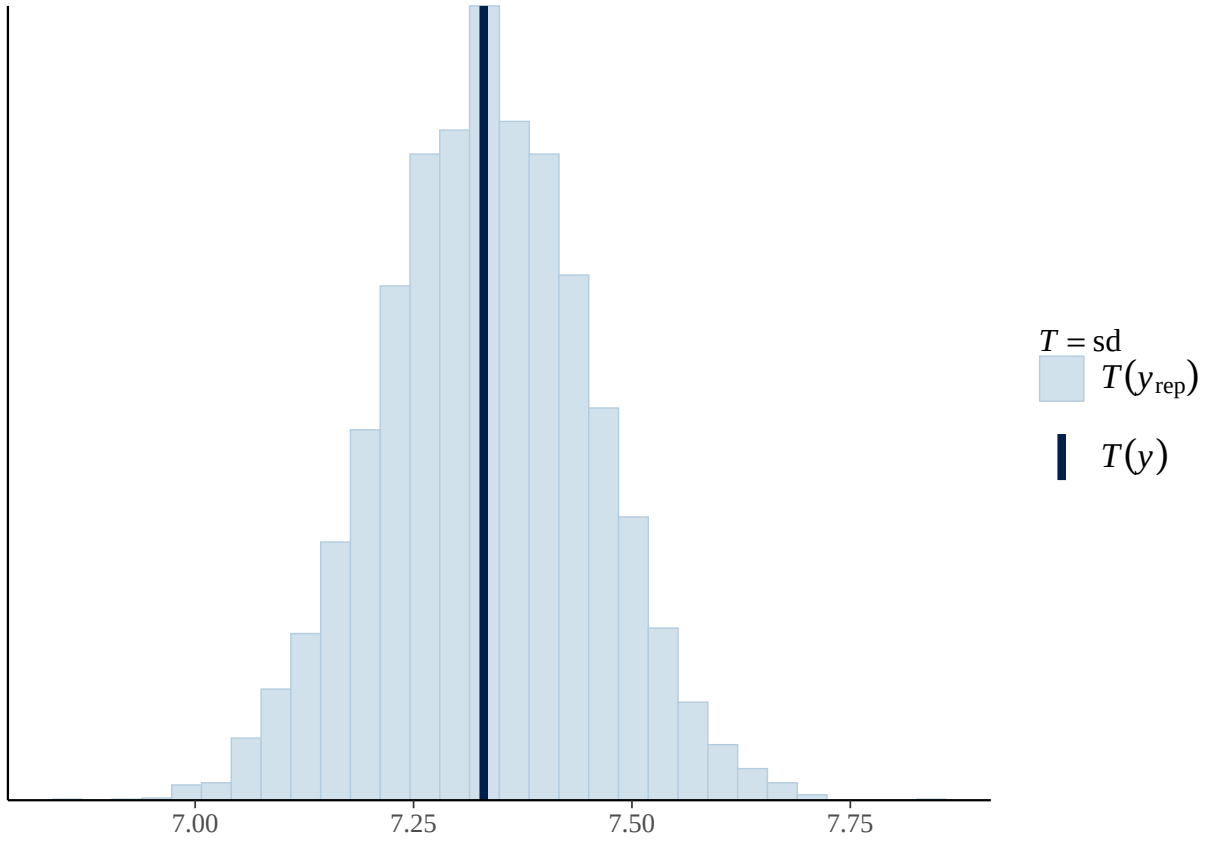

```
ppc_stat(y1,yrep2,stat="mean")
```



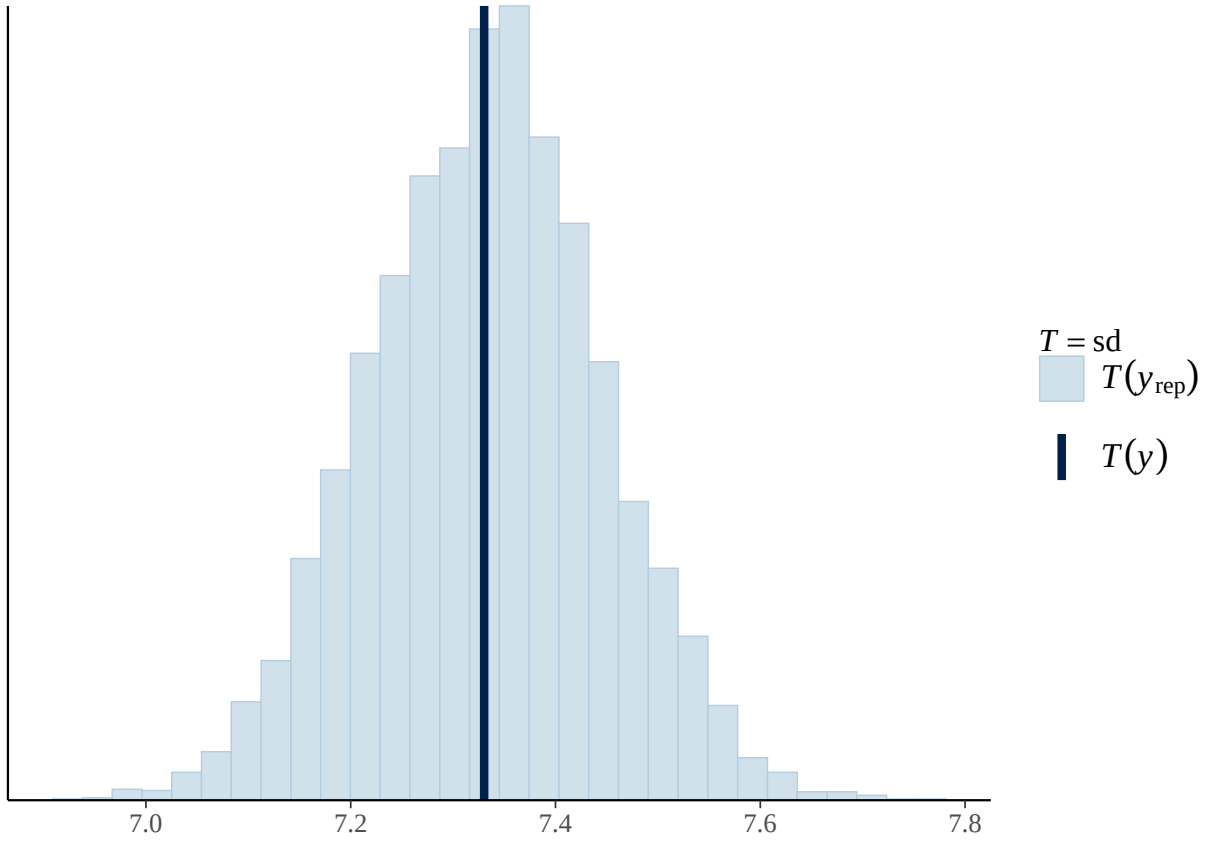

```
ppc_stat(y1,yrep3,stat="mean")
```



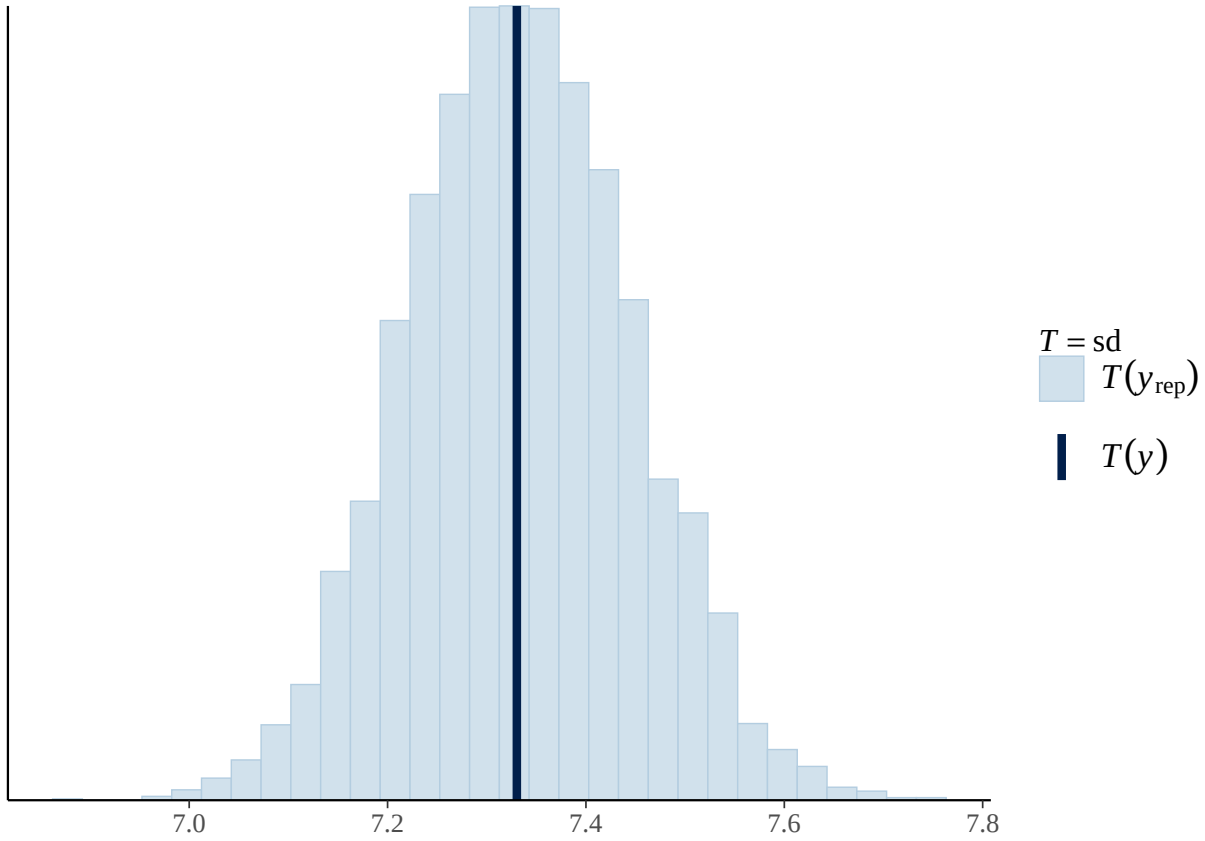

```
ppc_stat(y1,yrep1,stat="sd")
```



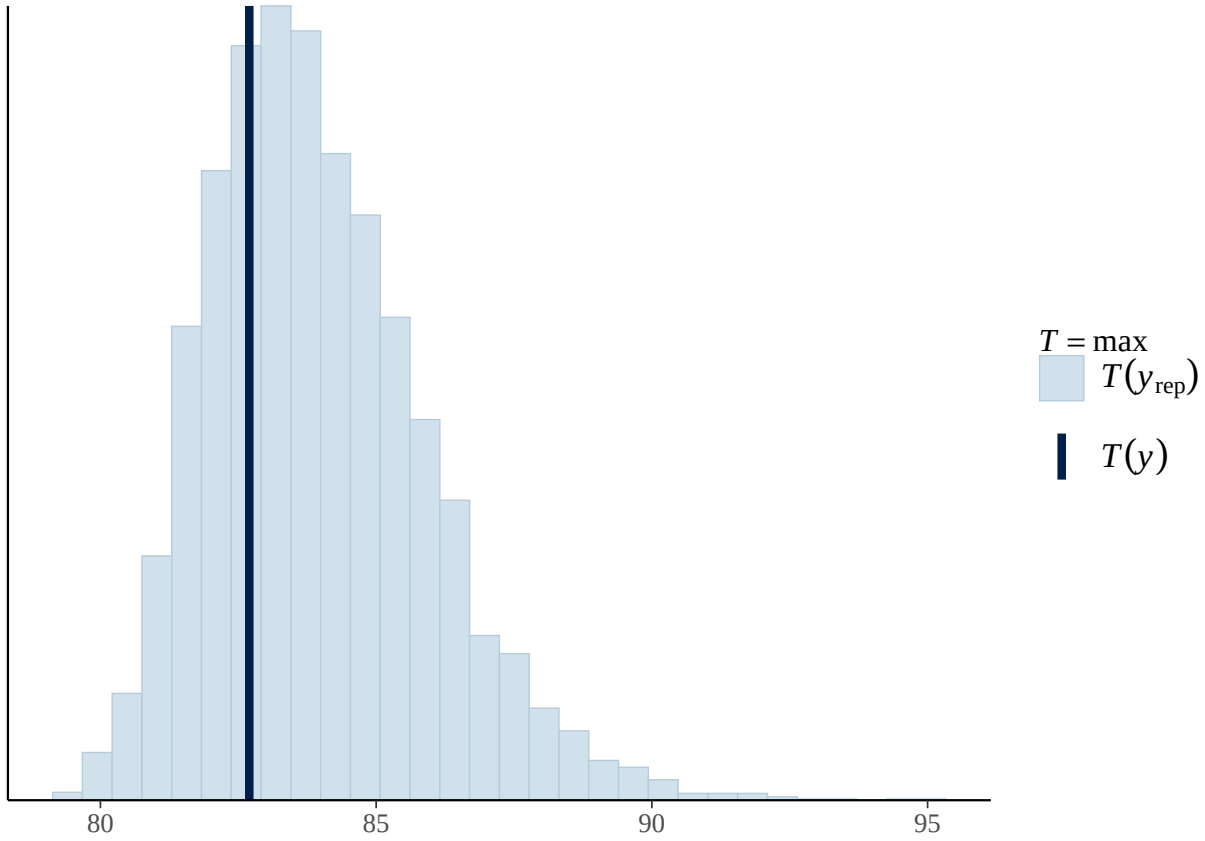

```
ppc_stat(y1,yrep2,stat="sd")
```



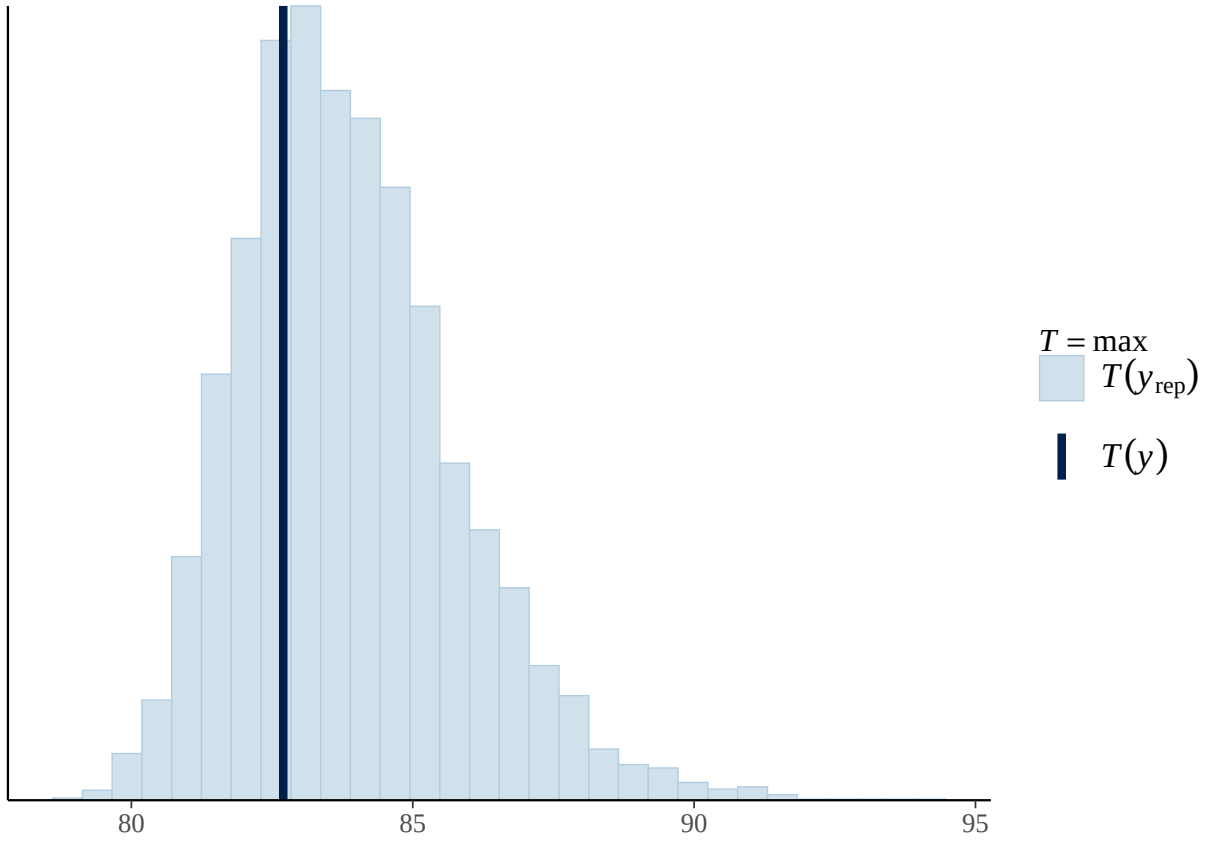

```
ppc_stat(y1,yrep3,stat="sd")
```



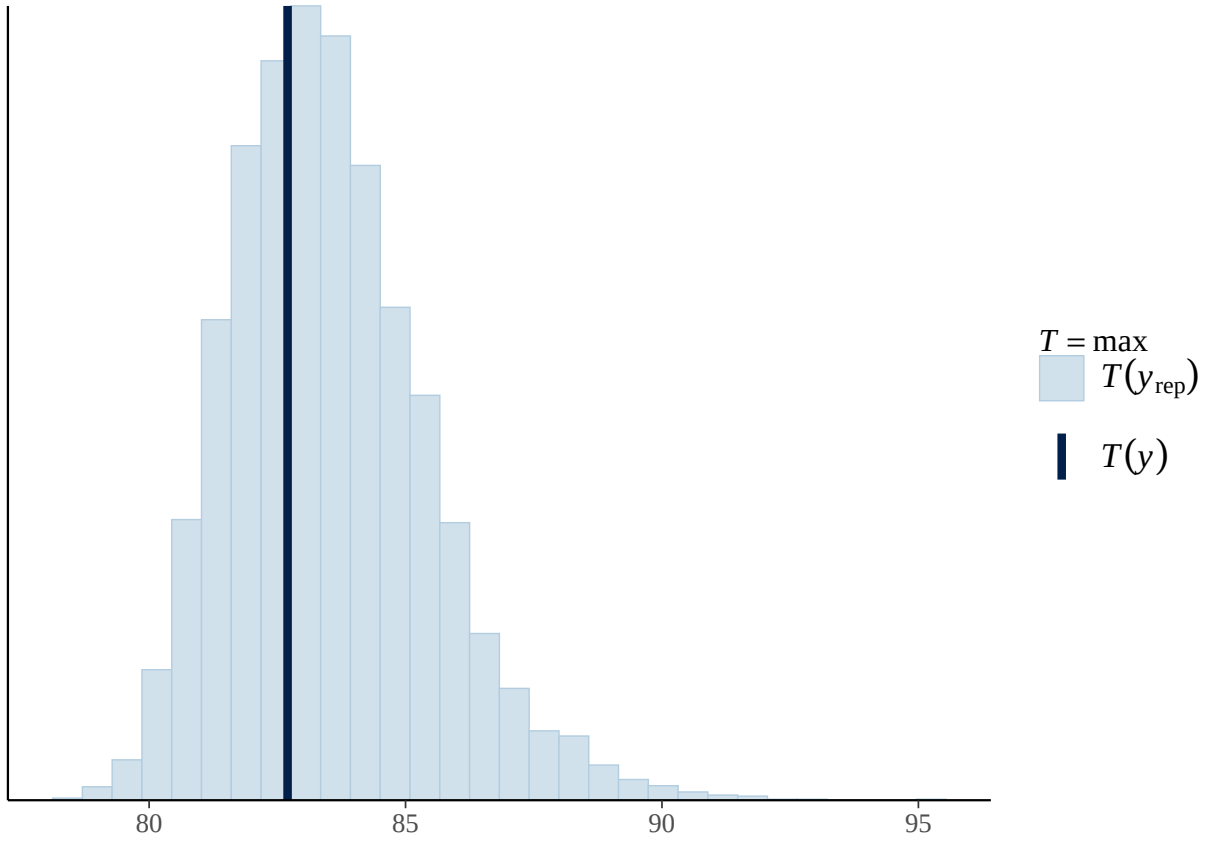

```
ppc_stat(y1,yrep1,stat="max")
```



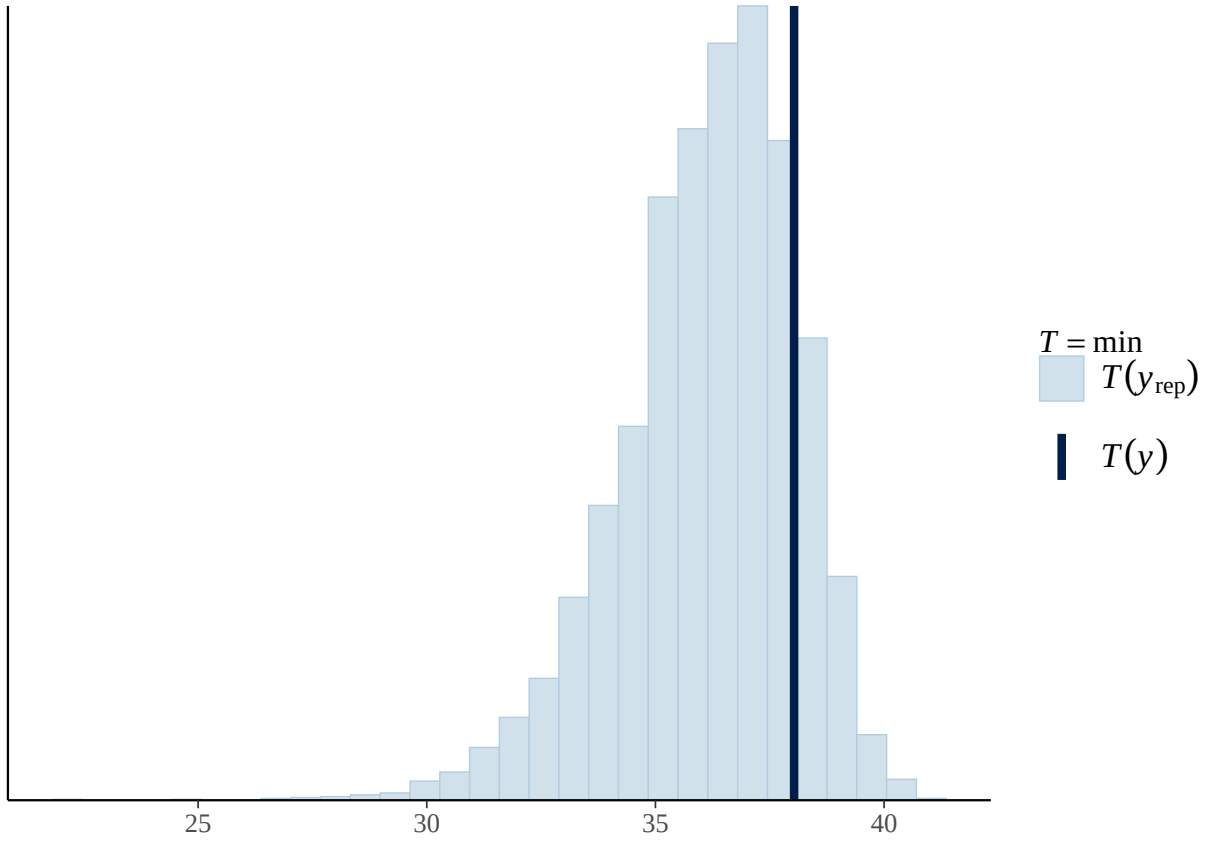

```
ppc_stat(y1,yrep2,stat="max")
```



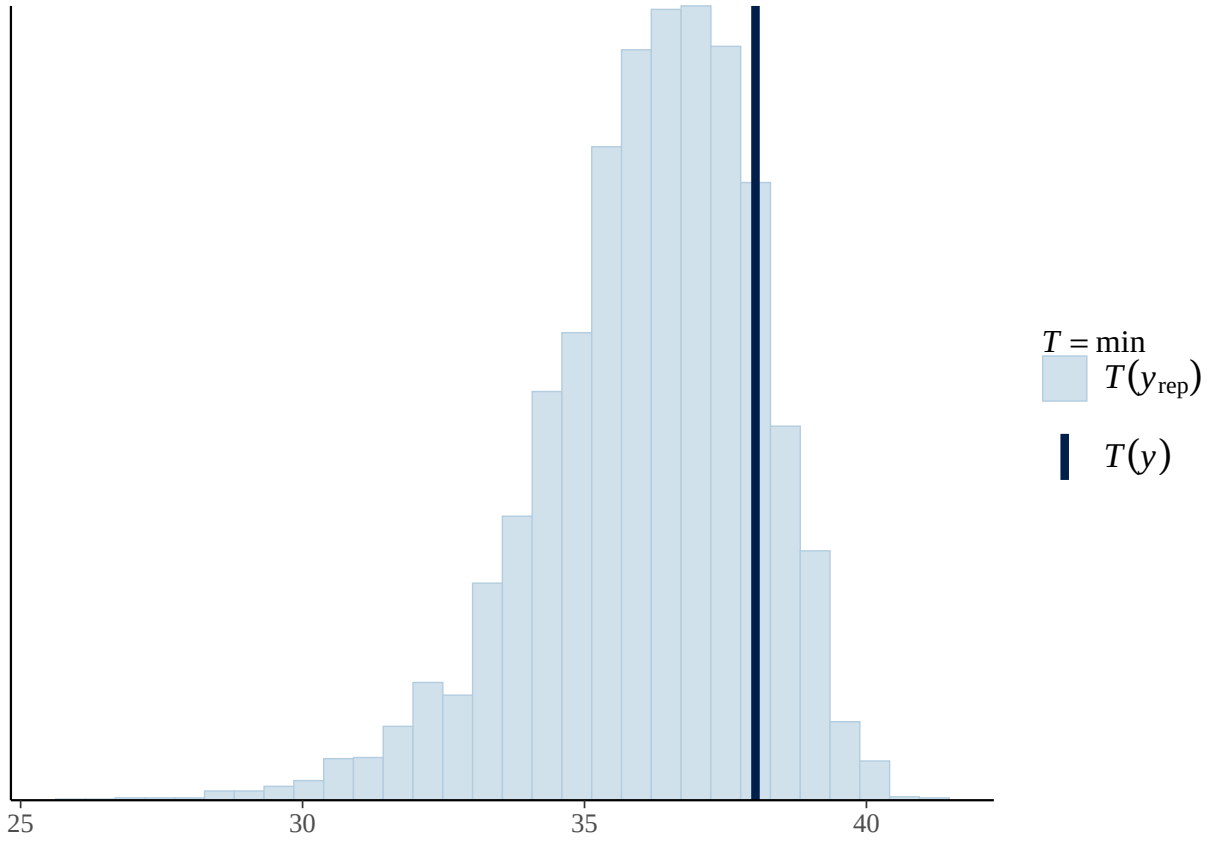

```
ppc_stat(y1,yrep3,stat="max")
```



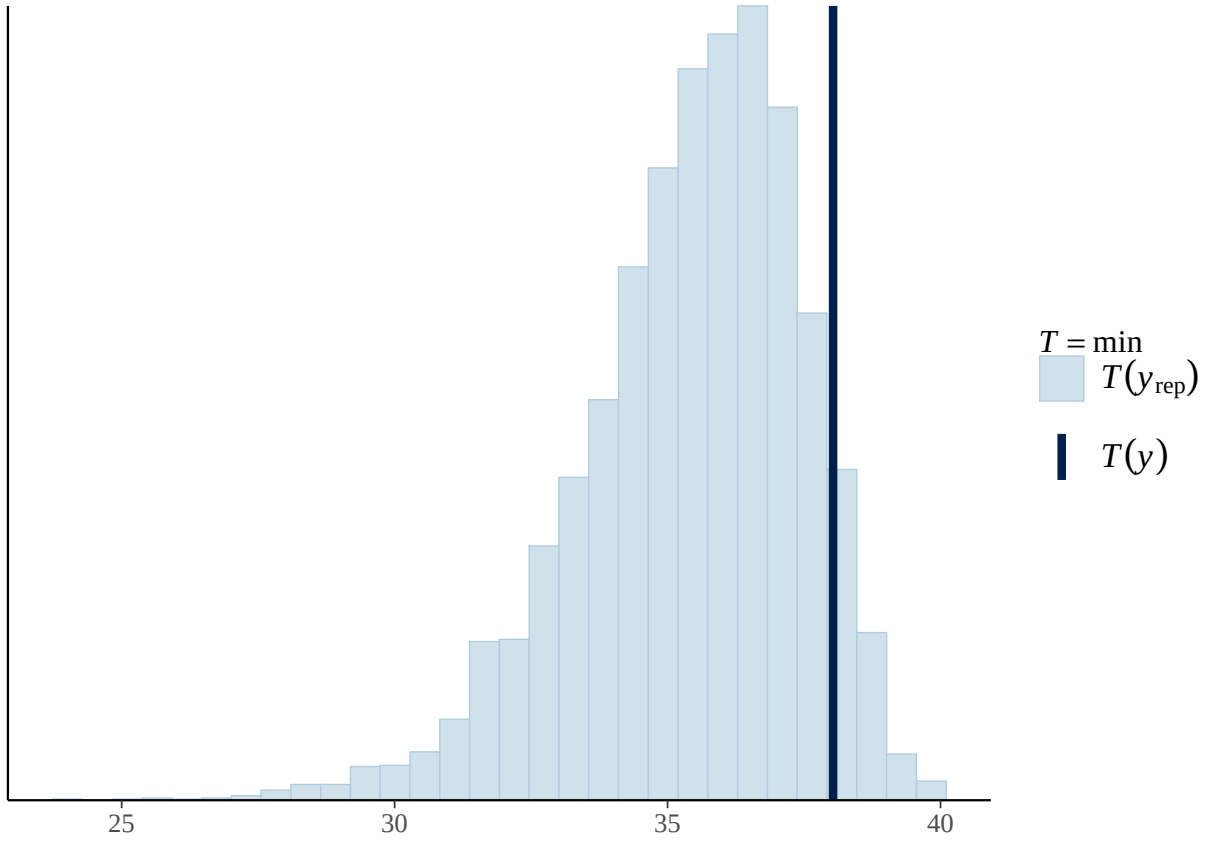

```
ppc_stat(y1,yrep1,stat="min")
```




```
ppc_stat(y1,yrep2,stat="min")
```




```
ppc_stat(y1,yrep3,stat="min")
```



This looks quite OK (worst is for the min).

Measure of fit: Bayesian R2, which looks at the model expected variance / (expected variance + residual variance).

```
bayes_R2(univar.FFD_yearonly.all.brm)
```

```
##     Estimate   Est.Error      Q2.5     Q97.5
## R2 0.5835971 0.008309393 0.5668564 0.5995905
```

```
bayes_R2(univar.FFD.all.brm)
```

```
##     Estimate   Est.Error      Q2.5     Q97.5
## R2 0.6320013 0.009207869 0.6128424 0.6492304
```

```
bayes_R2(univar.FFD_RR.all.brm)
```

```
##     Estimate   Est.Error      Q2.5     Q97.5
## R2 0.6456223 0.009494824 0.6264514 0.6639331
```

Leave-one-out cross validation (LOO):

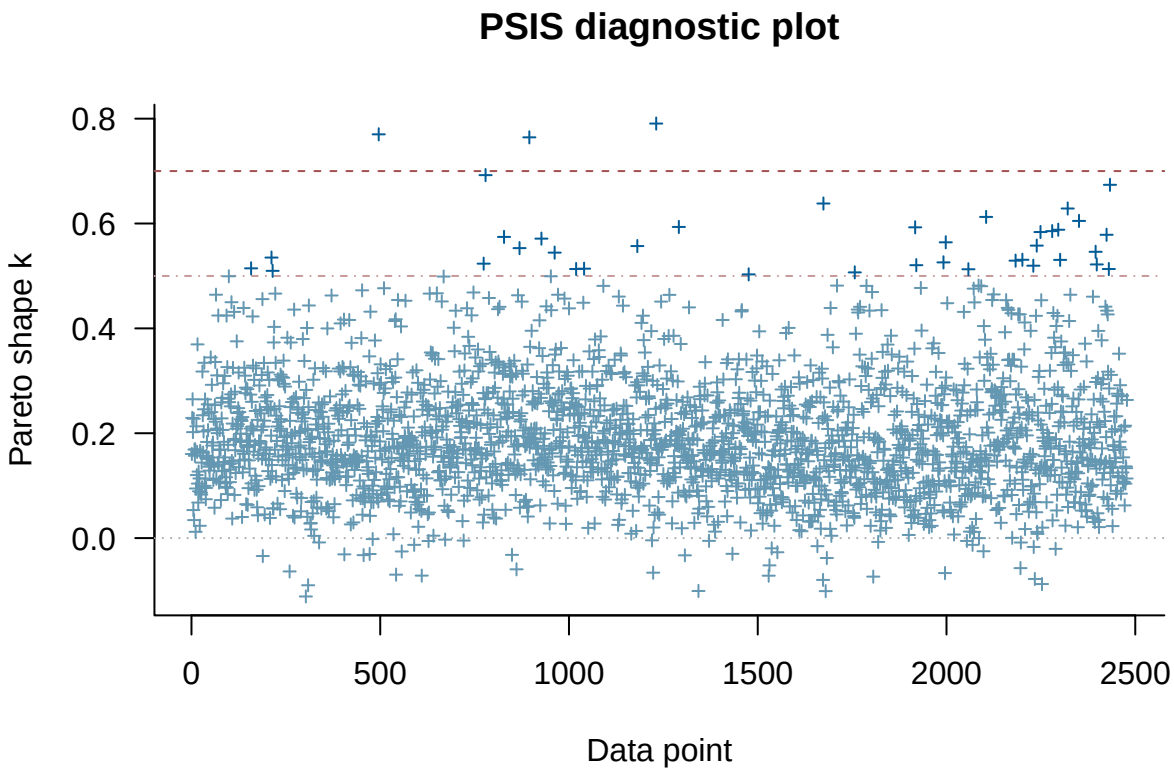
From https://www.monicaalexander.com/posts/2020-28-02-bayes_viz/

We are interested in estimating the out-of-sample predictive accuracy at each point i, when all we have to fit the model is data that without point i. We want to estimate the leave-one-out (LOO) posterior predictive

densities and a summary of these across all points, which is called the LOO expected log pointwise predictive density. The bigger the numbers, the better we are at predicting the left out point i.

The output mentions Pareto k estimates, which give an indication of how 'influential' each point is. The higher the value of k, the more influential the point. Values of k over 0.7 are not good, and suggest the need for model re-specification.

```
plot(loo1)
```



We have 3 obs with k > 0.7 - how bad is this?

Is this OK?

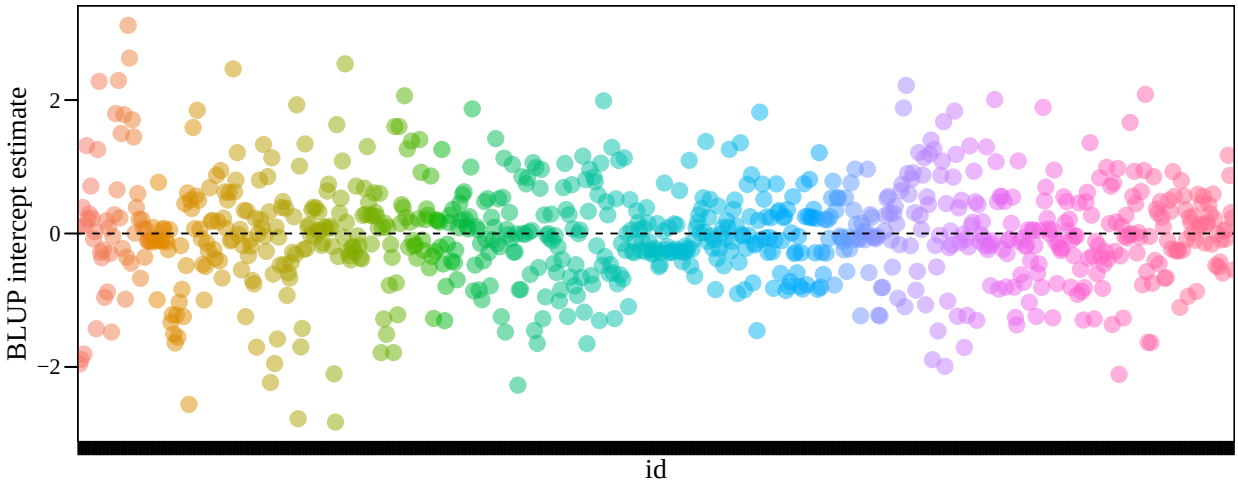
Extract BLUPs from random regression model

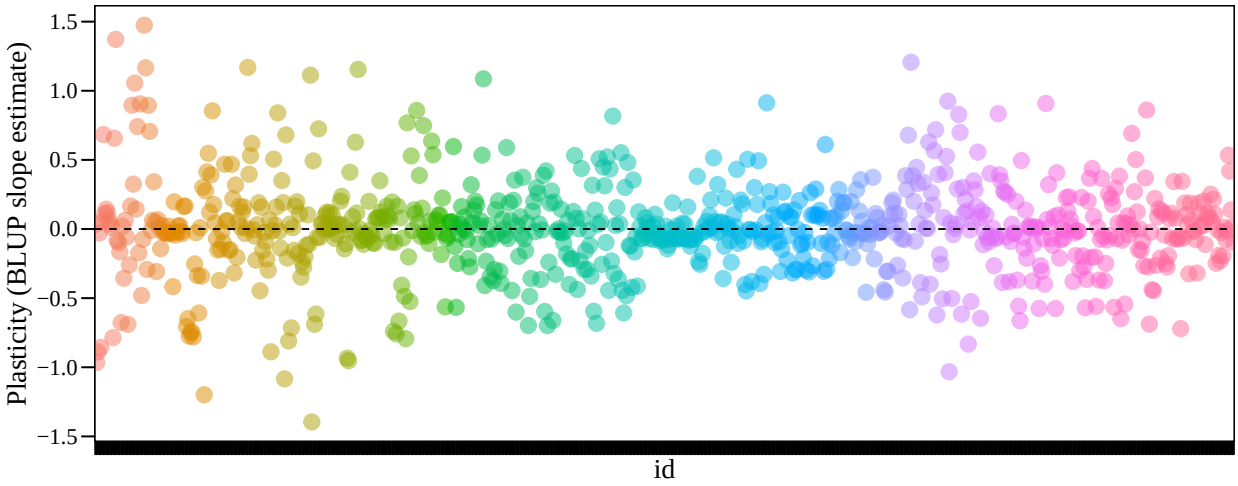
Code was checked by Piet.

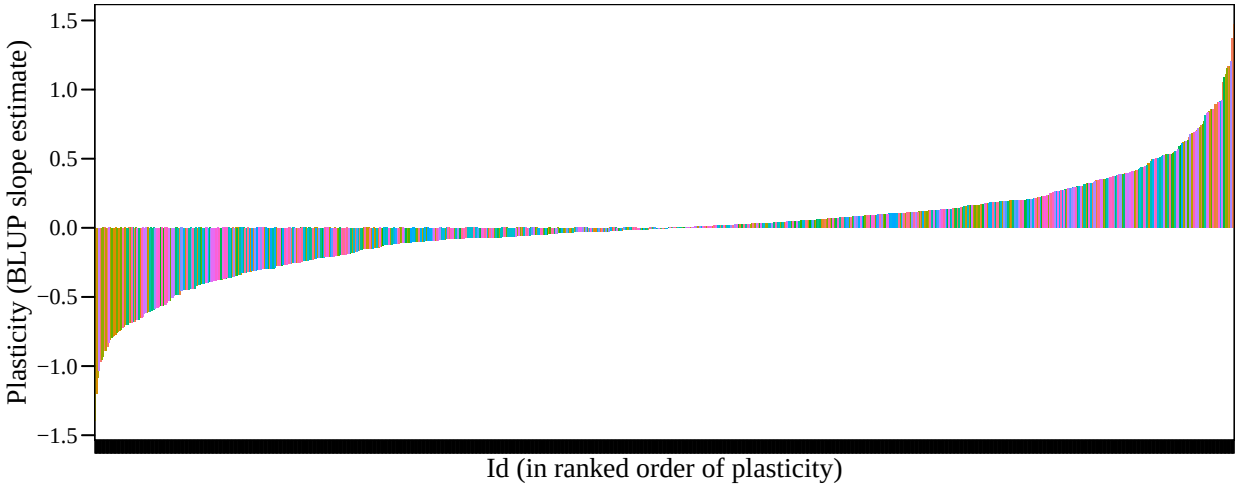
```
BLUPs_MCMC.all.brms  <- cbind(as.factor(c(1:837)),
                              as.data.frame(ranef(univar.FFD_RR.all.brm)$id)
                              [c(1,5)])
colnames(BLUPs_MCMC.all.brms) <- c("id", "intercept", "slope")
with(BLUPs_MCMC.all.brms,cor(intercept,slope)) # highly correlated!
```

```
## [1] 0.9623359
```

Plots with BLUPs







Bivariate models

1. mean_fitness_fl, no condition variable

Using the ID-syntax to specify fitness to be correlated with the intercept and slope of FFD on temperature - code has been checked by Piet.

Regarding distributions, I tried Poisson distribution for fitness, but not sure how eventual overdispersion is handled. I also tried adding an observation-level random effect, and using a negative binomial distribution. Results seem quite similar.

Poisson distribution

```
datadef<-left_join(datadef,datadef_total[c(1:3,9)])
# Add info on mean fitness and mean shoot volume
bf_FFD <- bf(FFD ~ cmean_4 + (cmean_4|ID1|id) + (1|year)) # Set up model formula
bf_fitness <- bf(round(mean_fitness_fl) ~  (1|ID1|id)) # Set up model formula
# Specifying group-level effects of the same grouping factor (id here)
# to be correlated across formulas
# Expand the | operator into |<ID>|, where <ID> can be any value (ID1 here)
# Group-level terms with the same ID1 will be modeled as correlated
# if they share same grouping factor(s)
```

```
bivar1.all.brm.pois<-brm(bf_FFD + bf_fitness, family = c(gaussian, poisson),
                         data = datadef,
                         warmup = 1000,iter = 4000,chains=4,thin=2,
                         inits = "random",seed = 12345,cores = my.cores,
                         save_all_pars=TRUE)
# Total of 6000 post-warmup samples
```

```
summary(bivar1.all.brm.pois)
```

```
##  Family: MV(gaussian, poisson)
##   Links: mu = identity; sigma = identity
##          mu = log
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_fl) ~ (1 | ID1 | id)
##    Data: datadef (Number of observations: 2478)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
##
## Group-Level Effects:
## ~id (Number of levels: 837)
##                                                 Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                   1.63      0.14     1.35
## sd(FFD_cmean_4)                                     0.78      0.13     0.53
## sd(roundmeanfitnessfl_Intercept)                    1.37      0.05     1.28
## cor(FFD_Intercept,FFD_cmean_4)                      0.70      0.13     0.41
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)    -0.53      0.07    -0.67
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)      -0.03      0.13    -0.29
##                                                 u-95% CI Rhat Bulk_ESS Tail_ESS
```

```
## sd(FFD_Intercept)                                    1.92 1.00     2782     4369
## sd(FFD_cmean_4)                                      1.05 1.00     1997     4396
## sd(roundmeanfitnessfl_Intercept)                     1.48 1.00     3029     4381
## cor(FFD_Intercept,FFD_cmean_4)                       0.92 1.01      513     2118
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)     -0.39 1.00      855     1779
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)        0.23 1.00     1287     1890
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.12      0.86     3.76     7.17 1.00     2467     3688
##
## Population-Level Effects:
##                             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## FFD_Intercept                  58.72      1.12    56.52    60.98 1.00     1964
## roundmeanfitnessfl_Intercept    0.76      0.06     0.65     0.87 1.00     4507
## FFD_cmean_4                    -2.37      0.85    -4.06    -0.71 1.00     2475
##                             Tail_ESS
## FFD_Intercept                   3071
## roundmeanfitnessfl_Intercept    5035
## FFD_cmean_4                     3638
##
## Family Specific Parameters:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma_FFD     4.35      0.07     4.21     4.50 1.00     2036     4072
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Negative binomial distribution

```
bivar1.all.brm.nb<-brm(bf_FFD + bf_fitness, family = c(gaussian, negbinomial),
                       data = datadef,warmup = 1000,iter = 4000,chains=4,thin=2,
                         inits = "random",seed = 12345,cores = my.cores,
                       save_all_pars=TRUE)
```

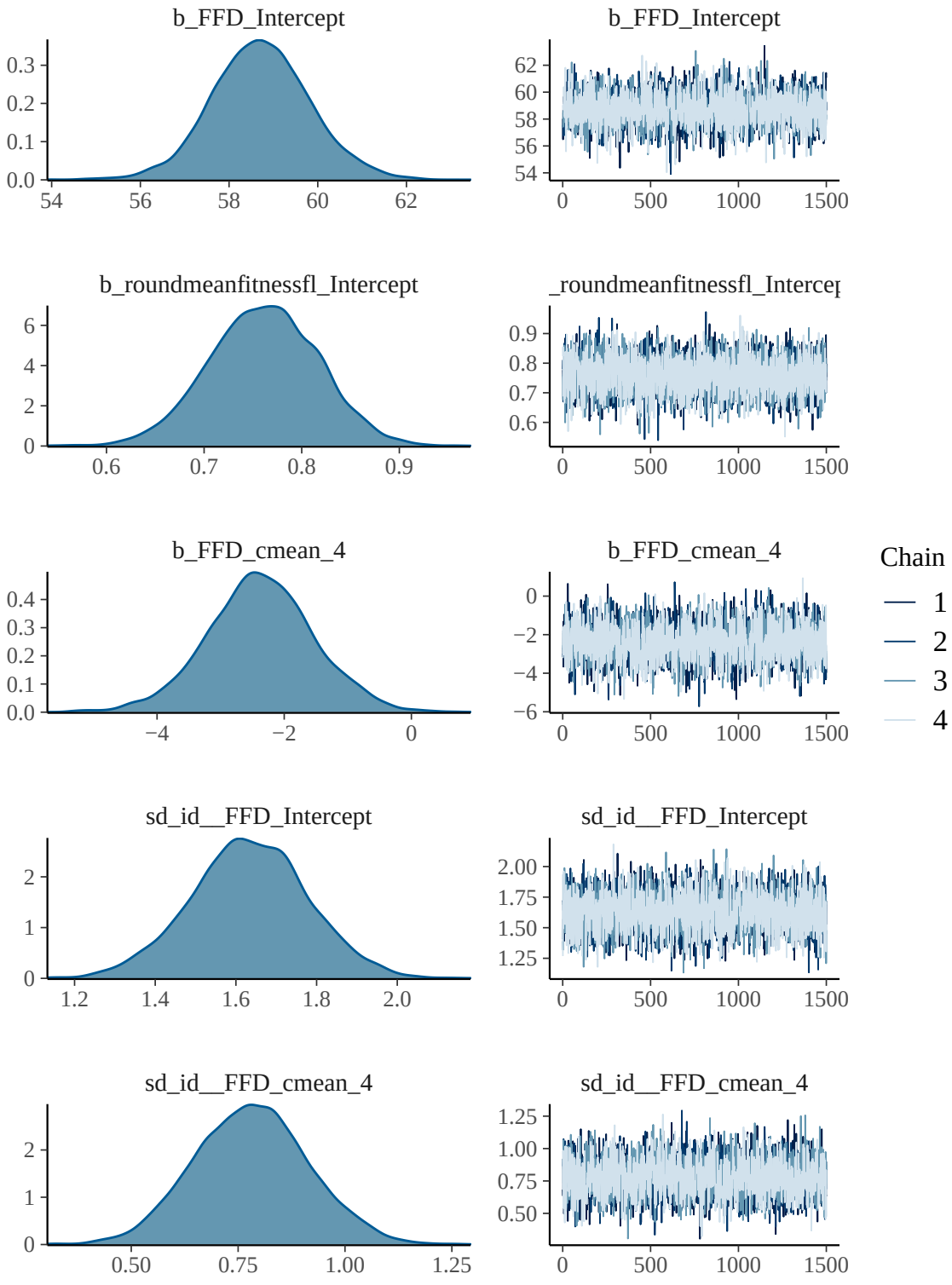
```
summary(bivar1.all.brm.nb)
```

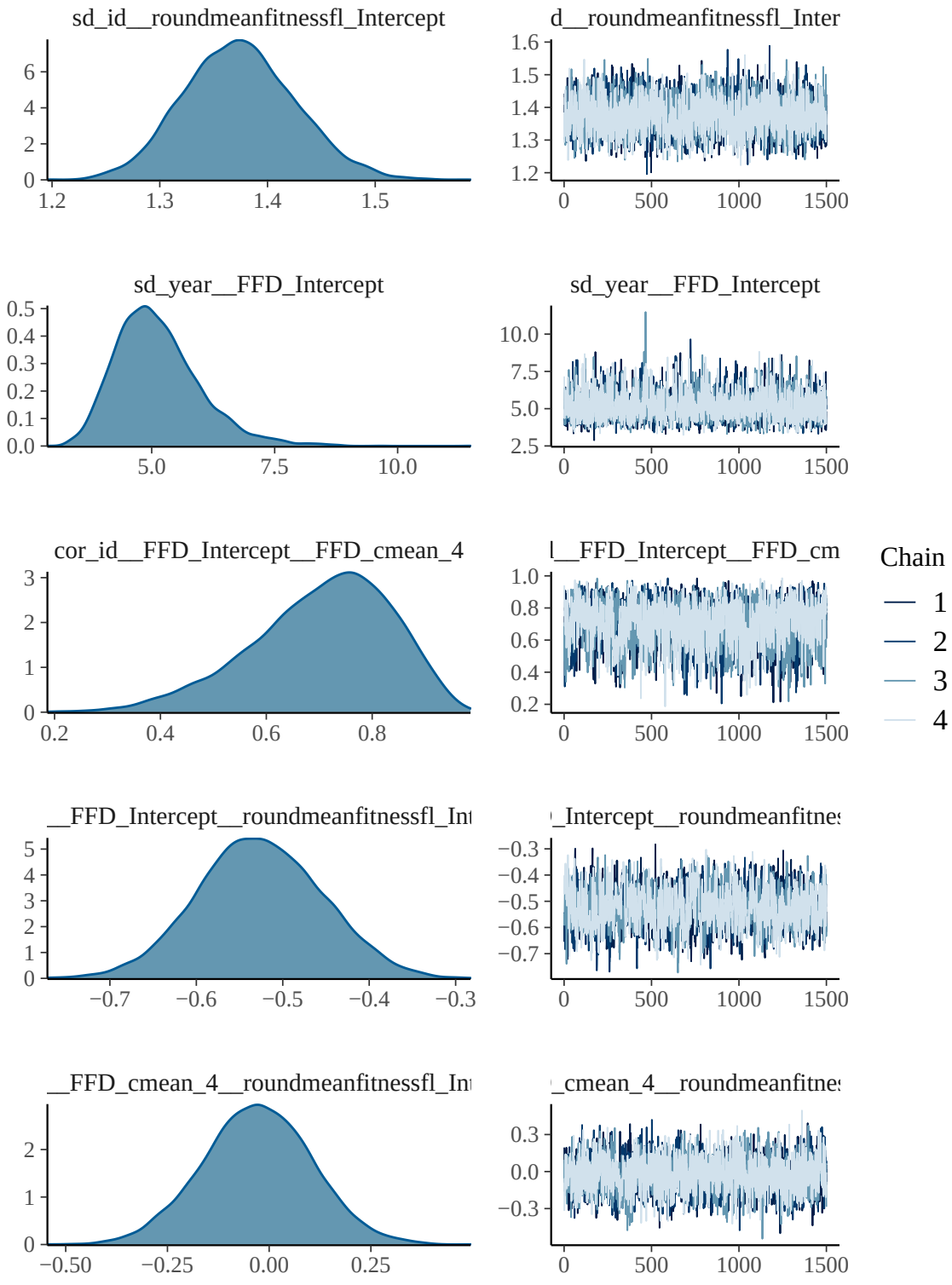
```
##  Family: MV(gaussian, negbinomial)
##   Links: mu = identity; sigma = identity
##          mu = log; shape = identity
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_fl) ~ (1 | ID1 | id)
##    Data: datadef (Number of observations: 2478)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
##
## Group-Level Effects:
## ~id (Number of levels: 837)
##                                              Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                1.64      0.15     1.34
```

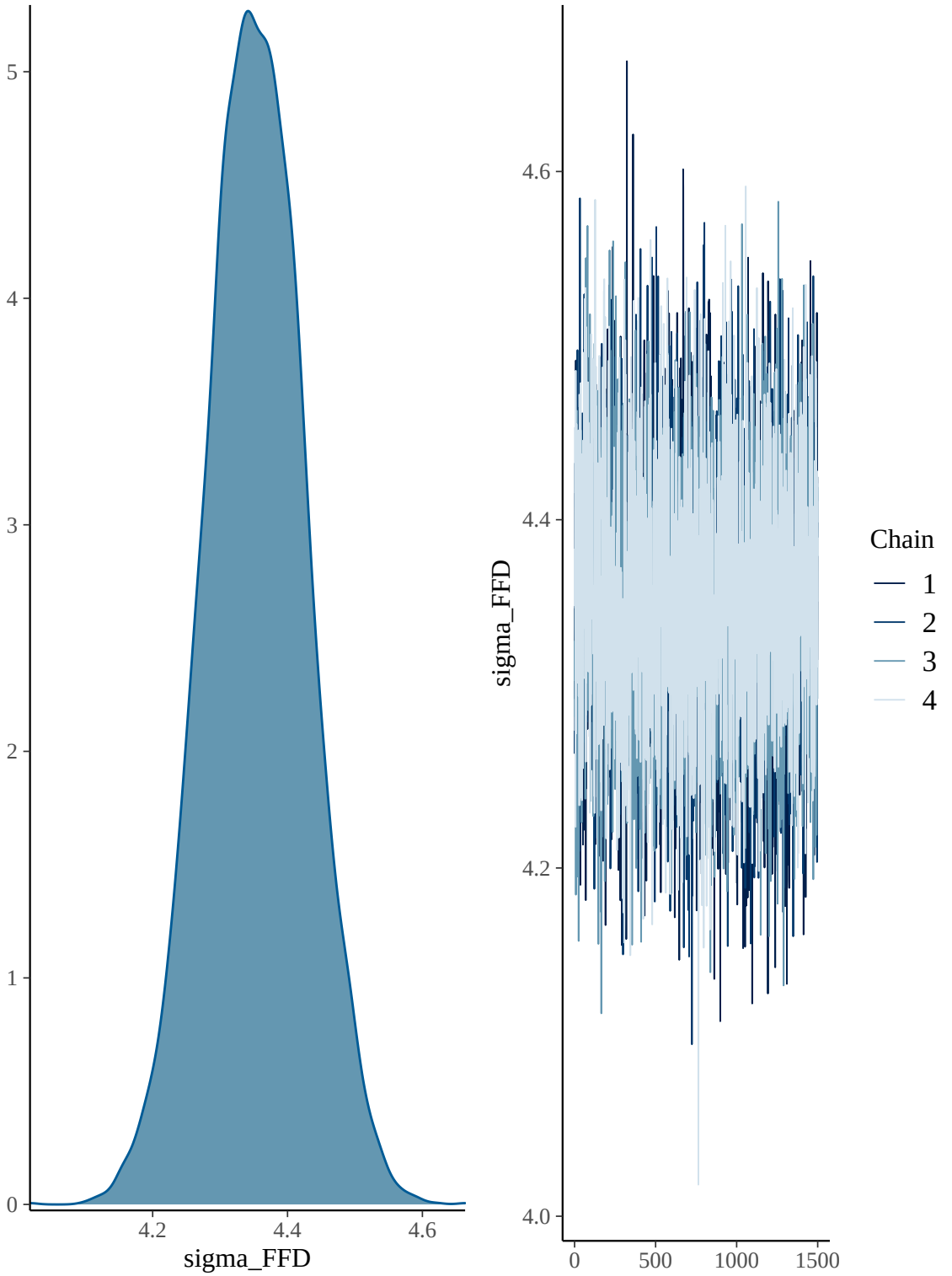
```
## sd(FFD_cmean_4)                                      0.79      0.13     0.53
## sd(roundmeanfitnessfl_Intercept)                     1.38      0.05     1.28
## cor(FFD_Intercept,FFD_cmean_4)                       0.69      0.14     0.39
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)     -0.53      0.07    -0.67
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)       -0.05      0.14    -0.32
##                                                  u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)                                    1.93 1.00     1730     3232
## sd(FFD_cmean_4)                                      1.05 1.00     1920     4218
## sd(roundmeanfitnessfl_Intercept)                     1.48 1.00     3083     4380
## cor(FFD_Intercept,FFD_cmean_4)                       0.92 1.01      409     2057
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)     -0.38 1.00     1113     2049
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)        0.22 1.00      788     1232
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.09      0.84     3.76     7.04 1.00     2897     3957
##
## Population-Level Effects:
##                              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## FFD_Intercept                   58.76      1.10    56.60    61.01 1.00     2173
## roundmeanfitnessfl_Intercept     0.76      0.06     0.65     0.87 1.00     5297
## FFD_cmean_4                     -2.35      0.82    -3.94    -0.69 1.00     2504
##                              Tail_ESS
## FFD_Intercept                    3260
## roundmeanfitnessfl_Intercept     5417
## FFD_cmean_4                      3962
##
## Family Specific Parameters:
##                          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sigma_FFD                    4.35      0.07     4.20     4.50 1.00     1640
## shape_roundmeanfitnessfl   670.43    188.30   379.86  1125.08 1.00     4261
##                          Tail_ESS
## sigma_FFD                    4270
## shape_roundmeanfitnessfl     4661
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Model evaluation: Compare models

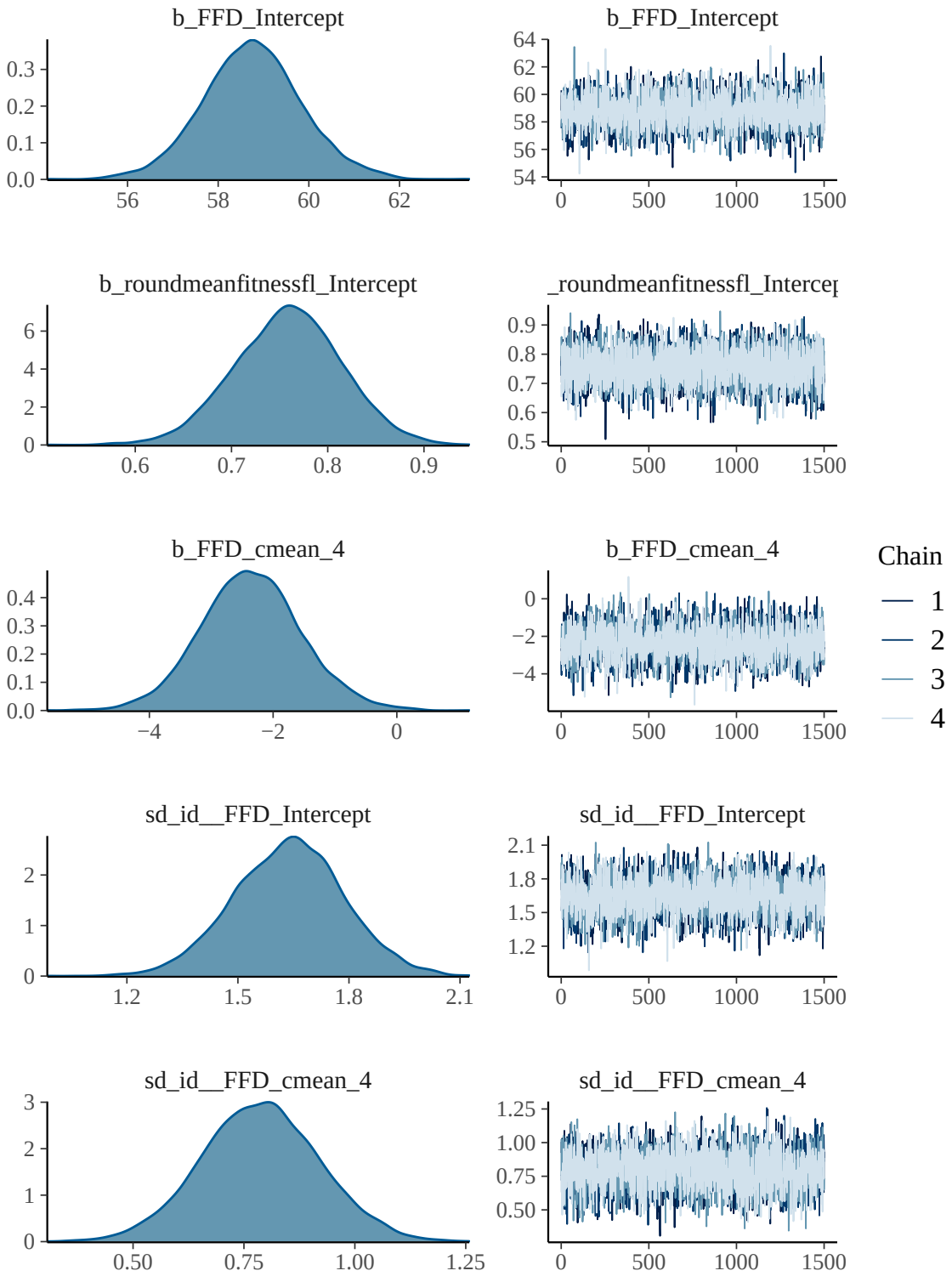
```
plot(bivar1.all.brm.pois)
```

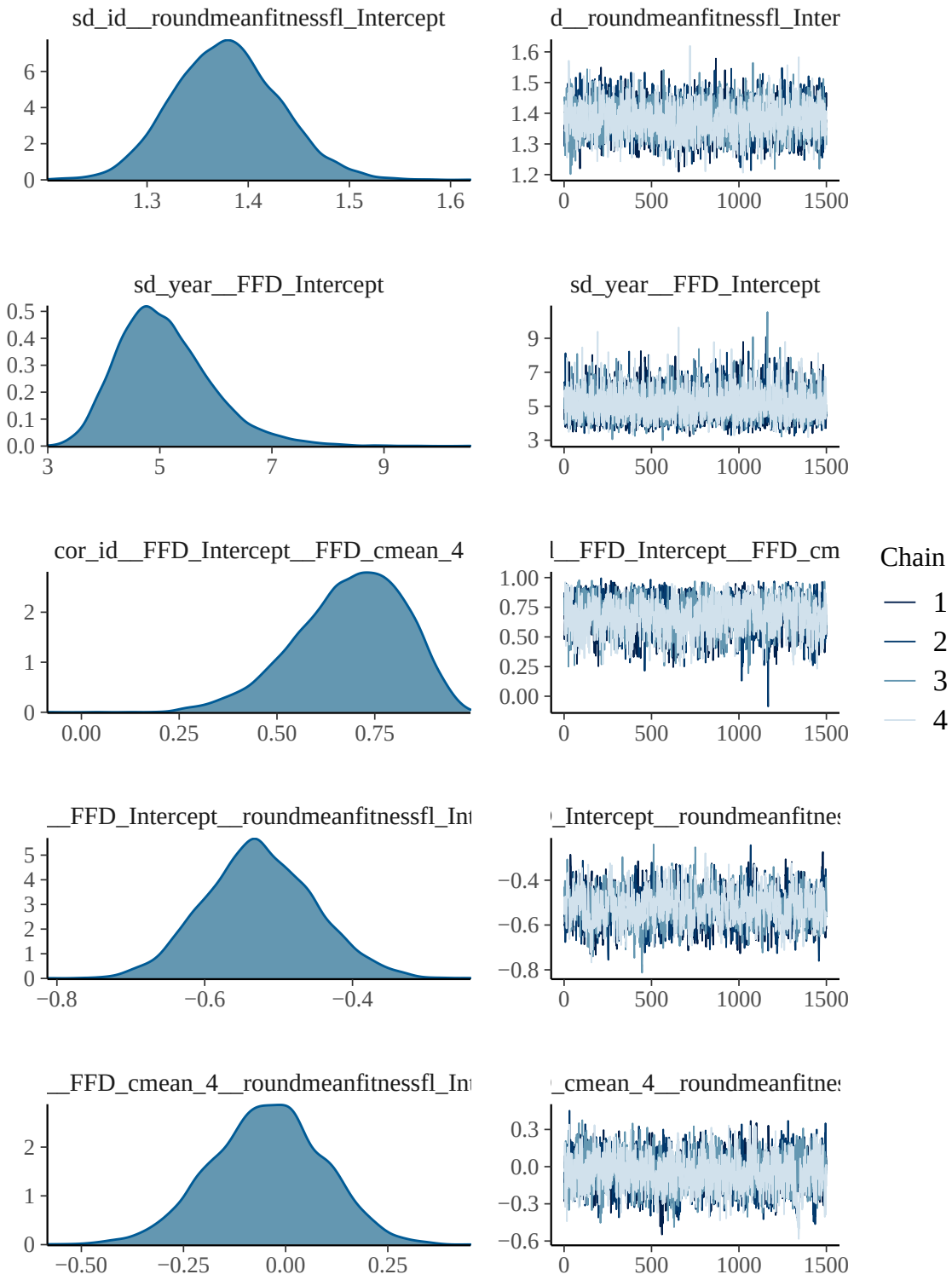

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessfl_Intercept
_roundmeanfitnessfl_Intercep
b_FFD_cmean_4
b_FFD_cmean_4
Chain
1
2
3
4
sd_id__FFD_Intercept
sd_id__FFD_Intercept
sd_id__FFD_cmean_4
sd_id__FFD_cmean_4

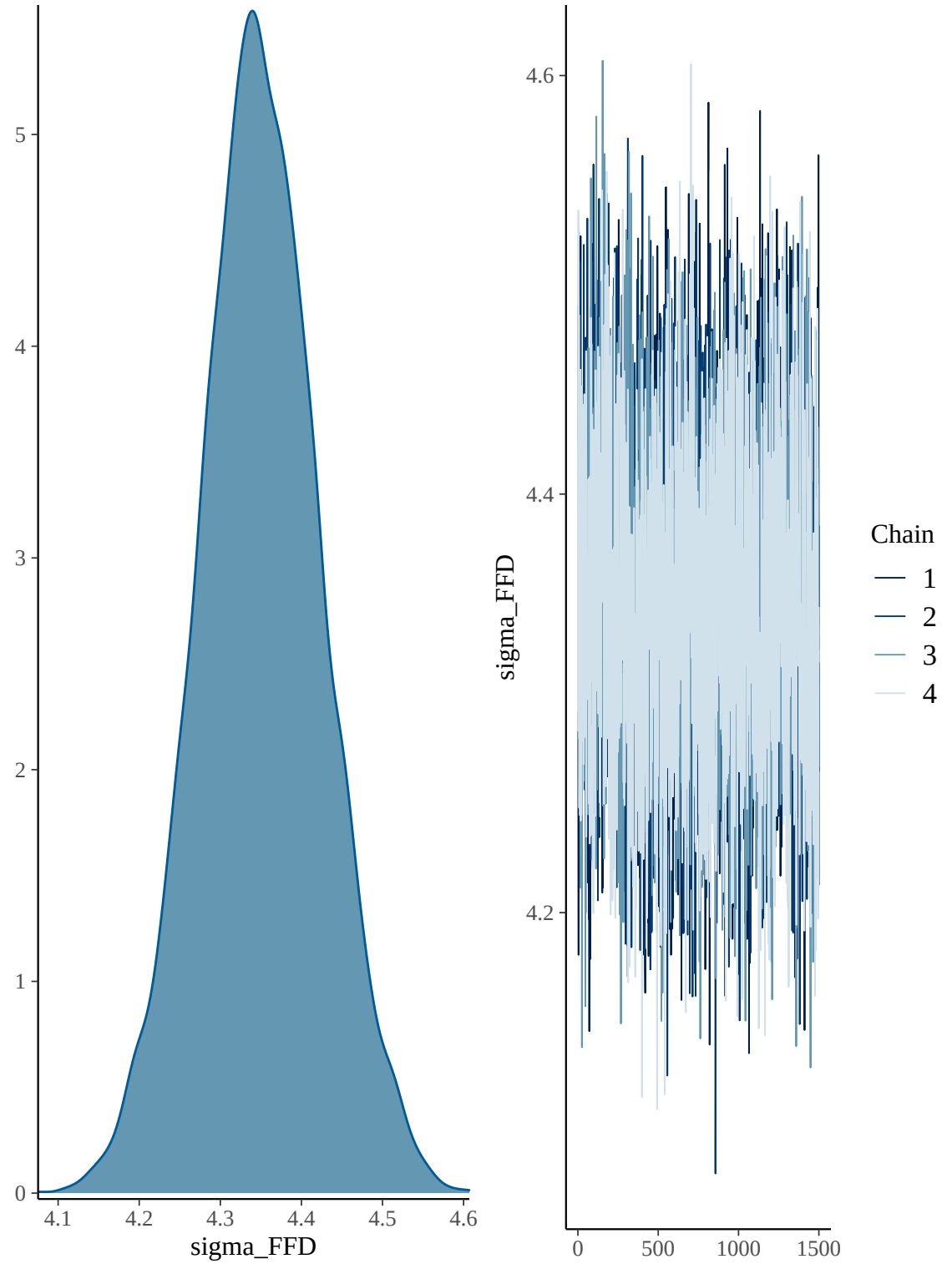

sd_id__roundmeanfitnessfl_Intercept
d__roundmeanfitnessfl_Inter
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
__FFD_Intercept__roundmeanfitnessfl_Int
_Intercept__roundmeanfitnes
__FFD_cmean_4__roundmeanfitnessfl_Int
_cmean_4__roundmeanfitnes
Chain
1
2
3
4




```
plot(bivar1.all.brm.nb)
```

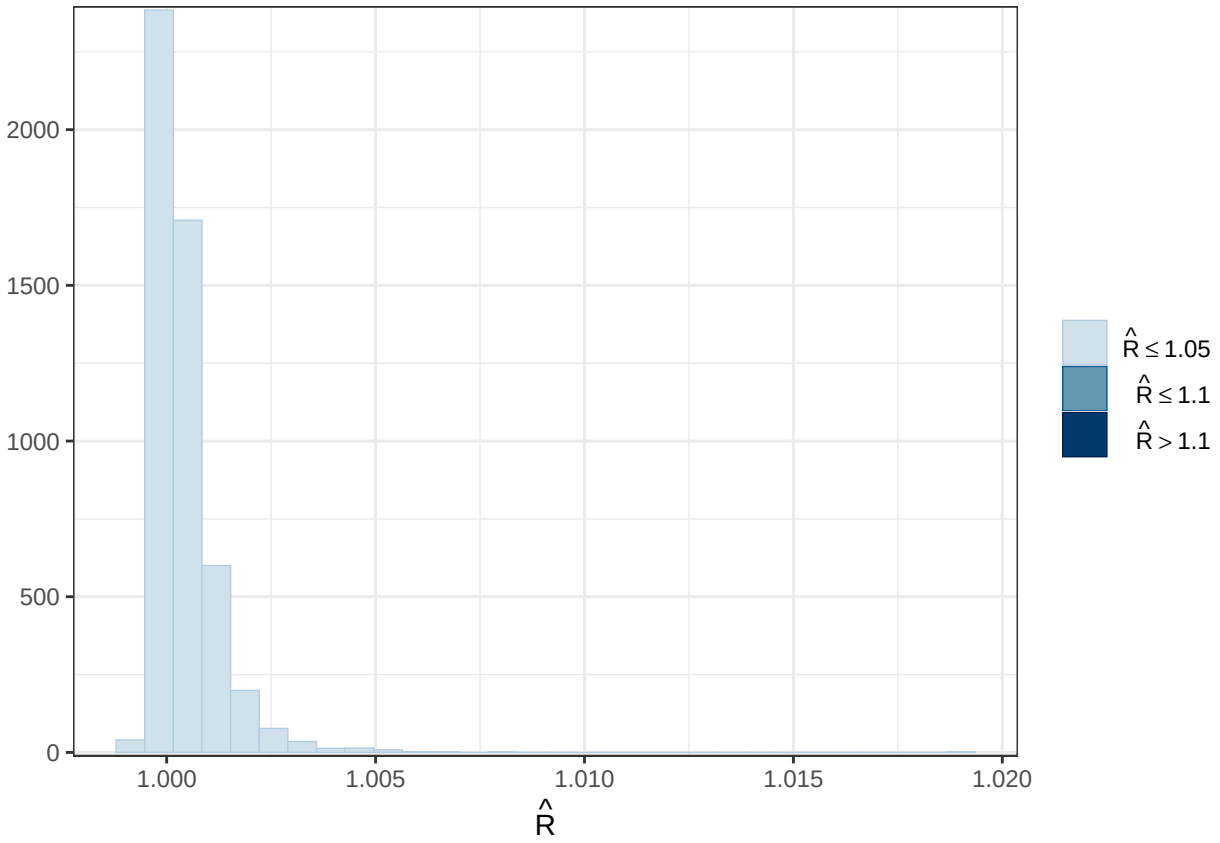

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessfl_Intercept
_roundmeanfitnessfl_Intercep
b_FFD_cmean_4
b_FFD_cmean_4
Chain
1
2
3
4
sd_id__FFD_Intercept
sd_id__FFD_Intercept
sd_id__FFD_cmean_4
sd_id__FFD_cmean_4


sd_id__roundmeanfitnessfl_Intercept
d__roundmeanfitnessfl_Inter
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
__FFD_Intercept__roundmeanfitnessfl_Int
_Intercept__roundmeanfitnes
__FFD_cmean_4__roundmeanfitnessfl_Int
_cmean_4__roundmeanfitnes
Chain
1
2
3
4

