Lathyrus ms2: selection on reaction norms for flowering time

Models with all data performed with brms

Alicia Valdés

Contents

Read data and check ns **2**

Data preparation **3**

Univariate models **6**

FFD with random effect of year only . . . 6

FFD with random effects of year and individual-intercept . . . 7

Random regression for FFD, including random effects of individual slopes and covariance between intercept and slope . . . 8

Graph model prediction for each id . . . 8

Compare models . . . 9

Model evaluation for random regression model . . . 10

Extract BLUPs from random regression model . . . 19

Figure 1 . . . 20

Bivariate models **21**

1. mean_fitness_fl, no condition variable . . . 21

Poisson distribution . . . 22

Negative binomial distribution . . . 23

Negative binomial distribution with zero-inflation . . . 24

Model evaluation: Compare models . . . 26

Prior predictive checks . . . 60

Extract selection coefficients . . . 61

Poisson model . . . 61

Negative binomial model . . . 64

Negative binomial model with zero-inflation . . . 66

2. mean_fitness_fl, with shoot volume . . . 67

Poisson distribution . . . 67

Negative binomial distribution . . . 68
Negative binomial distribution with zero-inflation . . . 69
Model evaluation: Compare models . . . 71
Extract selection coefficients . . . 93
Poisson model (not run due to warnings) . . . 94
Negative binomial model . . . 95
Negative binomial model with zero-inflation . . . 96
3. mean_fitness_study, no condition variable . . . 97
Poisson distribution . . . 97
Negative binomial distribution . . . 98
Negative binomial distribution with zero-inflation . . . 100
Model evaluation: Compare models . . . 101
Extract selection coefficients . . . 132
Poisson model . . . 133
Negative binomial model . . . 134
Negative binomial model with zero-inflation . . . 136
4. mean_fitness_study, with shoot volume . . . 137
Poisson distribution . . . 137
Negative binomial distribution . . . 137
Negative binomial distribution with zero-inflation . . . 139
Model evaluation: Compare models . . . 140
Extract selection coefficients . . . 161
Poisson model (not run due to warnings) . . . 161
Negative binomial model . . . 162
Negative binomial model with zero-inflation . . . 164
Figure 2: Selection differentials and gradients . . . 164

Save large objects as .RData file **166**

166

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
```

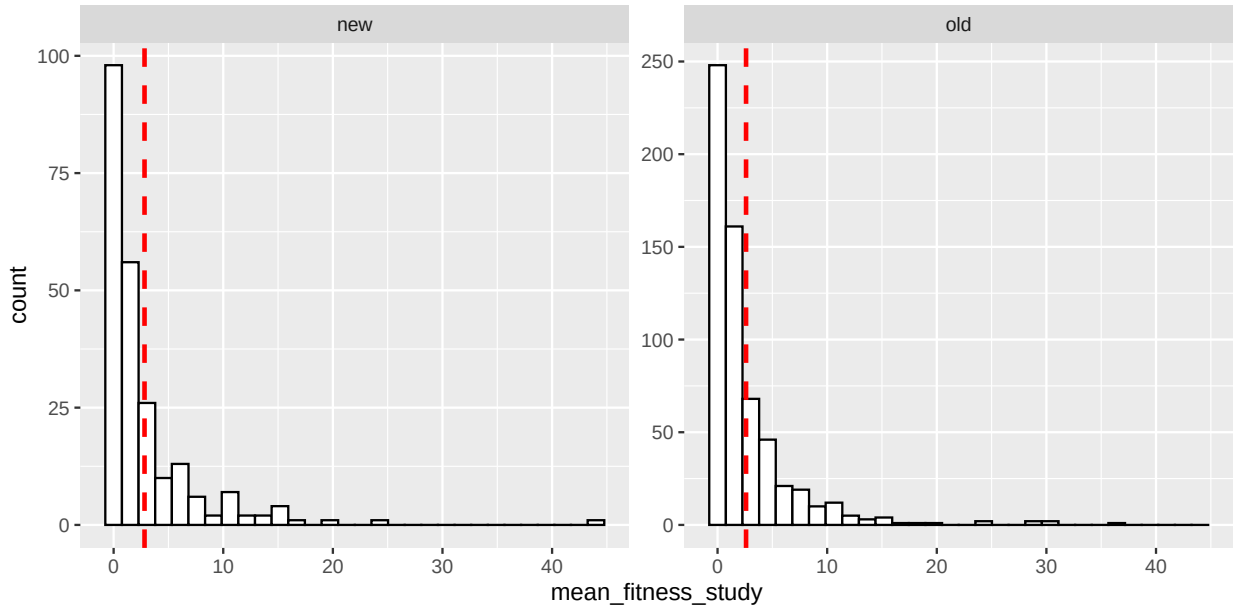
```
nrow(subset(datadef_total,is.na(shoot_vol_mean)))
```

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

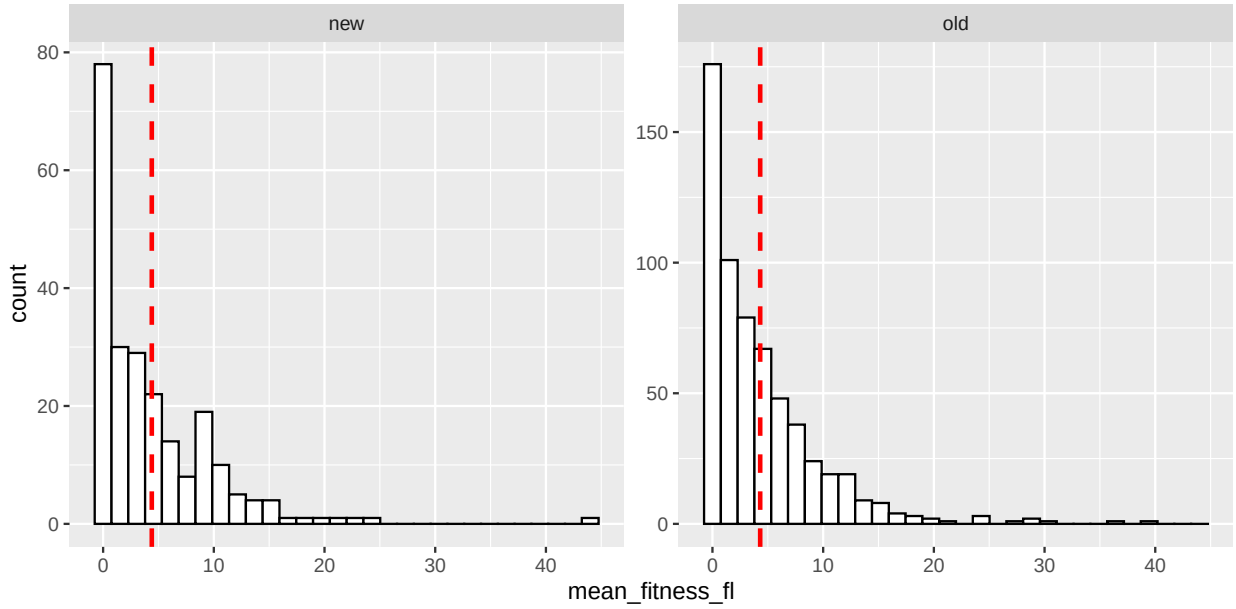
```
# 46 ids with no info on shoot volume

# Add first_yr to total data (this was needed for MCMCglmm models)
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
```

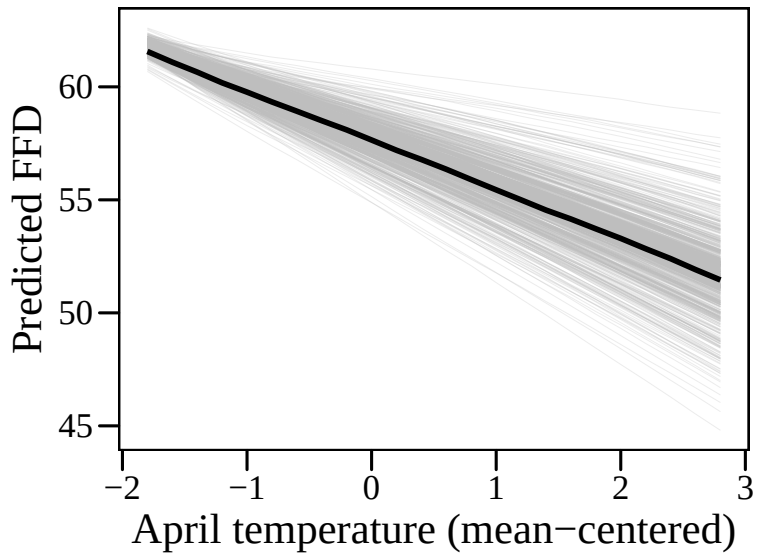
```
summary(univar.FFD_RR.all.brm)
```

```
##  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).
```

Graph model prediction for each id

```
predict_id<-ggpredict(univar.FFD_RR.all.brm,
                          terms=c("cmean_4","id"),type="random")
predict_mean<-ggpredict(univar.FFD_RR.all.brm,terms=c("cmean_4"),type="random")
```

```
ggplot()+
  geom_line(data=data.frame(predict_id),aes(x=x,y=predicted,group=group),
            color="grey",size=0.01,alpha=0.3)+
  geom_line(data=data.frame(predict_mean),aes(x=x,y=predicted),
            color="black",size=1)+
  my_theme()+
  xlab("April temperature (mean-centered)")+
  ylab("Predicted FFD")
```



Compare models

This allows us to test if adding among-individual variation in elevation and slope of the RN improves model fit.

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

```
loo_comp
```

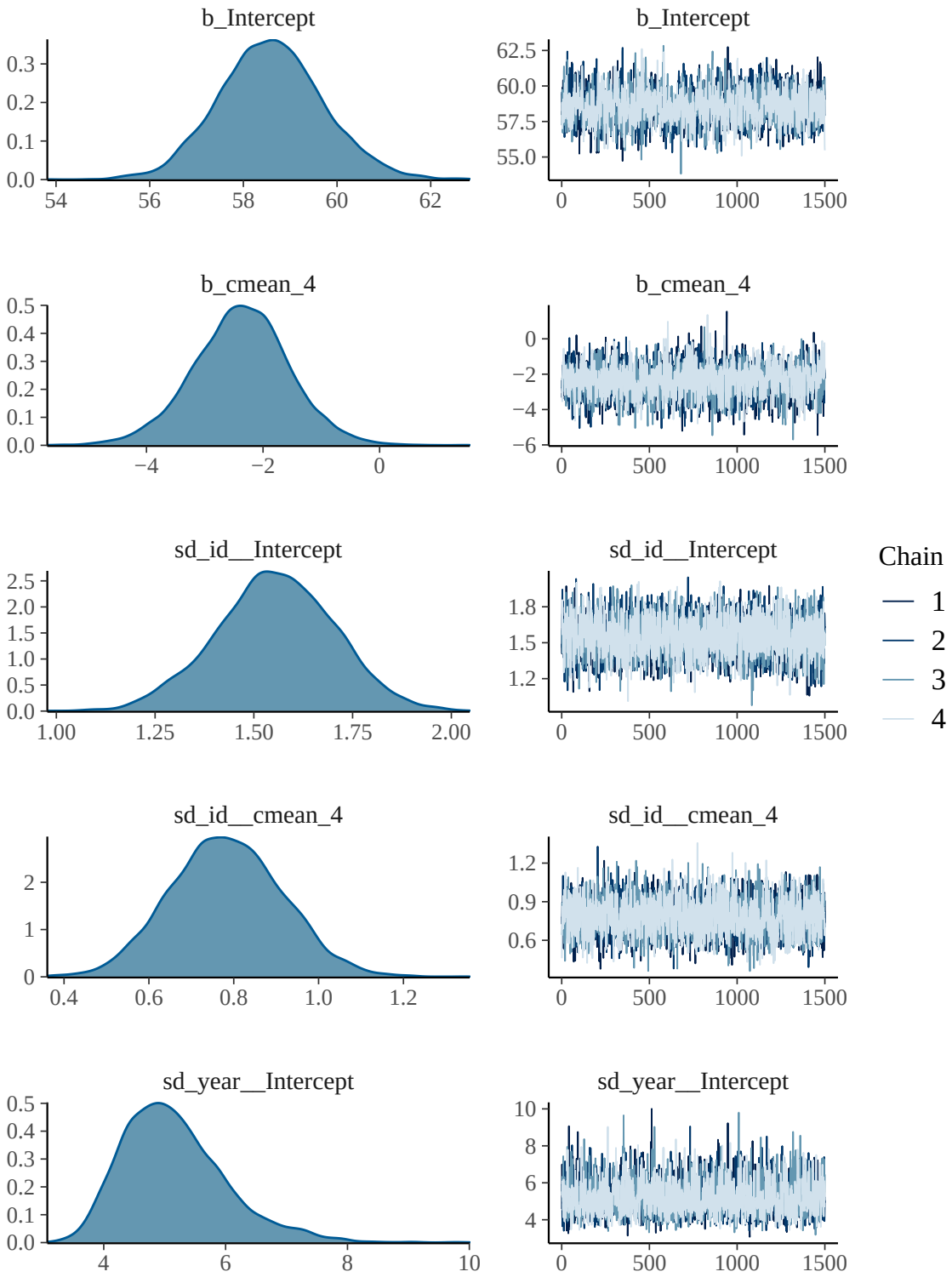
```
##                              elpd_diff se_diff
## univar.FFD_RR.all.brm          0.0       0.0
## univar.FFD.all.brm           -13.2       8.1
## univar.FFD_yearonly.all.brm -58.9      13.4
```

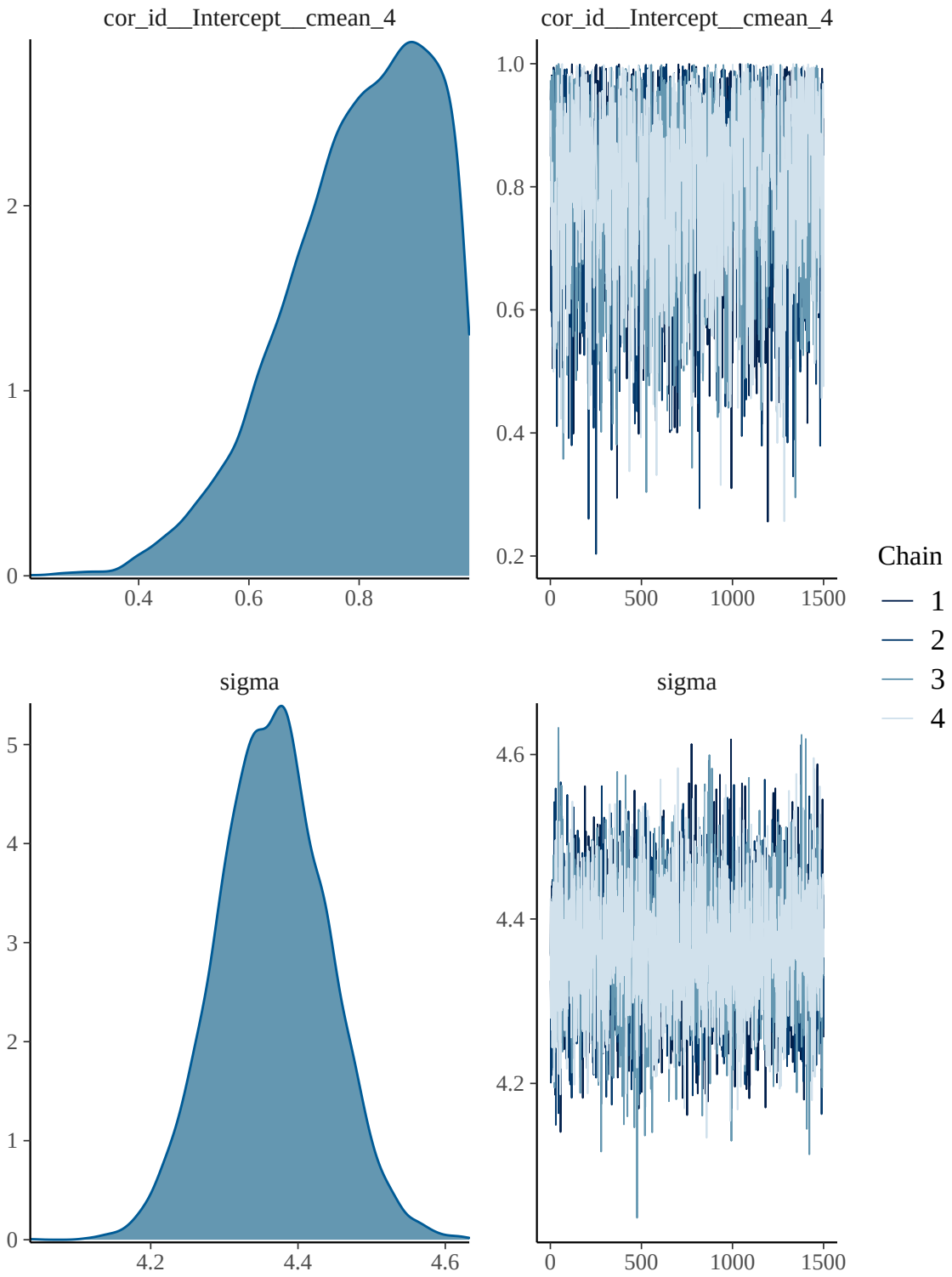
The results indicate that adding a random intercept (i.e. among-individual variation in RN elevation) improves the fit, and that adding a random slope (i.e. among-individual variation in RN slope) improves the fit even more. Thus, the random regression model is the best model.

Model evaluation for random regression model

I used some code from https://biol609.github.io/lectures/23c_brms_prediction.html#24_evaluating_brms_models

```
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

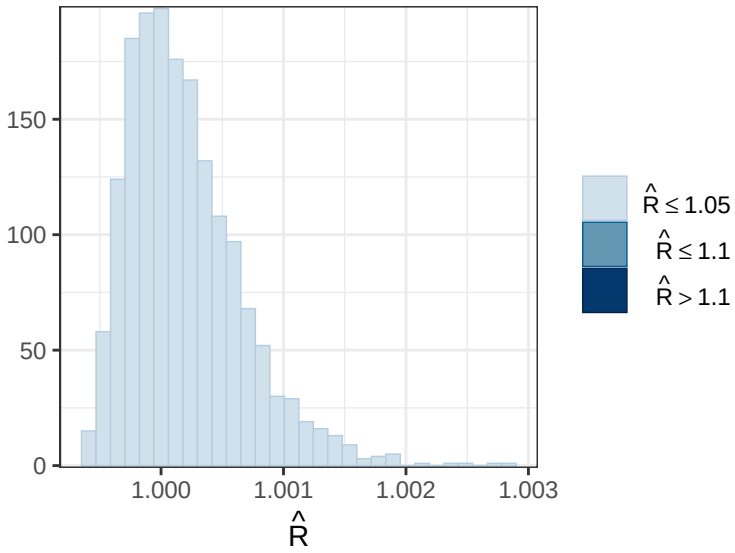


Rhat (potential scale reduction statistic): monitors whether a chain has converged to the equilibrium distribution by comparing its behavior to other randomly initialized chains.

- 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_RR.all.brm))+theme_bw()
```

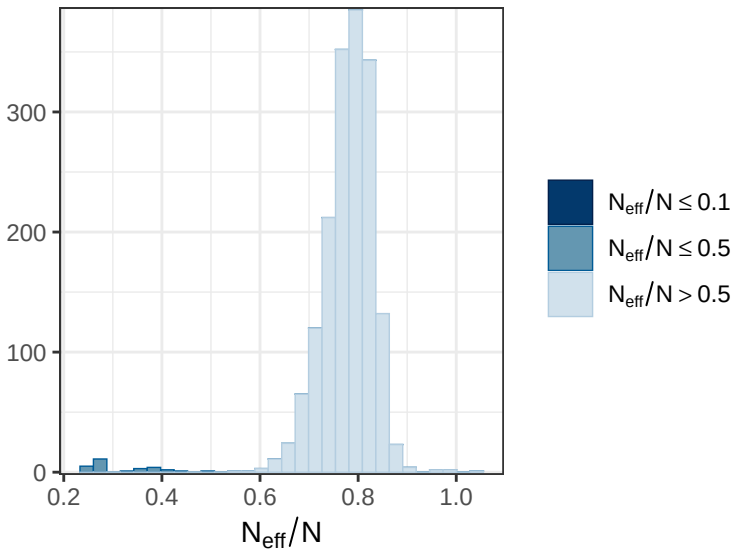


Effective sample size: an estimate of the number of independent draws from the posterior distribution of the estimand of interest.

- 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_RR.all.brm))+theme_bw()
```



Everything seems OK till here.

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).

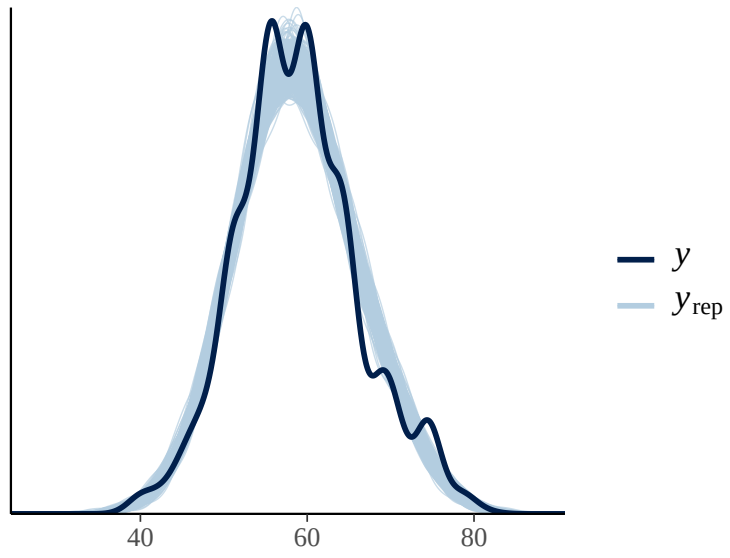
I used some code from https://cran.r-project.org/web/packages/bayesplot/vignettes/graphical-ppcs.html#:~:text=MCMC%20diagnostics%20vignette.-,Graphical%20posterior%20predictive%20checks%20(PPCs),Gabry%20et

```
y1<-subset(datadef,!is.na(FFD))$FFD # vector of outcome values
yrep3<-posterior_predict(univar.FFD_RR.all.brm, draws = 500)
# matrix 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 distribution of the simulated dataset in the yrep matrix

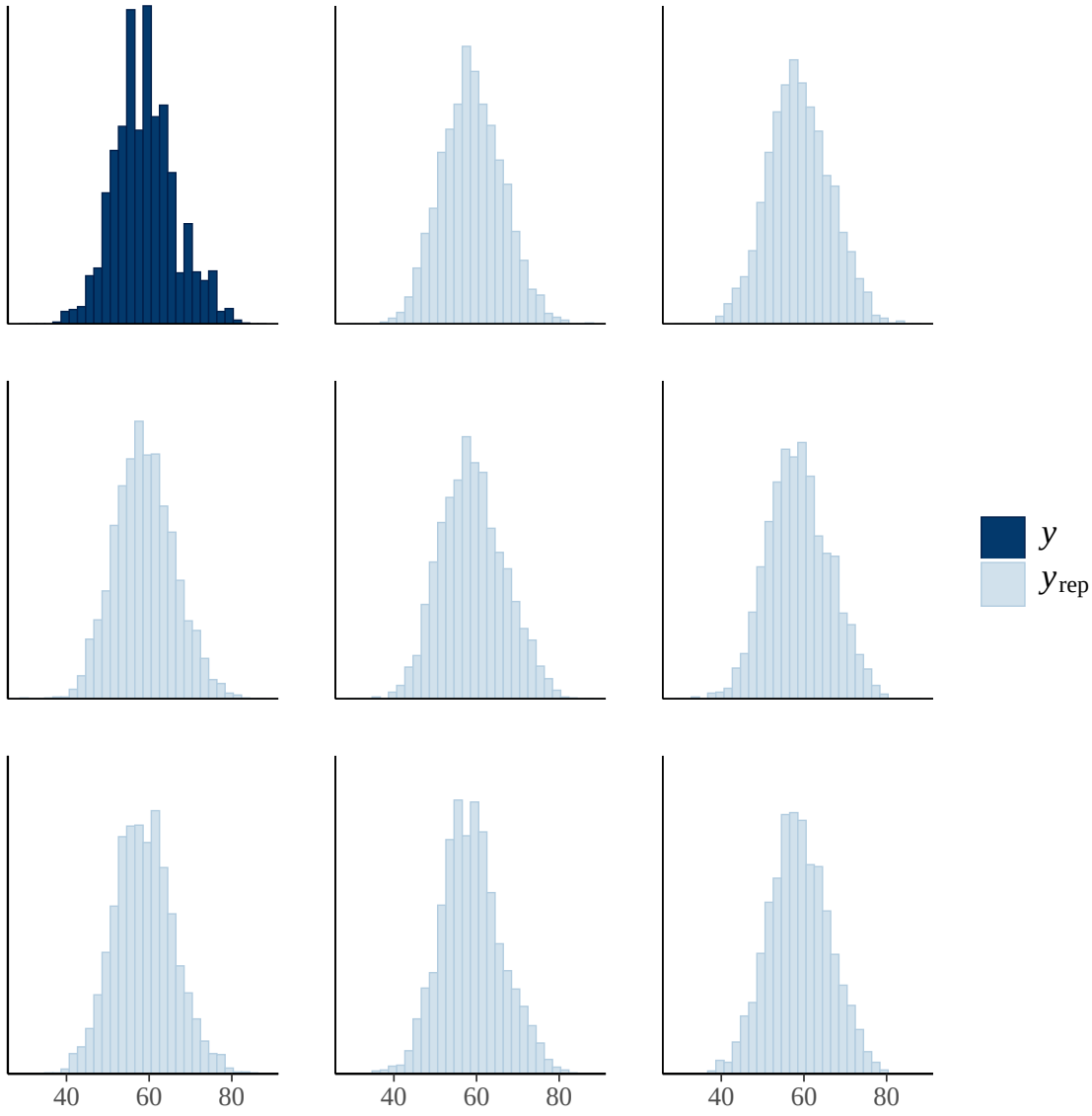
```
ppc_dens_overlay(y1, yrep3[1:500,])
```



Does this look good enough?

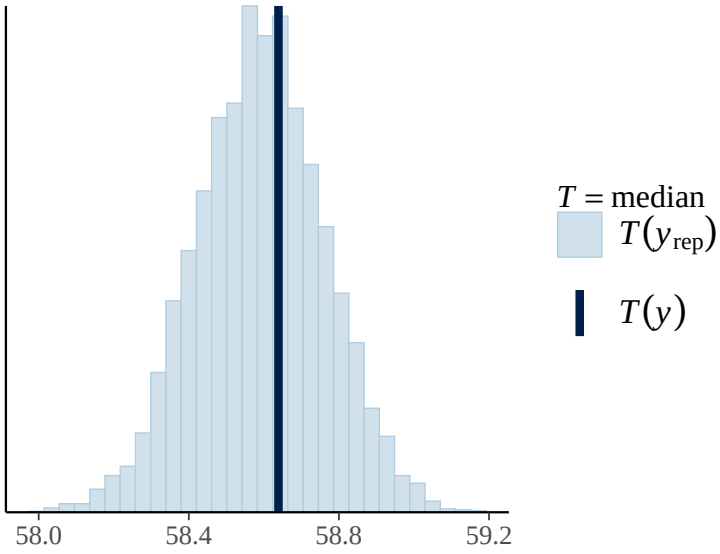
Separate histograms of y and some of the yrep datasets

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

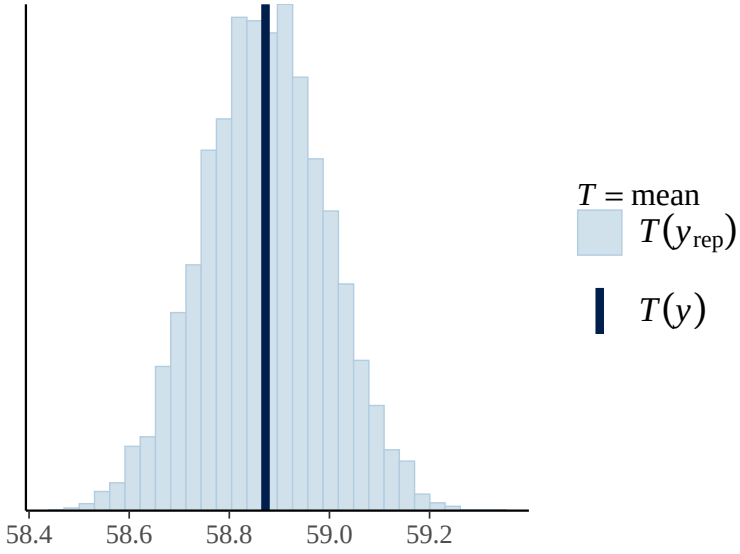


Does this look good enough?

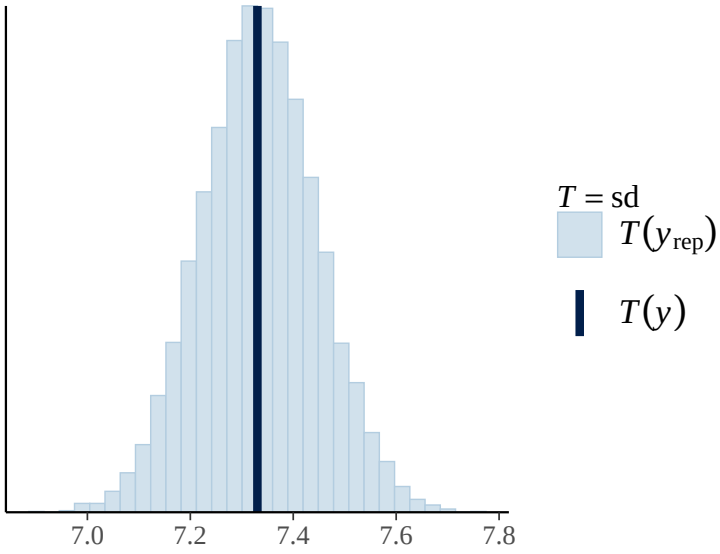
```
ppc_stat(y1,yrep3,stat="median")
```



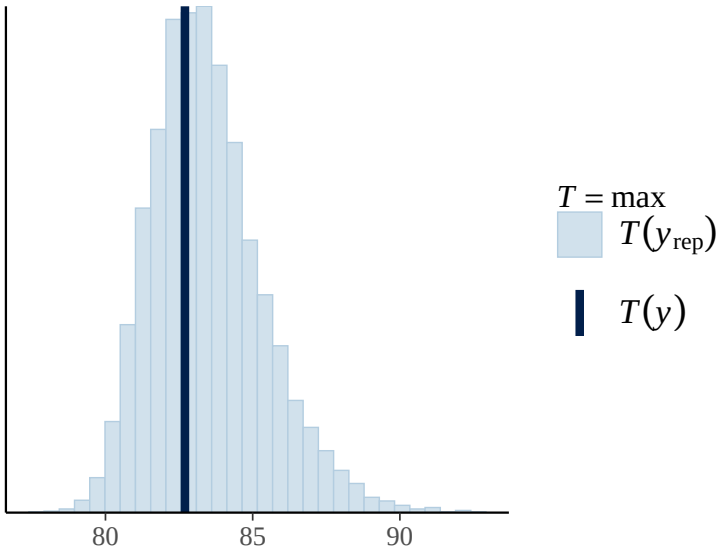