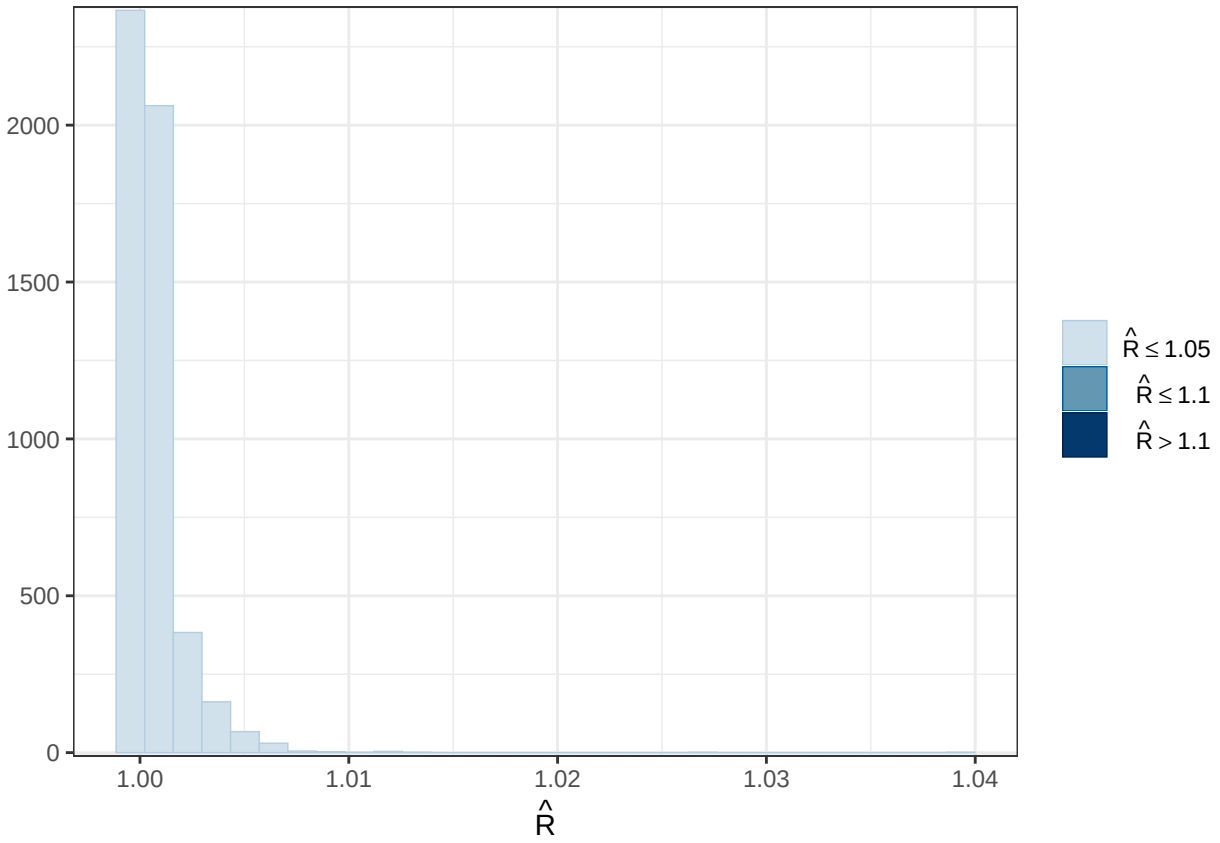

0
1
2
3
4
5
4.1
4.2
4.3
4.4
4.5
4.6
sigma_FFD
4.6
4.4
4.2
sigma_FFD
0
500
1000
1500
Chain
1
2
3
4

Rhat

```
mcmc_rhat_hist(rhat(bivar1.all.brm.pois))+theme_bw()
```

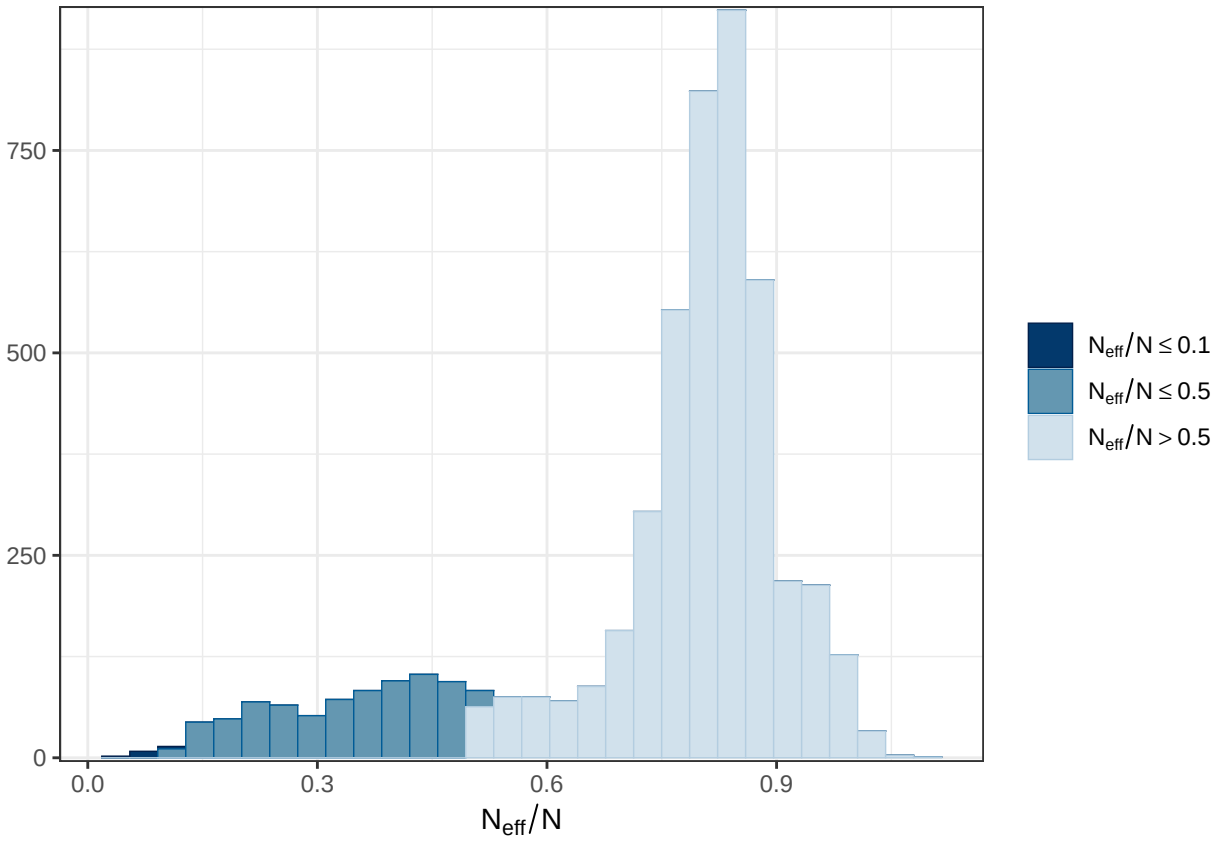



```
mcmc_rhat_hist(rhat(bivar1.all.brm.nb))+theme_bw()
```

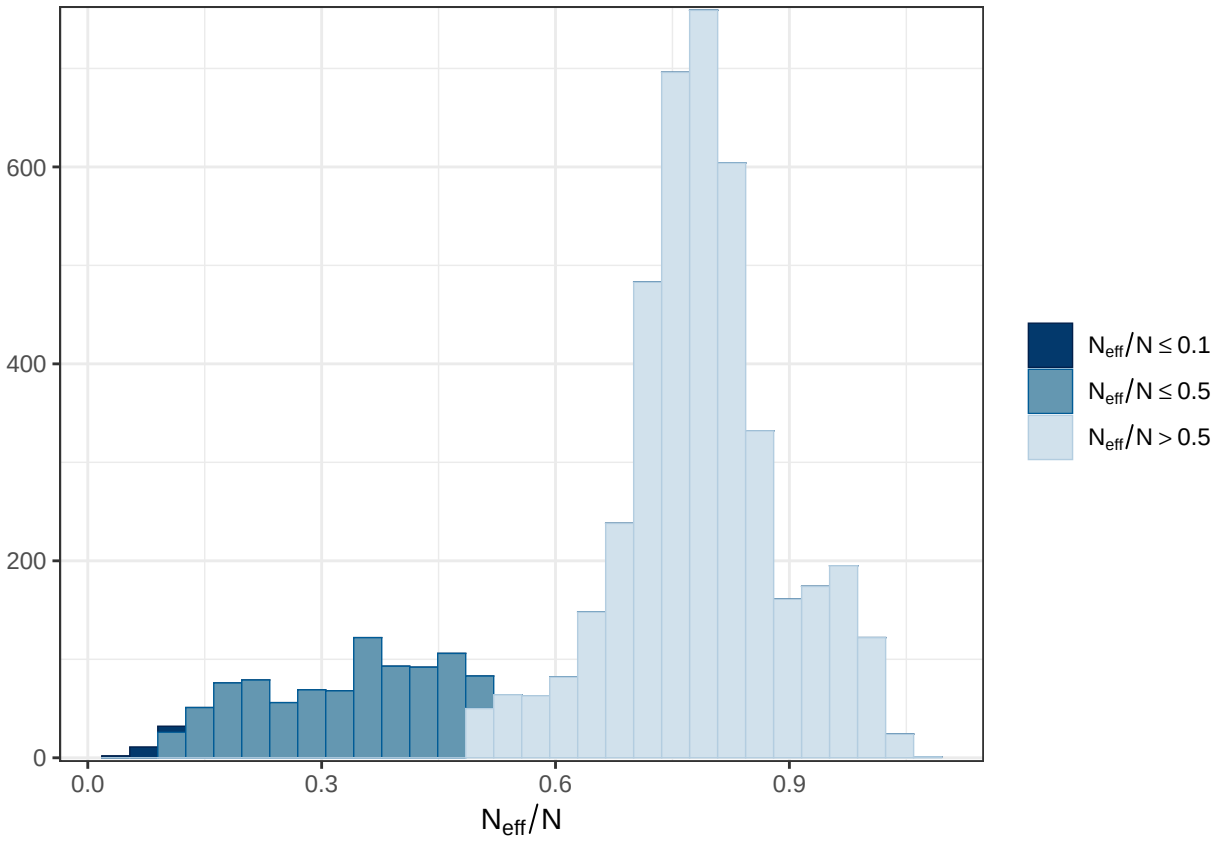


Effective sample size:

```
mcmc_neff_hist(neff_ratio(bivar1.all.brm.pois))+theme_bw()
```




```
mcmc_neff_hist(neff_ratio(bivar1.all.brm.nb))+theme_bw()
```



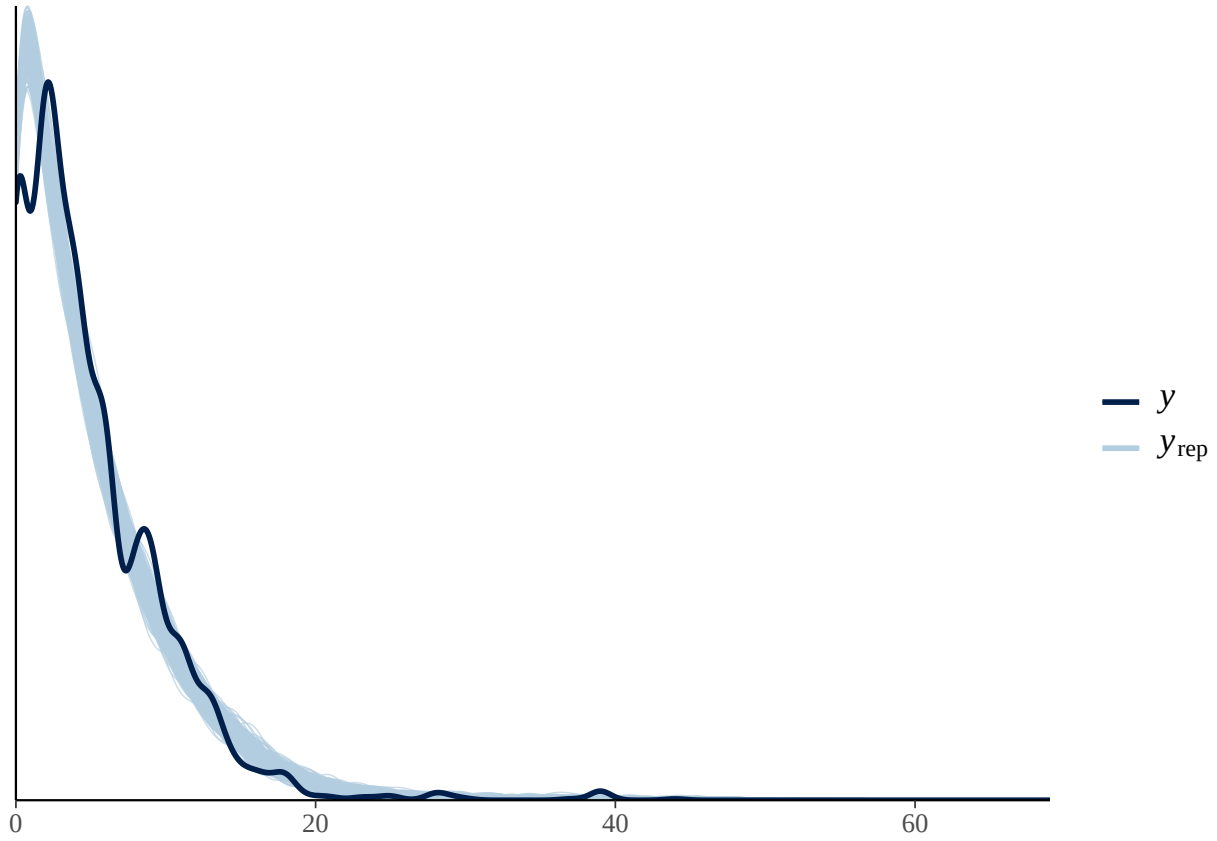
Posterior predictive checks:

```
y2_fitness<-round(subset(datadef,!is.na(FFD))$mean_fitness_fl)
y2_FFD<-subset(datadef,!is.na(FFD))$FFD # vectors of outcome values
yrep2_fitness_pois<-posterior_predict(bivar1.all.brm.pois,
                                draws = 500,resp="roundmeanfitnessfl")
yrep2_FFD_pois<-posterior_predict(bivar1.all.brm.pois,
                                draws = 500,resp="FFD")
yrep2_fitness_nb<-posterior_predict(bivar1.all.brm.nb,
                                draws = 500,resp="roundmeanfitnessfl")
yrep2_FFD_nb<-posterior_predict(bivar1.all.brm.nb,
                                draws = 500,resp="FFD")
# matrices of draws from the posterior predictive distribution
```

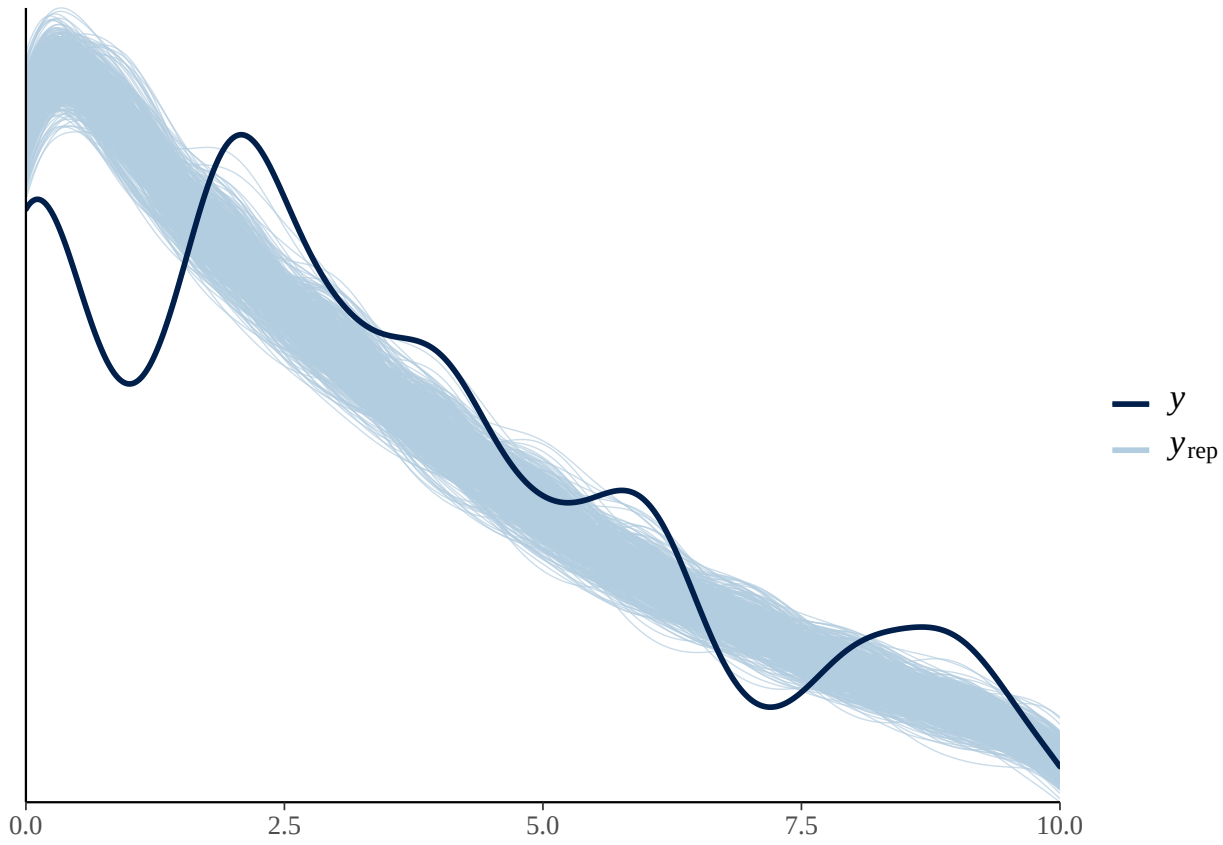
Each row of the matrix is a draw from the posterior predictive distribution, i.e. a vector with one element for each of the data points in y.

Comparison of the distribution of y and the distributions of the simulated datasets in the yrep matrix

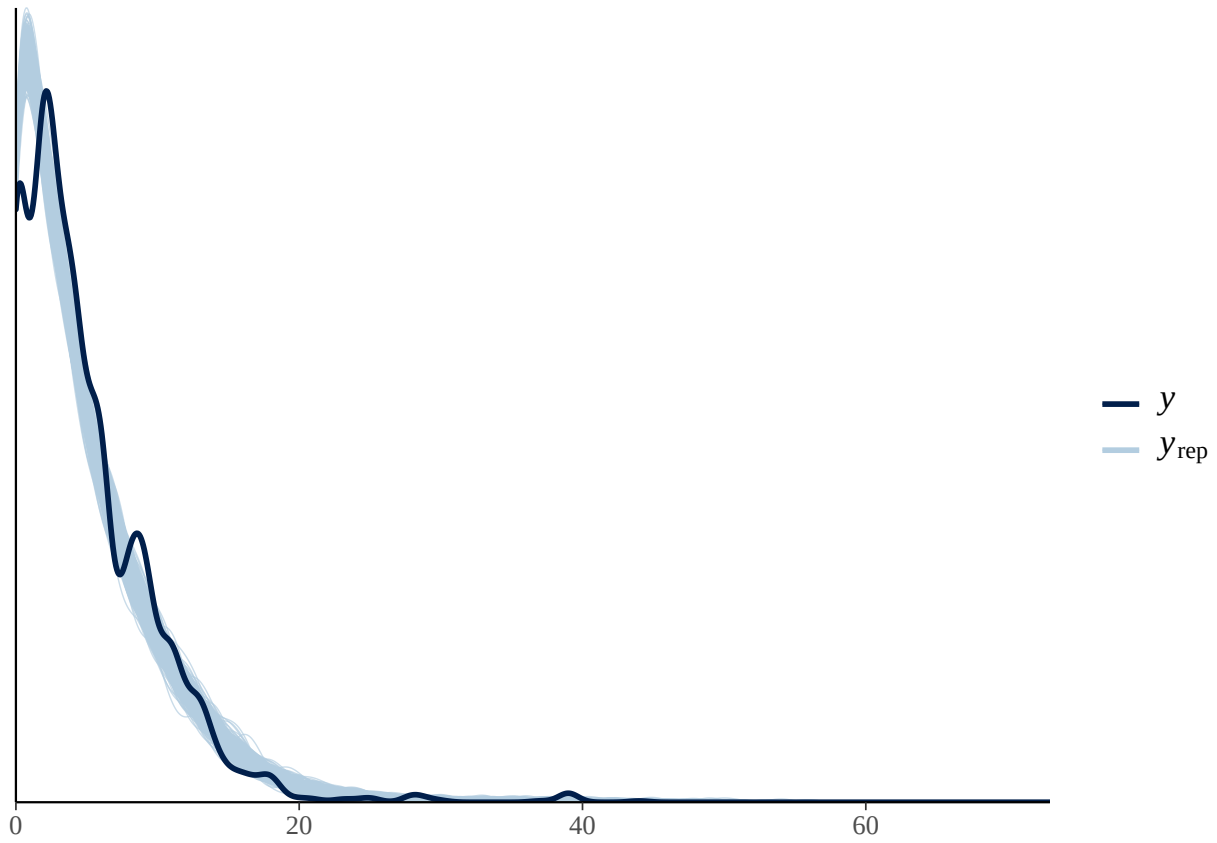
```
ppc_dens_overlay(y2_fitness, yrep2_fitness_pois[1:500,])
```



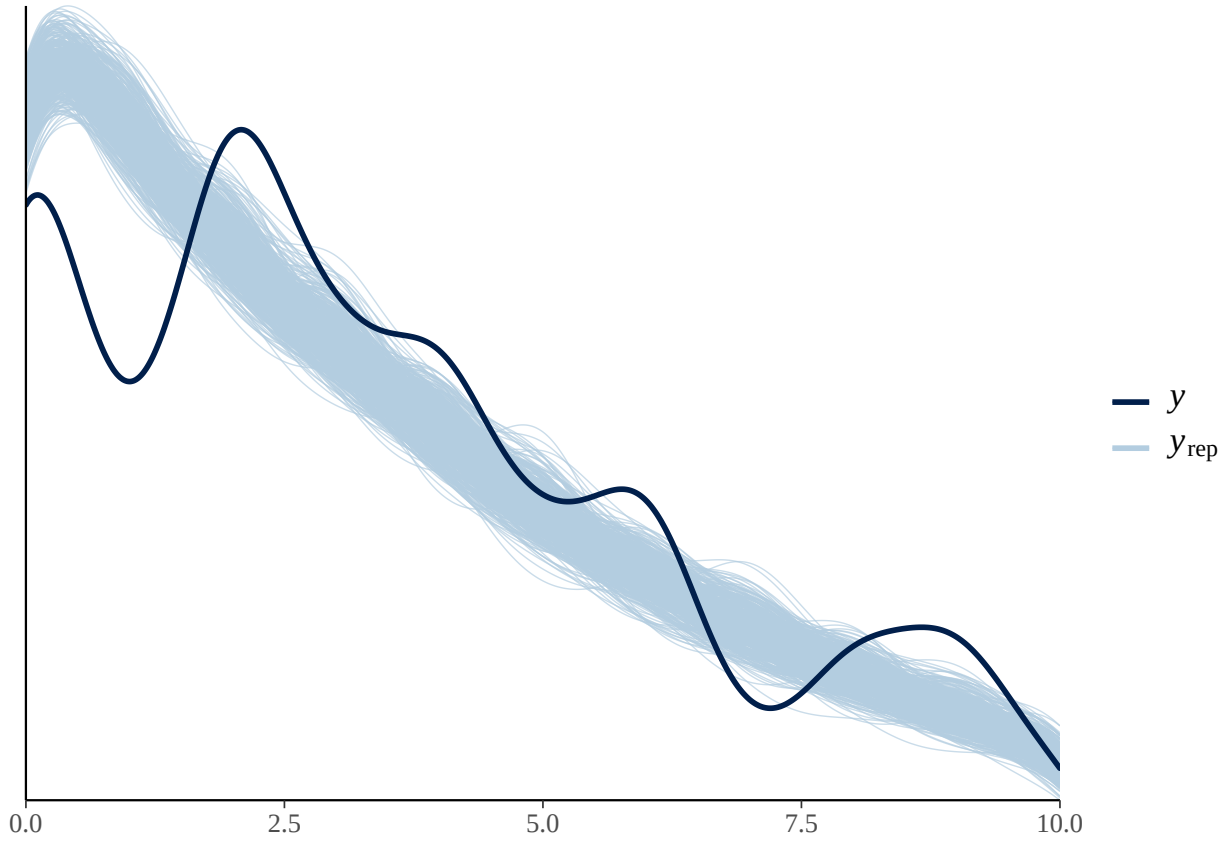

```
ppc_dens_overlay(y2_fitness, yrep2_fitness_pois[1:500,])+xlim(0, 10)
```



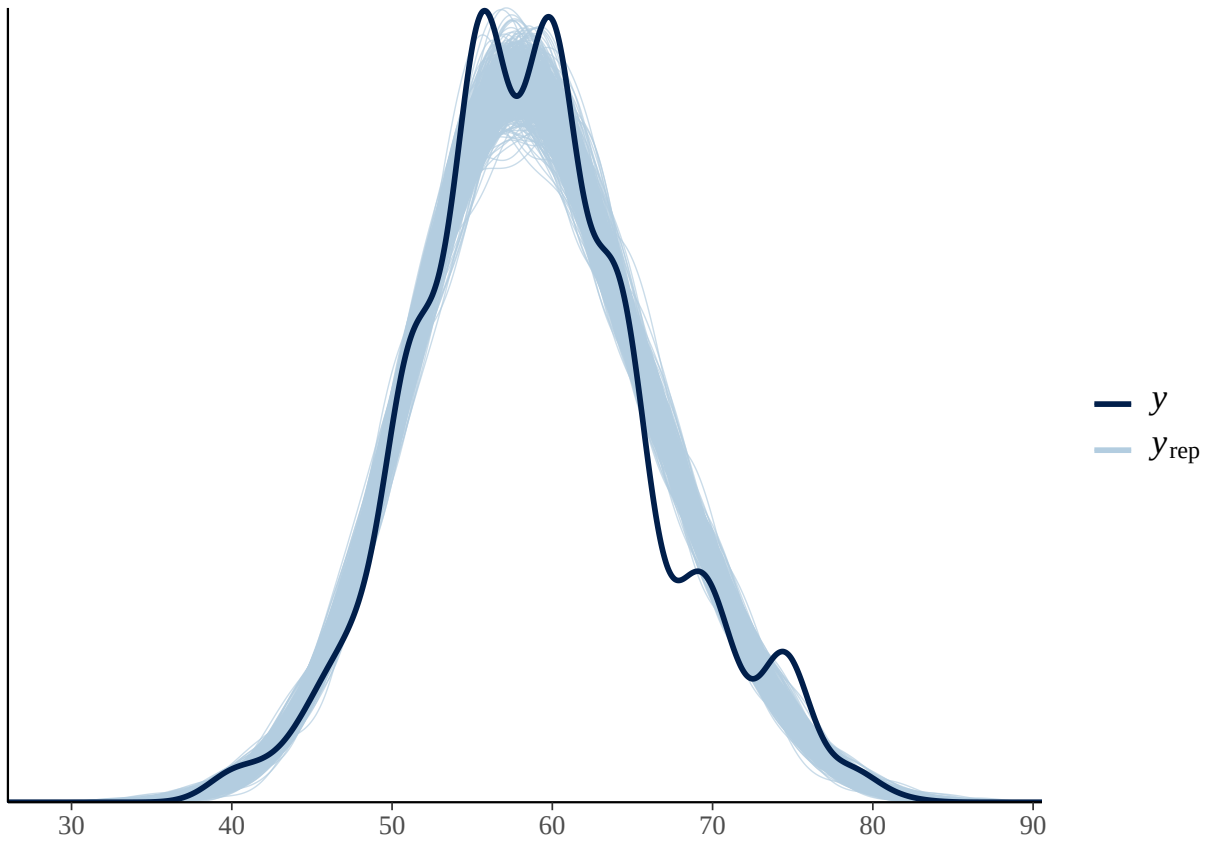

```
ppc_dens_overlay(y2_fitness, yrep2_fitness_nb[1:500,])
```



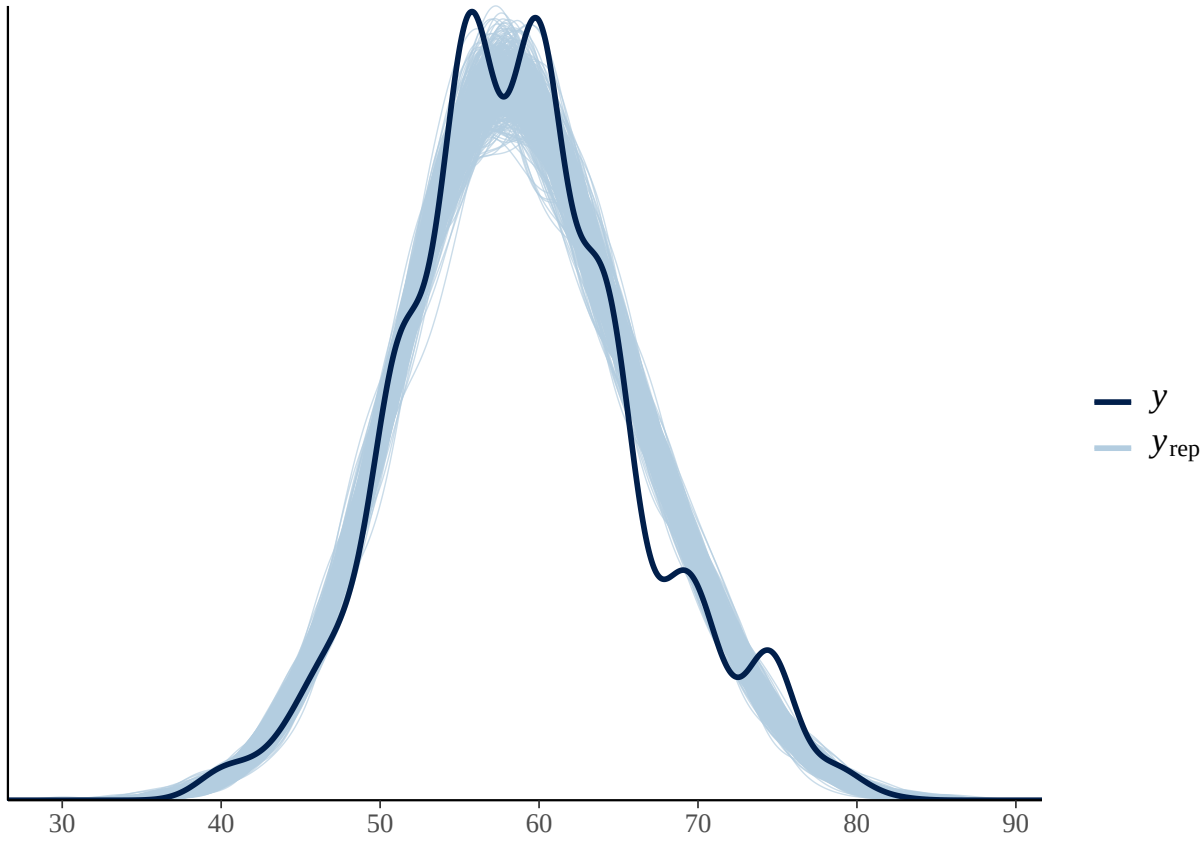

```
ppc_dens_overlay(y2_fitness, yrep2_fitness_nb[1:500,])+xlim(0,10)
```




```
ppc_dens_overlay(y2_FFD, yrep2_FFD_pois[1:500,])
```



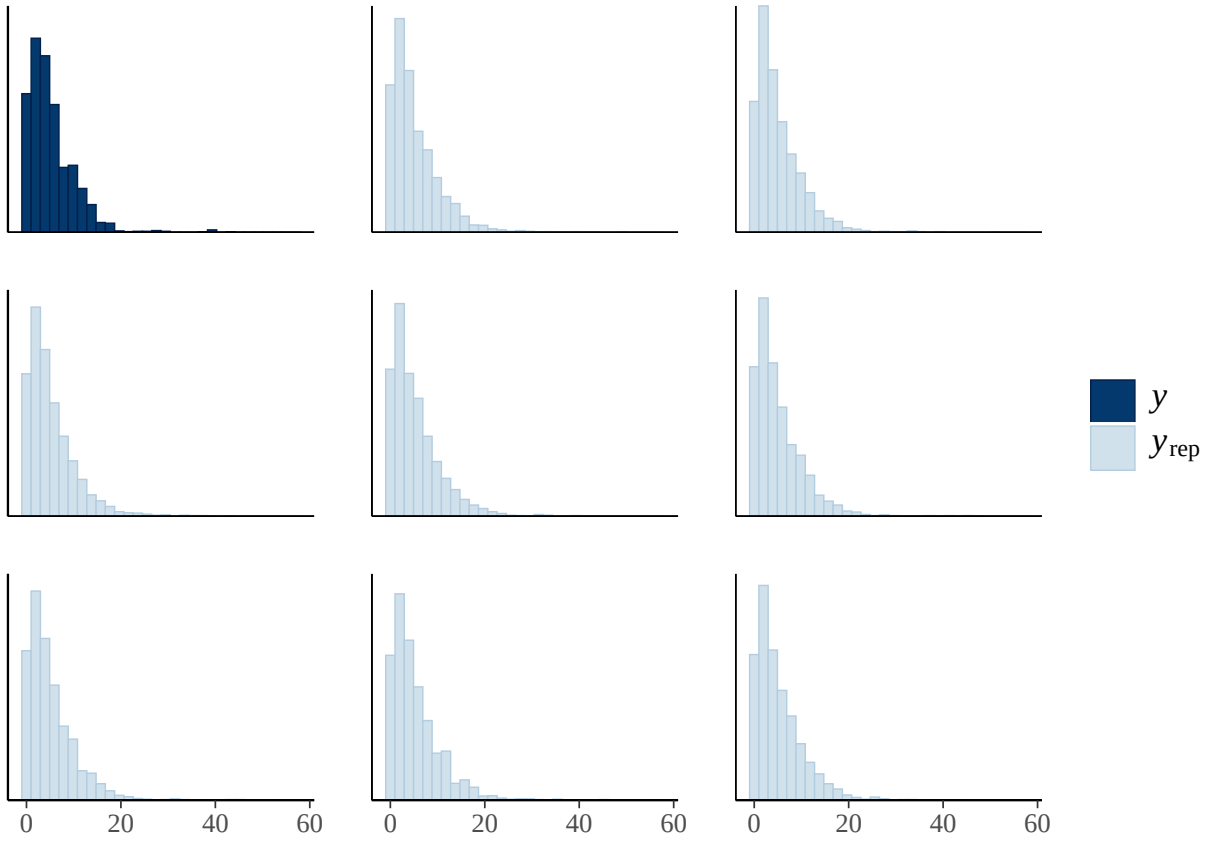

```
ppc_dens_overlay(y2_FFD, yrep2_FFD_nb[1:500,])
```



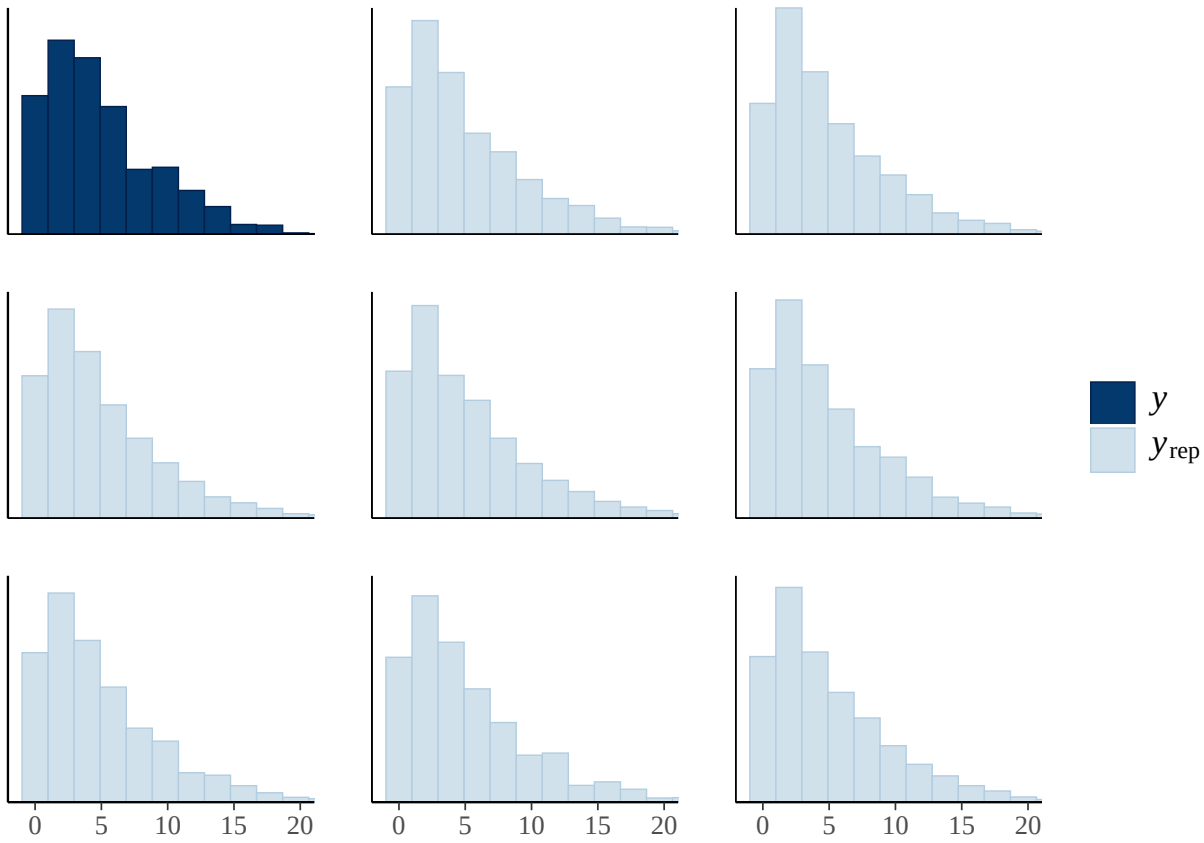
Poisson and negative binomial look pretty similar.

Separate histograms of y and some of the yrep datasets

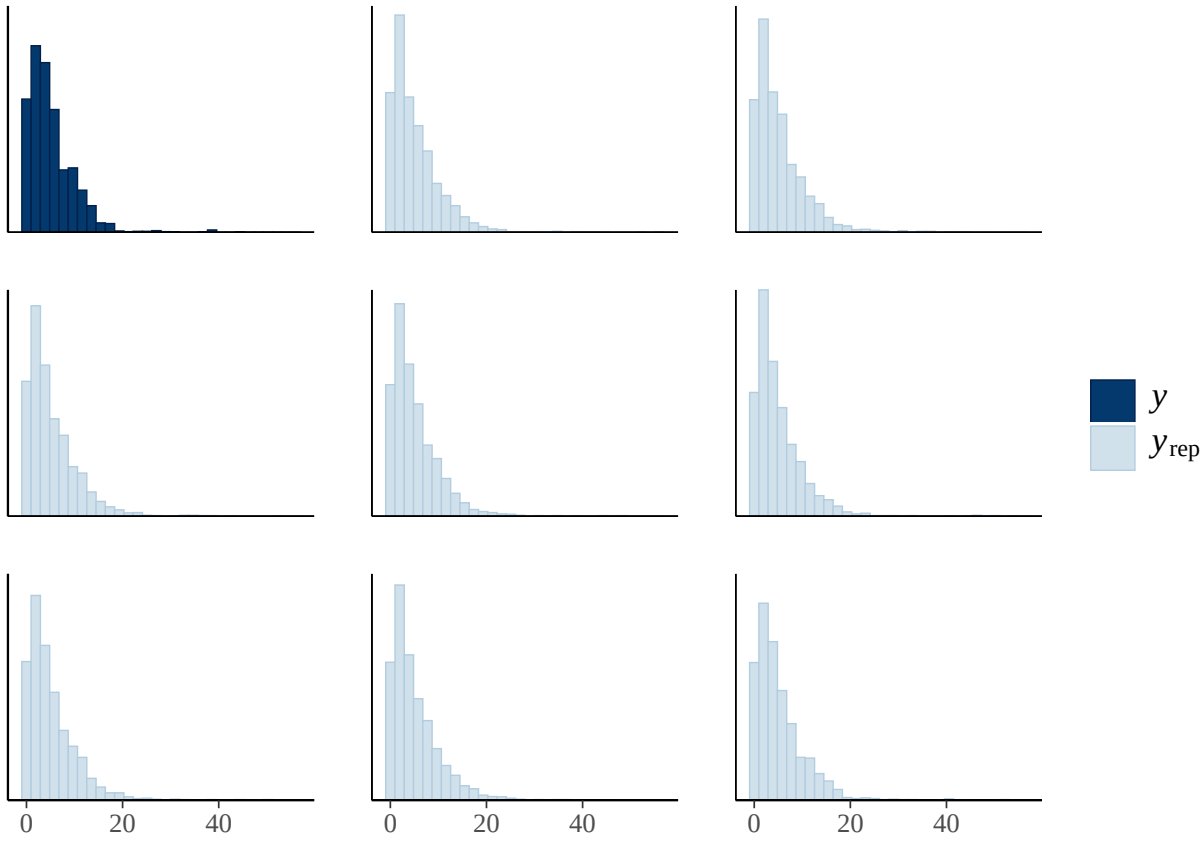
```
ppc_hist(y2_fitness, yrep2_fitness_pois[1:8, ])
```



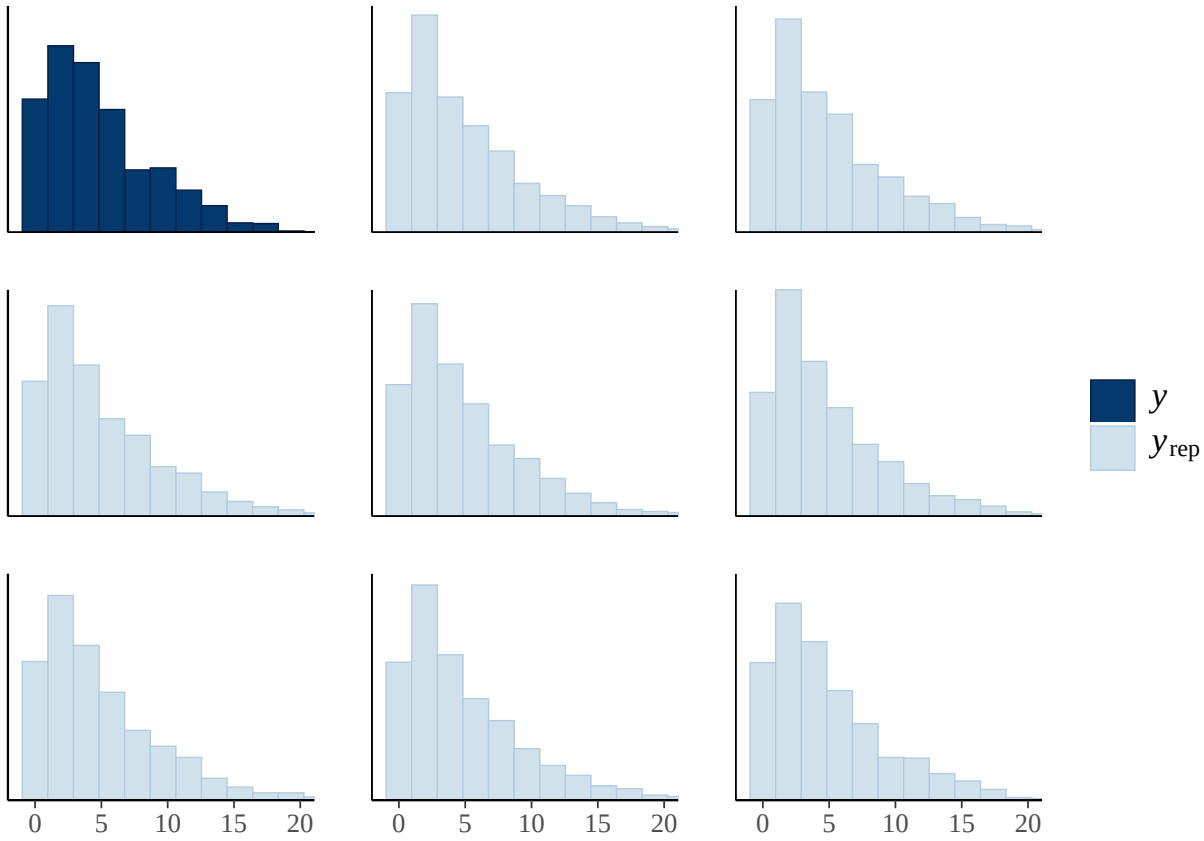

```
ppc_hist(y2_fitness, yrep2_fitness_pois[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```




```
ppc_hist(y2_fitness, yrep2_fitness_nb[1:8, ])
```

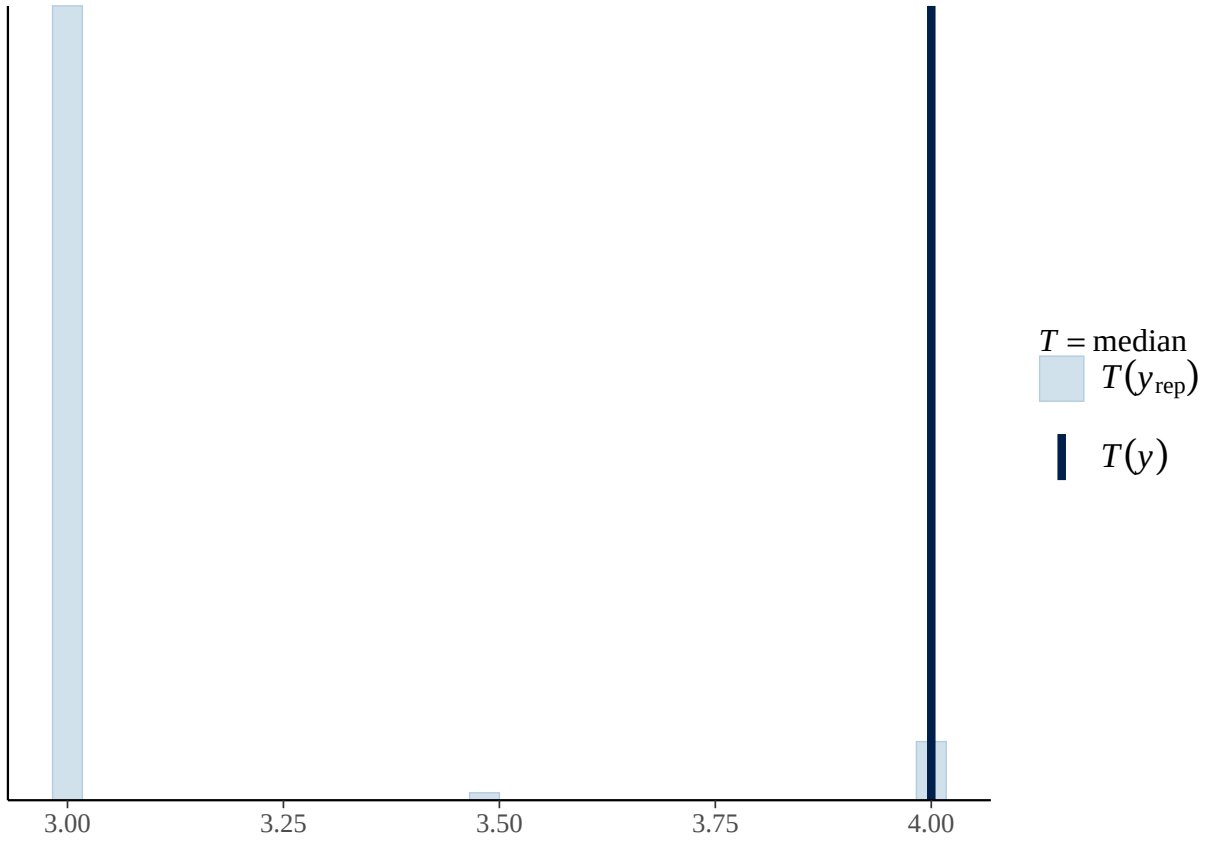



```
ppc_hist(y2_fitness, yrep2_fitness_nb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```

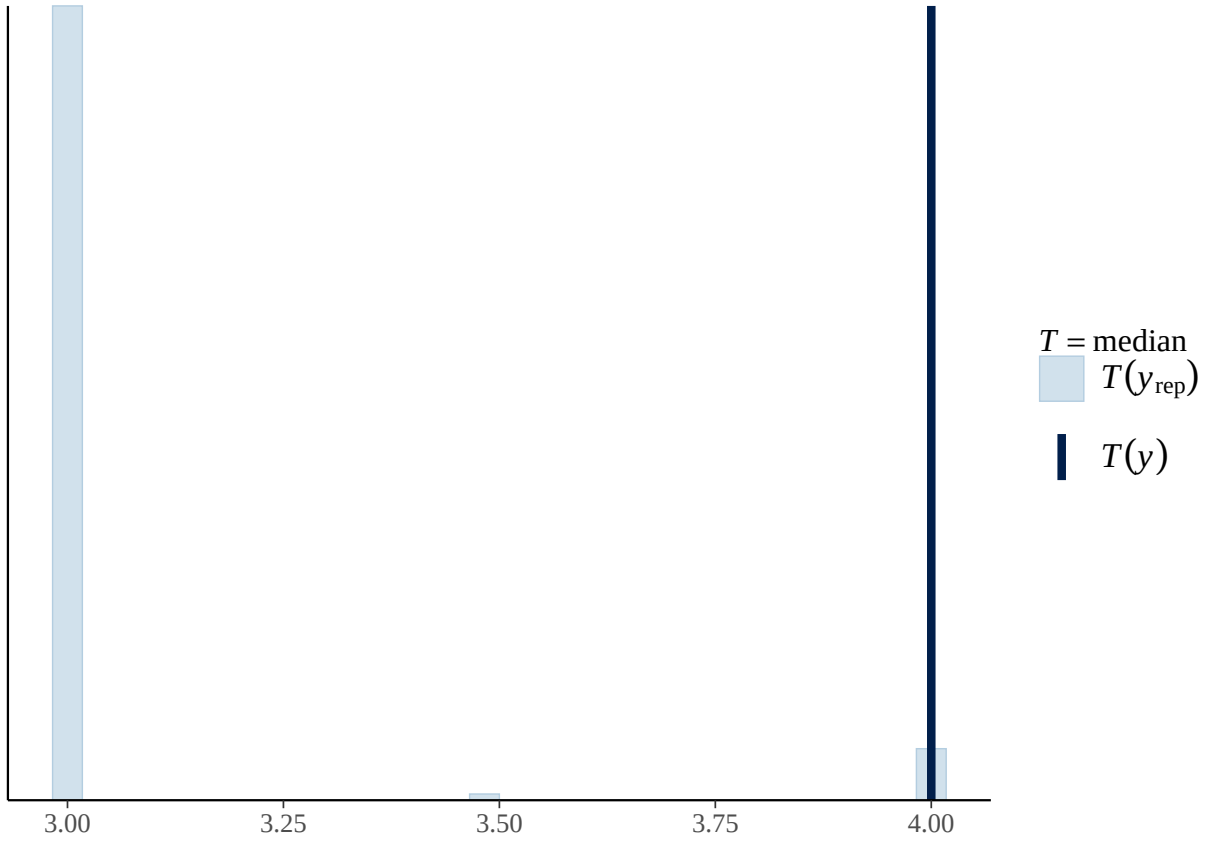


Here, negative binomial looks a bit better for fitness.

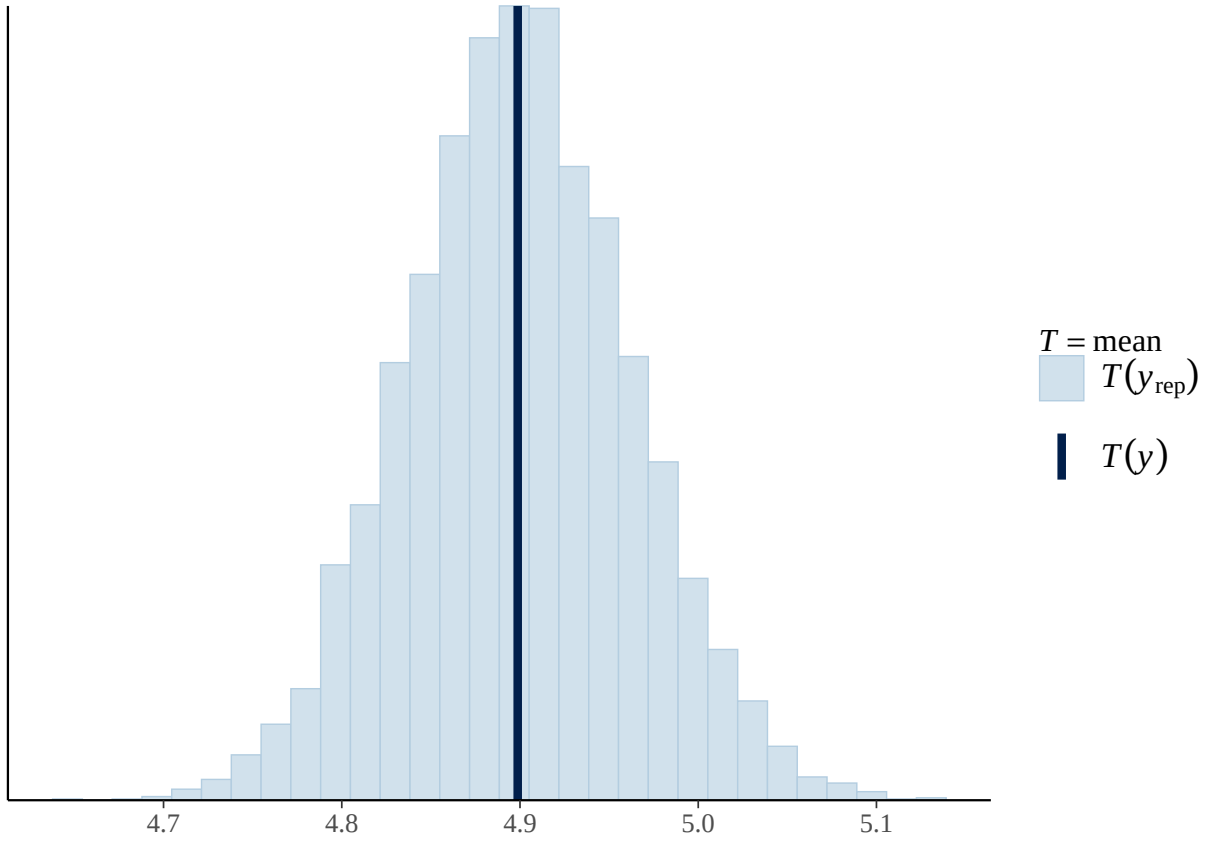
```
ppc_stat(y2_fitness, yrep2_fitness_pois,stat="median")
```



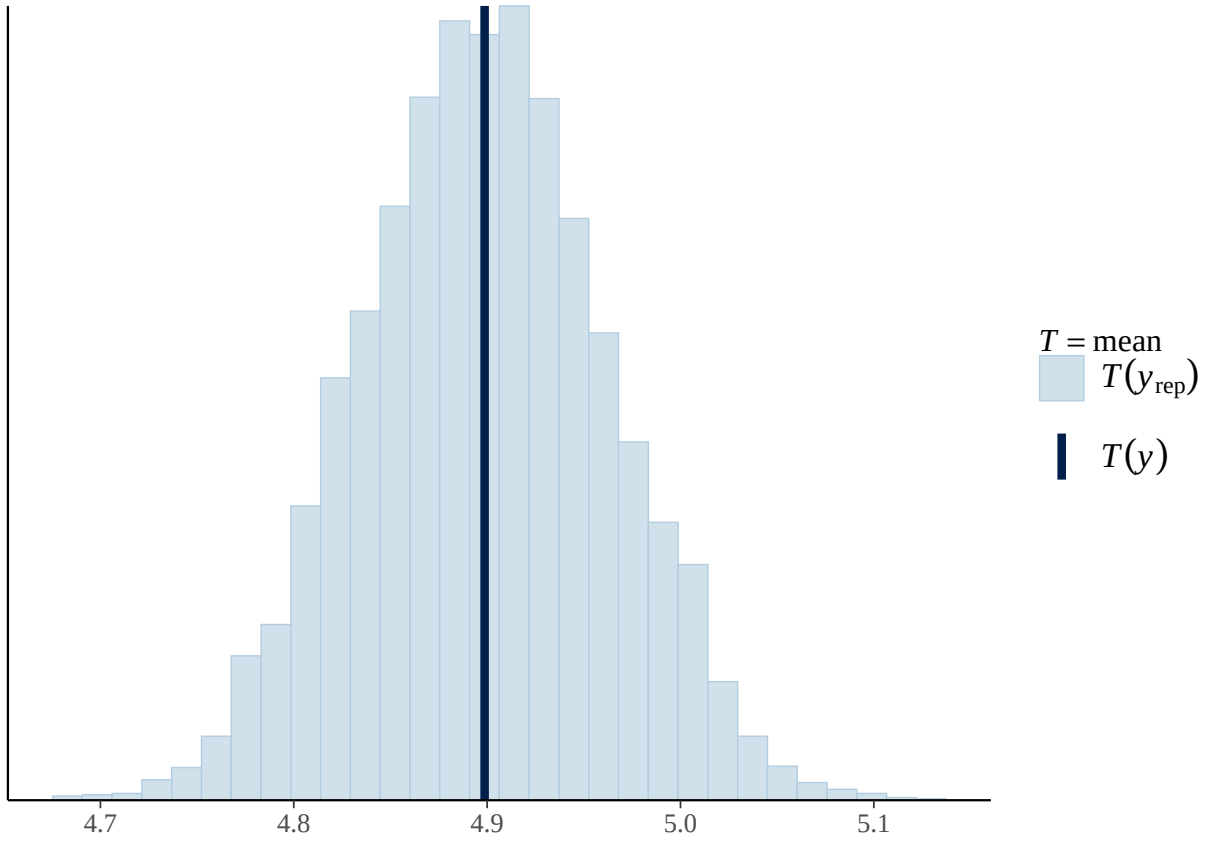

```
ppc_stat(y2_fitness, yrep2_fitness_nb,stat="median")
```



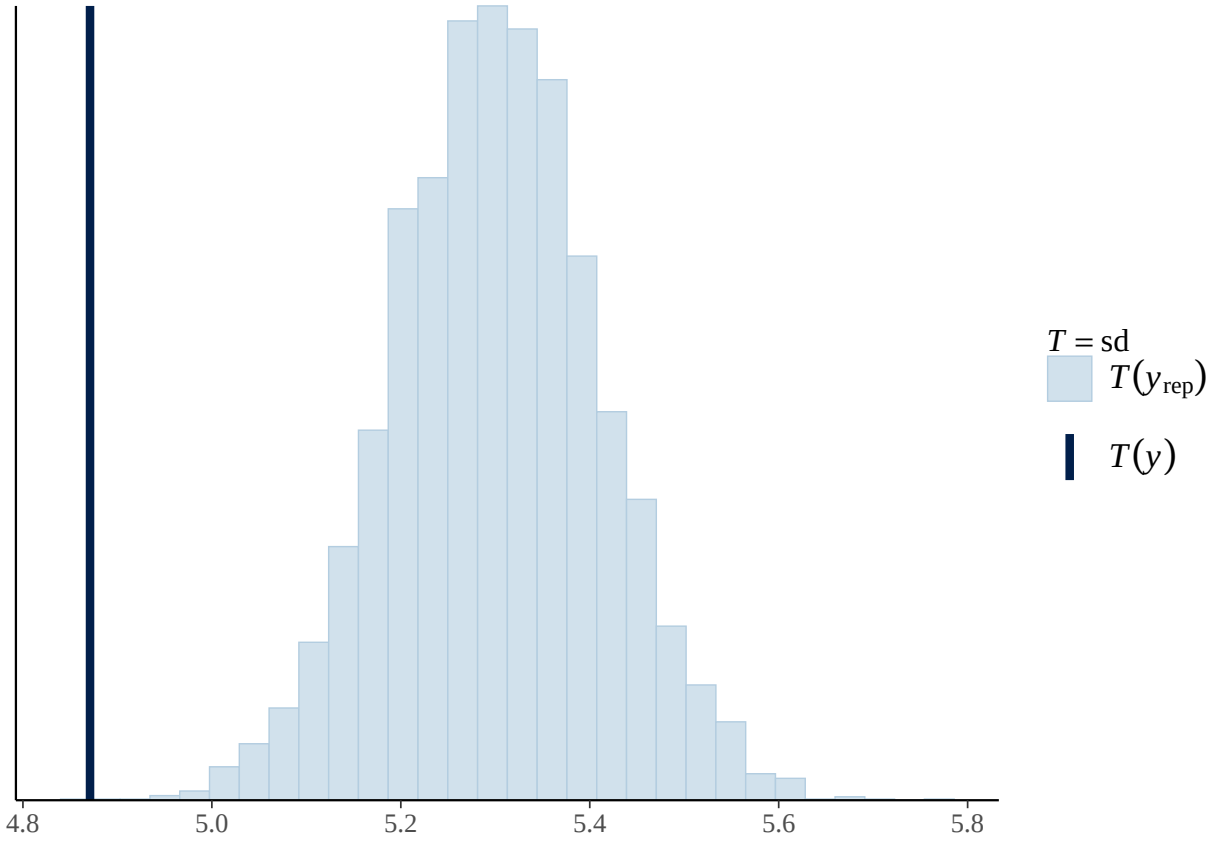

```
ppc_stat(y2_fitness, yrep2_fitness_pois,stat="mean")
```



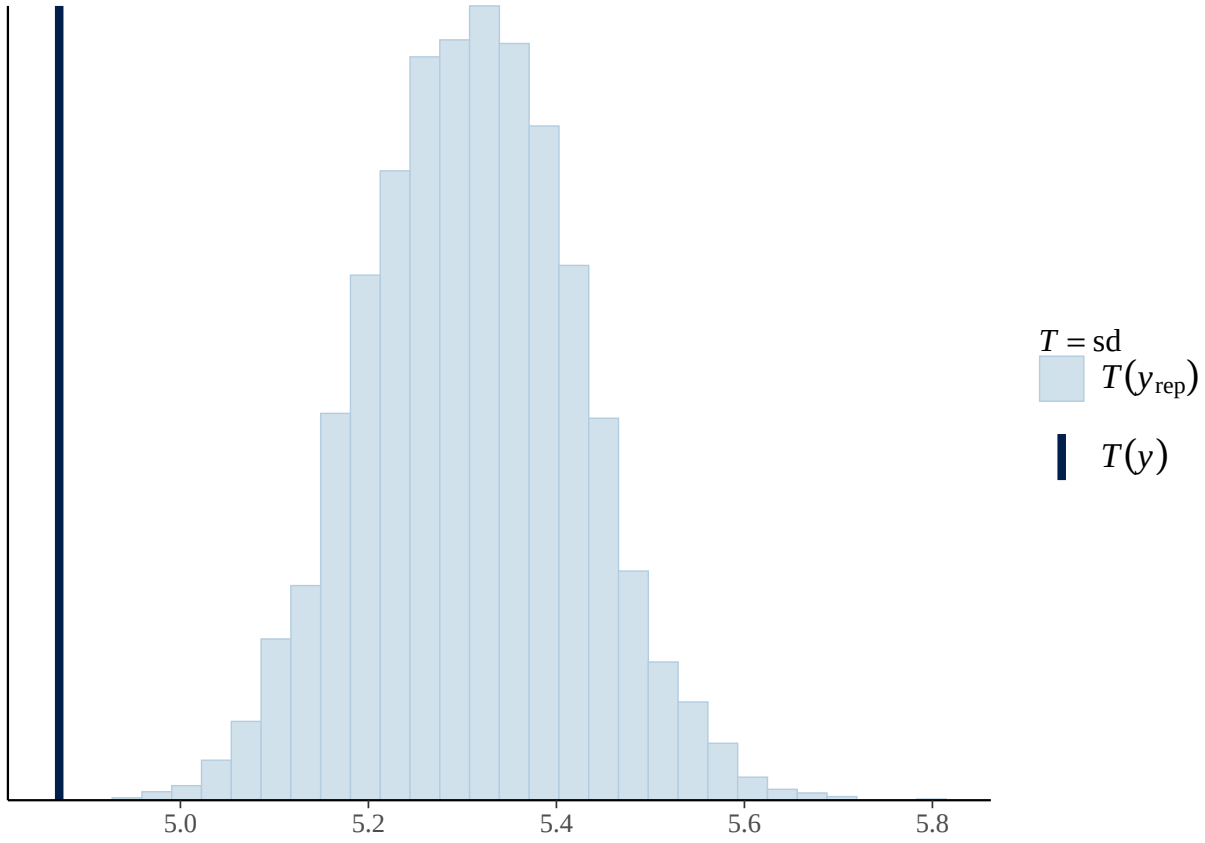

```
ppc_stat(y2_fitness, yrep2_fitness_nb,stat="mean")
```



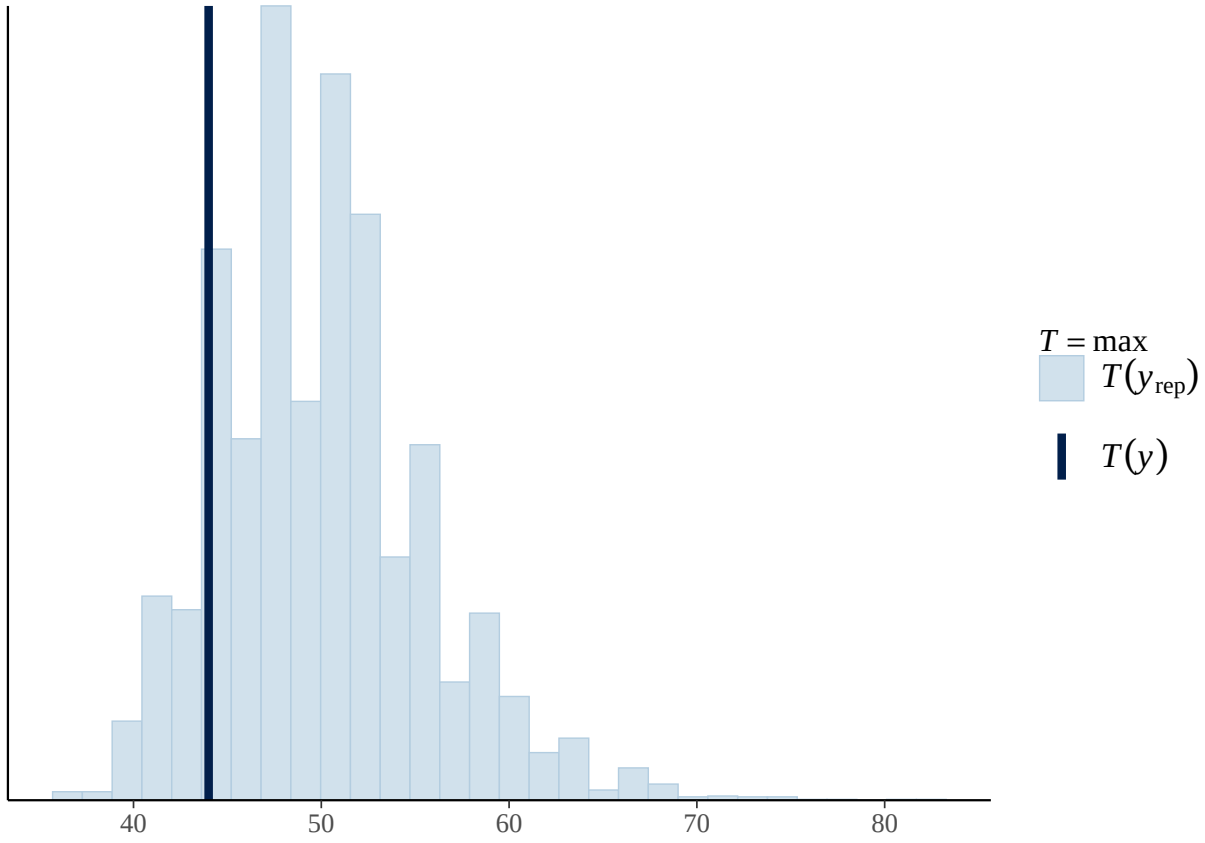

```
ppc_stat(y2_fitness, yrep2_fitness_pois,stat="sd")
```



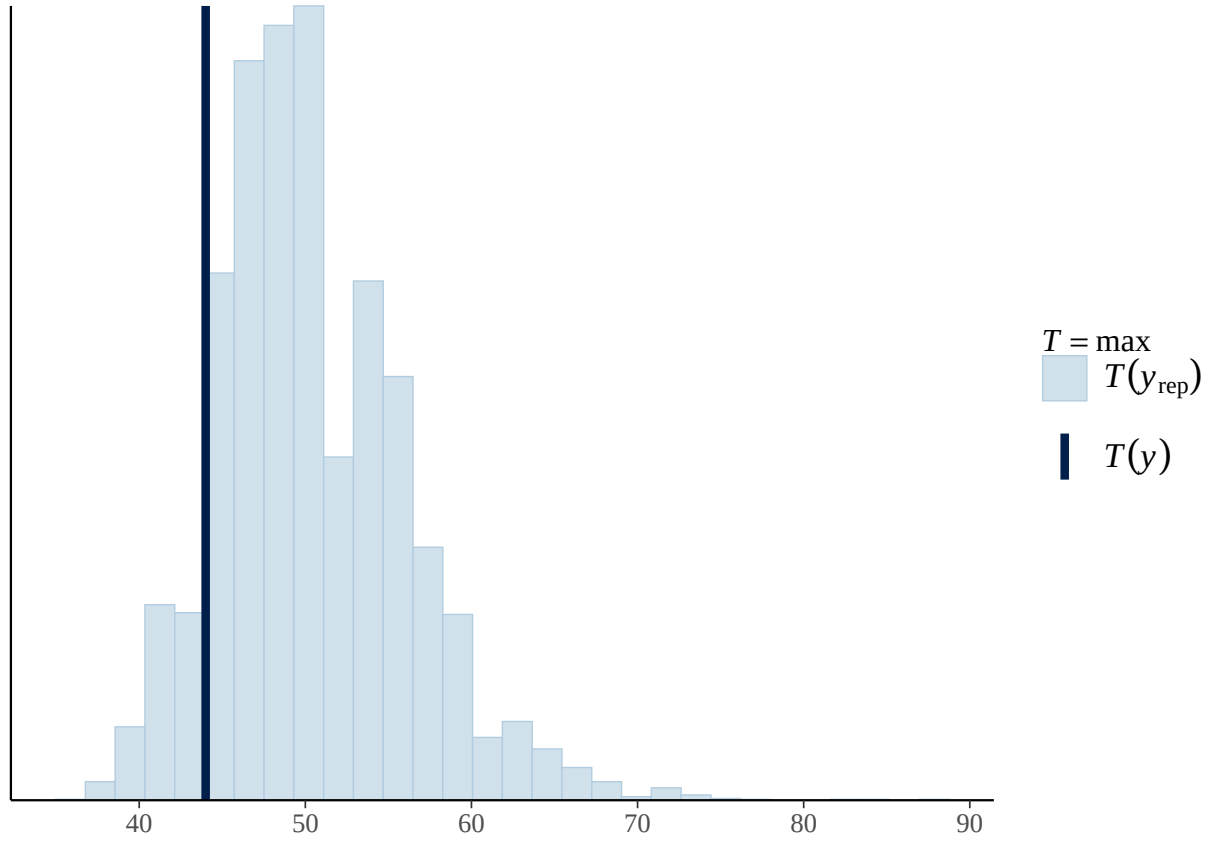

```
ppc_stat(y2_fitness, yrep2_fitness_nb,stat="sd")
```




```
ppc_stat(y2_fitness, yrep2_fitness_pois,stat="max")
```




```
ppc_stat(y2_fitness, yrep2_fitness_nb,stat="max")
```

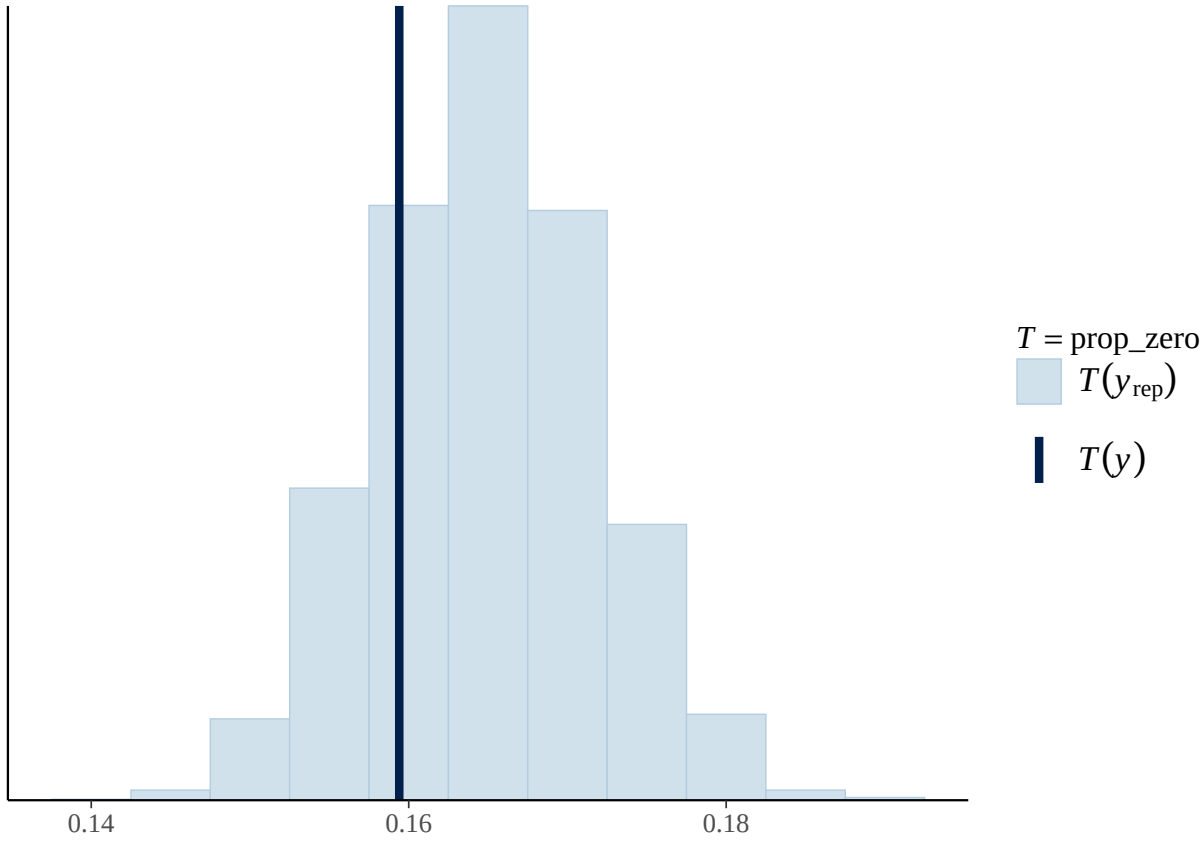


Look at the distribution of the proportion of zeros over the replicated datasets from the posterior predictive distribution in yrep and compare to the proportion of observed zeros in y.

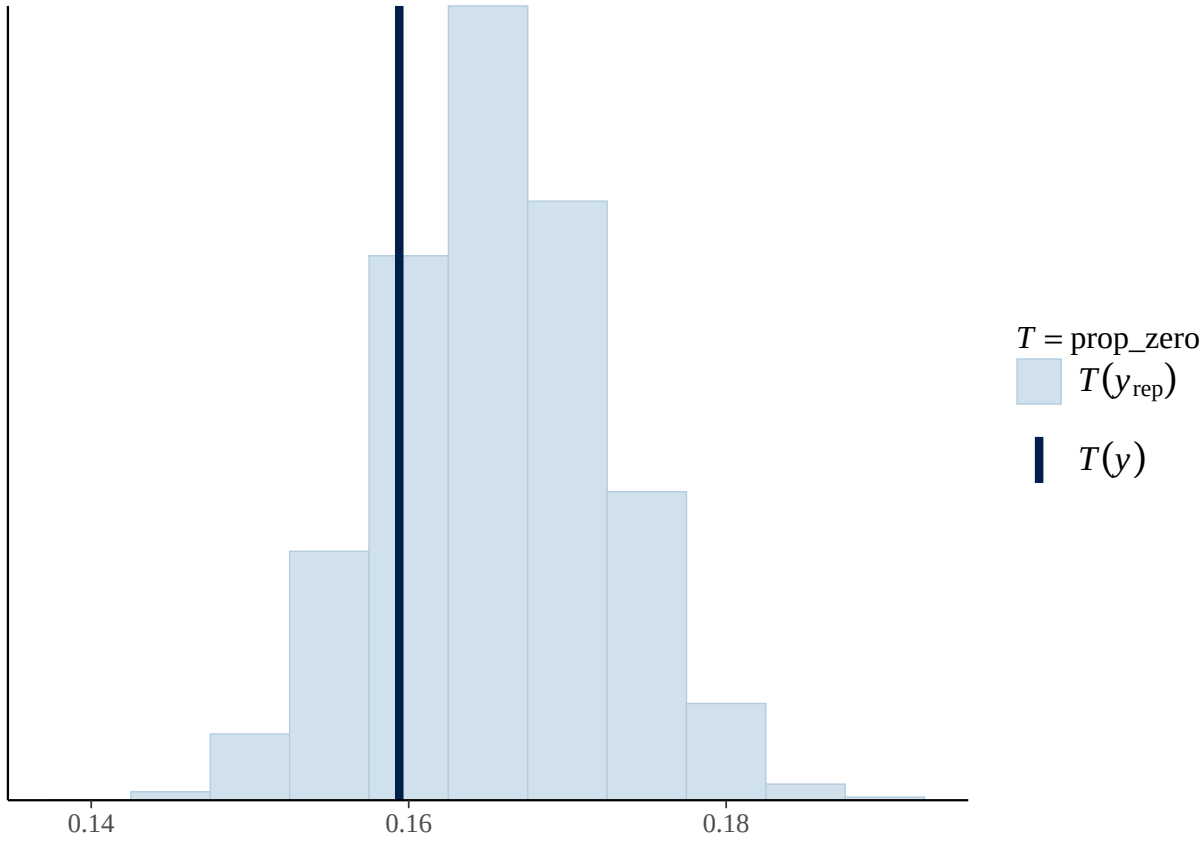
```
# Define a function that takes a vector as input
# and returns the proportion of zeros:
prop_zero <- function(x) mean(x == 0)
prop_zero(y2_fitness) # check proportion of zeros in y
```

```
## [1] 0.1594027
```

```
ppc_stat(y2_fitness, yrep2_fitness_pois, stat = "prop_zero", binwidth = 0.005)
```




```
ppc_stat(y2_fitness, yrep2_fitness_nb, stat = "prop_zero", binwidth = 0.005)
```



They look quite similar.

Measure of fit: Bayesian R2

```
bayes_R2(bivar1.all.brm.pois)
```

```
##                          Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                   0.6478432 0.009148772 0.6295761 0.6658871
## R2roundmeanfitnessfl 0.9394586 0.004492216 0.9300309 0.9474539
```

```
bayes_R2(bivar1.all.brm.nb)
```

```
##                          Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                   0.6483706 0.009431454 0.6297503 0.6665568
## R2roundmeanfitnessfl 0.9386089 0.004612906 0.9290325 0.9471120
```

Very similar.

Widely Applicable Information Criterion (WAIC):

```
waic1<-waic(bivar1.all.brm.pois,bivar1.all.brm.nb,compare=T)
```

```
waic1
```

```
## Output of model 'bivar1.all.brm.pois':
##
## Computed from 6000 by 2478 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic -11458.3  50.7
## p_waic       482.3  12.0
## waic       22916.5 101.4
##
## 456 (18.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.
##
## Output of model 'bivar1.all.brm.nb':
##
## Computed from 6000 by 2478 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic -11466.5  50.8
## p_waic       485.3  12.1
## waic       22932.9 101.6
##
## 451 (18.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.
##
## Model comparisons:
##                      elpd_diff se_diff
## bivar1.all.brm.pois   0.0       0.0
## bivar1.all.brm.nb    -8.2       1.0
```

Suggests that poisson is a bit better, but not sure we can trust this.

Leave-one-out cross validation (LOO):

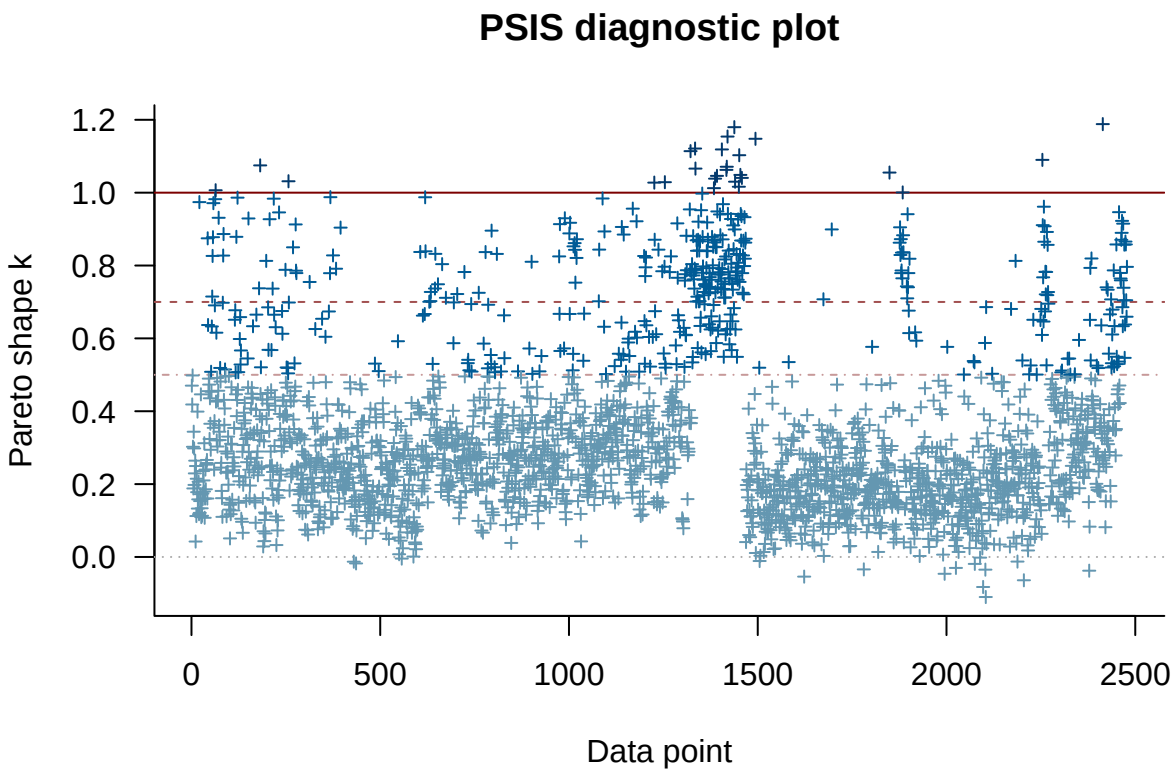
Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details. Found 263 observations with a pareto_k > 0.7 in model 'bivar1.all.brm.pois'. With this many problematic observations, it may be more appropriate to use 'kfold' with argument 'K = 10' to perform 10-fold cross-validation rather than LOO.

```
loo_compare(loo2_pois,loo2_nb)
```

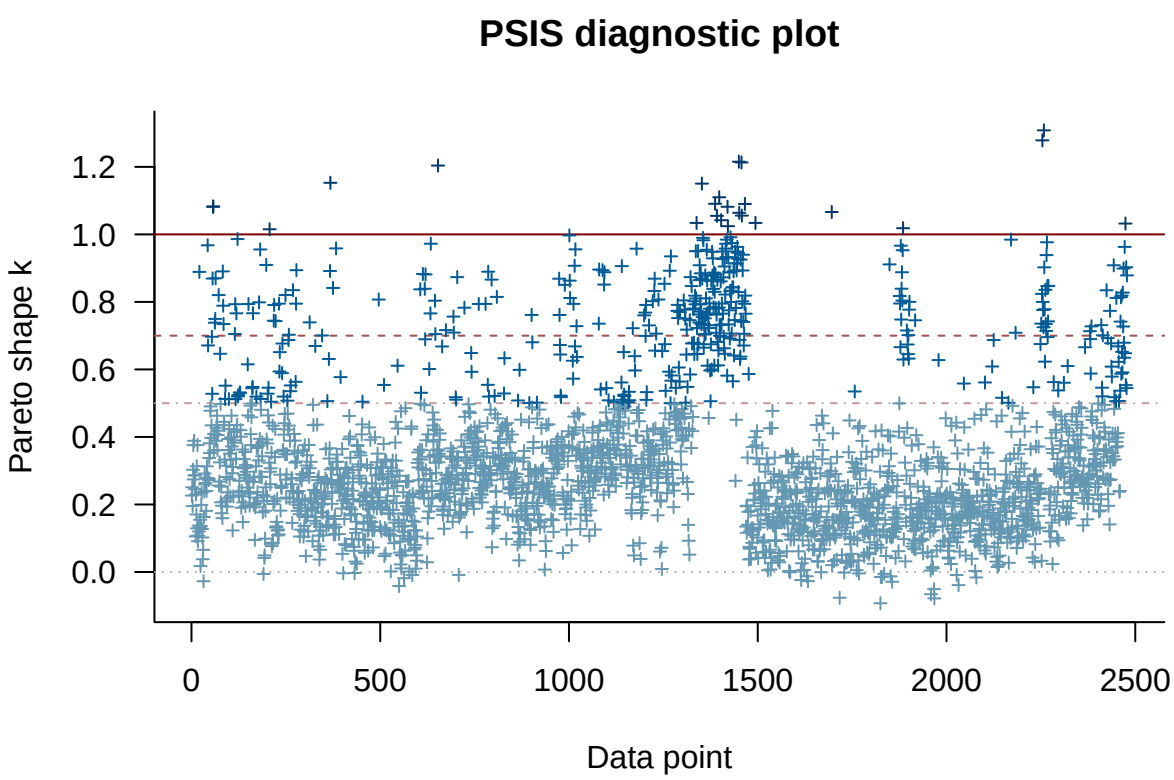
```
##                      elpd_diff se_diff
## bivar1.all.brm.pois    0.0       0.0
## bivar1.all.brm.nb    -11.7       2.9
```

Suggests that poisson is a bit better, but not sure we can trust this.

```
plot(loo2_pois)
```

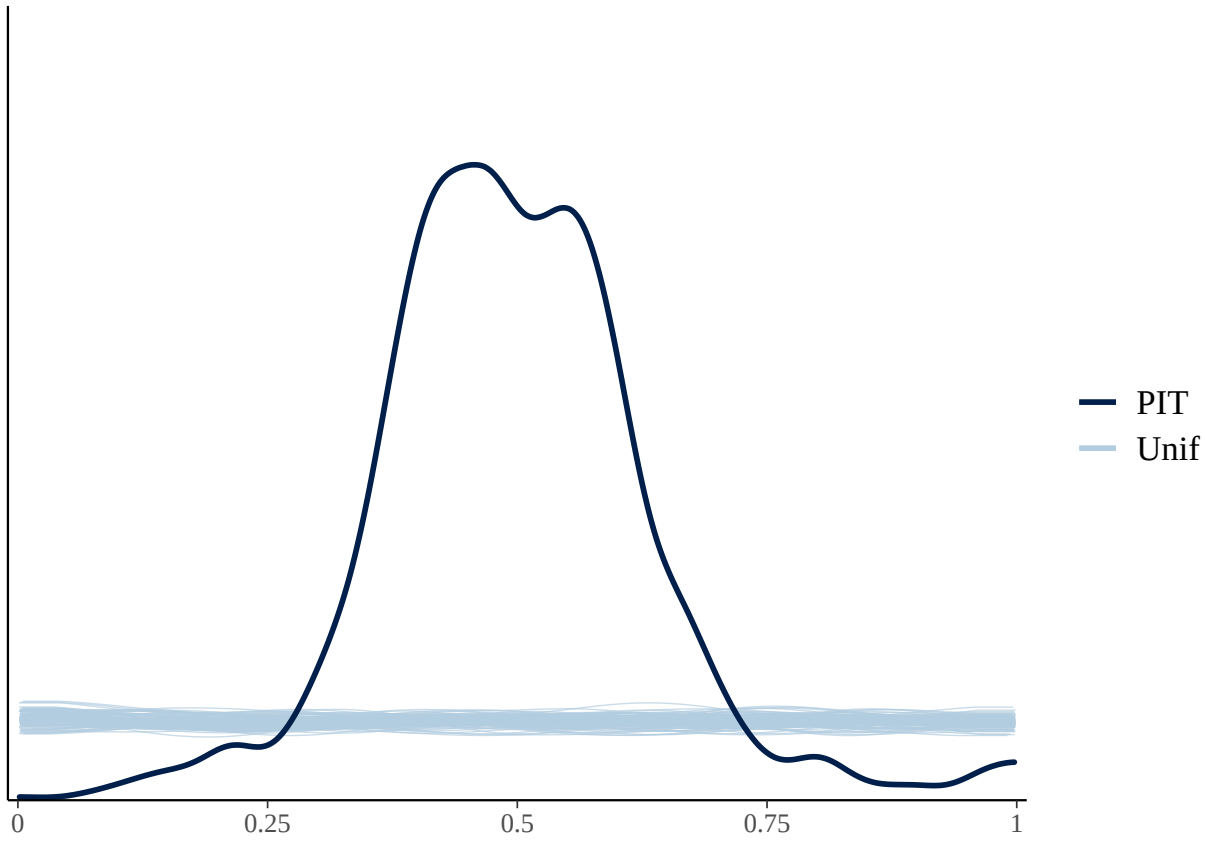



```
plot(loo2_nb)
```

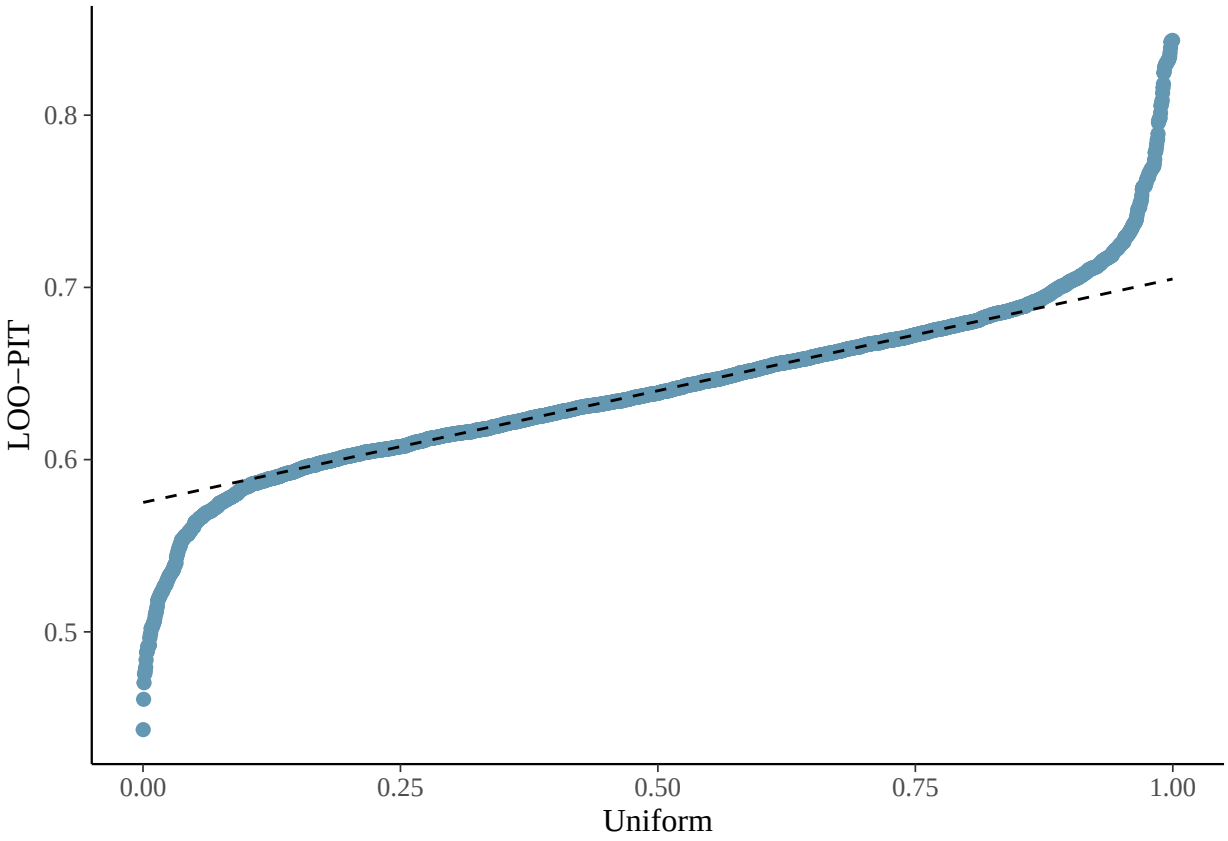


LOO predictive checks

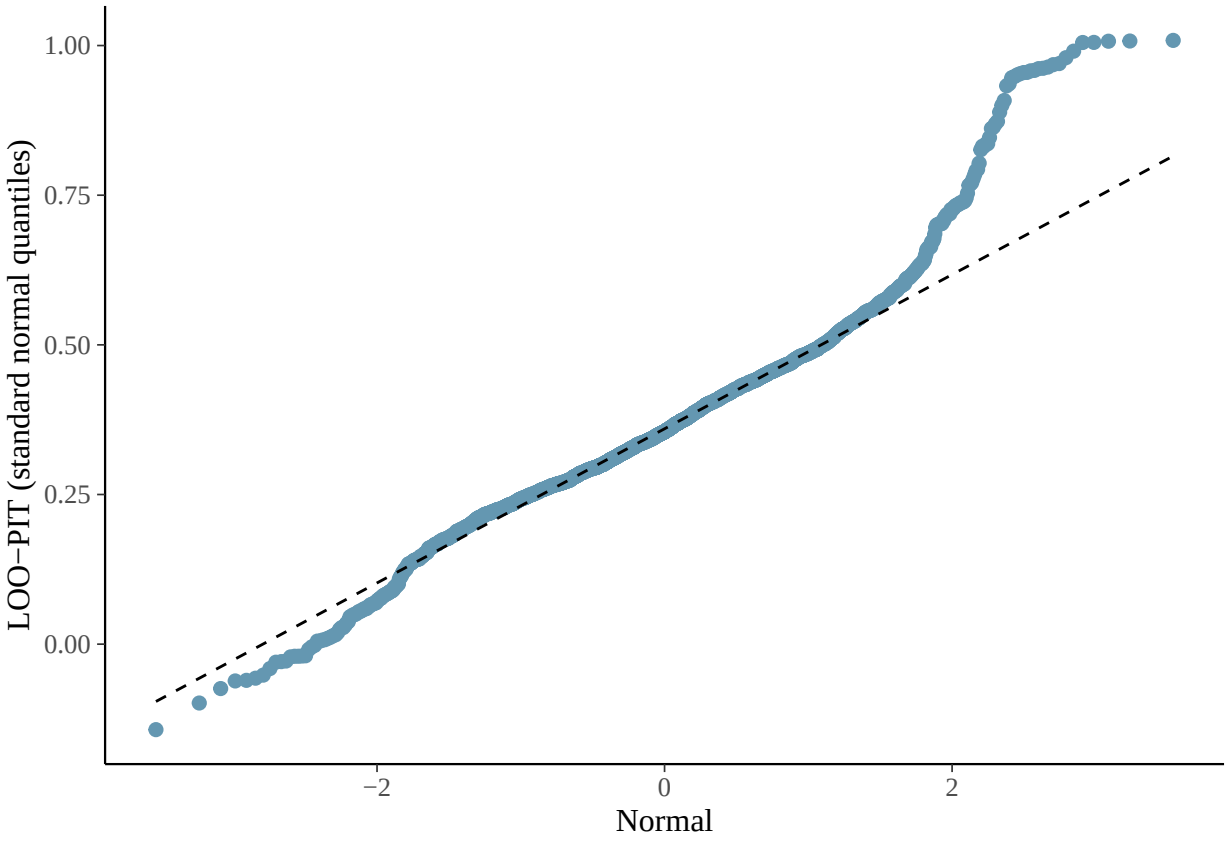
```
ppc_loo_pit_overlay(y2_fitness, yrep2_fitness_pois,
                    lw = weights(loo2_pois$psis_object))
```



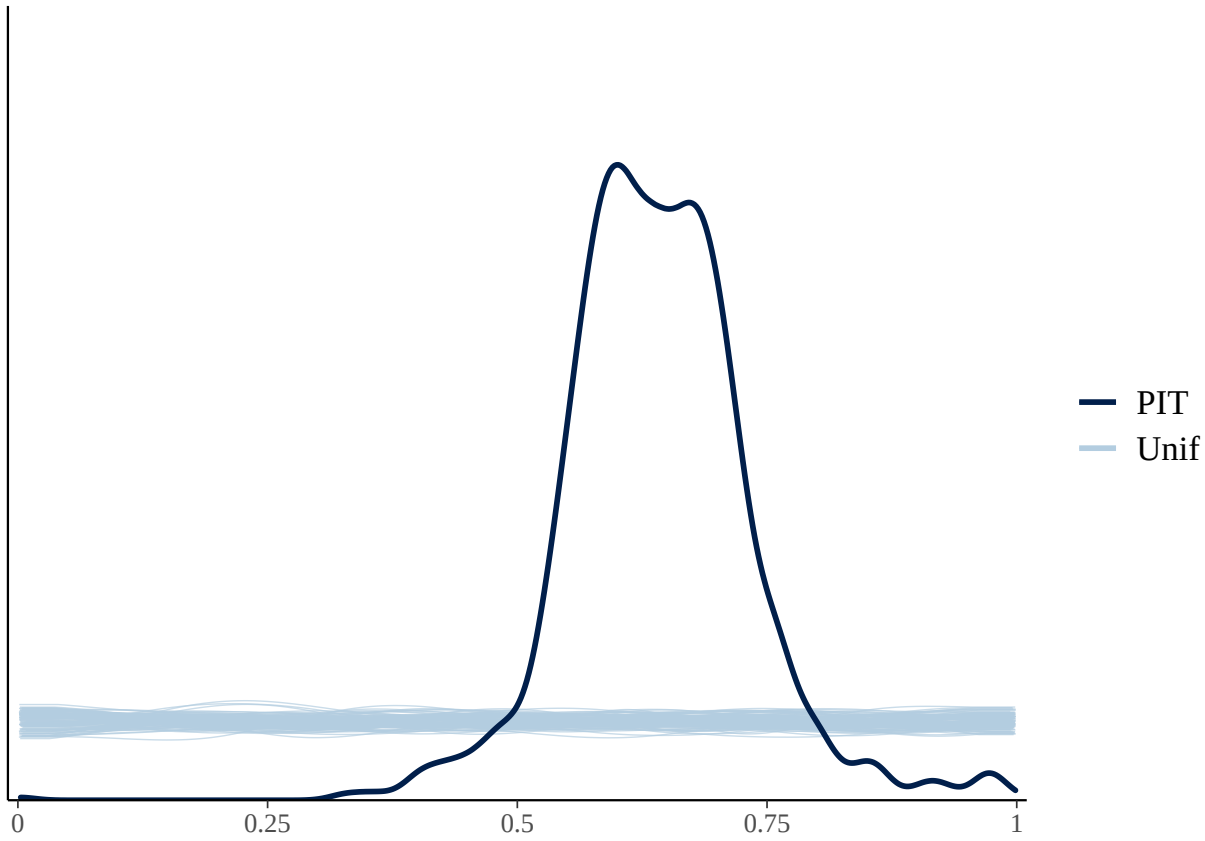

```
ppc_loo_pit_qq(y2_fitness, yrep2_fitness_pois,
                   lw = weights(loo2_pois$psis_object))
```



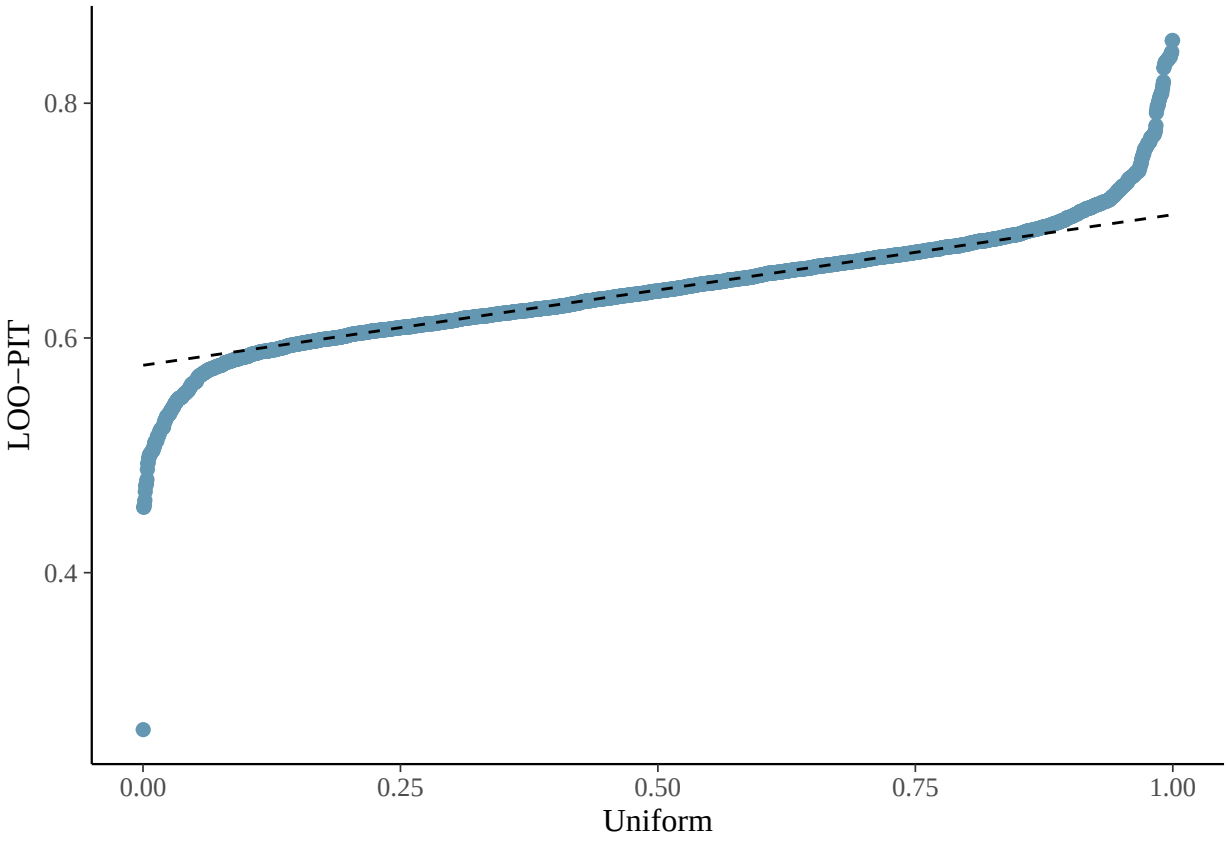

```
ppc_loo_pit_qq(y2_fitness, yrep2_fitness_pois,
                   lw = weights(loo2_pois$psis_object),compare="normal")
```



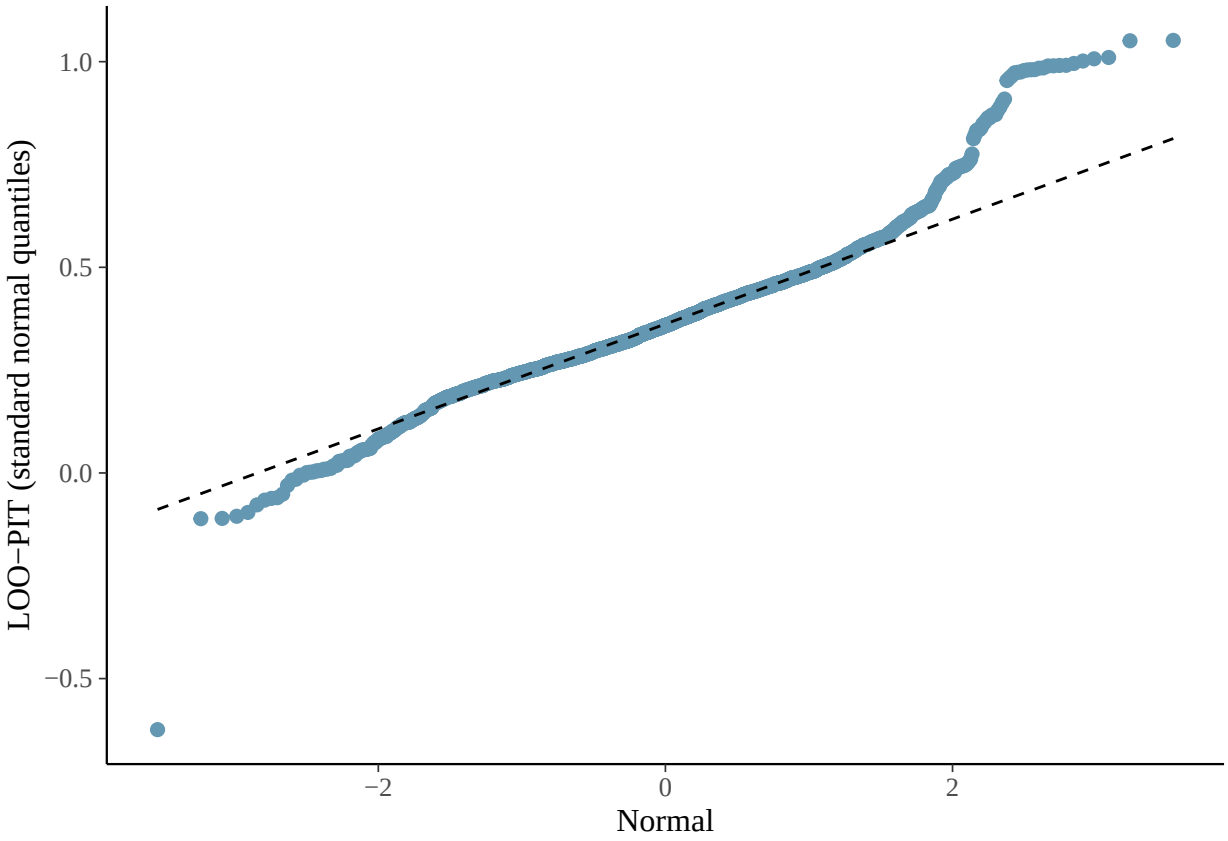

```
ppc_loo_pit_overlay(y2_fitness, yrep2_fitness_nb,
                    lw = weights(loo2_nb$psis_object))
```




```
ppc_loo_pit_qq(y2_fitness, yrep2_fitness_nb,
                   lw = weights(loo2_nb$psis_object))
```




```
ppc_loo_pit_qq(y2_fitness, yrep2_fitness_nb,
                   lw = weights(loo2_nb$psis_object),compare="normal")
```



All looking quite bad!

k-fold cross-validation (K=5):

```
loo_compare(kfold1_pois,kfold1_nb)
```

```
##                       elpd_diff se_diff
## bivar1.all.brm.pois    0.0       0.0
## bivar1.all.brm.nb     -7.8      12.7
```

Suggests that poisson is a bit better, and I guess we can trust this one as it gave no warnings?.

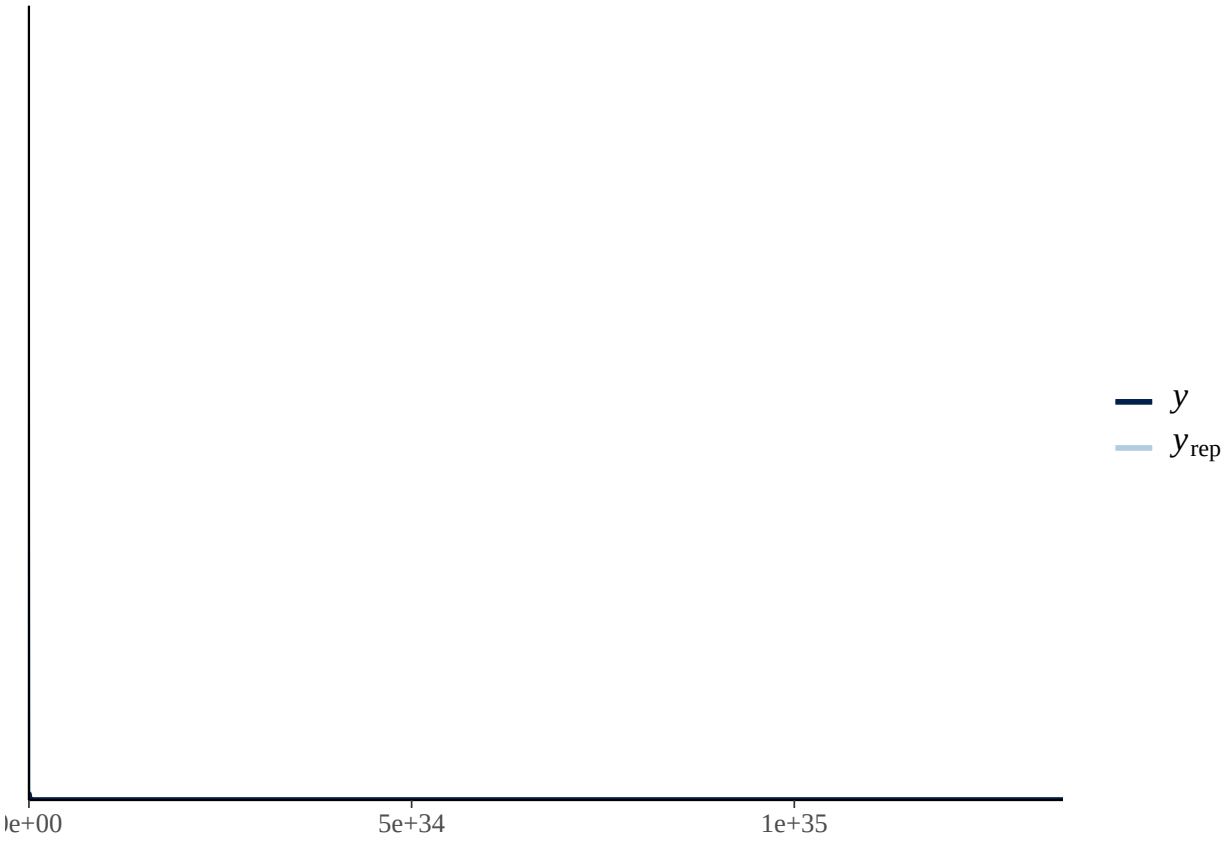
Prior predictive checks

Not sure what I am doing here...

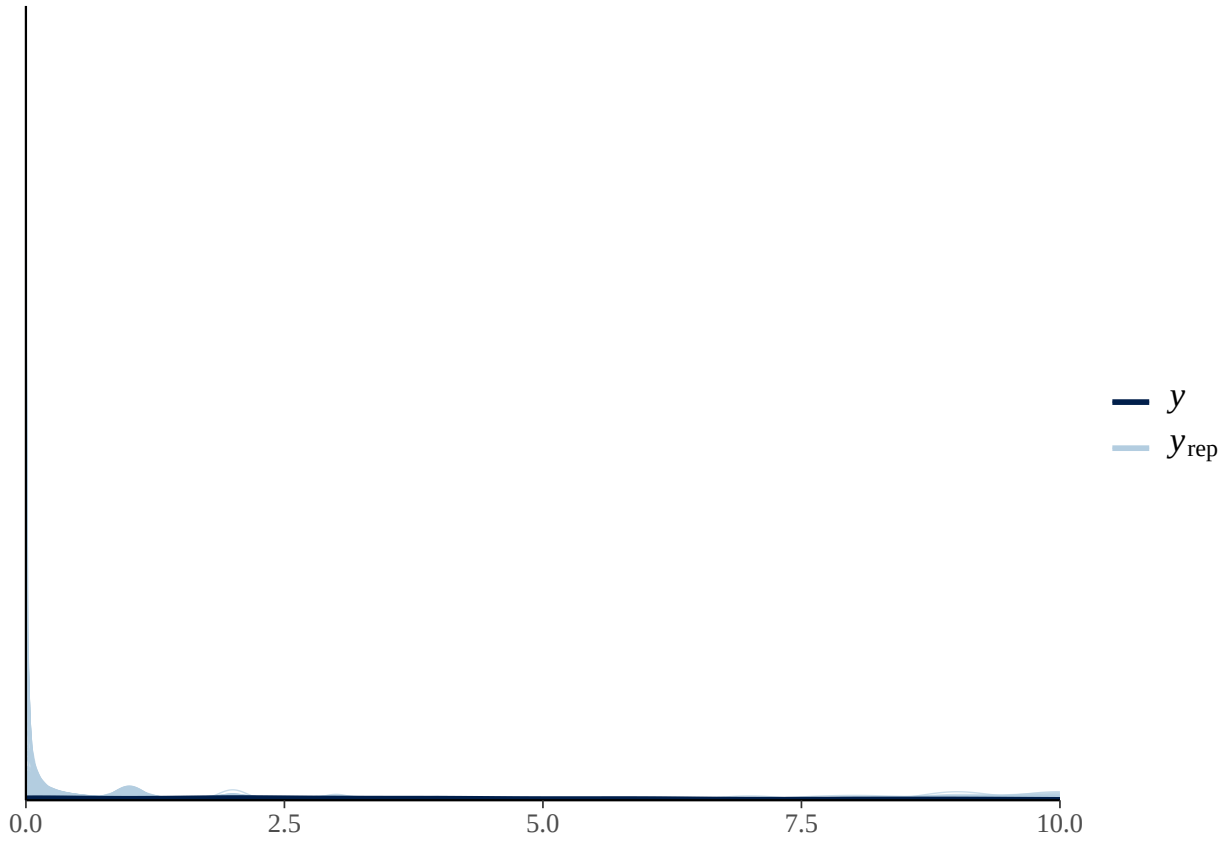
```
yrep2_fitness_pois_priors<-posterior_predict(bivar1.all.brm.pois_priors,
                                draws = 500,resp="roundmeanfitnessfl")
yrep2_FFD_pois_priors<-posterior_predict(bivar1.all.brm.pois_priors,
                                draws = 500,resp="FFD")
# matrices of draws from the posterior(prior??) predictive distribution
```

Comparison of the distribution of y and the distributions of the simulated datasets in the yrep matrix

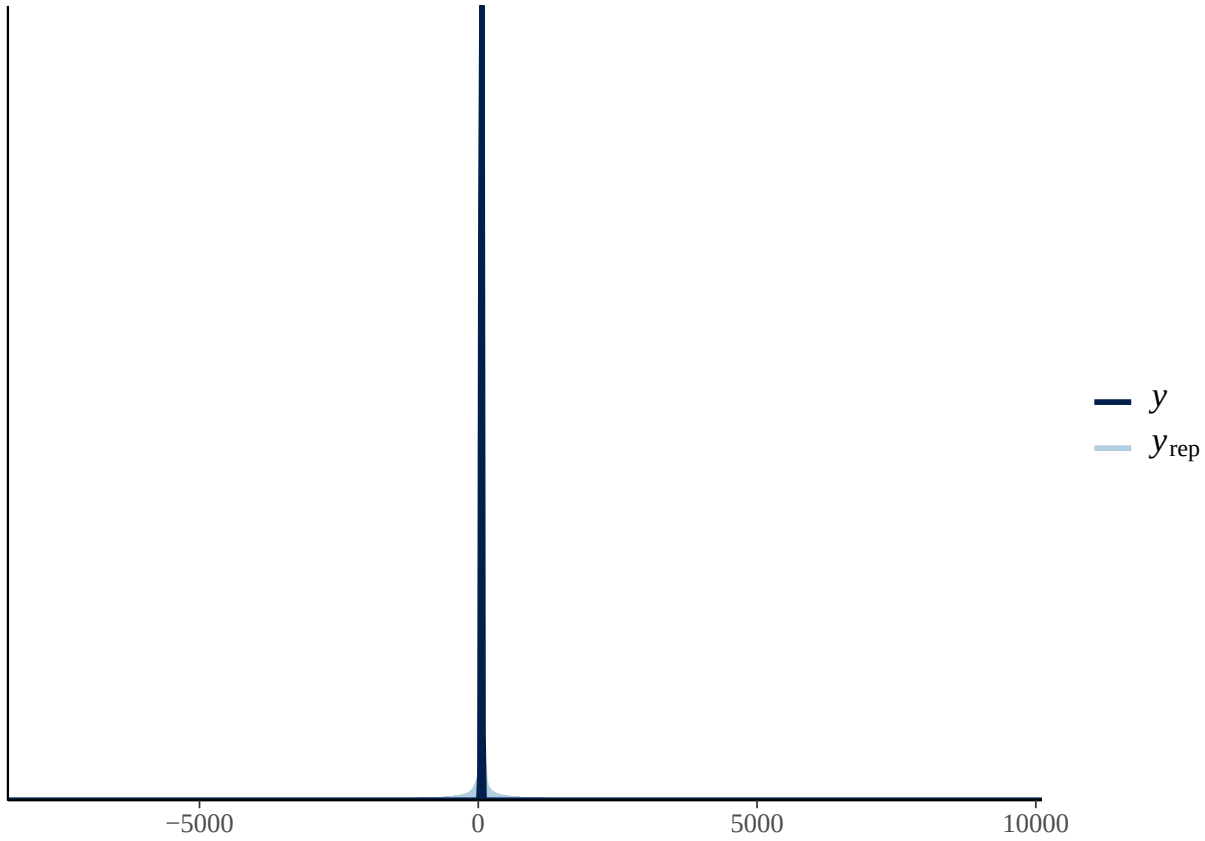
```
ppc_dens_overlay(y2_fitness, yrep2_fitness_pois_priors[1:500,])
```




```
ppc_dens_overlay(y2_fitness, yrep2_fitness_pois_priors[1:500,])+xlim(0, 10)
```




```
ppc_dens_overlay(y2_FFD, yrep2_FFD_pois_priors[1:500,])
```



Extract selection coefficients

Poisson model Using example code from Piet. I am not sure what I am doing here! The selection differentials and gradients are not so different from those obtained from MCMCglmm models so this is maybe right, but the code would need to be revised.

```
# Extract posterior samples
bivar1.all.brm.pois_post <- posterior_samples(bivar1.all.brm.pois)
bivar1.all.brm.pois_post <- as.mcmc(bivar1.all.brm.pois_post)
#head(bivar1.all.brm.pois_post)[,1:20]

# [,4] sd_id__FFD_Intercept
# [,5] sd_id__FFD_cmean_4
# [,6] sd_id__roundmeanfitnessfl_Intercept
# [,8] cor_id__FFD_Intercept__FFD_cmean_4
# [,9] cor_id__FFD_Intercept__roundmeanfitnessfl_Intercept
# [,10] cor_id__FFD_cmean_4__roundmeanfitnessfl_Intercept

# Sampling G-matrices from the posterior to calculate intervals estimates
# Use all posterior samples (no need to sample a random subset)
# Result is list with G-matrix
sample.gmat1 <- function(data, replicates = 5000) {

  ##Initialize the results list (list of lists)
  foo <- list(gmat = matrix(rep(0,3*3), ncol = 3))
```

```
  results.list <- list()
  for(j in 1:replicates) { results.list[[j]] <- foo }

  for(i in 1:replicates) {
    diag(results.list[[i]]$gmat) <- data[i,4:6]^2 #Get the diagonal

    #Upper diagonal
    results.list[[i]]$gmat[1,2] <- data[i,4]*data[i,5]*data[i,8]
    results.list[[i]]$gmat[1,3] <- data[i,4]*data[i,6]*data[i,9]
    results.list[[i]]$gmat[2,3] <- data[i,5]*data[i,6]*data[i,10]

    #Lower diagonal
    results.list[[i]]$gmat[2,1] <- results.list[[i]]$gmat[1,2]
    results.list[[i]]$gmat[3,1] <- results.list[[i]]$gmat[1,3]
    results.list[[i]]$gmat[3,2] <- results.list[[i]]$gmat[2,3]

  }

  return(results.list)
}

sampled.gmat1 <- sample.gmat1(bivar1.all.brm.pois_post, replicates = 1000)
sampled.gmat1[[2]]
```

```
## $gmat
##            [,1]        [,2]        [,3]
## [1,]  2.0024010  0.71950092 -1.12721914
## [2,]  0.7195009  0.40350332 -0.07387394
## [3,] -1.1272191 -0.07387394  1.85439970
```

```
sgmat1 <- lapply(sampled.gmat1, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat1 <- unname(sapply(sgmat1, '[[', 1)) #Change to matrix
str(sgmat1)
```

```
##  num [1:9, 1:1000] 1.979 0.623 -1.176 0.623 0.467 ...
```

```
sgmat1 <- t(sgmat1)

P.modelBV_RR1 <- sgmat1
P.modelBV_RR1.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR1.mode[k] <- posterior.mode(mcmc(sgmat1[,k]))
P.modelBV_RR1.mode
```

```
##            [,1]        [,2]        [,3]
## [1,]  2.3784864  0.91794834 -1.12681855
## [2,]  0.9179483  0.52394922 -0.03497739
## [3,] -1.1268185 -0.03497739  1.87639668
```

```
# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR1 <- sgmat1[,c(3,6)]
```

```r
colnames(S.modelBV_RR1) <- c("S_intercepts", "S_slopes")
S.modelBV_RR1.mode <- P.modelBV_RR1.mode[1:2, 3]
S.modelBV_RR1.mode
```

```
## [1] -1.12681855 -0.03497739
```

```r
posterior.mode(mcmc(S.modelBV_RR1))
```

```
## S_intercepts     S_slopes
##  -1.12681855  -0.03497739
```

```r
HPDinterval(mcmc(S.modelBV_RR1))
```

```
##                   lower      upper
## S_intercepts -1.5478386 -0.8355278
## S_slopes     -0.3316333  0.2289710
## attr(,"Probability")
## [1] 0.95
```

```r
# Estimate selection gradients for intercept and slope (beta = S / P)
# on each sample of posterior and extract their mode
beta_post_RR1 <- matrix(NA, nrow(S.modelBV_RR1) ,2)

for (i in 1:nrow(S.modelBV_RR1)) {
  P3_1 <- matrix(rep(NA, 9), nrow = 3)
  # 3x3 matrix of var-cov for individual X.int, X.slope and fitness
  for (k in 1:9) {P3_1[k] <- P.modelBV_RR1[i, k] }
  P2_1 <- P3_1[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S1 <- P3_1[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR1[i,] <- solve(P2_1) %*% S1   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR1) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR1))
```

```
## beta_intercepts     beta_slopes
##      -0.7303435       0.7299346
```

```r
HPDinterval(mcmc(beta_post_RR1))
```

```
##                     lower     upper
## beta_intercepts -1.7302444 -0.389902
## beta_slopes      0.1266127  3.364986
## attr(,"Probability")
## [1] 0.95
```

```
# Extract posterior samples
bivar1.all.brm.nb_post <- posterior_samples(bivar1.all.brm.nb)
bivar1.all.brm.nb_post <- as.mcmc(bivar1.all.brm.nb_post)
#head(bivar1.all.brm.nb_post)[,1:20]

# [,4] sd_id__FFD_Intercept
# [,5] sd_id__FFD_cmean_4
# [,6] sd_id__roundmeanfitnessfl_Intercept
# [,8] cor_id__FFD_Intercept__FFD_cmean_4
# [,9] cor_id__FFD_Intercept__roundmeanfitnessfl_Intercept
# [,10] cor_id__FFD_cmean_4__roundmeanfitnessfl_Intercept

sampled.gmat2 <- sample.gmat1(bivar1.all.brm.nb_post, replicates = 1000)
sampled.gmat2[[2]]
```

Negative binomial model

```
## $gmat
##            [,1]       [,2]      [,3]
## [1,]  2.8885359  0.8188655 -1.263859
## [2,]  0.8188655  1.0081979 -0.150029
## [3,] -1.2638593 -0.1500290  1.935479
```

```
sgmat2 <- lapply(sampled.gmat2, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat2 <- unname(sapply(sgmat2, '[[', 1)) #Change to matrix
str(sgmat2)
```

```
##  num [1:9, 1:1000] 3.147 0.905 -1.242 0.905 0.853 ...
```

```
sgmat2 <- t(sgmat2)

P.modelBV_RR2 <- sgmat2
P.modelBV_RR2.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR2.mode[k] <- posterior.mode(mcmc(sgmat2[,k]))
P.modelBV_RR2.mode
```

```
##            [,1]        [,2]        [,3]
## [1,]  2.9699133  0.84618715 -1.12521475
## [2,]  0.8461872  0.61682725 -0.02533719
## [3,] -1.1252147 -0.02533719  1.93077596
```

```
# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR2 <- sgmat2[,c(3,6)]
colnames(S.modelBV_RR2) <- c("S_intercepts", "S_slopes")
S.modelBV_RR2.mode <- P.modelBV_RR2.mode[1:2, 3]
S.modelBV_RR2.mode
```

```
## [1] -1.12521475 -0.02533719
```

```
posterior.mode(mcmc(S.modelBV_RR2))
```

```
## S_intercepts     S_slopes
##  -1.12521475  -0.02533719
```

```
HPDinterval(mcmc(S.modelBV_RR2))
```

```
##                   lower      upper
## S_intercepts -1.5420596 -0.8232000
## S_slopes     -0.3158794  0.2477157
## attr(,"Probability")
## [1] 0.95
```

```
# Estimate selection gradients for intercept and slope (beta = S / P)
# on each sample of posterior and extract their mode
beta_post_RR2 <- matrix(NA, nrow(S.modelBV_RR2) ,2)

for (i in 1:nrow(S.modelBV_RR2)) {
  P3_2 <- matrix(rep(NA, 9), nrow = 3)
  # 3x3 matrix of var-cov for individual X.int, X.slope and fitness
  for (k in 1:9) {P3_2[k] <- P.modelBV_RR2[i, k] }
  P2_2 <- P3_2[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S2 <- P3_2[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR2[i,] <- solve(P2_2) %*% S2   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR2) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR2))
```

```
## beta_intercepts     beta_slopes
##      -0.7696616       0.6759190
```

```
HPDinterval(mcmc(beta_post_RR2))
```

```
##                        lower      upper
## beta_intercepts -1.59536332 -0.3417754
## beta_slopes     -0.02078576  3.0013848
## attr(,"Probability")
## [1] 0.95
```

2. mean_fitness_fl, with shoot volume

```
bf_fitness_shoot <- bf(round(mean_fitness_fl) ~  cn_shoot_vol_mean_sqrt +
                         (1|ID1|id)) # Set up model formula
```

Poisson distribution

```
bivar2.all.brm.pois<-brm(bf_FFD+bf_fitness_shoot,
                         family = c(gaussian, poisson),
                         data = datadef,warmup = 2000,iter = 6000,chains=4,thin=2,
                         inits = "random",seed = 12345,cores = my.cores,
                         control = list(adapt_delta = 0.99, max_treedepth = 15))
```

```
summary(bivar2.all.brm.pois)
```

```
##  Family: MV(gaussian, poisson)
##   Links: mu = identity; sigma = identity
##          mu = log
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_fl) ~ cn_shoot_vol_mean_sqrt + (1 | ID1 | id)
##    Data: datadef (Number of observations: 2432)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
##
## Group-Level Effects:
## ~id (Number of levels: 791)
##                                                 Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                   1.01      0.78     0.00
## sd(FFD_cmean_4)                                     1.55      1.26     0.58
## sd(roundmeanfitnessfl_Intercept)                    0.97      0.48     0.15
## cor(FFD_Intercept,FFD_cmean_4)                      0.55      0.45    -0.75
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)    -0.40      0.46    -0.91
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)      -0.08      0.43    -0.78
##                                                 u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)                                   1.92 1.66        7       18
## sd(FFD_cmean_4)                                     3.72 1.67        6       11
## sd(roundmeanfitnessfl_Intercept)                    1.34 1.53        7       31
## cor(FFD_Intercept,FFD_cmean_4)                      0.93 2.18        5       12
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)     0.63 2.06        5       19
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)       0.40 1.63        7       13
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.62      1.08     3.86     7.12 1.32       10      142
##
## Population-Level Effects:
##                                         Estimate Est.Error l-95% CI u-95% CI
## FFD_Intercept                              44.44     24.58     1.90    60.75
## roundmeanfitnessfl_Intercept                0.03      1.32    -2.40     0.88
## FFD_cmean_4                                -1.59      1.60    -3.95     0.88
## roundmeanfitnessfl_cn_shoot_vol_mean_sqrt    0.06      0.04     0.03     0.14
##                                         Rhat Bulk_ESS Tail_ESS
## FFD_Intercept                           1.60        7       11
## roundmeanfitnessfl_Intercept            1.60        7       11
## FFD_cmean_4                             1.53        7       26
## roundmeanfitnessfl_cn_shoot_vol_mean_sqrt 1.69        6       11
##
## Family Specific Parameters:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
```

```
## sigma_FFD     3.43      1.70     0.48     4.68 2.14        5       11
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Due to these warnings, results of bivar2.all.brm.pois are probably not reliable! Try RERUN with above specifications.

Negative binomial distribution

```
bivar2.all.brm.nb<-brm(bf_FFD+bf_fitness_shoot,
                       family = c(gaussian,negbinomial),
                       data = datadef,warmup = 1000,iter = 4000,chains=4,thin=2,
                       inits = "random",seed = 12345,cores = my.cores,
                       control = list(adapt_delta = 0.99))
```

```
summary(bivar2.all.brm.nb)
```

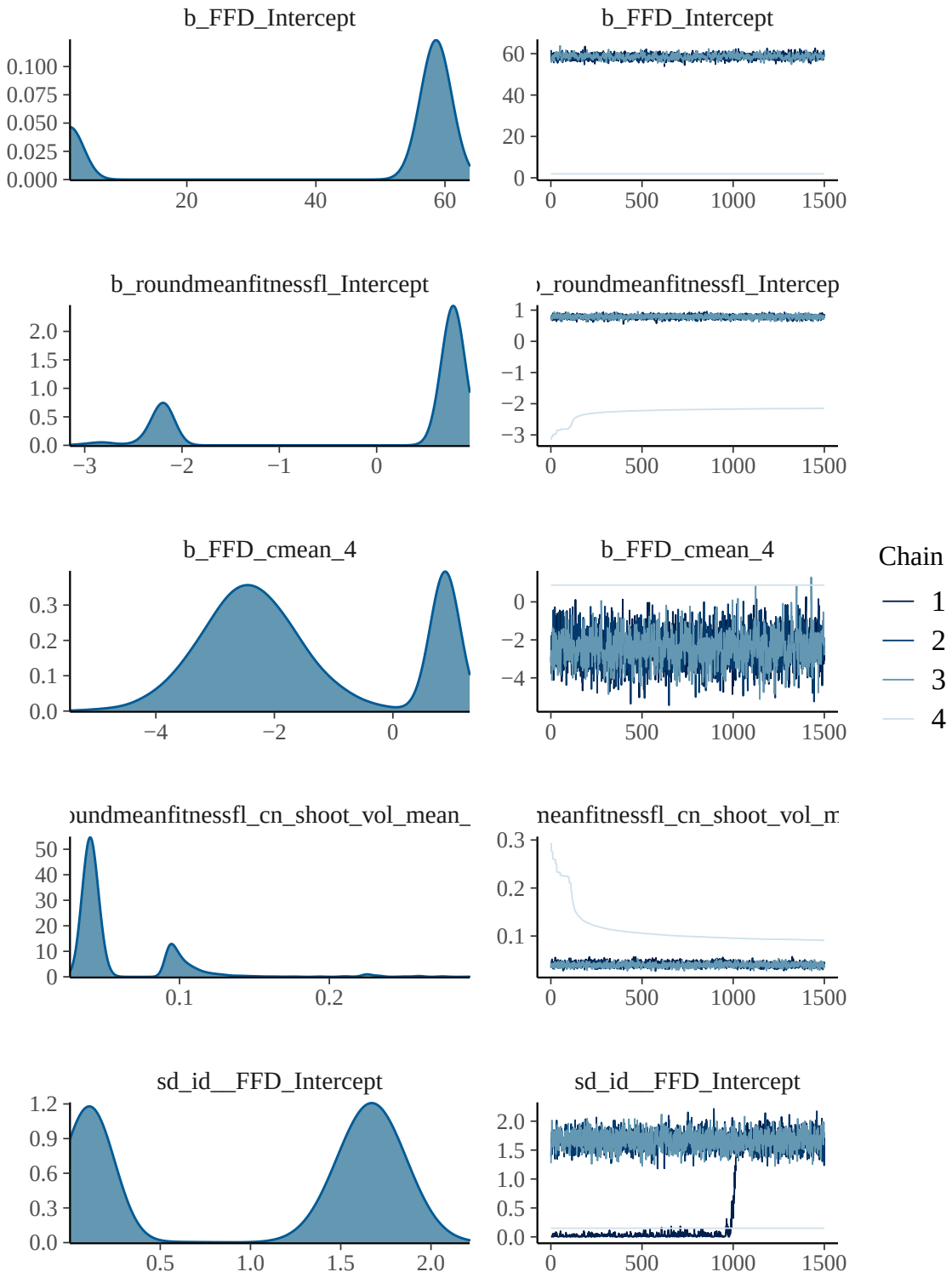
```
##  Family: MV(gaussian, negbinomial)
##   Links: mu = identity; sigma = identity
##          mu = log; shape = identity
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_fl) ~ cn_shoot_vol_mean_sqrt + (1 | ID1 | id)
##    Data: datadef (Number of observations: 2432)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
##
## Group-Level Effects:
## ~id (Number of levels: 791)
##                                                 Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                   1.67      0.15     1.38
## sd(FFD_cmean_4)                                     0.80      0.13     0.55
## sd(roundmeanfitnessfl_Intercept)                    1.24      0.05     1.15
## cor(FFD_Intercept,FFD_cmean_4)                      0.64      0.14     0.35
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)    -0.45      0.08    -0.59
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)       0.16      0.13    -0.10
##                                                 u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)                                   1.96 1.00     2888     4365
## sd(FFD_cmean_4)                                     1.07 1.00     1189     4337
## sd(roundmeanfitnessfl_Intercept)                    1.34 1.00     2969     4573
## cor(FFD_Intercept,FFD_cmean_4)                      0.87 1.00      580     3008
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)    -0.29 1.00     1358     2543
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)       0.43 1.00     1115     3771
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.22      0.88     3.82     7.15 1.00     2461     3998
##
## Population-Level Effects:
##                                       Estimate Est.Error l-95% CI u-95% CI
```

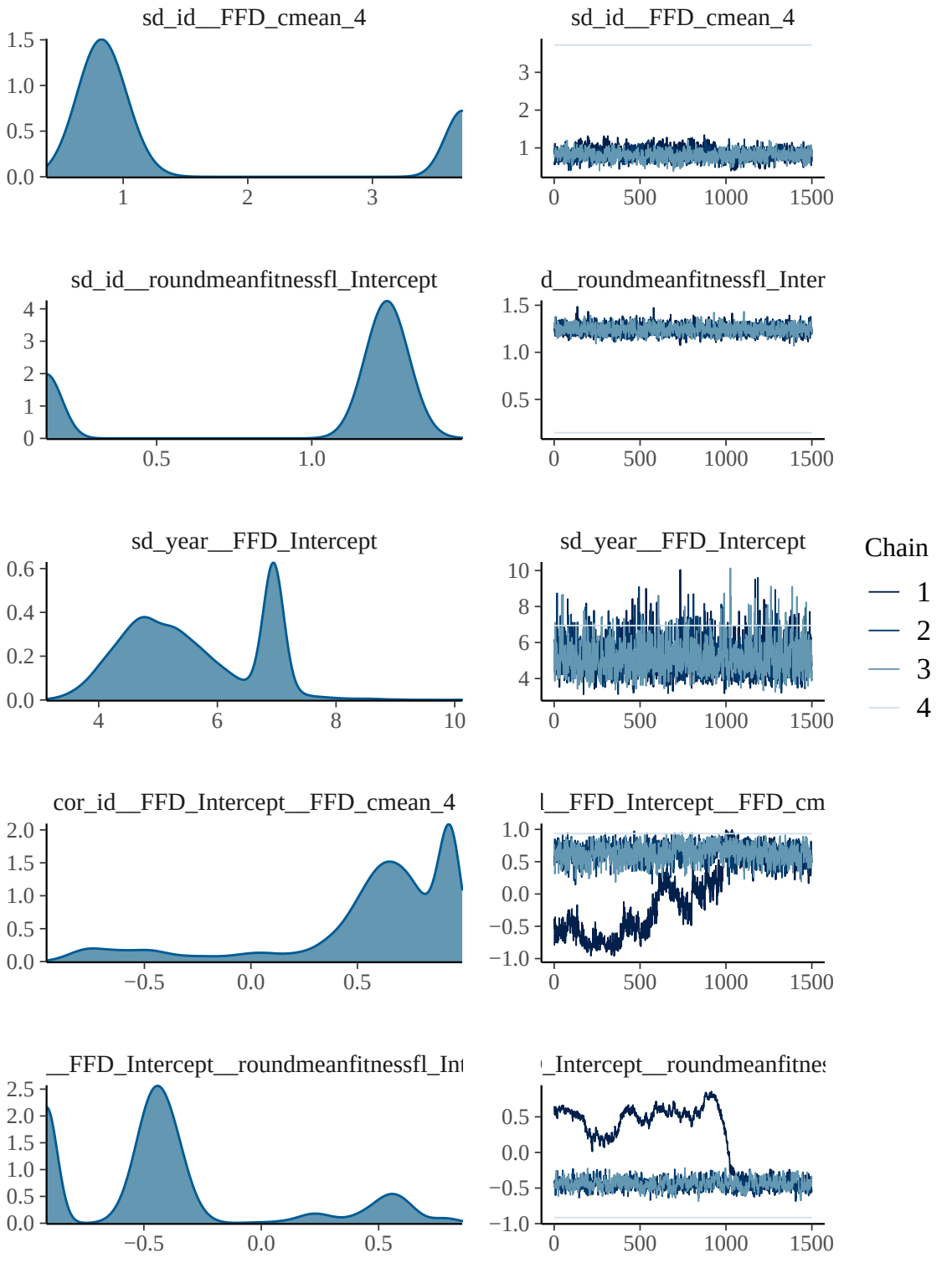
```
## FFD_Intercept                                58.64      1.12    56.47    60.87
## roundmeanfitnessfl_Intercept                  0.79      0.05     0.69     0.90
## FFD_cmean_4                                  -2.40      0.83    -4.05    -0.79
## roundmeanfitnessfl_cn_shoot_vol_mean_sqrt     0.04      0.00     0.03     0.05
##                                         Rhat Bulk_ESS Tail_ESS
## FFD_Intercept                           1.00     1670     3037
## roundmeanfitnessfl_Intercept            1.00     4645     5474
## FFD_cmean_4                             1.00     2309     3430
## roundmeanfitnessfl_cn_shoot_vol_mean_sqrt 1.00     4893     5127
##
## Family Specific Parameters:
##                          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sigma_FFD                    4.35      0.07     4.20     4.50 1.00     2031
## shape_roundmeanfitnessfl   673.57    185.20   381.51  1104.43 1.00     4745
##                          Tail_ESS
## sigma_FFD                    4717
## shape_roundmeanfitnessfl     4427
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

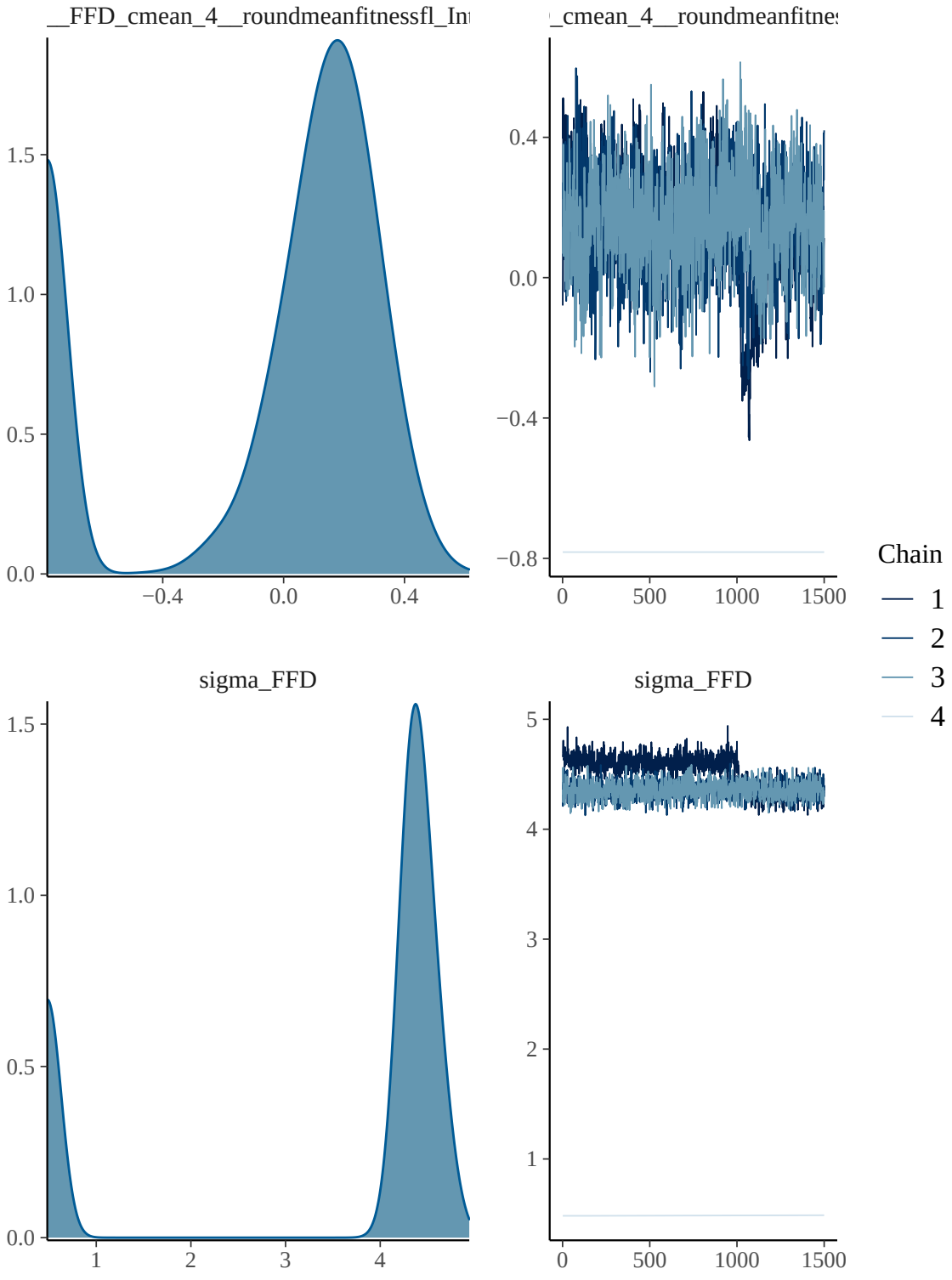
No warnings! :)

Model evaluation: Compare models

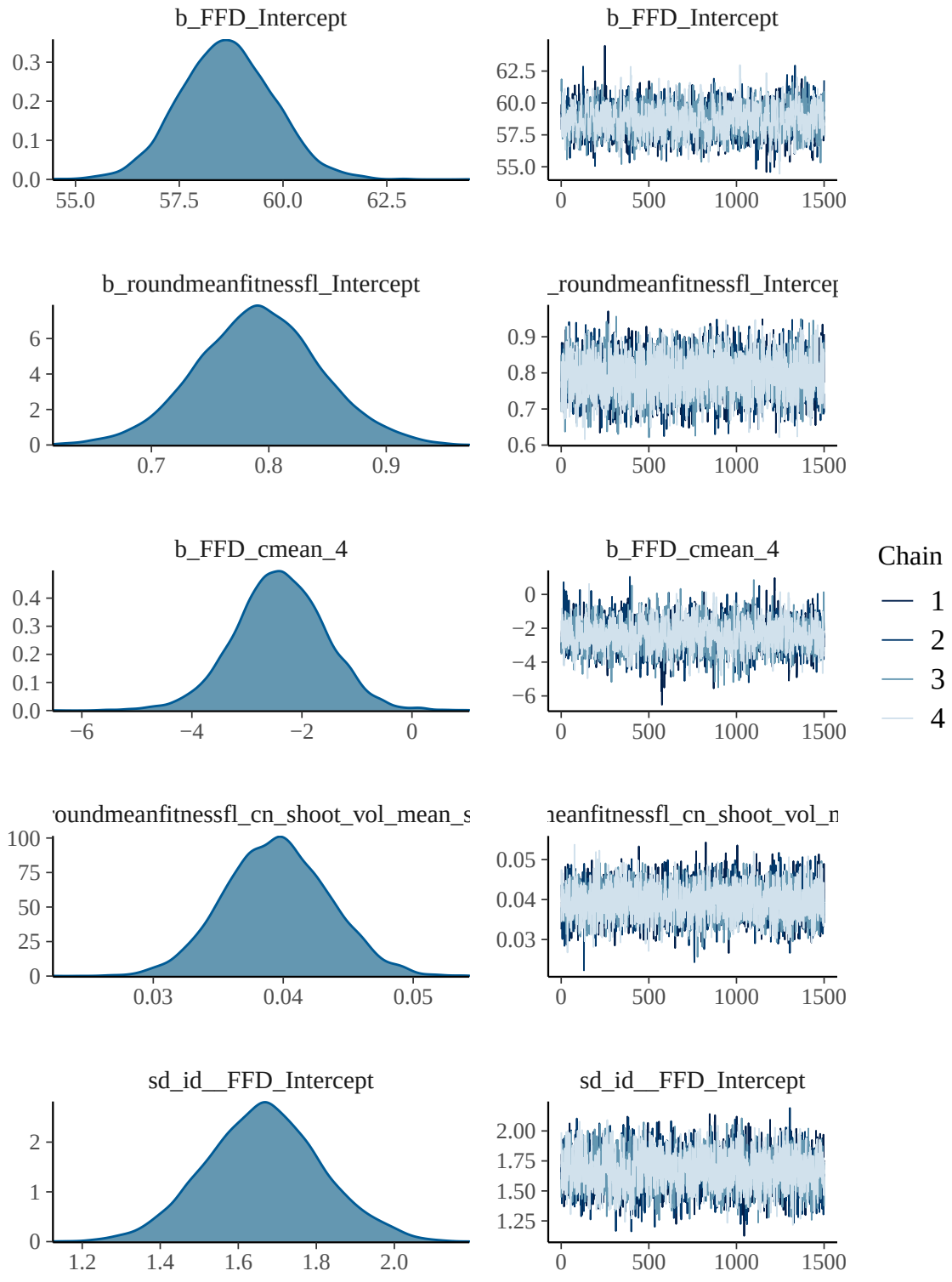
```
plot(bivar2.all.brm.pois)
```

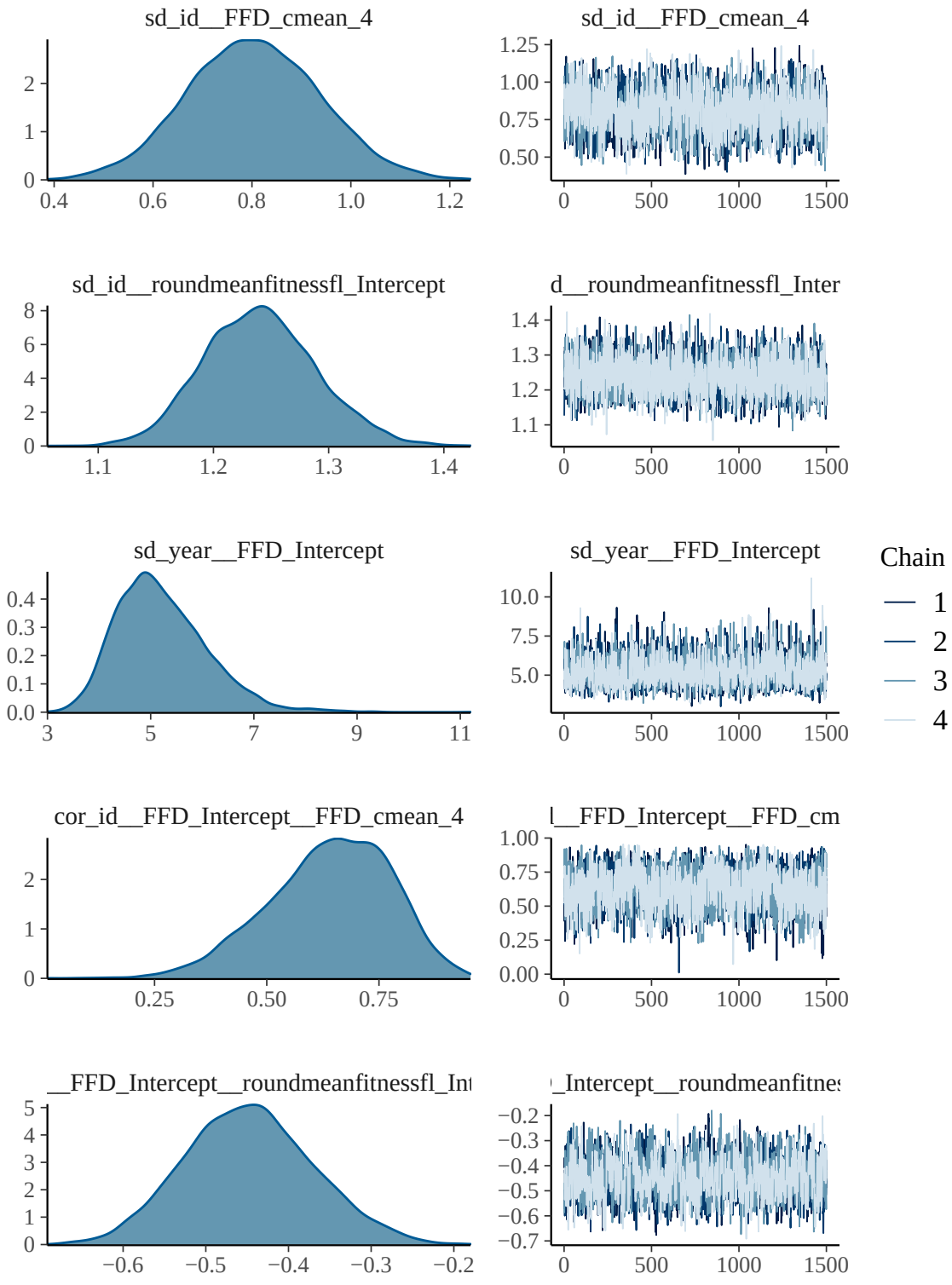

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessfl_Intercept
b_roundmeanfitnessfl_Intercep
b_FFD_cmean_4
b_FFD_cmean_4
Chain
1
2
3
4
oundmeanfitnessfl_cn_shoot_vol_mean_
neanfitnessfl_cn_shoot_vol_m
sd_id__FFD_Intercept
sd_id__FFD_Intercept