```
ppc_stat(y1,yrep3,stat="mean")
```



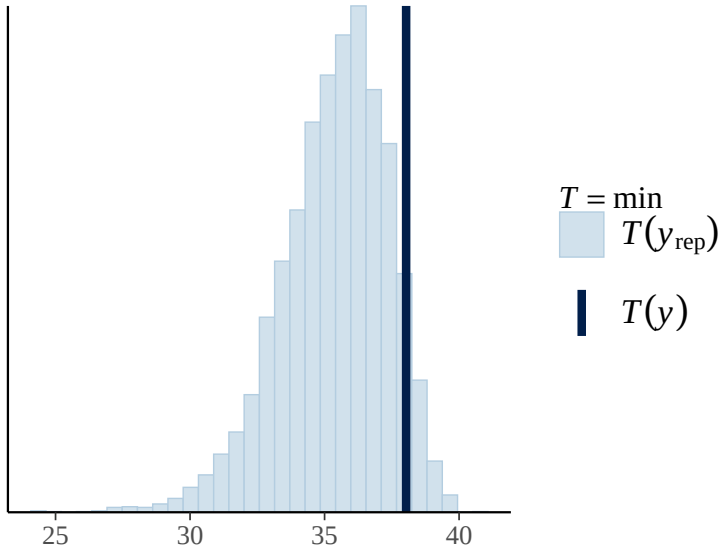

```
ppc_stat(y1,yrep3,stat="sd")
```




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




```
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_RR.all.brm)
```

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

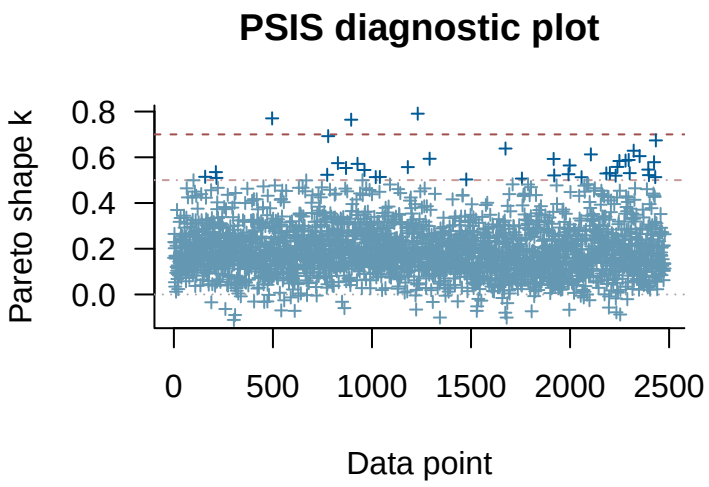
Leave-one-out cross validation (LOO):

I used code 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?

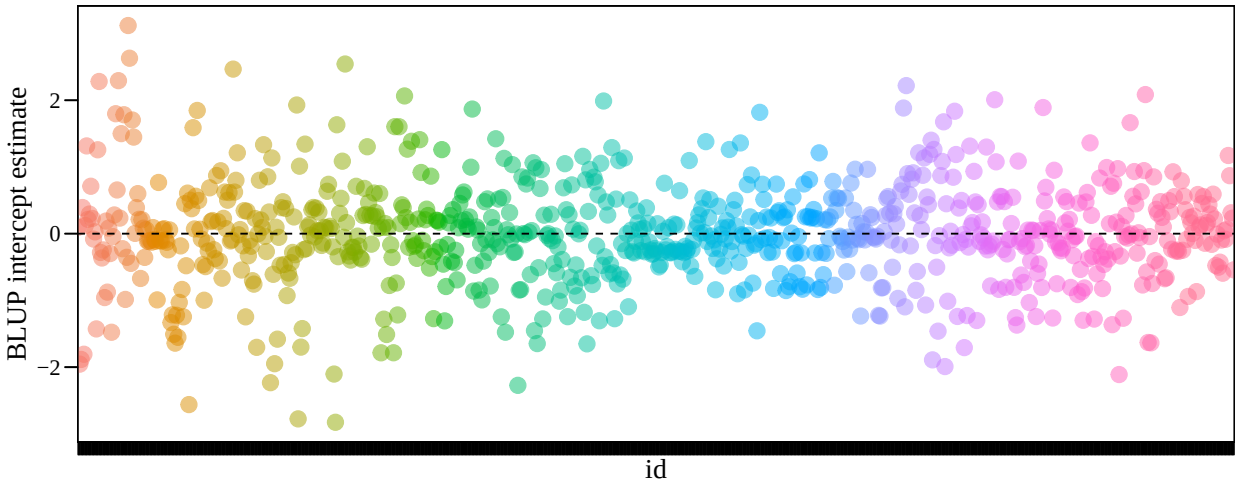
Extract BLUPs from random regression model

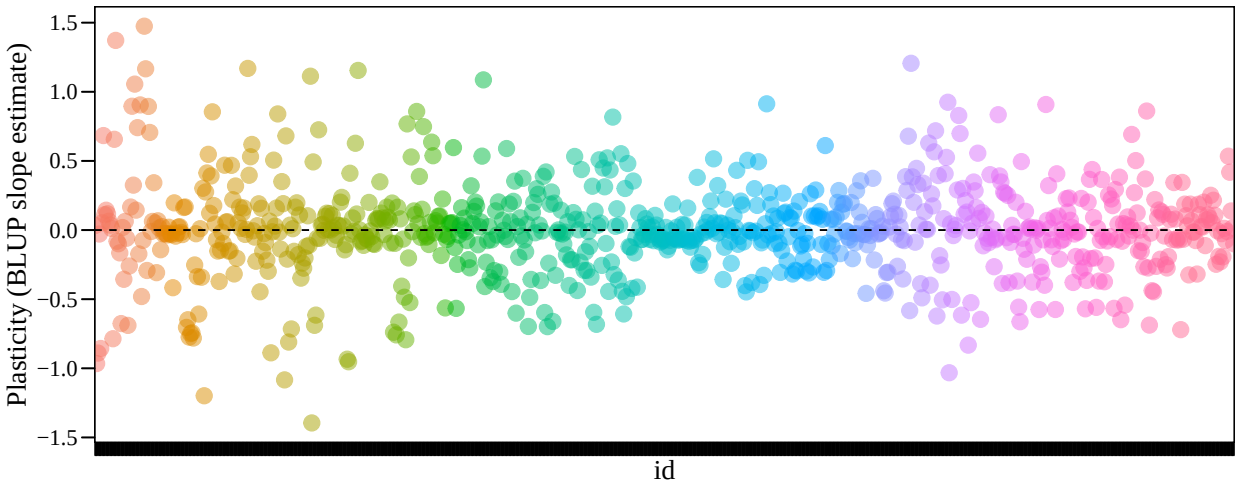
Code was checked by Piet.

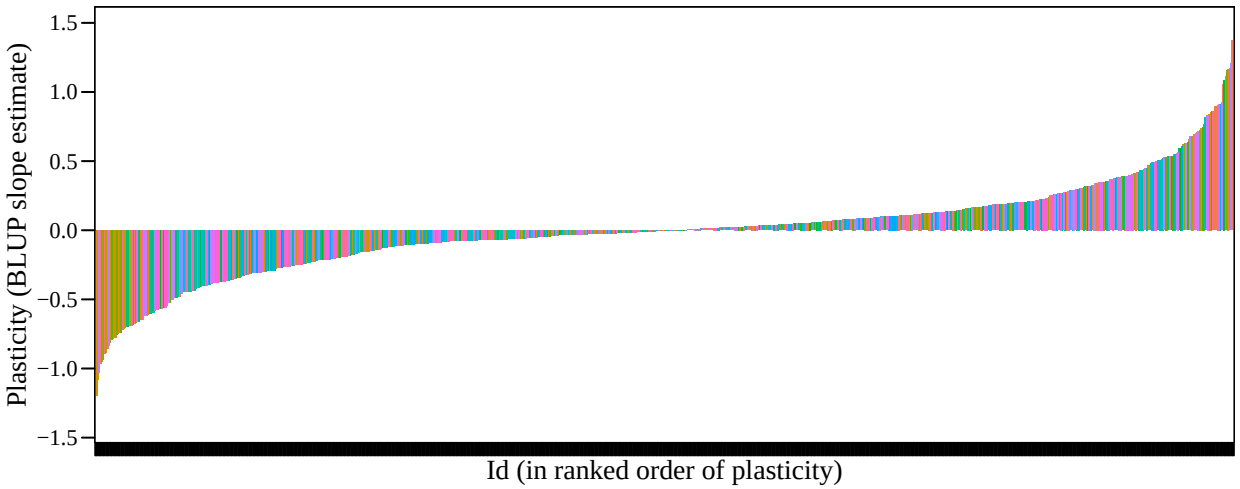
```
BLUPs_MCMC.all.brms  <- cbind(as.factor(c(1:837)),
                              as.data.frame(ranef(univar.FFD_RR.all.brm)$id)
                              [c(1,5)])
colnames(BLUPs_MCMC.all.brms) <- c("id", "intercept", "slope")
with(BLUPs_MCMC.all.brms,cor(intercept,slope)) # highly correlated!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

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

Plots with BLUPs







Add BLUPs to data set

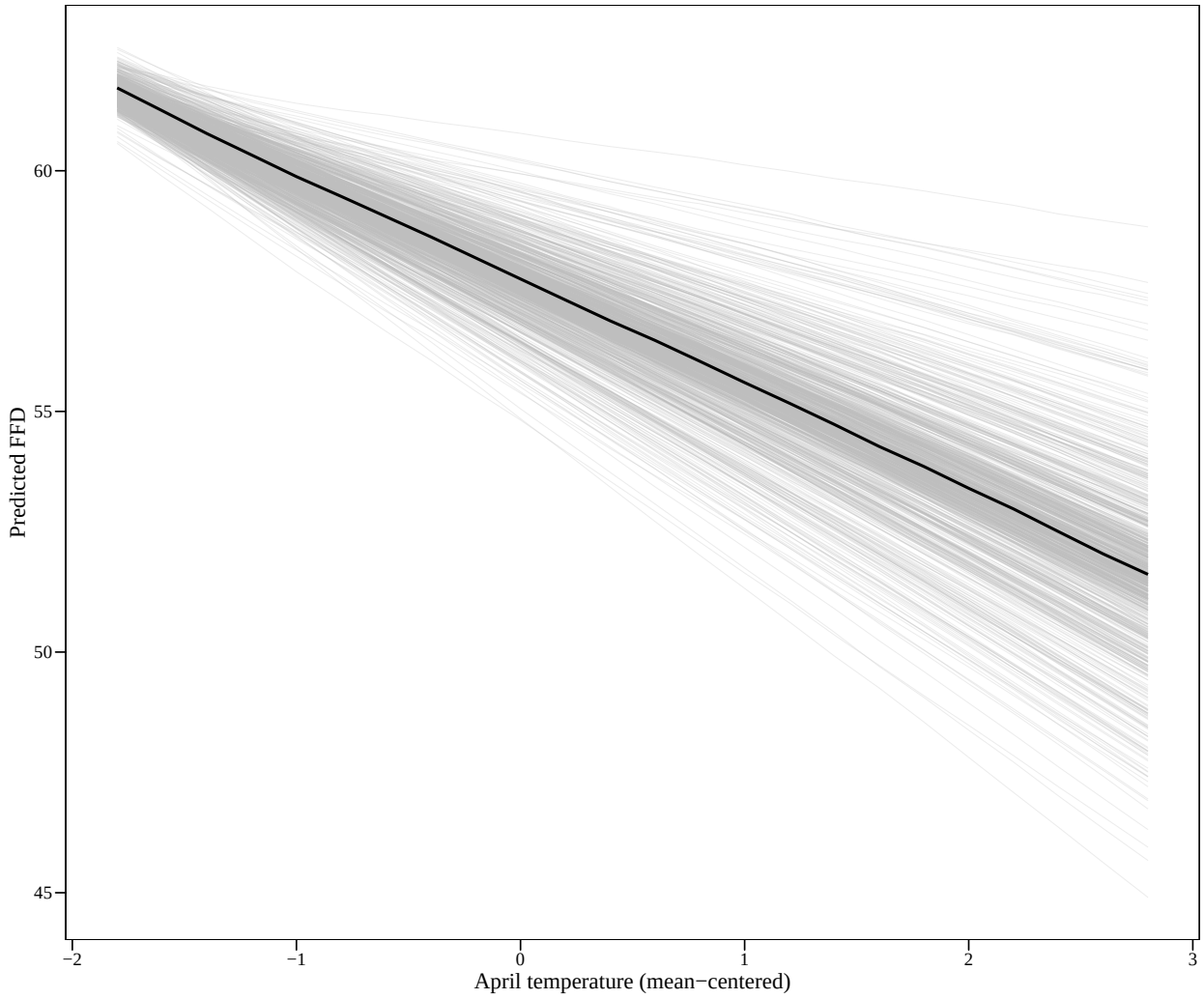
```
datadef<-datadef%>%left_join(BLUPs_MCMC.all.brms%>%
                               select(intercept,slope)%>%
                               rownames_to_column(var="id"))
```

Figure 1

```
predict_id_2<-ggpredict(univar.FFD_RR.all.brm,
                        terms=c("cmean_4","id"),type="random")
predict_mean_2<-ggpredict(univar.FFD_RR.all.brm,terms=c("cmean_4"),
                          type="random")
```

```
ggplot()+
  geom_line(data=data.frame(predict_id_2),aes(x=x,y=predicted,group=group),
            color="grey",size=0.01,alpha=0.3)+
  geom_line(data=data.frame(predict_mean_2),aes(x=x,y=predicted),
            color="black",size=1)+
  my_theme()+
```

```
  xlab("April temperature (mean-centered)")+
  ylab("Predicted FFD")
```




```
ggsave(filename="output/figures/fig1.tiff",
       device="tiff",width=12,height=10,units="cm",dpi=300,compression="lzw")
```

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 using a negative binomial distribution. Results seem quite similar.Maybe we should consider trying a zero-inflated negative binomial (I did some attempts but seems to take much longer!).

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)
```

Save data with BLUPs

```
write_csv(datadef,file = "data/datadef_BLUPs.csv")
```

```
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)
```

```
print(bivar1.all.brm.nb,digits=3)
```

```
##  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)                                 1.637     0.149    1.340
## sd(FFD_cmean_4)                                   0.787     0.133    0.528
## sd(roundmeanfitnessfl_Intercept)                  1.379     0.052    1.280
## cor(FFD_Intercept,FFD_cmean_4)                    0.688     0.138    0.385
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)  -0.525     0.074   -0.667
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)    -0.046     0.138   -0.320
##                                                u-95% CI  Rhat Bulk_ESS
## sd(FFD_Intercept)                                 1.927 1.002     1730
```

```
## sd(FFD_cmean_4)                                     1.052 1.004     1920
## sd(roundmeanfitnessfl_Intercept)                    1.485 1.002     3083
## cor(FFD_Intercept,FFD_cmean_4)                      0.916 1.012      409
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)    -0.377 1.002     1113
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)       0.216 1.002      788
##                                                  Tail_ESS
## sd(FFD_Intercept)                                    3232
## sd(FFD_cmean_4)                                      4218
## sd(roundmeanfitnessfl_Intercept)                     4380
## cor(FFD_Intercept,FFD_cmean_4)                       2057
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)      2049
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)        1232
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)    5.090     0.839    3.760    7.039 1.001     2897     3957
##
## Population-Level Effects:
##                              Estimate Est.Error l-95% CI u-95% CI  Rhat
## FFD_Intercept                  58.756     1.101   56.605   61.005 1.001
## roundmeanfitnessfl_Intercept    0.759     0.056    0.647    0.867 1.000
## FFD_cmean_4                    -2.354     0.817   -3.938   -0.690 1.000
##                              Bulk_ESS Tail_ESS
## FFD_Intercept                    2173     3260
## roundmeanfitnessfl_Intercept     5297     5417
## FFD_cmean_4                      2504     3962
##
## Family Specific Parameters:
##                           Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS
## sigma_FFD                    4.349     0.073    4.203    4.498 1.002     1640
## shape_roundmeanfitnessfl   670.434   188.304  379.864 1125.083 1.000     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).
```

Negative binomial distribution with zero-inflation

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

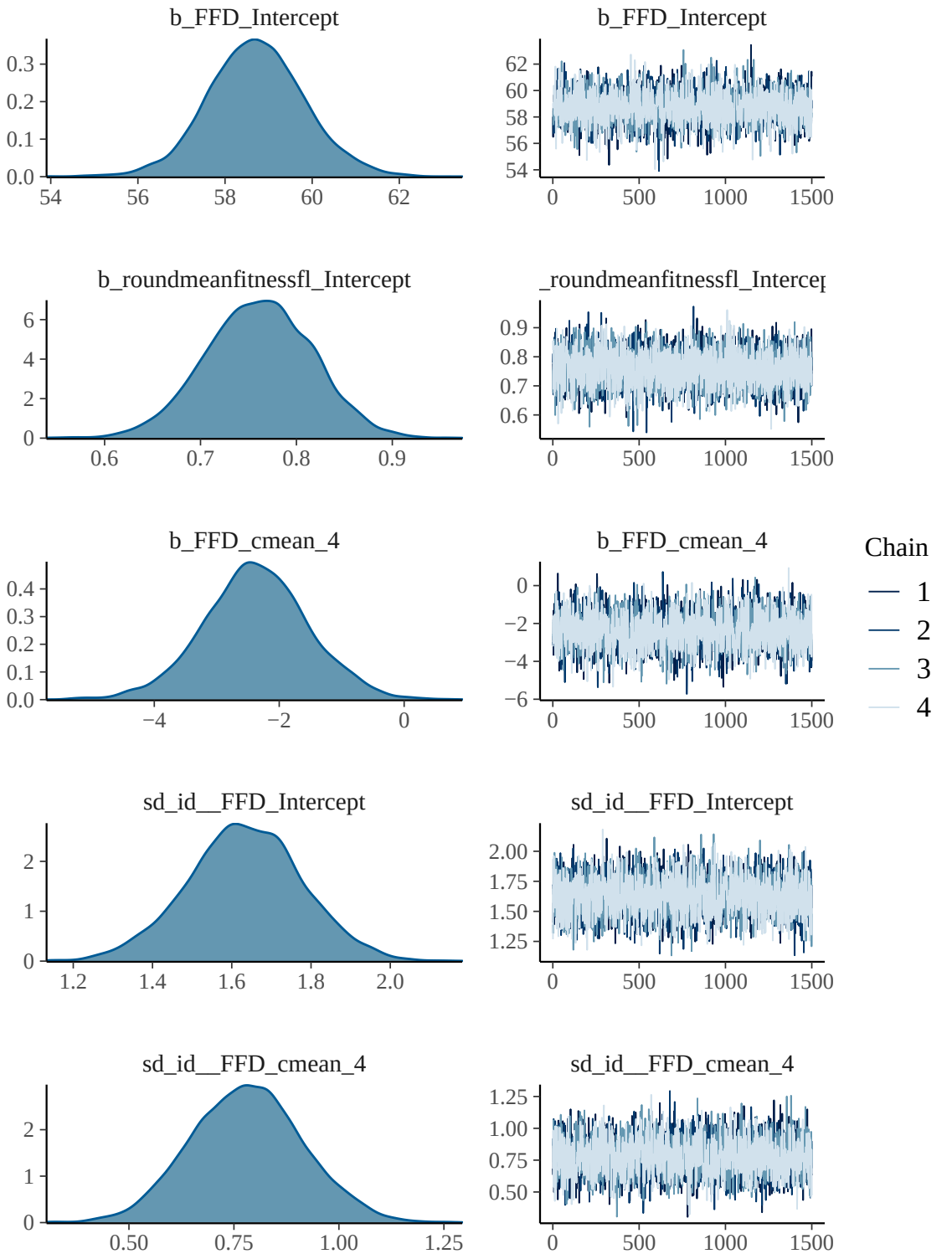
```
summary(bivar1.all.brm.zinb)
```

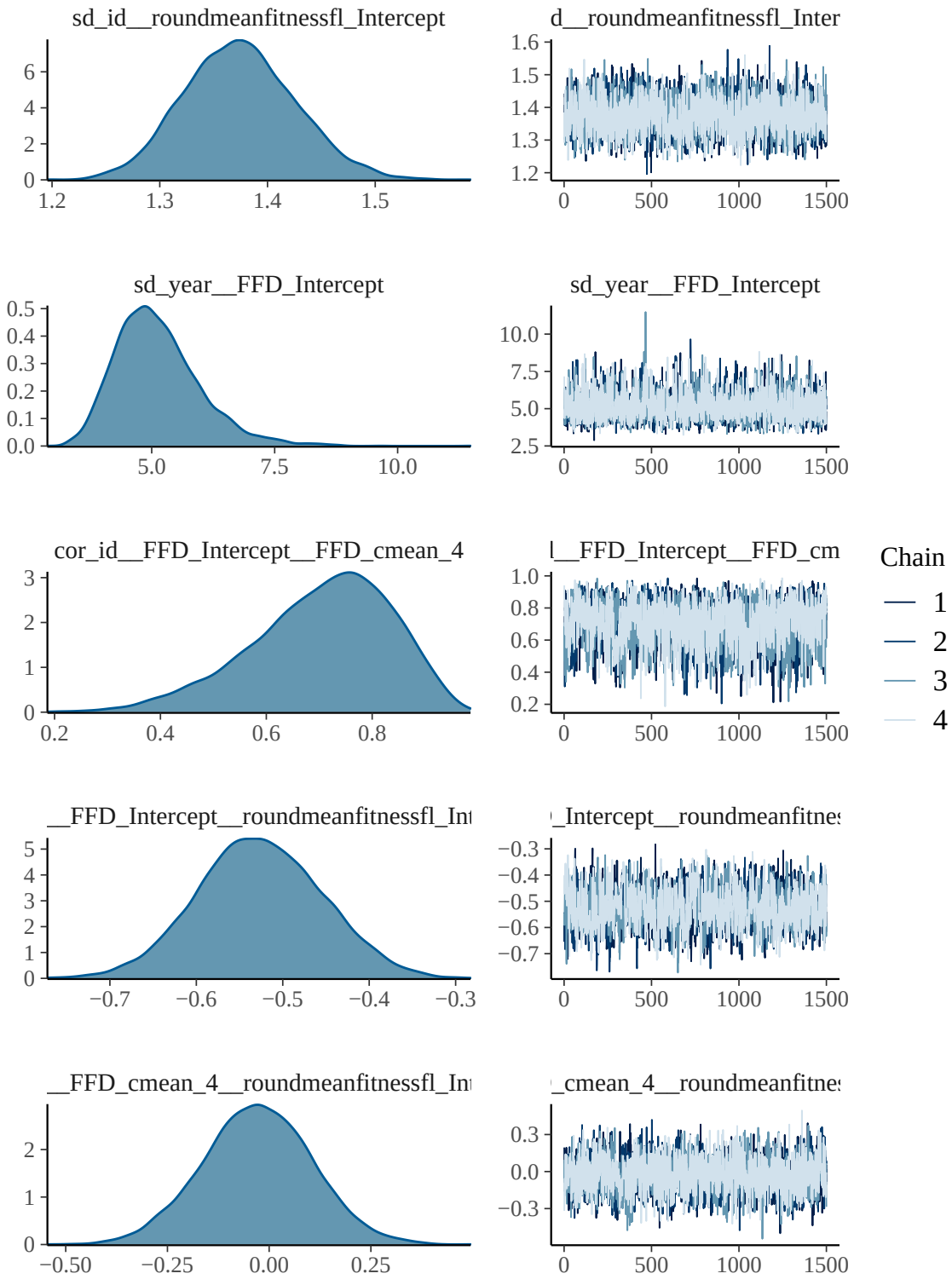
```
##  Family: MV(gaussian, zero_inflated_negbinomial)
##   Links: mu = identity; sigma = identity
##          mu = log; shape = identity; zi = identity
```

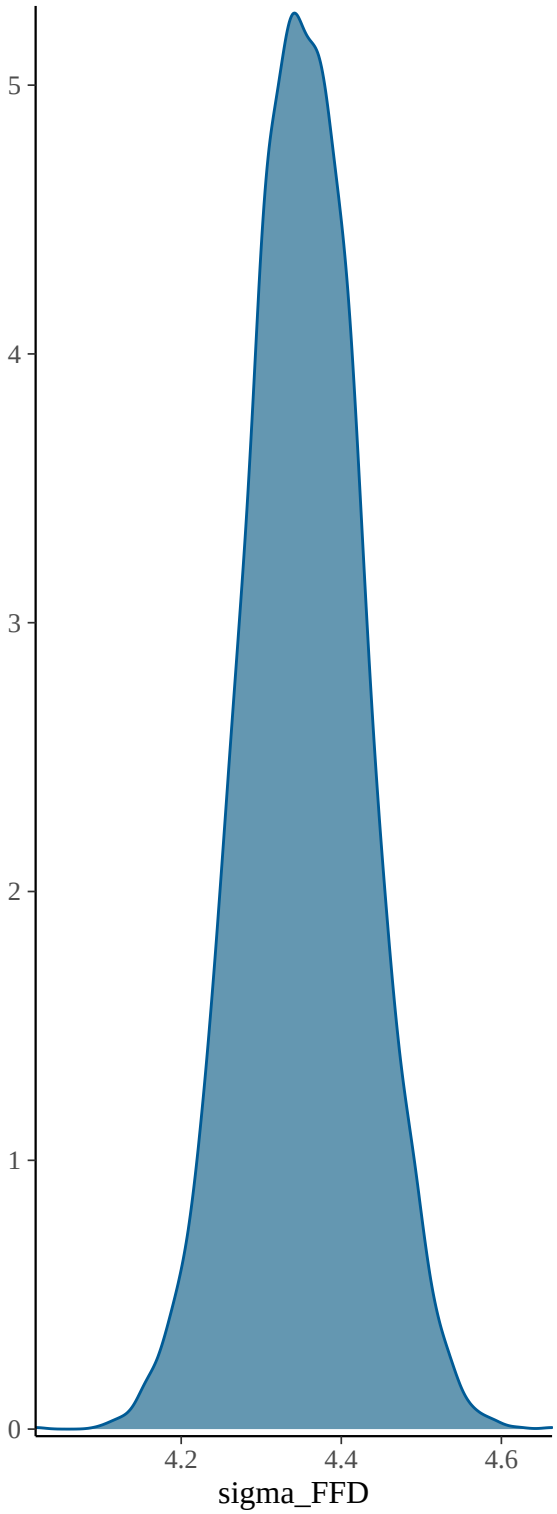
```
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_fl) ~ (1 | ID1 | id)
##    Data: datadef (Number of observations: 2478)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 2;
##          total post-warmup samples = 2000
##
## Group-Level Effects:
## ~id (Number of levels: 837)
##                                                 Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                   1.63      0.15     1.36
## sd(FFD_cmean_4)                                     0.78      0.13     0.51
## sd(roundmeanfitnessfl_Intercept)                    1.37      0.05     1.28
## cor(FFD_Intercept,FFD_cmean_4)                      0.71      0.13     0.43
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)    -0.53      0.07    -0.67
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)      -0.03      0.14    -0.29
##                                                 u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)                                   1.93 1.01      990     1446
## sd(FFD_cmean_4)                                     1.05 1.01      802     1752
## sd(roundmeanfitnessfl_Intercept)                    1.47 1.00     1059     1660
## cor(FFD_Intercept,FFD_cmean_4)                      0.92 1.04      138      611
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)    -0.39 1.01      529      799
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)       0.24 1.01      603      678
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.10      0.89     3.74     7.16 1.00     1067     1548
##
## Population-Level Effects:
##                              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## FFD_Intercept                   58.73      1.12    56.44    60.97 1.01      796
## roundmeanfitnessfl_Intercept     0.76      0.06     0.65     0.87 1.00     1827
## FFD_cmean_4                     -2.40      0.80    -4.05    -0.84 1.00     1212
##                              Tail_ESS
## FFD_Intercept                    1173
## roundmeanfitnessfl_Intercept     1566
## FFD_cmean_4                      1450
##
## 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 1.01      802
## shape_roundmeanfitnessfl   663.74    180.89   377.98  1084.85 1.00     1513
## zi_roundmeanfitnessfl        0.00      0.00     0.00     0.00 1.00     1755
##                          Tail_ESS
## sigma_FFD                    1465
## shape_roundmeanfitnessfl     1408
## zi_roundmeanfitnessfl        1692
##
## 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).
```

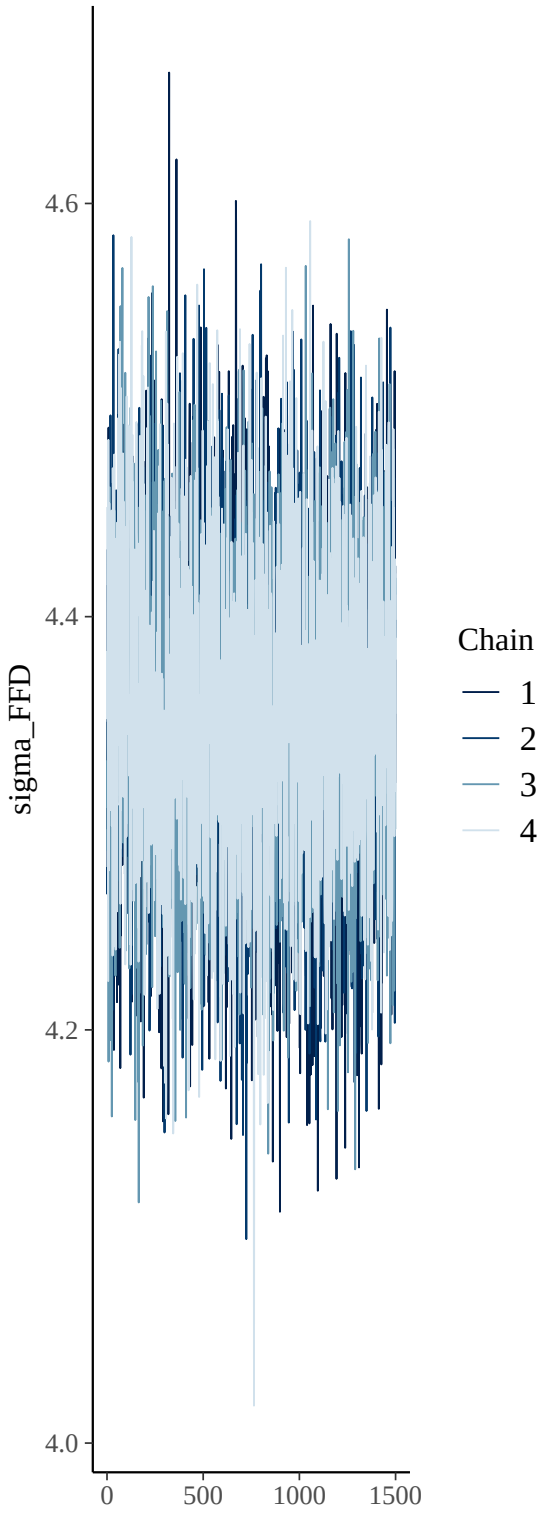
Model evaluation: Compare models

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

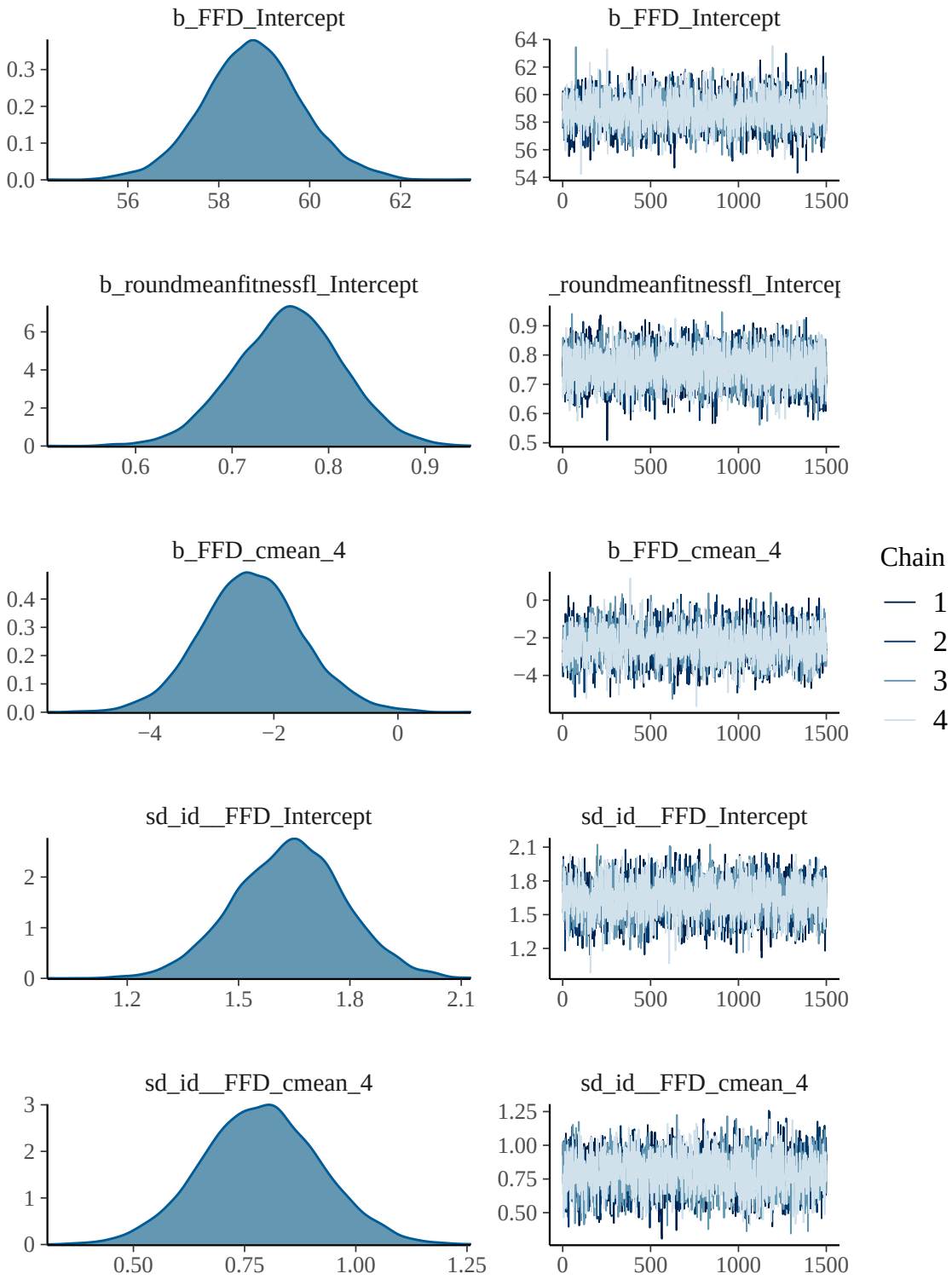


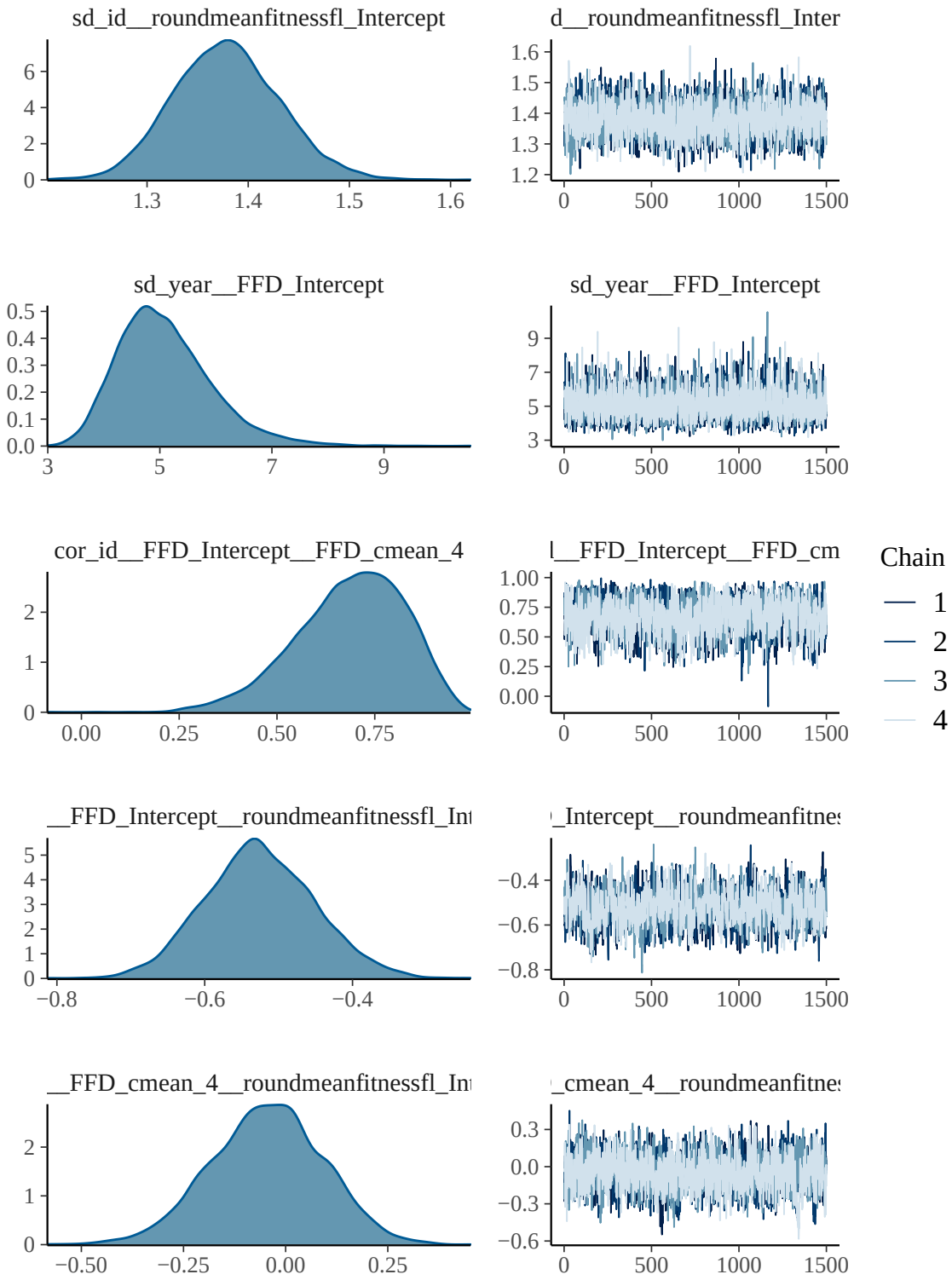

sd_id__roundmeanfitnessfl_Intercept
d__roundmeanfitnessfl_Inter
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
__FFD_Intercept__roundmeanfitnessfl_Int
_Intercept__roundmeanfitnes
__FFD_cmean_4__roundmeanfitnessfl_Int
_cmean_4__roundmeanfitnes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

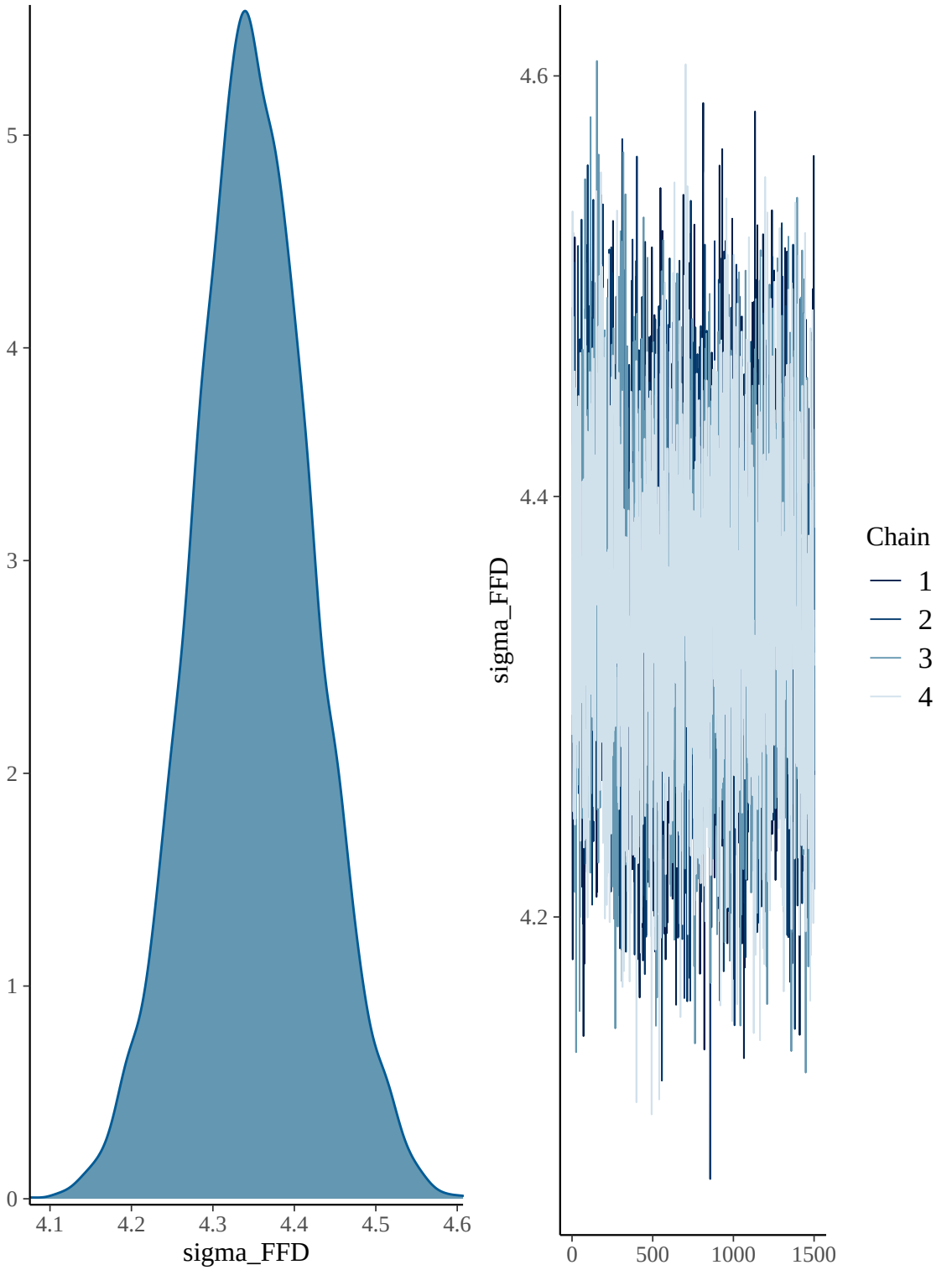




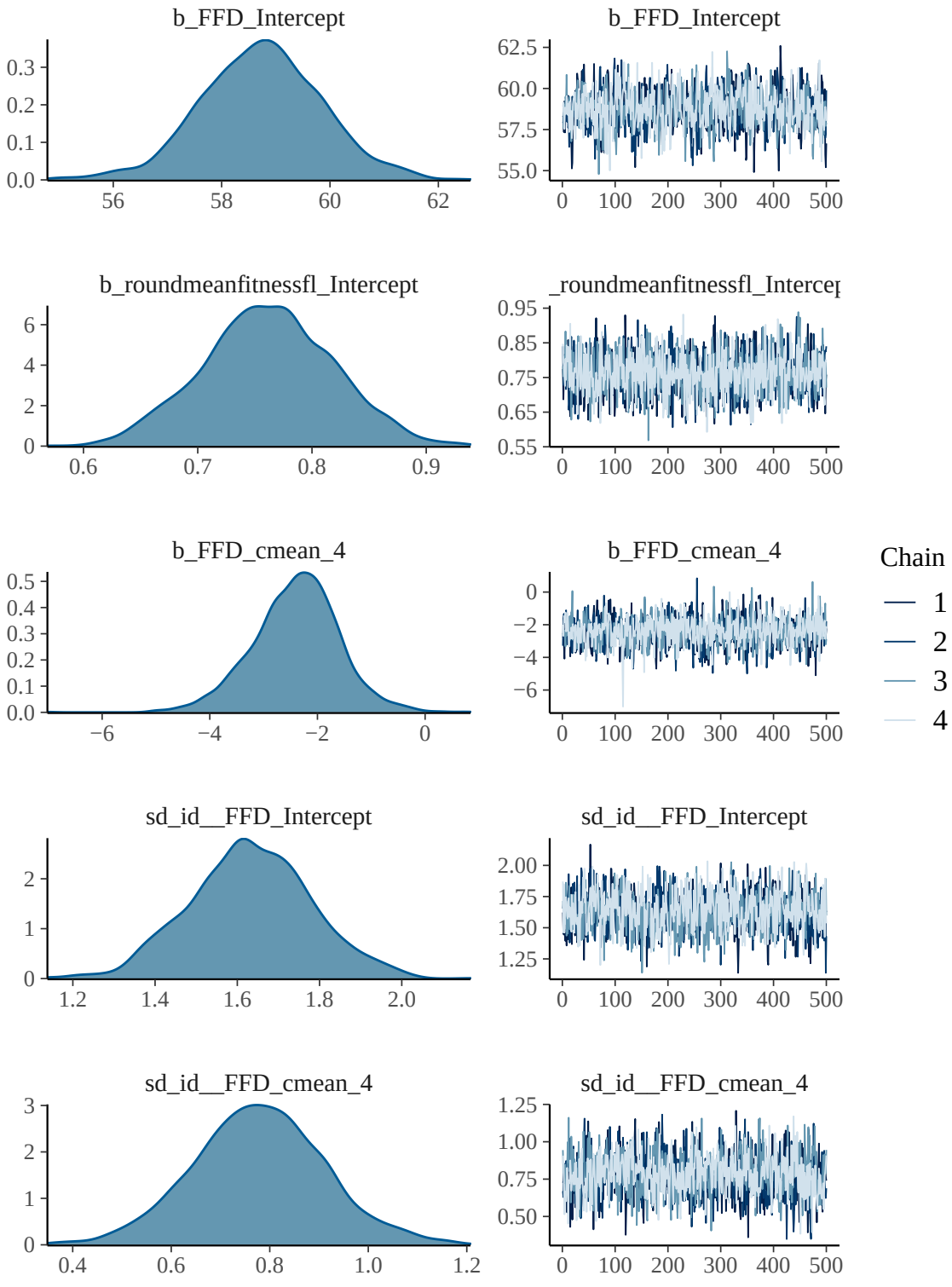