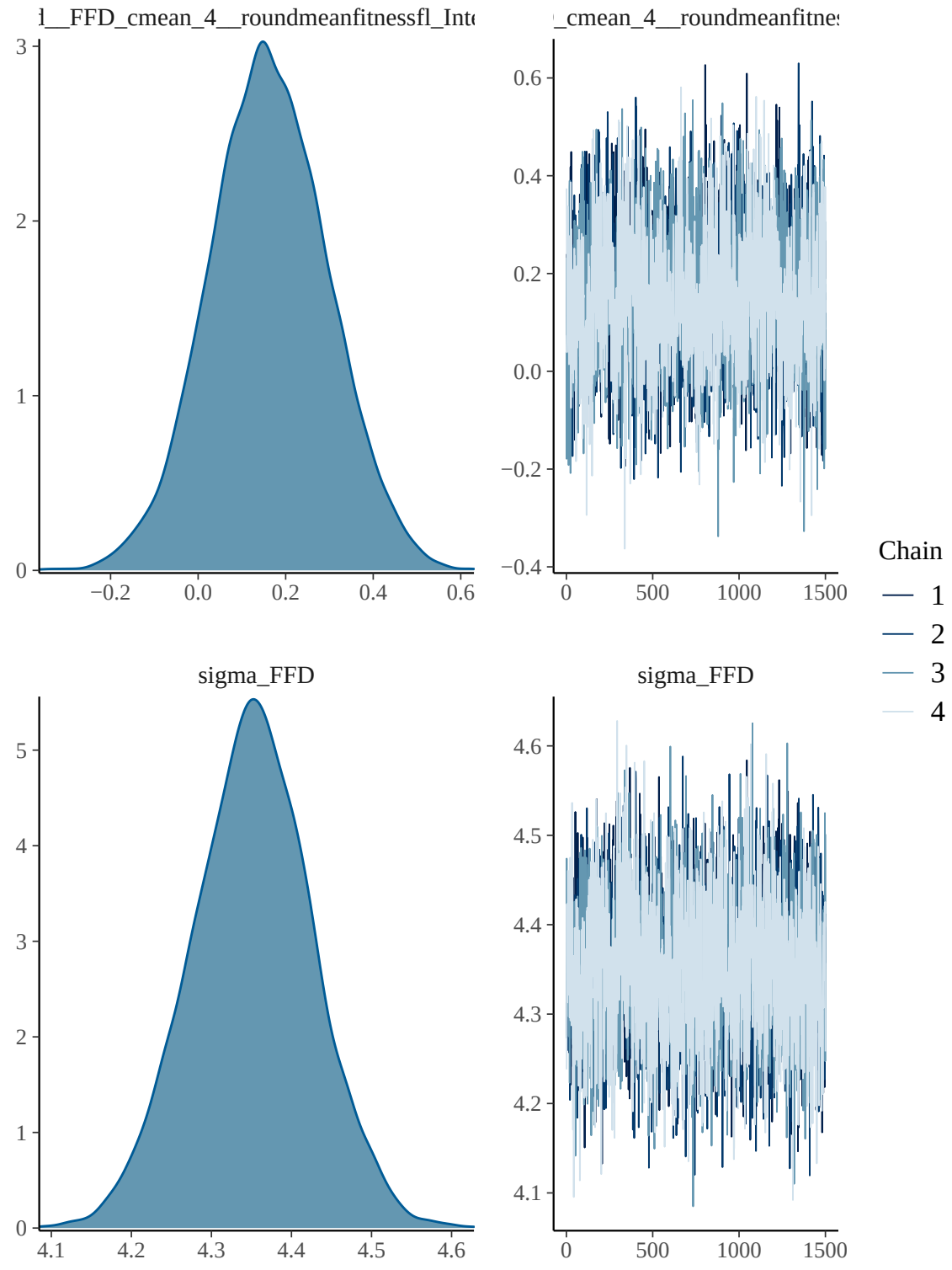

sd_id__FFD_cmean_4
sd_id__FFD_cmean_4
sd_id__roundmeanfitnessfl_Intercept
d__roundmeanfitnessfl_Inter
sd_year__FFD_Intercept
sd_year__FFD_Intercept
Chain
1
2
3
4
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
__FFD_Intercept__roundmeanfitnessfl_Int
_Intercept__roundmeanfitnes




```
plot(bivar2.all.brm.nb)
```

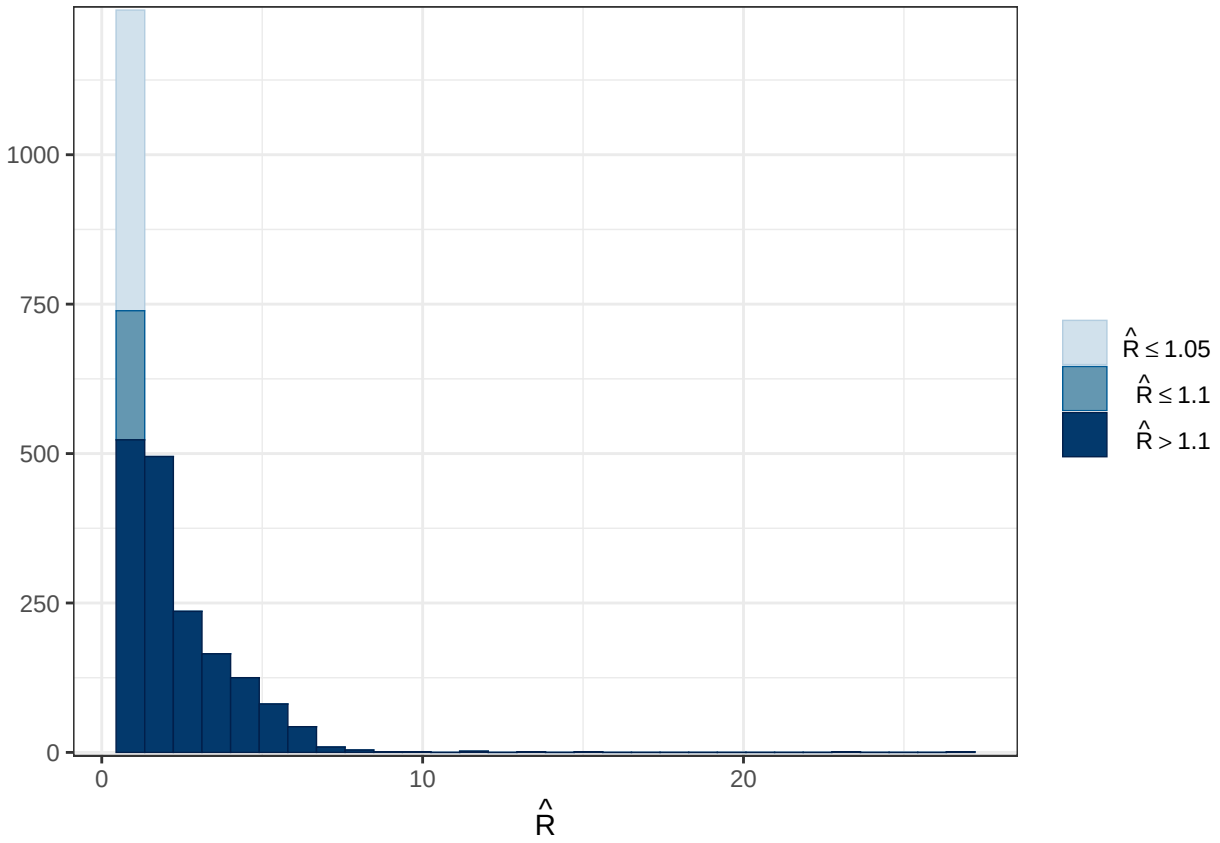

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessfl_Intercept
_roundmeanfitnessfl_Intercep
b_FFD_cmean_4
b_FFD_cmean_4
oundmeanfitnessfl_cn_shoot_vol_mean_s
eanfitnessfl_cn_shoot_vol_n
sd_id__FFD_Intercept
sd_id__FFD_Intercept
Chain
1
2
3
4


sd_id__FFD_cmean_4
sd_id__FFD_cmean_4
sd_id__roundmeanfitnessfl_Intercept
d__roundmeanfitnessfl_Inter
sd_year__FFD_Intercept
sd_year__FFD_Intercept
Chain
1
2
3
4
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
__FFD_Intercept__roundmeanfitnessfl_Int
_Intercept__roundmeanfitnes

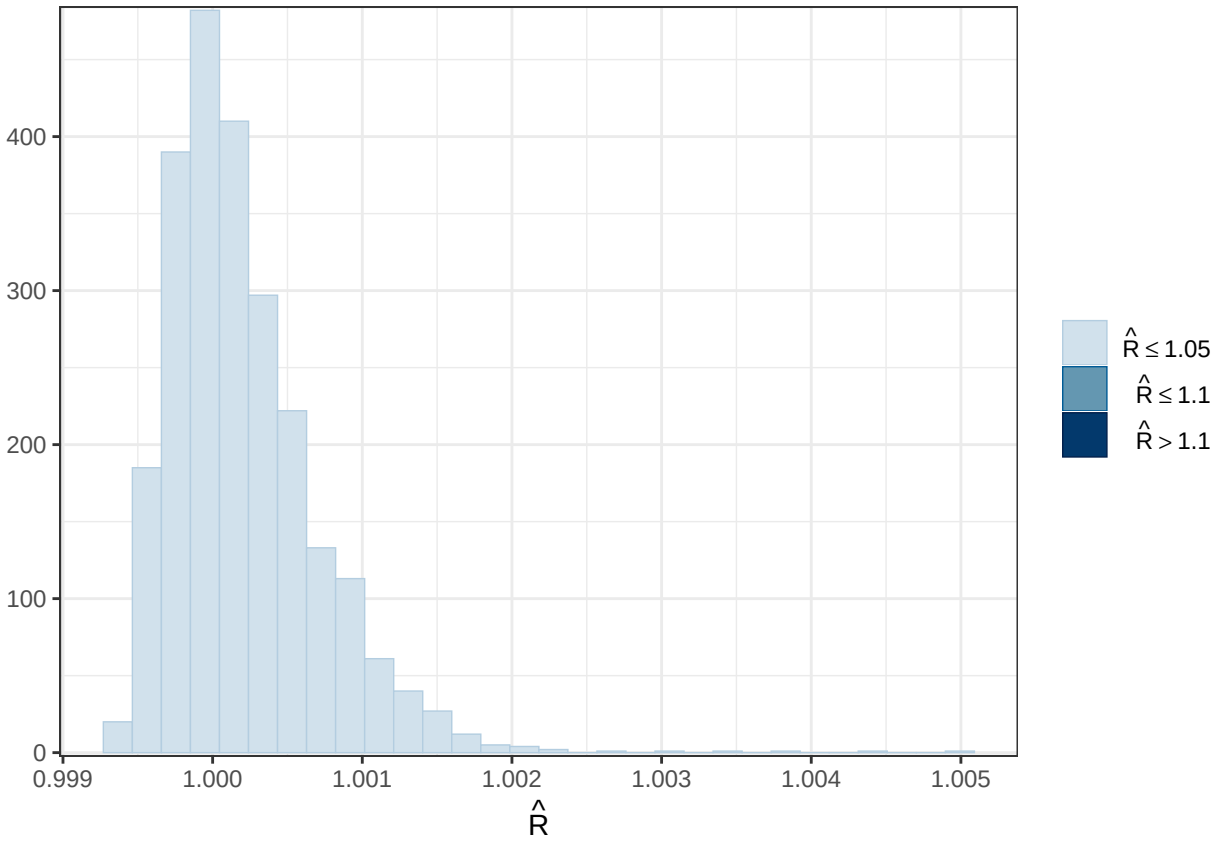

l__FFD_cmean_4__roundmeanfitnessfl_Int
_cmean_4__roundmeanfitnes
sigma_FFD
sigma_FFD
Chain
1
2
3
4

Rhat

```
mcmc_rhat_hist(rhat(bivar2.all.brm.pois))+theme_bw()
```

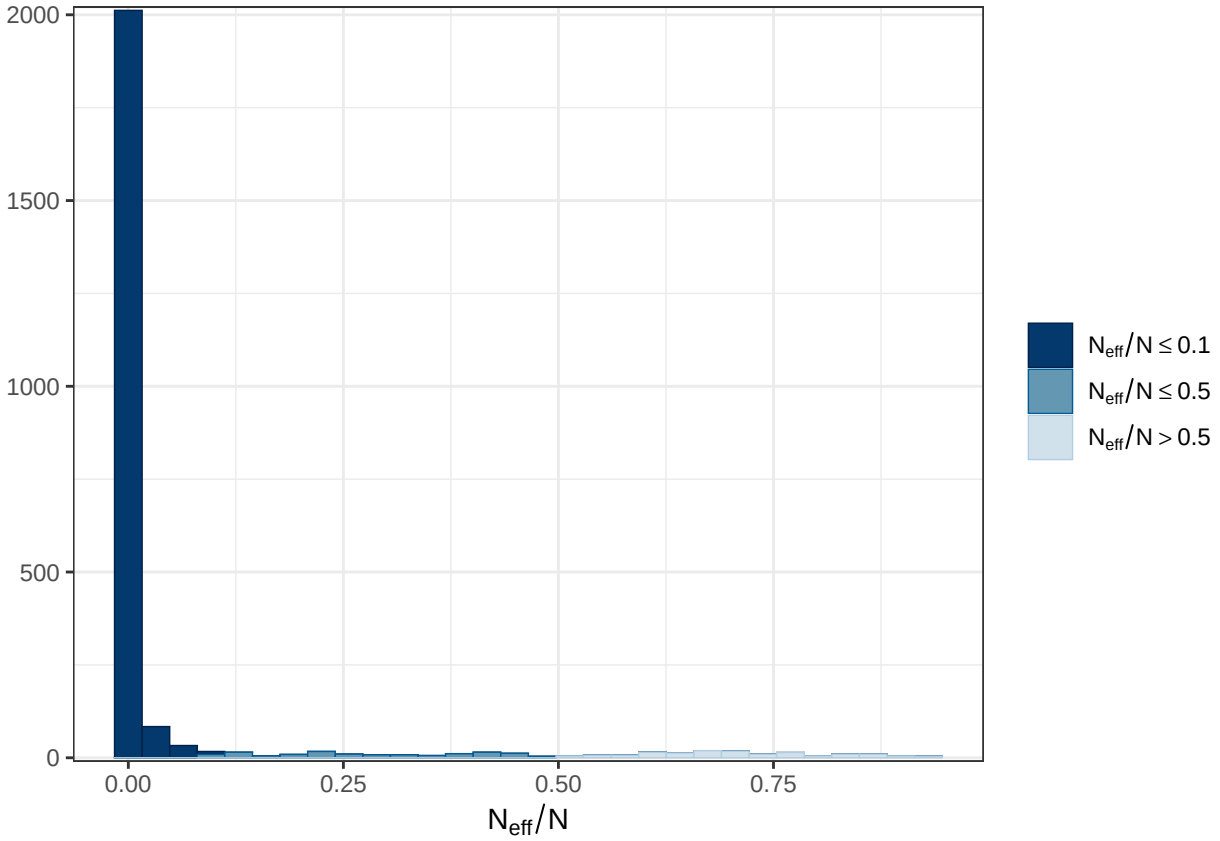



```
mcmc_rhat_hist(rhat(bivar2.all.brm.nb))+theme_bw()
```

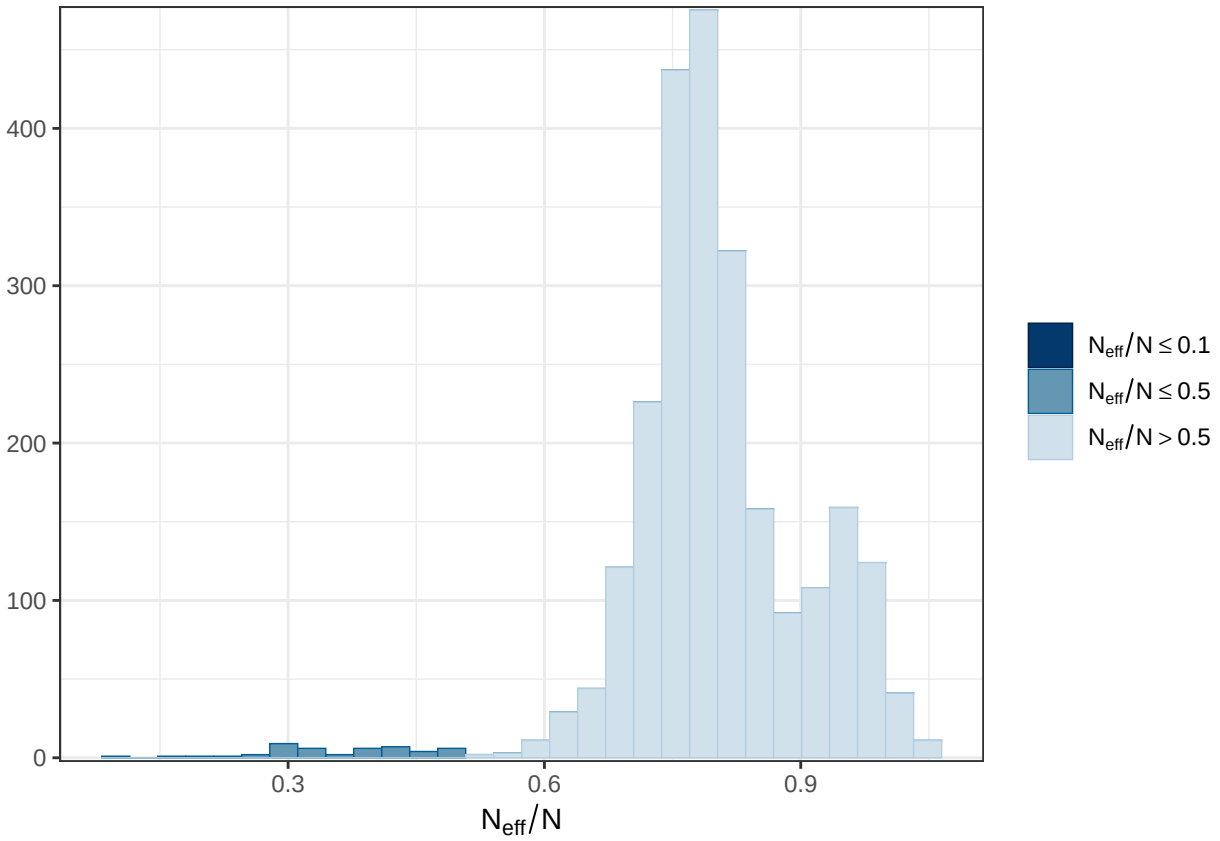


Effective sample size:

```
mcmc_neff_hist(neff_ratio(bivar2.all.brm.pois))+theme_bw()
```




```
mcmc_neff_hist(neff_ratio(bivar2.all.brm.nb))+theme_bw()
```



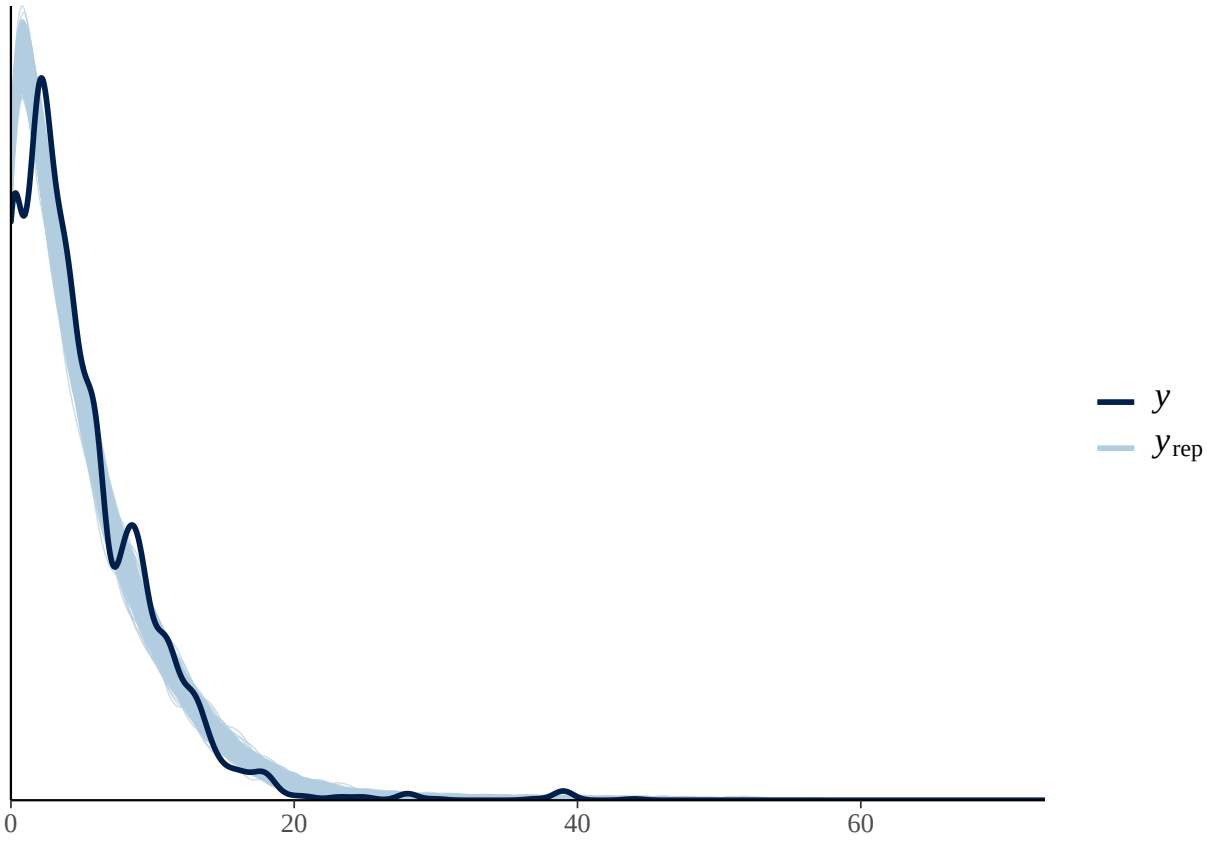
Posterior predictive checks:

```
y3_fitness<-round(subset(datadef,!is.na(FFD)&
                           !is.na(cn_shoot_vol_mean_sqrt))$mean_fitness_fl)
y3_FFD<-subset(datadef,!is.na(FFD)&
                           !is.na(cn_shoot_vol_mean_sqrt))$FFD
# vectors of outcome values
yrep3_fitness_pois<-posterior_predict(bivar2.all.brm.pois,
                                      draws = 500,resp="roundmeanfitnessfl")
yrep3_FFD_pois<-posterior_predict(bivar2.all.brm.pois,
                                    draws = 500,resp="FFD")
yrep3_fitness_nb<-posterior_predict(bivar2.all.brm.nb,
                                     draws = 500,resp="roundmeanfitnessfl")
yrep3_FFD_nb<-posterior_predict(bivar2.all.brm.nb,
                                  draws = 500,resp="FFD")
# matrices of draws from the posterior predictive distribution
```

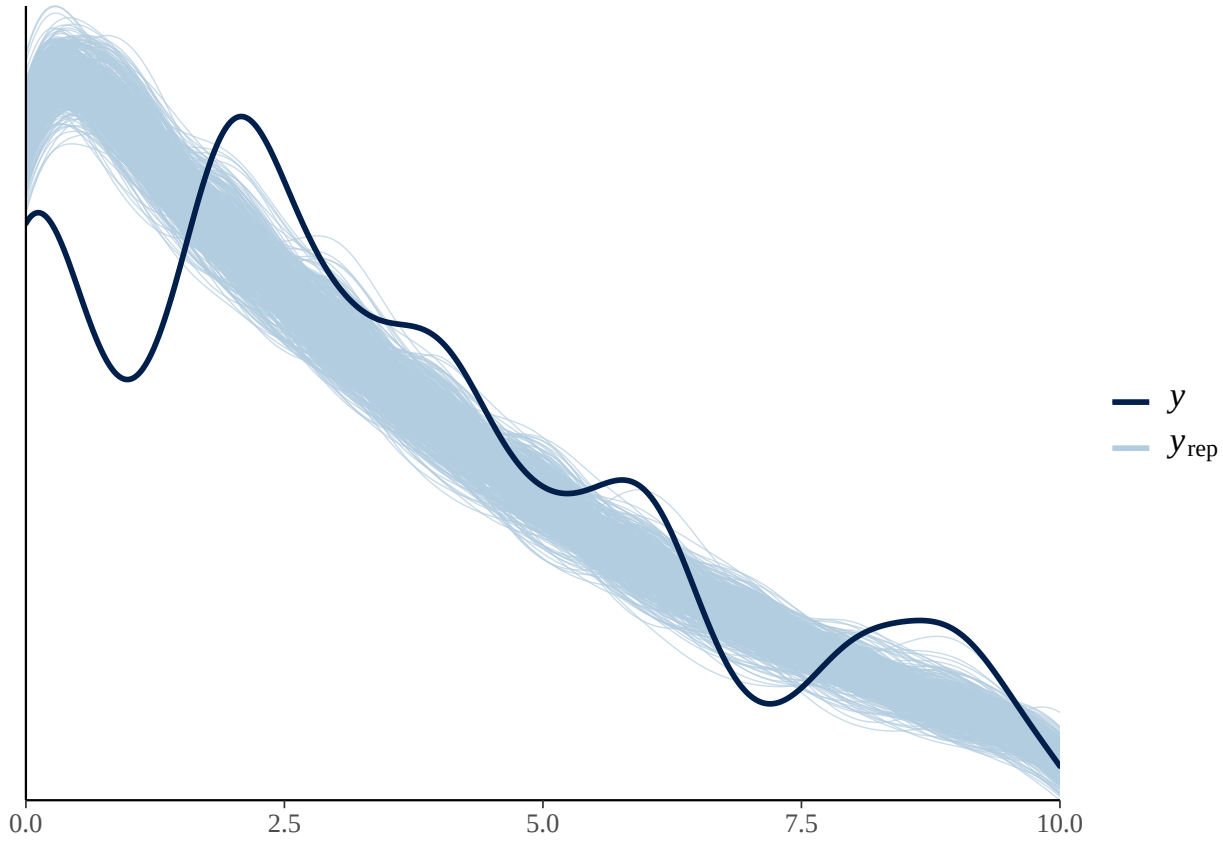
Each row of the matrix is a draw from the posterior predictive distribution, i.e. a vector with one element for each of the data points in y.

Comparison of the distribution of y and the distributions of the simulated datasets in the yrep matrix

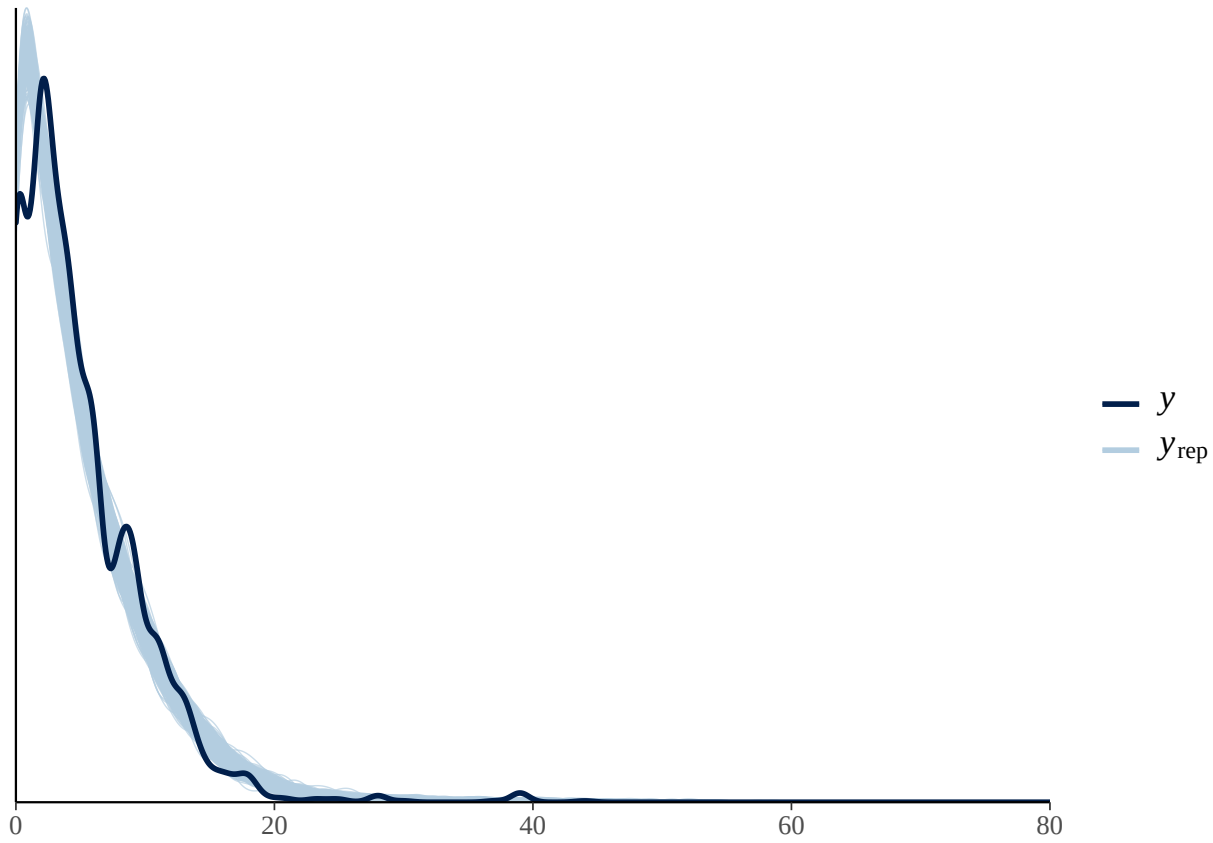
```
ppc_dens_overlay(y3_fitness, yrep3_fitness_pois[1:500,])
```



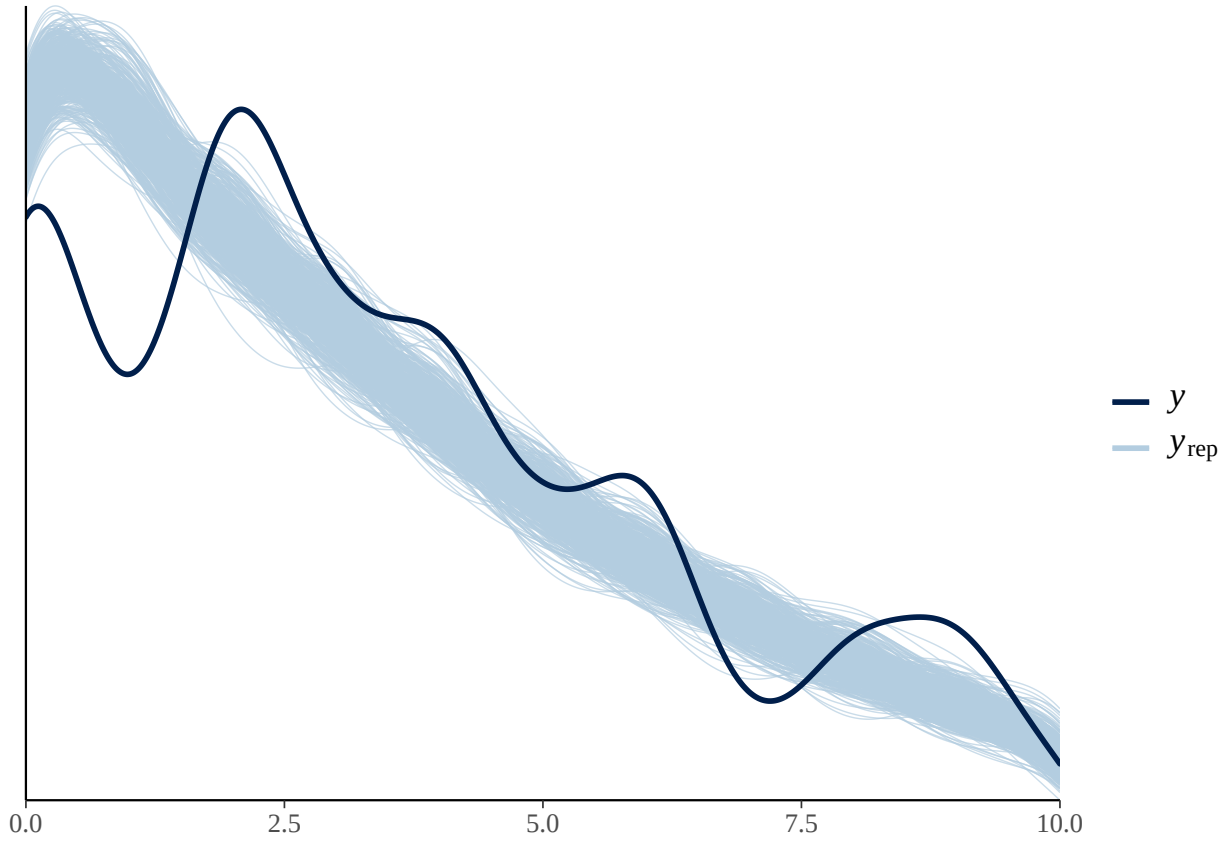

```
ppc_dens_overlay(y3_fitness, yrep3_fitness_pois[1:500,])+xlim(0, 10)
```



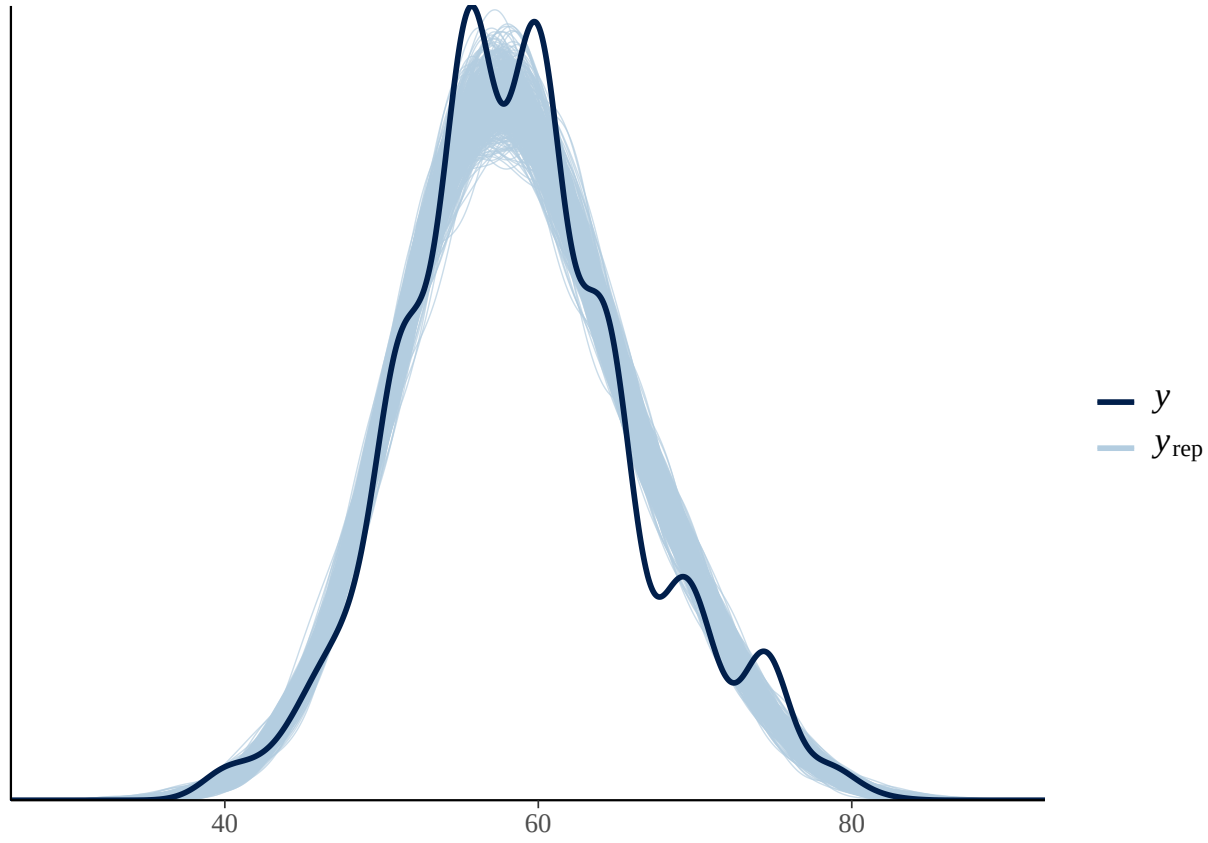

```
ppc_dens_overlay(y3_fitness, yrep3_fitness_nb[1:500,])
```



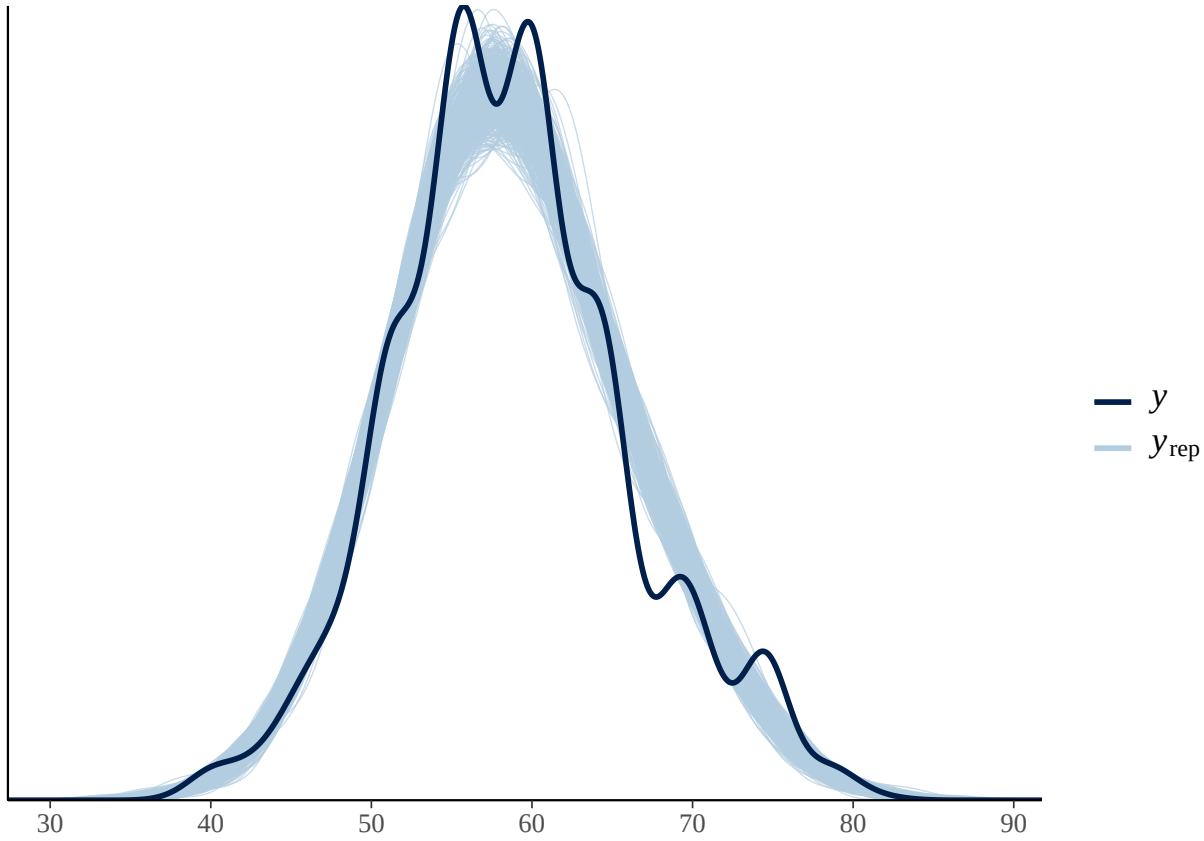

```
ppc_dens_overlay(y3_fitness, yrep3_fitness_nb[1:500,])+xlim(0,10)
```




```
ppc_dens_overlay(y3_FFD, yrep3_FFD_pois[1:500,])
```



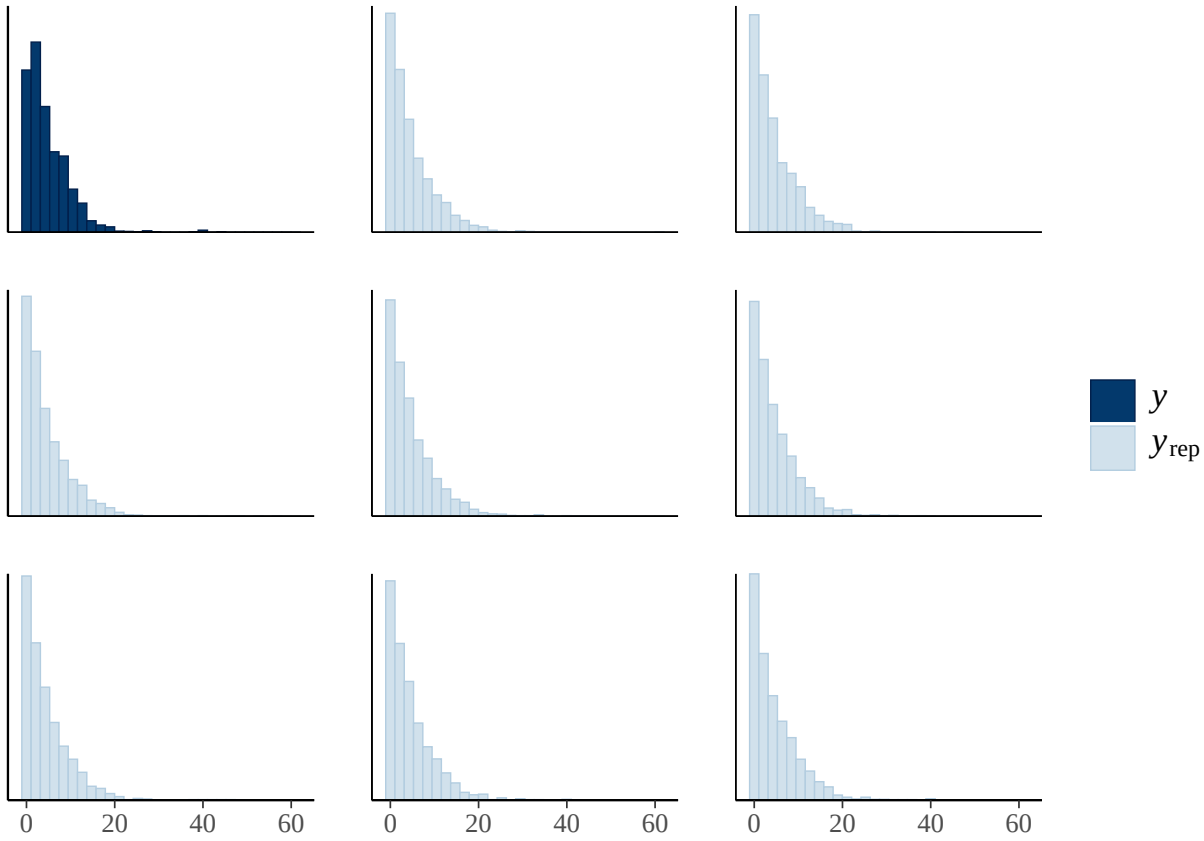

```
ppc_dens_overlay(y3_FFD, yrep3_FFD_nb[1:500,])
```



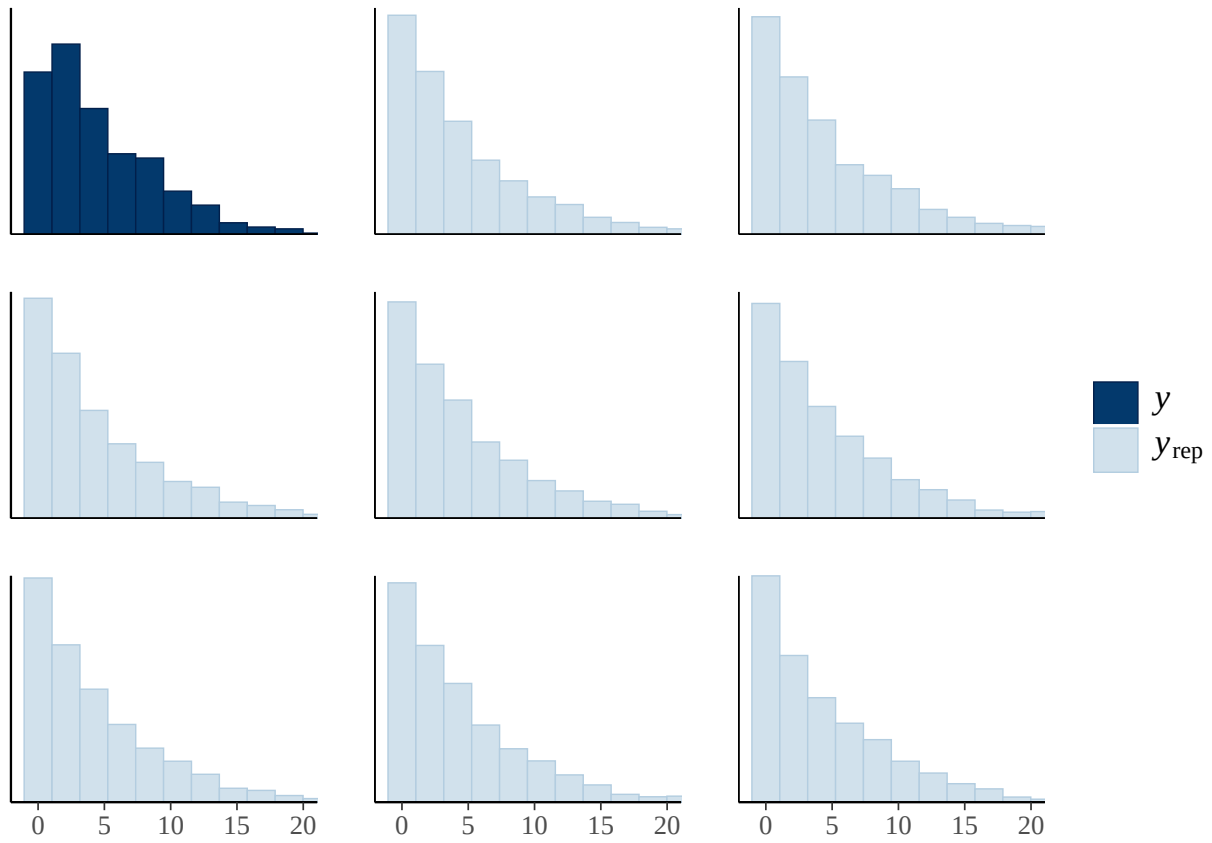
Poisson and negative binomial look pretty similar.

Separate histograms of y and some of the yrep datasets

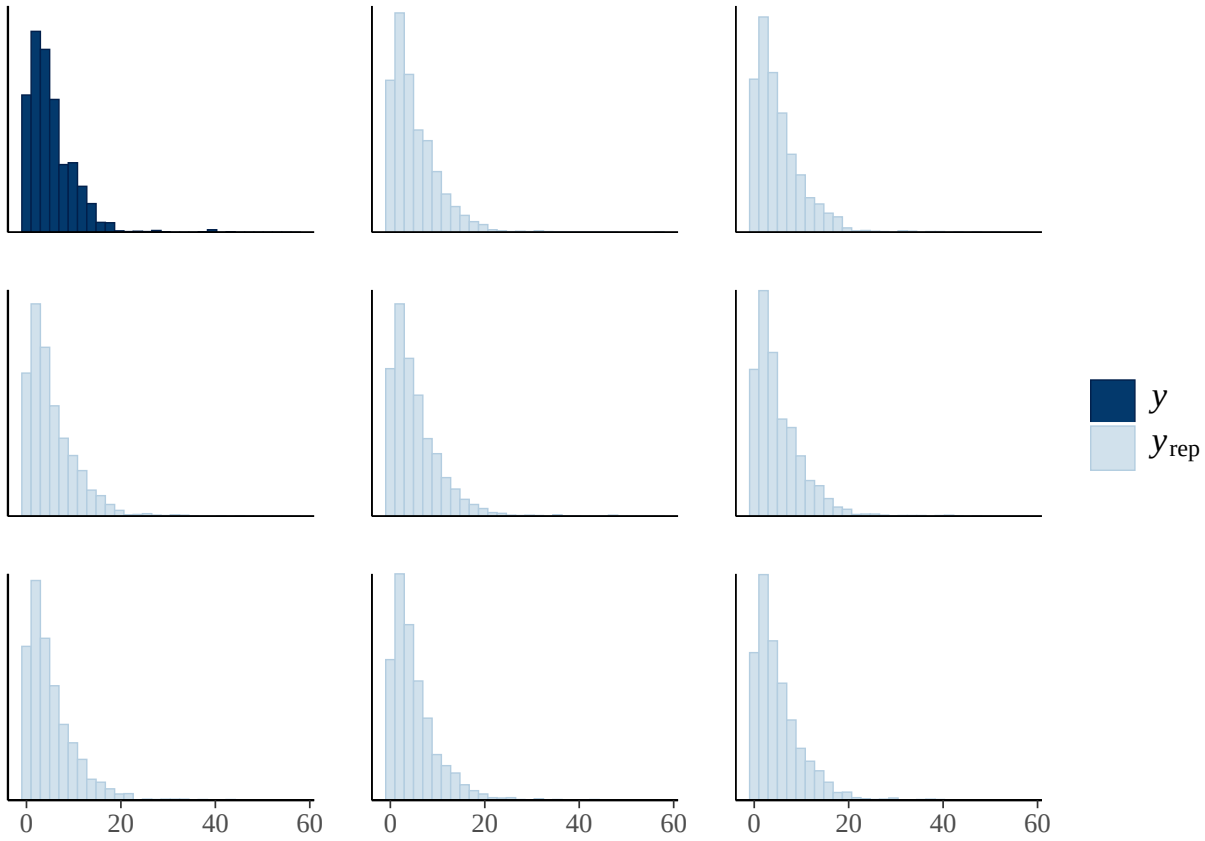
```
ppc_hist(y3_fitness, yrep3_fitness_pois[1:8, ])
```



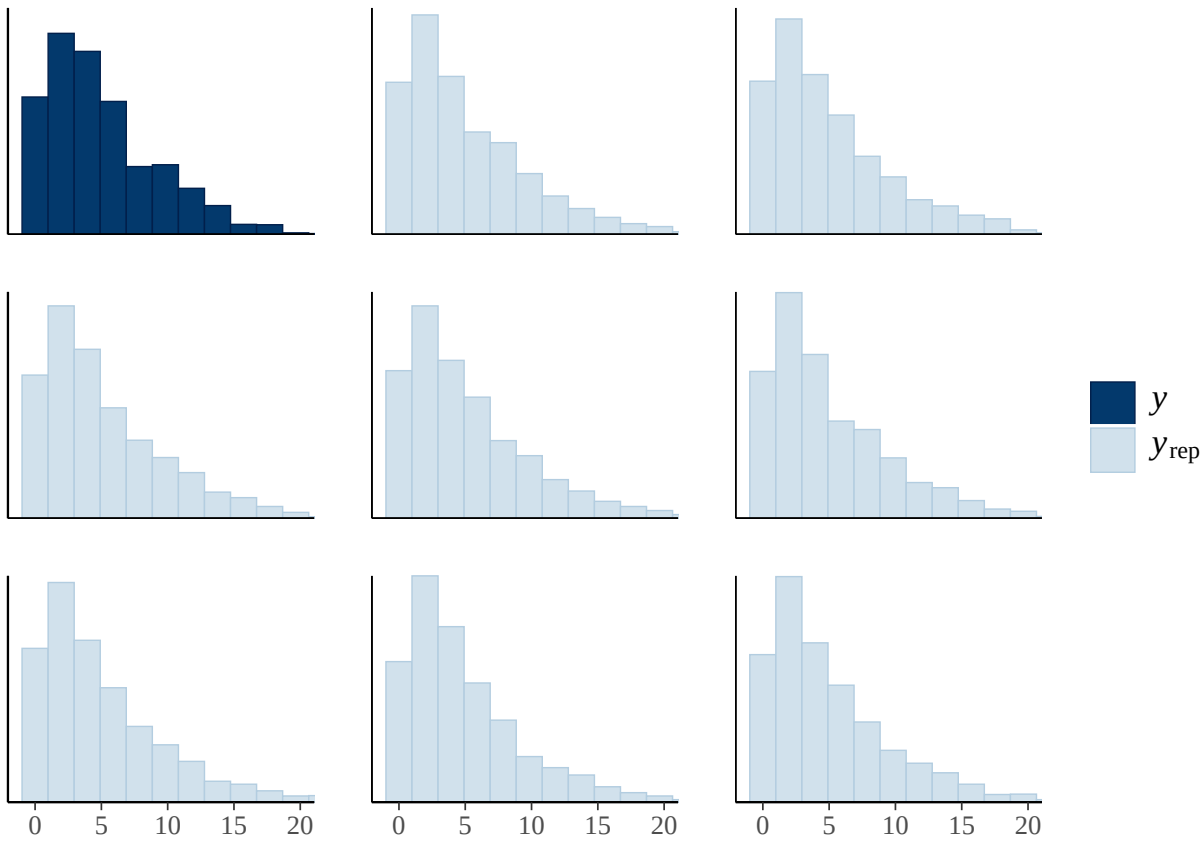

```
ppc_hist(y3_fitness, yrep3_fitness_pois[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```



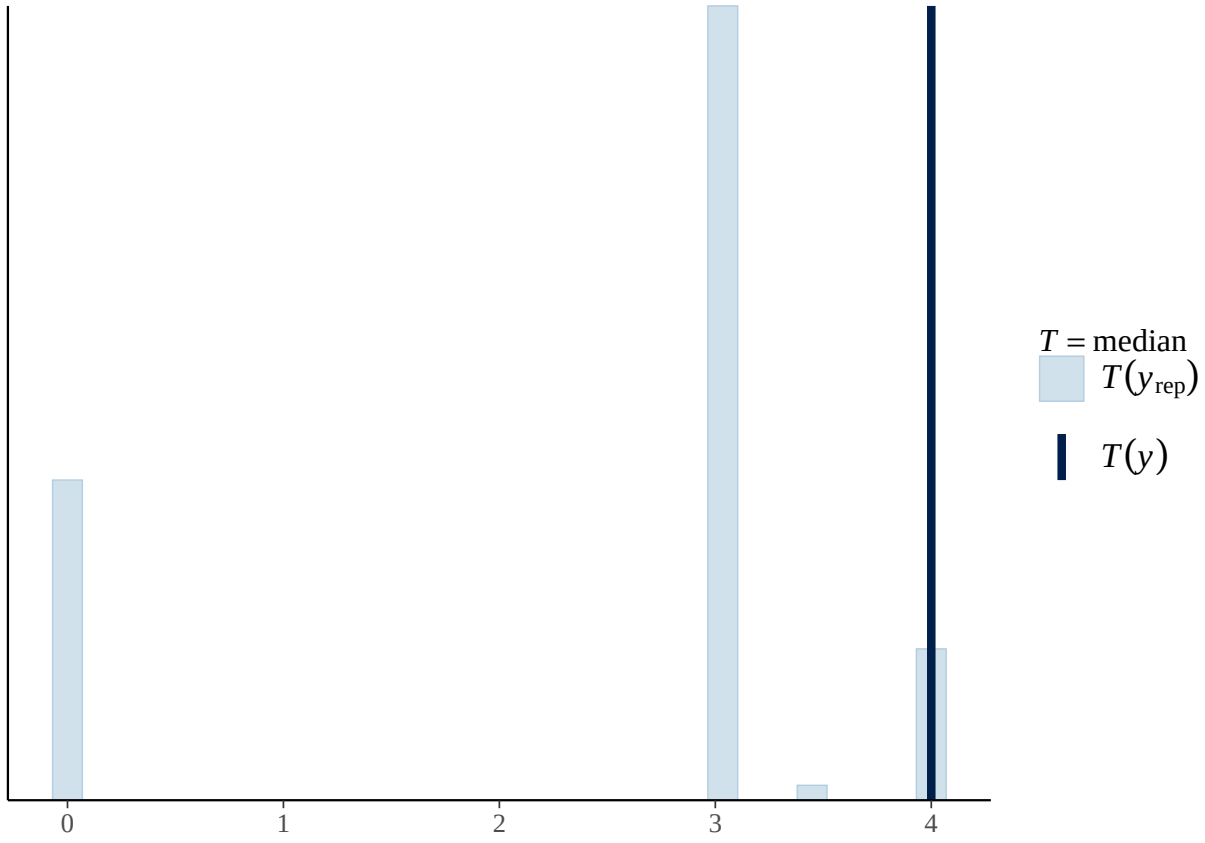

```
ppc_hist(y3_fitness, yrep3_fitness_nb[1:8, ])
```



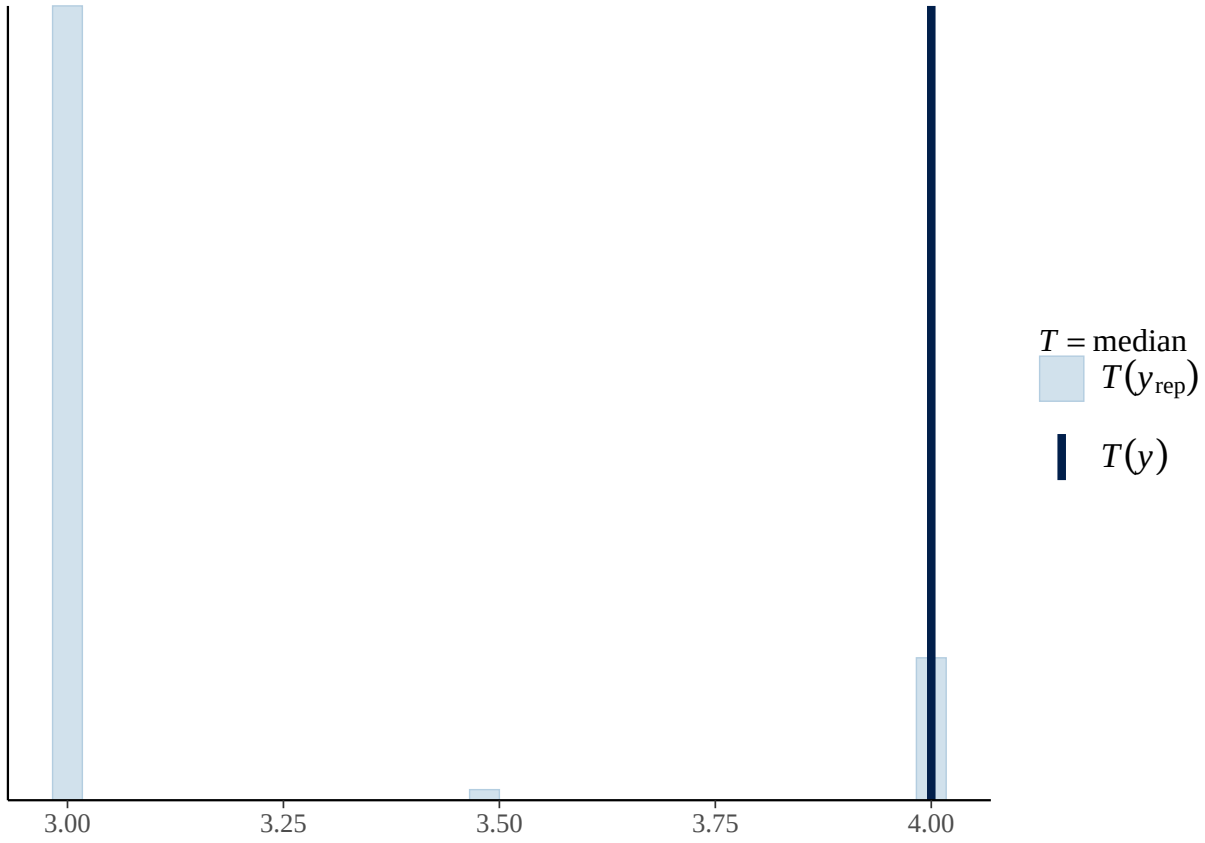

```
ppc_hist(y3_fitness, yrep3_fitness_nb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```




```
ppc_stat(y3_fitness, yrep3_fitness_pois,stat="median")
```

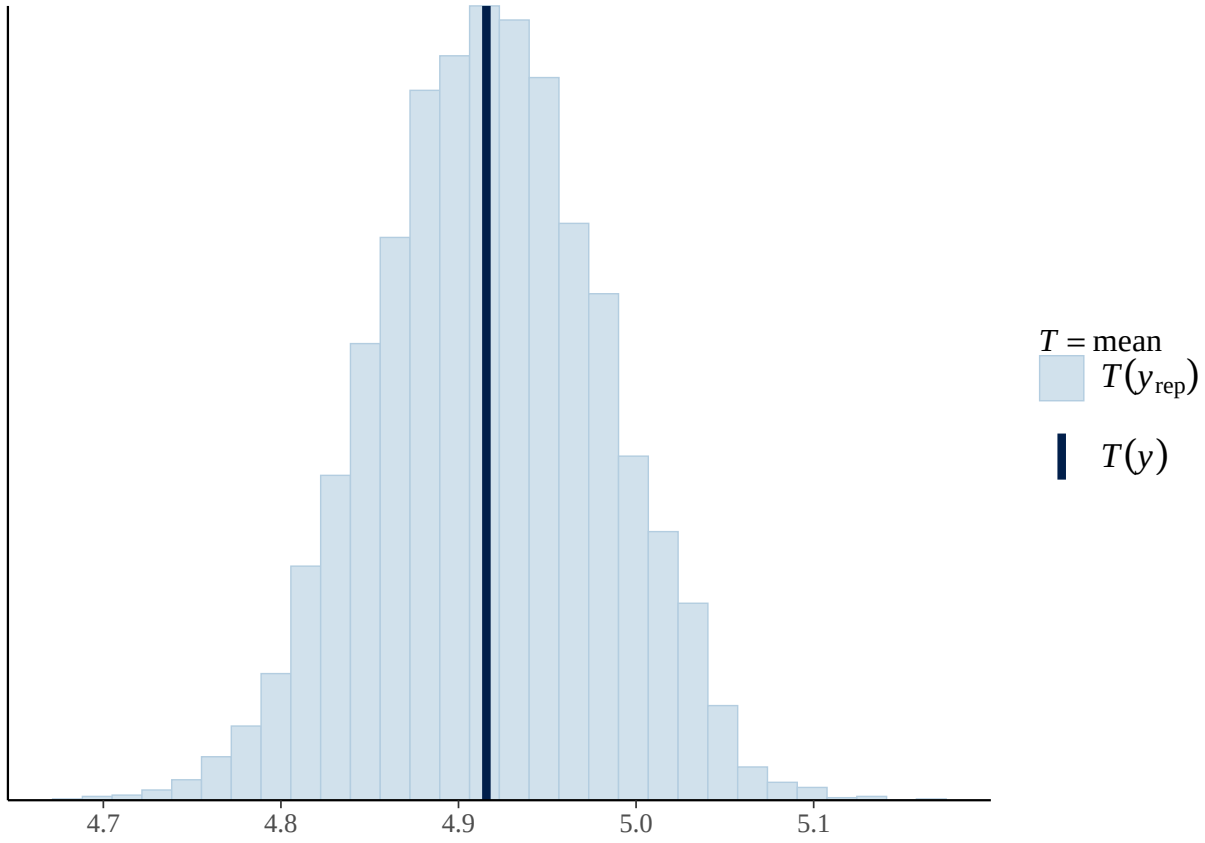



```
ppc_stat(y3_fitness, yrep3_fitness_nb,stat="median")
```



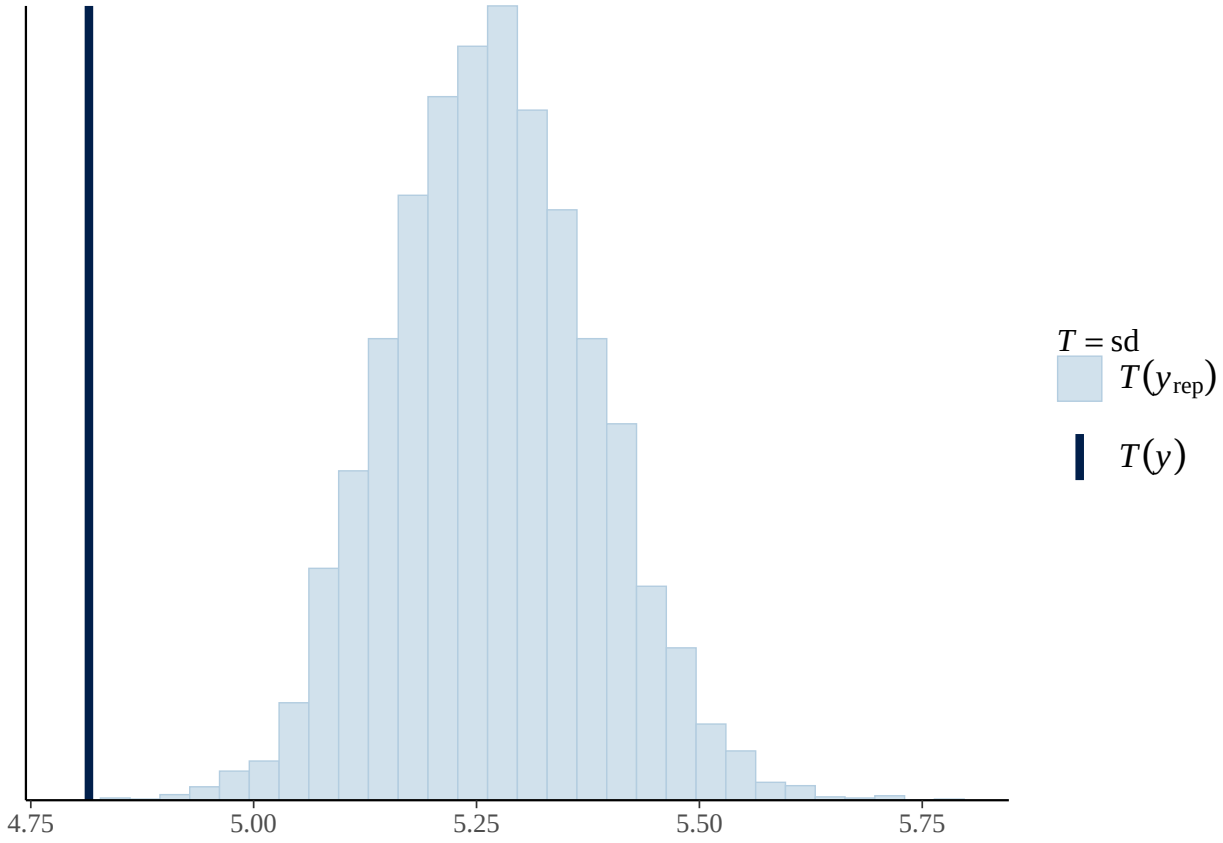

```
ppc_stat(y3_fitness, yrep3_fitness_pois,stat="mean")
```


```
ppc_stat(y3_fitness, yrep3_fitness_nb,stat="mean")
```

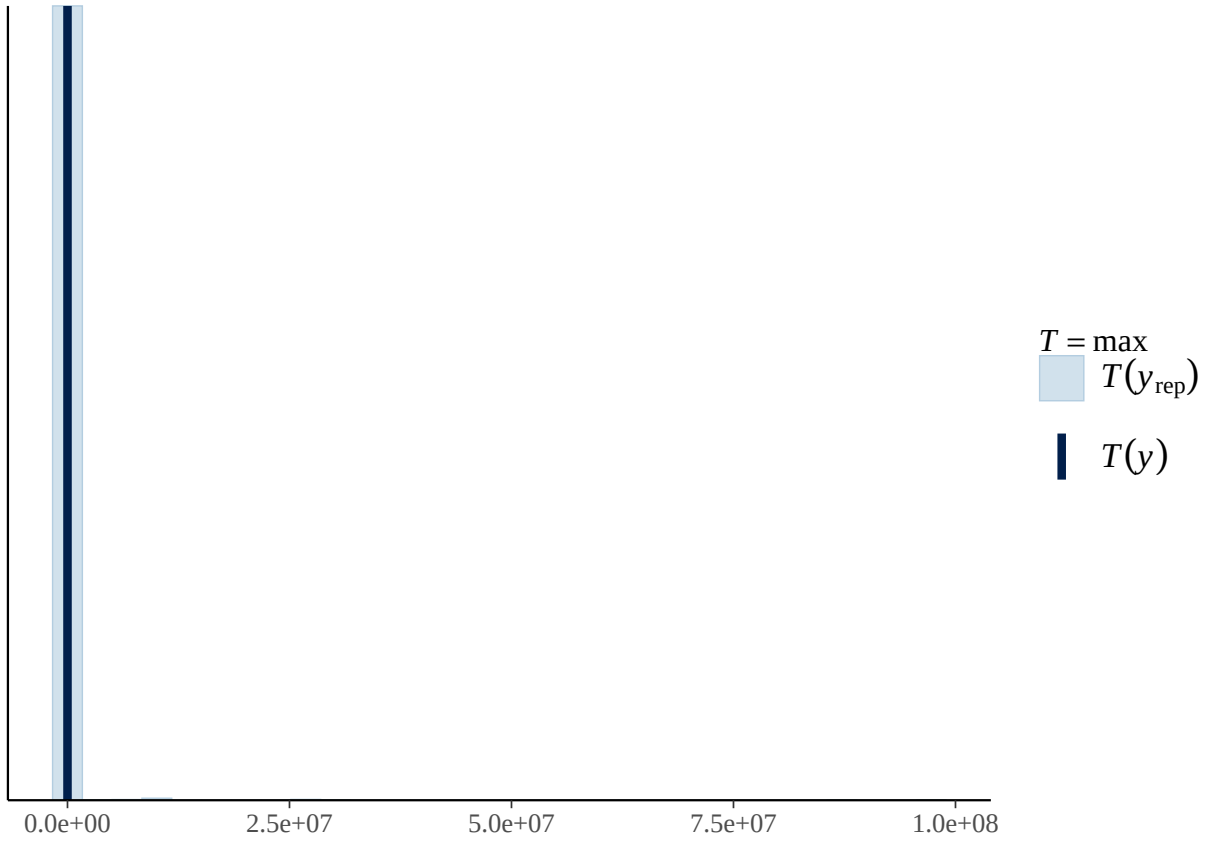



```
ppc_stat(y3_fitness, yrep3_fitness_pois,stat="sd")
```

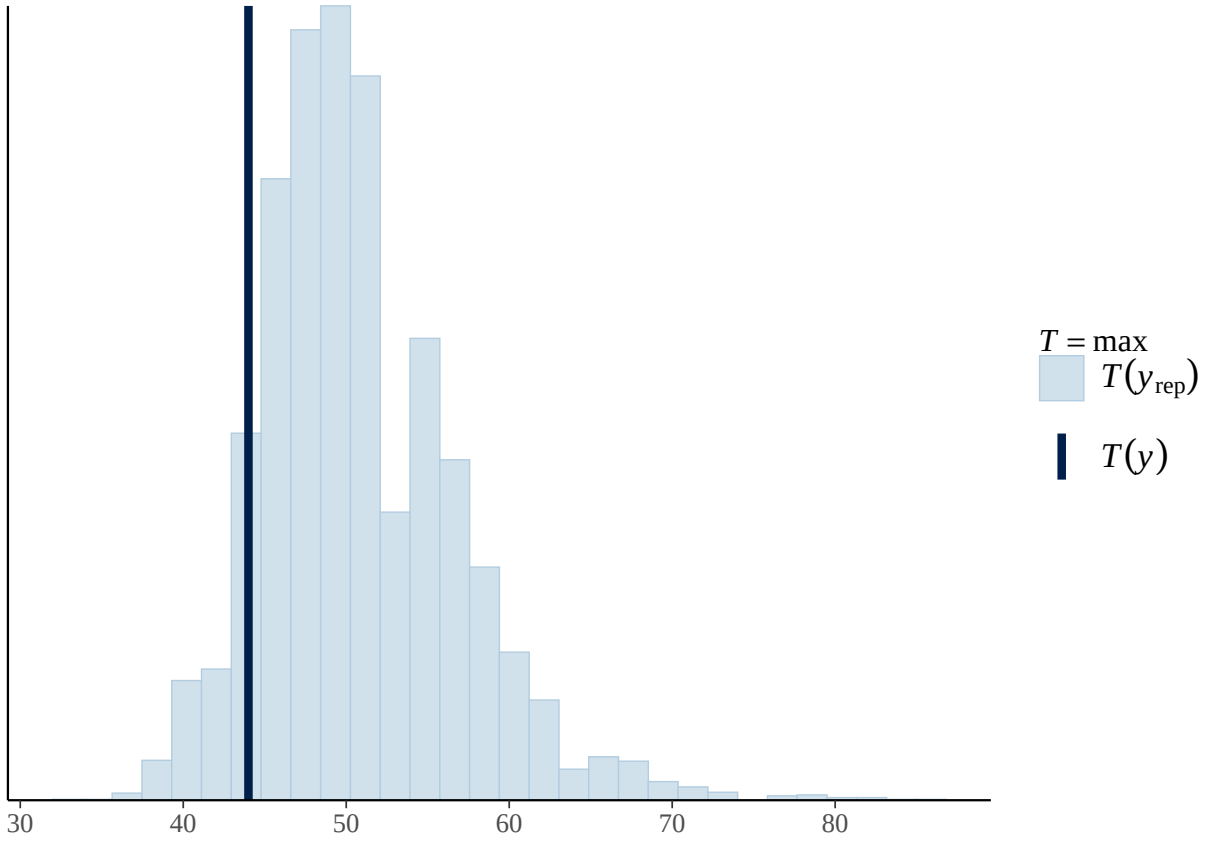

```
ppc_stat(y3_fitness, yrep3_fitness_nb,stat="sd")
```




```
ppc_stat(y3_fitness, yrep3_fitness_pois,stat="max")
```

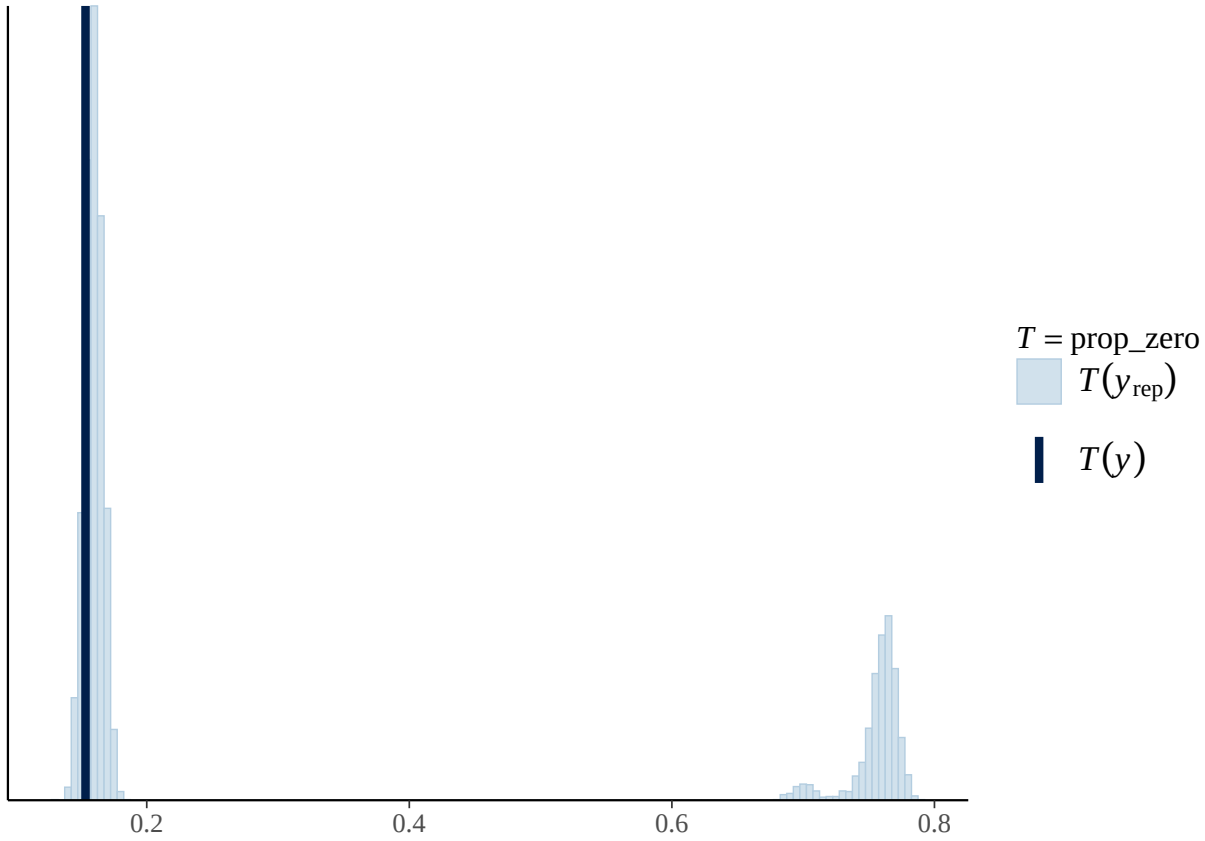



```
ppc_stat(y3_fitness, yrep3_fitness_nb,stat="max")
```

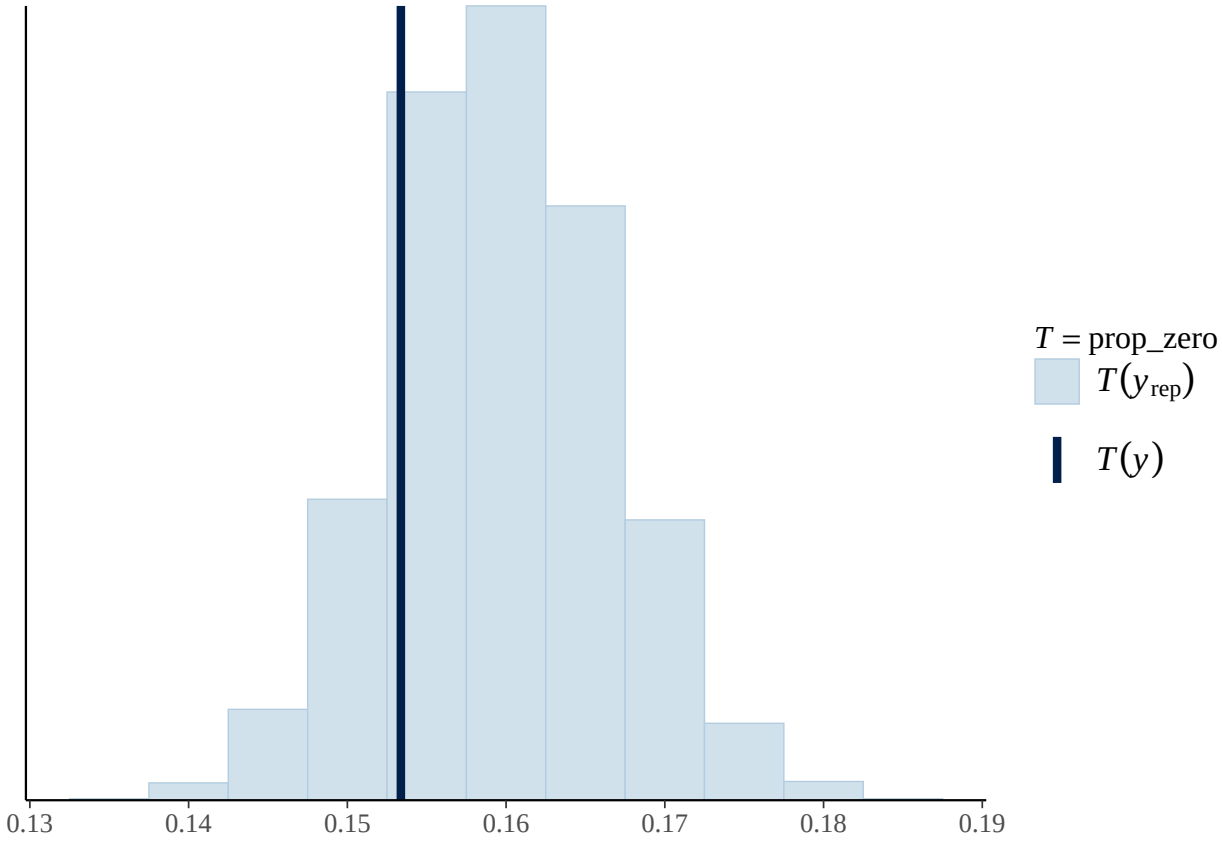


Look at the distribution of the proportion of zeros over the replicated datasets from the posterior predictive distribution in yrep and compare to the proportion of observed zeros in y.

```
ppc_stat(y3_fitness, yrep3_fitness_pois, stat = "prop_zero", binwidth = 0.005)
```




```
ppc_stat(y3_fitness, yrep3_fitness_nb, stat = "prop_zero", binwidth = 0.005)
```



They look quite similar.

Measure of fit: Bayesian R2

```
bayes_R2(bivar2.all.brm.pois)
```

```
##                         Estimate Est.Error      Q2.5     Q97.5
## R2FFD                  0.5660637 0.1277587 0.3469402 0.6645711
## R2roundmeanfitnessfl 0.8268181 0.1970997 0.4547354 0.9481001
```

```
bayes_R2(bivar2.all.brm.nb)
```

```
##                         Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                  0.6485588 0.009432702 0.6294919 0.6667641
## R2roundmeanfitnessfl 0.9394116 0.004604999 0.9298369 0.9477395
```

Very similar.

Leave-one-out cross validation (LOO): (not run)

Extract selection coefficients

```
# Extract posterior samples
bivar2.all.brm.pois_post <- posterior_samples(bivar2.all.brm.pois)
bivar2.all.brm.pois_post <- as.mcmc(bivar2.all.brm.pois_post)
#head(bivar2.all.brm.pois_post)[,1:20]

# [,5] sd_id__FFD_Intercept
# [,6] sd_id__FFD_cmean_4
# [,7] sd_id__roundmeanfitnessfl_Intercept
# [,9] cor_id__FFD_Intercept__FFD_cmean_4
# [,10] cor_id__FFD_Intercept__roundmeanfitnessfl_Intercept
# [,11] cor_id__FFD_cmean_4__roundmeanfitnessfl_Intercept

sample.gmat2 <- function(data, replicates = 5000) {

  ##Initialize the results list (list of lists)
  foo <- list(gmat = matrix(rep(0,3*3), ncol = 3))
  results.list <- list()
  for(j in 1:replicates) { results.list[[j]] <- foo }

  for(i in 1:replicates) {
    diag(results.list[[i]]$gmat) <- data[i,5:7]^2 #Get the diagonal

    #Upper diagonal
    results.list[[i]]$gmat[1,2] <- data[i,5]*data[i,6]*data[i,9]
    results.list[[i]]$gmat[1,3] <- data[i,5]*data[i,7]*data[i,10]
    results.list[[i]]$gmat[2,3] <- data[i,6]*data[i,7]*data[i,11]

    #Lower diagonal
    results.list[[i]]$gmat[2,1] <- results.list[[i]]$gmat[1,2]
    results.list[[i]]$gmat[3,1] <- results.list[[i]]$gmat[1,3]
    results.list[[i]]$gmat[3,2] <- results.list[[i]]$gmat[2,3]

  }

  return(results.list)
}

sampled.gmat3 <- sample.gmat2(bivar2.all.brm.pois_post, replicates = 1000)

sgmat3 <- lapply(sampled.gmat3, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat3 <- unname(sapply(sgmat3, '[[', 1)) #Change to matrix

sgmat3 <- t(sgmat3)

P.modelBV_RR3 <- sgmat3
P.modelBV_RR3.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR3.mode[k] <- posterior.mode(mcmc(sgmat3[,k]))

# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR3 <- sgmat3[,c(3,6)]
colnames(S.modelBV_RR3) <- c("S_intercepts", "S_slopes")
S.modelBV_RR3.mode <- P.modelBV_RR3.mode[1:2, 3]
```

```
posterior.mode(mcmc(S.modelBV_RR3))
```

Poisson model

```
## S_intercepts     S_slopes
## 0.0005858343 0.2889593685
```

```
HPDinterval(mcmc(S.modelBV_RR3))
```

```
##                      lower      upper
## S_intercepts 3.046961e-06 0.06399234
## S_slopes     7.805280e-03 0.54356905
## attr(,"Probability")
## [1] 0.95
```

```
# Estimate selection gradients for intercept and slope (beta = S / P)
# on each sample of posterior and extract their mode
beta_post_RR3 <- matrix(NA, nrow(S.modelBV_RR3) ,2)

for (i in 1:nrow(S.modelBV_RR3)) {
  P3_3 <- matrix(rep(NA, 9), nrow = 3)
  # 3x3 matrix of var-cov for individual X.int, X.slope and fitness
  for (k in 1:9) {P3_3[k] <- P.modelBV_RR3[i, k] }
  P2_3 <- P3_3[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S3 <- P3_3[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR3[i,] <- solve(P2_3) %*% S3   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR3) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR3))
```

```
## beta_intercepts     beta_slopes
##      -6.9659208       0.2109236
```

```
HPDinterval(mcmc(beta_post_RR3))
```

```
##                       lower       upper
## beta_intercepts  1.1728703 1367.060383
## beta_slopes     -0.1001685    1.847697
## attr(,"Probability")
## [1] 0.95
```

```
# Extract posterior samples
bivar2.all.brm.nb_post <- posterior_samples(bivar2.all.brm.nb)
bivar2.all.brm.nb_post <- as.mcmc(bivar2.all.brm.nb_post)
#head(bivar2.all.brm.nb_post)[,1:20]
```

```
# [,5] sd_id__FFD_Intercept
# [,6] sd_id__FFD_cmean_4
# [,7] sd_id__roundmeanfitnessfl_Intercept
# [,9] cor_id__FFD_Intercept__FFD_cmean_4
# [,10] cor_id__FFD_Intercept__roundmeanfitnessfl_Intercept
# [,11] cor_id__FFD_cmean_4__roundmeanfitnessfl_Intercept

sampled.gmat4 <- sample.gmat2(bivar2.all.brm.nb_post, replicates = 1000)

sgmat4 <- lapply(sampled.gmat4, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat4 <- unname(sapply(sgmat4, '[[', 1)) #Change to matrix

sgmat4 <- t(sgmat4)

P.modelBV_RR4 <- sgmat4
P.modelBV_RR4.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR4.mode[k] <- posterior.mode(mcmc(sgmat4[,k]))

# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR4 <- sgmat4[,c(3,6)]
colnames(S.modelBV_RR4) <- c("S_intercepts", "S_slopes")
S.modelBV_RR4.mode <- P.modelBV_RR4.mode[1:2, 3]

posterior.mode(mcmc(S.modelBV_RR4))
```

Negative binomial model

```
## S_intercepts     S_slopes
##   -0.8684709    0.1833387
```

```
HPDinterval(mcmc(S.modelBV_RR4))
```

```
##                    lower      upper
## S_intercepts -1.24638336 -0.5516807
## S_slopes     -0.06561457  0.4313555
## attr(,"Probability")
## [1] 0.95
```

```
# Estimate selection gradients for intercept and slope (beta = S / P)
# on each sample of posterior and extract their mode
beta_post_RR4 <- matrix(NA, nrow(S.modelBV_RR4) ,2)

for (i in 1:nrow(S.modelBV_RR4)) {
  P3_4 <- matrix(rep(NA, 9), nrow = 3)
  # 3x3 matrix of var-cov for individual X.int, X.slope and fitness
  for (k in 1:9) {P3_4[k] <- P.modelBV_RR4[i, k] }
  P2_4 <- P3_4[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S4 <- P3_4[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR4[i,] <- solve(P2_4) %*% S4   # selection gradients beta = P^-1 * S
}
```

```r
colnames(beta_post_RR4) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR4))
```

```
## beta_intercepts     beta_slopes
##      -0.6899606       1.2256471
```

```r
HPDinterval(mcmc(beta_post_RR4))
```

```
##                       lower      upper
## beta_intercepts -1.2599271 -0.3966808
## beta_slopes      0.5623511  2.8390875
## attr(,"Probability")
## [1] 0.95
```

3. mean_fitness_study, no condition variable

```r
bf_fitness_study <- bf(round(mean_fitness_study) ~  (1|ID1|id))
# Set up model formula
```

Poisson distribution

```r
bivar3.all.brm.pois<-brm(bf_FFD+bf_fitness_study,
                         family = c(gaussian,poisson),
                         data = datadef,warmup = 1000,iter = 4000,chains=4,thin=2,
                         inits = "random",seed = 12345,cores = my.cores,
                         control = list(adapt_delta = 0.99))
```

```r
summary(bivar3.all.brm.pois)
```

```
##  Family: MV(gaussian, poisson)
##   Links: mu = identity; sigma = identity
##          mu = log
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_study) ~ (1 | ID1 | id)
##    Data: datadef (Number of observations: 2478)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
##
## Group-Level Effects:
## ~id (Number of levels: 837)
##                                                 Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                   1.62      0.15     1.32
## sd(FFD_cmean_4)                                     0.78      0.13     0.53
## sd(roundmeanfitnessstudy_Intercept)                 1.43      0.06     1.33
## cor(FFD_Intercept,FFD_cmean_4)                      0.74      0.13     0.46
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)  -0.57      0.07    -0.71
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)    -0.18      0.13    -0.43
```

```
##                                                    u-95% CI Rhat Bulk_ESS
## sd(FFD_Intercept)                                      1.90 1.00     1566
## sd(FFD_cmean_4)                                        1.05 1.00     2456
## sd(roundmeanfitnessstudy_Intercept)                    1.55 1.00     3226
## cor(FFD_Intercept,FFD_cmean_4)                         0.94 1.01      491
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)    -0.42 1.00      683
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)       0.07 1.00     1089
##                                                    Tail_ESS
## sd(FFD_Intercept)                                      2863
## sd(FFD_cmean_4)                                        4469
## sd(roundmeanfitnessstudy_Intercept)                    4639
## cor(FFD_Intercept,FFD_cmean_4)                         2307
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)     1399
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)       1879
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.11      0.86     3.76     7.07 1.00     3020     4476
##
## Population-Level Effects:
##                                 Estimate Est.Error l-95% CI u-95% CI Rhat
## FFD_Intercept                      58.81      1.07    56.70    60.93 1.00
## roundmeanfitnessstudy_Intercept     0.13      0.06     0.01     0.25 1.00
## FFD_cmean_4                        -2.34      0.83    -3.98    -0.74 1.00
##                                 Bulk_ESS Tail_ESS
## FFD_Intercept                       2146     3628
## roundmeanfitnessstudy_Intercept     4787     4695
## FFD_cmean_4                         2585     3743
##
## Family Specific Parameters:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma_FFD     4.36      0.07     4.22     4.51 1.00     2330     4358
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Negative binomial distribution

```
bivar3.all.brm.nb<-brm(bf_FFD+bf_fitness_study,
                       family = c(gaussian,negbinomial),
                       data = datadef,warmup = 1000,iter = 4000,chains=4,thin=2,
                       inits = "random",seed = 12345,cores = my.cores,
                       control = list(adapt_delta = 0.99, max_treedepth = 15))
```

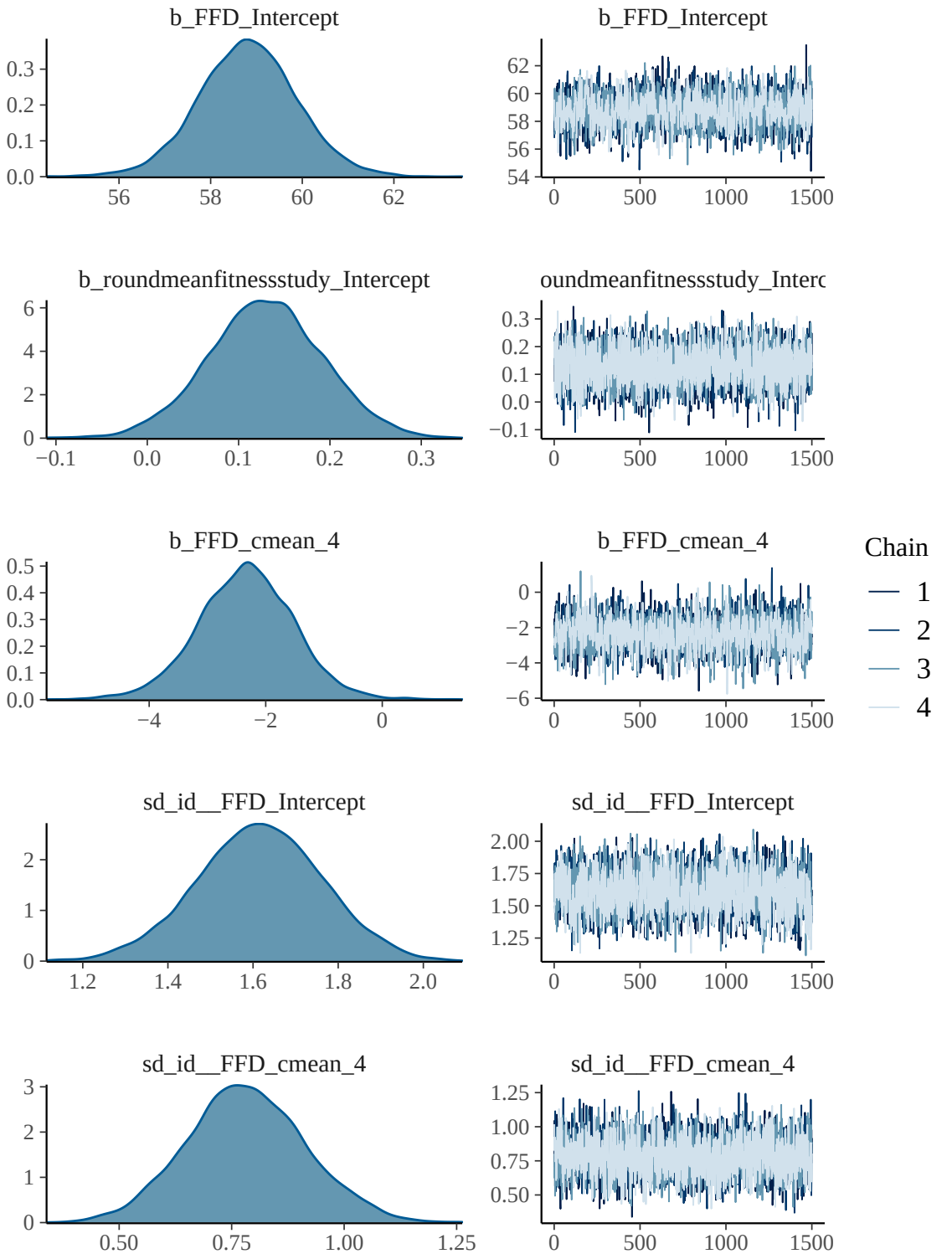
```
summary(bivar3.all.brm.nb)
```

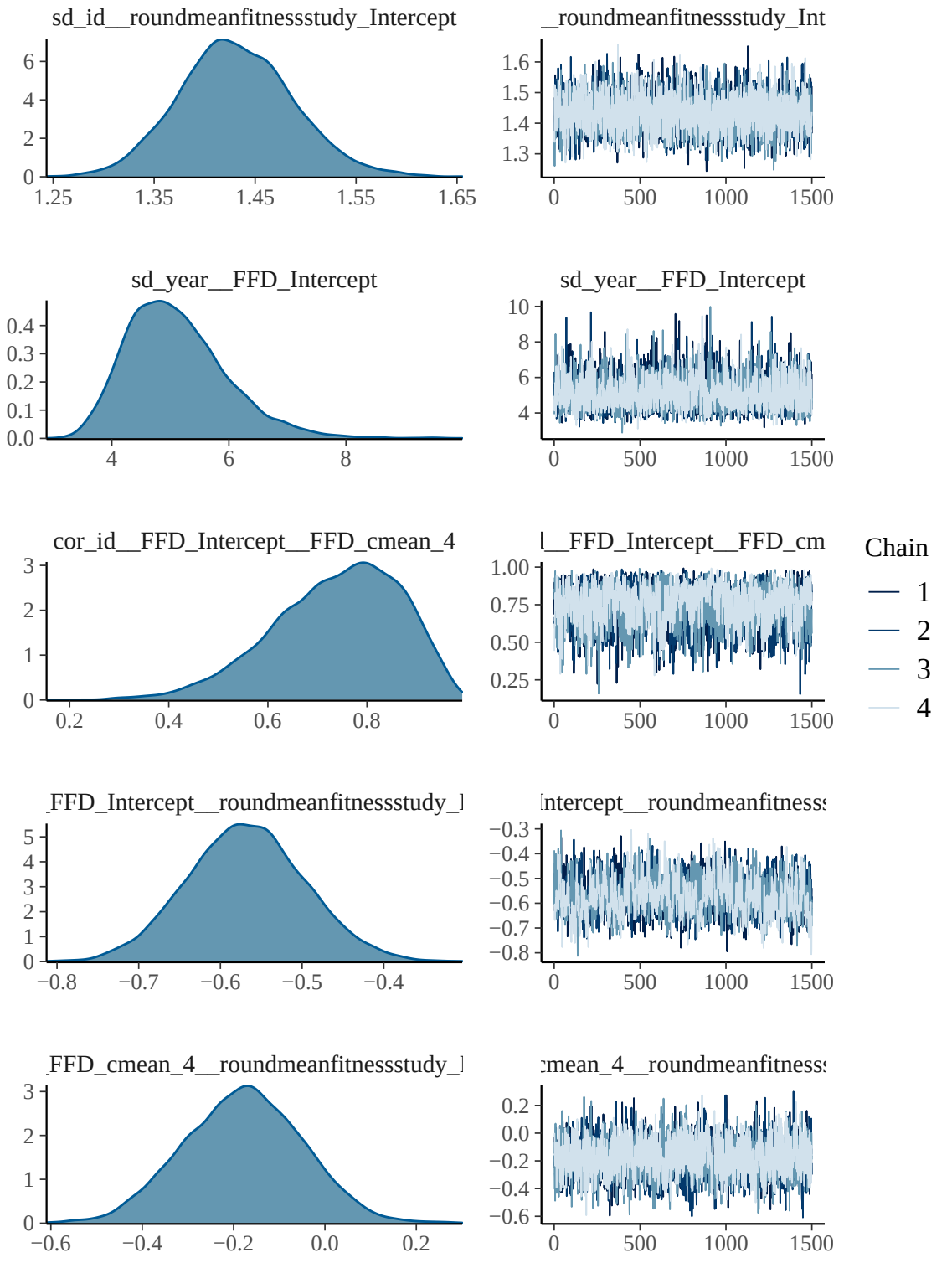
```
##  Family: MV(gaussian, negbinomial)
##   Links: mu = identity; sigma = identity
##          mu = log; shape = identity
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
```

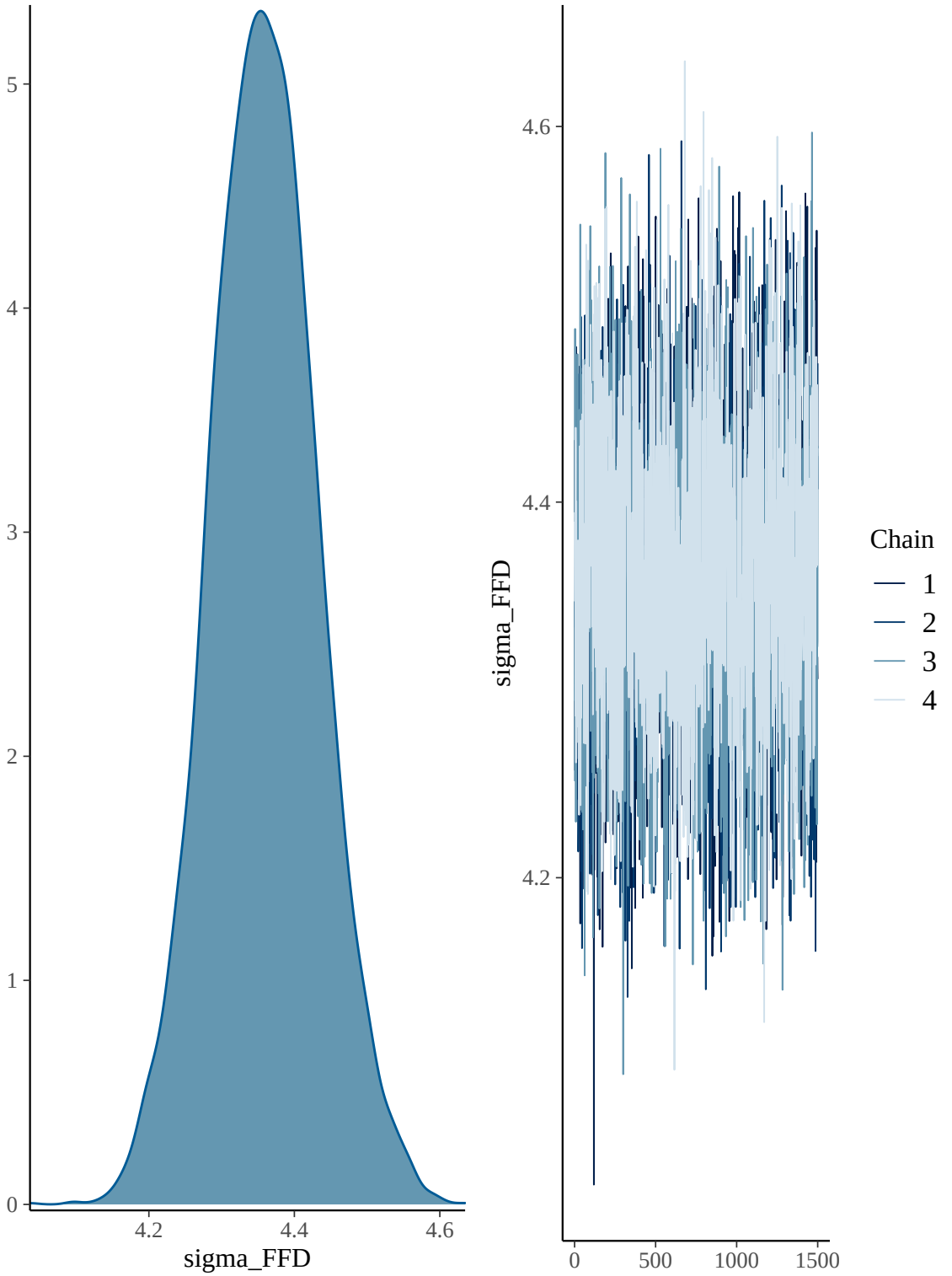
```
##          round(mean_fitness_study) ~ (1 | ID1 | id)
##    Data: datadef (Number of observations: 2478)
## Samples: 4 chains, each with iter = 6000; warmup = 1000; thin = 2;
##          total post-warmup samples = 10000
##
## Group-Level Effects:
## ~id (Number of levels: 837)
##                                                    Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                      1.61      0.14     1.33
## sd(FFD_cmean_4)                                        0.78      0.13     0.52
## sd(roundmeanfitnessstudy_Intercept)                    1.43      0.06     1.32
## cor(FFD_Intercept,FFD_cmean_4)                         0.74      0.13     0.45
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)    -0.57      0.07    -0.71
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)      -0.18      0.13    -0.45
##                                                    u-95% CI Rhat Bulk_ESS
## sd(FFD_Intercept)                                      1.89 1.00     4007
## sd(FFD_cmean_4)                                        1.05 1.00     3784
## sd(roundmeanfitnessstudy_Intercept)                    1.55 1.00     5134
## cor(FFD_Intercept,FFD_cmean_4)                         0.95 1.00      696
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)    -0.43 1.00     2117
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)       0.08 1.00     1119
##                                                    Tail_ESS
## sd(FFD_Intercept)                                      6441
## sd(FFD_cmean_4)                                        6650
## sd(roundmeanfitnessstudy_Intercept)                    7295
## cor(FFD_Intercept,FFD_cmean_4)                         3982
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)     3652
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)       1150
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.15      0.87     3.79     7.18 1.00     4758     7035
##
## Population-Level Effects:
##                                Estimate Est.Error l-95% CI u-95% CI Rhat
## FFD_Intercept                     58.78      1.13    56.59    61.04 1.00
## roundmeanfitnessstudy_Intercept    0.13      0.06     0.00     0.25 1.00
## FFD_cmean_4                       -2.30      0.84    -3.94    -0.65 1.00
##                                Bulk_ESS Tail_ESS
## FFD_Intercept                      3403     5333
## roundmeanfitnessstudy_Intercept    7454     8292
## FFD_cmean_4                        4905     6320
##
## Family Specific Parameters:
##                            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sigma_FFD                      4.36      0.07     4.22     4.50 1.00     3842
## shape_roundmeanfitnessstudy  554.28    170.50   292.28   957.22 1.00     7454
##                            Tail_ESS
## sigma_FFD                      6990
## shape_roundmeanfitnessstudy    7798
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Model evaluation: Compare models

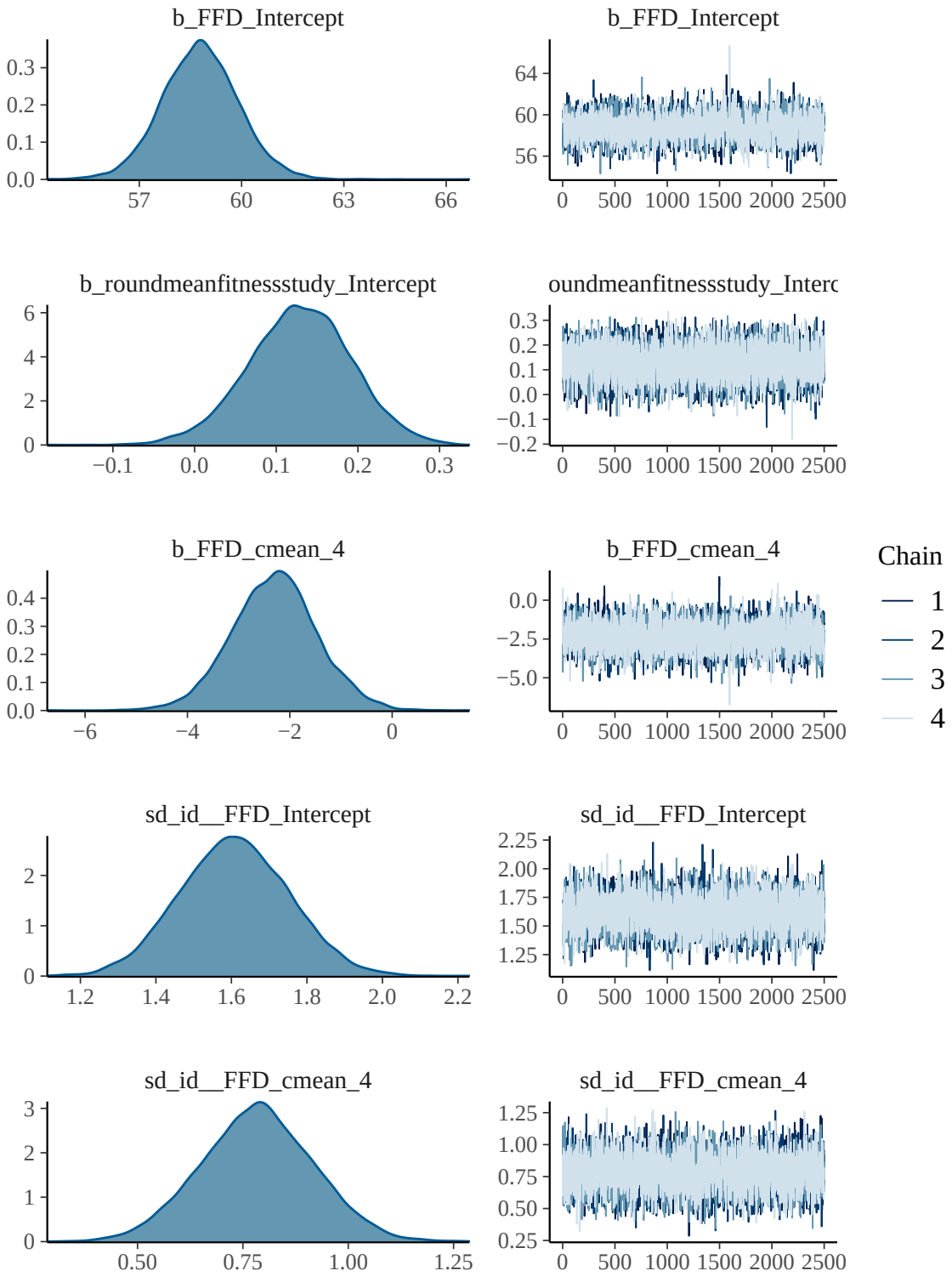
```
plot(bivar3.all.brm.pois)
```

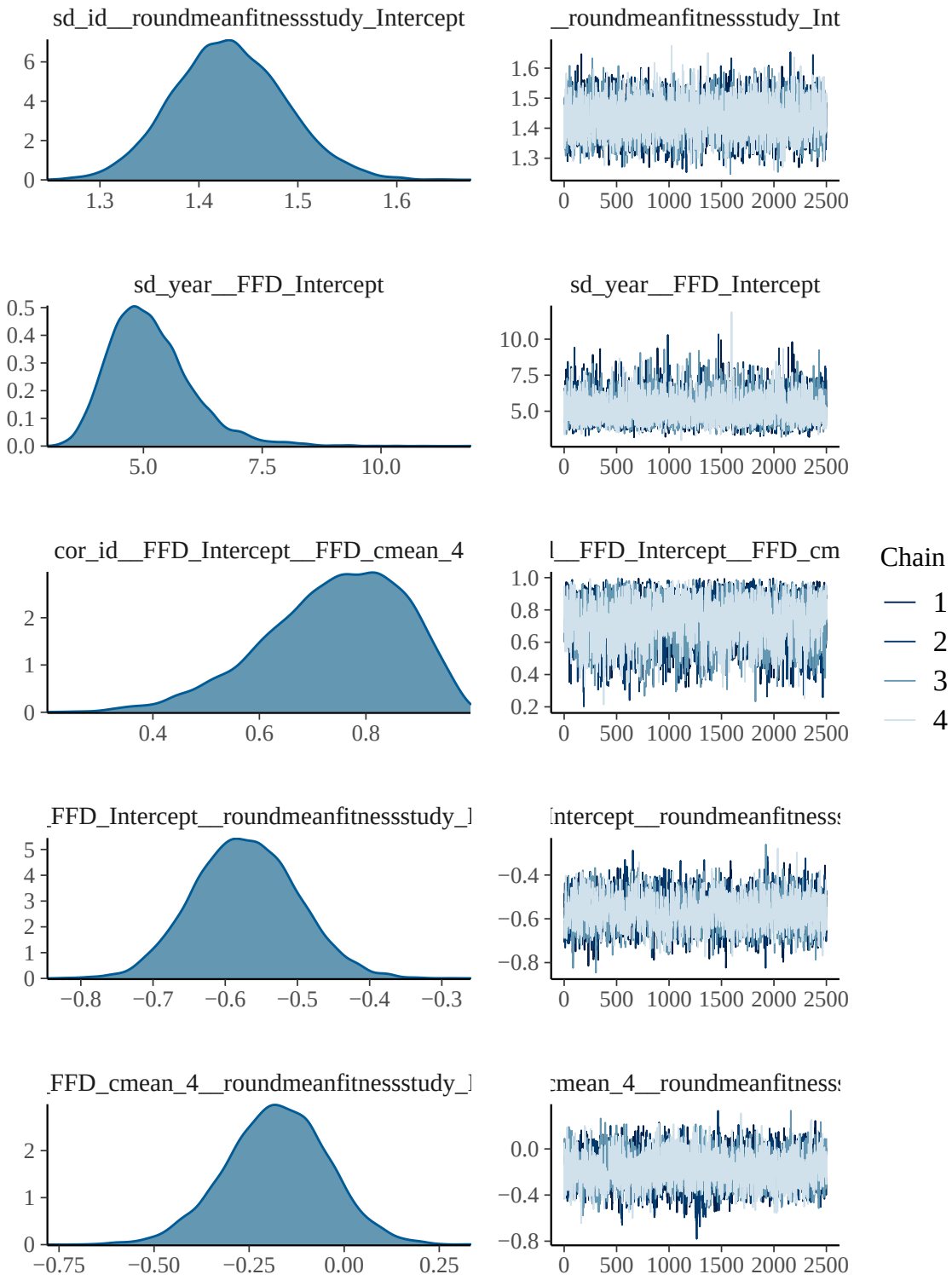


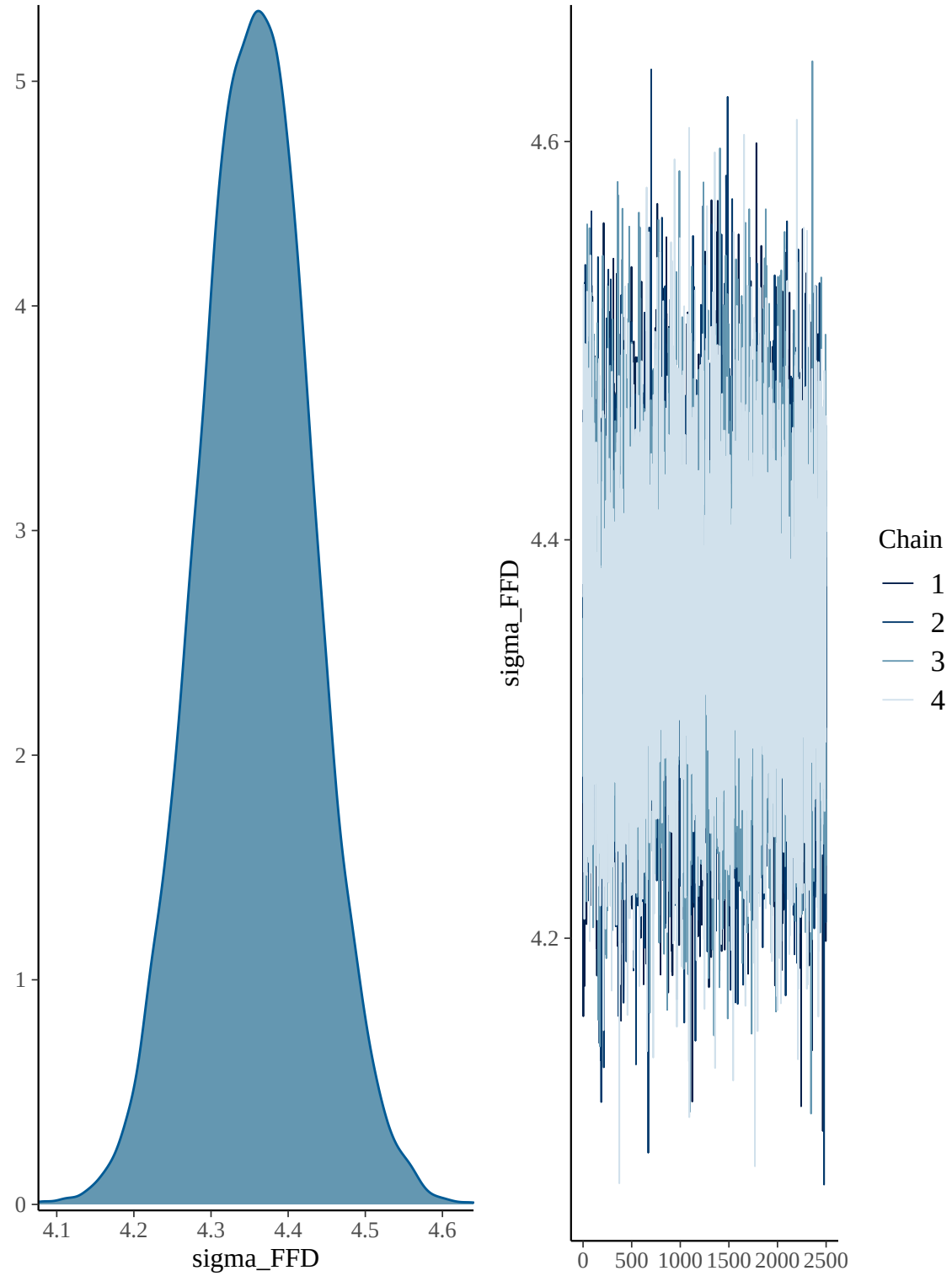

sd_id__roundmeanfitnessstudy_Intercept
__roundmeanfitnessstudy_Int
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
_FFD_Intercept__roundmeanfitnessstudy_
ntercept__roundmeanfitness
FFD_cmean_4__roundmeanfitnessstudy_
cmean_4__roundmeanfitness
Chain
1
2
3
4




```
plot(bivar3.all.brm.nb)
```

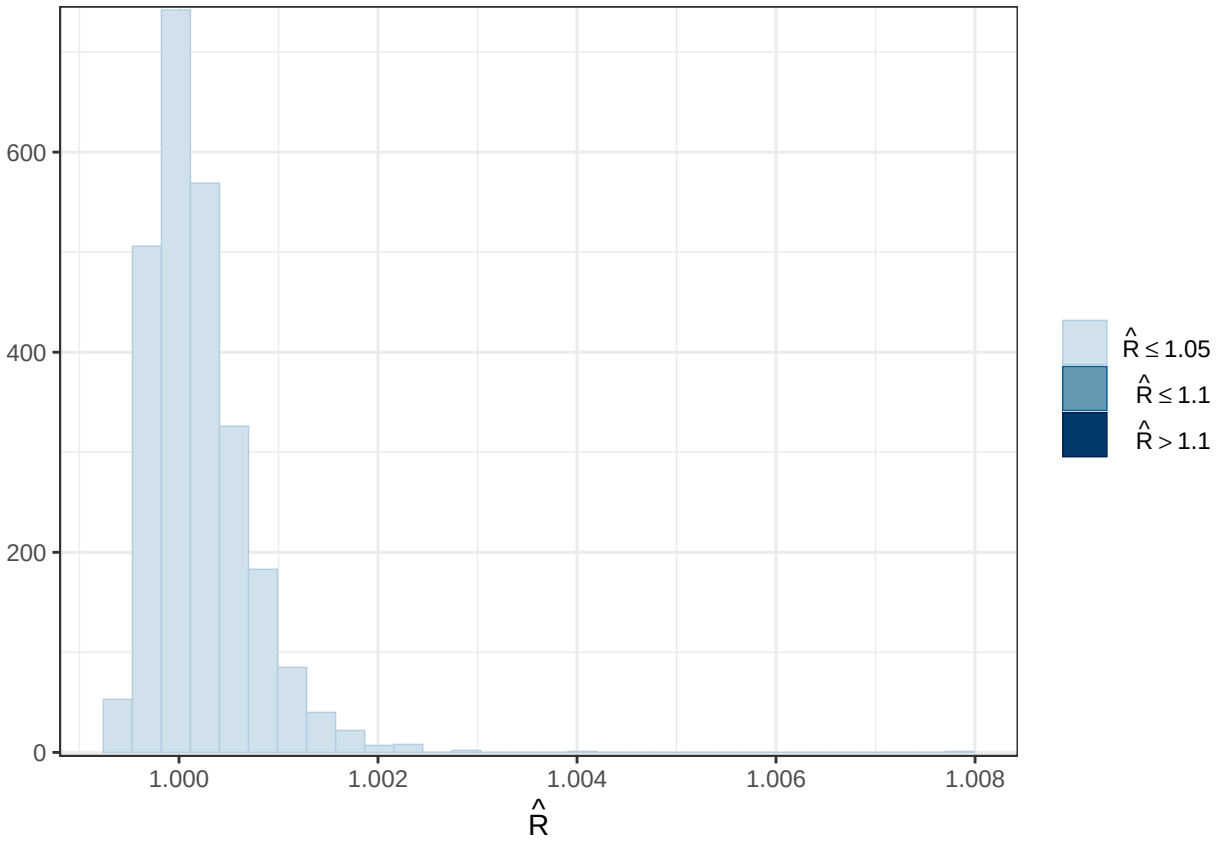

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessstudy_Intercept
oundmeanfitnessstudy_Interc
b_FFD_cmean_4
b_FFD_cmean_4
Chain
1
2
3
4
sd_id__FFD_Intercept
sd_id__FFD_Intercept
sd_id__FFD_cmean_4
sd_id__FFD_cmean_4


sd_id__roundmeanfitnessstudy_Intercept
__roundmeanfitnessstudy_Int
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
FFD_Intercept__roundmeanfitnessstudy_
ntercept__roundmeanfitness
FFD_cmean_4__roundmeanfitnessstudy_
cmean_4__roundmeanfitness
Chain
1
2
3
4

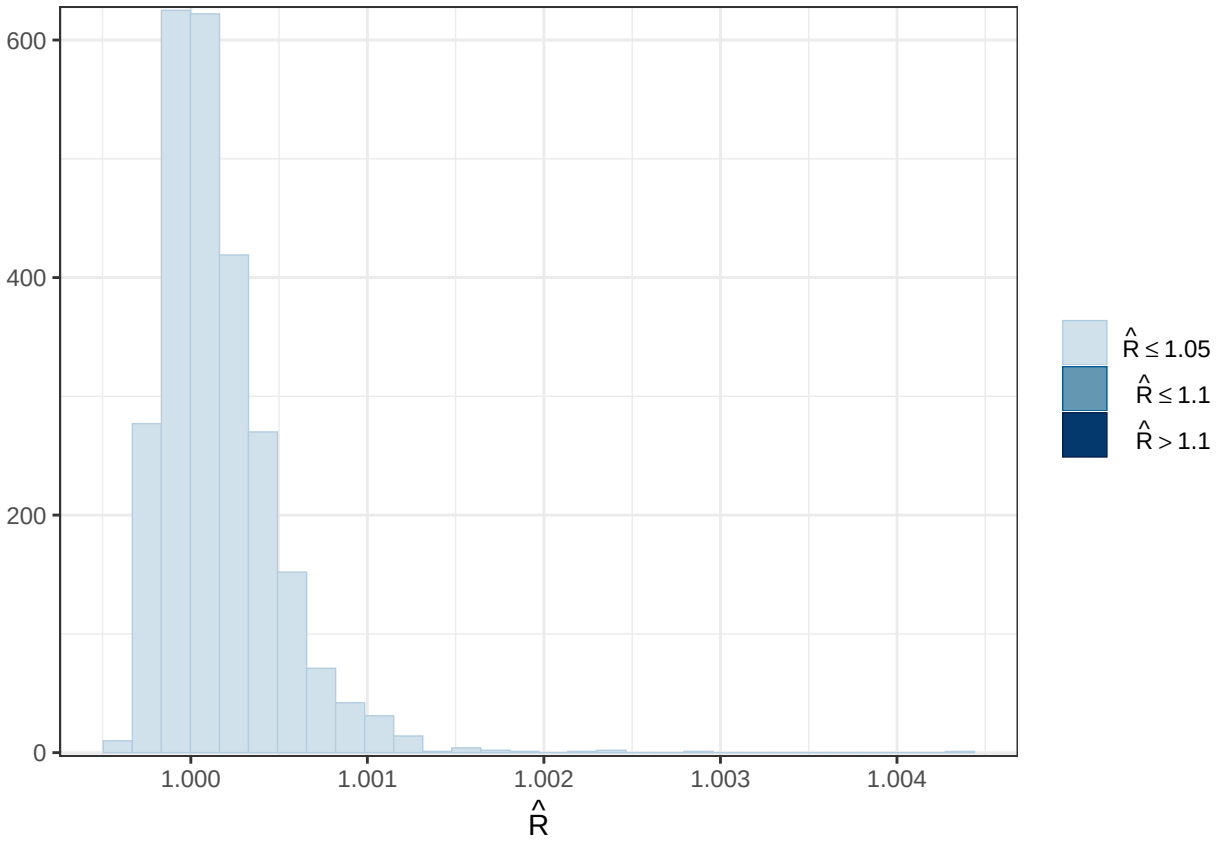

5
4
3
2
1
0
4.1
4.2
4.3
4.4
4.5
4.6
sigma_FFD
4.6
4.4
4.2
sigma_FFD
0
500
1000
1500
2000
2500
Chain
1
2
3
4

Rhat

```
mcmc_rhat_hist(rhat(bivar3.all.brm.pois))+theme_bw()
```

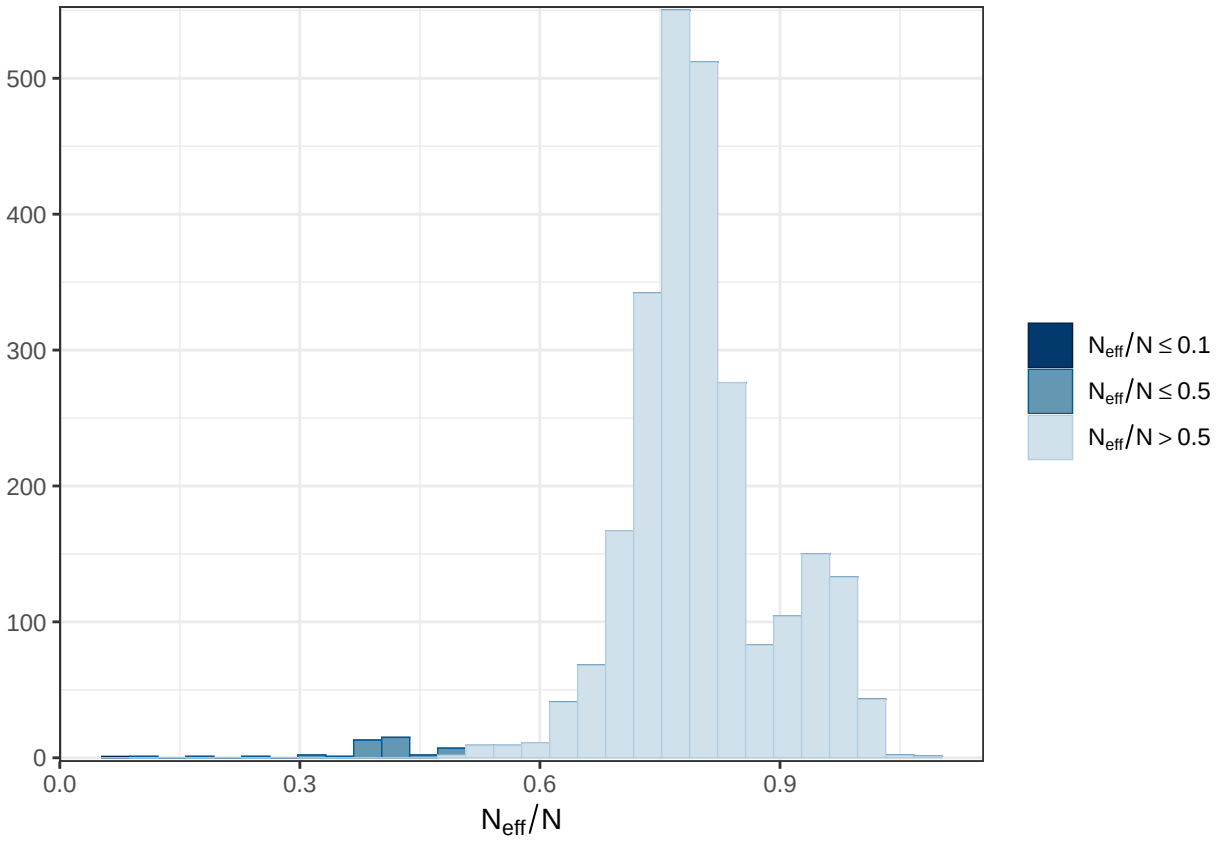



```
mcmc_rhat_hist(rhat(bivar3.all.brm.nb))+theme_bw()
```

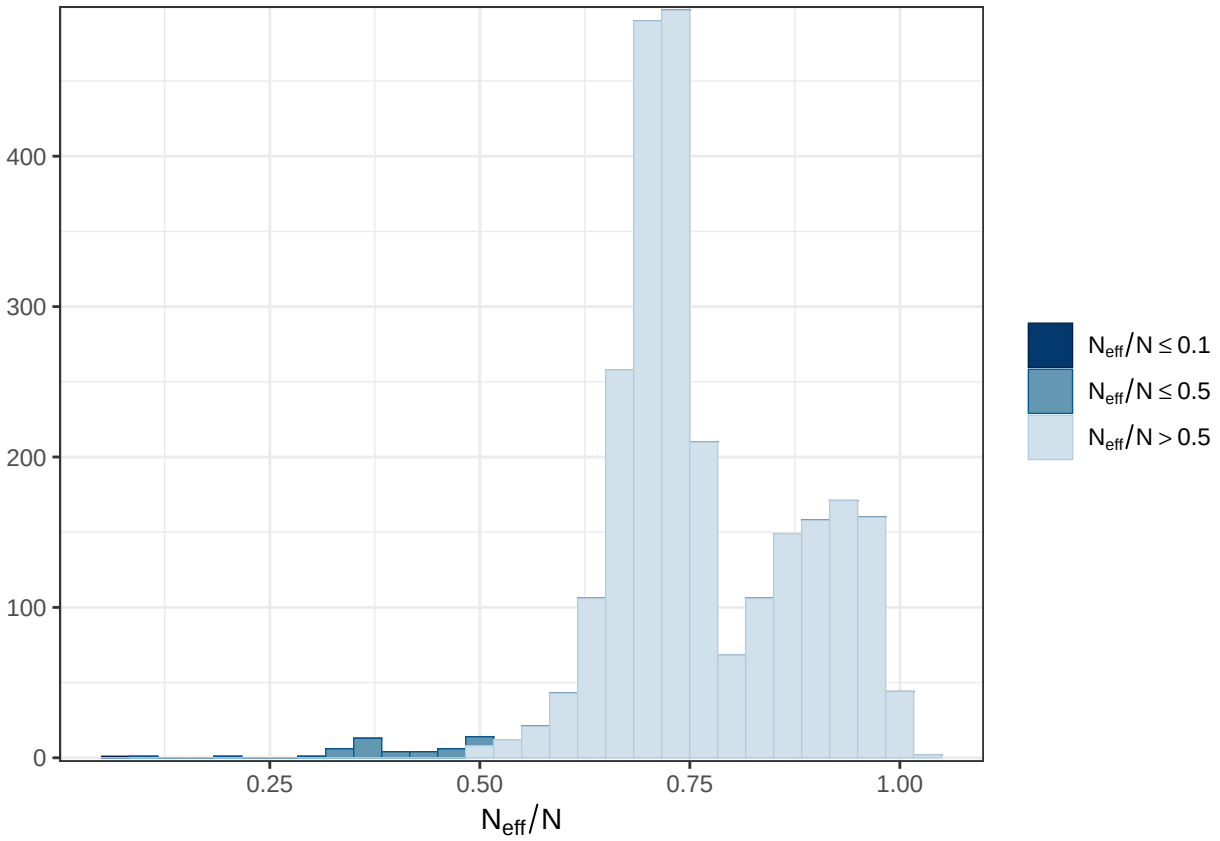


Effective sample size:

```
mcmc_neff_hist(neff_ratio(bivar3.all.brm.pois))+theme_bw()
```




```
mcmc_neff_hist(neff_ratio(bivar3.all.brm.nb))+theme_bw()
```



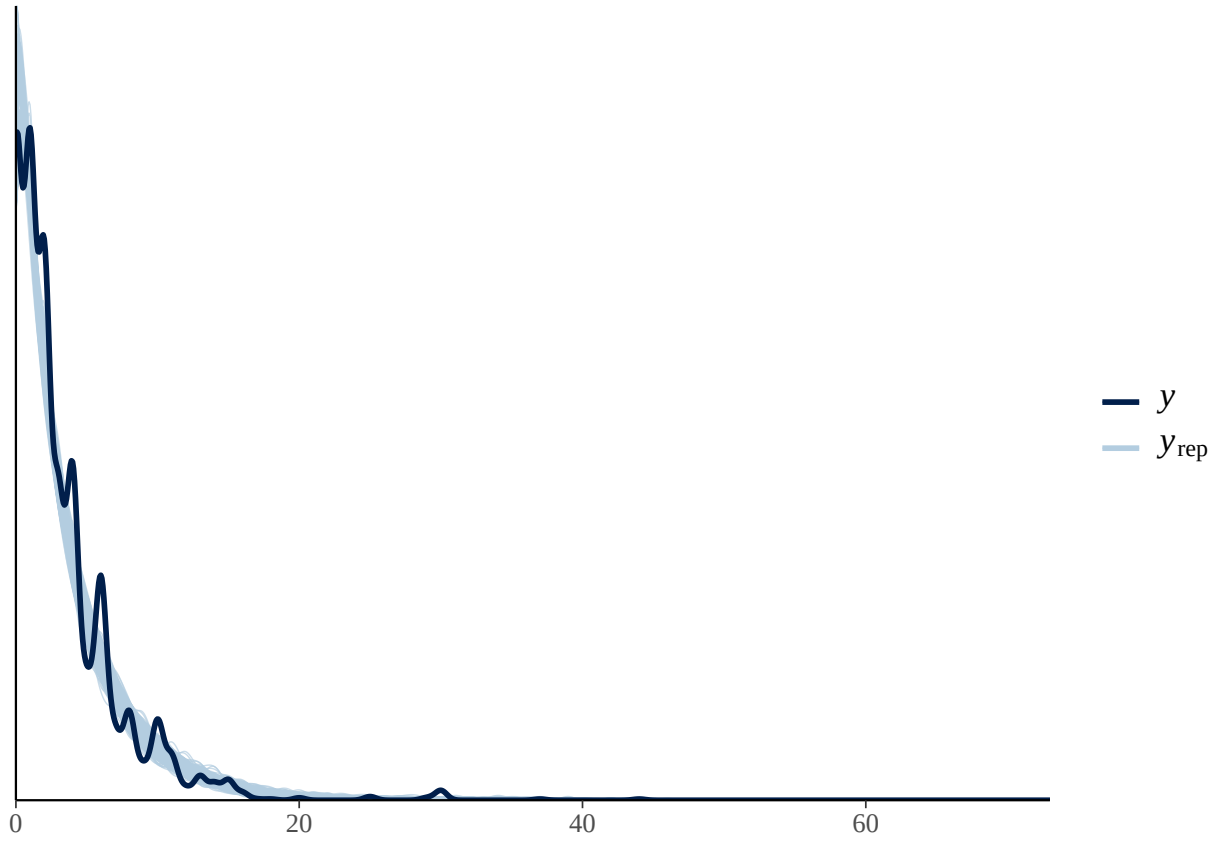
Posterior predictive checks:

```
y3_fitness<-round(subset(datadef,!is.na(FFD))$mean_fitness_study)
yrep4_fitness_pois<-posterior_predict(bivar3.all.brm.pois,
                                      draws = 500,resp="roundmeanfitnessstudy")
yrep4_FFD_pois<-posterior_predict(bivar3.all.brm.pois,
                                  draws = 500,resp="FFD")
yrep4_fitness_nb<-posterior_predict(bivar3.all.brm.nb,
                                    draws = 500,resp="roundmeanfitnessstudy")
yrep4_FFD_nb<-posterior_predict(bivar3.all.brm.nb,
                                draws = 500,resp="FFD")
# matrices of draws from the posterior predictive distribution
```

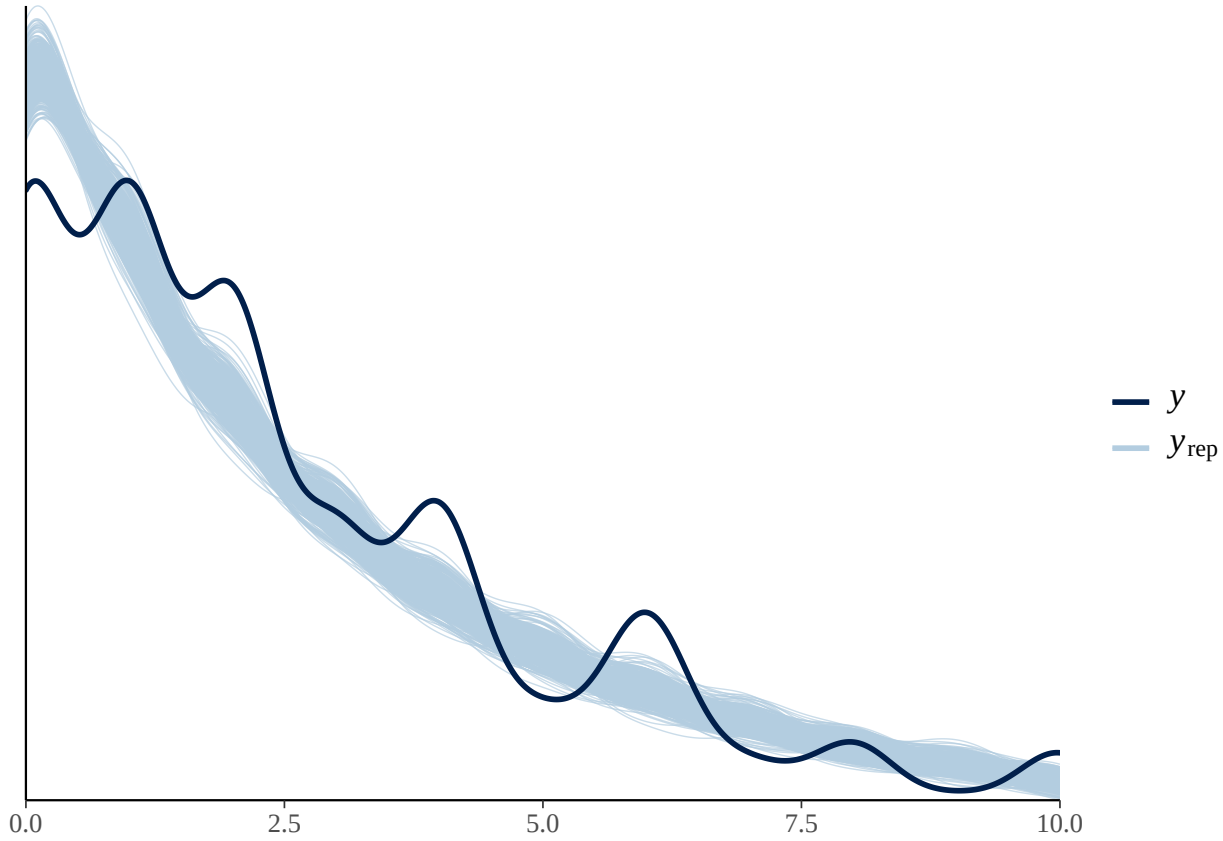
Each row of the matrix is a draw from the posterior predictive distribution, i.e. a vector with one element for each of the data points in y.

Comparison of the distribution of y and the distributions of the simulated datasets in the yrep matrix

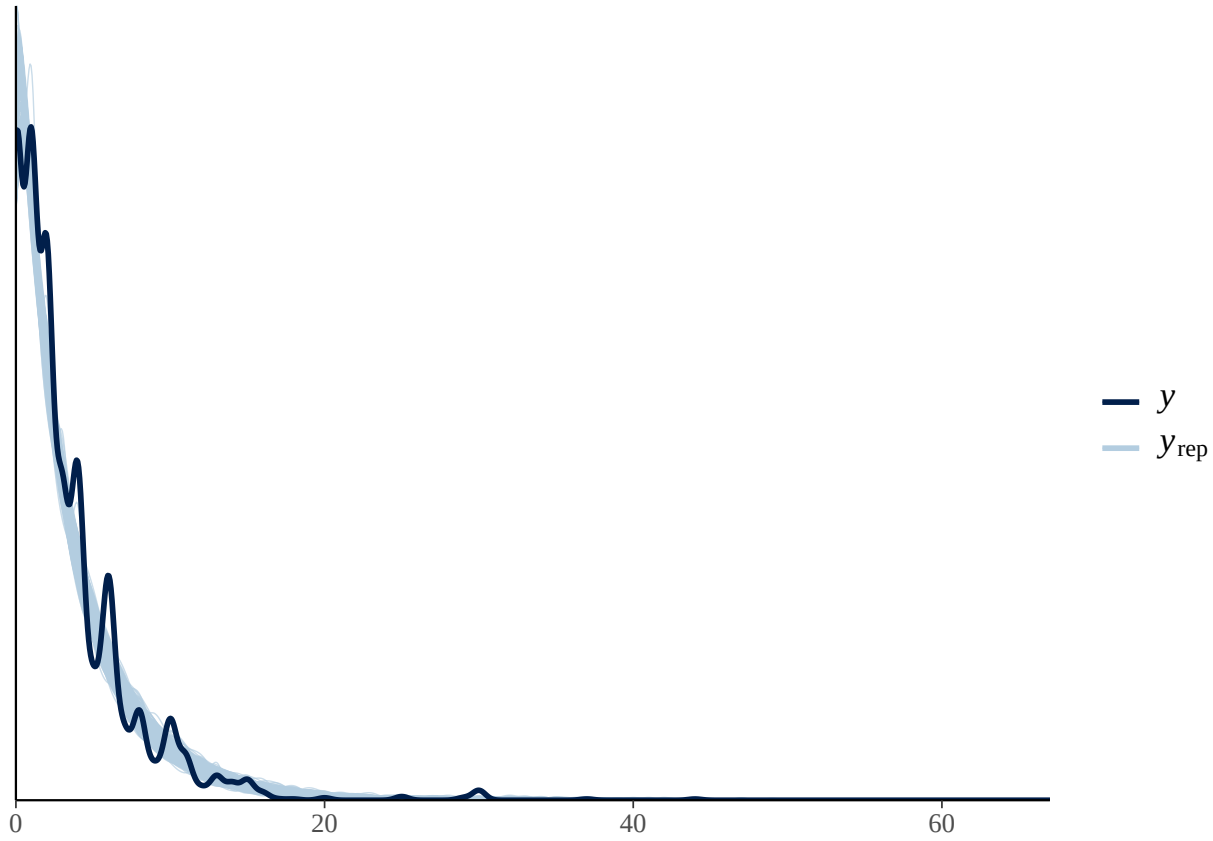
```
ppc_dens_overlay(y3_fitness, yrep4_fitness_pois[1:500,])
```



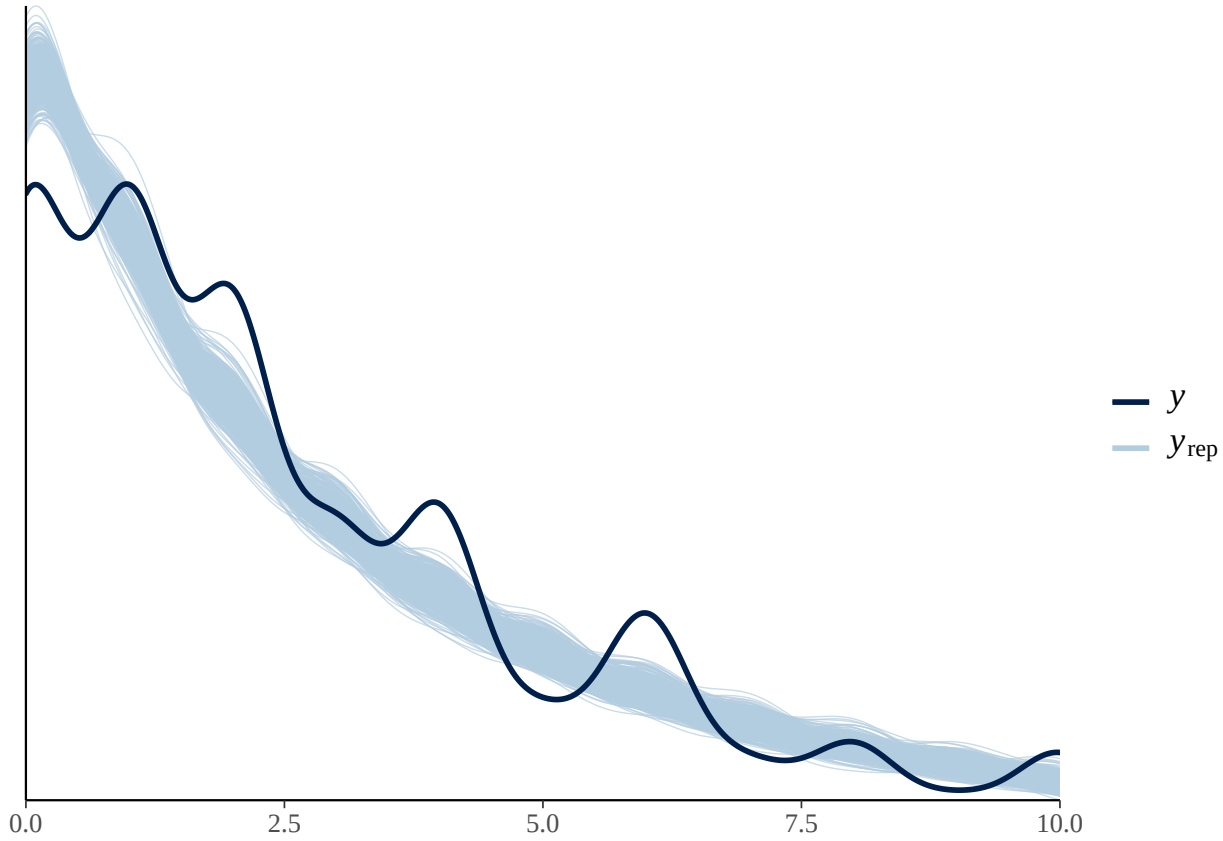

```
ppc_dens_overlay(y3_fitness, yrep4_fitness_pois[1:500,])+xlim(0, 10)
```



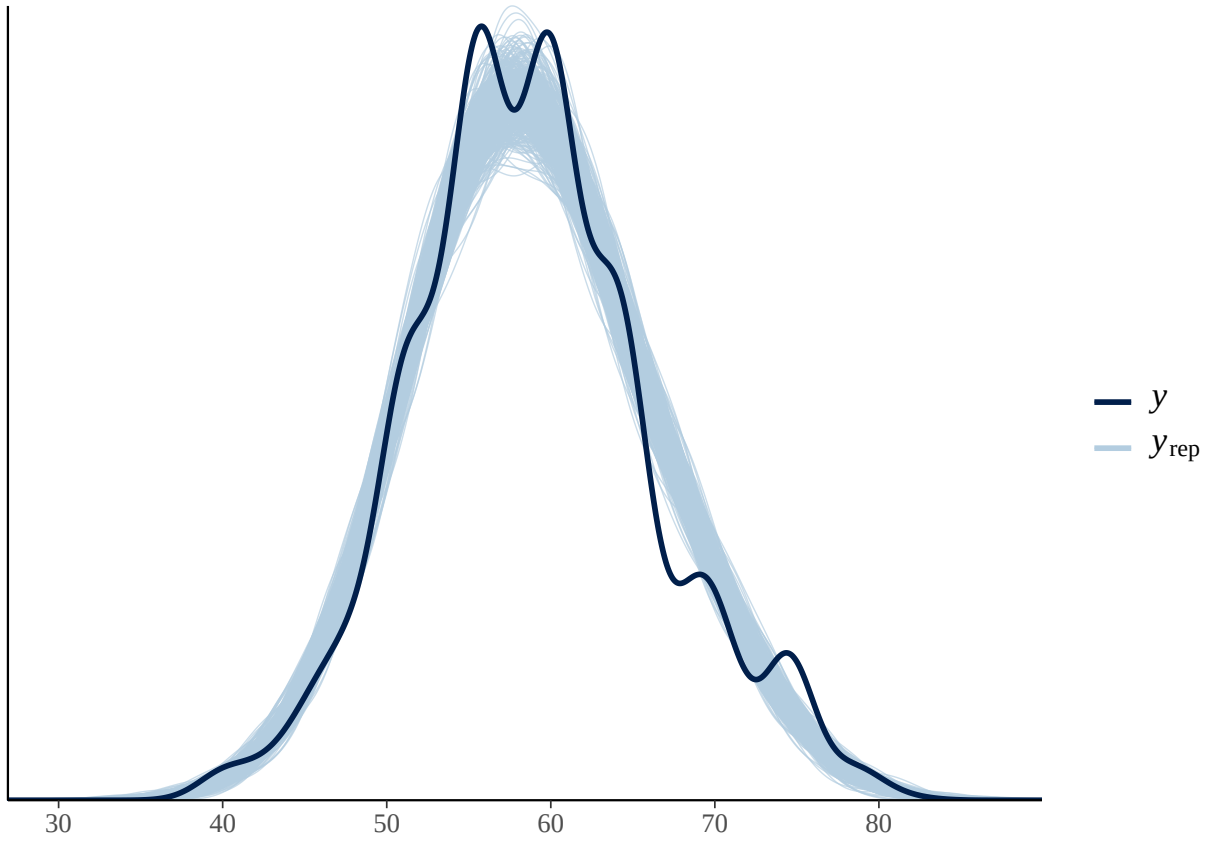

```
ppc_dens_overlay(y3_fitness, yrep4_fitness_nb[1:500,])
```



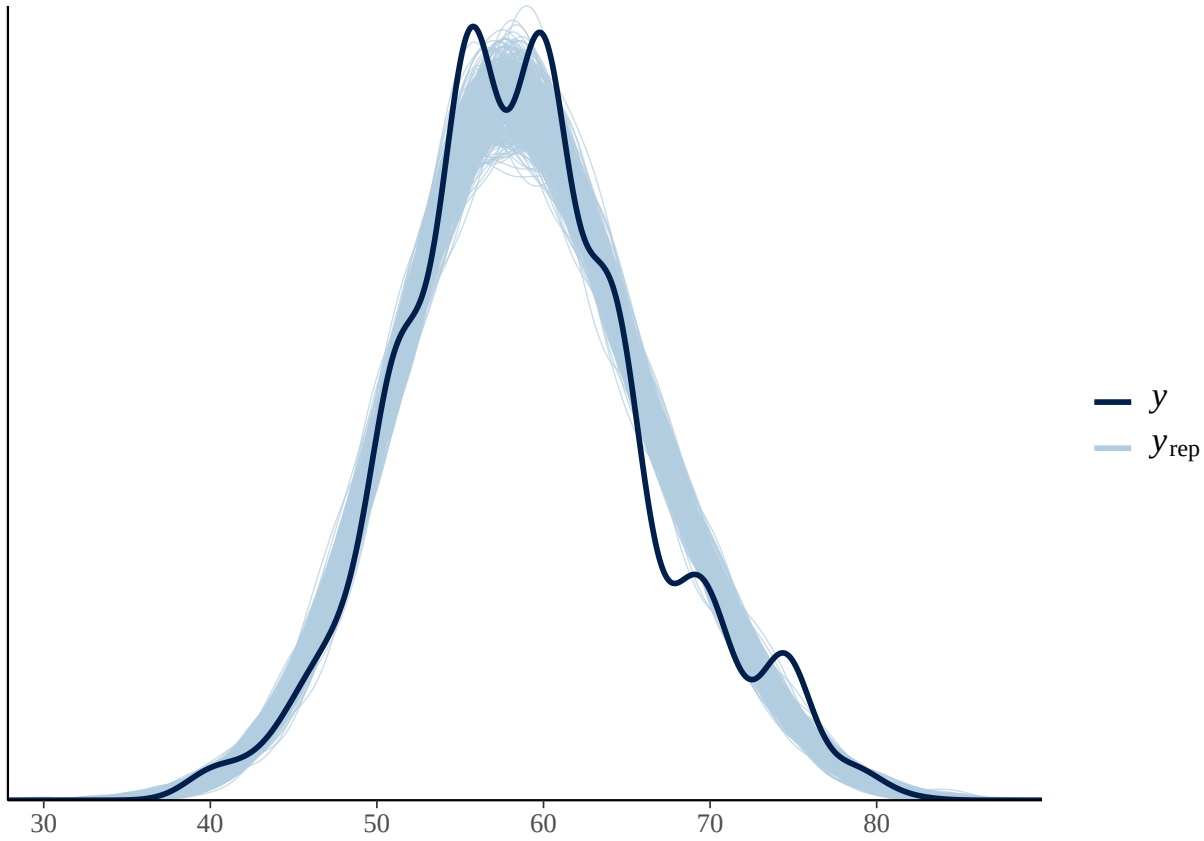

```
ppc_dens_overlay(y3_fitness, yrep4_fitness_nb[1:500,])+xlim(0,10)
```




```
ppc_dens_overlay(y2_FFD, yrep4_FFD_pois[1:500,])
```



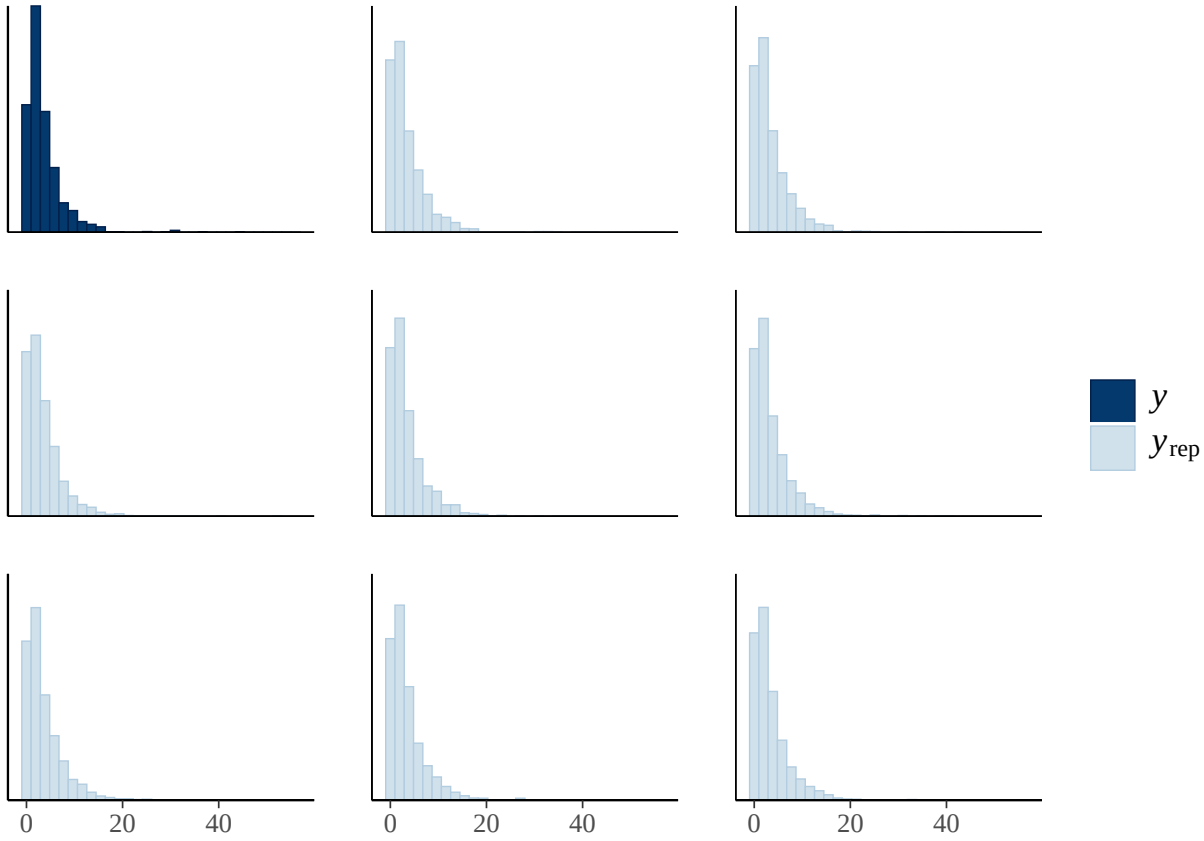

```
ppc_dens_overlay(y2_FFD, yrep4_FFD_nb[1:500,])
```



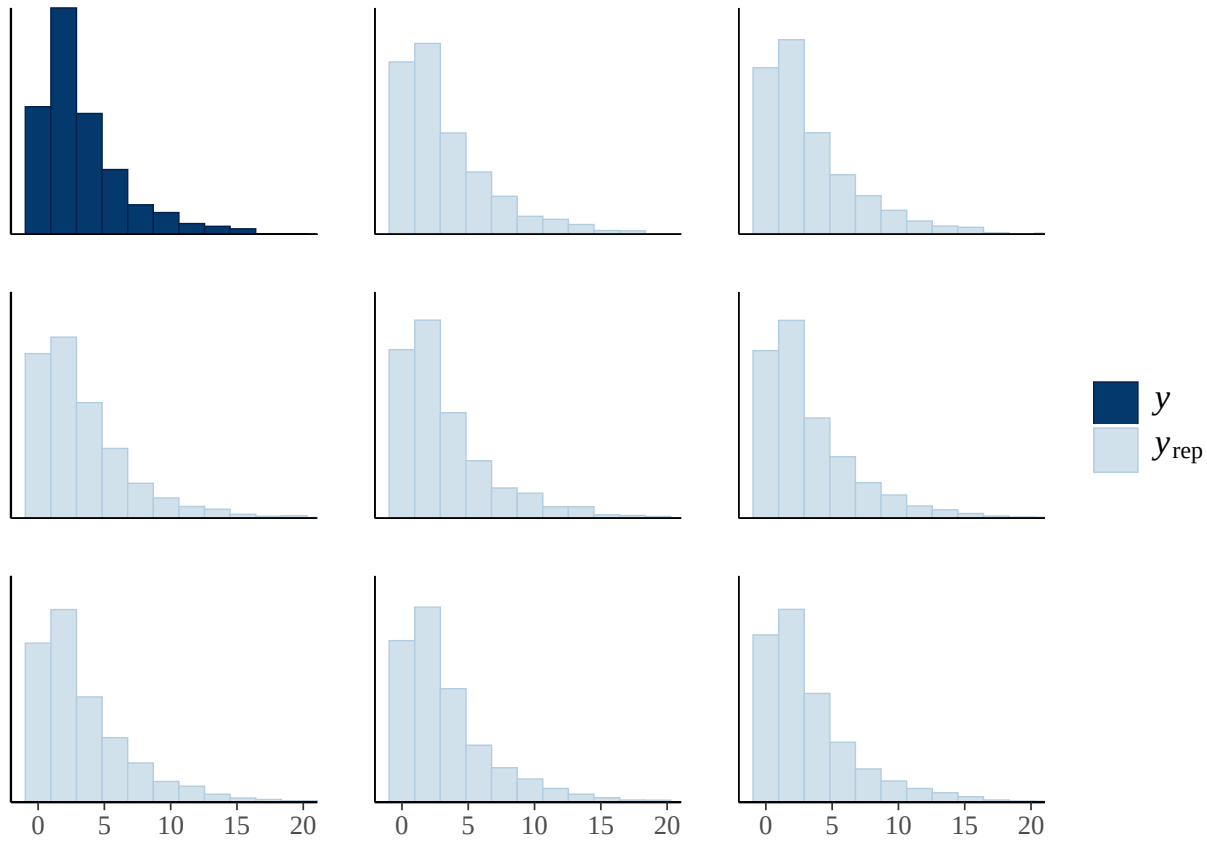
Poisson and negative binomial look pretty similar.