```
plot(bivar1.all.brm.nb)
```

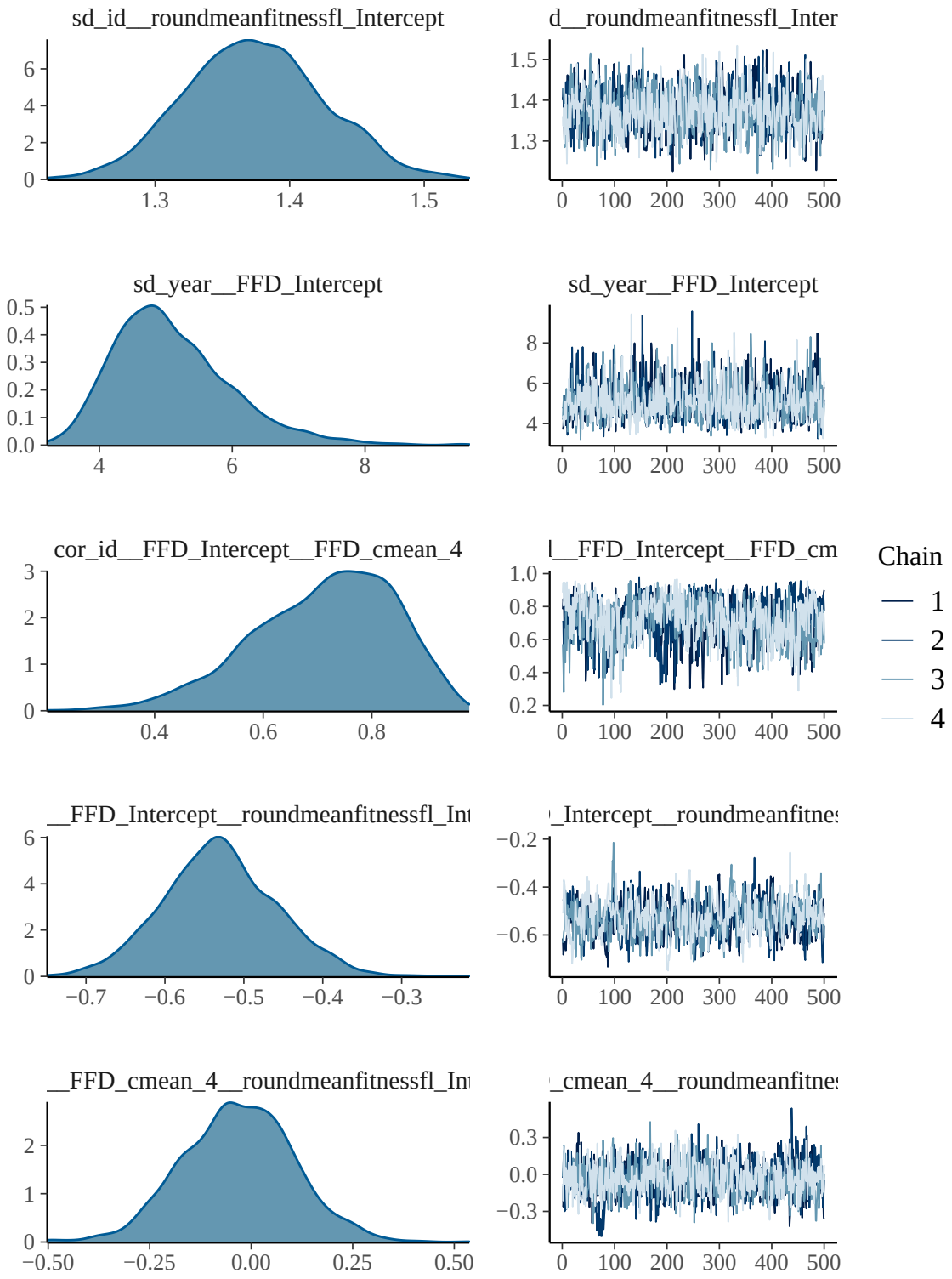

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessfl_Intercept
_roundmeanfitnessfl_Intercep
b_FFD_cmean_4
b_FFD_cmean_4
Chain
1
2
3
4
sd_id__FFD_Intercept
sd_id__FFD_Intercept
sd_id__FFD_cmean_4
sd_id__FFD_cmean_4

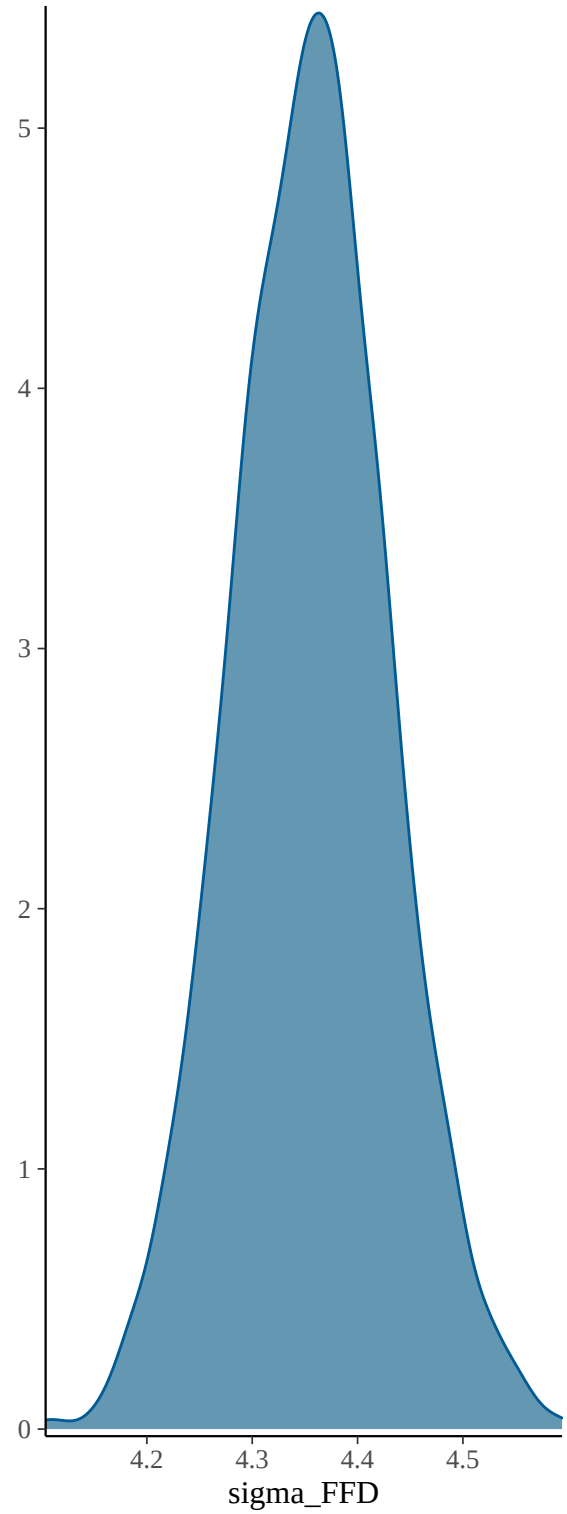

sd_id__roundmeanfitnessfl_Intercept
d__roundmeanfitnessfl_Inter
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
__FFD_Intercept__roundmeanfitnessfl_Int
_Intercept__roundmeanfitnes
__FFD_cmean_4__roundmeanfitnessfl_Int
_cmean_4__roundmeanfitnes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

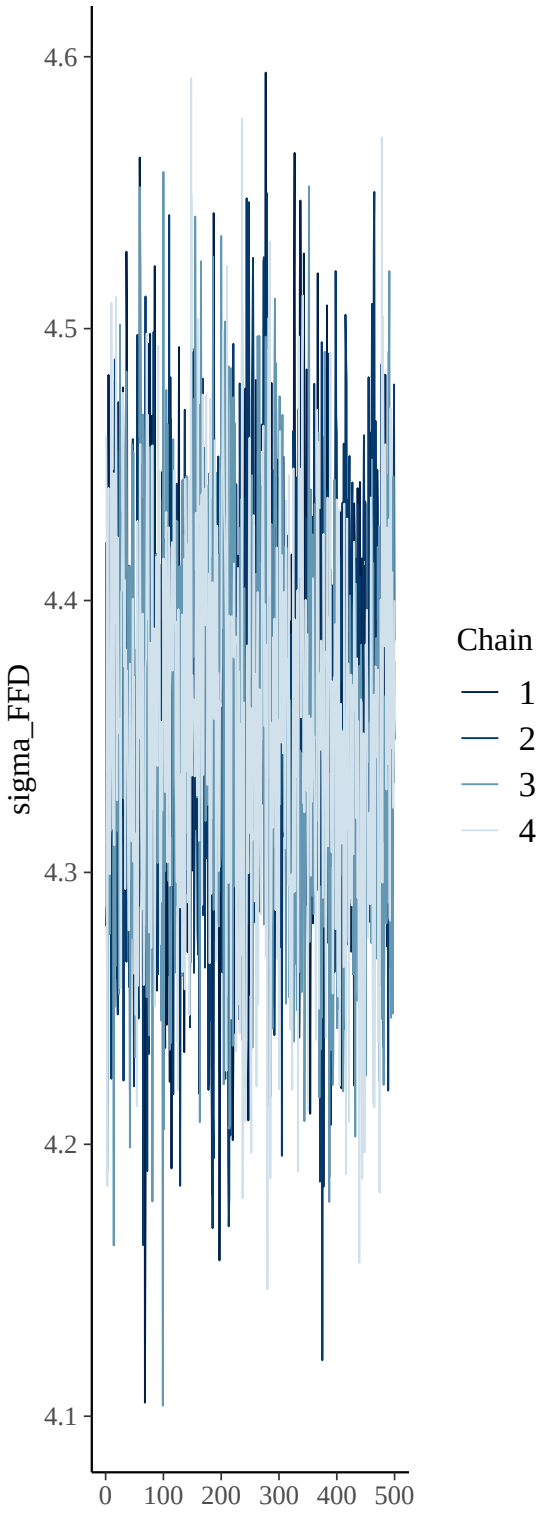



```
plot(bivar1.all.brm.zinb)
```


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

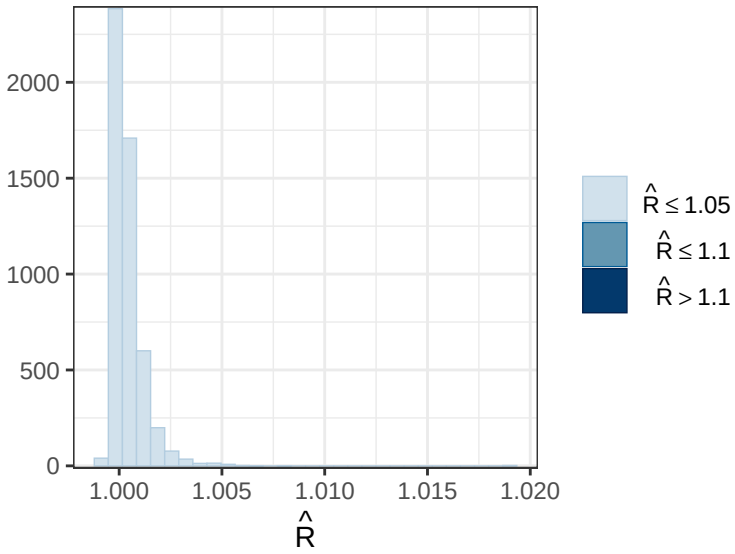

sd_id__roundmeanfitnessfl_Intercept
d__roundmeanfitnessfl_Inter
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
__FFD_Intercept__roundmeanfitnessfl_Int
_Intercept__roundmeanfitnes
__FFD_cmean_4__roundmeanfitnessfl_Int
_cmean_4__roundmeanfitnes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


5
4
3
2
1
0
4.2
4.3
4.4
4.5
sigma_FFD

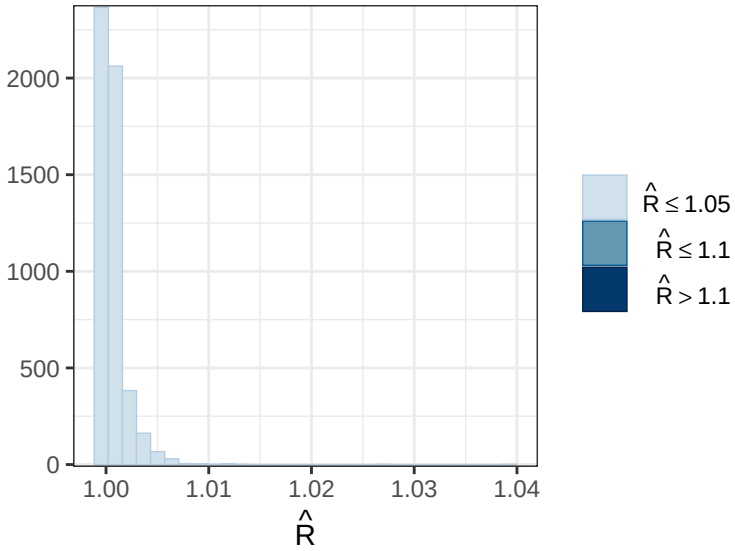

4.6
4.5
4.4
4.3
4.2
4.1
sigma_FFD
0
100
200
300
400
500
Chain
1
2
3
4

Rhat

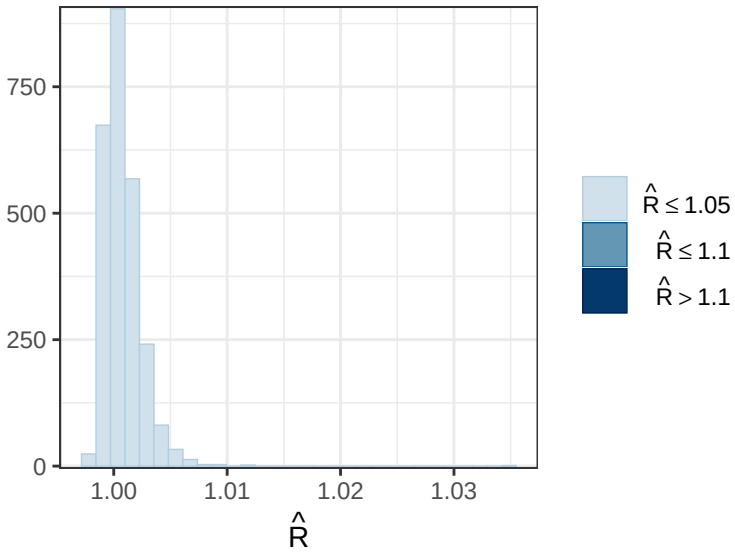
```
mcmc_rhat_hist(rhat(bivar1.all.brm.pois))+theme_bw()
```




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

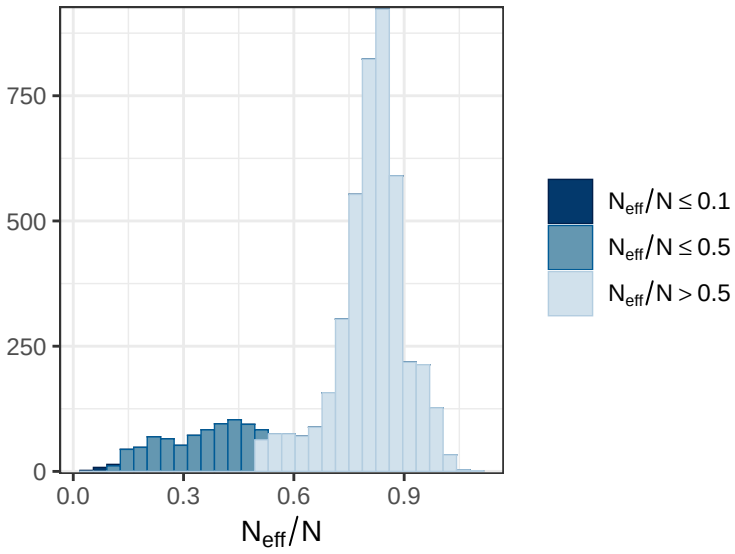