Separate histograms of y and some of the yrep datasets

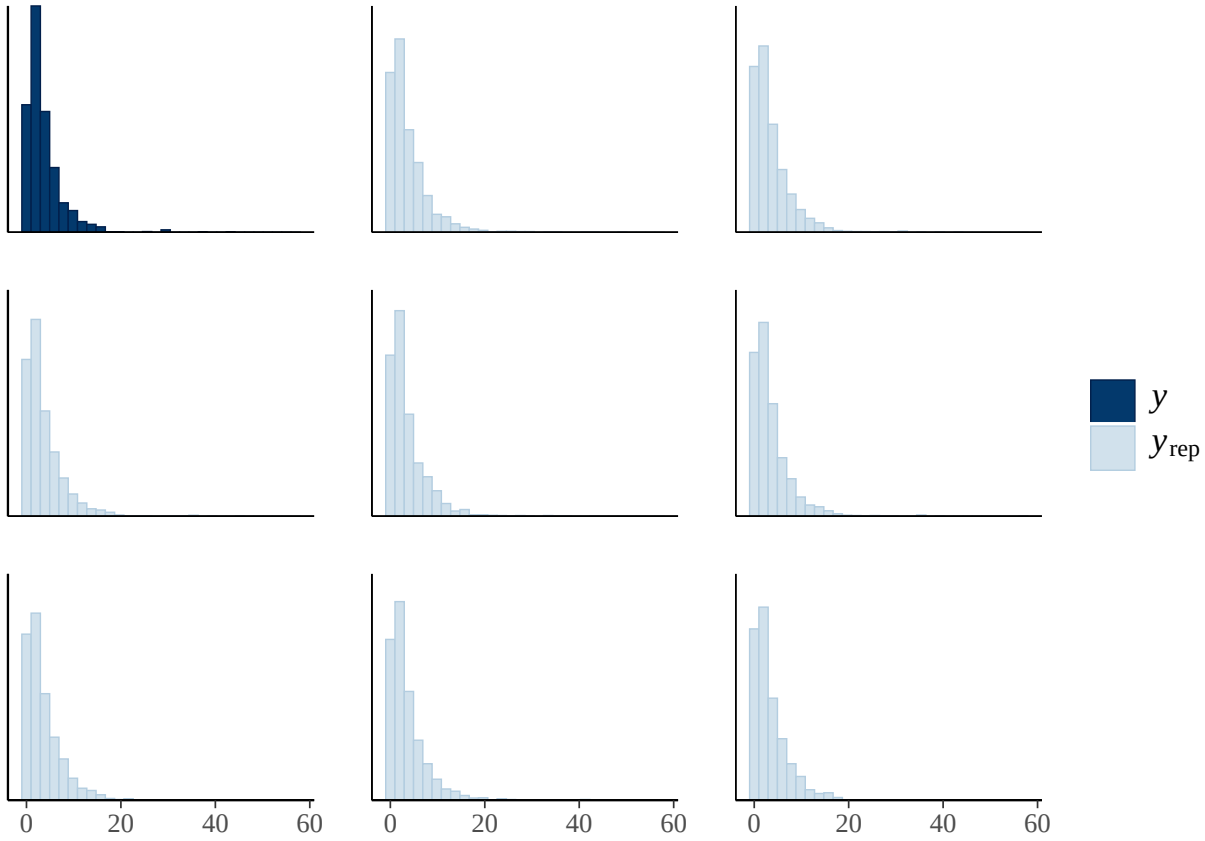
```
ppc_hist(y3_fitness, yrep4_fitness_pois[1:8, ])
```



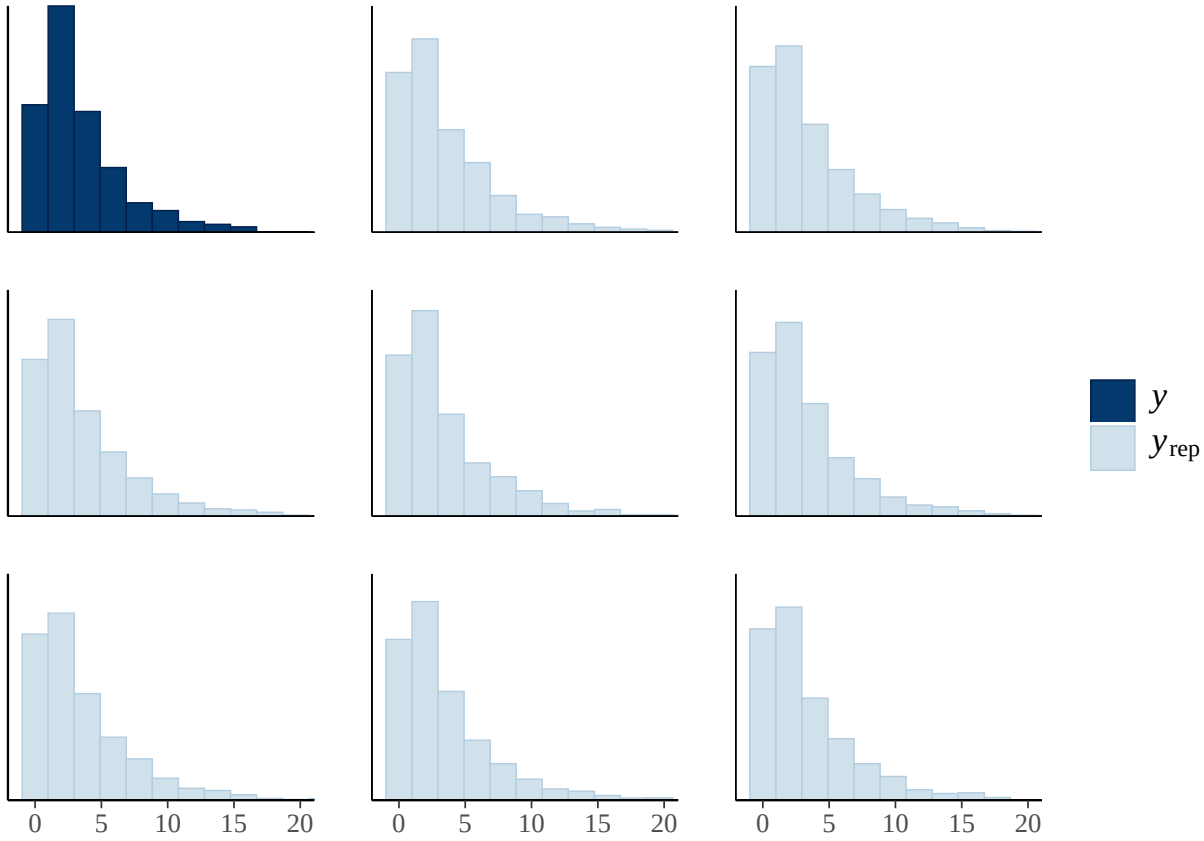

```
ppc_hist(y3_fitness, yrep4_fitness_pois[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```




```
ppc_hist(y3_fitness, yrep4_fitness_nb[1:8, ])
```

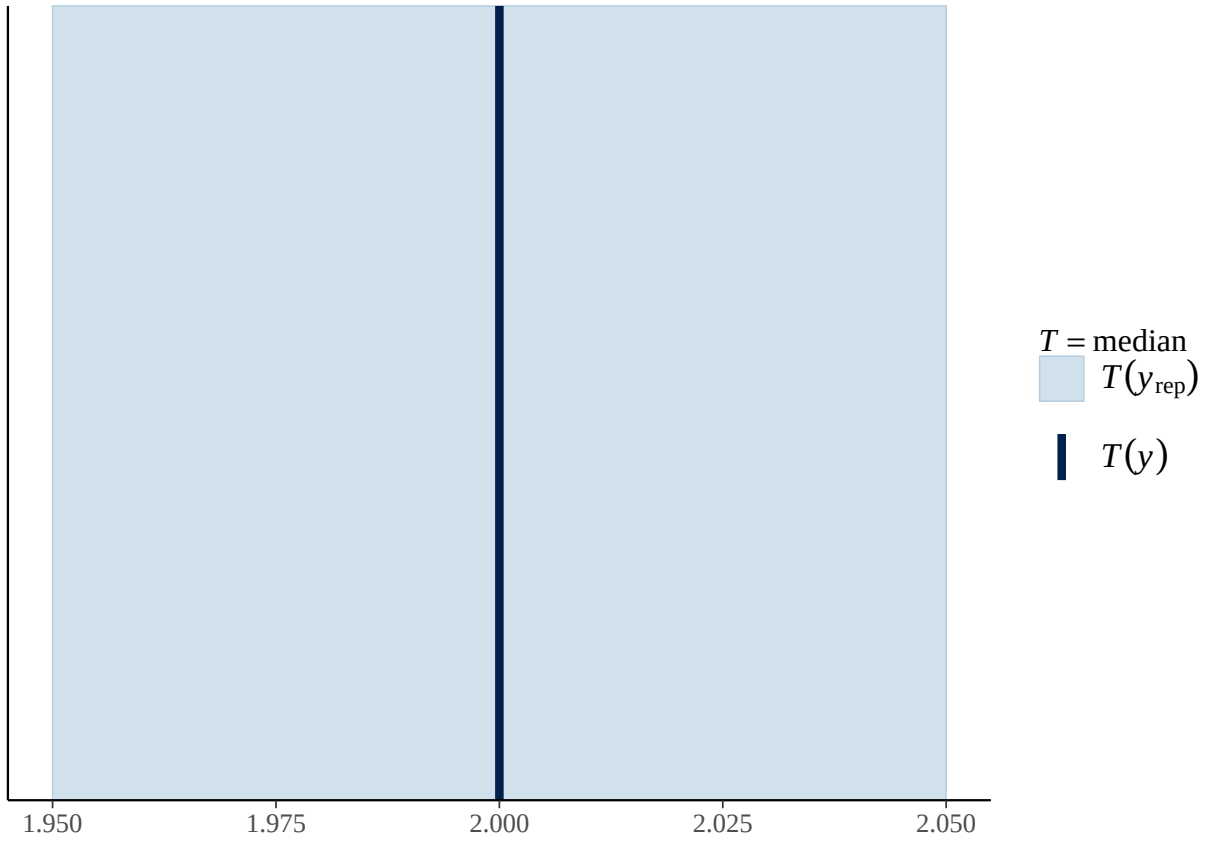



```
ppc_hist(y3_fitness, yrep4_fitness_nb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```

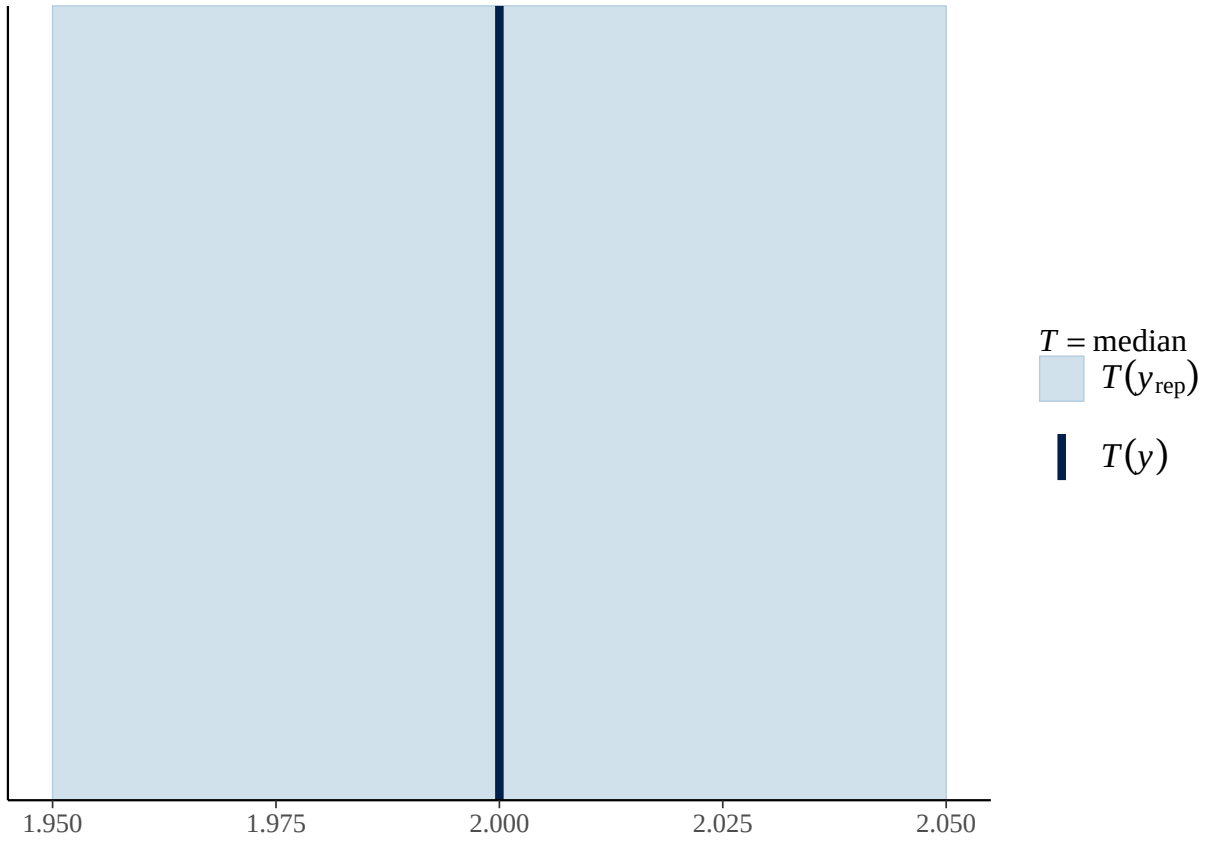


Here, negative binomial looks a bit better for fitness.

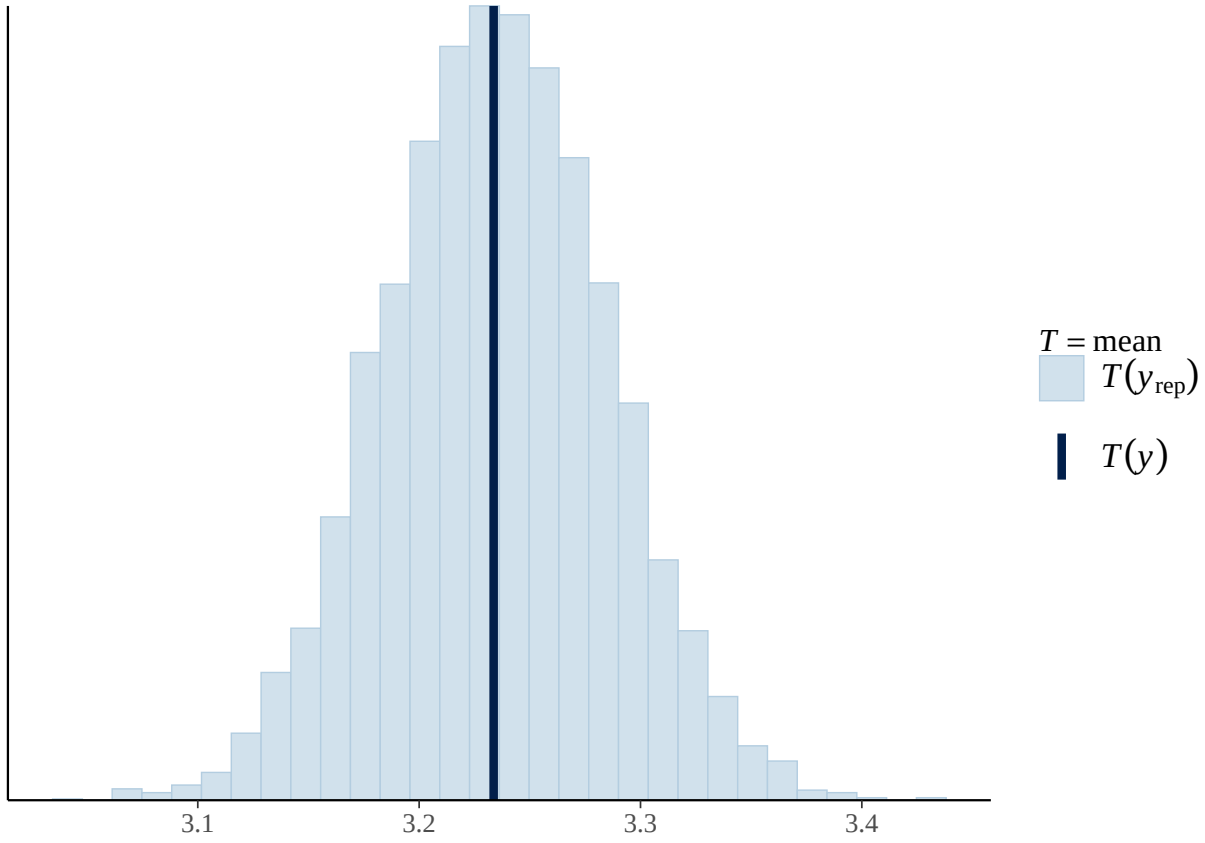
```
ppc_stat(y3_fitness, yrep4_fitness_pois,stat="median")
```



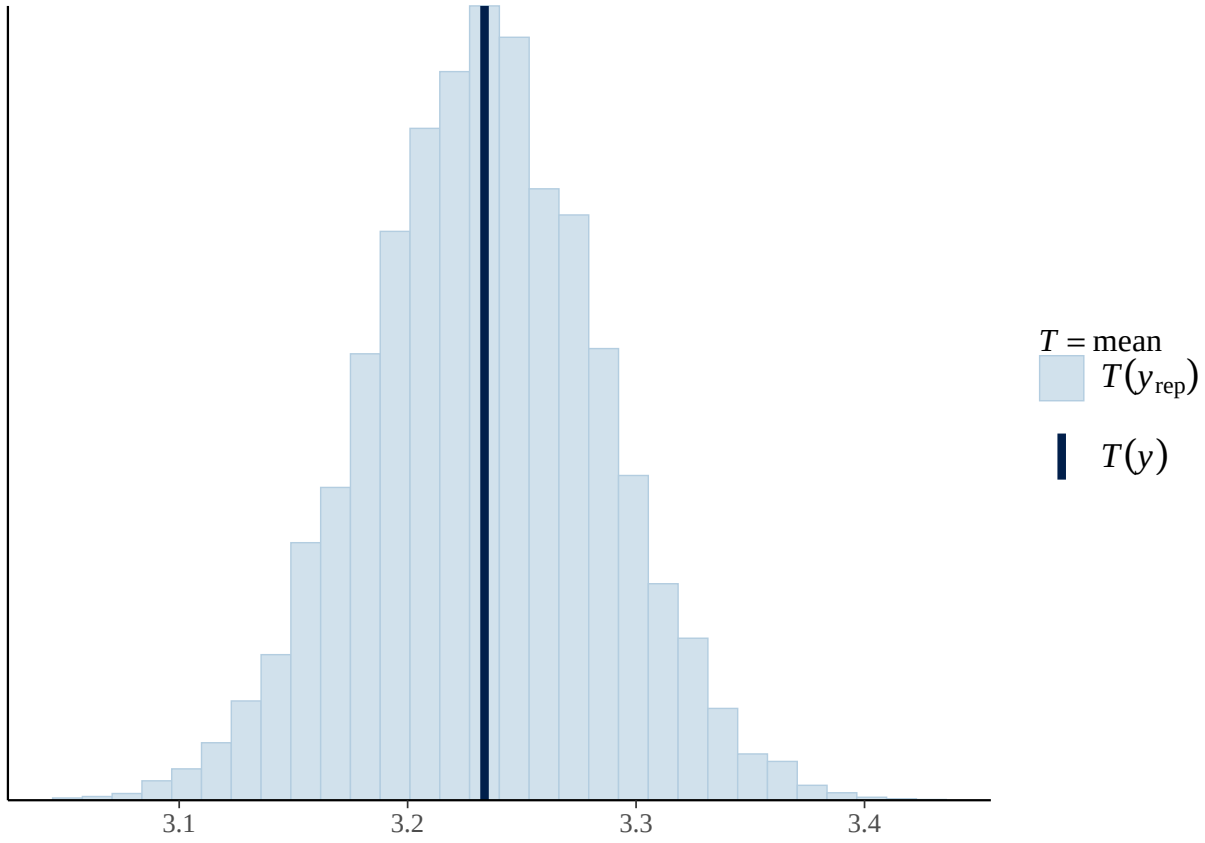

```
ppc_stat(y3_fitness, yrep4_fitness_nb,stat="median")
```



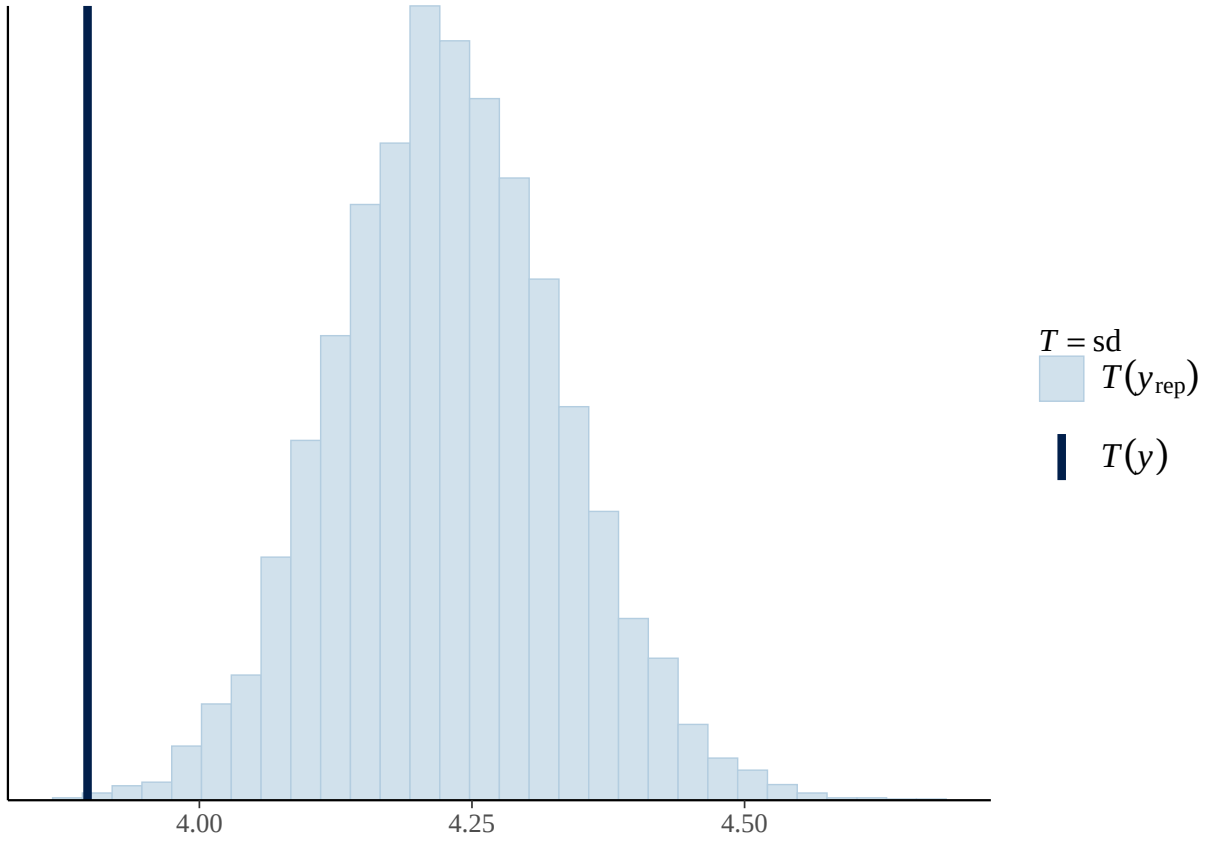

```
ppc_stat(y3_fitness, yrep4_fitness_pois,stat="mean")
```



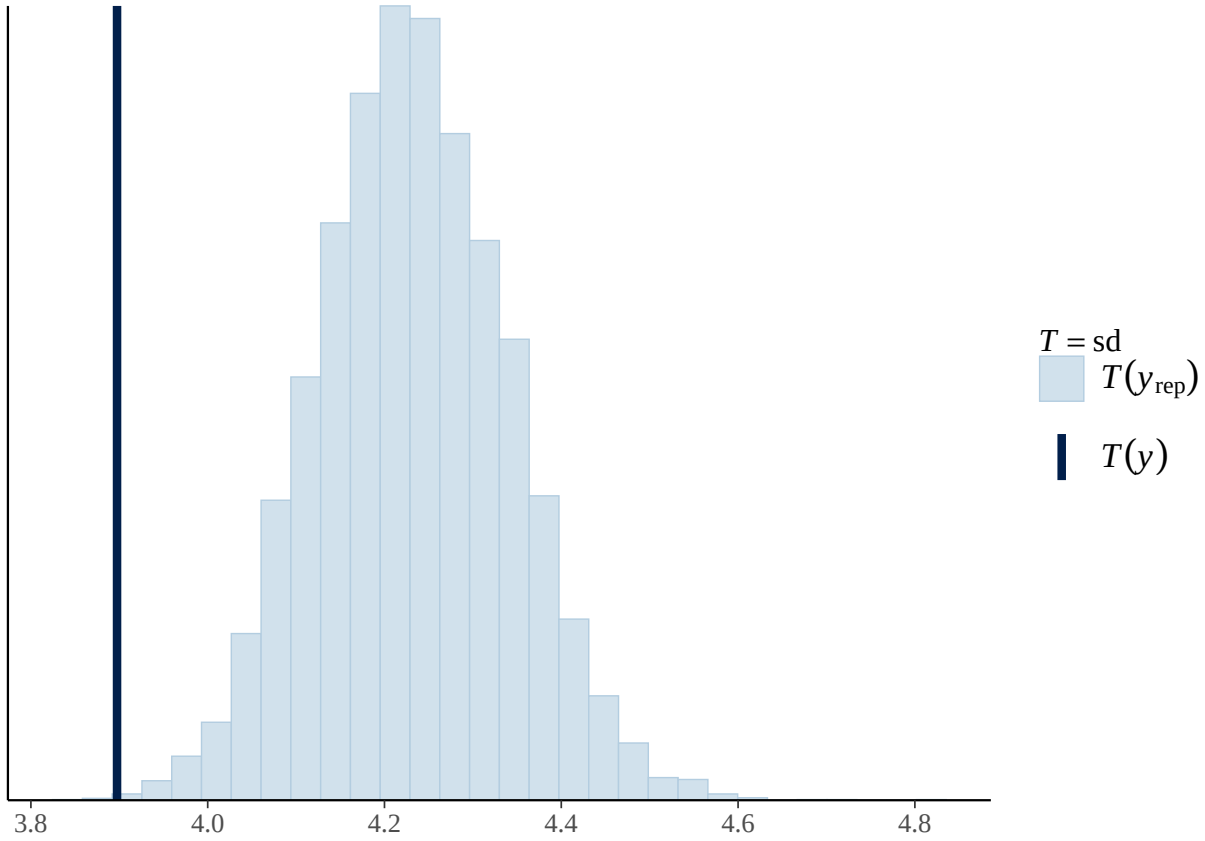

```
ppc_stat(y3_fitness, yrep4_fitness_nb,stat="mean")
```



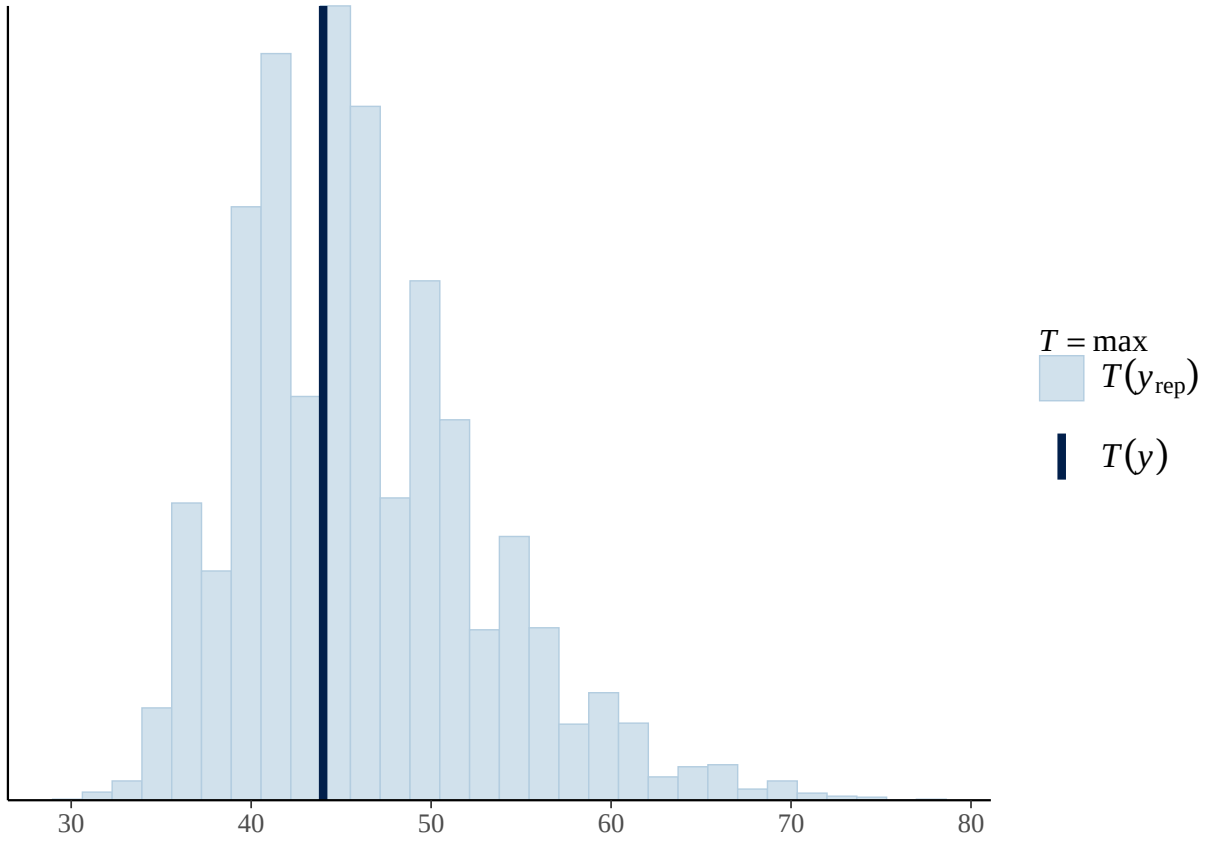

```
ppc_stat(y3_fitness, yrep4_fitness_pois,stat="sd")
```



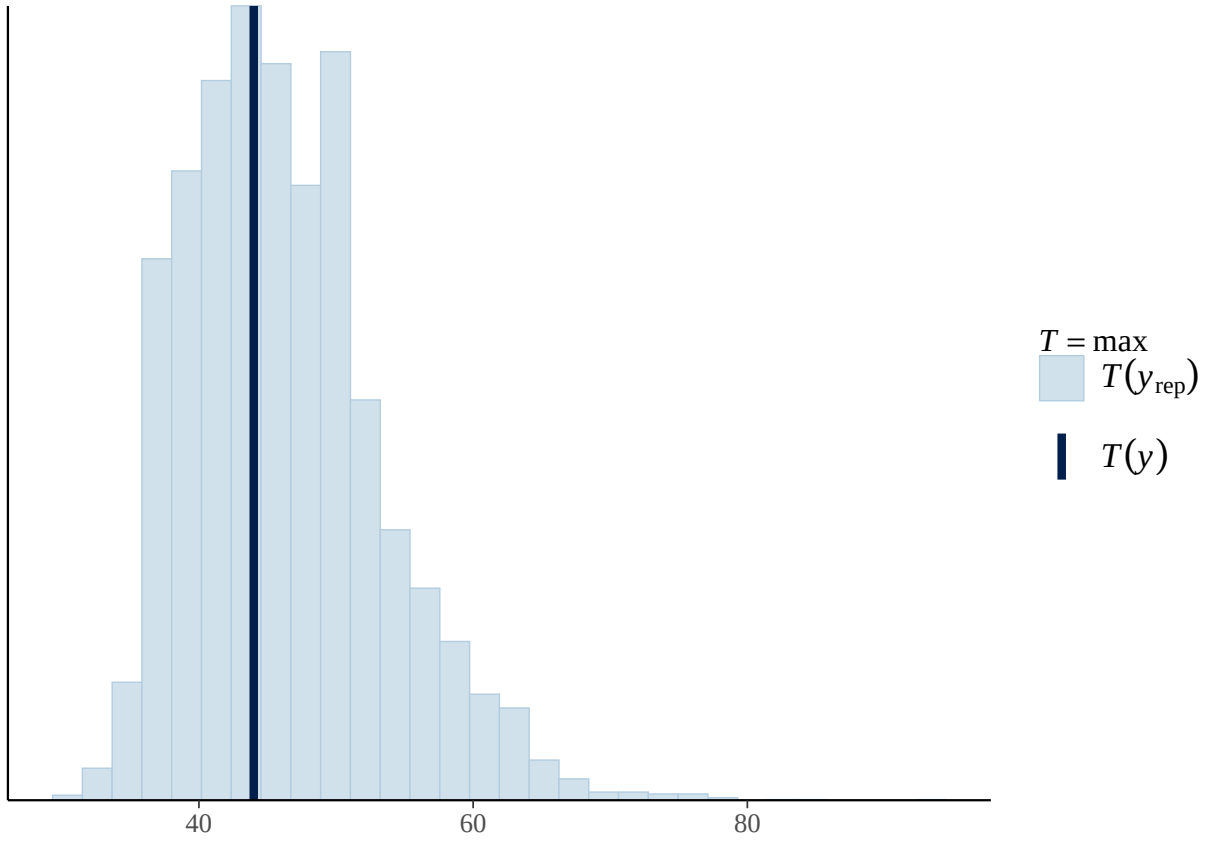

```
ppc_stat(y3_fitness, yrep4_fitness_nb,stat="sd")
```




```
ppc_stat(y3_fitness, yrep4_fitness_pois,stat="max")
```

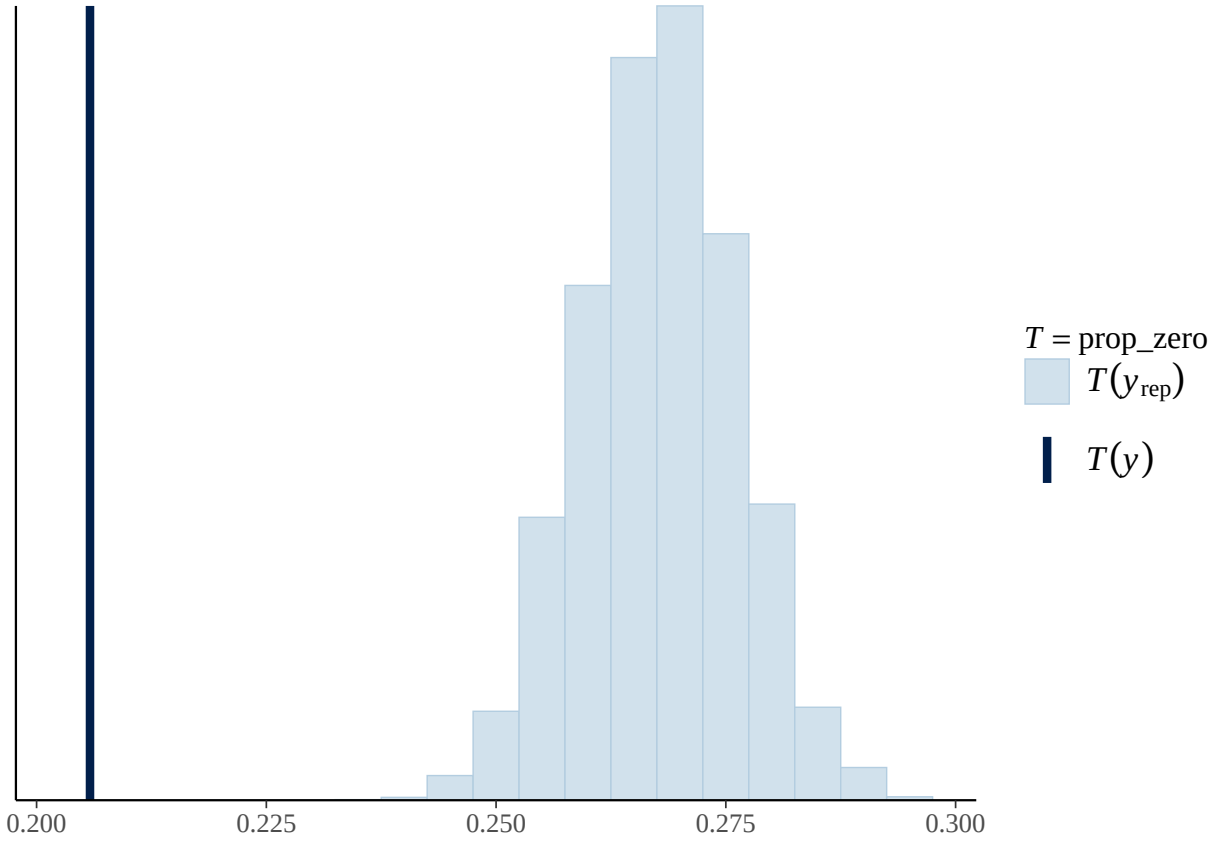



```
ppc_stat(y3_fitness, yrep4_fitness_nb,stat="max")
```

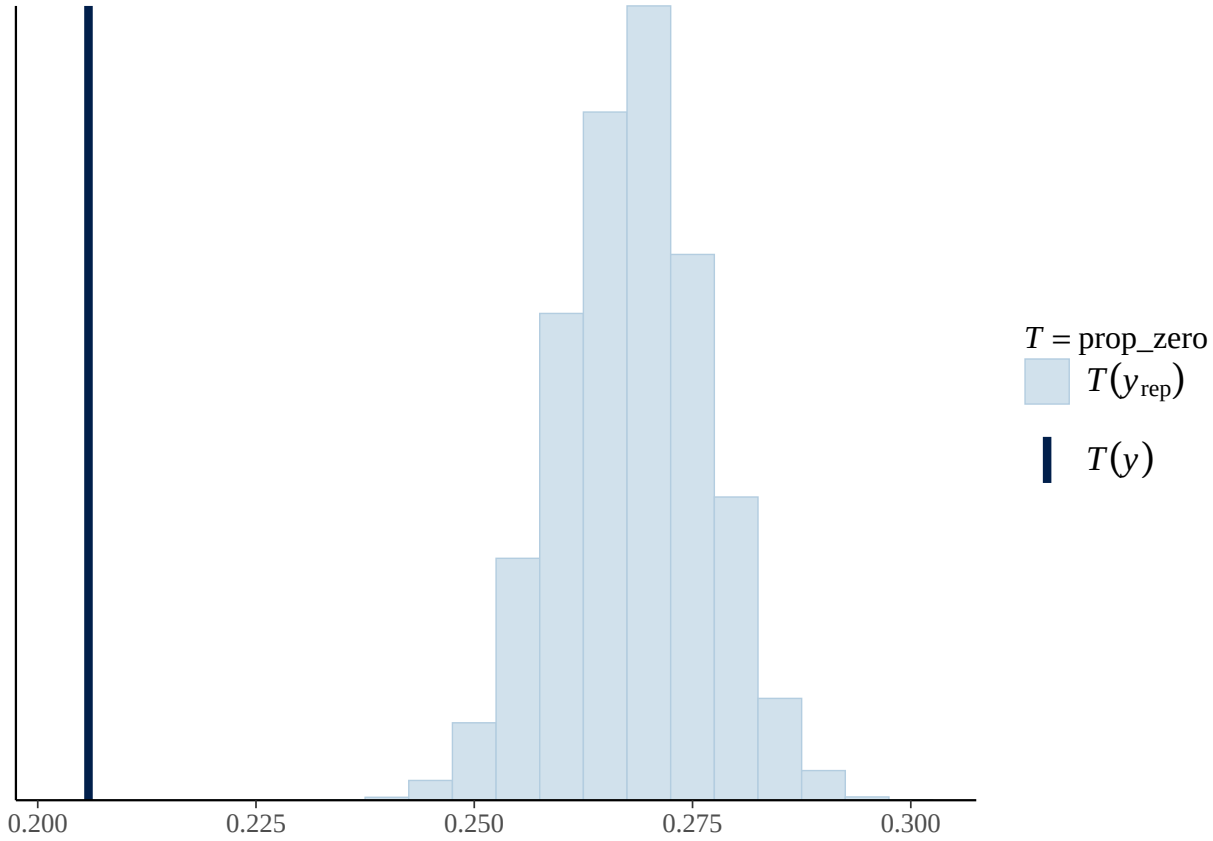


Look at the distribution of the proportion of zeros over the replicated datasets from the posterior predictive distribution in yrep and compare to the proportion of observed zeros in y.

```
ppc_stat(y3_fitness, yrep4_fitness_pois, stat = "prop_zero", binwidth = 0.005)
```




```
ppc_stat(y3_fitness, yrep4_fitness_nb, stat = "prop_zero", binwidth = 0.005)
```



They look quite similar.

Measure of fit: Bayesian R2

```
bayes_R2(bivar3.all.brm.pois)
```

```
##                            Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                     0.6470589 0.009447465 0.6280842 0.6652336
## R2roundmeanfitnessstudy 0.9402814 0.005339740 0.9289467 0.9499403
```

```
bayes_R2(bivar3.all.brm.nb)
```

```
##                            Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                     0.6470670 0.009320742 0.6286629 0.6653362
## R2roundmeanfitnessstudy 0.9393788 0.005578438 0.9274224 0.9494326
```

Very similar.

Leave-one-out cross validation (LOO): (not run)

Extract selection coefficients

```
# Extract posterior samples
bivar3.all.brm.pois_post <- posterior_samples(bivar3.all.brm.pois)
bivar3.all.brm.pois_post <- as.mcmc(bivar3.all.brm.pois_post)
#head(bivar3.all.brm.pois_post)[,1:20]

# [,4] sd_id__FFD_Intercept
# [,5] sd_id__FFD_cmean_4
# [,6] sd_id__roundmeanfitnessfl_Intercept
# [,8] cor_id__FFD_Intercept__FFD_cmean_4
# [,9] cor_id__FFD_Intercept__roundmeanfitnessfl_Intercept
# [,10] cor_id__FFD_cmean_4__roundmeanfitnessfl_Intercept

sampled.gmat5 <- sample.gmat1(bivar3.all.brm.pois_post, replicates = 1000)

sgmat5 <- lapply(sampled.gmat5, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat5 <- unname(sapply(sgmat5, '[[', 1)) #Change to matrix

sgmat5 <- t(sgmat5)

P.modelBV_RR5 <- sgmat5
P.modelBV_RR5.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR5.mode[k] <- posterior.mode(mcmc(sgmat5[,k]))

# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR5 <- sgmat5[,c(3,6)]
colnames(S.modelBV_RR5) <- c("S_intercepts", "S_slopes")
S.modelBV_RR5.mode <- P.modelBV_RR5.mode[1:2, 3]

posterior.mode(mcmc(S.modelBV_RR5))
```

Poisson model

```
## S_intercepts     S_slopes
##   -1.3966145   -0.1702053
```

```
HPDinterval(mcmc(S.modelBV_RR5))
```

```
##                   lower      upper
## S_intercepts -1.6590918 -0.9375602
## S_slopes     -0.4555992  0.0897808
## attr(,"Probability")
## [1] 0.95
```

```
# Estimate selection gradients for intercept and slope (beta = S / P)
# on each sample of posterior and extract their mode
beta_post_RR5 <- matrix(NA, nrow(S.modelBV_RR5) ,2)

for (i in 1:nrow(S.modelBV_RR5)) {
  P3_5 <- matrix(rep(NA, 9), nrow = 3)
  # 3x3 matrix of var-cov for individual X.int, X.slope and fitness
  for (k in 1:9) {P3_5[k] <- P.modelBV_RR5[i, k] }
```

```
  P2_5 <- P3_5[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S5 <- P3_5[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR5[i,] <- solve(P2_5) %*% S5   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR5) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR5))
```

```
## beta_intercepts     beta_slopes
##      -0.6109206       0.5925177
```

```
HPDinterval(mcmc(beta_post_RR5))
```

```
##                        lower      upper
## beta_intercepts -1.97763499 -0.3595693
## beta_slopes      0.01087413  3.6740509
## attr(,"Probability")
## [1] 0.95
```

```
# Extract posterior samples
bivar3.all.brm.nb_post <- posterior_samples(bivar3.all.brm.nb)
bivar3.all.brm.nb_post <- as.mcmc(bivar3.all.brm.nb_post)
#head(bivar3.all.brm.nb_post)[,1:20]

# [,4] sd_id__FFD_Intercept
# [,5] sd_id__FFD_cmean_4
# [,6] sd_id__roundmeanfitnessfl_Intercept
# [,8] cor_id__FFD_Intercept__FFD_cmean_4
# [,9] cor_id__FFD_Intercept__roundmeanfitnessfl_Intercept
# [,10] cor_id__FFD_cmean_4__roundmeanfitnessfl_Intercept

sampled.gmat6 <- sample.gmat1(bivar3.all.brm.nb_post, replicates = 1000)

sgmat6 <- lapply(sampled.gmat6, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat6 <- unname(sapply(sgmat6, '[[', 1)) #Change to matrix

sgmat6 <- t(sgmat6)

P.modelBV_RR6 <- sgmat6
P.modelBV_RR6.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR6.mode[k] <- posterior.mode(mcmc(sgmat6[,k]))

# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR6 <- sgmat6[,c(3,6)]
colnames(S.modelBV_RR6) <- c("S_intercepts", "S_slopes")
S.modelBV_RR6.mode <- P.modelBV_RR6.mode[1:2, 3]

posterior.mode(mcmc(S.modelBV_RR6))
```

Negative binomial model

```
## S_intercepts     S_slopes
##   -1.2868110   -0.1633548
```

```
HPDinterval(mcmc(S.modelBV_RR6))
```

```
##                  lower      upper
## S_intercepts -1.716891 -0.9292763
## S_slopes     -0.469778  0.0994991
## attr(,"Probability")
## [1] 0.95
```

```
# Estimate selection gradients for intercept and slope (beta = S / P)
# on each sample of posterior and extract their mode
beta_post_RR6 <- matrix(NA, nrow(S.modelBV_RR6) ,2)

for (i in 1:nrow(S.modelBV_RR6)) {
  P3_6 <- matrix(rep(NA, 9), nrow = 3)
  # 3x3 matrix of var-cov for individual X.int, X.slope and fitness
  for (k in 1:9) {P3_6[k] <- P.modelBV_RR6[i, k] }
  P2_6 <- P3_6[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S6 <- P3_6[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR6[i,] <- solve(P2_6) %*% S6   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR6) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR6))
```

```
## beta_intercepts     beta_slopes
##      -0.7181076       0.9081047
```

```
HPDinterval(mcmc(beta_post_RR6))
```

```
##                         lower      upper
## beta_intercepts -1.970986132 -0.4543508
## beta_slopes      0.003247242  3.7034310
## attr(,"Probability")
## [1] 0.95
```

4. mean_fitness_study, with shoot volume

```
bf_fitness_study_shoot <- bf(round(mean_fitness_study) ~
                               cn_shoot_vol_mean_sqrt +
                           (1|ID1|id)) # Set up model formula
```

Poisson distribution

```
bivar4.all.brm.pois<-brm(bf_FFD+bf_fitness_study_shoot,
                         family = c(gaussian,poisson),
                         data = datadef,warmup = 2000,iter = 6000,chains=4,thin=2,
                         inits = "random",seed = 12345,cores = my.cores,
                         control = list(adapt_delta = 0.99, max_treedepth = 15))
```

```
summary(bivar4.all.brm.pois)
```

```
##  Family: MV(gaussian, poisson)
##   Links: mu = identity; sigma = identity
##          mu = log
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_study) ~ cn_shoot_vol_mean_sqrt + (1 | ID1 | id)
##    Data: datadef (Number of observations: 2432)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
##
## Group-Level Effects:
## ~id (Number of levels: 791)
##                                                    Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                      1.65      0.14     1.36
## sd(FFD_cmean_4)                                        0.80      0.13     0.55
## sd(roundmeanfitnessstudy_Intercept)                    1.20      0.05     1.11
## cor(FFD_Intercept,FFD_cmean_4)                         0.70      0.13     0.43
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)    -0.47      0.07    -0.61
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)       0.06      0.14    -0.20
##                                                    u-95% CI Rhat Bulk_ESS
## sd(FFD_Intercept)                                      1.93 1.00     2216
## sd(FFD_cmean_4)                                        1.05 1.01     1122
## sd(roundmeanfitnessstudy_Intercept)                    1.30 1.00     1651
## cor(FFD_Intercept,FFD_cmean_4)                         0.91 1.02      319
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)    -0.32 1.01     1227
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)       0.33 1.00     1004
##                                                    Tail_ESS
## sd(FFD_Intercept)                                      2348
## sd(FFD_cmean_4)                                        1041
## sd(roundmeanfitnessstudy_Intercept)                    3073
## cor(FFD_Intercept,FFD_cmean_4)                         2625
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)     1111
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)       2429
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.21      0.88     3.85     7.23 1.00      917     1080
##
## Population-Level Effects:
##                                            Estimate Est.Error l-95% CI
## FFD_Intercept                                 58.66      1.18    56.39
## roundmeanfitnessstudy_Intercept                0.13      0.06     0.02
## FFD_cmean_4                                   -2.39      0.83    -4.12
## roundmeanfitnessstudy_cn_shoot_vol_mean_sqrt   0.05      0.00     0.05
##                                            u-95% CI Rhat Bulk_ESS Tail_ESS
```

```
## FFD_Intercept                                    61.29 1.01      592      134
## roundmeanfitnessstudy_Intercept                   0.24 1.00     1646     3153
## FFD_cmean_4                                      -0.79 1.00     2126     3294
## roundmeanfitnessstudy_cn_shoot_vol_mean_sqrt      0.06 1.01      903      698
##
## Family Specific Parameters:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma_FFD     4.36      0.08     4.21     4.50 1.01      517     1096
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Negative binomial distribution

```
bivar4.all.brm.nb<-brm(bf_FFD+bf_fitness_study_shoot,
                       family = c(gaussian,negbinomial),
                       data = datadef,warmup = 1000,iter = 4000,chains=4,thin=2,
                       inits = "random",seed = 12345,cores = my.cores)
```

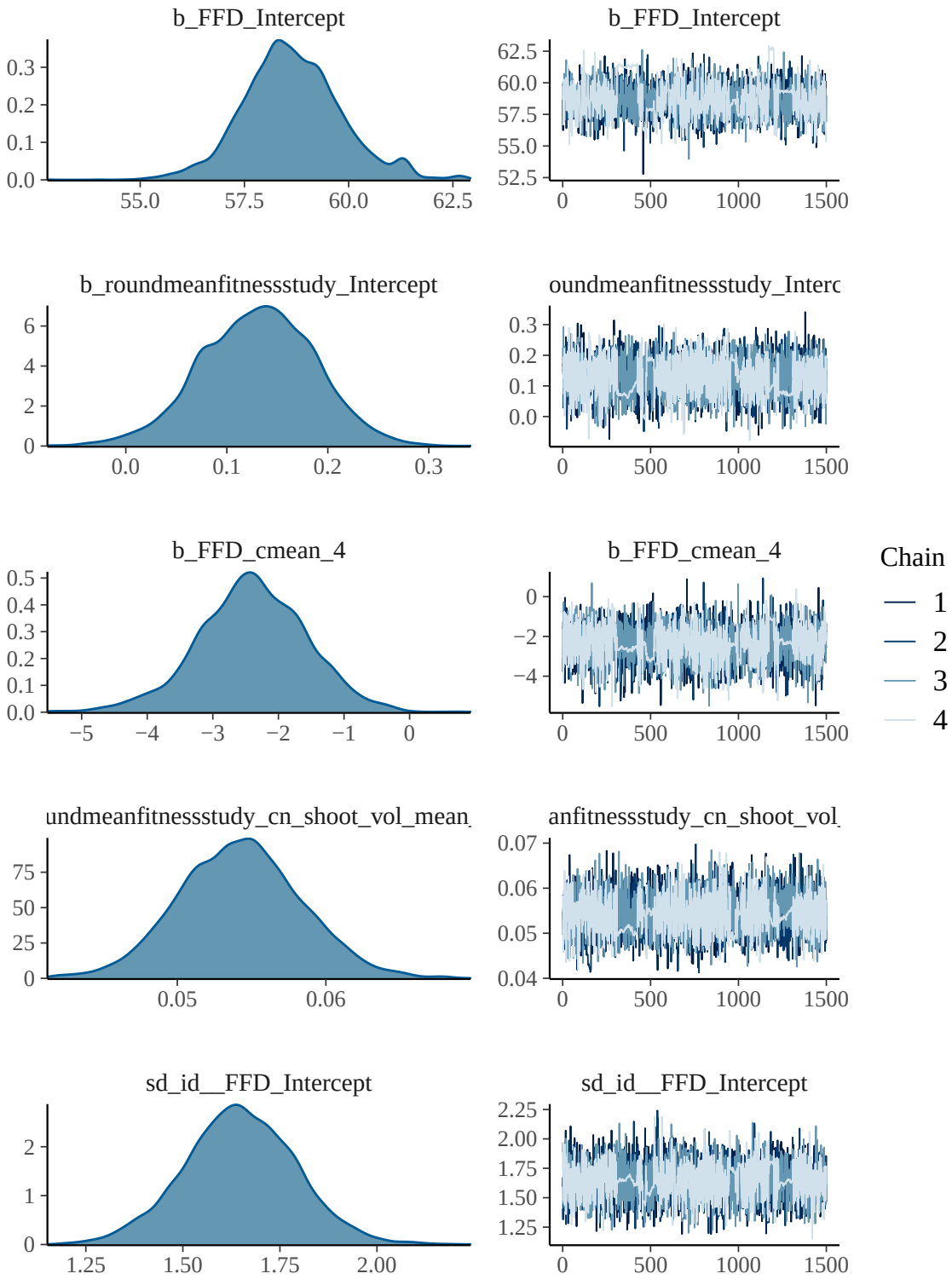
```
summary(bivar4.all.brm.nb)
```

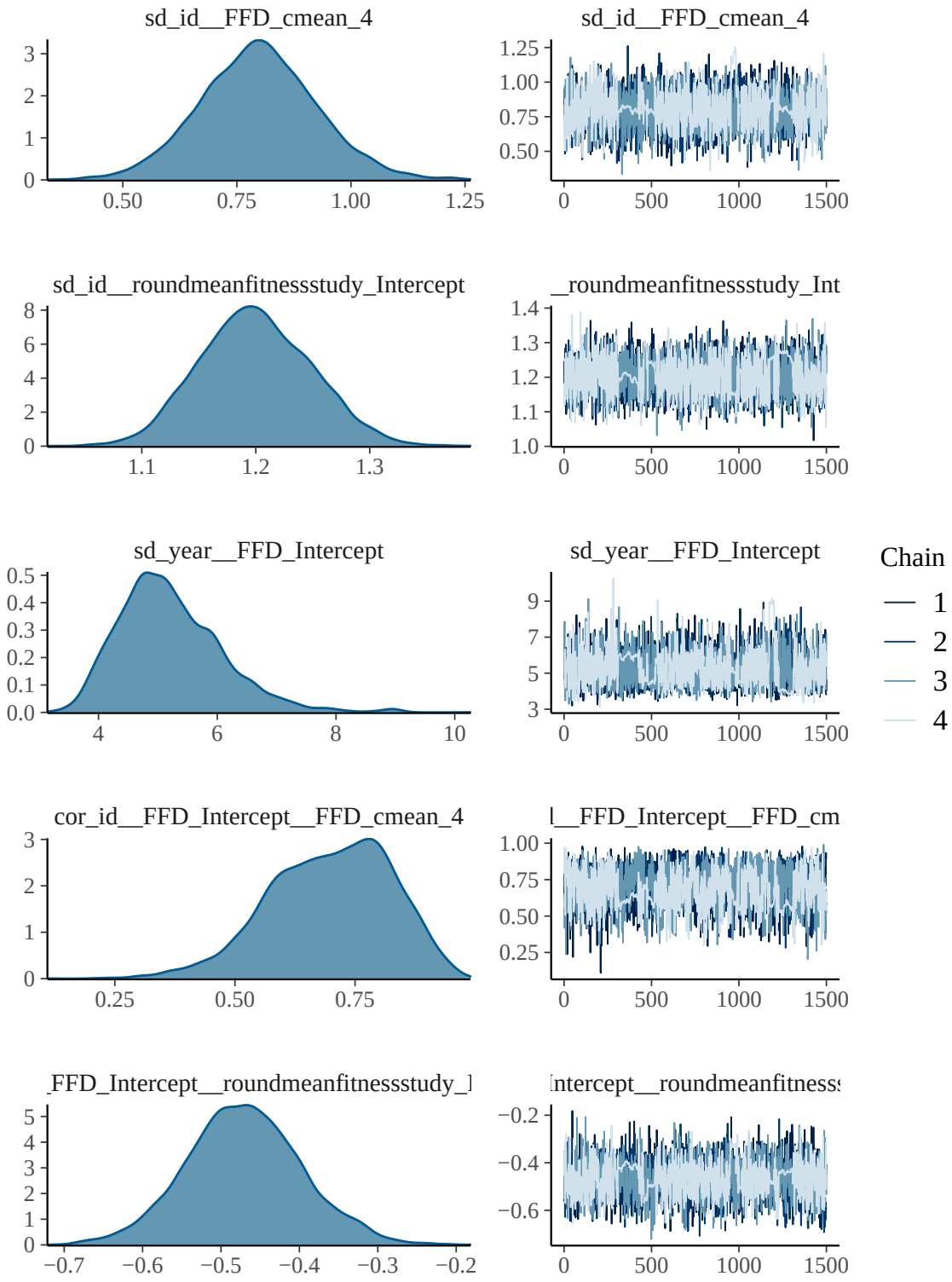
```
##  Family: MV(gaussian, negbinomial)
##   Links: mu = identity; sigma = identity
##          mu = log; shape = identity
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_study) ~ cn_shoot_vol_mean_sqrt + (1 | ID1 | id)
##    Data: datadef (Number of observations: 2432)
## Samples: 4 chains, each with iter = 4000; warmup = 1000; thin = 2;
##          total post-warmup samples = 6000
##
## Group-Level Effects:
## ~id (Number of levels: 791)
##                                                 Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                   1.65      0.15     1.36
## sd(FFD_cmean_4)                                     0.80      0.13     0.54
## sd(roundmeanfitnessstudy_Intercept)                 1.20      0.05     1.11
## cor(FFD_Intercept,FFD_cmean_4)                      0.69      0.13     0.41
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)  -0.47      0.07    -0.61
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)     0.05      0.14    -0.21
##                                                 u-95% CI Rhat Bulk_ESS
## sd(FFD_Intercept)                                   1.94 1.00     3171
## sd(FFD_cmean_4)                                     1.06 1.00     2065
## sd(roundmeanfitnessstudy_Intercept)                 1.30 1.00     3097
## cor(FFD_Intercept,FFD_cmean_4)                      0.90 1.00     1008
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)  -0.31 1.00     1350
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)     0.32 1.00     1454
##                                                 Tail_ESS
## sd(FFD_Intercept)                                   4805
## sd(FFD_cmean_4)                                     4382
## sd(roundmeanfitnessstudy_Intercept)                 4266
```

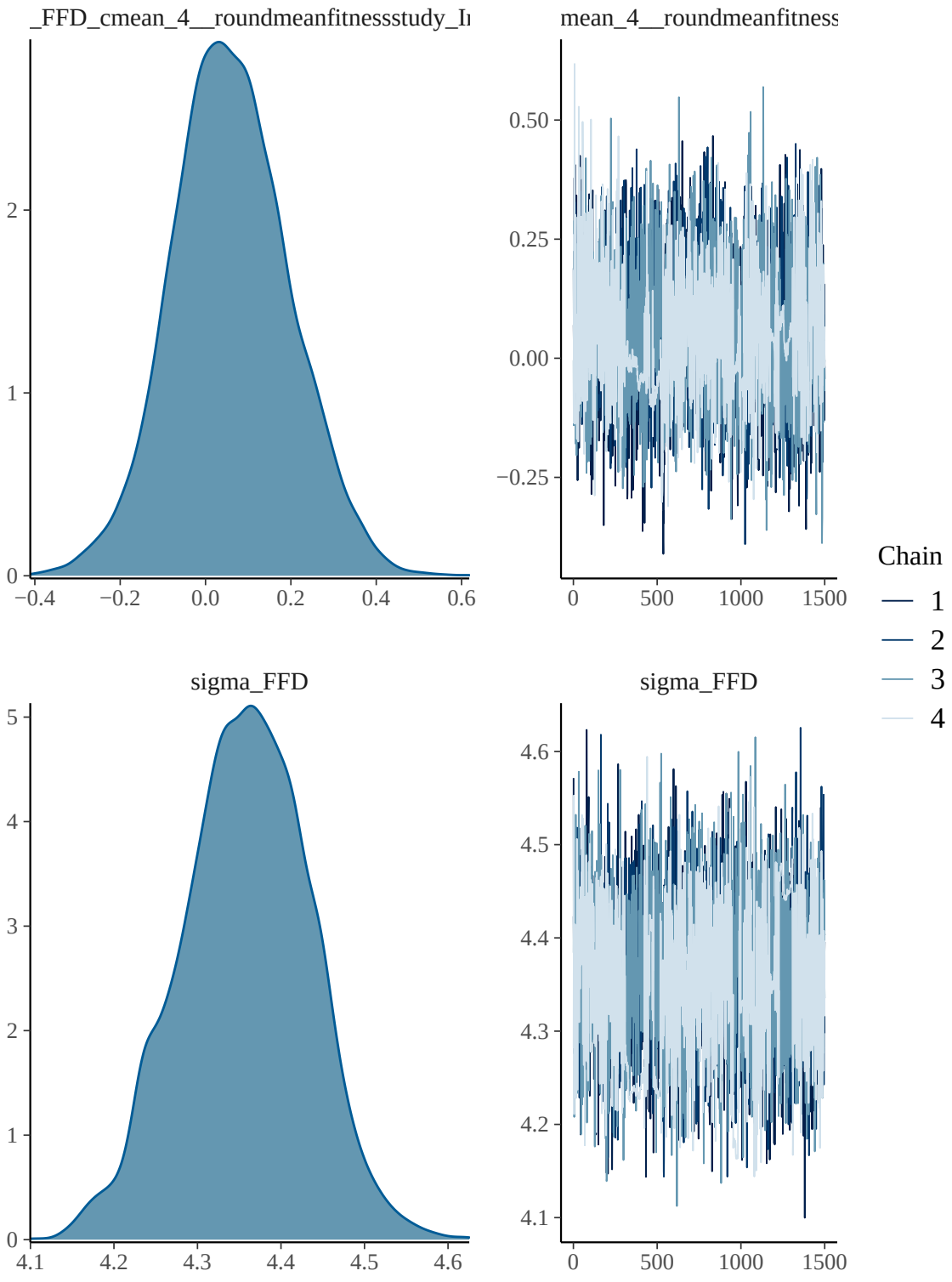
```
## cor(FFD_Intercept,FFD_cmean_4)                          3180
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)     2593
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)       3116
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.22      0.86     3.84     7.16 1.00     3348     4518
##
## Population-Level Effects:
##                                            Estimate Est.Error l-95% CI
## FFD_Intercept                                 58.63      1.11    56.43
## roundmeanfitnessstudy_Intercept                0.13      0.06     0.02
## FFD_cmean_4                                   -2.42      0.83    -4.05
## roundmeanfitnessstudy_cn_shoot_vol_mean_sqrt     0.05      0.00     0.05
##                                            u-95% CI Rhat Bulk_ESS Tail_ESS
## FFD_Intercept                                 60.83 1.00     2350     3538
## roundmeanfitnessstudy_Intercept                0.24 1.00     4767     5088
## FFD_cmean_4                                   -0.80 1.00     2863     4181
## roundmeanfitnessstudy_cn_shoot_vol_mean_sqrt     0.06 1.00     4886     5162
##
## Family Specific Parameters:
##                             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sigma_FFD                       4.36      0.07     4.21     4.50 1.00     3138
## shape_roundmeanfitnessstudy   551.33    168.60   288.20   944.64 1.00     3724
##                             Tail_ESS
## sigma_FFD                       4462
## shape_roundmeanfitnessstudy     4414
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Model evaluation: Compare models

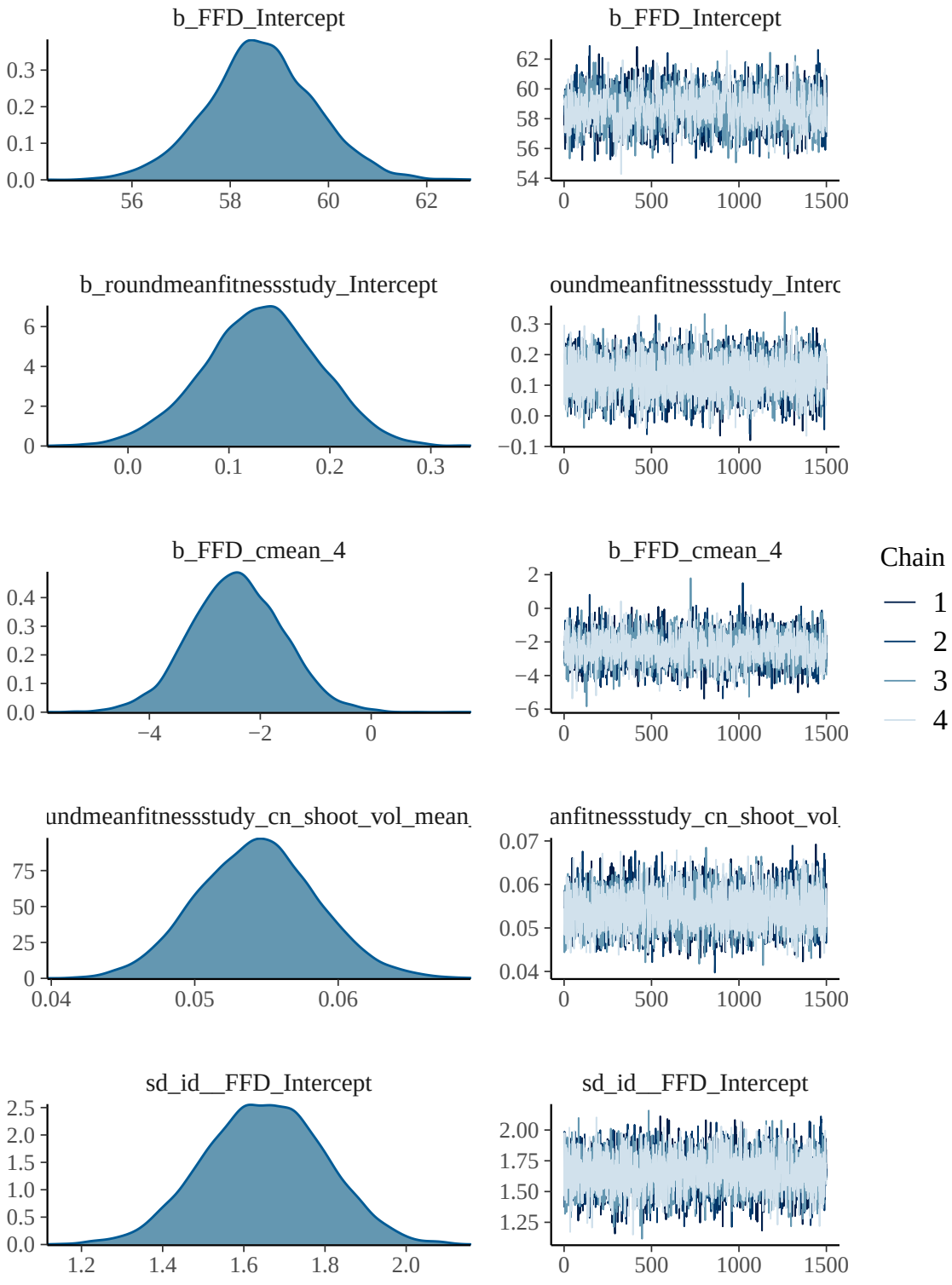
```
plot(bivar4.all.brm.pois)
```

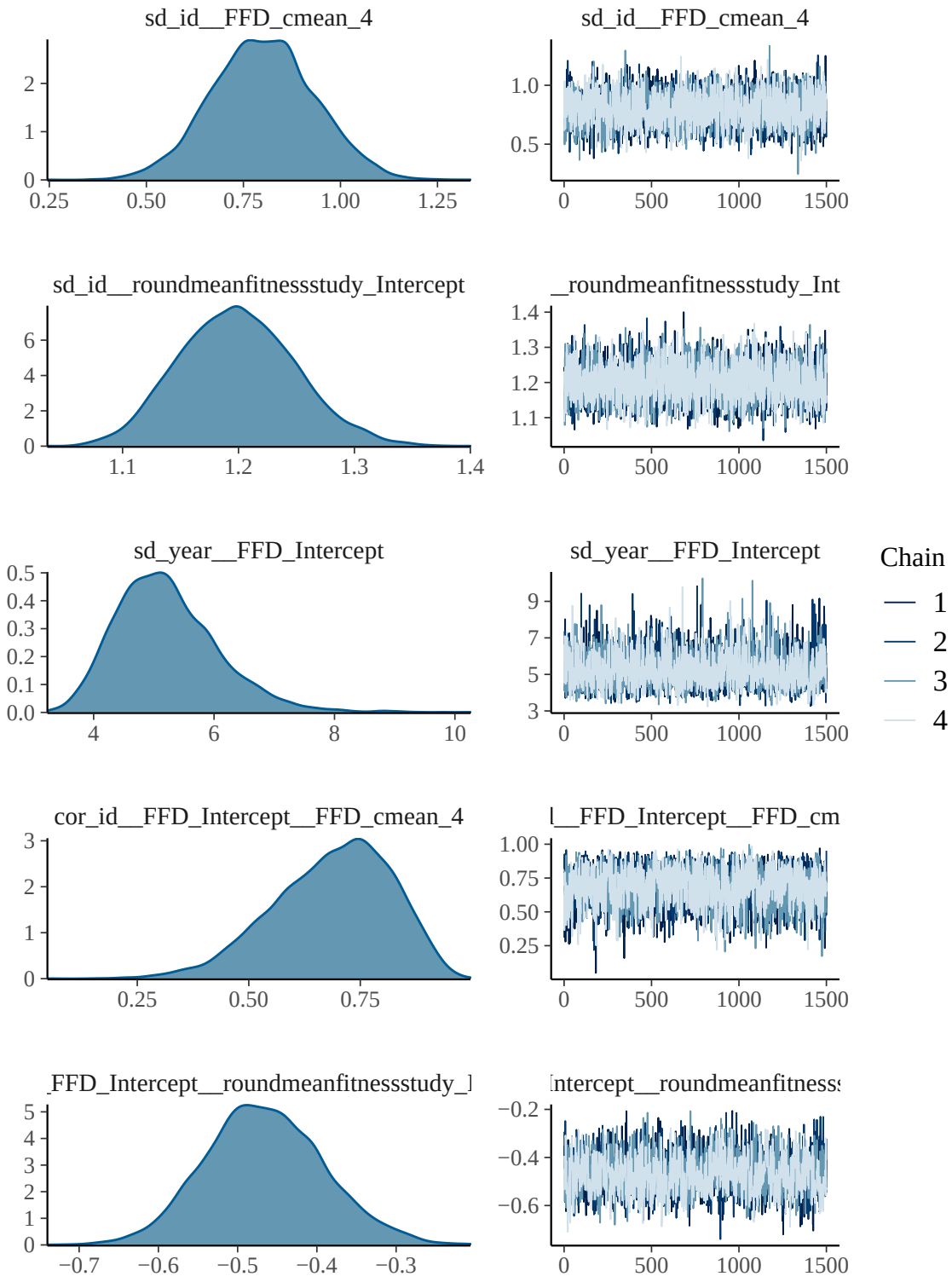

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessstudy_Intercept
oundmeanfitnessstudy_Interc
b_FFD_cmean_4
b_FFD_cmean_4
undmeanfitnessstudy_cn_shoot_vol_mean
anfitnessstudy_cn_shoot_vol
sd_id__FFD_Intercept
sd_id__FFD_Intercept
Chain
1
2
3
4

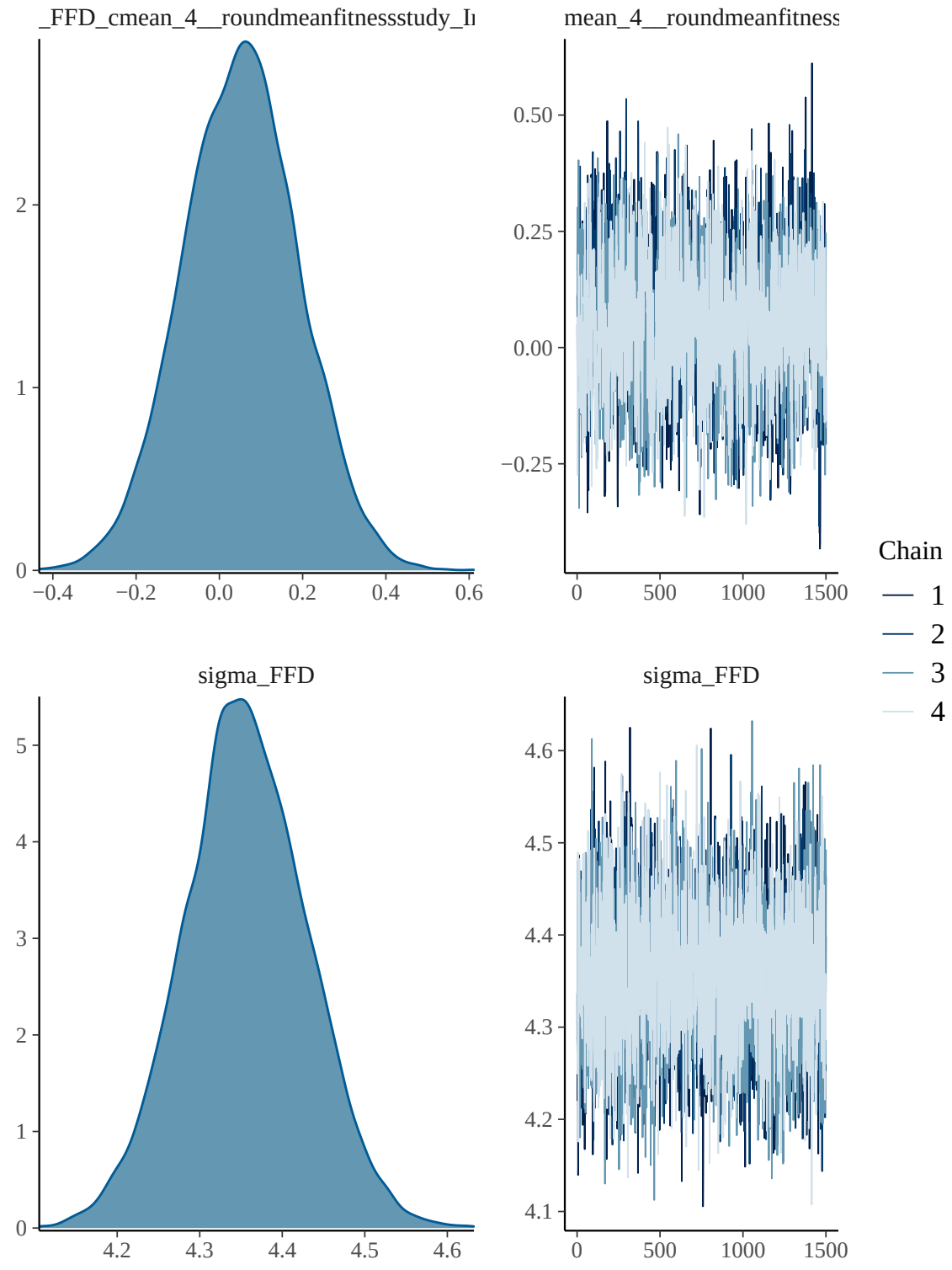

sd_id__FFD_cmean_4
sd_id__FFD_cmean_4
sd_id__roundmeanfitnessstudy_Intercept
__roundmeanfitnessstudy_Int
sd_year__FFD_Intercept
sd_year__FFD_Intercept
Chain
1
2
3
4
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
_FFD_Intercept__roundmeanfitnessstudy_
ntercept__roundmeanfitness




```
plot(bivar4.all.brm.nb)
```

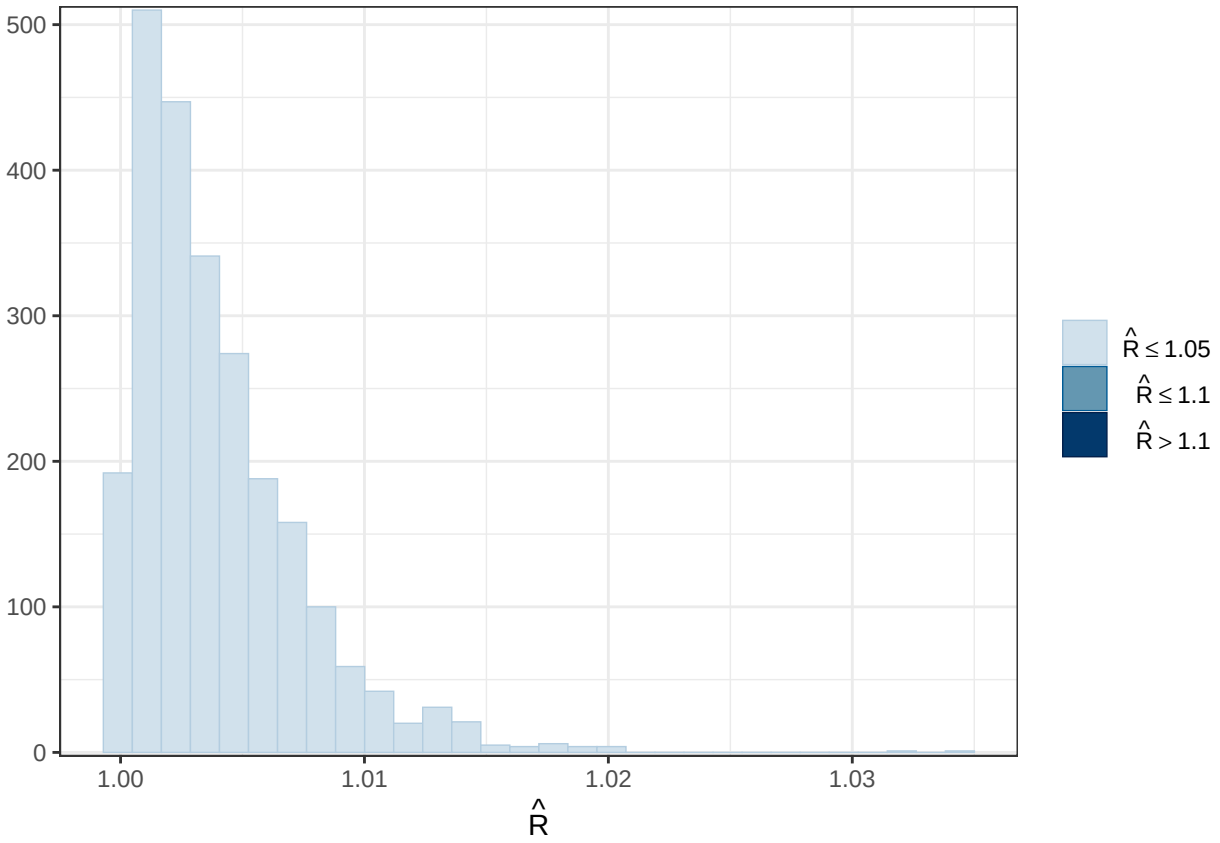

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessstudy_Intercept
oundmeanfitnessstudy_Interc
b_FFD_cmean_4
b_FFD_cmean_4
undmeanfitnessstudy_cn_shoot_vol_mean
anfitnessstudy_cn_shoot_vol
sd_id__FFD_Intercept
sd_id__FFD_Intercept
Chain
1
2
3
4


sd_id__FFD_cmean_4
sd_id__FFD_cmean_4
sd_id__roundmeanfitnessstudy_Intercept
__roundmeanfitnessstudy_Int
sd_year__FFD_Intercept
sd_year__FFD_Intercept
Chain
1
2
3
4
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
_FFD_Intercept__roundmeanfitnessstudy_
ntercept__roundmeanfitness

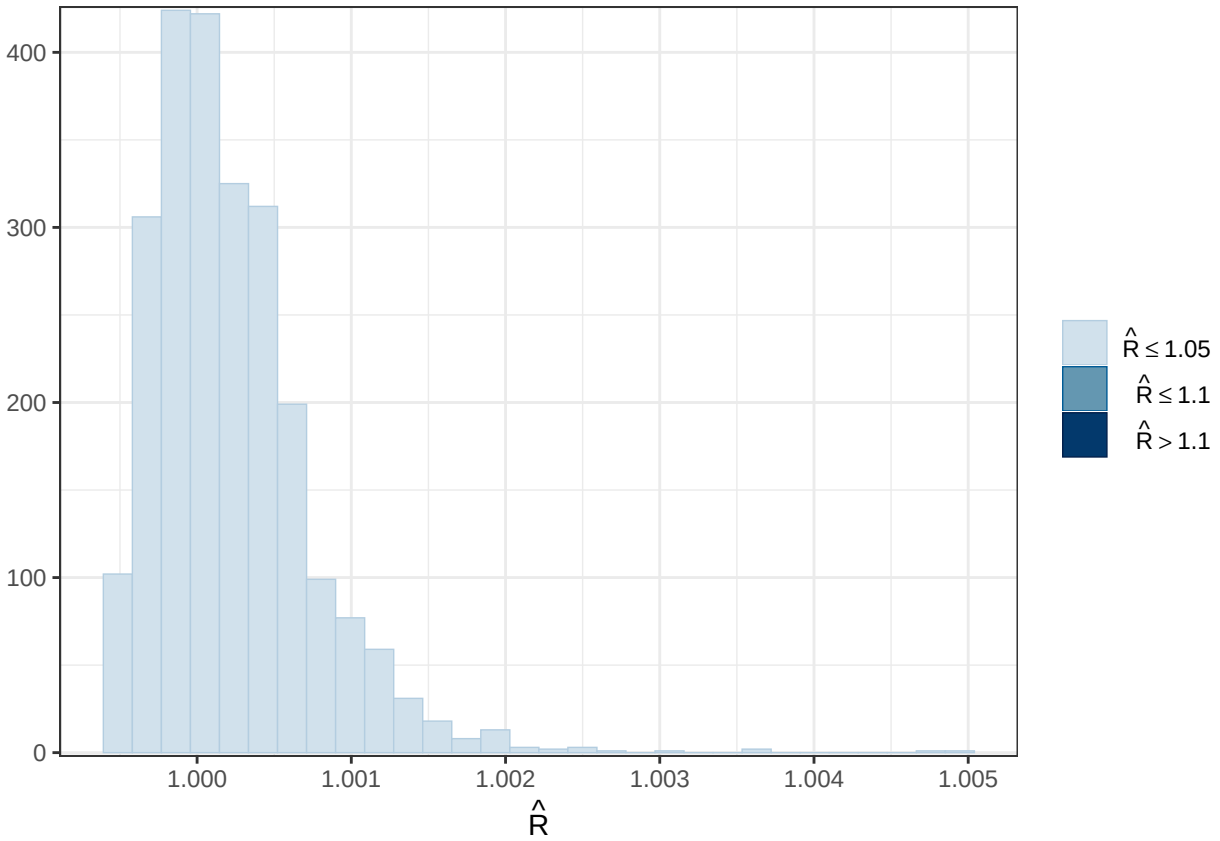

_FFD_cmean_4__roundmeanfitnessstudy_I
mean_4__roundmeanfitness
sigma_FFD
sigma_FFD
Chain
1
2
3
4

Rhat

```
mcmc_rhat_hist(rhat(bivar4.all.brm.pois))+theme_bw()
```

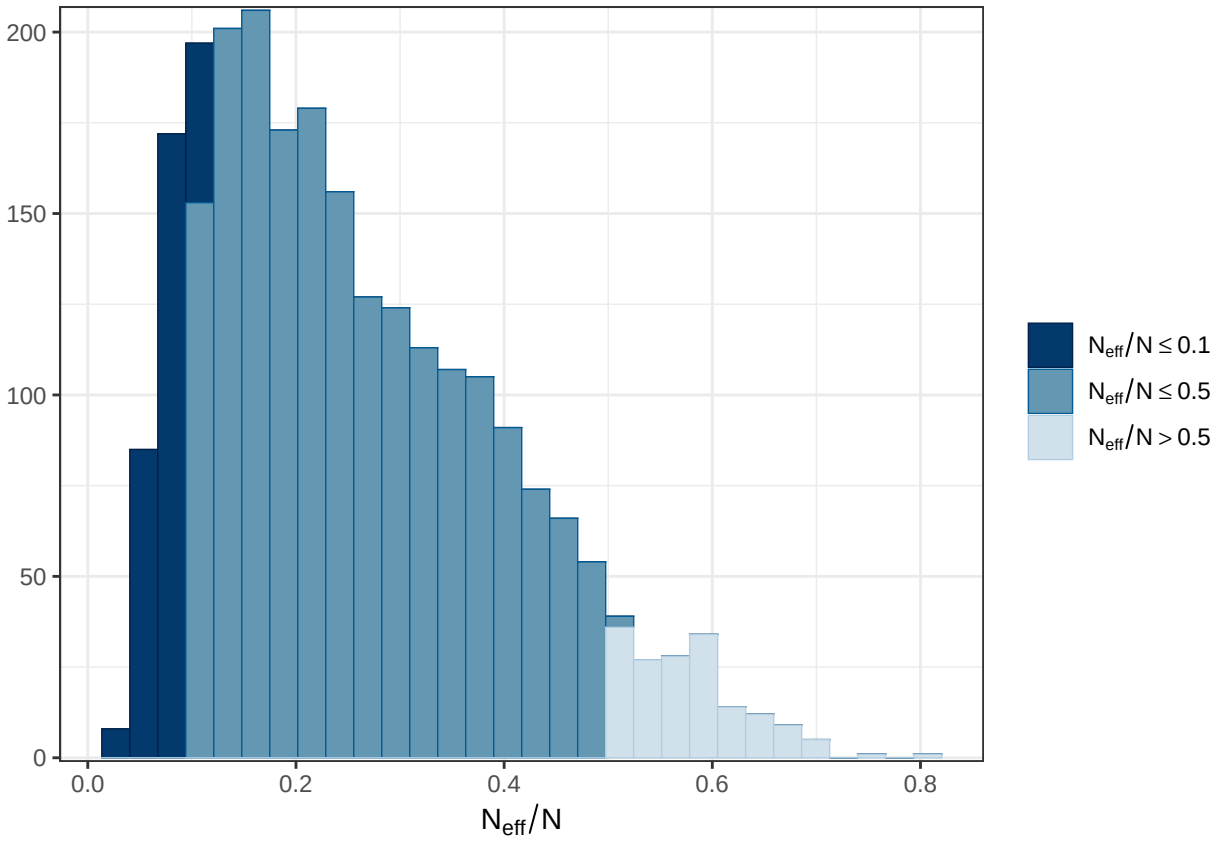



```
mcmc_rhat_hist(rhat(bivar4.all.brm.nb))+theme_bw()
```

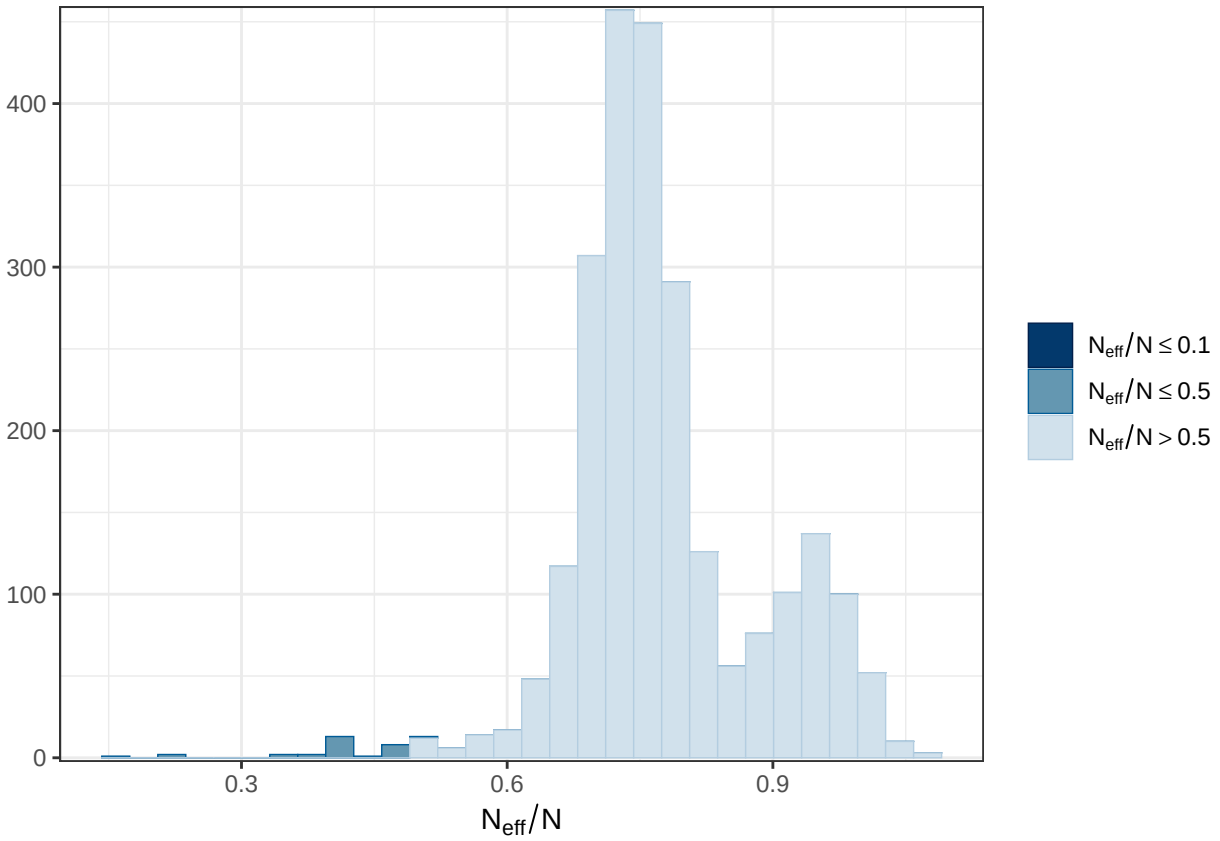


Effective sample size:

```
mcmc_neff_hist(neff_ratio(bivar4.all.brm.pois))+theme_bw()
```




```
mcmc_neff_hist(neff_ratio(bivar4.all.brm.nb))+theme_bw()
```



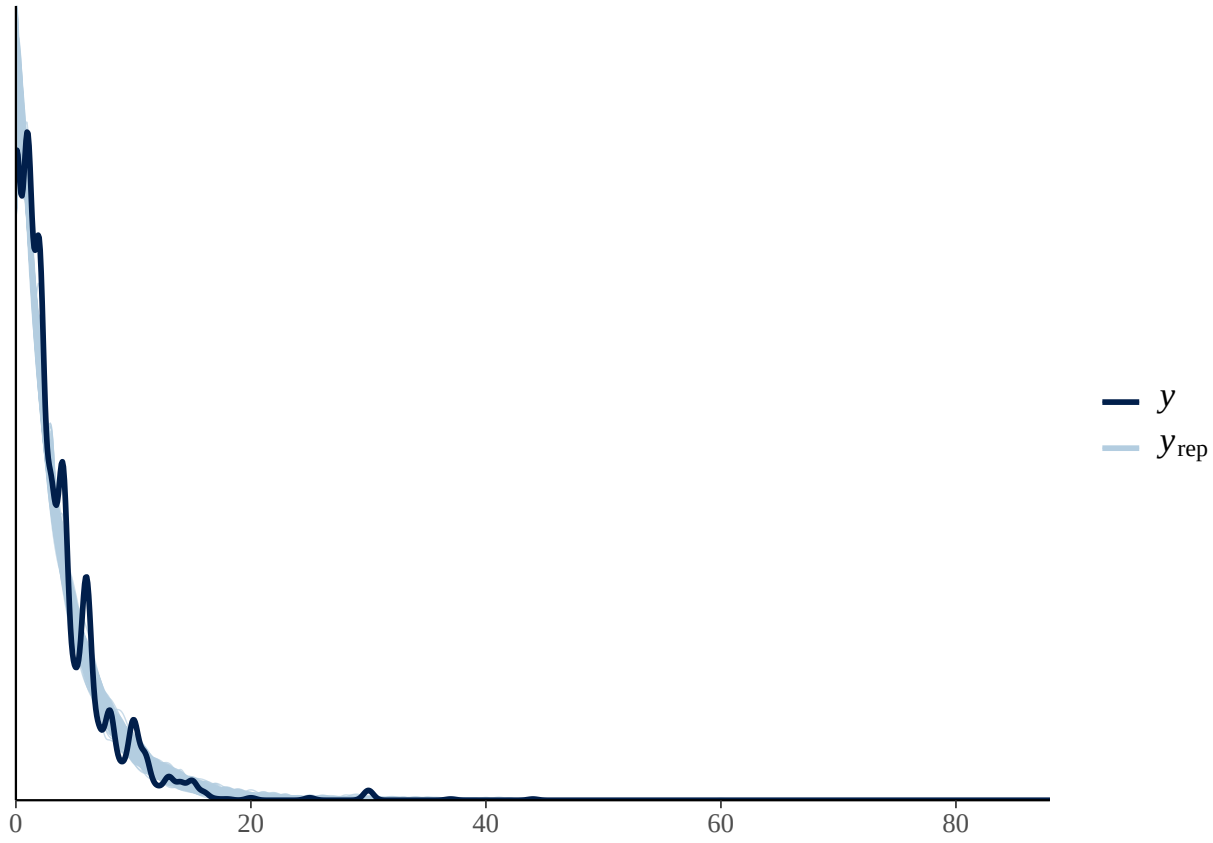
Posterior predictive checks:

```
y3_FFD<-subset(datadef,!is.na(FFD)&
                            !is.na(cn_shoot_vol_mean_sqrt))$FFD
y4_fitness<-round(subset(datadef,!is.na(FFD)&
                            !is.na(cn_shoot_vol_mean_sqrt))$mean_fitness_study)
yrep5_fitness_pois<-posterior_predict(bivar4.all.brm.pois,
                                      draws = 500,resp="roundmeanfitnessstudy")
yrep5_FFD_pois<-posterior_predict(bivar4.all.brm.pois,
                                  draws = 500,resp="FFD")
yrep5_fitness_nb<-posterior_predict(bivar4.all.brm.nb,
                                    draws = 500,resp="roundmeanfitnessstudy")
yrep5_FFD_nb<-posterior_predict(bivar4.all.brm.nb,
                                draws = 500,resp="FFD")
# matrices of draws from the posterior predictive distribution
```

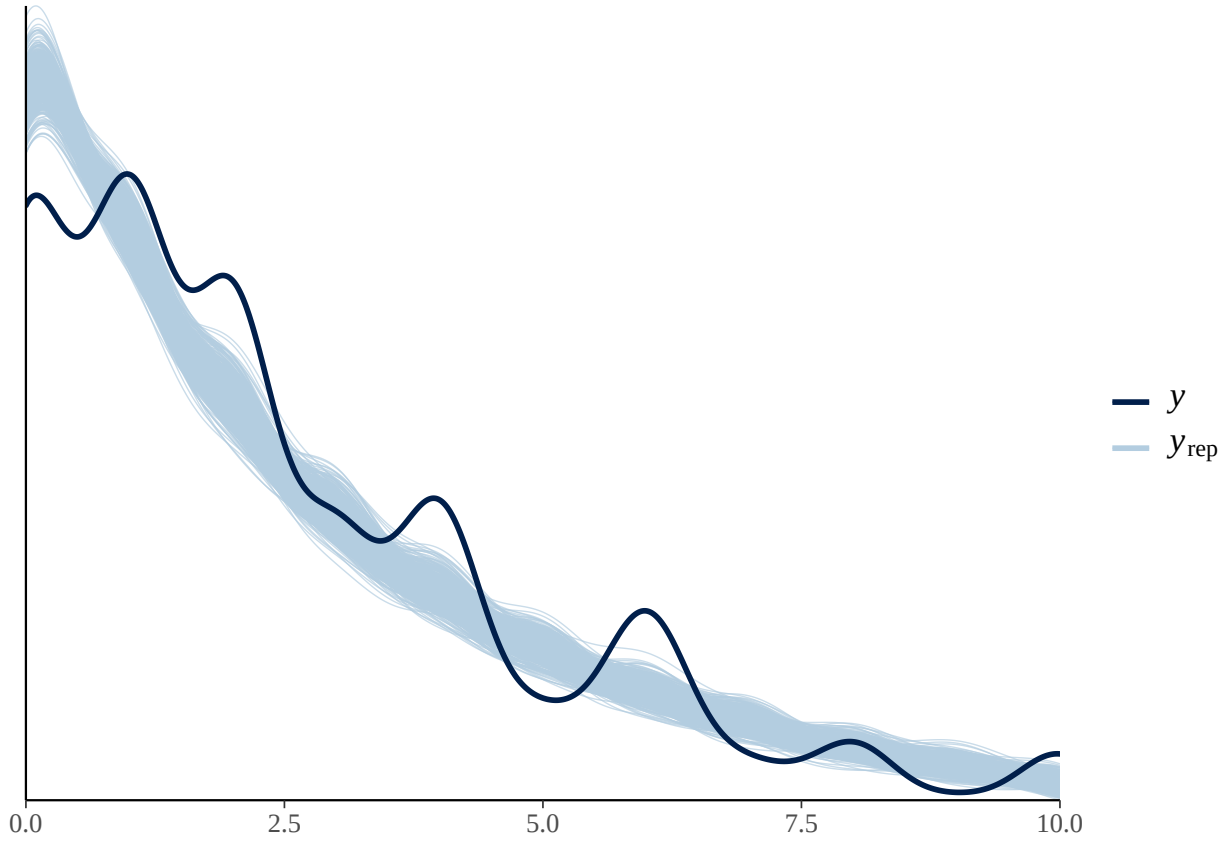
Each row of the matrix is a draw from the posterior predictive distribution, i.e. a vector with one element for each of the data points in y.

Comparison of the distribution of y and the distributions of the simulated datasets in the yrep matrix

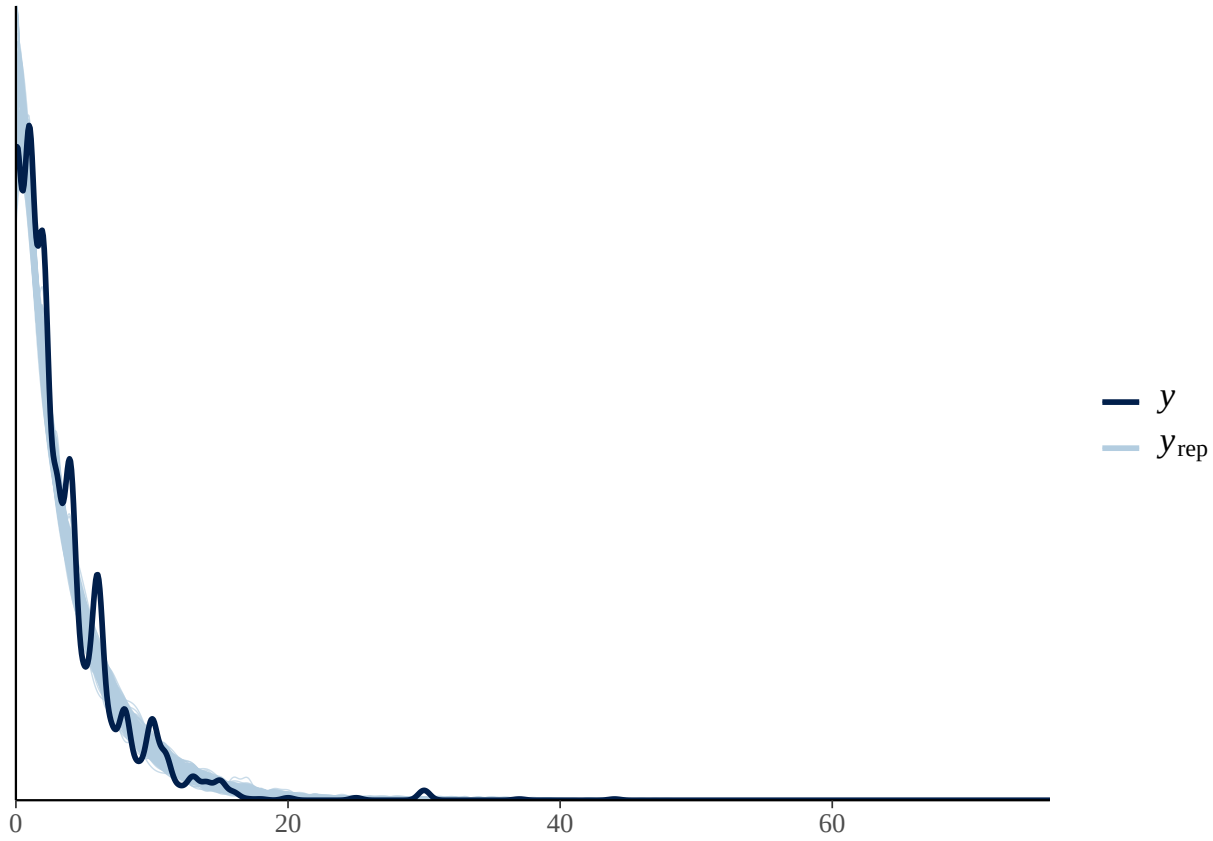
```
ppc_dens_overlay(y4_fitness, yrep5_fitness_pois[1:500,])
```



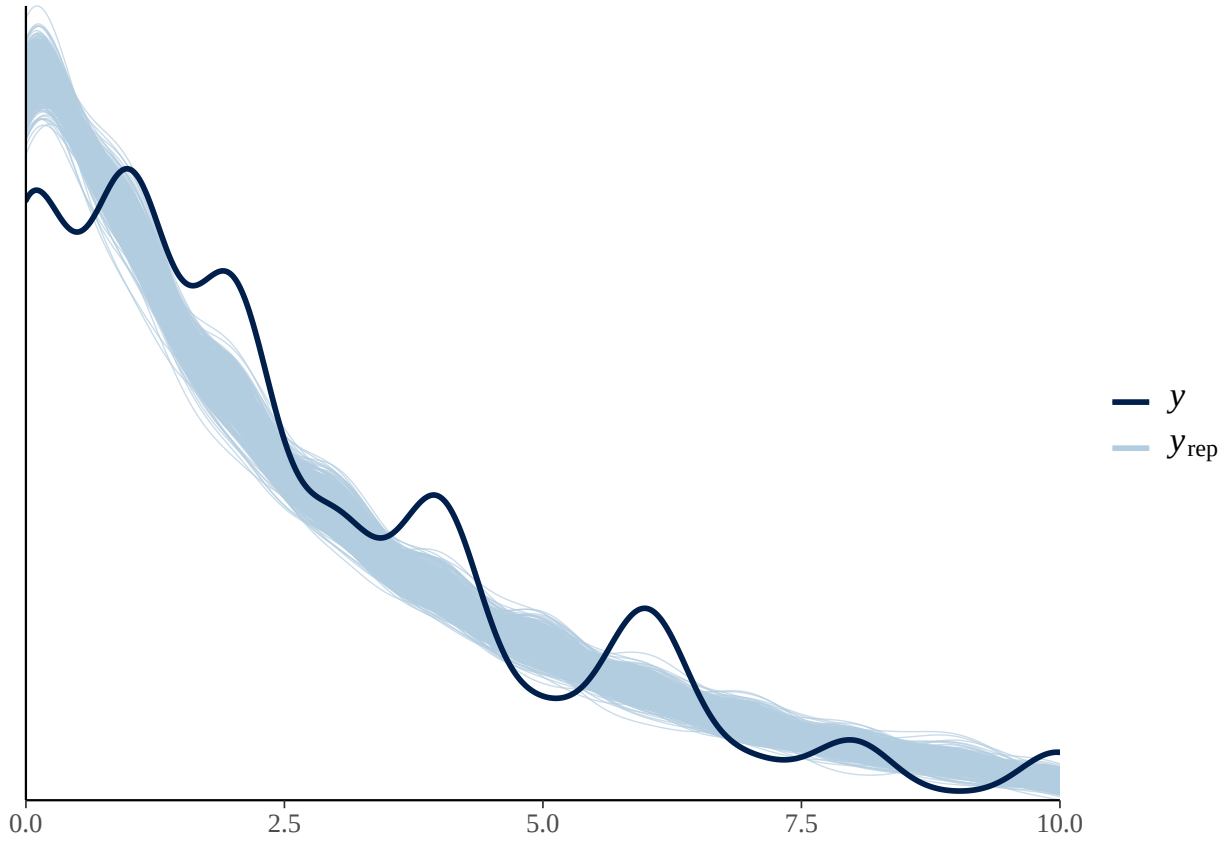

```
ppc_dens_overlay(y4_fitness, yrep5_fitness_pois[1:500,])+xlim(0, 10)
```



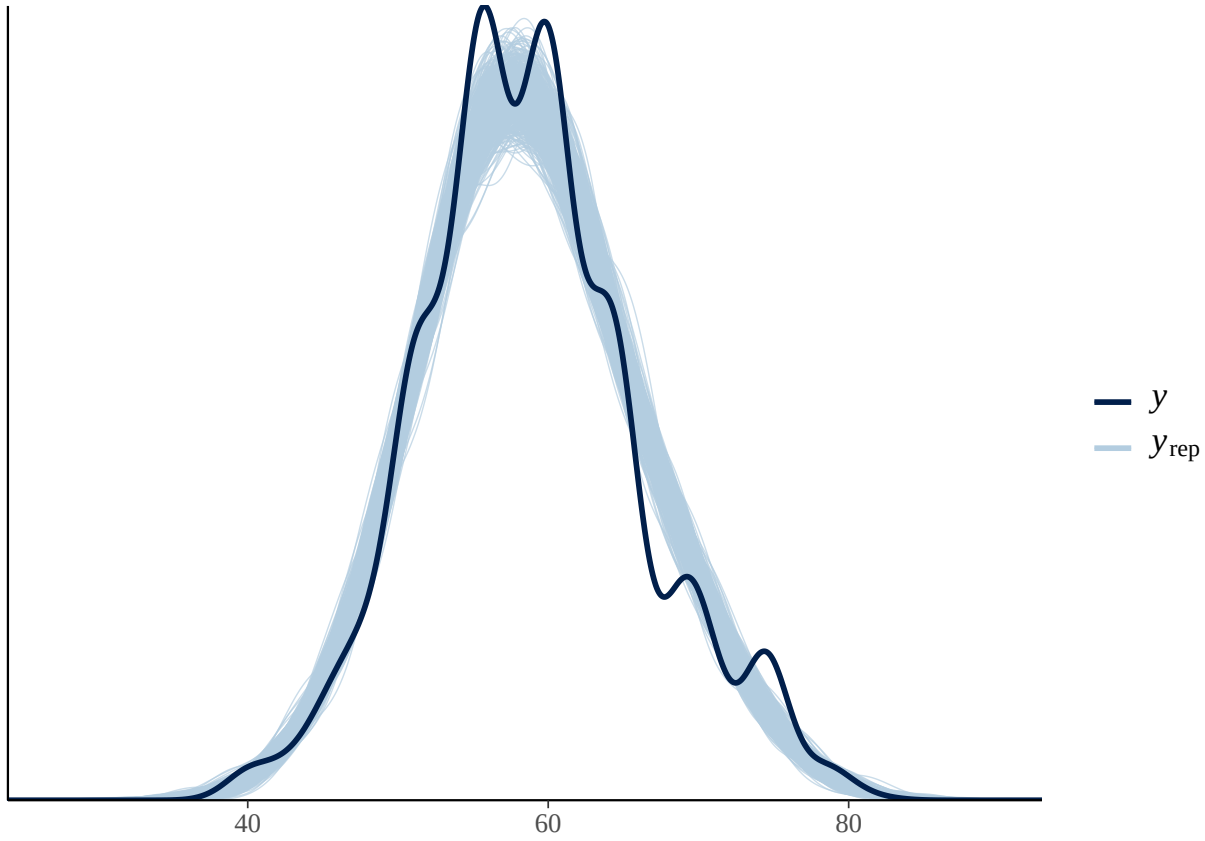

```
ppc_dens_overlay(y4_fitness, yrep5_fitness_nb[1:500,])
```



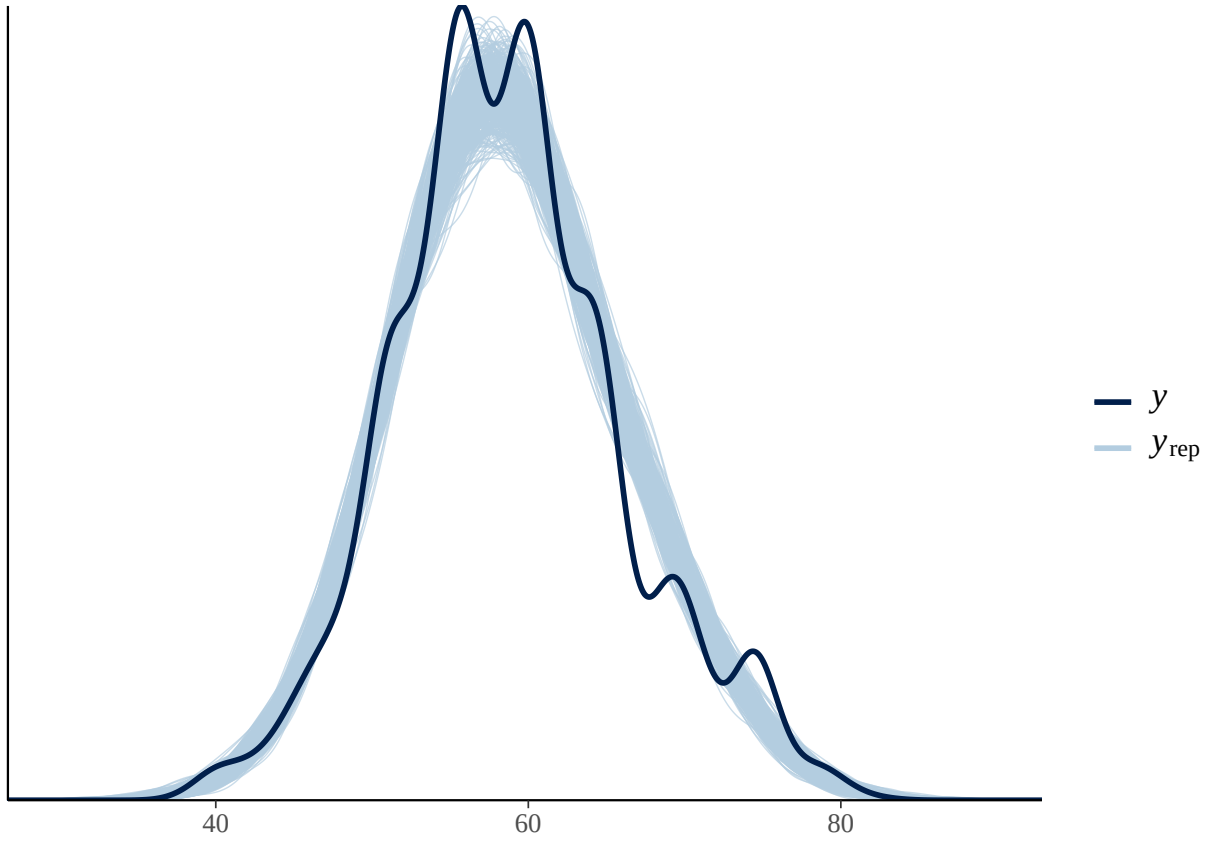

```
ppc_dens_overlay(y4_fitness, yrep5_fitness_nb[1:500,])+xlim(0,10)
```




```
ppc_dens_overlay(y3_FFD, yrep5_FFD_pois[1:500,])
```



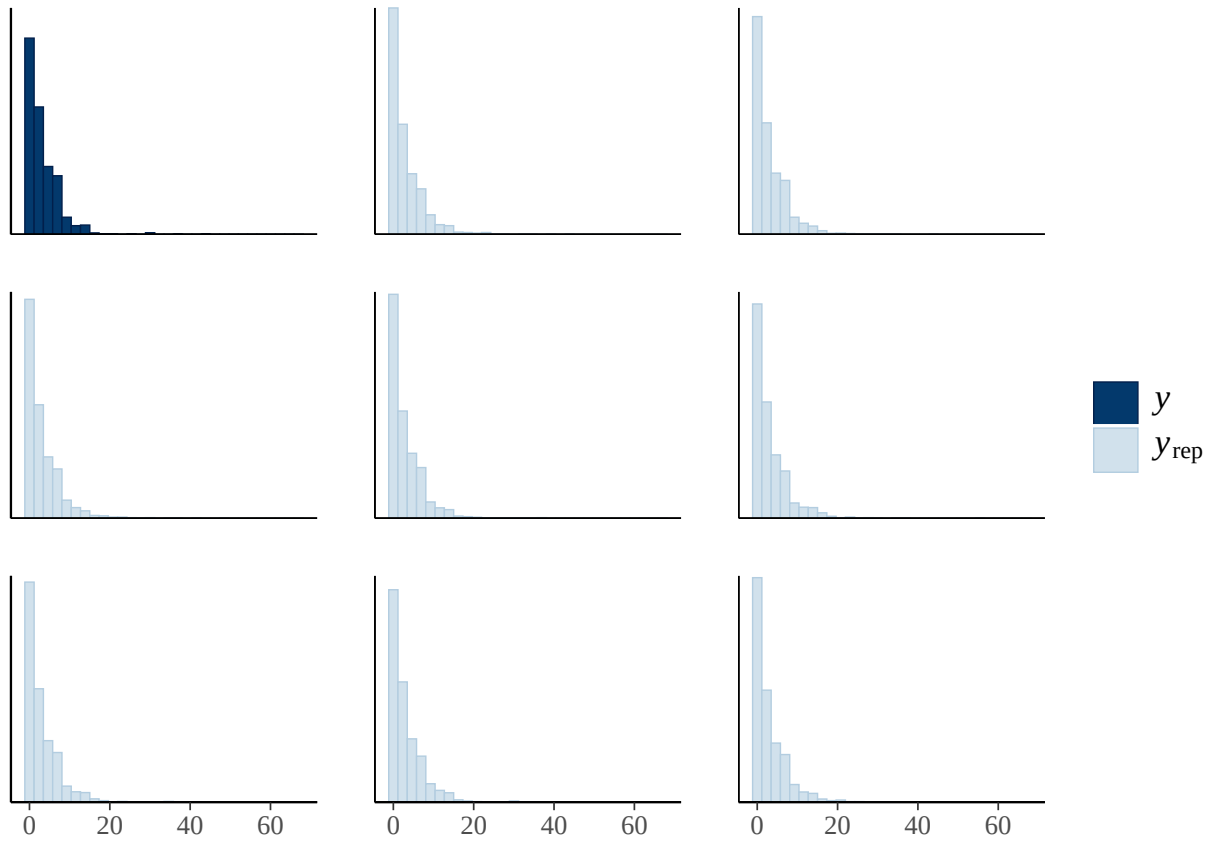

```
ppc_dens_overlay(y3_FFD, yrep5_FFD_nb[1:500,])
```



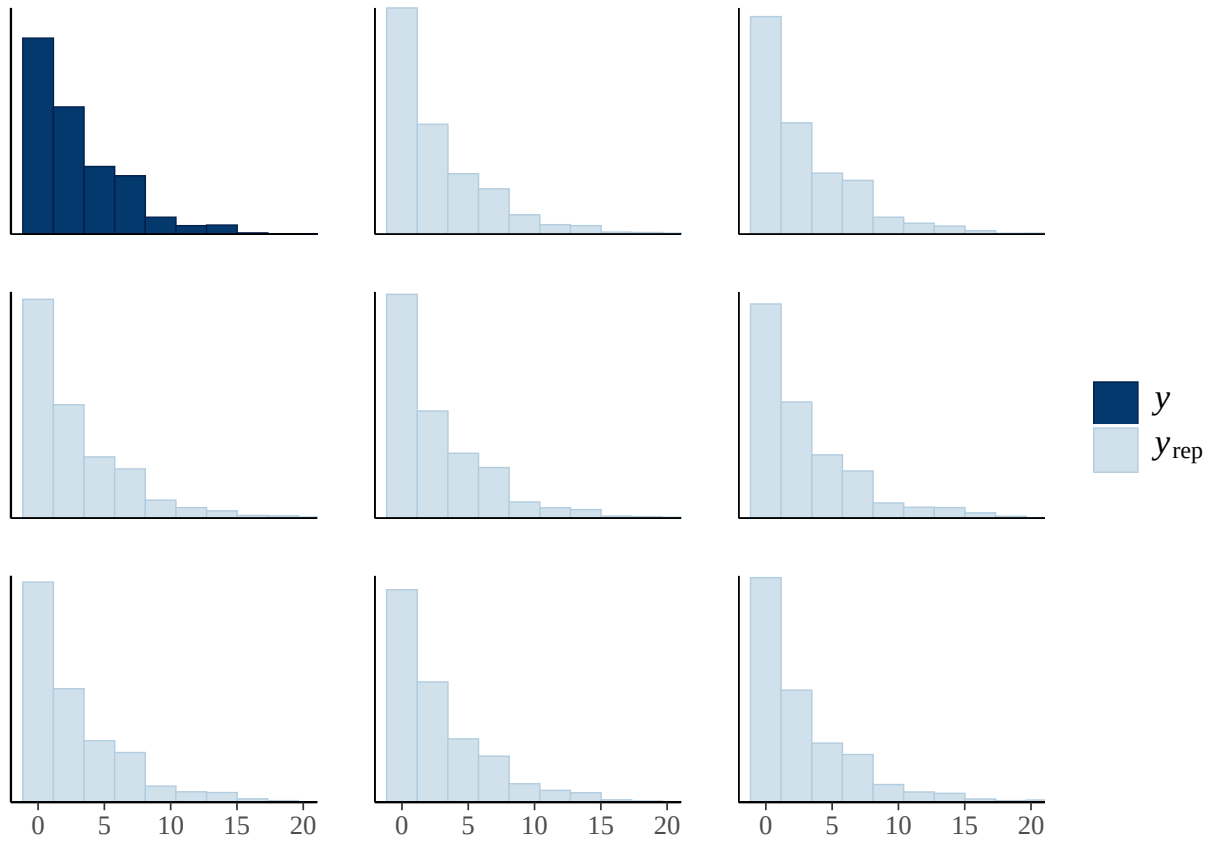
Poisson and negative binomial look pretty similar.

Separate histograms of y and some of the yrep datasets

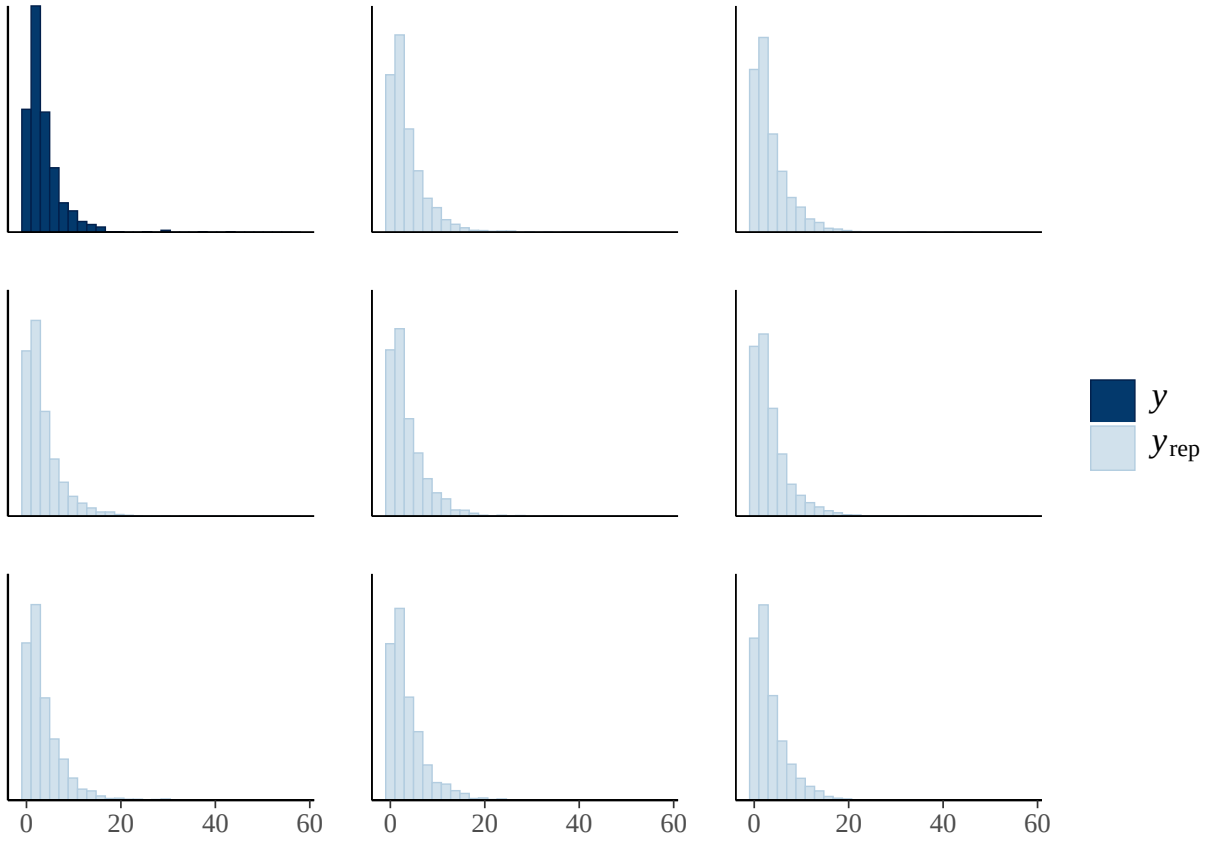
```
ppc_hist(y4_fitness, yrep5_fitness_pois[1:8, ])
```



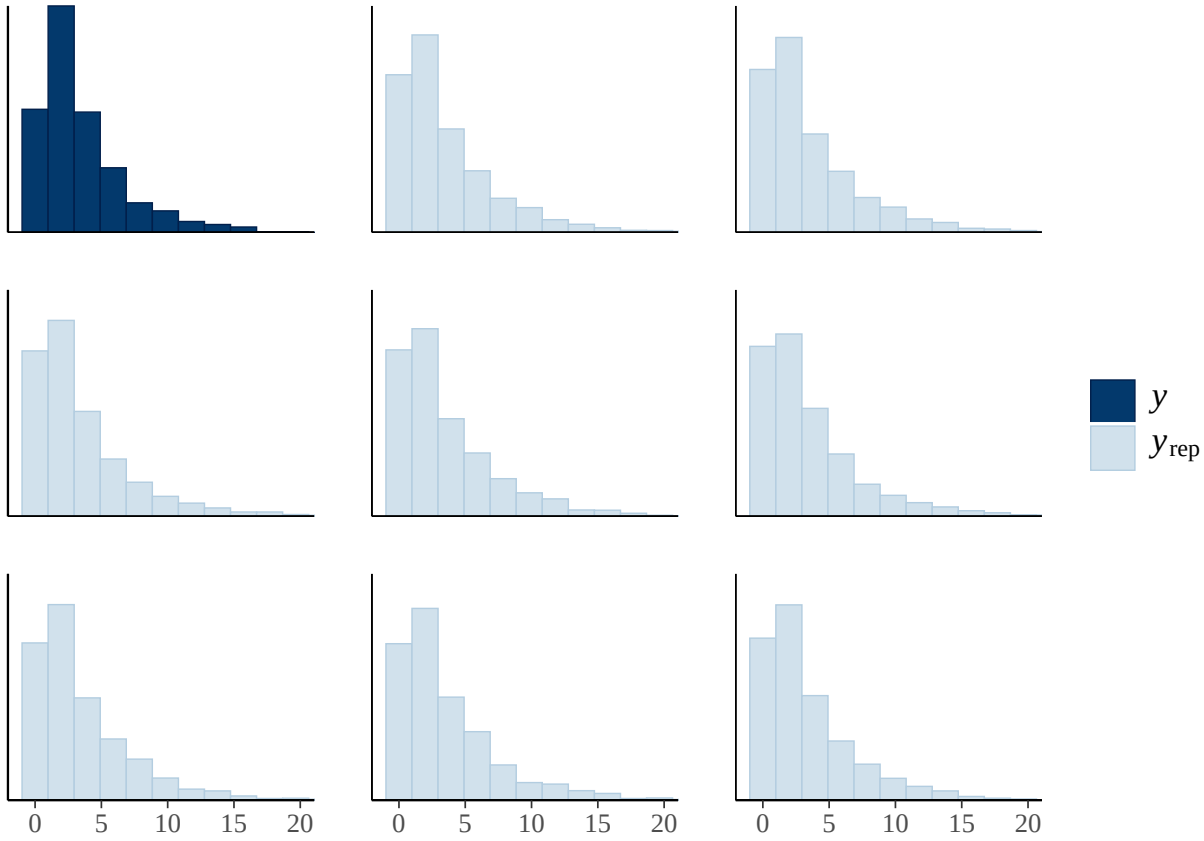

```
ppc_hist(y4_fitness, yrep5_fitness_pois[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```




```
ppc_hist(y4_fitness, yrep5_fitness_nb[1:8, ])
```

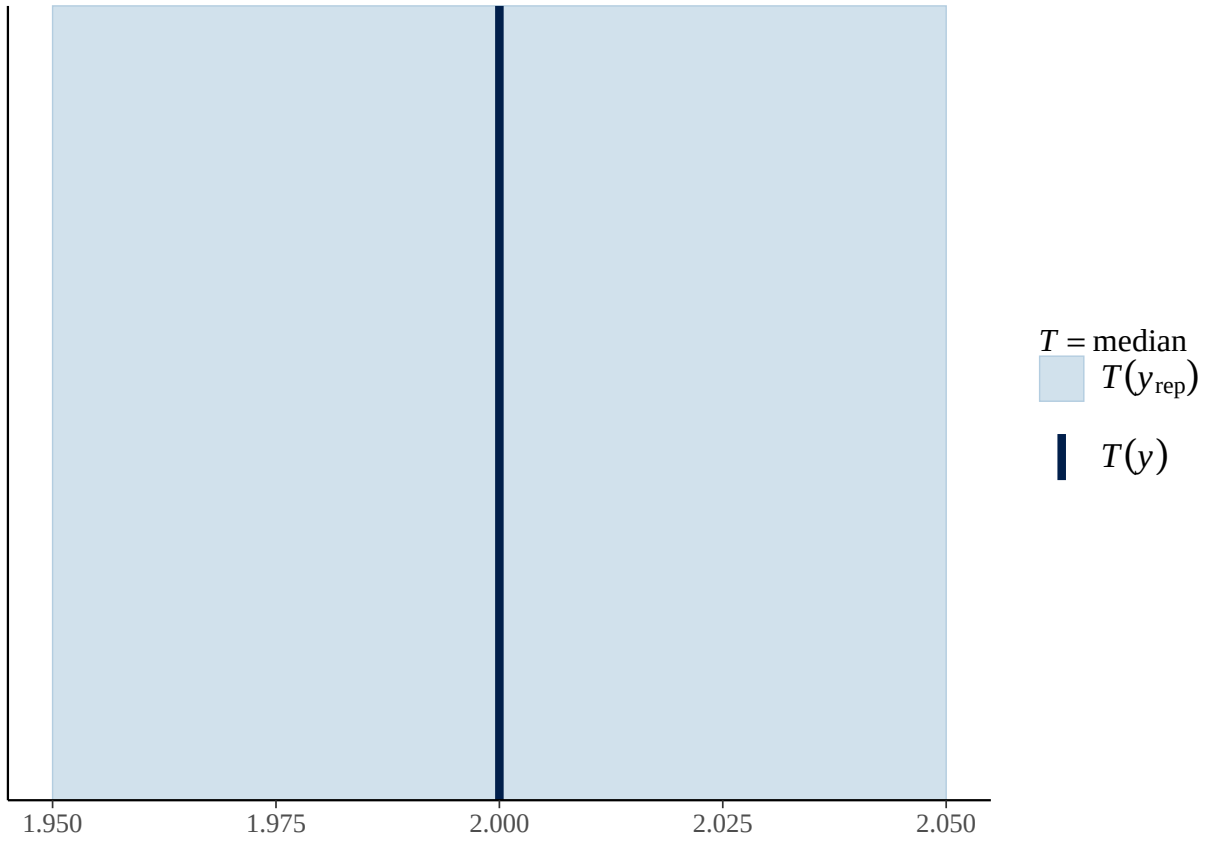



```
ppc_hist(y4_fitness, yrep5_fitness_nb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```

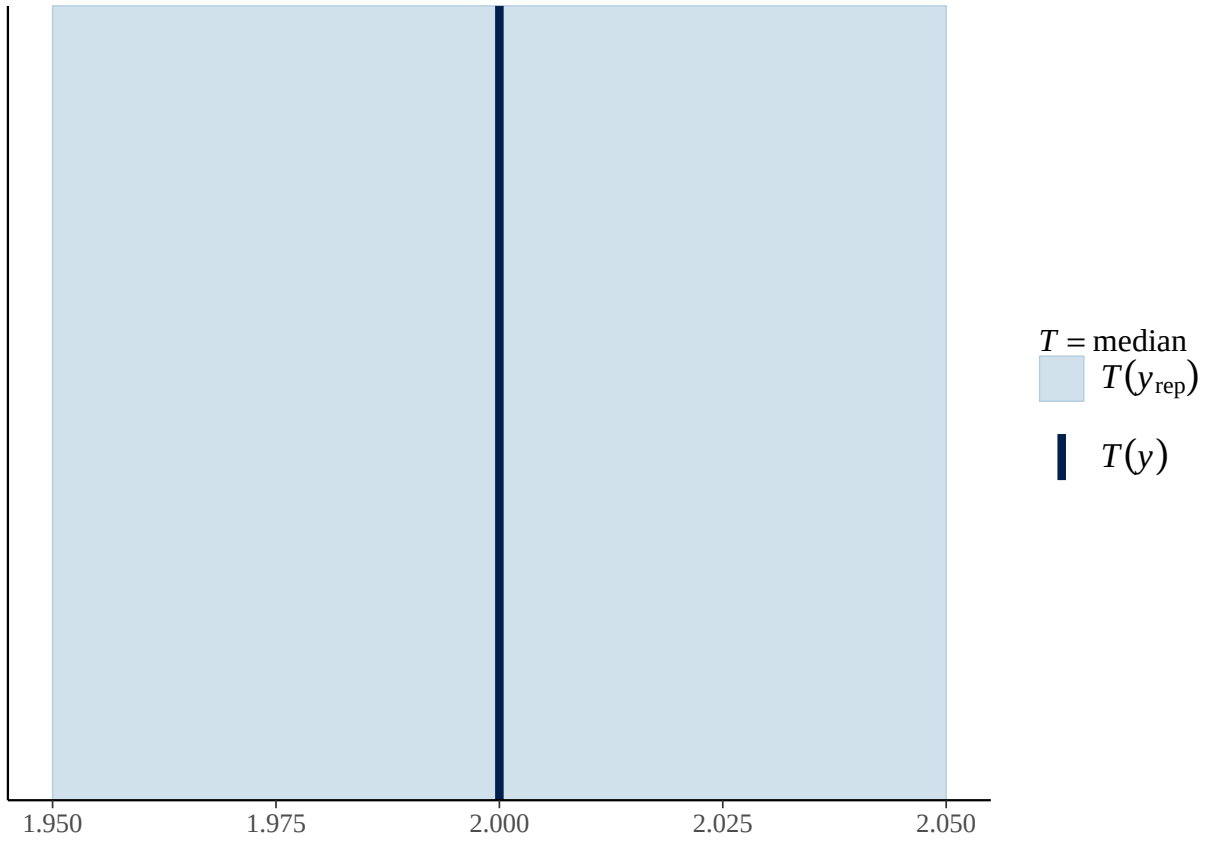


Here, negative binomial looks a bit better for fitness.