```
mcmc_rhat_hist(rhat(bivar1.all.brm.zinb))+theme_bw()
```

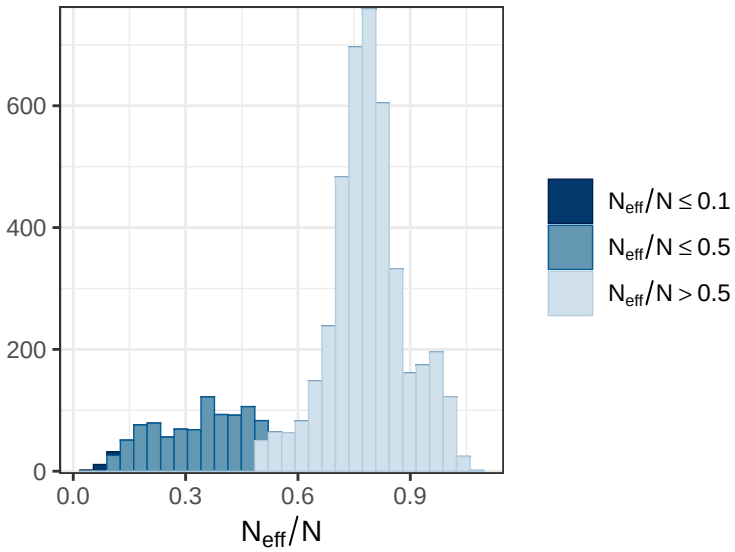


Effective sample size:

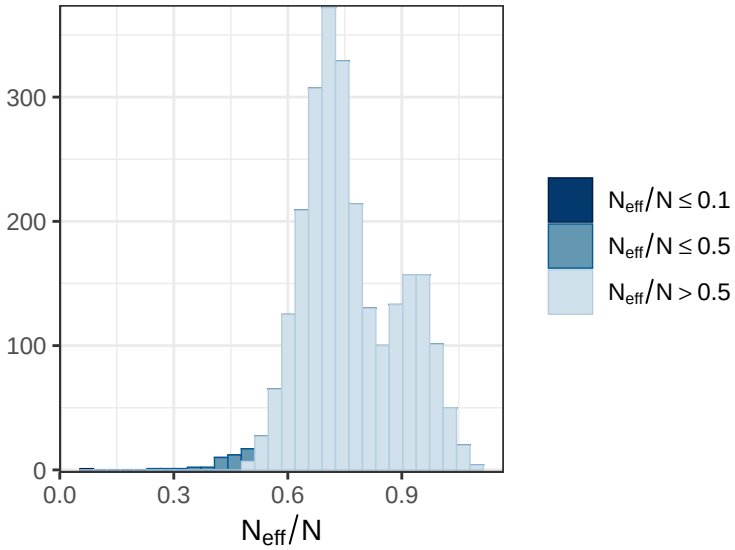
```
mcmc_neff_hist(neff_ratio(bivar1.all.brm.pois))+theme_bw()
```




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




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



Some (few) are < 0.1 - Do we need to worry about this?

Posterior predictive checks:

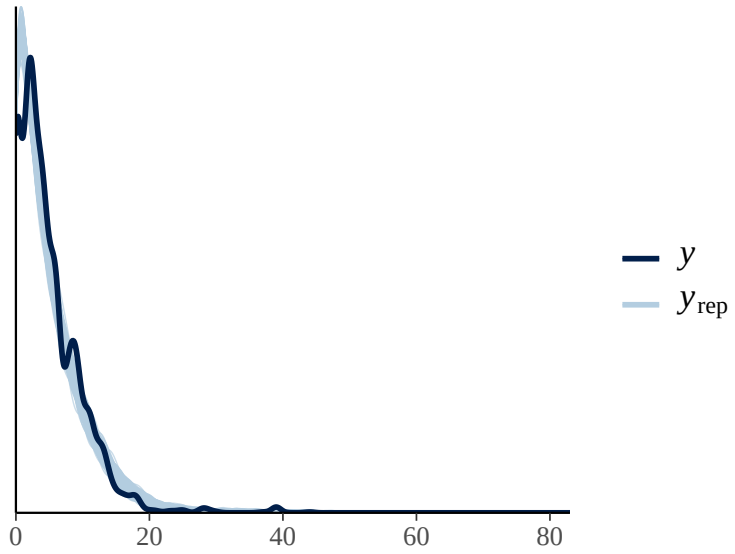
```
y2_fitness<-round(subset(datadef,!is.na(FFD))$mean_fitness_fl)
y2_FFD<-subset(datadef,!is.na(FFD))$FFD # vectors of outcome values
yrep2_fitness_pois<-posterior_predict(bivar1.all.brm.pois,
                                      draws = 500,resp="roundmeanfitnessfl")
yrep2_FFD_pois<-posterior_predict(bivar1.all.brm.pois,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

```
yrep2_fitness_zinb<-posterior_predict(bivar1.all.brm.zinb,
                                      draws = 500,resp="roundmeanfitnessfl")
yrep2_FFD_zinb<-posterior_predict(bivar1.all.brm.zinb,
                                      draws = 500,resp="FFD")
# matrices of draws from the posterior predictive distribution
```

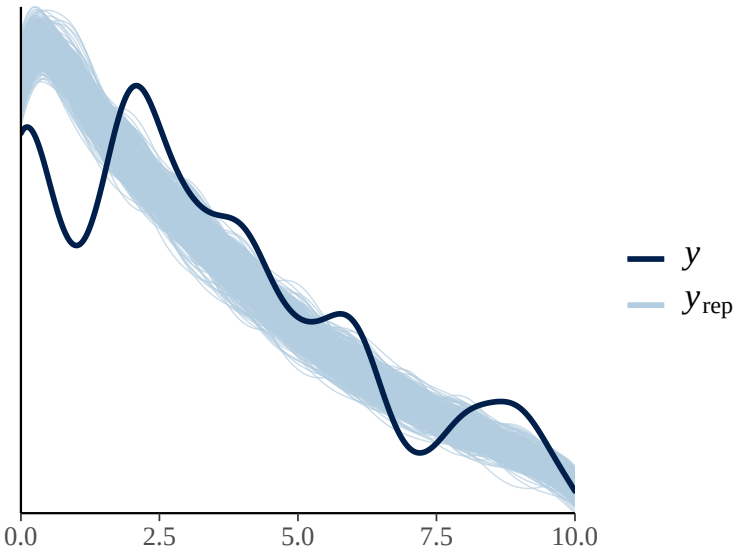
Each row of the matrix is a draw from the posterior predictive distribution, i.e. a vector with one element for each of the data points in y.

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

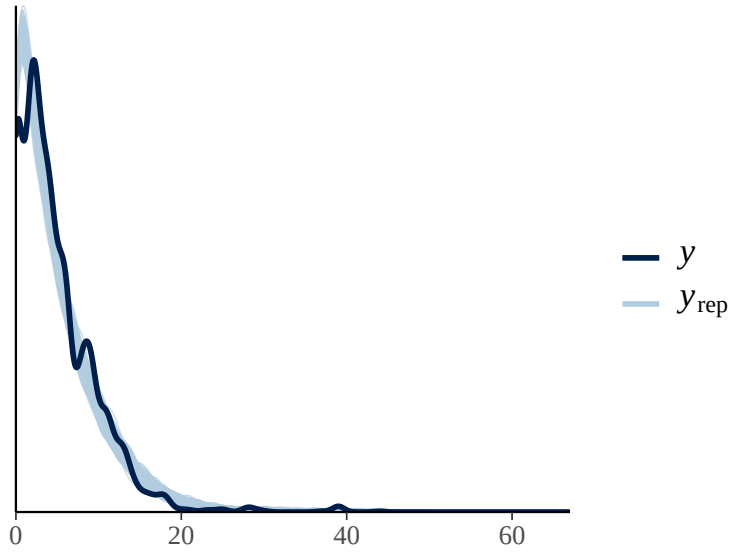
```
ppc_dens_overlay(y2_fitness, yrep2_fitness_pois[1:500,])
```



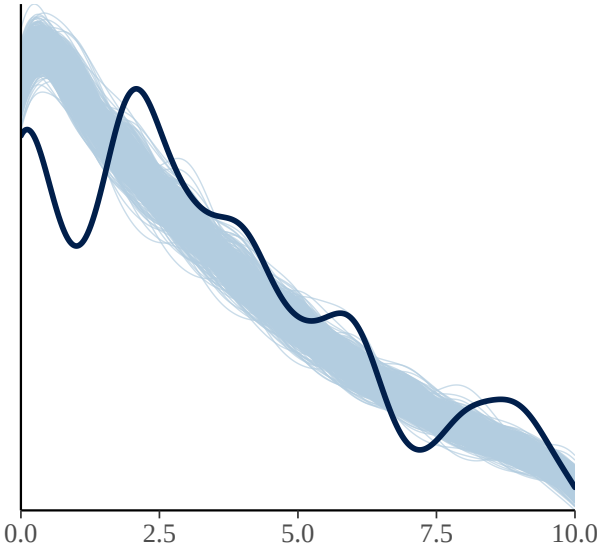

```
ppc_dens_overlay(y2_fitness, yrep2_fitness_pois[1:500,])+xlim(0, 10)
```



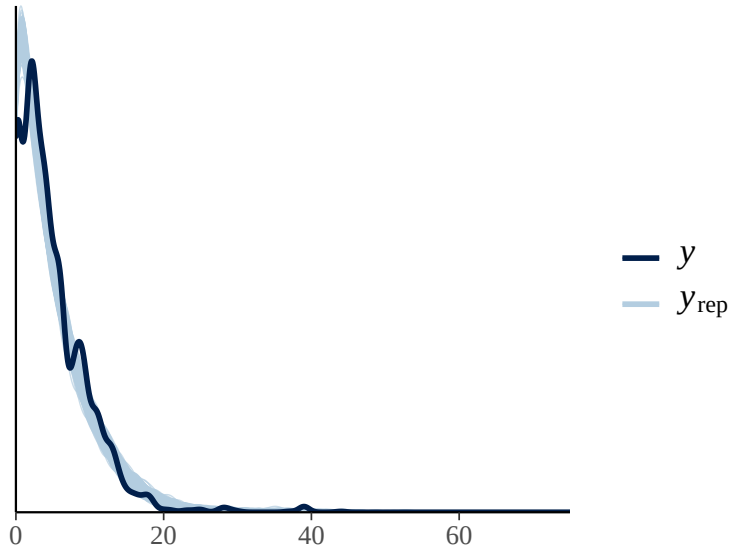
```
ppc_dens_overlay(y2_fitness, yrep2_fitness_nb[1:500,])
```



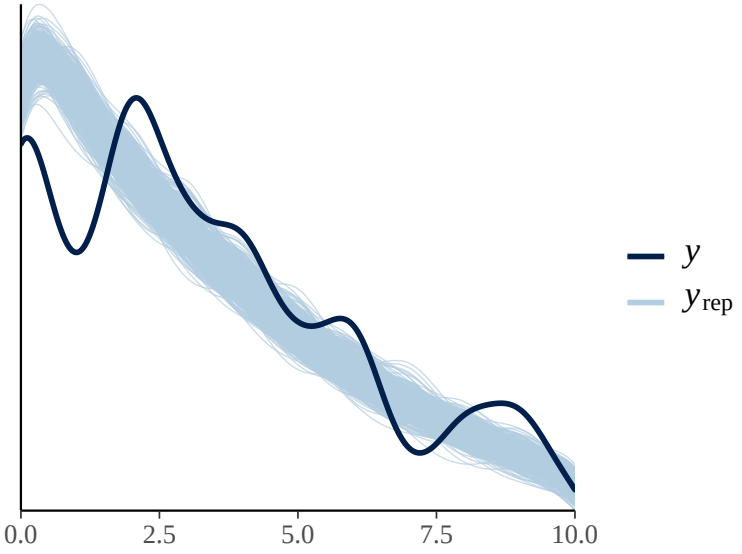

```
ppc_dens_overlay(y2_fitness, yrep2_fitness_nb[1:500,])+xlim(0,10)
```



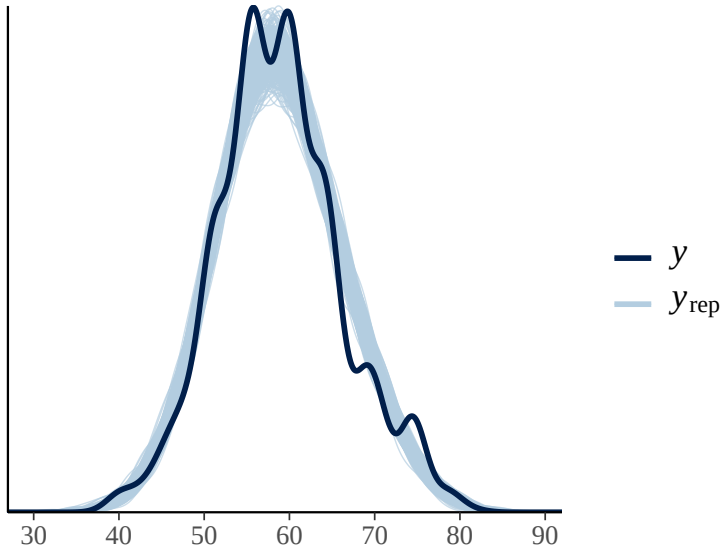

```
ppc_dens_overlay(y2_fitness, yrep2_fitness_zinb[1:500,])
```



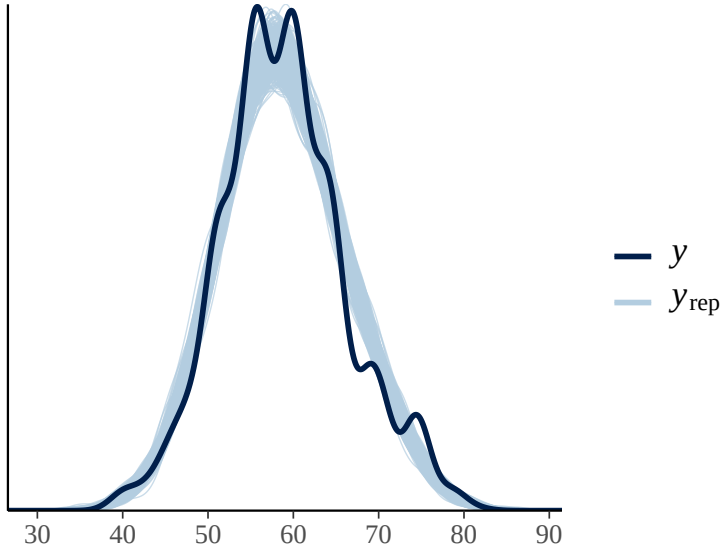

```
ppc_dens_overlay(y2_fitness, yrep2_fitness_zinb[1:500,])+xlim(0,10)
```



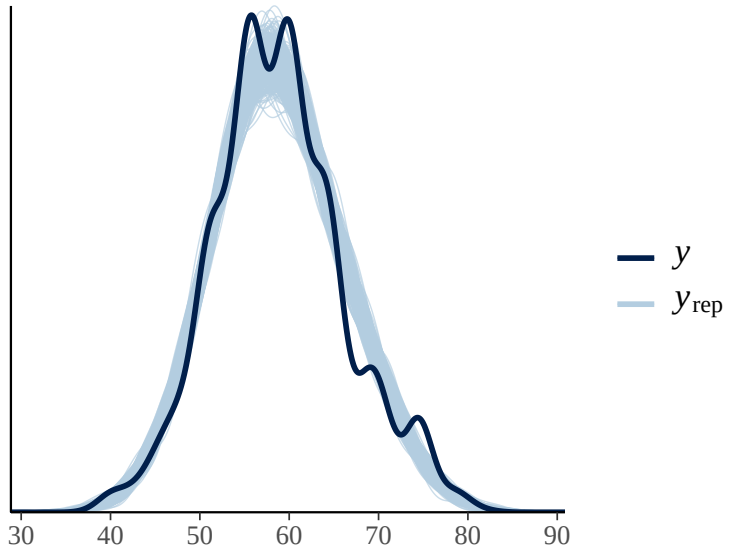

```
ppc_dens_overlay(y2_FFD, yrep2_FFD_pois[1:500,])
```




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



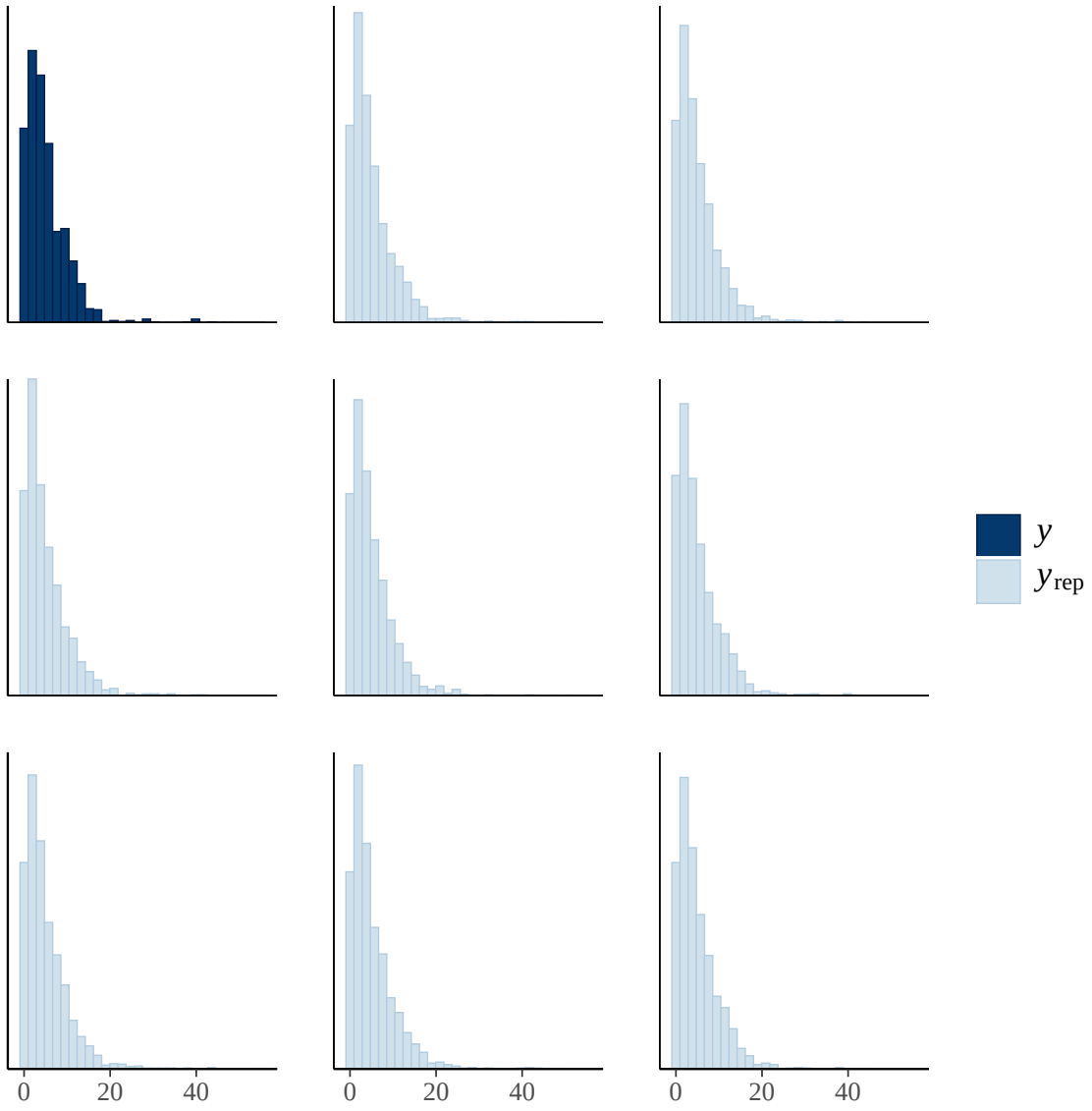

```
ppc_dens_overlay(y2_FFD, yrep2_FFD_zinb[1:500,])
```



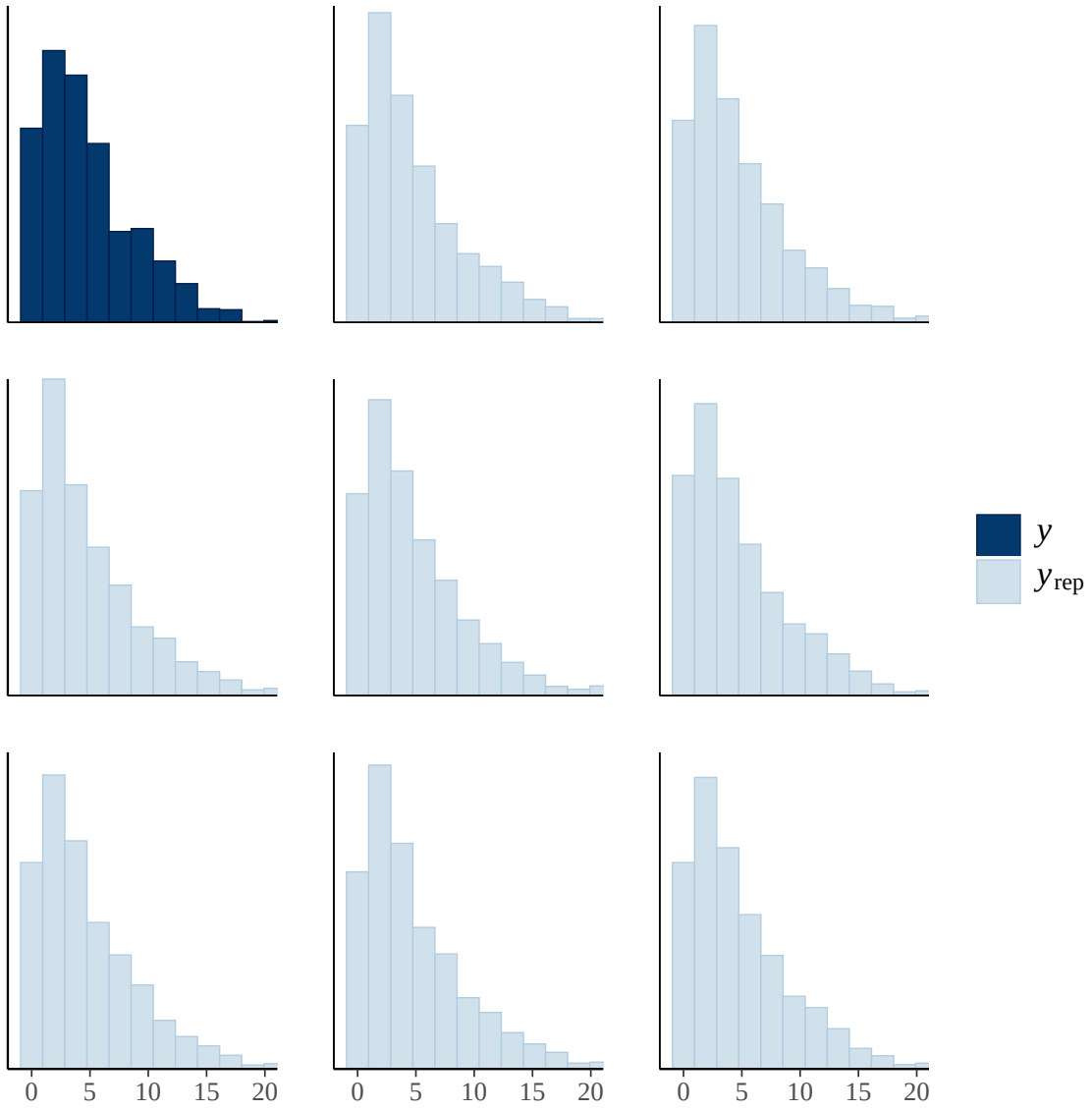
All look pretty similar.

Separate histograms of y and some of the yrep datasets

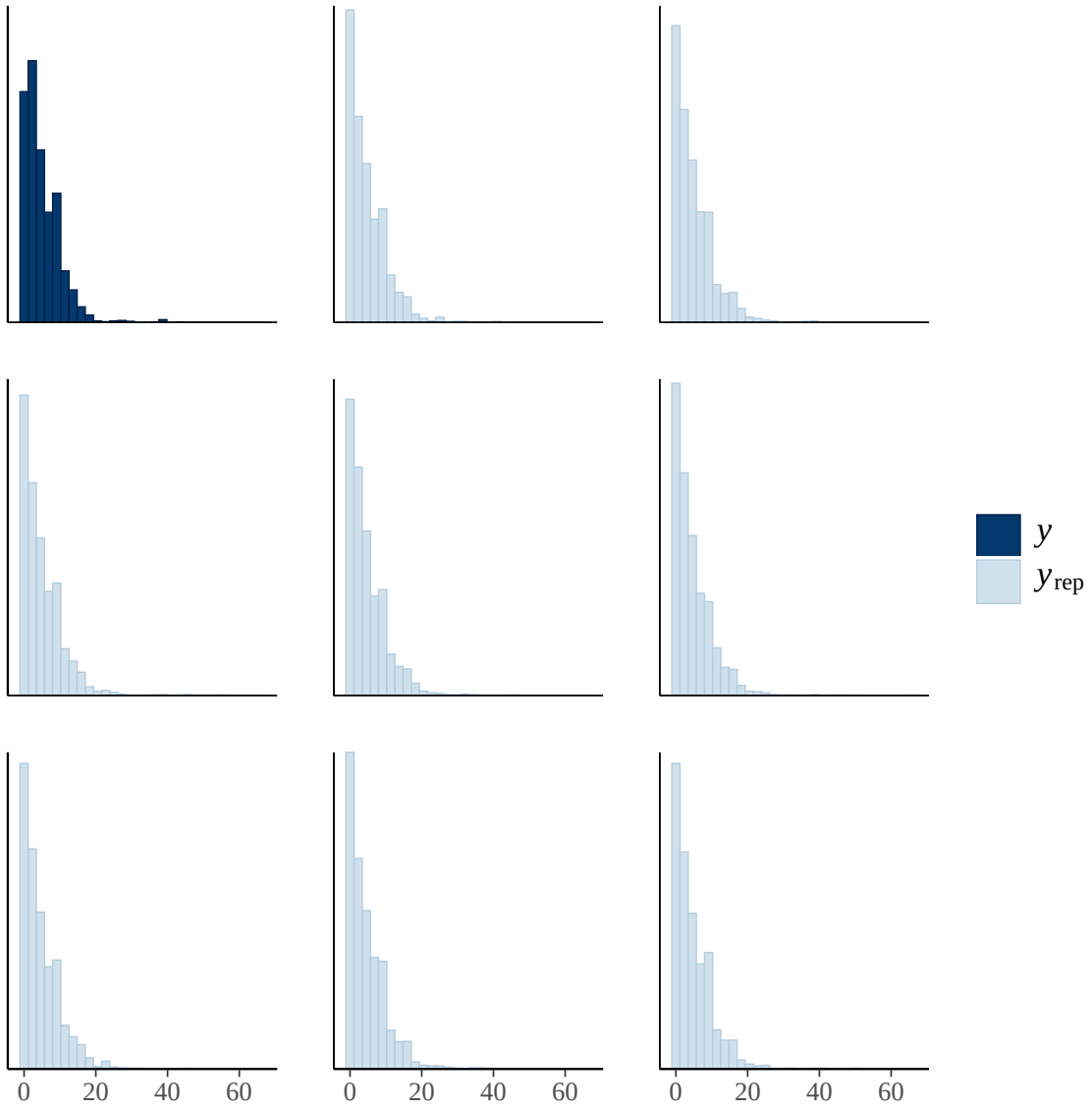
```
ppc_hist(y2_fitness, yrep2_fitness_pois[1:8, ])
```



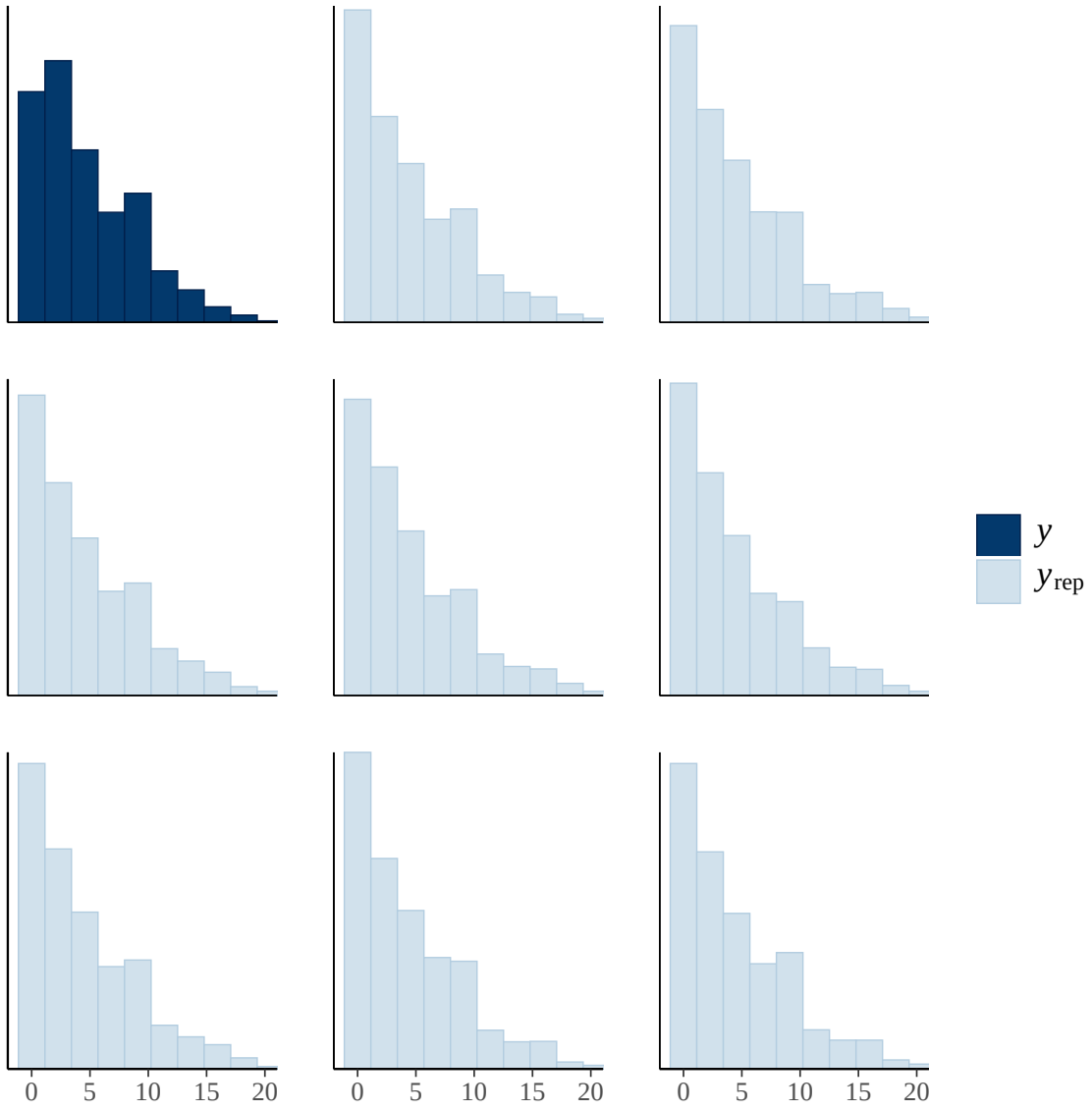

```
ppc_hist(y2_fitness, yrep2_fitness_pois[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```



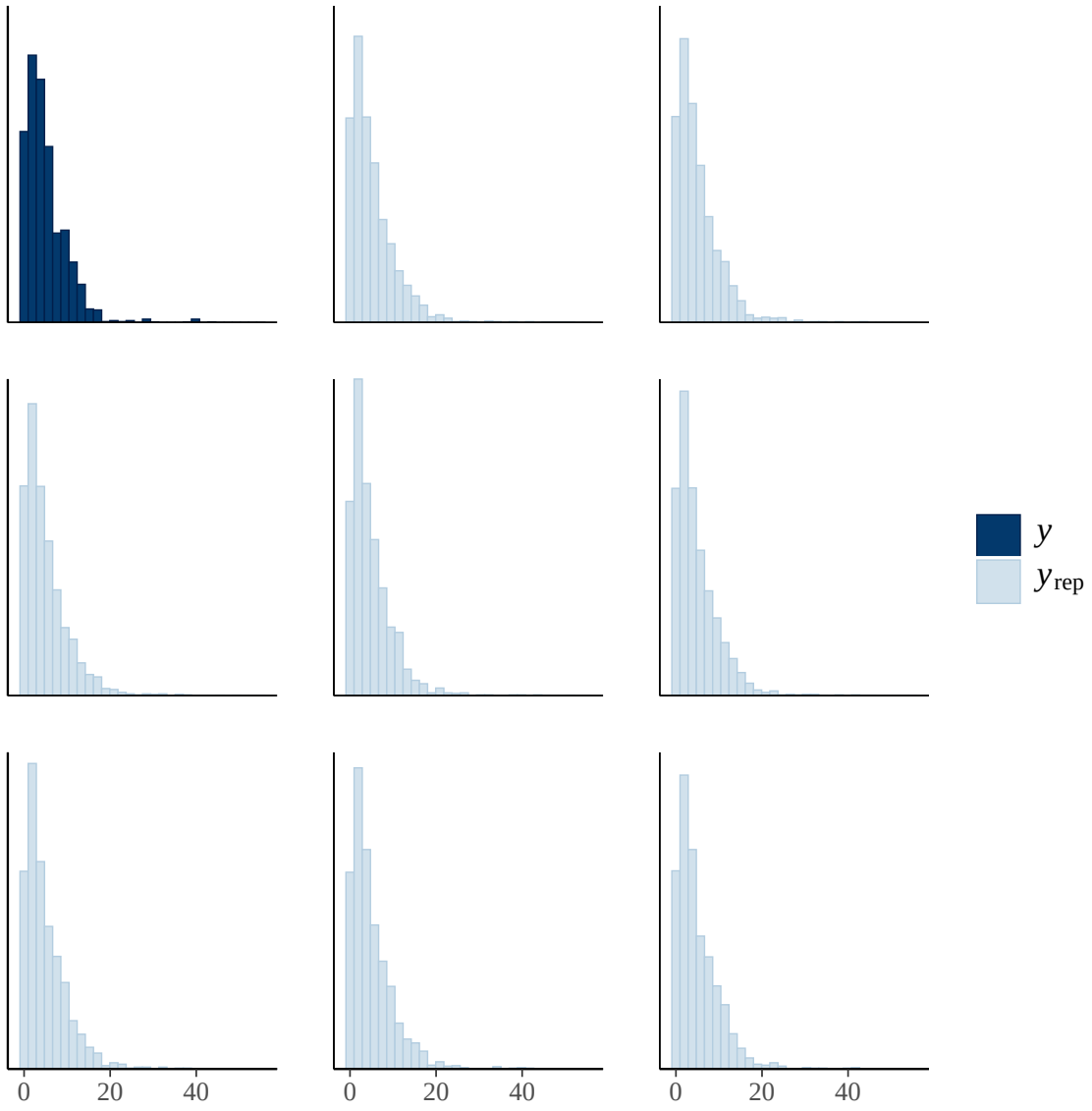

```
ppc_hist(y2_fitness, yrep2_fitness_nb[1:8, ])
```



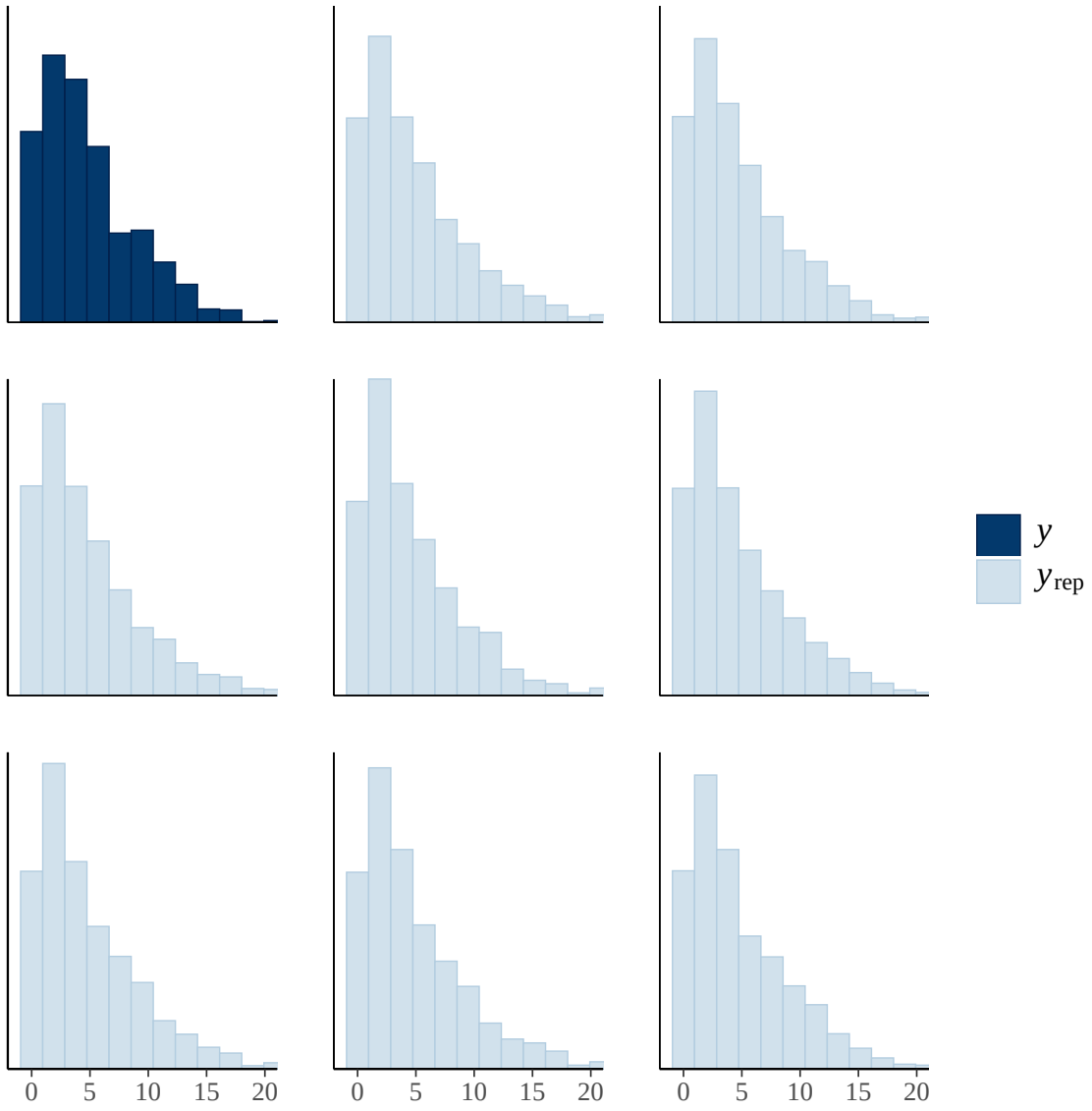

```
ppc_hist(y2_fitness, yrep2_fitness_nb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```




```
ppc_hist(y2_fitness, yrep2_fitness_zinb[1:8, ])
```

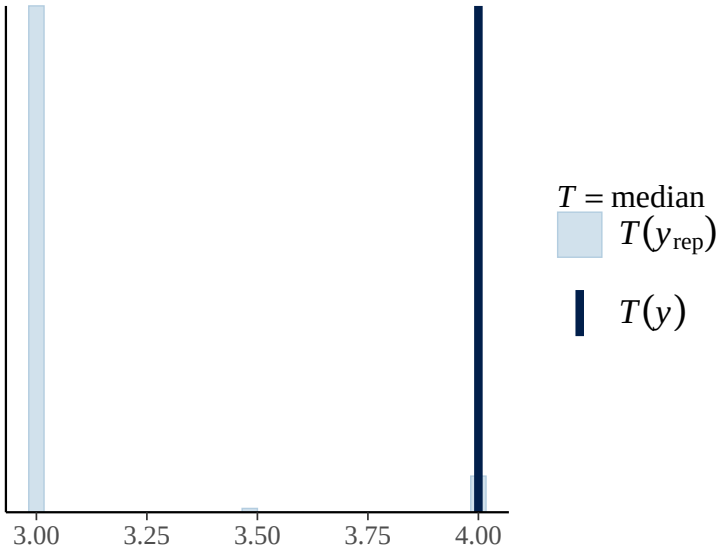



```
ppc_hist(y2_fitness, yrep2_fitness_zinb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```

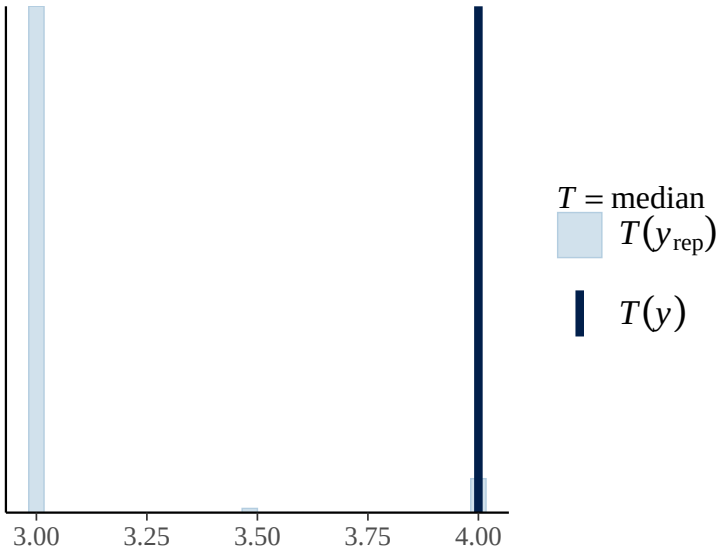


Here, negative binomial looks a bit better for fitness.

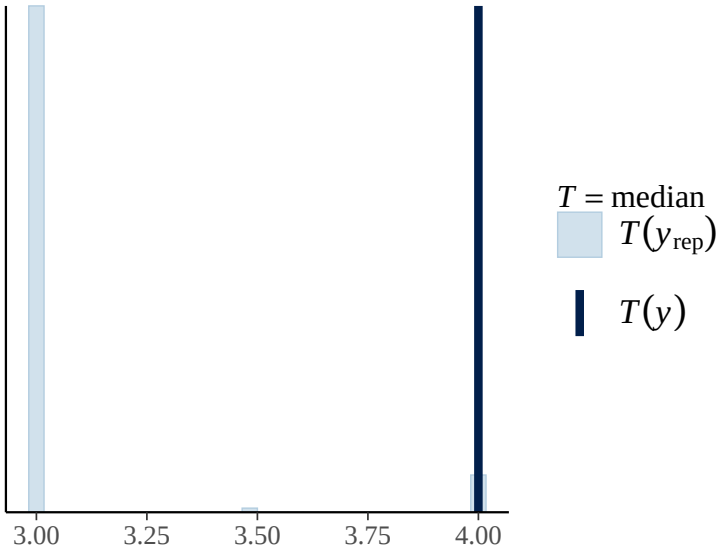
```
ppc_stat(y2_fitness, yrep2_fitness_pois,stat="median")
```



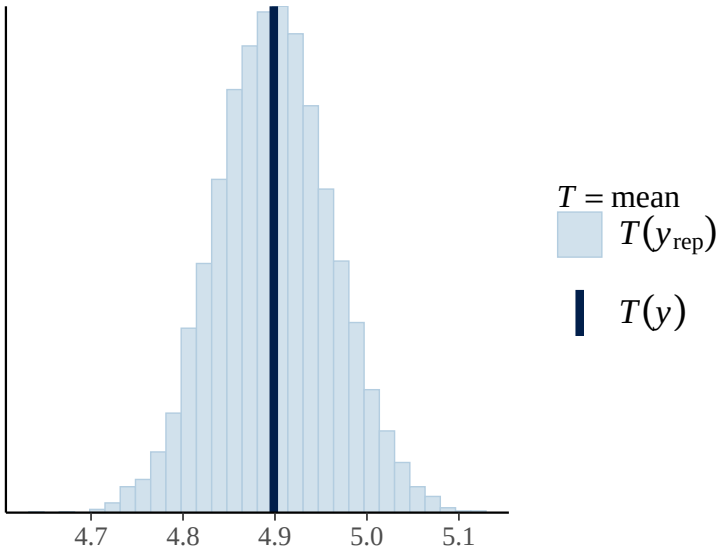

```
ppc_stat(y2_fitness, yrep2_fitness_nb,stat="median")
```



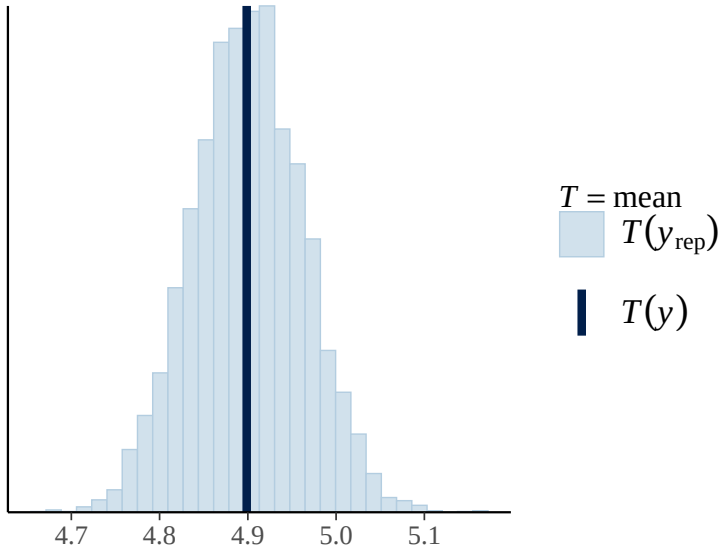

```
ppc_stat(y2_fitness, yrep2_fitness_zinb,stat="median")
```



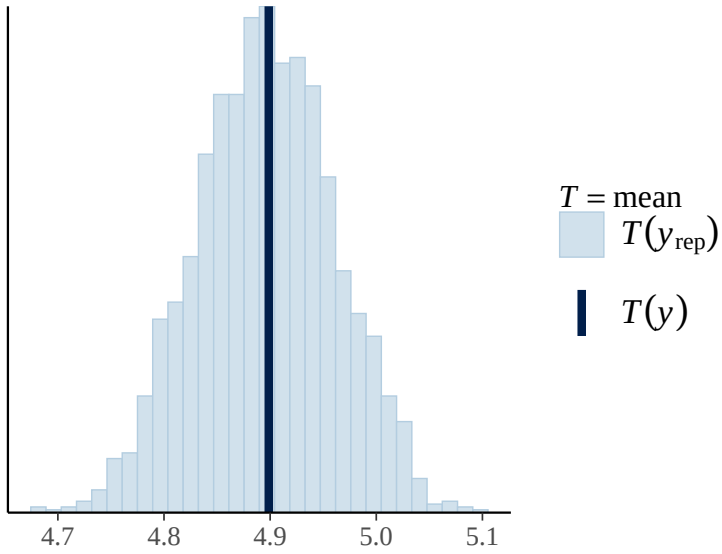

```
ppc_stat(y2_fitness, yrep2_fitness_pois,stat="mean")
```



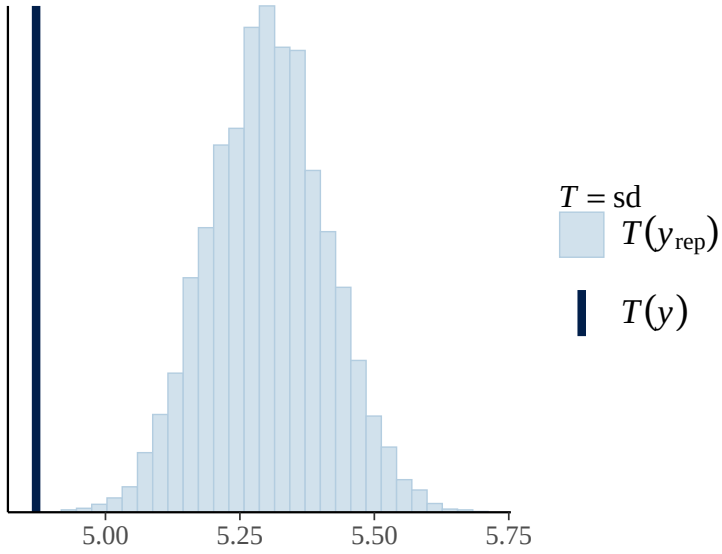

```
ppc_stat(y2_fitness, yrep2_fitness_nb,stat="mean")
```



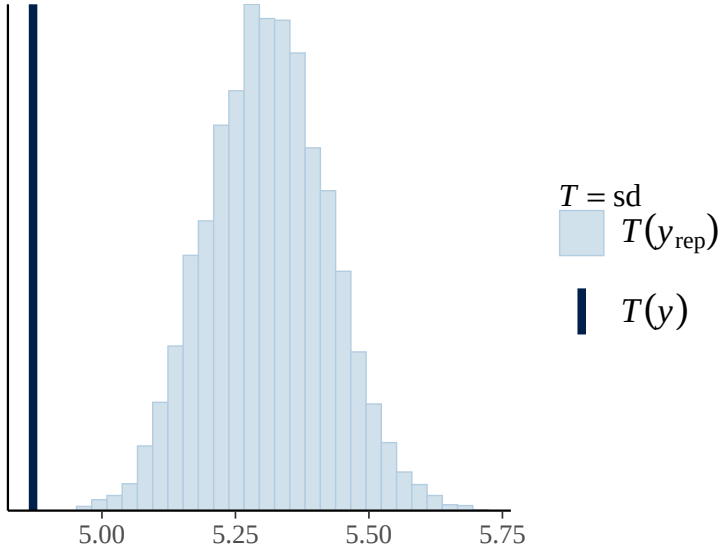

```
ppc_stat(y2_fitness, yrep2_fitness_zinb,stat="mean")
```



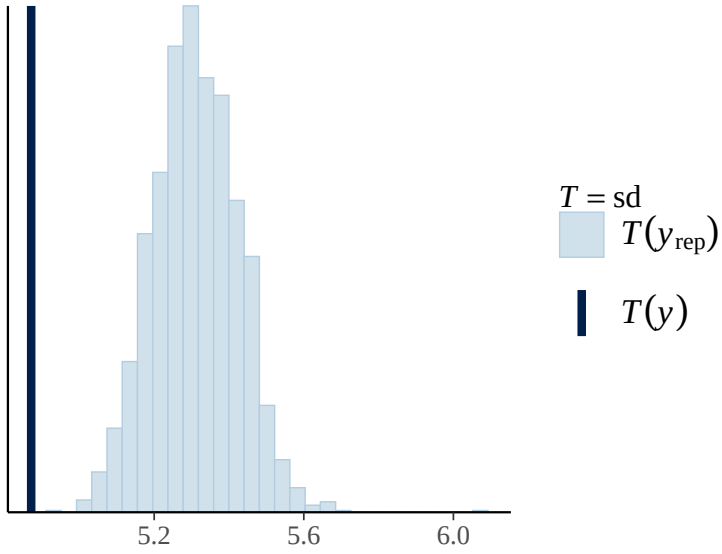

```
ppc_stat(y2_fitness, yrep2_fitness_pois,stat="sd")
```



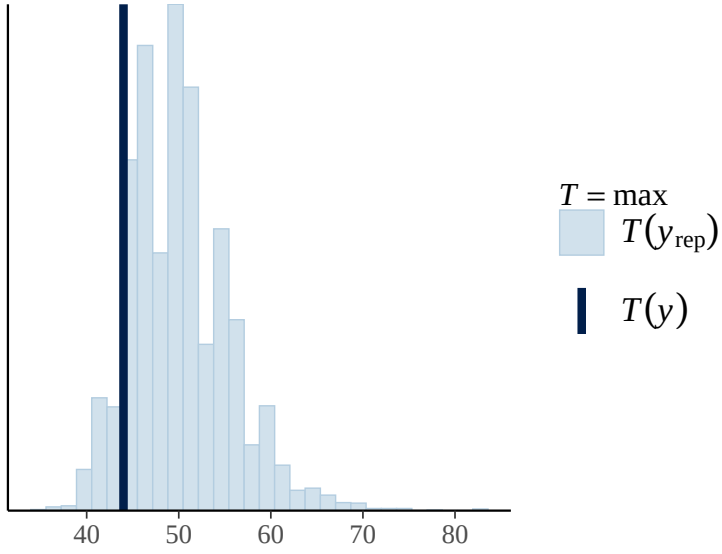

```
ppc_stat(y2_fitness, yrep2_fitness_nb,stat="sd")
```



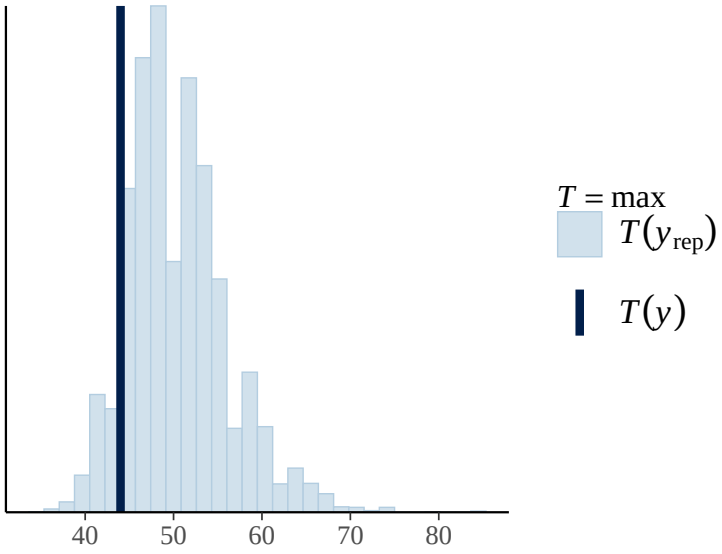

```
ppc_stat(y2_fitness, yrep2_fitness_zinb,stat="sd")
```



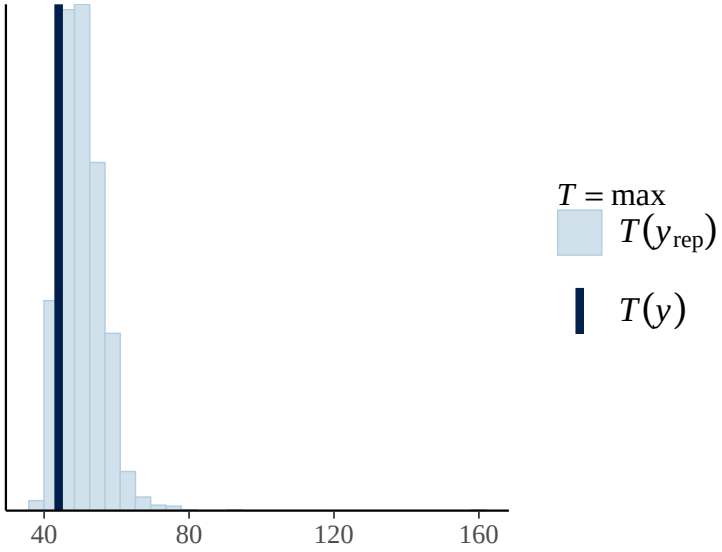

```
ppc_stat(y2_fitness, yrep2_fitness_pois,stat="max")
```




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




```
ppc_stat(y2_fitness, yrep2_fitness_zinb,stat="max")
```

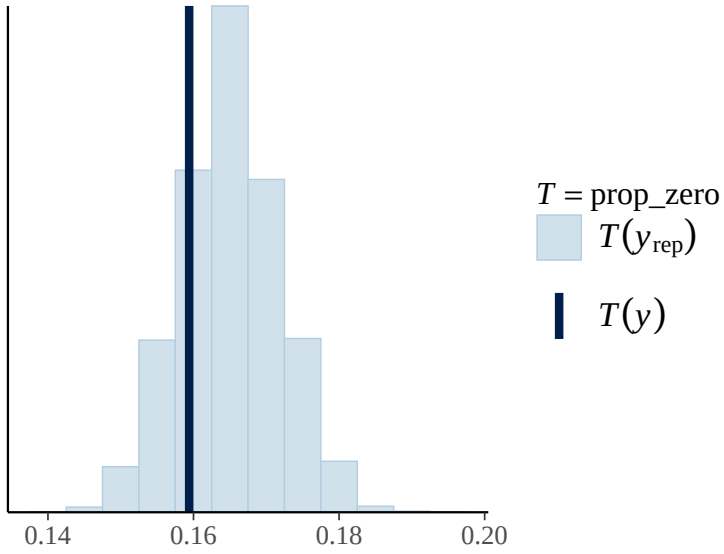


Look at the distribution of the proportion of zeros over the replicated datasets from the posterior predictive distribution in yrep and compare to the proportion of observed zeros in y.

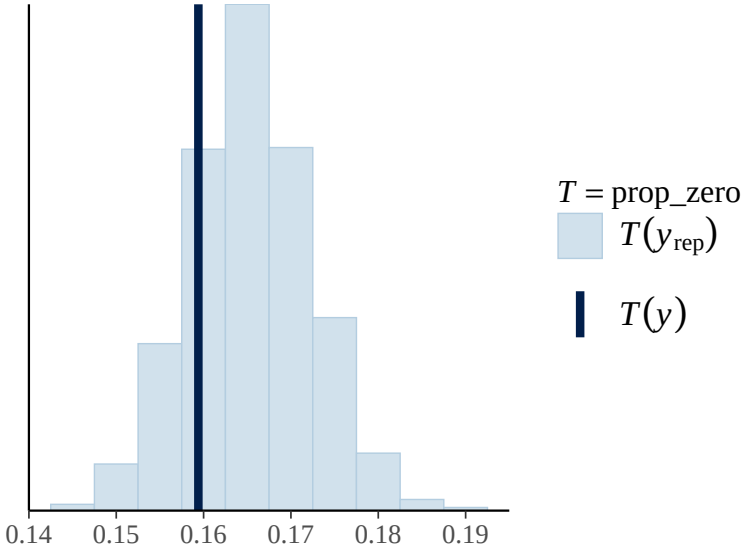
```
# Define a function that takes a vector as input
# and returns the proportion of zeros:
prop_zero <- function(x) mean(x == 0)
prop_zero(y2_fitness) # check proportion of zeros in y
```

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

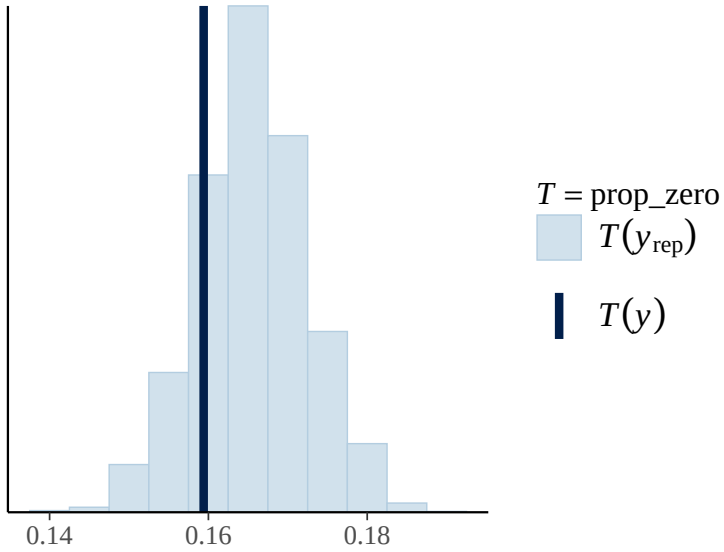
```
ppc_stat(y2_fitness, yrep2_fitness_pois, stat = "prop_zero", binwidth = 0.005)
```




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




```
ppc_stat(y2_fitness, yrep2_fitness_zinb, 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
```

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

```
##                         Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                  0.6475578 0.009160637 0.6288990 0.6652379
## R2roundmeanfitnessfl 0.9378118 0.007990778 0.9269299 0.9469240
```

Very similar.

Widely Applicable Information Criterion (WAIC):

```
waic1<-waic(bivar1.all.brm.pois,bivar1.all.brm.nb,bivar1.all.brm.zinb,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.
##
## Output of model 'bivar1.all.brm.zinb':
##
## Computed from 2000 by 2478 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic -11471.5  50.7
## p_waic       482.3  12.0
## waic       22942.9 101.4
##
## 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
## bivar1.all.brm.zinb -13.2       1.4
```

Suggests that poisson is a bit better, but not sure we can trust this (gave some warnings).

Leave-one-out cross validation (LOO):

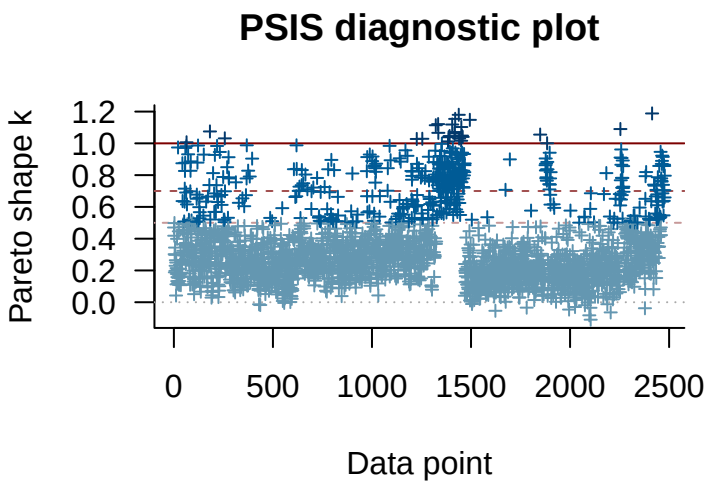
Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details. Found 263 observations with a pareto_k > 0.7 in model 'bivar1.all.brm.pois'. With this many problematic observations, it may be more appropriate to use 'kfold' with argument 'K = 10' to perform 10-fold cross-validation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

```
loo_compare(loo2_pois,loo2_nb,loo2_zinb)
```

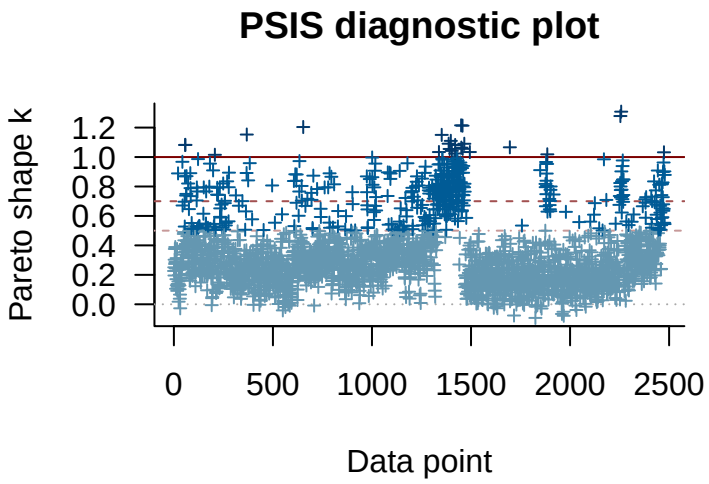
```
##                      elpd_diff se_diff
## bivar1.all.brm.pois    0.0       0.0
## bivar1.all.brm.zinb   -3.6       3.3
## bivar1.all.brm.nb    -11.7       2.9
```

Suggests that poisson is a bit better, but not sure we can trust this (gave warnings about too high pareto k diagnostic values).

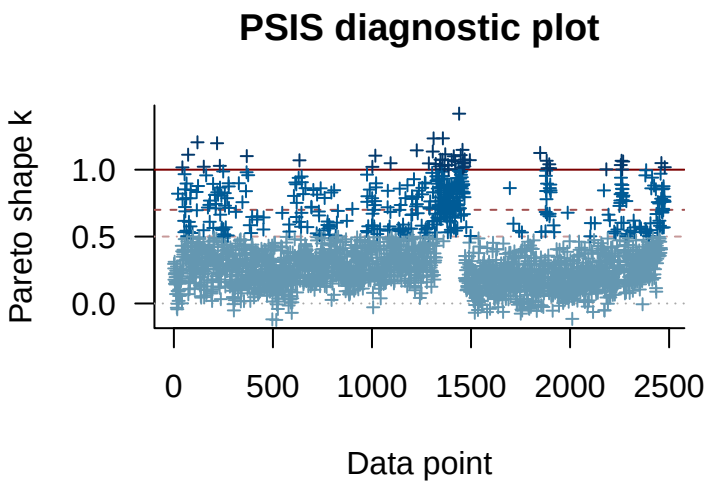
```
plot(loo2_pois)
```




```
plot(loo2_nb)
```




```
plot(loo2_zinb)
```



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")
# ppc_loo_pit_overlay(y2_fitness, yrep2_fitness_zinb,
#                     lw = weights(loo2_zinb$psis_object))
# ppc_loo_pit_qq(y2_fitness, yrep2_fitness_zinb,
#                     lw = weights(loo2_zinb$psis_object))
# ppc_loo_pit_qq(y2_fitness, yrep2_fitness_zinb,
#                     lw = weights(loo2_zinb$psis_object),compare="normal")
```

All looking quite bad!