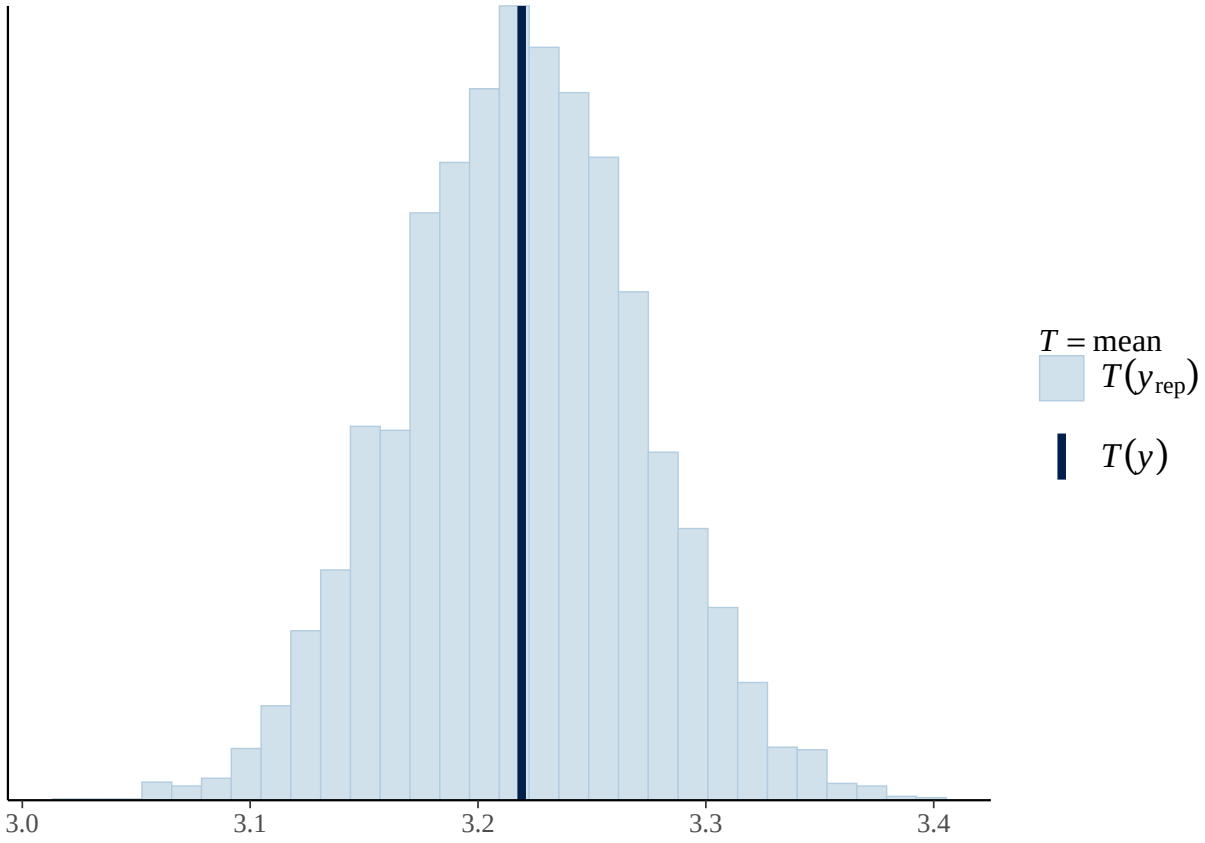
```
ppc_stat(y4_fitness, yrep5_fitness_pois,stat="median")
```



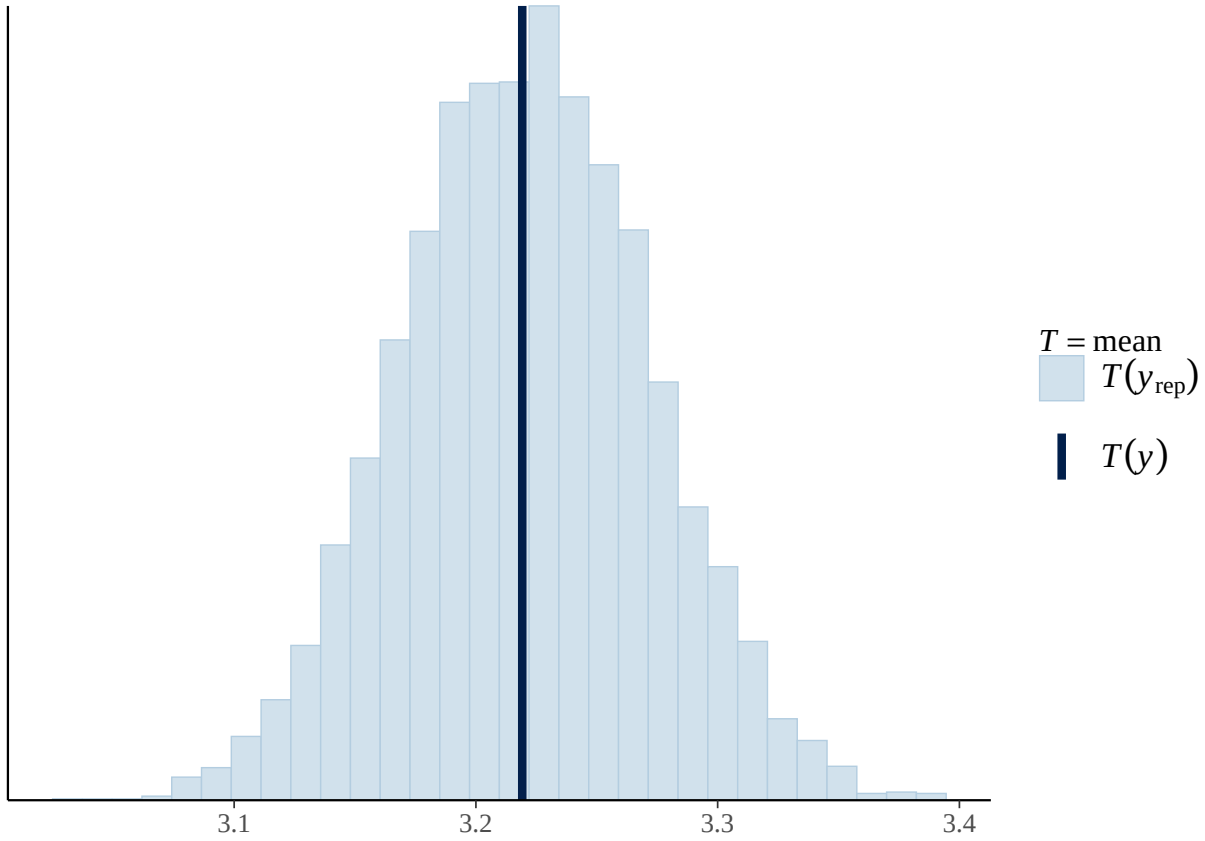

```
ppc_stat(y4_fitness, yrep5_fitness_nb,stat="median")
```



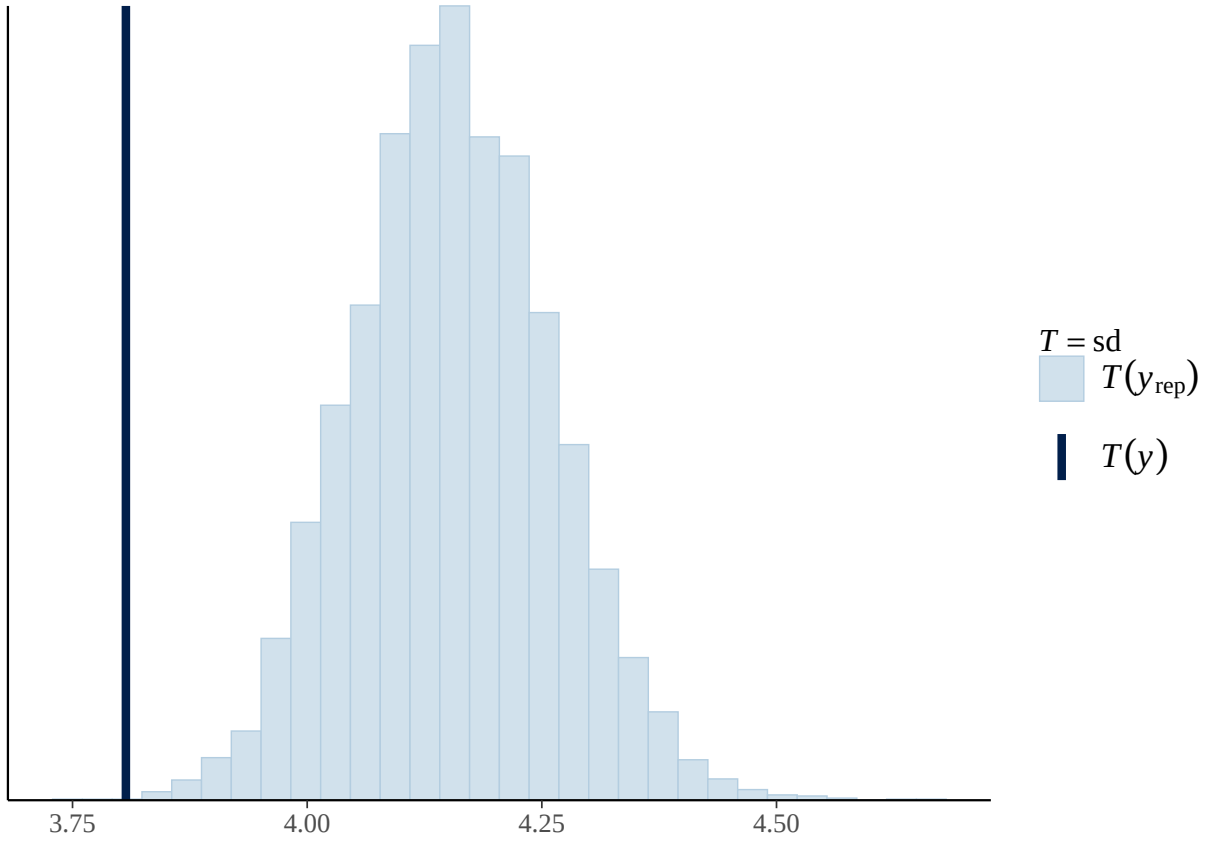

```
ppc_stat(y4_fitness, yrep5_fitness_pois,stat="mean")
```



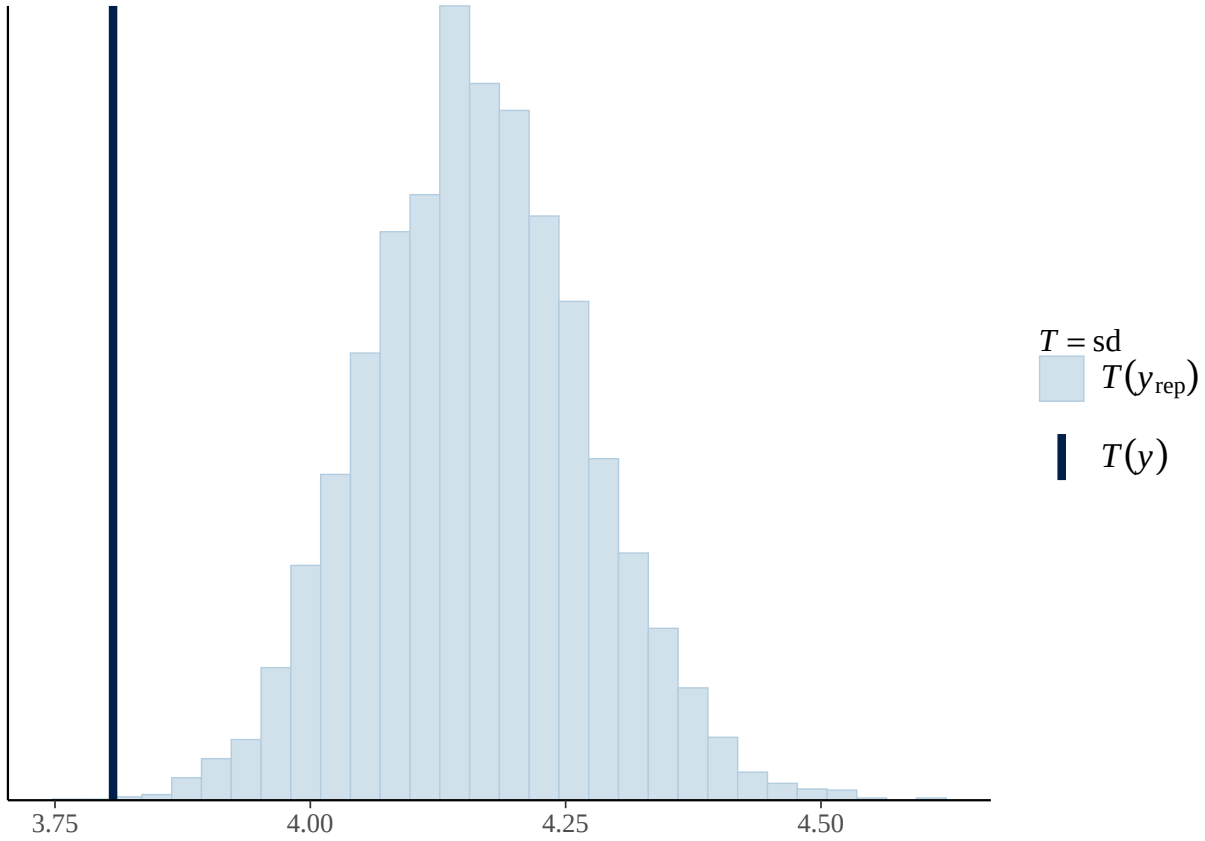

```
ppc_stat(y4_fitness, yrep5_fitness_nb,stat="mean")
```



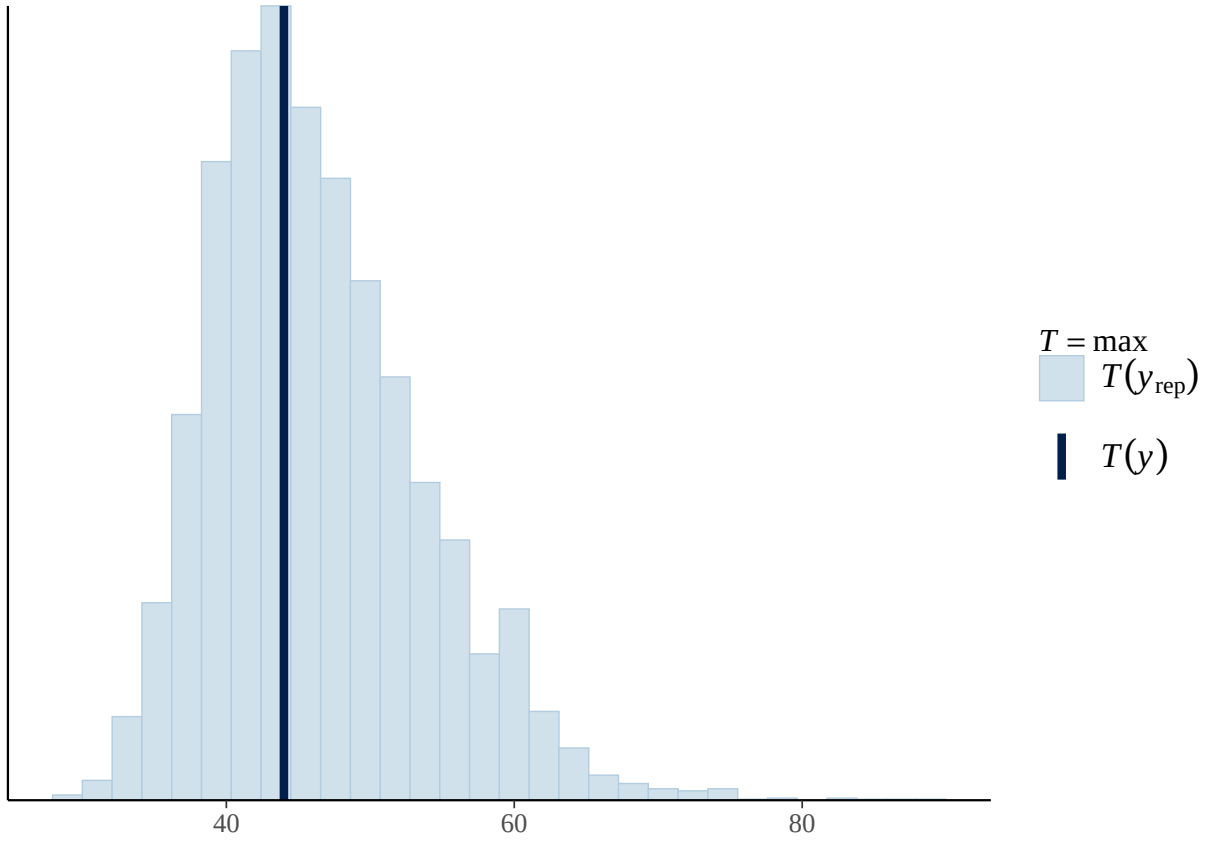

```
ppc_stat(y4_fitness, yrep5_fitness_pois,stat="sd")
```



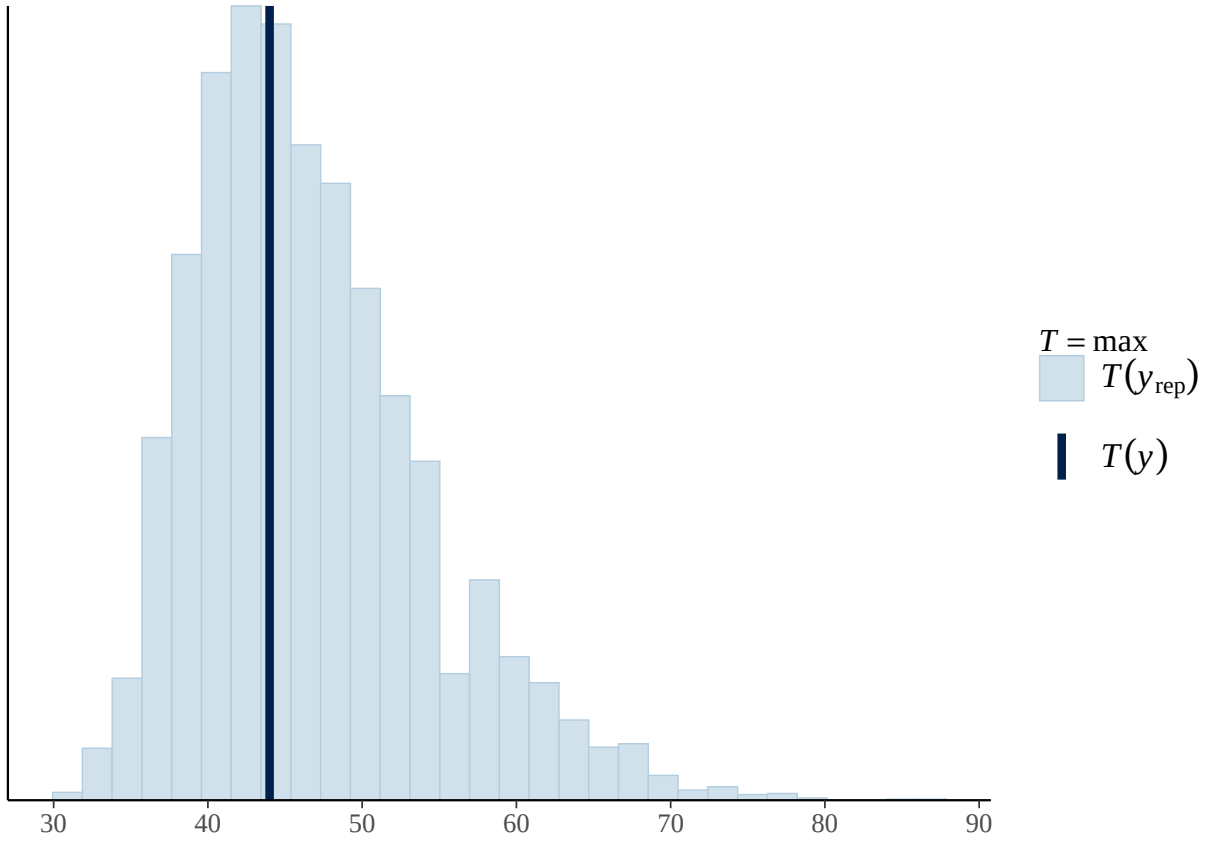

```
ppc_stat(y4_fitness, yrep5_fitness_nb,stat="sd")
```




```
ppc_stat(y4_fitness, yrep5_fitness_pois,stat="max")
```

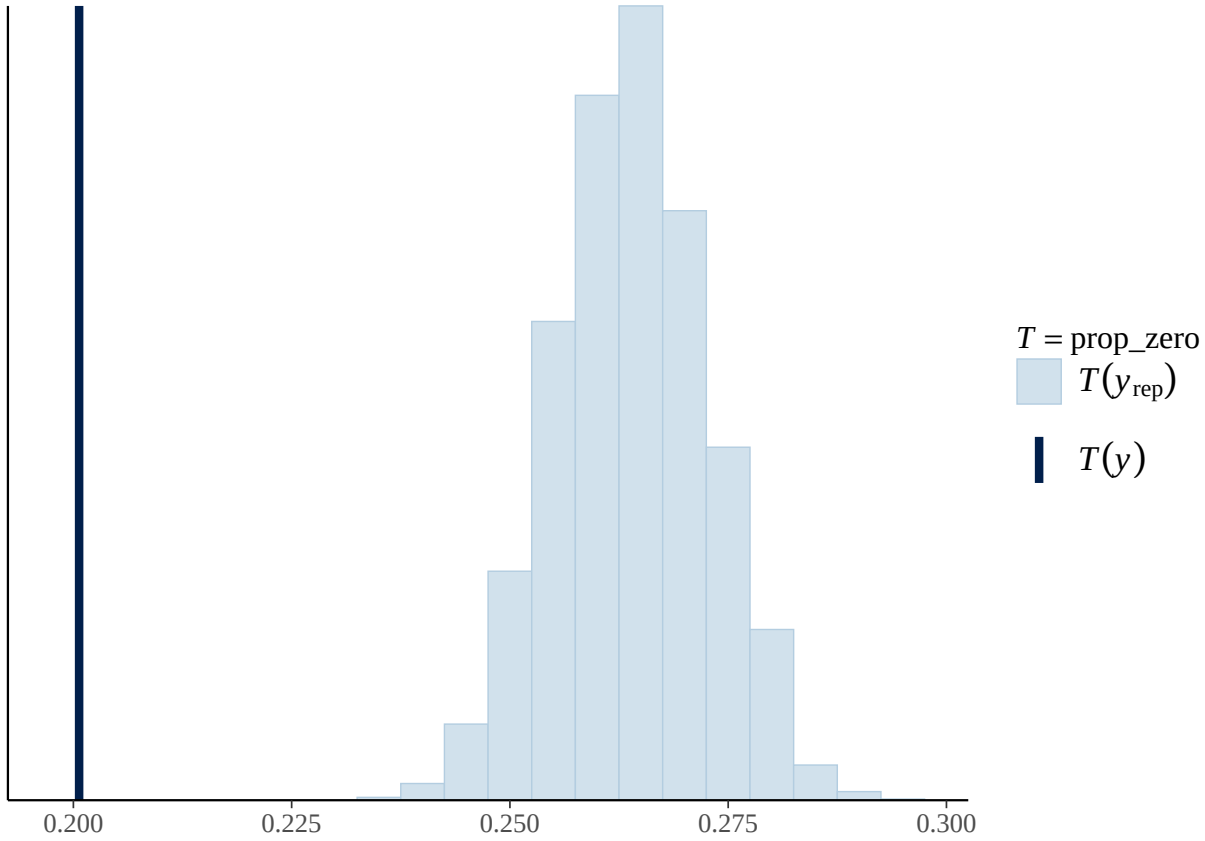



```
ppc_stat(y4_fitness, yrep5_fitness_nb,stat="max")
```

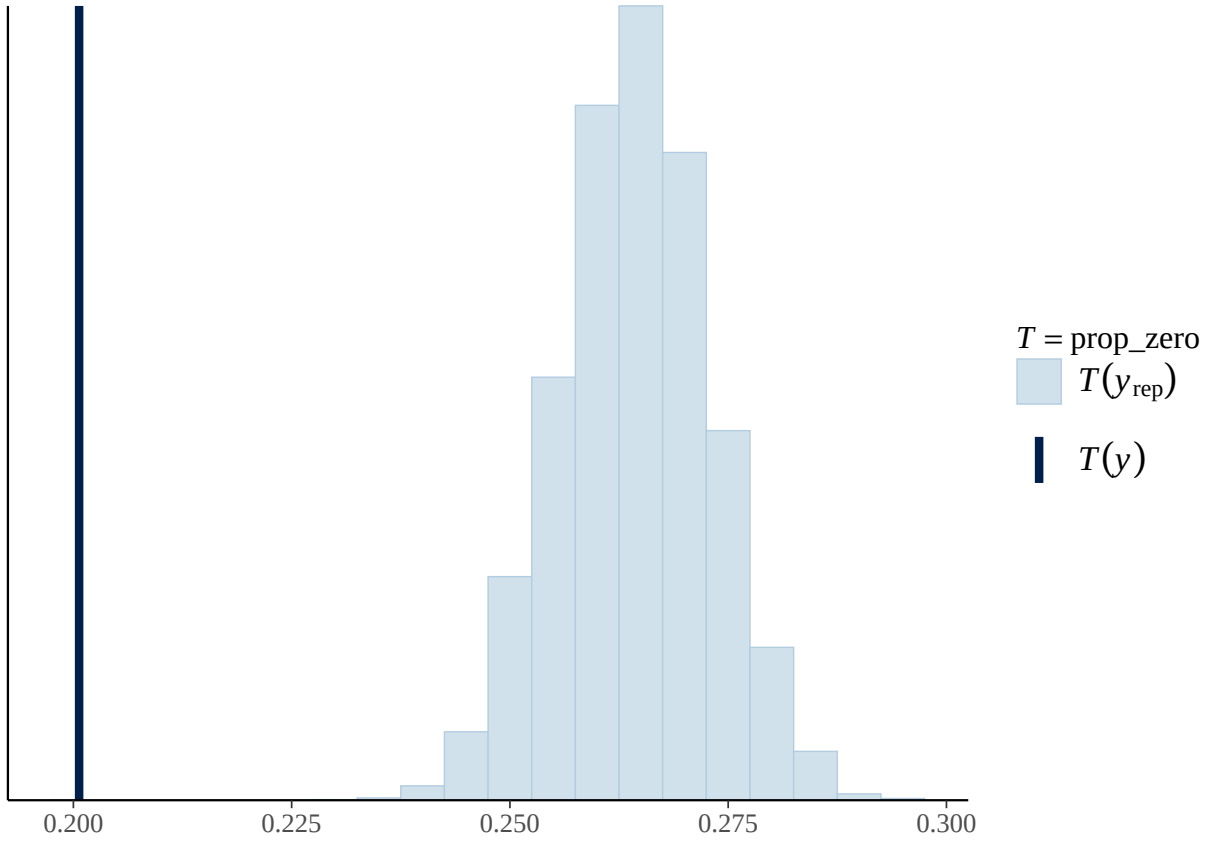


Look at the distribution of the proportion of zeros over the replicated datasets from the posterior predictive distribution in yrep and compare to the proportion of observed zeros in y.

```
ppc_stat(y4_fitness, yrep5_fitness_pois, stat = "prop_zero", binwidth = 0.005)
```




```
ppc_stat(y4_fitness, yrep5_fitness_nb, stat = "prop_zero", binwidth = 0.005)
```



They look quite similar.

Measure of fit: Bayesian R2

```
bayes_R2(bivar4.all.brm.pois)
```

```
##                            Estimate   Est.Error      Q2.5    Q97.5
## R2FFD                     0.6474613 0.009020140 0.6290550 0.664698
## R2roundmeanfitnessstudy 0.9419692 0.005269178 0.9305268 0.951232
```

```
bayes_R2(bivar4.all.brm.nb)
```

```
##                            Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                     0.6479816 0.009367079 0.6296746 0.6660829
## R2roundmeanfitnessstudy 0.9410392 0.005461288 0.9292700 0.9507628
```

Very similar.

Leave-one-out cross validation (LOO) (not run):

Extract selection coefficients

```
# Extract posterior samples
bivar4.all.brm.pois_post <- posterior_samples(bivar4.all.brm.pois)
bivar4.all.brm.pois_post <- as.mcmc(bivar4.all.brm.pois_post)
#head(bivar4.all.brm.pois_post)[,1:20]

# [,4] sd_id__FFD_Intercept
# [,5] sd_id__FFD_cmean_4
# [,6] sd_id__roundmeanfitnessfl_Intercept
# [,8] cor_id__FFD_Intercept__FFD_cmean_4
# [,9] cor_id__FFD_Intercept__roundmeanfitnessfl_Intercept
# [,10] cor_id__FFD_cmean_4__roundmeanfitnessfl_Intercept

sampled.gmat7 <- sample.gmat2(bivar4.all.brm.pois_post, replicates = 1000)

sgmat7 <- lapply(sampled.gmat7, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat7 <- unname(sapply(sgmat7, '[[', 1)) #Change to matrix

sgmat7 <- t(sgmat7)

P.modelBV_RR7 <- sgmat7
P.modelBV_RR7.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR7.mode[k] <- posterior.mode(mcmc(sgmat7[,k]))

# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR7 <- sgmat7[,c(3,6)]
colnames(S.modelBV_RR7) <- c("S_intercepts", "S_slopes")
S.modelBV_RR7.mode <- P.modelBV_RR7.mode[1:2, 3]

posterior.mode(mcmc(S.modelBV_RR7))
```

Poisson model

```
## S_intercepts     S_slopes
## -0.875166649  0.005193005
```

```
HPDinterval(mcmc(S.modelBV_RR7))
```

```
##                   lower      upper
## S_intercepts -1.3122218 -0.5975252
## S_slopes     -0.2331768  0.2884107
## attr(,"Probability")
## [1] 0.95
```

```
# Estimate selection gradients for intercept and slope (beta = S / P)
# on each sample of posterior and extract their mode
beta_post_RR7 <- matrix(NA, nrow(S.modelBV_RR7) ,2)

for (i in 1:nrow(S.modelBV_RR7)) {
  P3_7 <- matrix(rep(NA, 9), nrow = 3)
  # 3x3 matrix of var-cov for individual X.int, X.slope and fitness
  for (k in 1:9) {P3_7[k] <- P.modelBV_RR7[i, k] }
```

```
  P2_7 <- P3_7[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S7 <- P3_7[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR7[i,] <- solve(P2_7) %*% S7   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR7) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR7))
```

```
## beta_intercepts     beta_slopes
##      -0.6145579       0.7034820
```

```
HPDinterval(mcmc(beta_post_RR7))
```

```
##                        lower      upper
## beta_intercepts -1.42397377 -0.2906506
## beta_slopes      0.09790335  2.8874037
## attr(,"Probability")
## [1] 0.95
```

```
# Extract posterior samples
bivar4.all.brm.nb_post <- posterior_samples(bivar4.all.brm.nb)
bivar4.all.brm.nb_post <- as.mcmc(bivar4.all.brm.nb_post)
#head(bivar4.all.brm.nb_post)[,1:20]

# [,4] sd_id__FFD_Intercept
# [,5] sd_id__FFD_cmean_4
# [,6] sd_id__roundmeanfitnessfl_Intercept
# [,8] cor_id__FFD_Intercept__FFD_cmean_4
# [,9] cor_id__FFD_Intercept__roundmeanfitnessfl_Intercept
# [,10] cor_id__FFD_cmean_4__roundmeanfitnessfl_Intercept

sampled.gmat8 <- sample.gmat2(bivar4.all.brm.nb_post, replicates = 1000)

sgmat8 <- lapply(sampled.gmat8, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat8 <- unname(sapply(sgmat8, '[[', 1)) #Change to matrix

sgmat8 <- t(sgmat8)

P.modelBV_RR8 <- sgmat8
P.modelBV_RR8.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR8.mode[k] <- posterior.mode(mcmc(sgmat8[,k]))

# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR8 <- sgmat8[,c(3,6)]
colnames(S.modelBV_RR8) <- c("S_intercepts", "S_slopes")
S.modelBV_RR8.mode <- P.modelBV_RR8.mode[1:2, 3]

posterior.mode(mcmc(S.modelBV_RR8))
```

Negative binomial model

```
## S_intercepts     S_slopes
##  -0.84848121   0.05332679
```

```
HPDinterval(mcmc(S.modelBV_RR8))
```

```
##                   lower      upper
## S_intercepts -1.2965363 -0.6182487
## S_slopes     -0.2036388  0.3350643
## attr(,"Probability")
## [1] 0.95
```

```
# Estimate selection gradients for intercept and slope (beta = S / P)
# on each sample of posterior and extract their mode
beta_post_RR8 <- matrix(NA, nrow(S.modelBV_RR8) ,2)

for (i in 1:nrow(S.modelBV_RR8)) {
  P3_8 <- matrix(rep(NA, 9), nrow = 3)
  # 3x3 matrix of var-cov for individual X.int, X.slope and fitness
  for (k in 1:9) {P3_8[k] <- P.modelBV_RR8[i, k] }
  P2_8 <- P3_8[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S8 <- P3_8[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR8[i,] <- solve(P2_8) %*% S8   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR8) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR8))
```

```
## beta_intercepts     beta_slopes
##      -0.5096578       0.8302126
```

```
HPDinterval(mcmc(beta_post_RR8))
```

```
##                      lower      upper
## beta_intercepts -1.3419178 -0.3366617
## beta_slopes      0.1523333  2.6104435
## attr(,"Probability")
## [1] 0.95
```

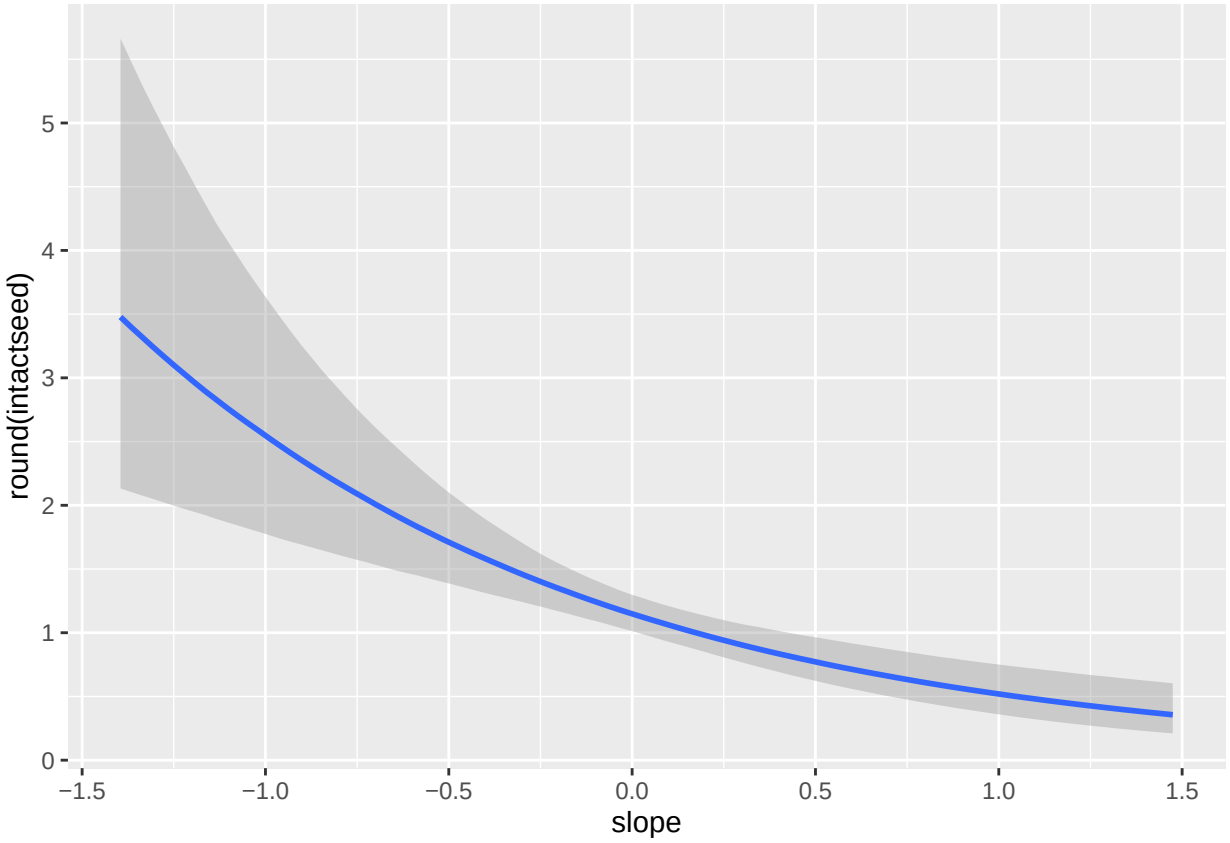
Variation in selection among years with BLUPs

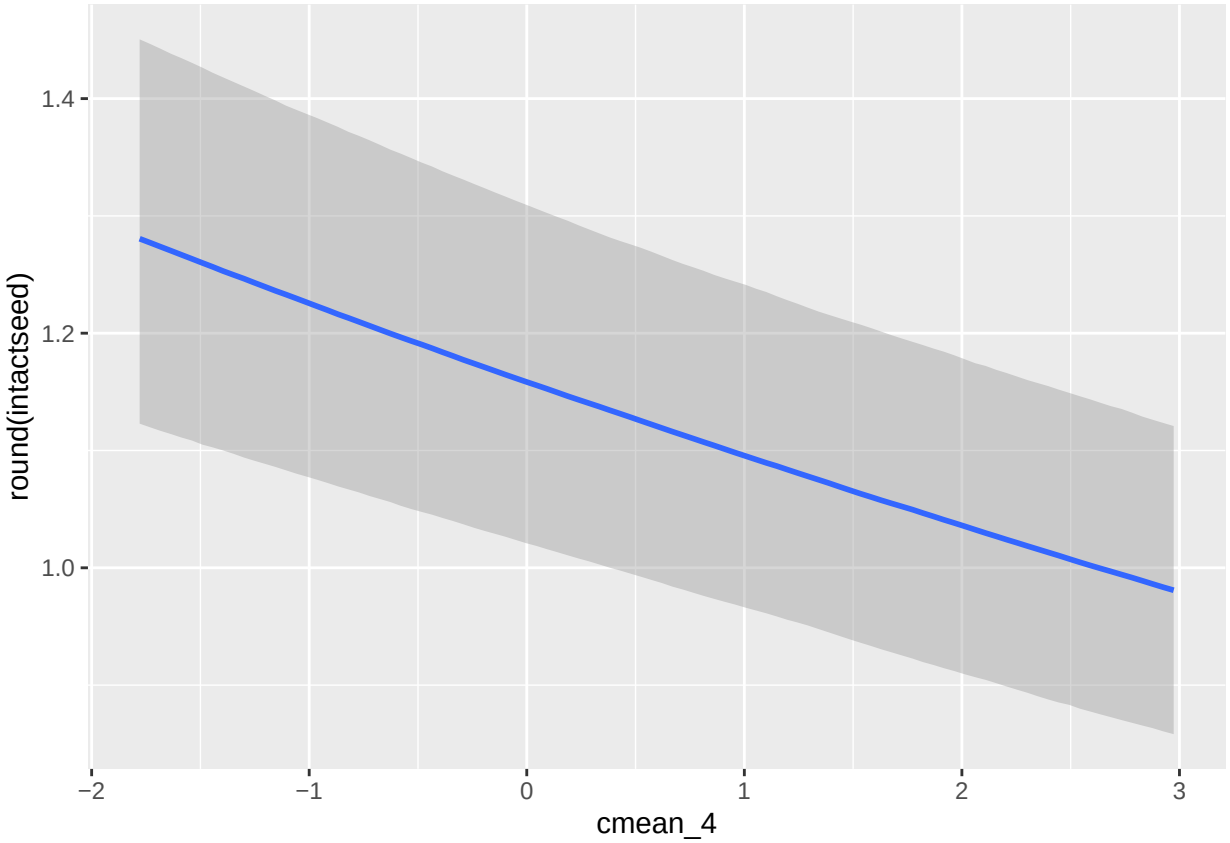
Add BLUPs to data set

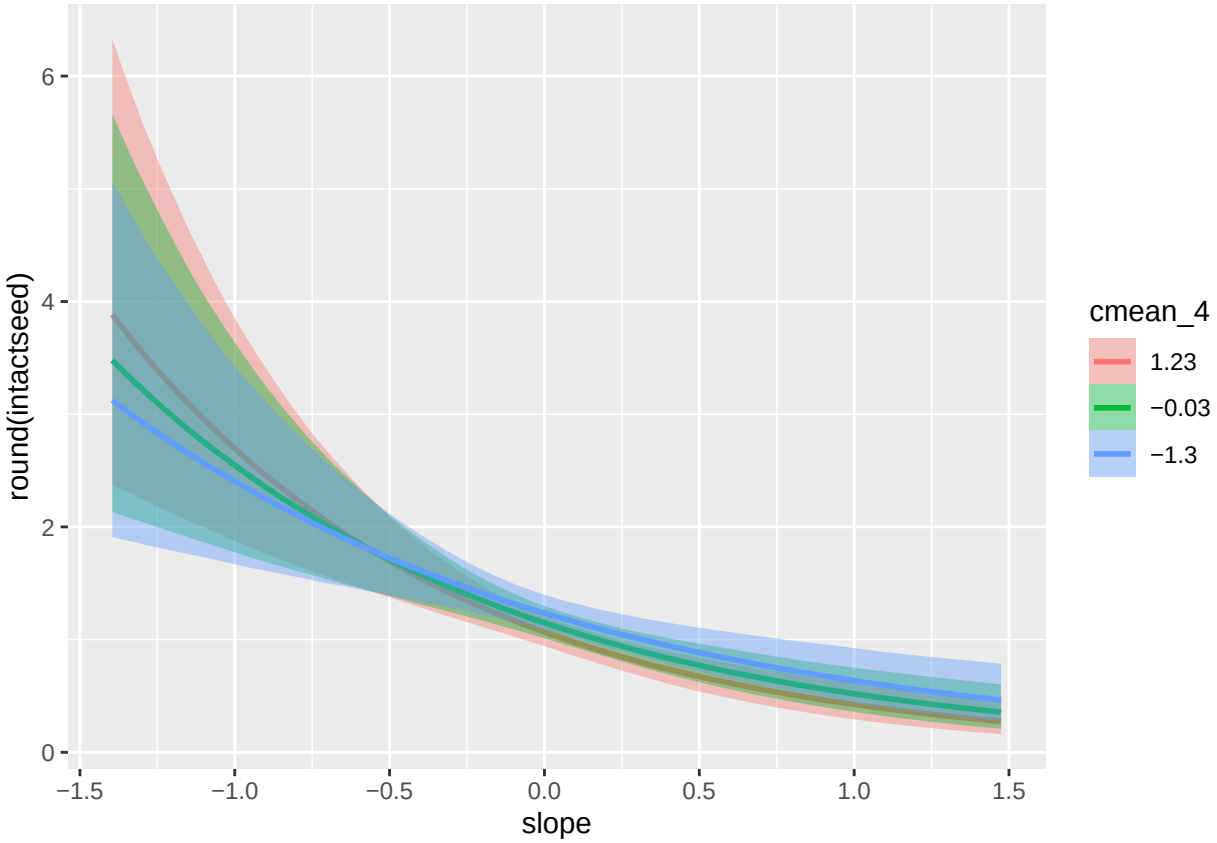
```
datadef<-datadef%>%left_join(BLUPs_MCMC.all.brms%>%
                               select(intercept,slope)%>%
                               rownames_to_column(var="id"))
```

Temp*slope

```
conditional_effects(modelBLUP_1)
```

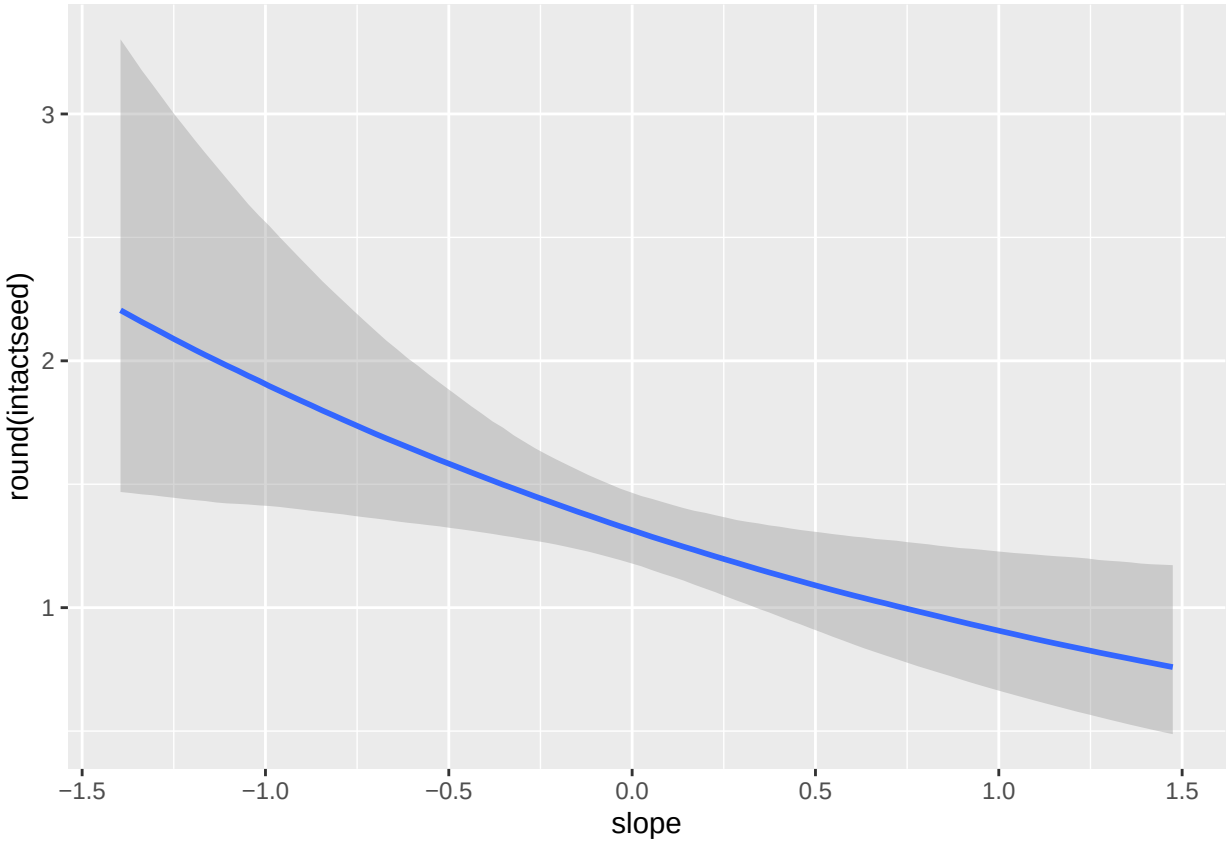


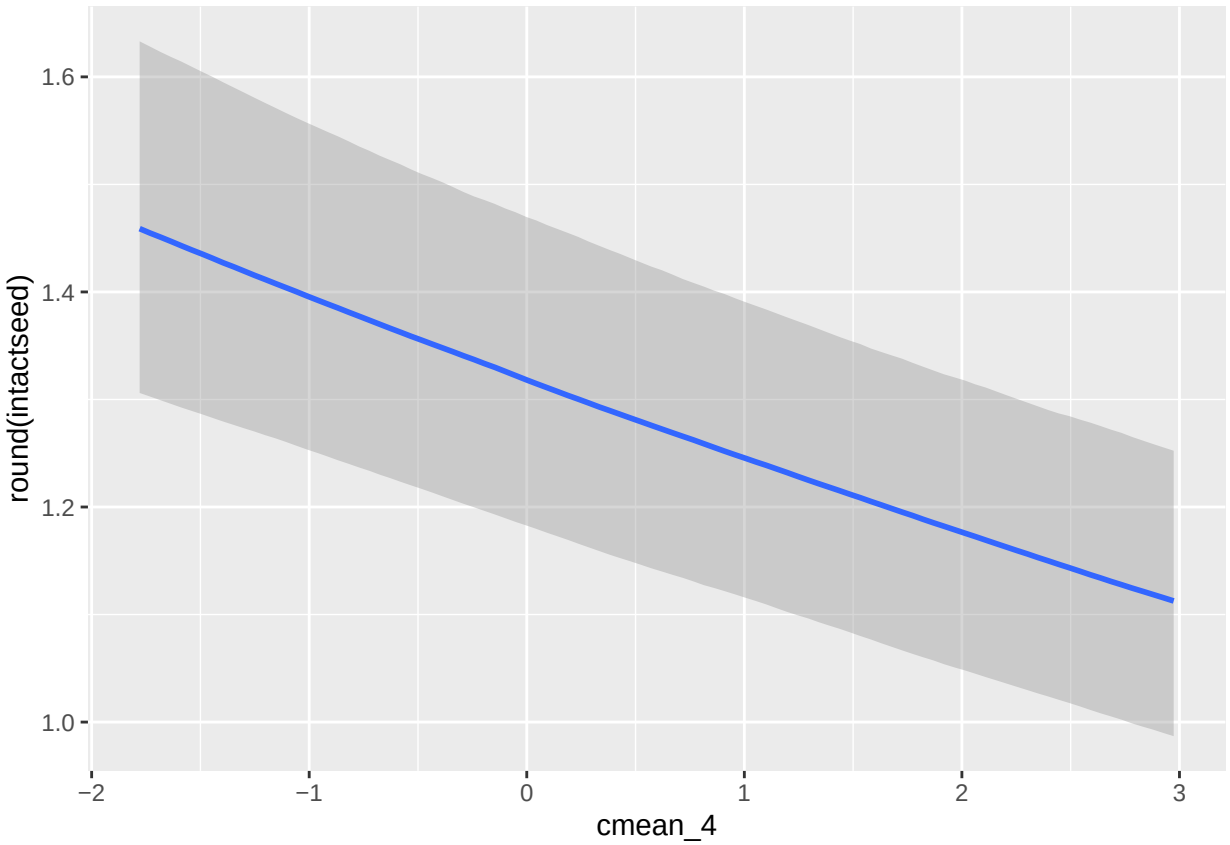

1.4
1.2
1.0
round(intactseed)
−2
−1
0
1
2
3
cmean_4

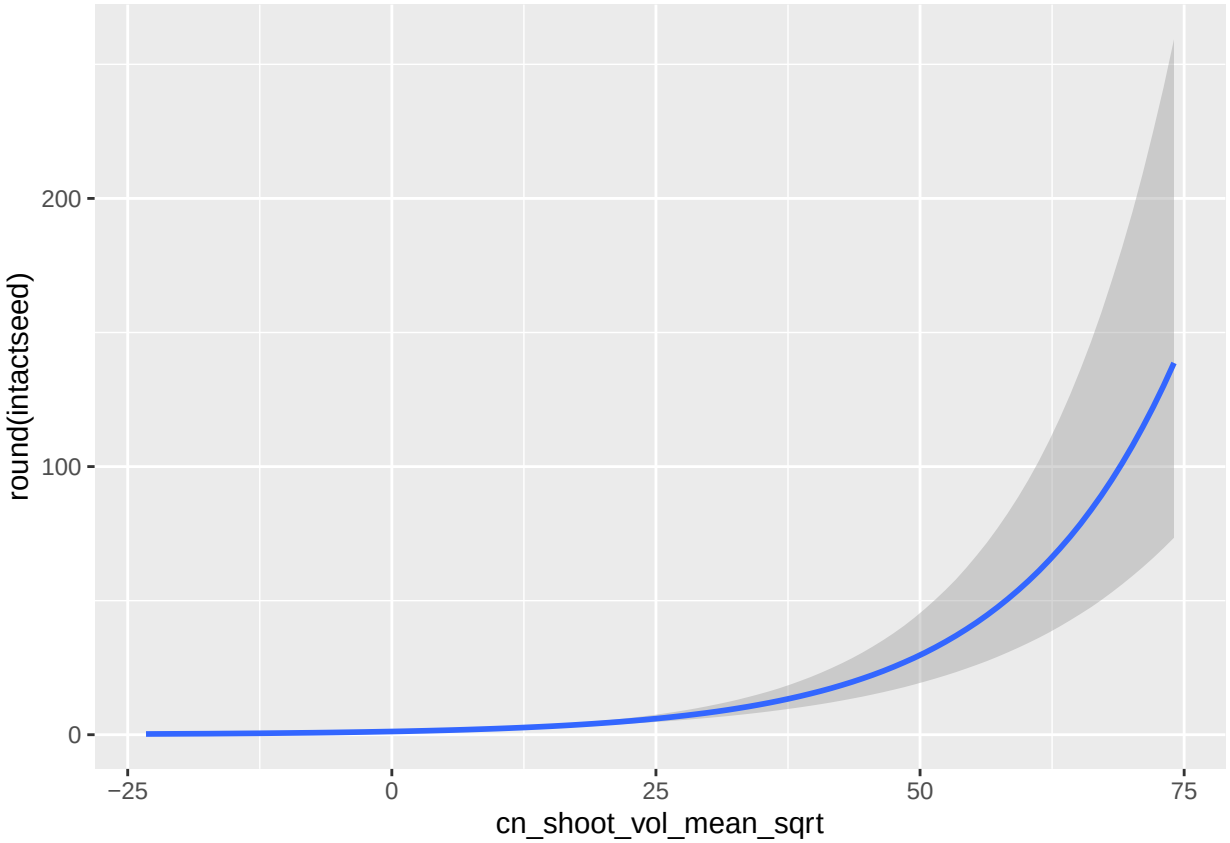


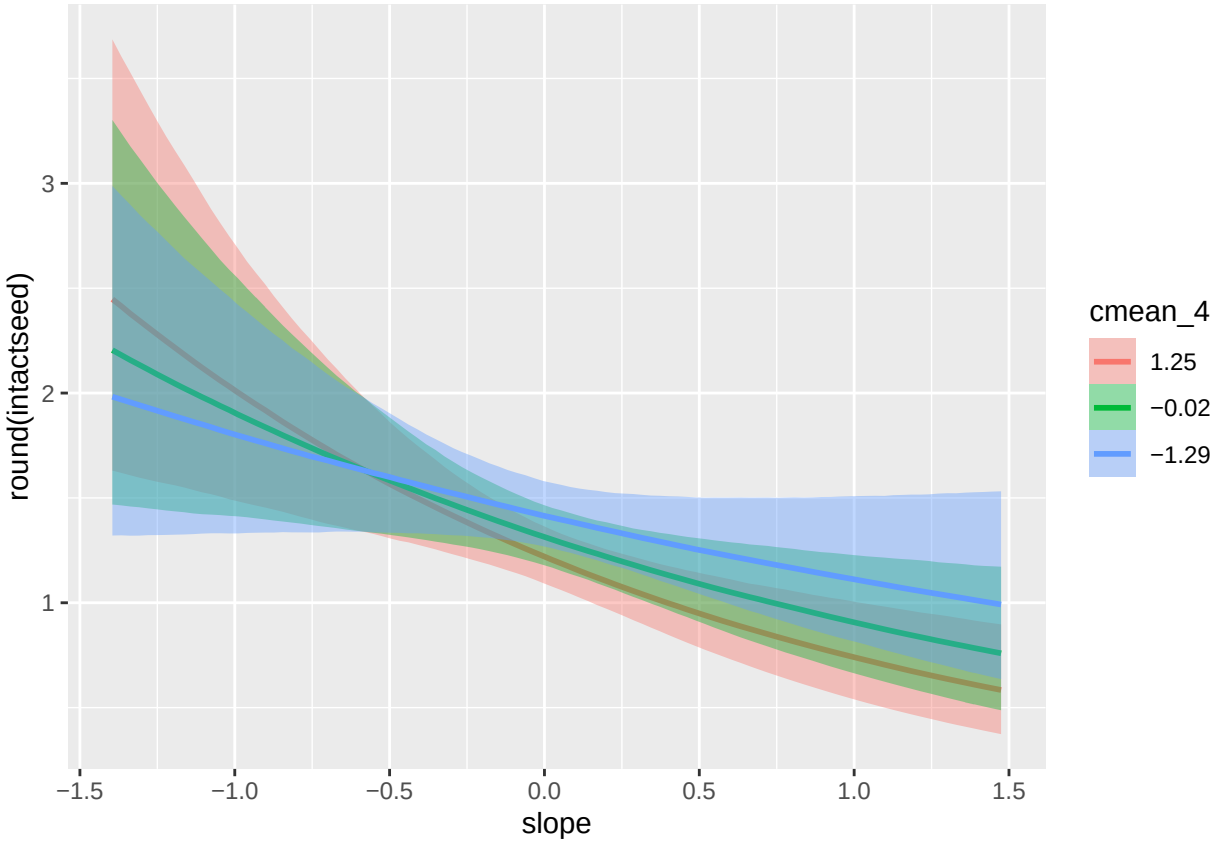
Temp*slope+volume

```
conditional_effects(modelBLUP_2)
```


round(intactseed)
3
2
1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
slope

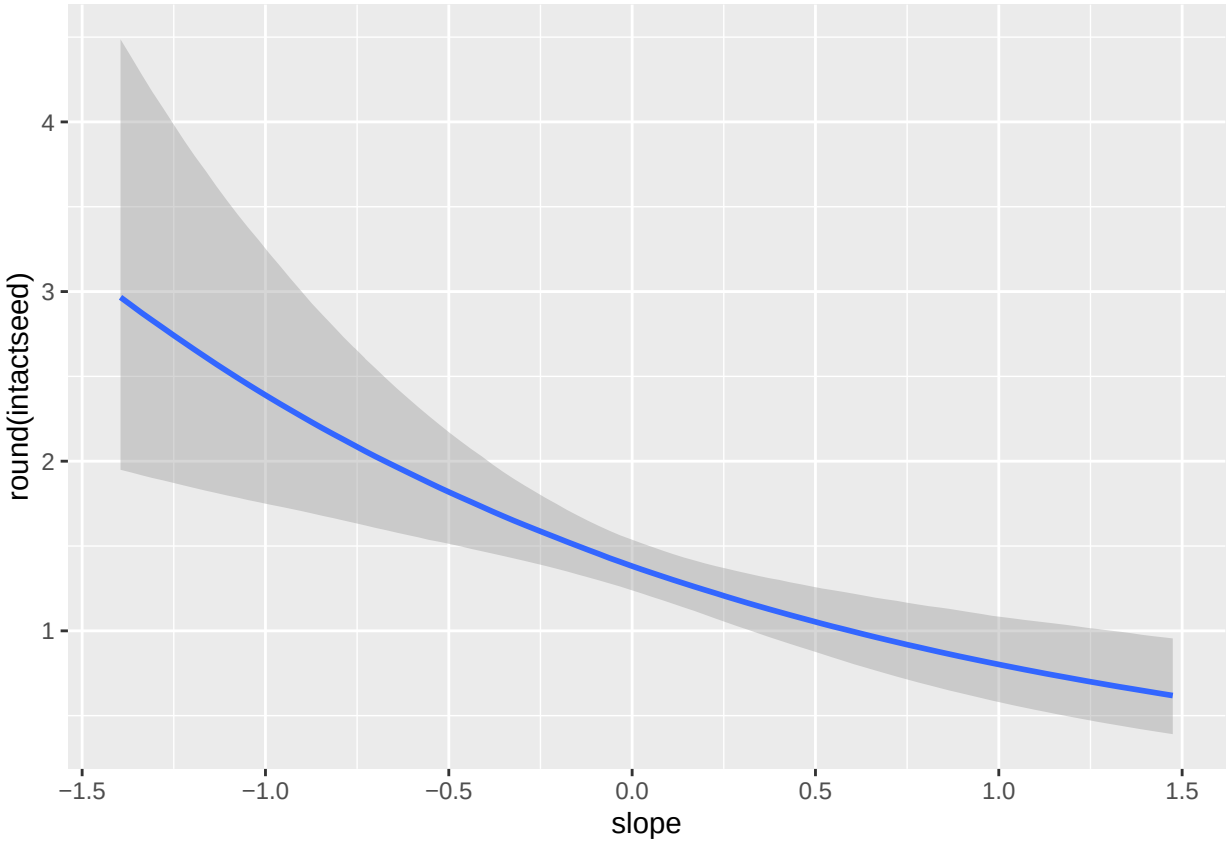

round(intactseed)
1.6
1.4
1.2
1.0
−2
−1
0
1
2
3
cmean_4

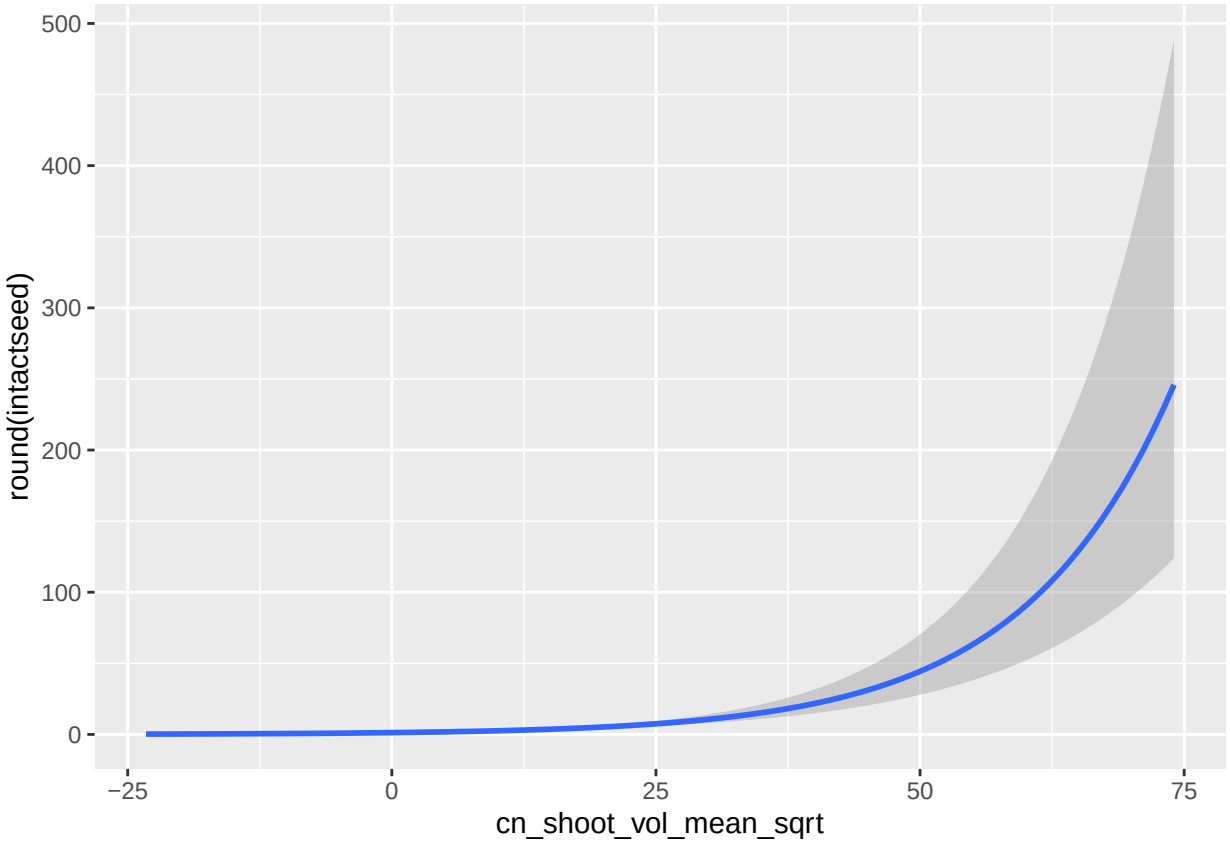

round(intactseed)
200
100
0
−25
0
25
50
75
cn_shoot_vol_mean_sqrt

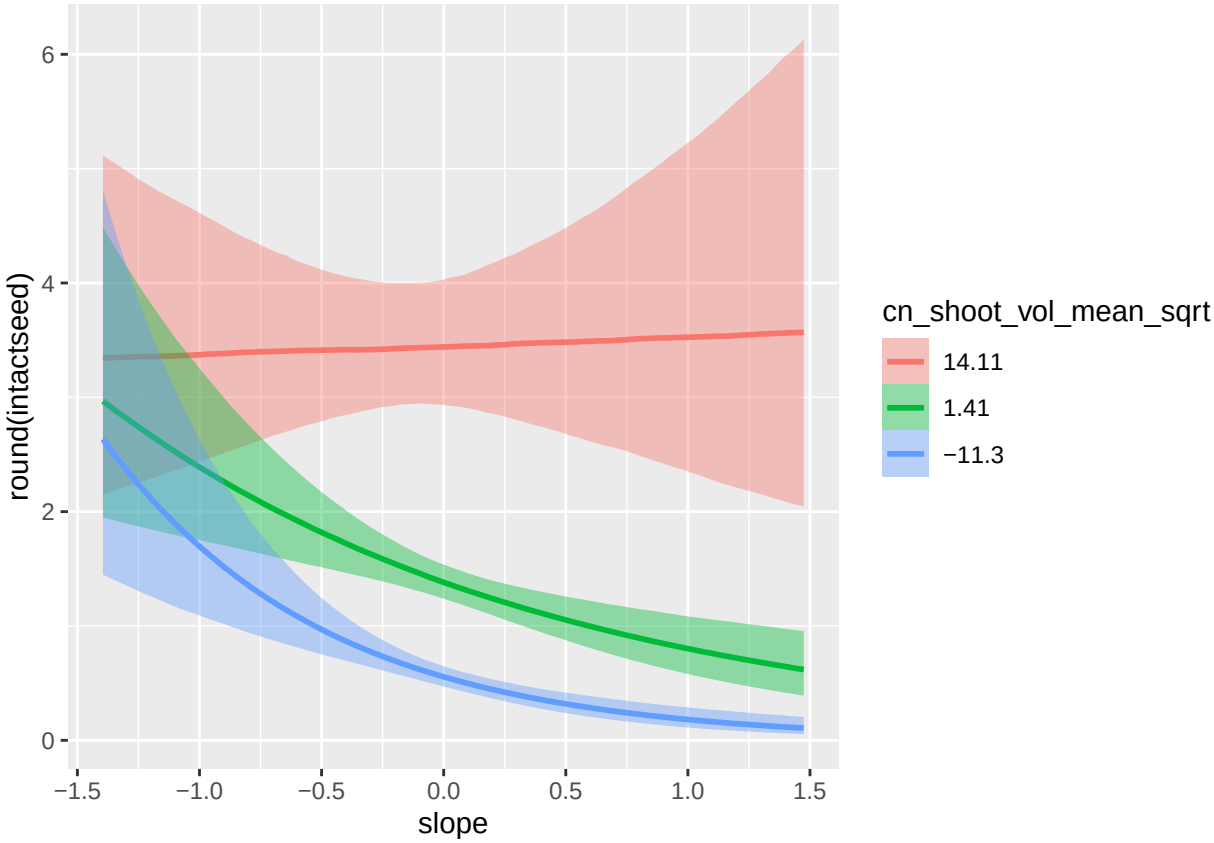


Volume*slope

```
conditional_effects(modelBLUP_3)
```

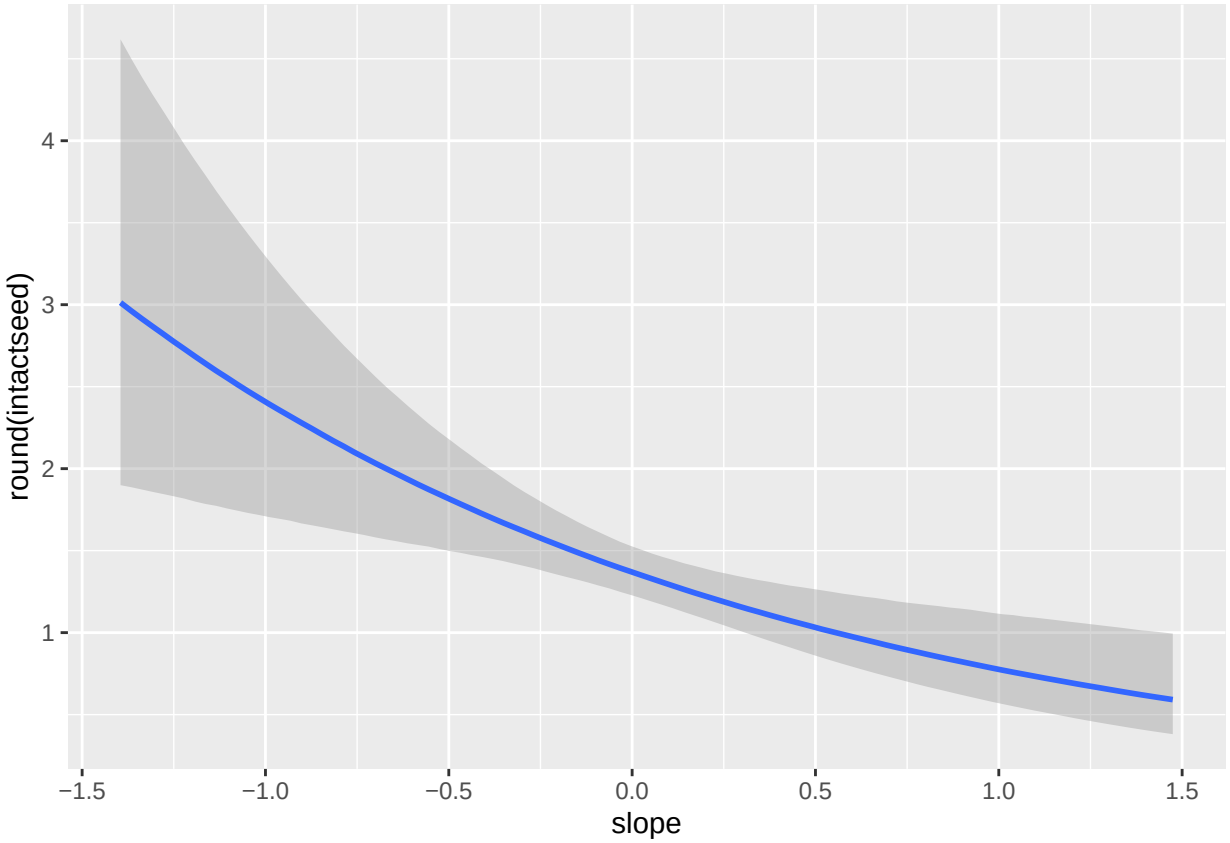

round(intactseed)
4
3
2
1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
slope

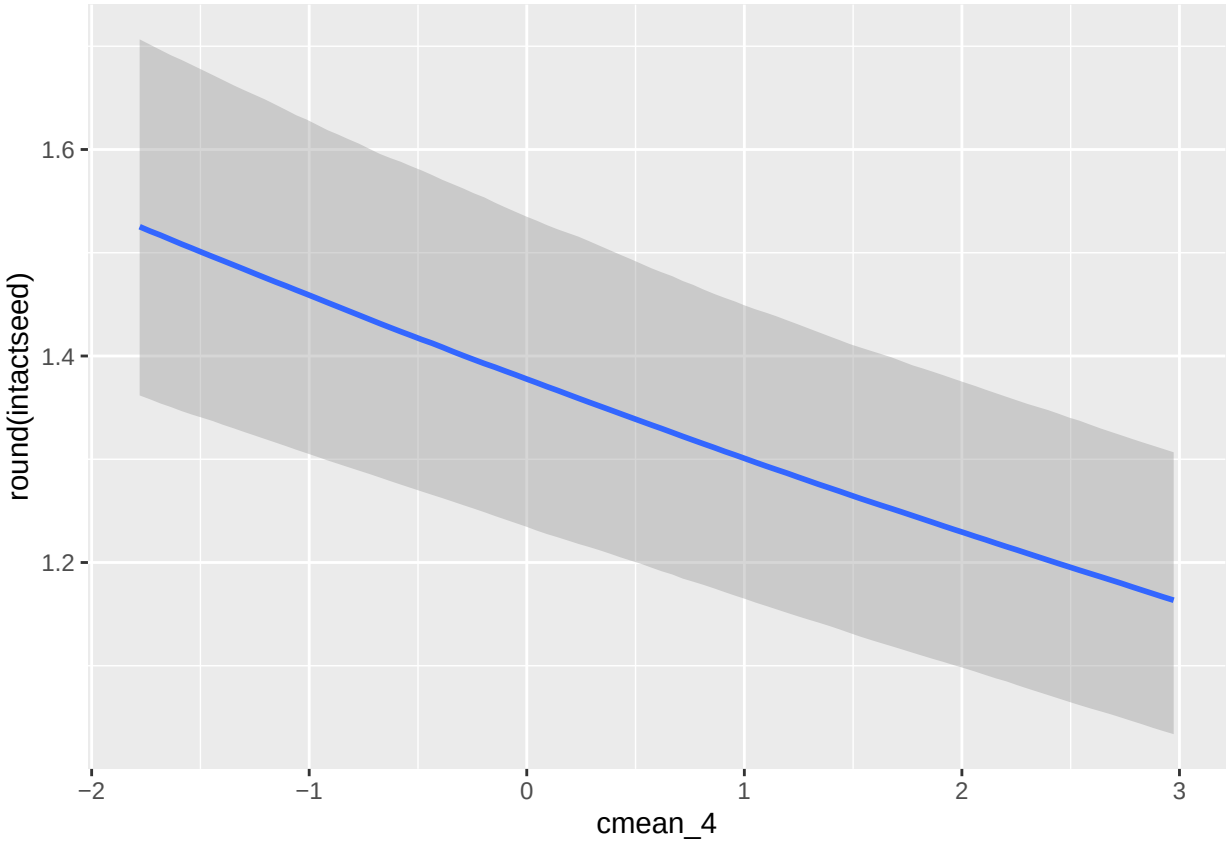

500
400
300
200
100
0
round(intactseed)
−25
0
25
50
75
cn_shoot_vol_mean_sqrt

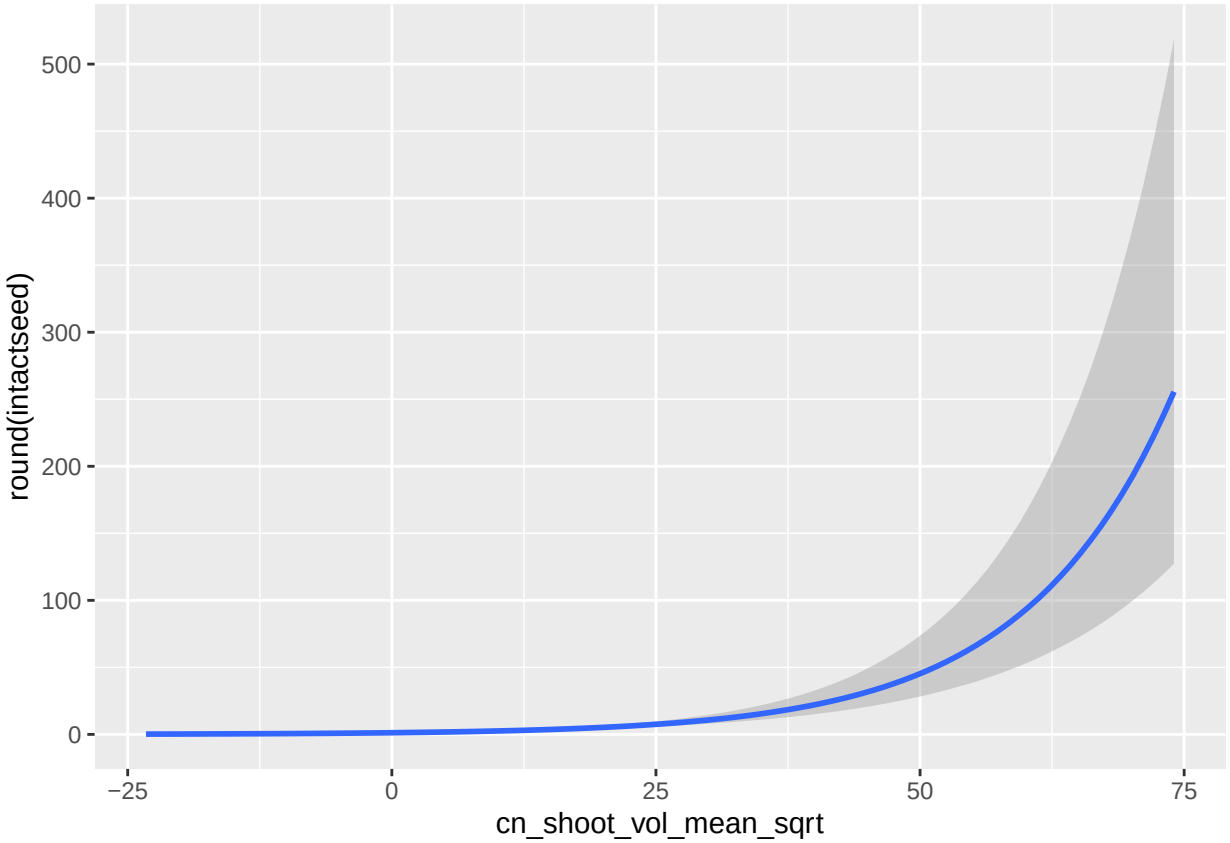


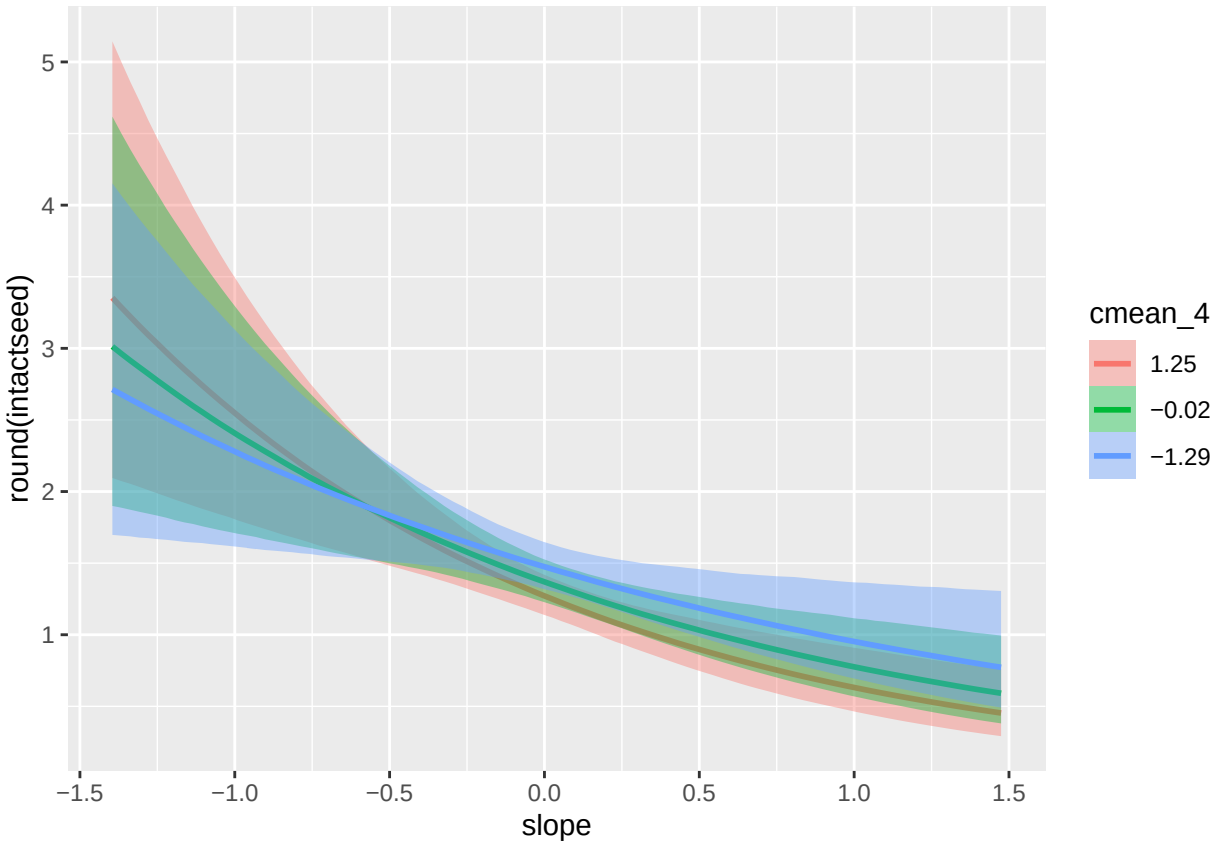
Temp*slope*+*volume*slope

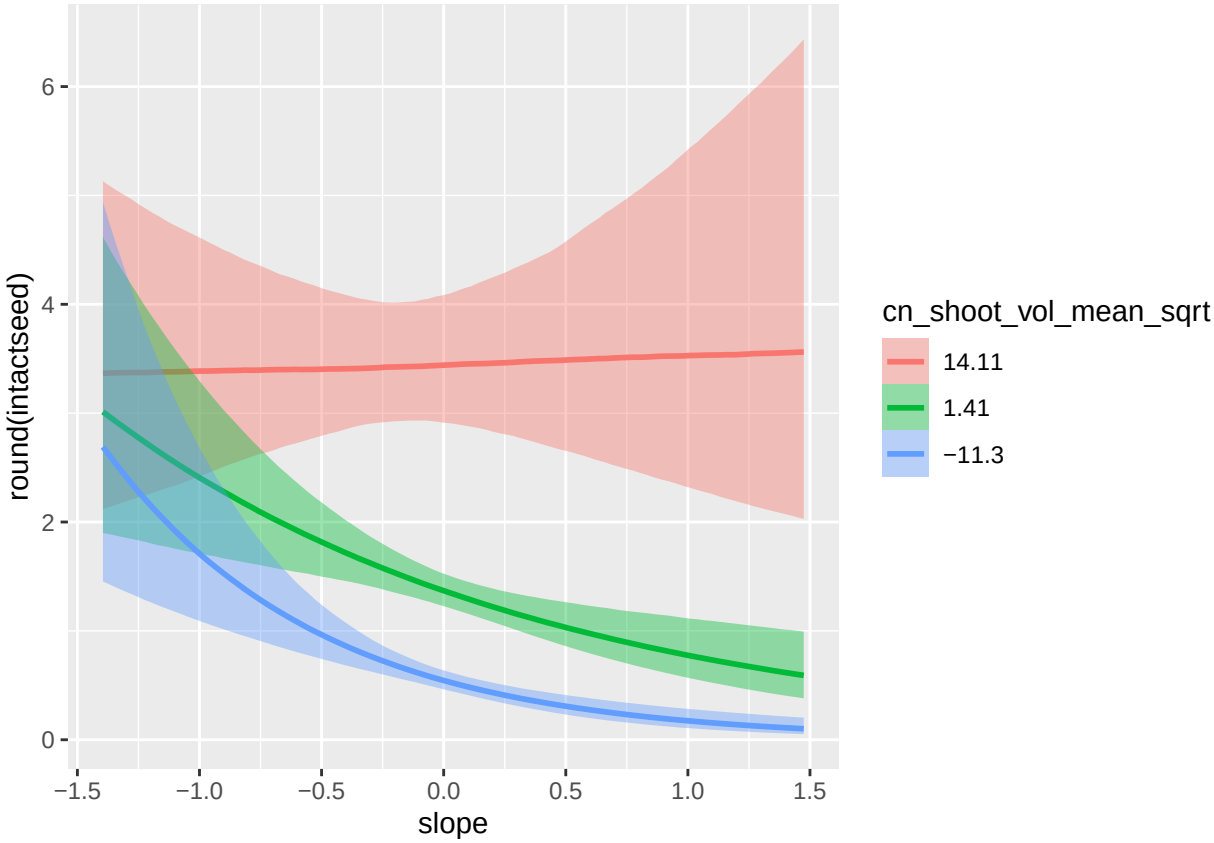
```
conditional_effects(modelBLUP_4)
```


round(intactseed)
4
3
2
1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
slope


round(intactseed)
1.6
1.4
1.2
−2
−1
0
1
2
3
cmean_4


500
400
300
200
100
0
round(intactseed)
−25
0
25
50
75
cn_shoot_vol_mean_sqrt


round(intactseed)
5
4
3
2
1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
slope
cmean_4
1.25
−0.02
−1.29



Save large objects as .RData file