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

```
loo_compare(kfold1_pois,kfold1_nb
            #,kfold1_zinb
            )
```

```
##                      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?.

Decide if using Poisson or negative binomial! Not yet run for zero-inflated negative binomial.

Prior predictive checks

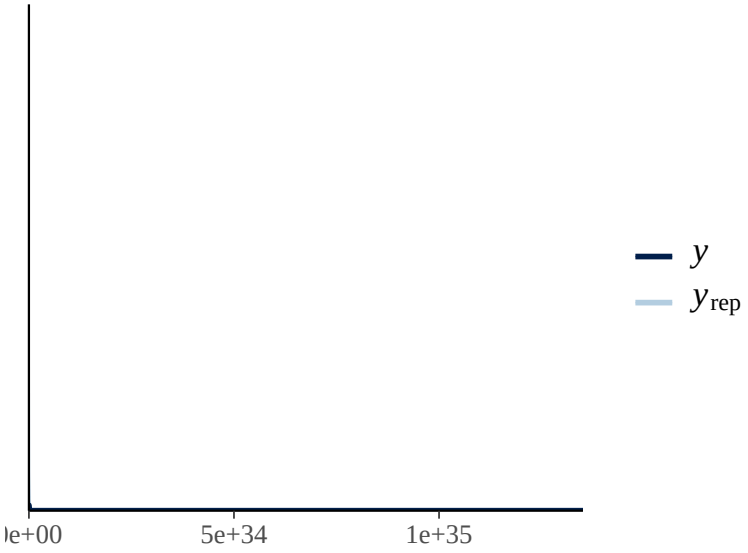
Not sure what I am doing here! I would need help from Pieter. I know you have to sample the prior by using sample_prior=TRUE, but I don't have much idea about what kind of priors we could specify to improve the model, and if this is really needed or if according to the posterior predictive checks we can stay with the default priors.

Any help welcome!

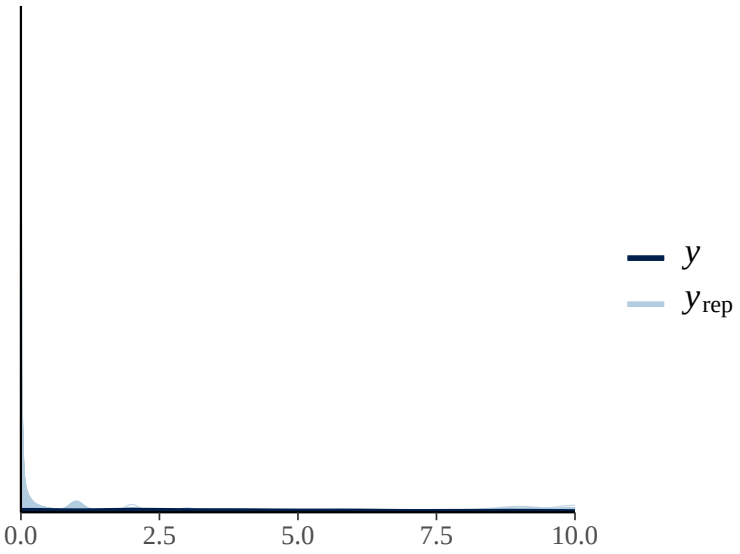
```
yrep2_fitness_pois_priors<-posterior_predict(bivar1.all.brm.pois_priors,
                                  draws = 500,resp="roundmeanfitnessfl")
yrep2_FFD_pois_priors<-posterior_predict(bivar1.all.brm.pois_priors,
                                  draws = 500,resp="FFD")
# matrices of draws from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

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

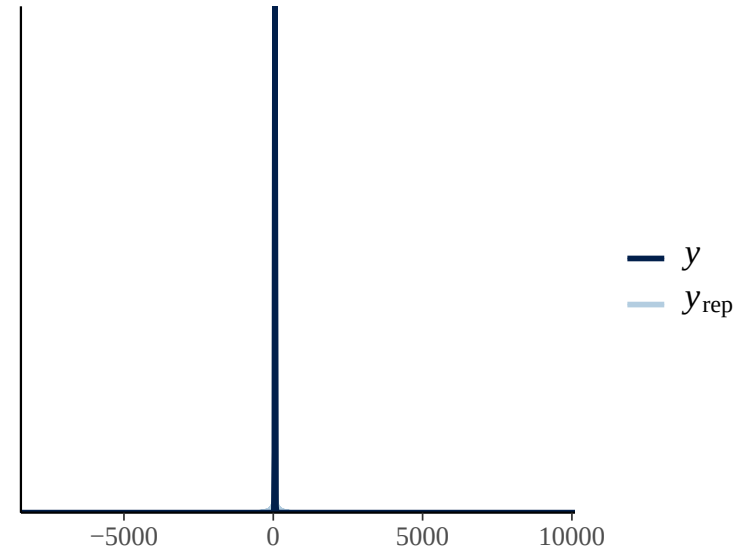
```
ppc_dens_overlay(y2_fitness, yrep2_fitness_pois_priors[1:500,])
```




```
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 100% sure about what I am doing here! This 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 = 6000) {

  ##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 = 6000)
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, 1:6000] 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,]  2.5285029  0.85016570 -1.16950607
## [2,]  0.8501657  0.68394731 -0.03729014
## [3,] -1.1695061 -0.03729014  1.90386052
```

```
# 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)]
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] -1.16950607 -0.03729014
```

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

```
## S_intercepts     S_slopes
##  -1.16950607  -0.03729014
```

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

```
##                   lower      upper
## S_intercepts -1.5567128 -0.8355278
## S_slopes     -0.3063104  0.2563138
## 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_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
##      -0.6894984       0.8298690
```

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

```
##                       lower     upper
## beta_intercepts -1.6915177 -0.427016
## beta_slopes      0.1597875  3.216674
## 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 = 6000)
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, 1:6000] 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,]  2.619912  0.84477099 -1.14772667
## [2,]  0.844771  0.68618021 -0.05491133
## [3,] -1.147727 -0.05491133  1.89459333
```

```r
# 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] -1.14772667 -0.05491133
```

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

```
## S_intercepts     S_slopes
##  -1.14772667  -0.05491133
```

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

```
##                   lower      upper
## S_intercepts -1.5452164 -0.7956288
## S_slopes     -0.3384208  0.2340203
## 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_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
##      -0.6667632       0.7369160
```

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

```
##                       lower      upper
## beta_intercepts -1.66780198 -0.3710434
## beta_slopes      0.05393317  3.2204163
## attr(,"Probability")
## [1] 0.95
```

```
# Extract posterior samples
bivar1.all.brm.zinb_post <- posterior_samples(bivar1.all.brm.zinb)
bivar1.all.brm.zinb_post <- as.mcmc(bivar1.all.brm.zinb_post)
#head(bivar1.all.brm.zinb_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.gmat9 <- sample.gmat1(bivar1.all.brm.zinb_post, replicates = 2000)
sampled.gmat9[[2]]
```

Negative binomial model with zero-inflation

```
## $gmat
##           [,1]       [,2]       [,3]
## [1,]  2.108954  0.9052240 -1.3950987
## [2,]  0.905224  0.5419099 -0.2723274
## [3,] -1.395099 -0.2723274  1.9821025
```

```
sgmat9 <- lapply(sampled.gmat9, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat9 <- unname(sapply(sgmat9, '[[', 1)) #Change to matrix
str(sgmat9)
```

```
##  num [1:9, 1:2000] 2.631 0.923 -1.397 0.923 0.527 ...
```

```
sgmat9 <- t(sgmat9)

P.modelBV_RR9 <- sgmat9
P.modelBV_RR9.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR9.mode[k] <- posterior.mode(mcmc(sgmat9[,k]))
P.modelBV_RR9.mode
```

```
##            [,1]        [,2]        [,3]
## [1,]  2.5732438  0.83935807 -1.19283720
## [2,]  0.8393581  0.56756626 -0.05493354
## [3,] -1.1928372 -0.05493354  1.94128867
```

```
# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR9 <- sgmat9[,c(3,6)]
colnames(S.modelBV_RR9) <- c("S_intercepts", "S_slopes")
S.modelBV_RR9.mode <- P.modelBV_RR9.mode[1:2, 3]
S.modelBV_RR9.mode
```

```
## [1] -1.19283720 -0.05493354
```

```r
posterior.mode(mcmc(S.modelBV_RR9))
```

```
## S_intercepts     S_slopes
##  -1.19283720  -0.05493354
```

```r
HPDinterval(mcmc(S.modelBV_RR9))
```

```
##                   lower      upper
## S_intercepts -1.5507723 -0.8527207
## S_slopes     -0.2939937  0.2722513
## 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_RR9 <- matrix(NA, nrow(S.modelBV_RR9) ,2)

for (i in 1:nrow(S.modelBV_RR9)) {
  P3_9 <- 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_9[k] <- P.modelBV_RR9[i, k] }
  P2_9 <- P3_9[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S9 <- P3_9[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR9[i,] <- solve(P2_9) %*% S9   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR9) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR9))
```

```
## beta_intercepts     beta_slopes
##      -0.7198987       0.8056513
```

```r
HPDinterval(mcmc(beta_post_RR9))
```

```
##                     lower      upper
## beta_intercepts -1.719390 -0.3498433
## beta_slopes      0.260573  3.6789743
## attr(,"Probability")
## [1] 0.95
```

2. mean_fitness_fl, with shoot volume

```r
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 = 1000,iter = 6000,chains=4,thin=2,
                         inits = "random",seed = 12345,cores = my.cores,
                         control = list(adapt_delta = 0.99, max_treedepth = 17))
```

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

Got warnings:

Due to these warnings, results of bivar2.all.brm.pois are probably not reliable! I have been trying to increase max_treedepth and running more iterations, but this takes extremely long to run (a couple of days with these parameters). I am not sure it makes sense to try increasing max_treedepth and iterations even more, or if we should just stay with the negative binomial distribution, which works fine and gives no warnings.

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))
```

```
print(bivar2.all.brm.nb,digits=3)
```

```
##  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)                                 1.665     0.148    1.377
## sd(FFD_cmean_4)                                   0.804     0.133    0.546
## sd(roundmeanfitnessfl_Intercept)                  1.240     0.049    1.150
## cor(FFD_Intercept,FFD_cmean_4)                    0.641     0.135    0.352
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)  -0.448     0.077   -0.594
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)     0.162     0.134   -0.103
##                                                u-95% CI  Rhat Bulk_ESS
## sd(FFD_Intercept)                                 1.964 1.001     2888
## sd(FFD_cmean_4)                                   1.068 1.004     1189
## sd(roundmeanfitnessfl_Intercept)                  1.341 1.000     2969
## cor(FFD_Intercept,FFD_cmean_4)                    0.873 1.005      580
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)  -0.290 1.001     1358
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)     0.427 1.004     1115
```

```
##                                                 Tail_ESS
## sd(FFD_Intercept)                                   4365
## sd(FFD_cmean_4)                                     4337
## sd(roundmeanfitnessfl_Intercept)                    4573
## cor(FFD_Intercept,FFD_cmean_4)                      3008
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)     2543
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)       3771
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)    5.219     0.882    3.821    7.152 1.000     2461     3998
##
## Population-Level Effects:
##                                         Estimate Est.Error l-95% CI u-95% CI
## FFD_Intercept                             58.644     1.124   56.467   60.869
## roundmeanfitnessfl_Intercept               0.791     0.053    0.686    0.898
## FFD_cmean_4                               -2.396     0.831   -4.046   -0.794
## roundmeanfitnessfl_cn_shoot_vol_mean_sqrt   0.040     0.004    0.032    0.047
##                                         Rhat Bulk_ESS Tail_ESS
## FFD_Intercept                          1.001     1670     3037
## roundmeanfitnessfl_Intercept           1.001     4645     5474
## FFD_cmean_4                            1.000     2309     3430
## roundmeanfitnessfl_cn_shoot_vol_mean_sqrt 1.000     4893     5127
##
## Family Specific Parameters:
##                          Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS
## sigma_FFD                   4.351     0.075    4.204    4.499 1.002     2031
## shape_roundmeanfitnessfl  673.569   185.196  381.507 1104.426 1.001     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).
```

Negative binomial distribution with zero-inflation

```
bivar2.all.brm.zinb<-brm(bf_FFD+bf_fitness_shoot,
                      family = c(gaussian,zero_inflated_negbinomial),
                      data = datadef,warmup = 1000,iter = 2000,chains=4,thin=2,
                      inits = "random",seed = 12345,cores = my.cores,
                      control = list(adapt_delta = 0.99))
```

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

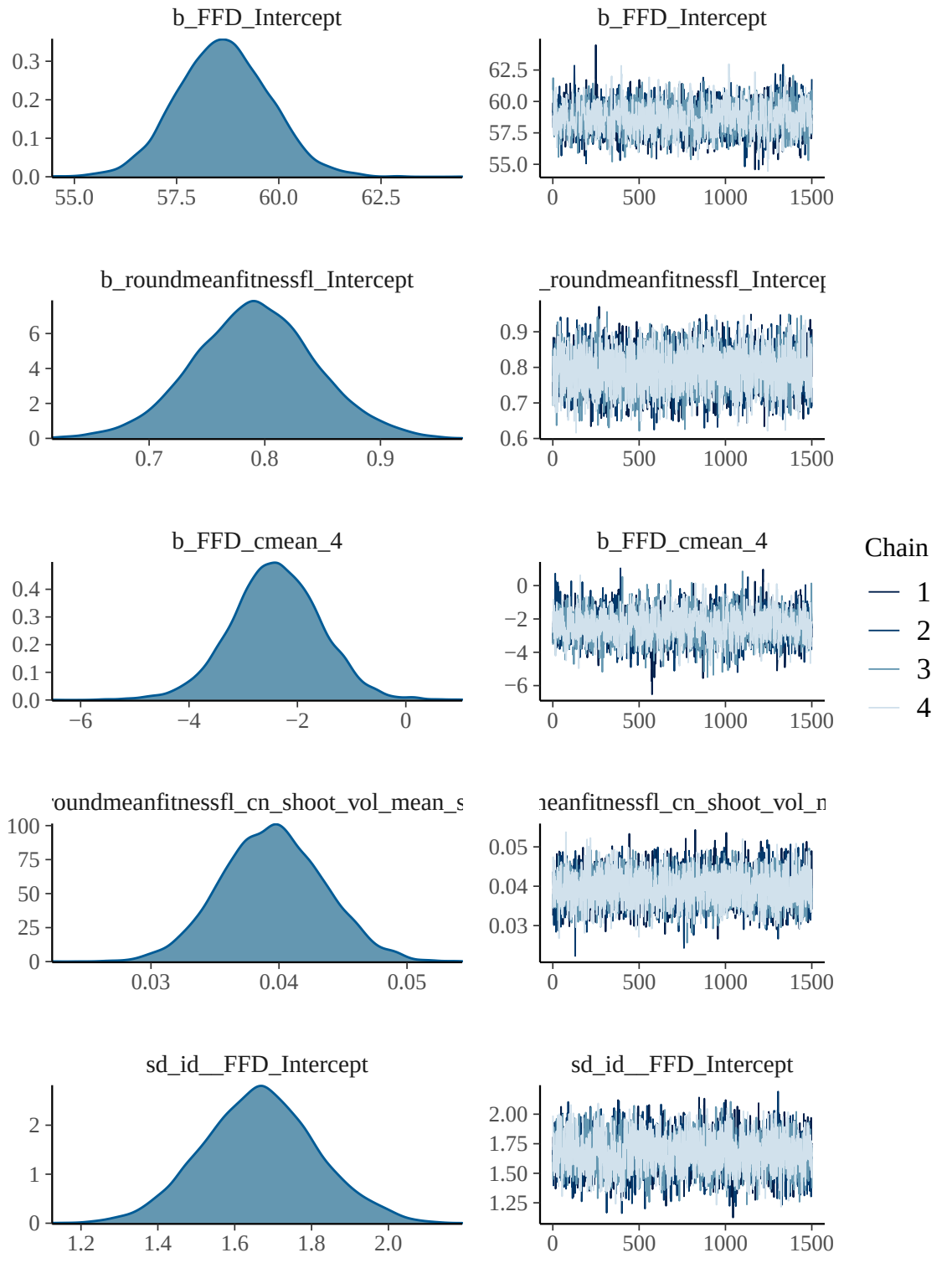
```
##  Family: MV(gaussian, zero_inflated_negbinomial)
##   Links: mu = identity; sigma = identity
##          mu = log; shape = identity; zi = identity
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_fl) ~ cn_shoot_vol_mean_sqrt + (1 | ID1 | id)
```

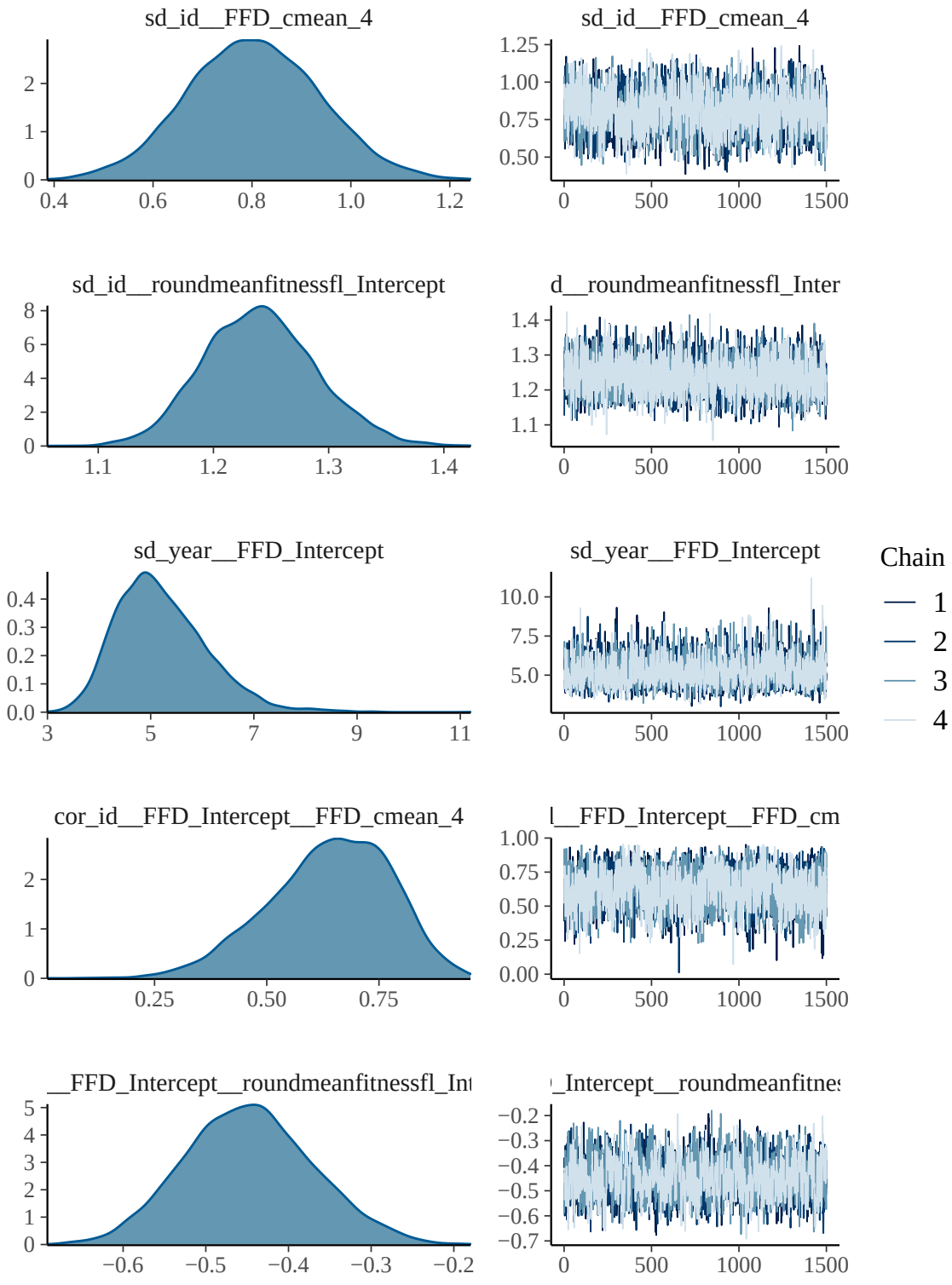
```
##    Data: datadef (Number of observations: 2432)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 2;
##          total post-warmup samples = 2000
##
## Group-Level Effects:
## ~id (Number of levels: 791)
##                                                Estimate Est.Error l-95% CI
## sd(FFD_Intercept)                                  1.67      0.15     1.37
## sd(FFD_cmean_4)                                    0.81      0.13     0.56
## sd(roundmeanfitnessfl_Intercept)                   1.24      0.05     1.14
## cor(FFD_Intercept,FFD_cmean_4)                     0.64      0.13     0.35
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)   -0.44      0.07    -0.58
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)      0.15      0.13    -0.11
##                                                u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)                                  1.97 1.00     1207     1693
## sd(FFD_cmean_4)                                    1.09 1.00      765     1594
## sd(roundmeanfitnessfl_Intercept)                   1.34 1.00     1333     1668
## cor(FFD_Intercept,FFD_cmean_4)                     0.87 1.01      306      822
## cor(FFD_Intercept,roundmeanfitnessfl_Intercept)   -0.30 1.01      542     1101
## cor(FFD_cmean_4,roundmeanfitnessfl_Intercept)      0.41 1.00      591     1267
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.20      0.86     3.78     7.16 1.00     1054     1362
##
## Population-Level Effects:
##                                         Estimate Est.Error l-95% CI u-95% CI
## FFD_Intercept                              58.53      1.12    56.23    60.74
## roundmeanfitnessfl_Intercept                0.79      0.05     0.69     0.90
## FFD_cmean_4                                -2.37      0.86    -4.19    -0.70
## roundmeanfitnessfl_cn_shoot_vol_mean_sqrt    0.04      0.00     0.03     0.05
##                                         Rhat Bulk_ESS Tail_ESS
## FFD_Intercept                           1.00      829     1273
## roundmeanfitnessfl_Intercept            1.00     1741     1615
## FFD_cmean_4                             1.00      925     1326
## roundmeanfitnessfl_cn_shoot_vol_mean_sqrt 1.00     1814     1858
##
## Family Specific Parameters:
##                          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sigma_FFD                    4.34      0.07     4.21     4.49 1.00     1237
## shape_roundmeanfitnessfl   661.16    186.43   362.87  1098.48 1.00     1444
## zi_roundmeanfitnessfl        0.00      0.00     0.00     0.00 1.00     1649
##                          Tail_ESS
## sigma_FFD                    1630
## shape_roundmeanfitnessfl     1430
## zi_roundmeanfitnessfl        1542
##
## 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).
```

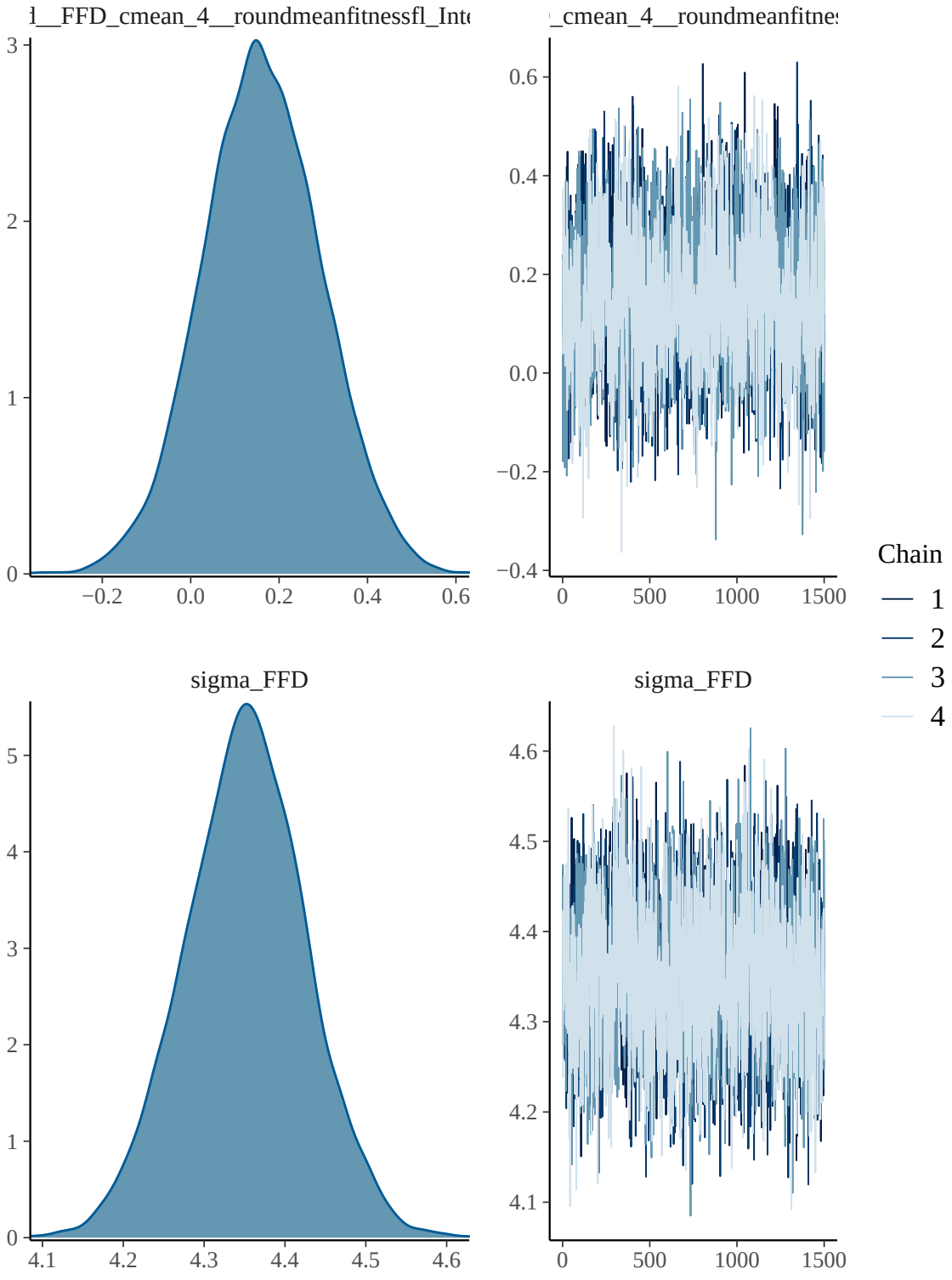
Model evaluation: Compare models

Looking only at negative binomial models by now, as Poisson model gave warnings.

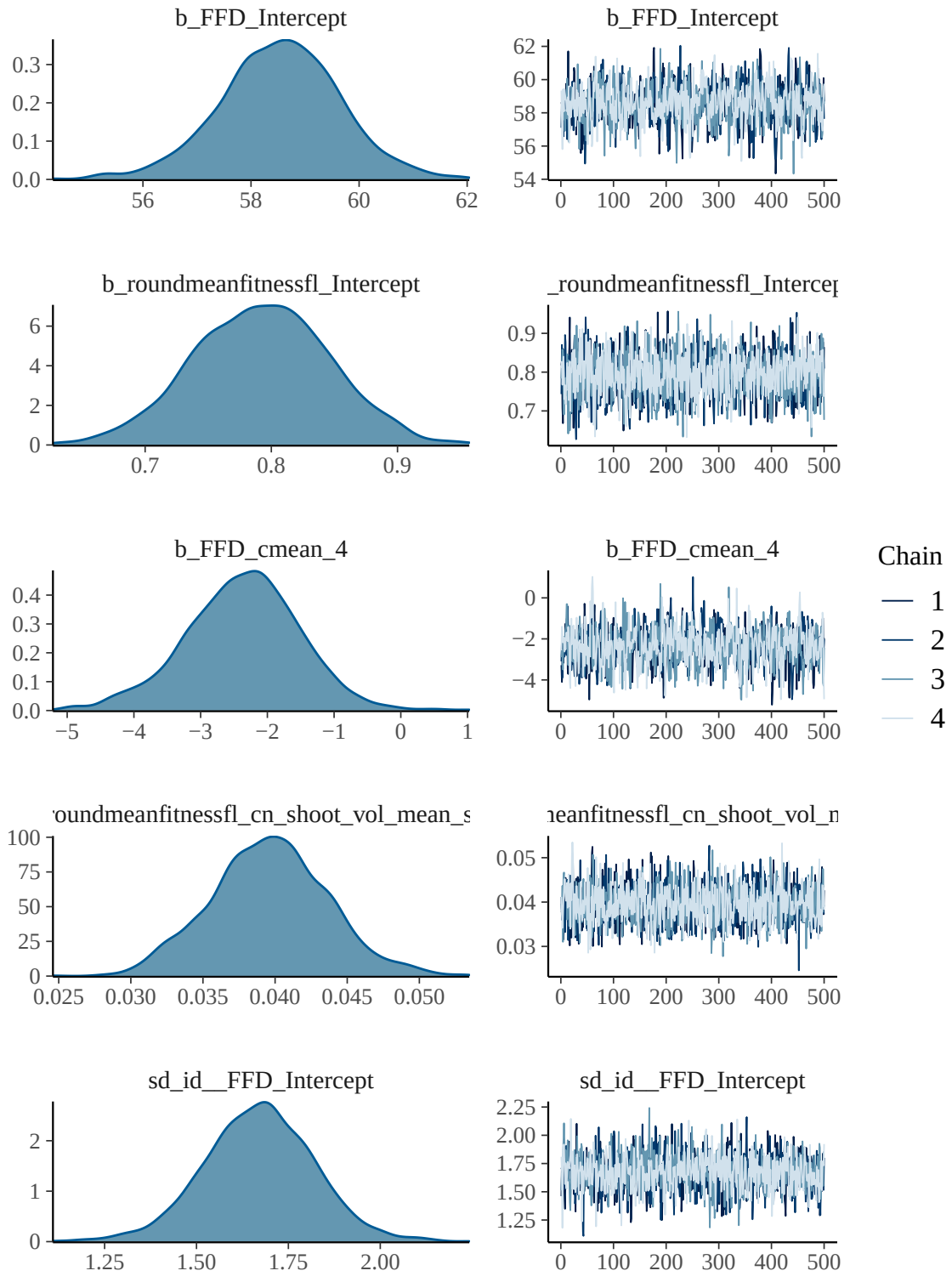
```
#plot(bivar2.all.brm.pois)
plot(bivar2.all.brm.nb)
```

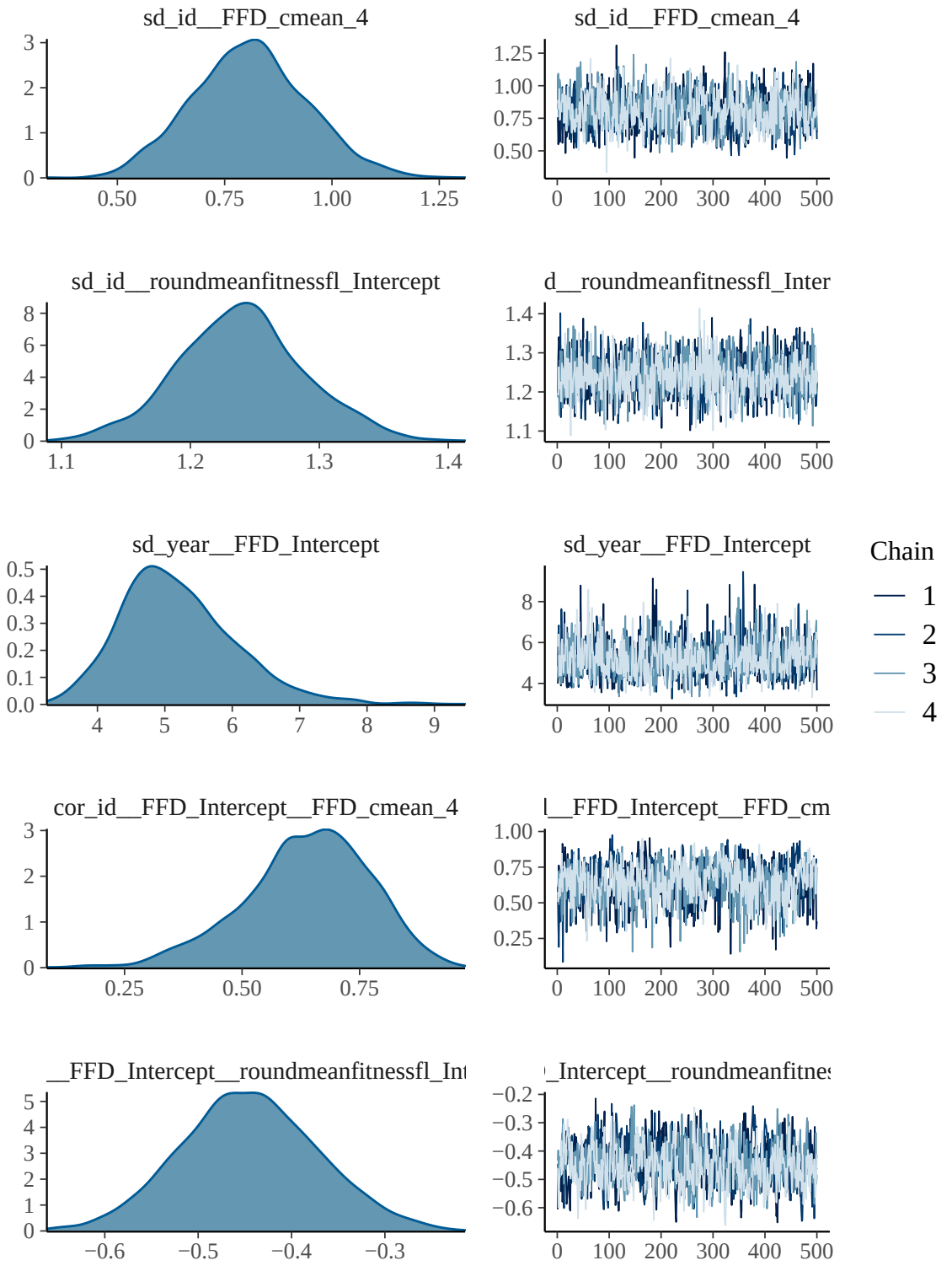

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessfl_Intercept
_roundmeanfitnessfl_Intercep
b_FFD_cmean_4
b_FFD_cmean_4
Chain
1
2
3
4
oundmeanfitnessfl_cn_shoot_vol_mean_s
eanfitnessfl_cn_shoot_vol_n
sd_id__FFD_Intercept
sd_id__FFD_Intercept

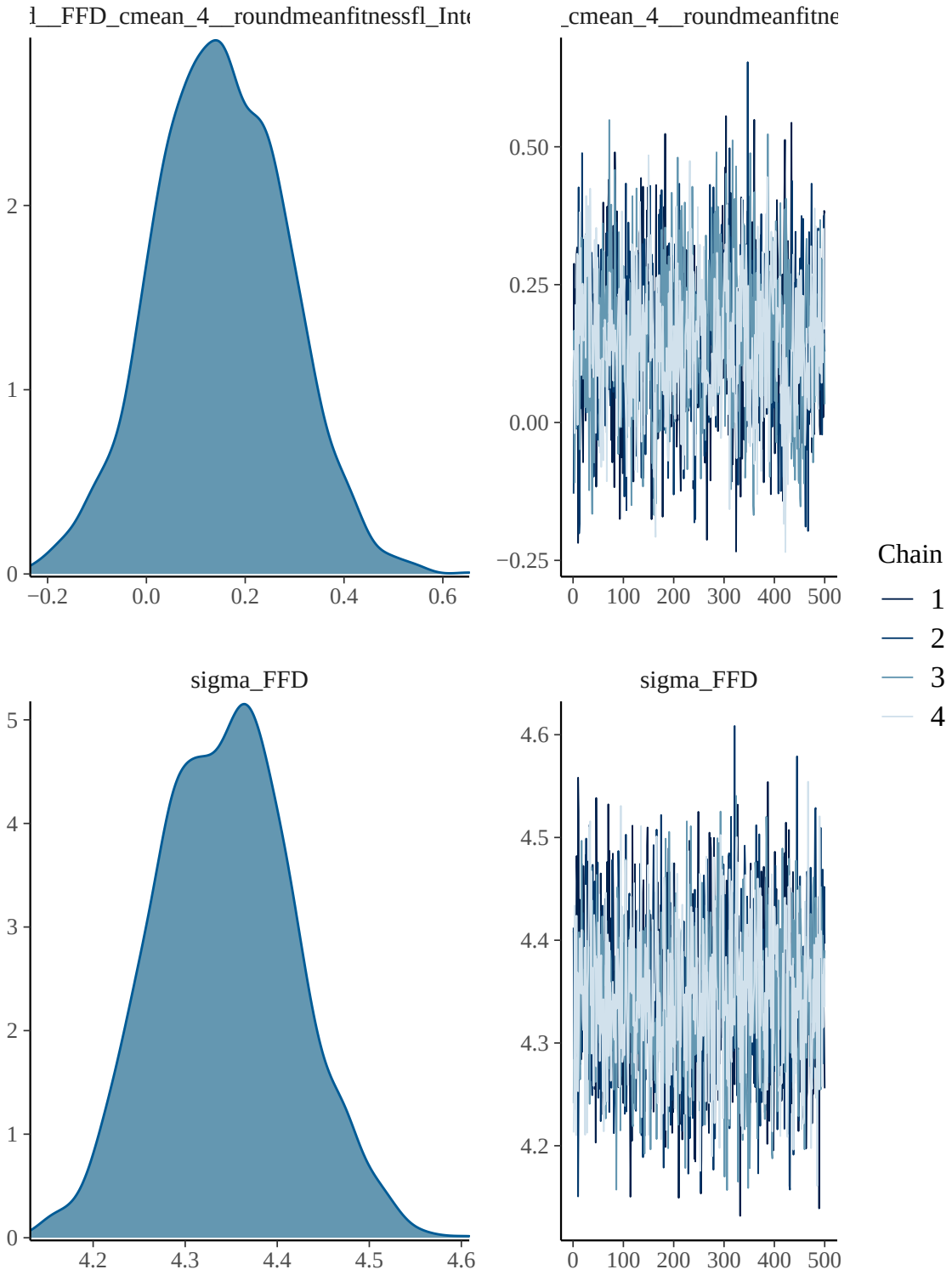

sd_id__FFD_cmean_4
sd_id__FFD_cmean_4
sd_id__roundmeanfitnessfl_Intercept
d__roundmeanfitnessfl_Inter
sd_year__FFD_Intercept
sd_year__FFD_Intercept
Chain
1
2
3
4
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
__FFD_Intercept__roundmeanfitnessfl_Int
_Intercept__roundmeanfitnes




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

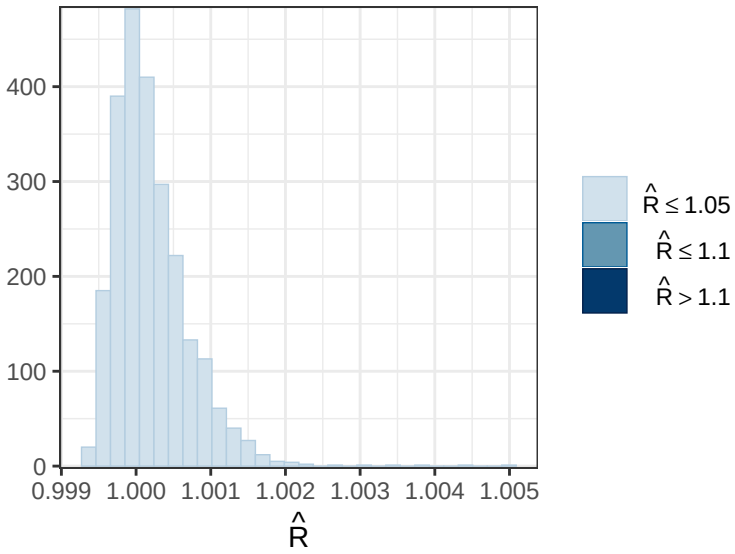

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessfl_Intercept
_roundmeanfitnessfl_Intercep
b_FFD_cmean_4
b_FFD_cmean_4
oundmeanfitnessfl_cn_shoot_vol_mean_s
eanfitnessfl_cn_shoot_vol_n
sd_id__FFD_Intercept
sd_id__FFD_Intercept
Chain
1
2
3
4


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
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
__FFD_Intercept__roundmeanfitnessfl_Int
_Intercept__roundmeanfitnes
Chain
1
2
3
4

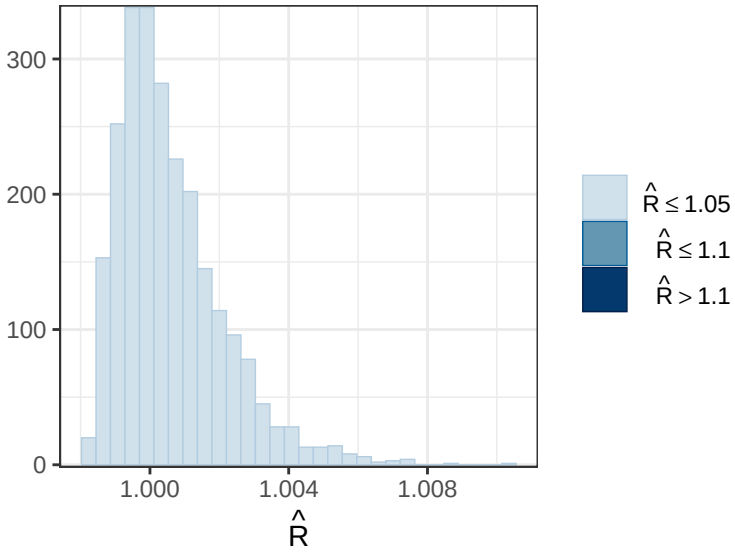


Rhat

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

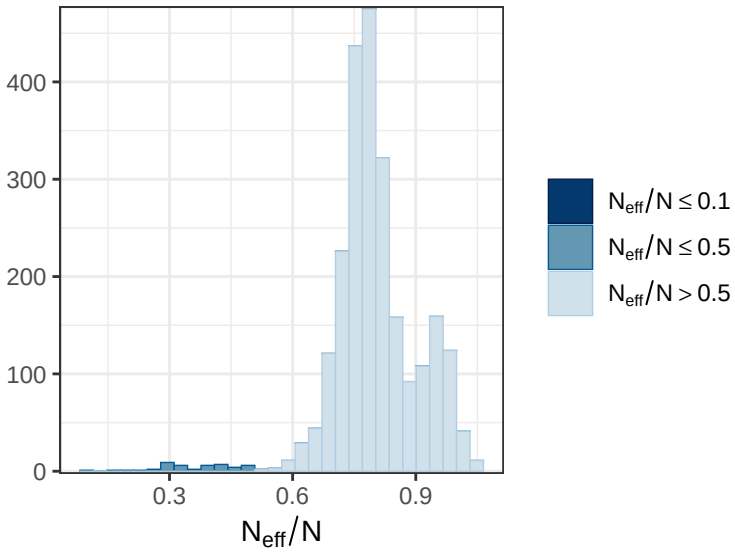



```
mcmc_rhat_hist(rhat(bivar2.all.brm.zinb))+theme_bw()
```

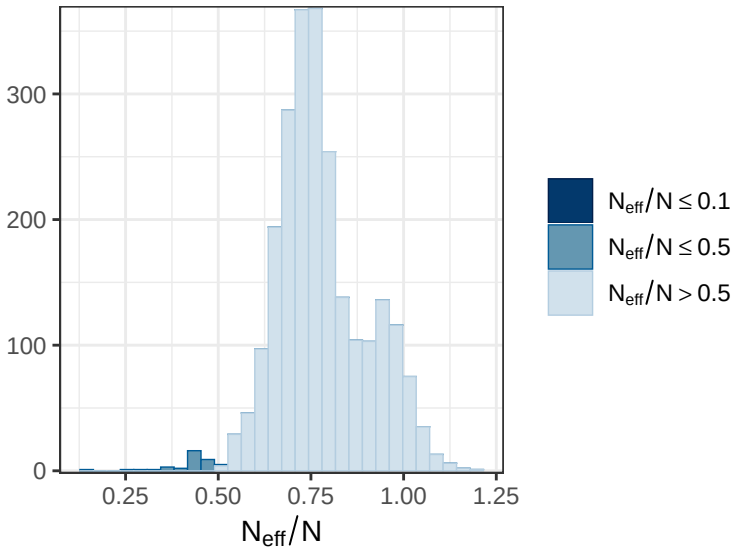


Effective sample size:

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




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



Posterior predictive checks:

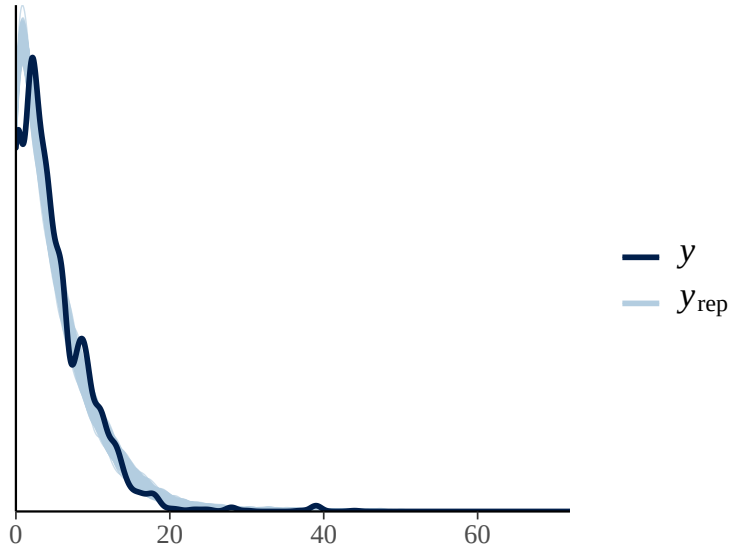
```
y3_fitness<-round(subset(datadef,!is.na(FFD)&
                           !is.na(cn_shoot_vol_mean_sqrt))$mean_fitness_fl)
y3_FFD<-subset(datadef,!is.na(FFD)&
                           !is.na(cn_shoot_vol_mean_sqrt))$FFD
# vectors of outcome values
#yrep3_fitness_pois<-posterior_predict(bivar2.all.brm.pois,
#                                      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")
yrep3_fitness_zinb<-posterior_predict(bivar2.all.brm.zinb,
                                      draws = 500,resp="roundmeanfitnessfl")
yrep3_FFD_zinb<-posterior_predict(bivar2.all.brm.zinb,
                                  draws = 500,resp="FFD")
# matrices of draws from the posterior predictive distribution
```

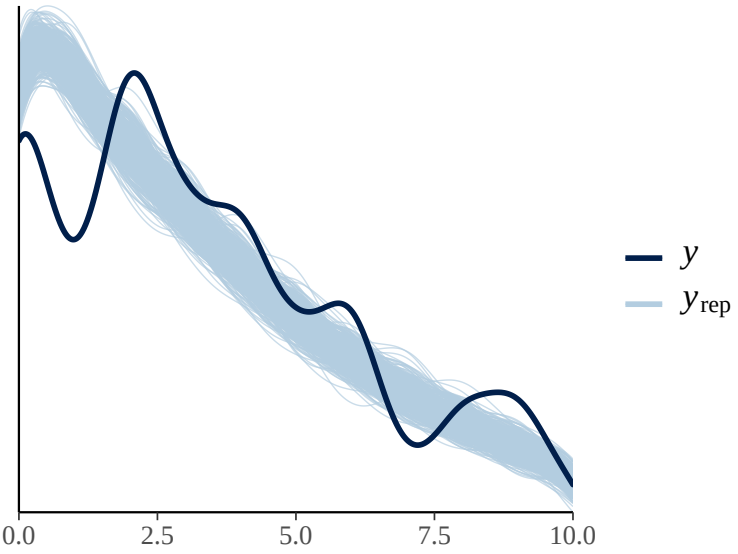
Each row of the matrix is a draw from the posterior predictive distribution, i.e. a vector with one element for each of the data points in y.

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

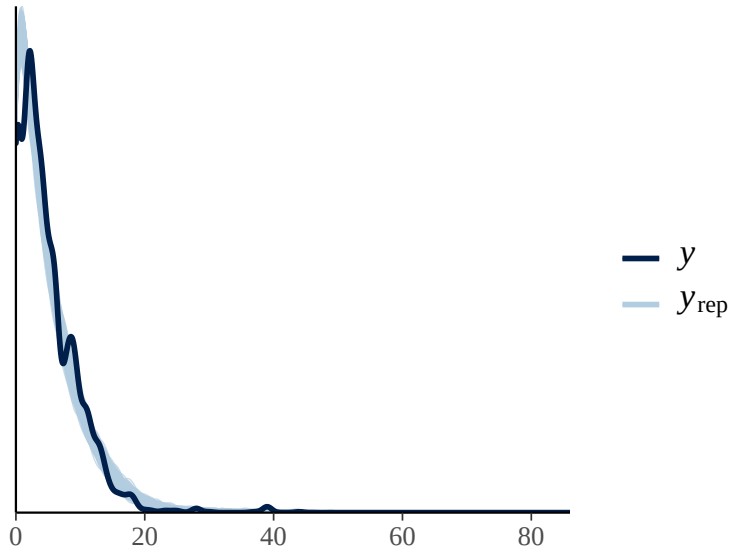
```
#ppc_dens_overlay(y3_fitness, yrep3_fitness_pois[1:500,])
#ppc_dens_overlay(y3_fitness, yrep3_fitness_pois[1:500,])+xlim(0, 10)
ppc_dens_overlay(y3_fitness, yrep3_fitness_nb[1:500,])
```



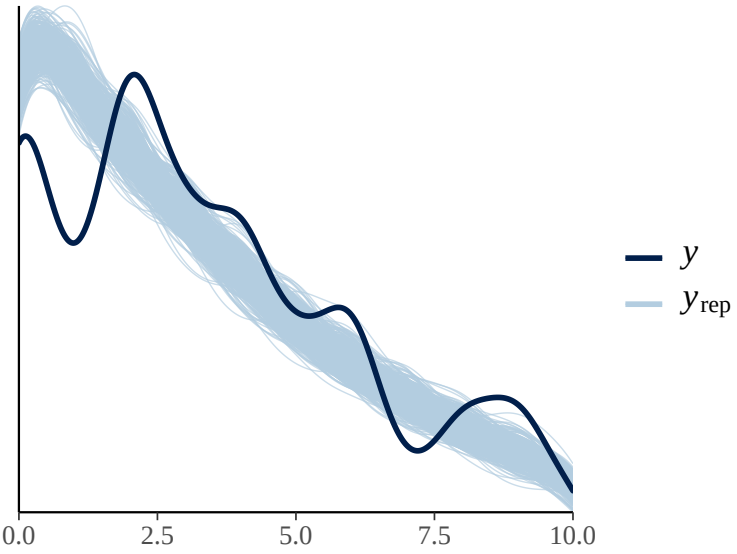

```
ppc_dens_overlay(y3_fitness, yrep3_fitness_nb[1:500,])+xlim(0,10)
```



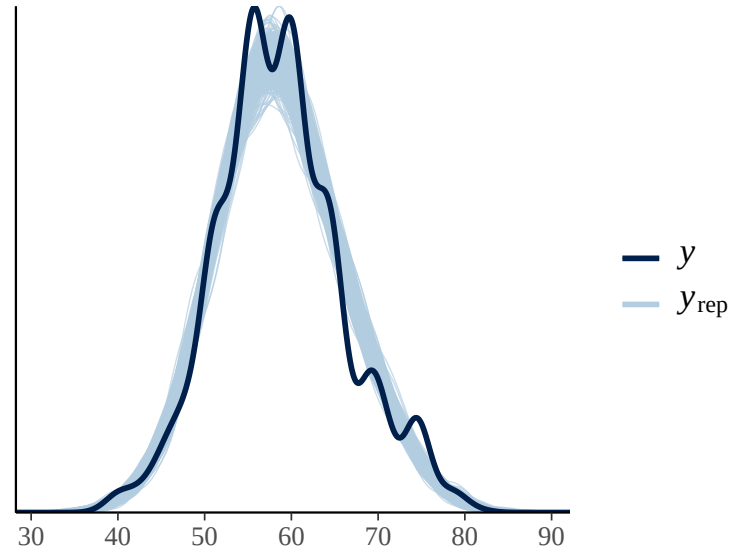

```
ppc_dens_overlay(y3_fitness, yrep3_fitness_zinb[1:500,])
```



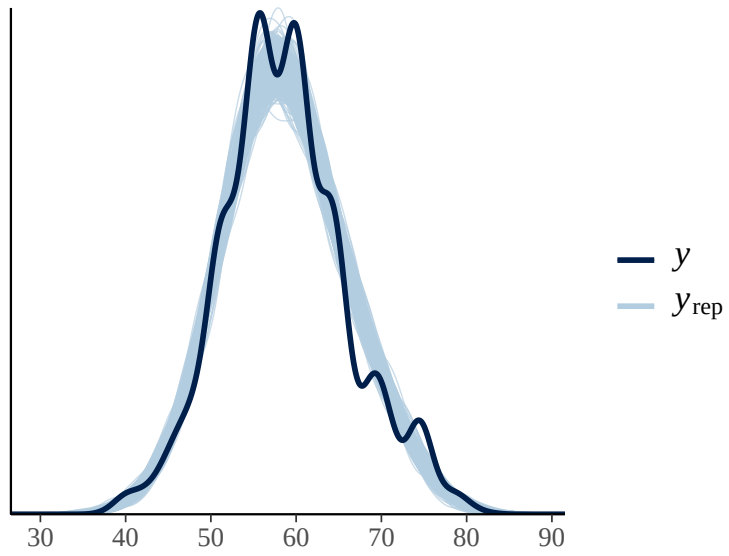

```
ppc_dens_overlay(y3_fitness, yrep3_fitness_zinb[1:500,])+xlim(0,10)
```




```
#ppc_dens_overlay(y3_FFD, yrep3_FFD_pois[1:500,])
ppc_dens_overlay(y3_FFD, yrep3_FFD_nb[1:500,])
```

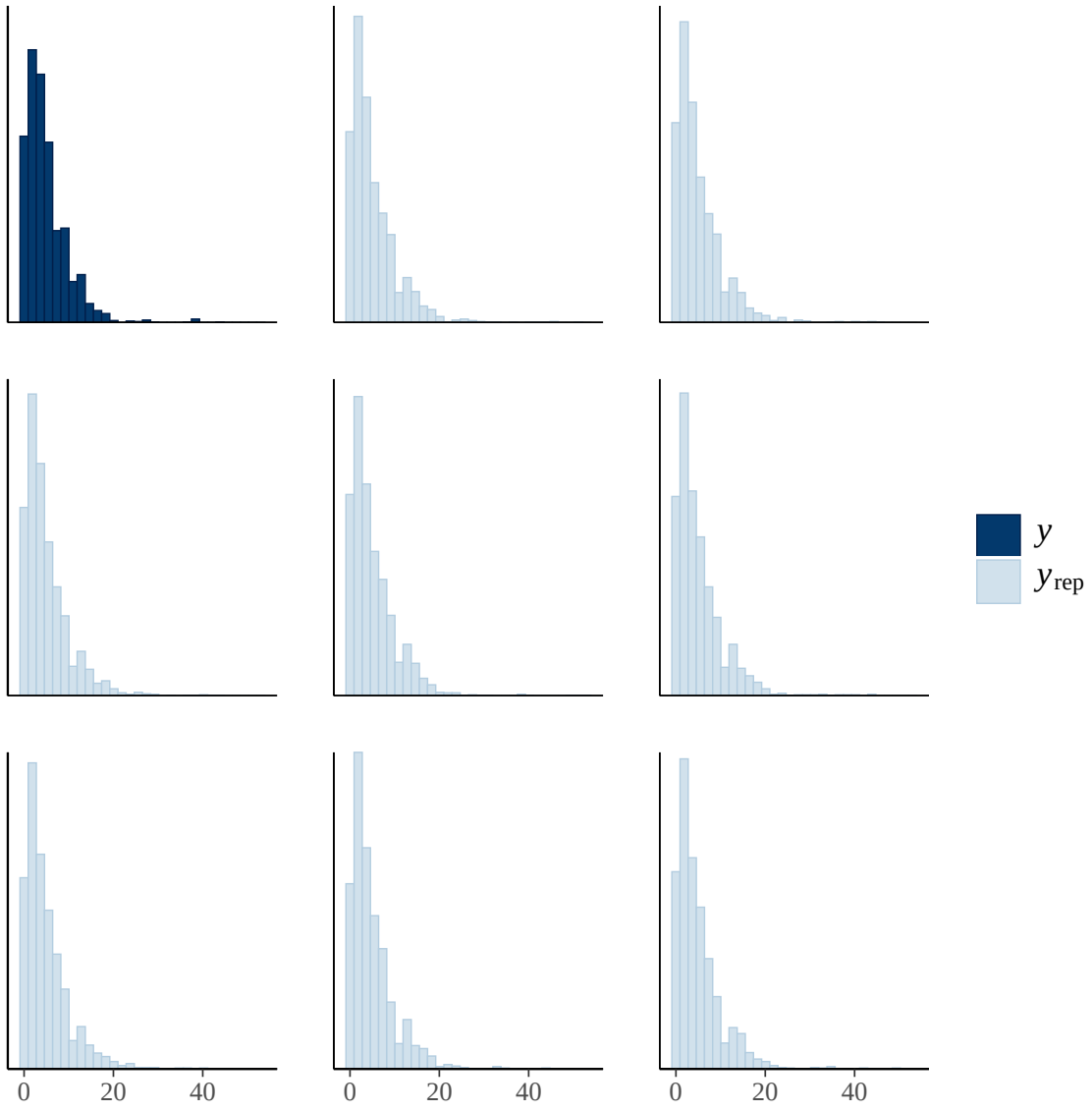



```
ppc_dens_overlay(y3_FFD, yrep3_FFD_zinb[1:500,])
```

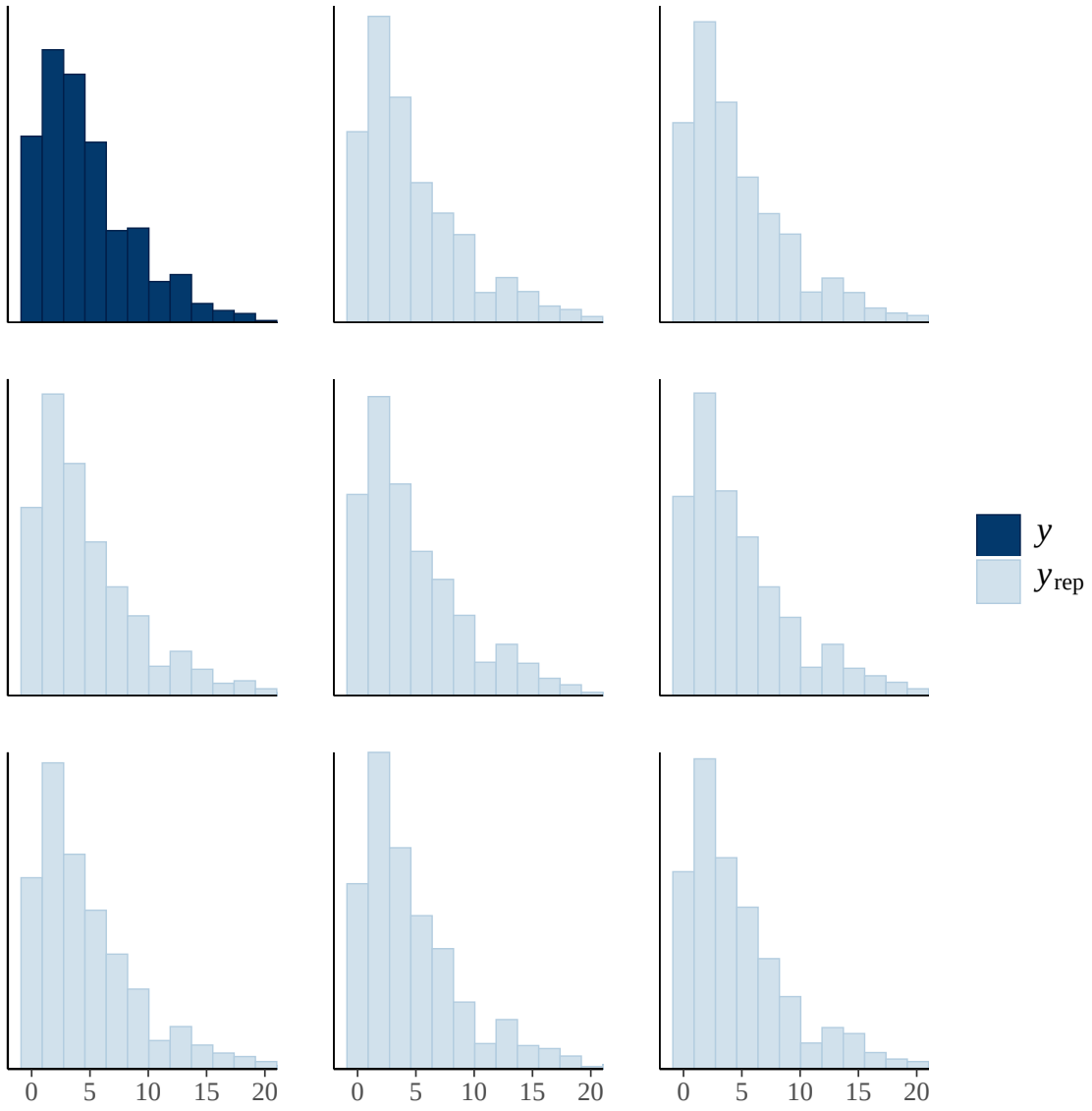


Separate histograms of y and some of the yrep datasets

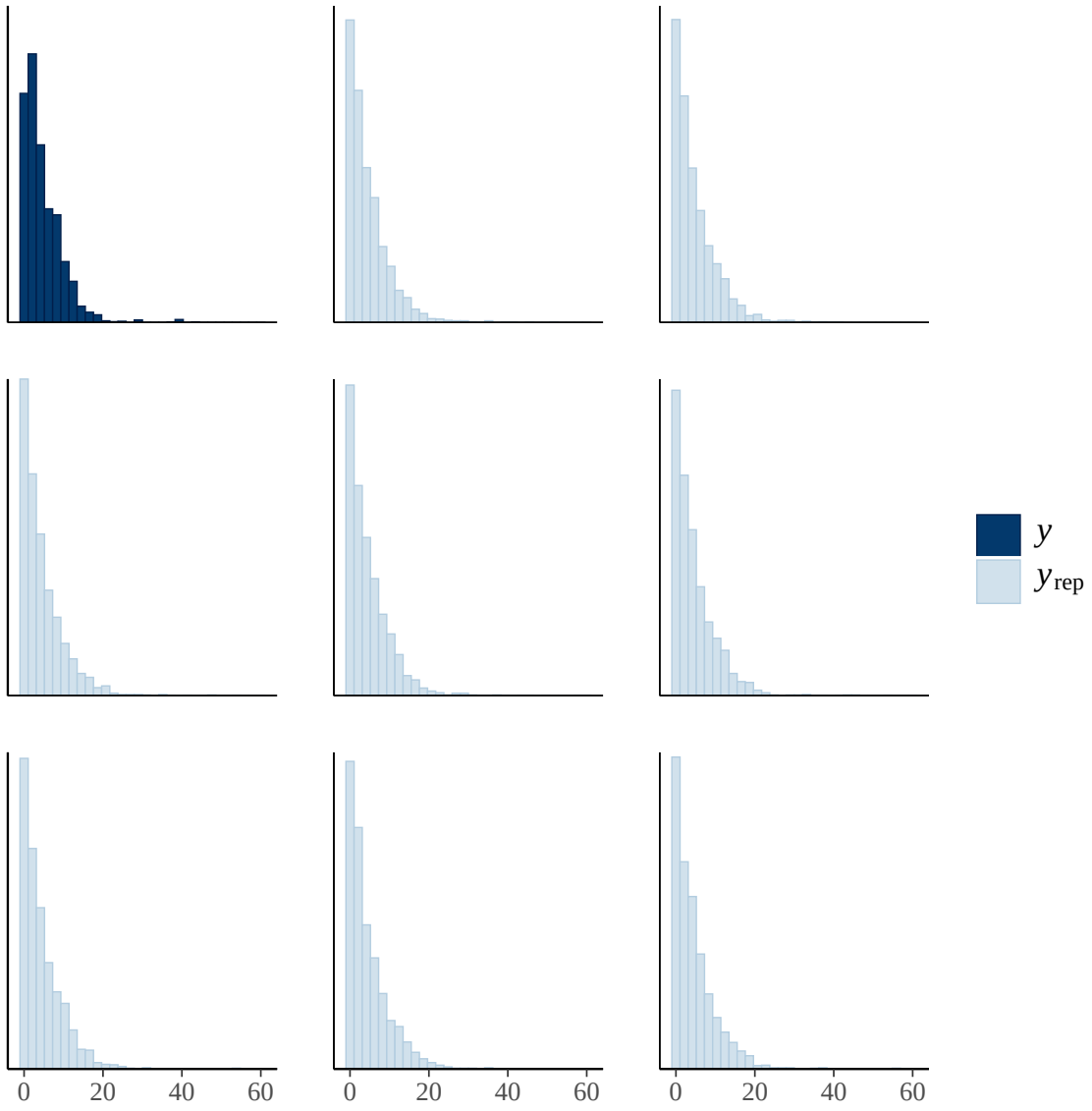
```
#ppc_hist(y3_fitness, yrep3_fitness_pois[1:8, ])
#ppc_hist(y3_fitness, yrep3_fitness_pois[1:8, ])+
#  coord_cartesian(xlim = c(-1, 20))
ppc_hist(y3_fitness, yrep3_fitness_nb[1:8, ])
```



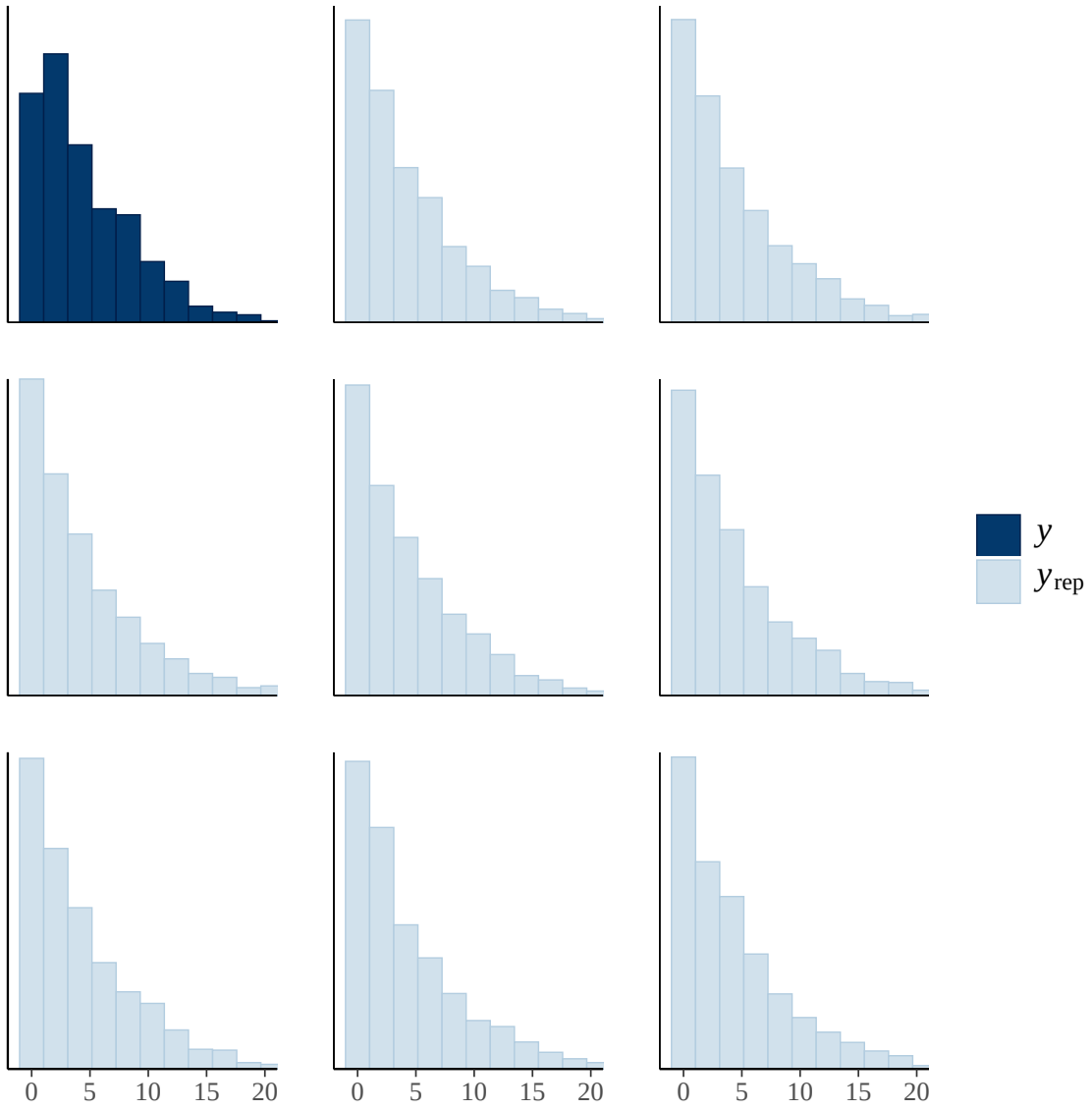

```
ppc_hist(y3_fitness, yrep3_fitness_nb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```



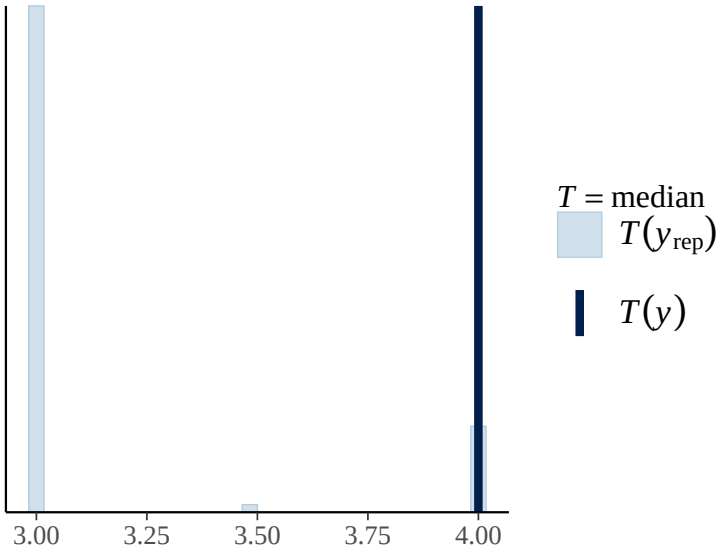

```
ppc_hist(y3_fitness, yrep3_fitness_zinb[1:8, ])
```



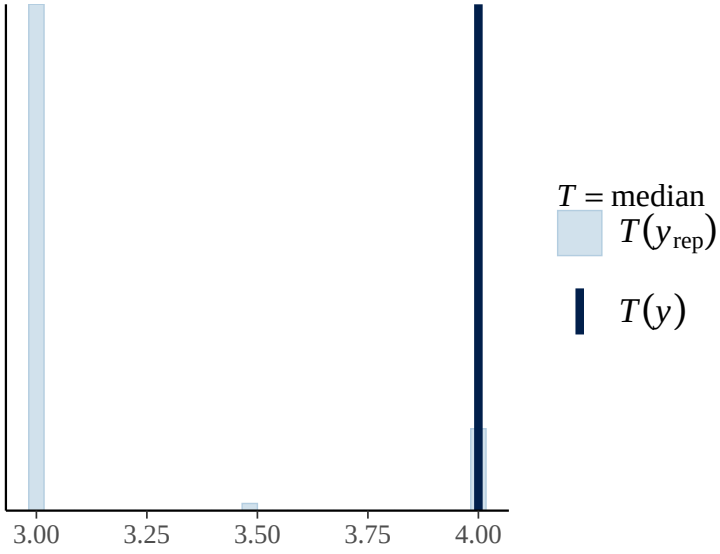

```
ppc_hist(y3_fitness, yrep3_fitness_zinb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```



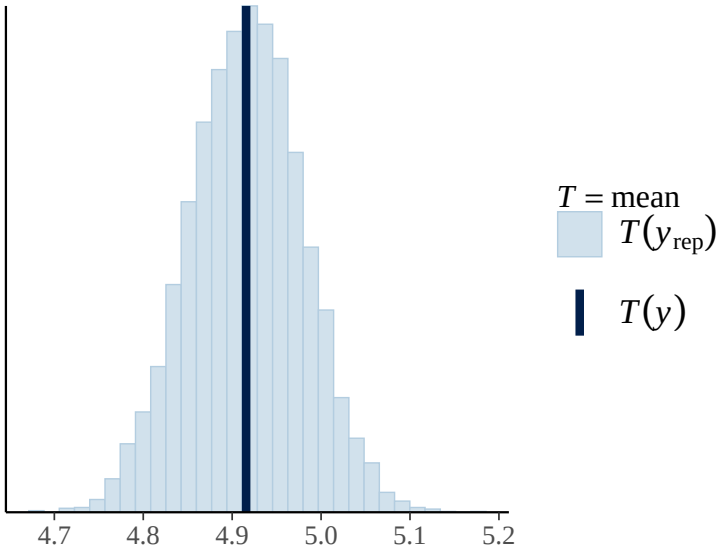

```
#ppc_stat(y3_fitness, yrep3_fitness_pois,stat="median")
ppc_stat(y3_fitness, yrep3_fitness_nb,stat="median")
```



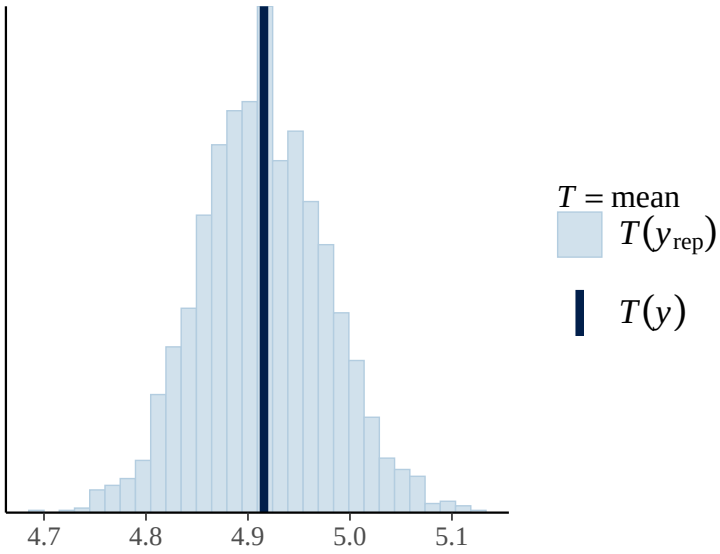

```
ppc_stat(y3_fitness, yrep3_fitness_zinb,stat="median")
```



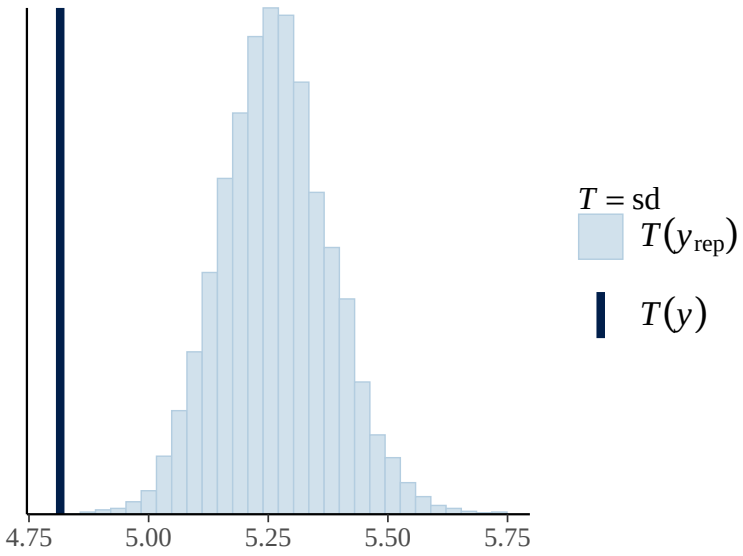

```
#ppc_stat(y3_fitness, yrep3_fitness_pois,stat="mean")
ppc_stat(y3_fitness, yrep3_fitness_nb,stat="mean")
```



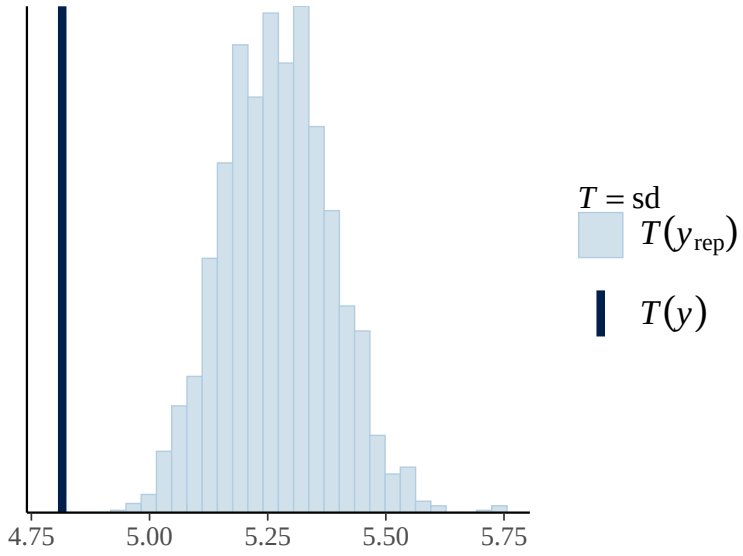

```
ppc_stat(y3_fitness, yrep3_fitness_zinb,stat="mean")
```



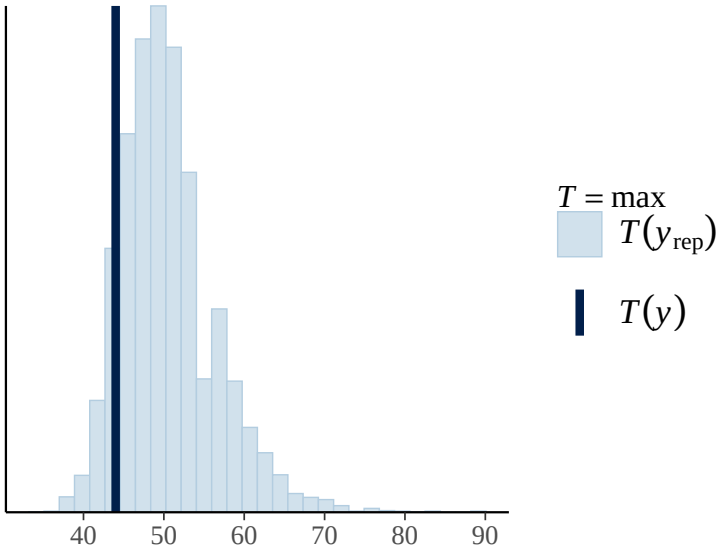

```
#ppc_stat(y3_fitness, yrep3_fitness_pois,stat="sd")
ppc_stat(y3_fitness, yrep3_fitness_nb,stat="sd")
```



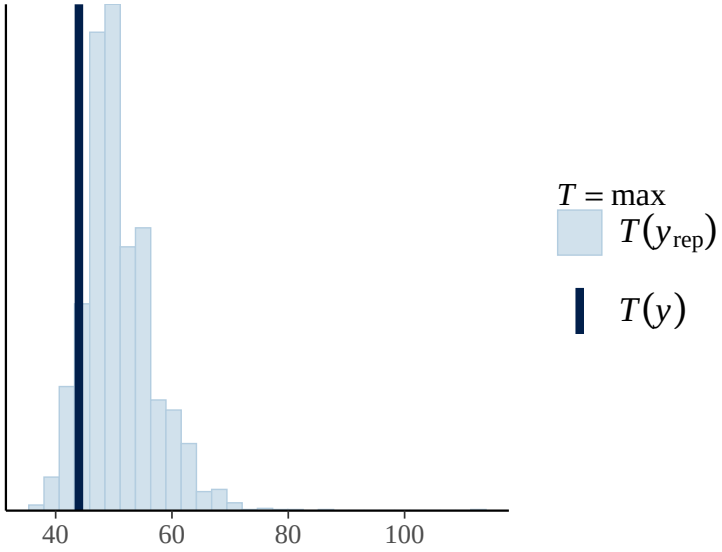

```
ppc_stat(y3_fitness, yrep3_fitness_zinb,stat="sd")
```




```
#ppc_stat(y3_fitness, yrep3_fitness_pois,stat="max")
ppc_stat(y3_fitness, yrep3_fitness_nb,stat="max")
```

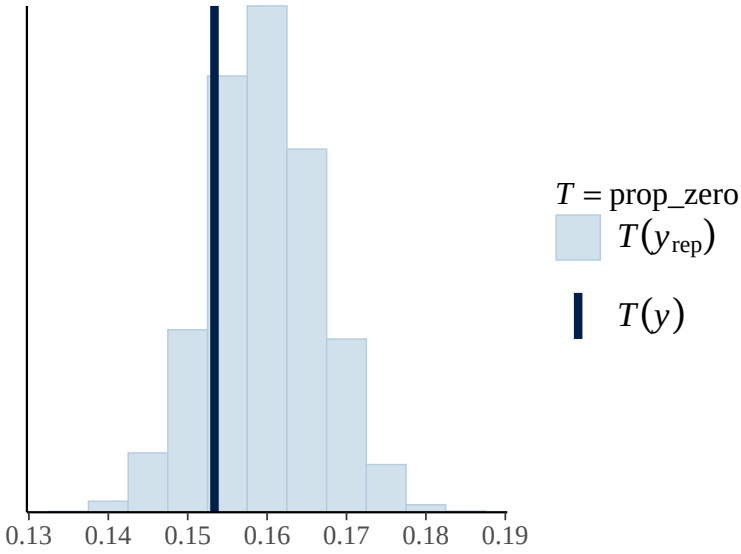



```
ppc_stat(y3_fitness, yrep3_fitness_zinb,stat="max")
```

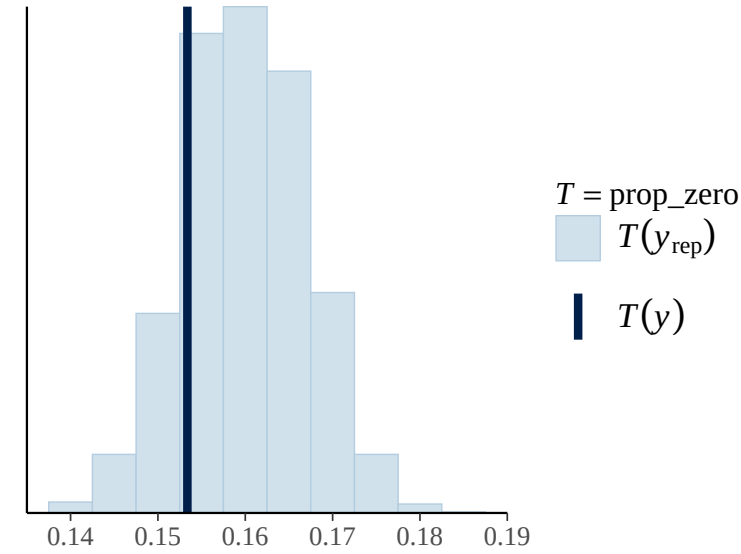


Look at the distribution of the proportion of zeros over the replicated datasets from the posterior predictive distribution in yrep and compare to the proportion of observed zeros in y.

```
#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)
```




```
ppc_stat(y3_fitness, yrep3_fitness_zinb, stat = "prop_zero", binwidth = 0.005)
```



Measure of fit: Bayesian R2

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

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

```
##                       Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                0.6493677 0.009803748 0.6293253 0.6678998
## R2roundmeanfitnessfl 0.9390054 0.005938114 0.9291607 0.9476260
```

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 = 6000) {

  ##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 = 6000)
#
# 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))
# HPDinterval(mcmc(S.modelBV_RR3))
#
# # 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))
# HPDinterval(mcmc(beta_post_RR3))
```

Poisson model (not run due to warnings)

```
# 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 = 6000)

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:
```

```r
# 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
##   -0.8660104    0.1494767
```

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

```
##                   lower      upper
## S_intercepts -1.2749421 -0.5516734
## S_slopes     -0.1049619  0.4259553
## 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_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
}

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

```
## beta_intercepts     beta_slopes
##      -0.6455921       1.0577005
```

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

```
##                      lower      upper
## beta_intercepts -1.2936484 -0.3545207
## beta_slopes      0.3610851  2.6991428
## attr(,"Probability")
## [1] 0.95
```

```
# Extract posterior samples
bivar2.all.brm.zinb_post <- posterior_samples(bivar2.all.brm.zinb)
bivar2.all.brm.zinb_post <- as.mcmc(bivar2.all.brm.zinb_post)
#head(bivar2.all.brm.zinb_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.gmat10 <- sample.gmat2(bivar2.all.brm.zinb_post, replicates = 2000)

sgmat10 <- lapply(sampled.gmat10, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat10 <- unname(sapply(sgmat10, '[[', 1)) #Change to matrix

sgmat10 <- t(sgmat10)

P.modelBV_RR10 <- sgmat10
P.modelBV_RR10.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR10.mode[k] <- posterior.mode(mcmc(sgmat10[,k]))

# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR10 <- sgmat10[,c(3,6)]
colnames(S.modelBV_RR10) <- c("S_intercepts", "S_slopes")
S.modelBV_RR10.mode <- P.modelBV_RR10.mode[1:2, 3]

posterior.mode(mcmc(S.modelBV_RR10))
```

Negative binomial model with zero-inflation

```
## S_intercepts     S_slopes
##   -0.9652113    0.1589030
```

```
HPDinterval(mcmc(S.modelBV_RR10))
```

```
##                    lower      upper
## S_intercepts -1.26818955 -0.6034589
## S_slopes     -0.09907964  0.4146302
## 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_RR10 <- matrix(NA, nrow(S.modelBV_RR10) ,2)

for (i in 1:nrow(S.modelBV_RR10)) {
  P3_10 <- 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_10[k] <- P.modelBV_RR10[i, k] }
```

```
  P2_10 <- P3_10[1:2, 1:2]    # 2x2 matrix of just trait intercept & slope var-cov
  S10 <- P3_10[1:2, 3]    # selection differentials on traits (last column of P3)
  beta_post_RR10[i,] <- solve(P2_10) %*% S10    # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR10) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR10))
```

```
## beta_intercepts     beta_slopes
##      -0.6042664       0.9263423
```

```
HPDinterval(mcmc(beta_post_RR10))
```

```
##                       lower      upper
## beta_intercepts -1.2259436 -0.3318432
## beta_slopes      0.3225026  2.4917723
## attr(,"Probability")
## [1] 0.95
```

3. mean_fitness_study, no condition variable

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

Poisson distribution

```
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))
```

```
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))
```

```
print(bivar3.all.brm.nb,digits=10)
```

```
##  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
## sd(FFD_Intercept)                                 1.6089389482 0.1440415561
## sd(FFD_cmean_4)                                   0.7843693859 0.1336041898
## sd(roundmeanfitnessstudy_Intercept)               1.4307805475 0.0561195741
## cor(FFD_Intercept,FFD_cmean_4)                    0.7416896000 0.1304752299
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept) -0.5717586818 0.0711033440
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)   -0.1765120854 0.1340853343
##                                                      l-95% CI      u-95% CI
## sd(FFD_Intercept)                                 1.3304011106  1.8902861858
## sd(FFD_cmean_4)                                   0.5219494969  1.0474817159
## sd(roundmeanfitnessstudy_Intercept)               1.3243884964  1.5459139181
## cor(FFD_Intercept,FFD_cmean_4)                    0.4497196601  0.9488515396
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept) -0.7070119194 -0.4295880869
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)   -0.4466447464  0.0834144544
##                                                         Rhat Bulk_ESS
## sd(FFD_Intercept)                                 1.0013869965     4007
## sd(FFD_cmean_4)                                   1.0014833897     3784
## sd(roundmeanfitnessstudy_Intercept)               1.0008687673     5134
## cor(FFD_Intercept,FFD_cmean_4)                    1.0044637695      696
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept) 1.0024661258     2117
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)   1.0027675934     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
## sd(FFD_Intercept) 5.1492530838 0.8727589456 3.7869281968 7.1789412866
##                            Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept) 1.0000819024     4758     7035
##
## Population-Level Effects:
##                                    Estimate    Est.Error      l-95% CI
## FFD_Intercept                   58.7842337265 1.1254444761 56.5862146532
## roundmeanfitnessstudy_Intercept  0.1290315093 0.0629281379  0.0013526587
## FFD_cmean_4                     -2.2979481127 0.8388750216 -3.9413066872
##                                      u-95% CI         Rhat Bulk_ESS Tail_ESS
## FFD_Intercept                   61.0379292995 1.0006208137     3403     5333
## roundmeanfitnessstudy_Intercept  0.2510237542 1.0000628431     7454     8292
```

```
## FFD_cmean_4                      -0.6549760291 1.0001122004     4905     6320
##
## Family Specific Parameters:
##                                   Estimate      Est.Error       l-95% CI
## sigma_FFD                     4.3576673507   0.0732053734   4.2165834764
## shape_roundmeanfitnessstudy 554.2768415839 170.4999244810 292.2832481891
##                                   u-95% CI         Rhat Bulk_ESS Tail_ESS
## sigma_FFD                     4.5026027346 1.0004406229     3842     6990
## shape_roundmeanfitnessstudy 957.2237075869 1.0002010690     7454     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).
```

Negative binomial distribution with zero-inflation

```
bivar3.all.brm.zinb<-brm(bf_FFD+bf_fitness_study,
                         family = c(gaussian,zero_inflated_negbinomial),
                         data = datadef,warmup = 1000,iter = 2000,chains=4,thin=2,
                         inits = "random",seed = 12345,cores = my.cores,
                         control = list(adapt_delta = 0.99))
```

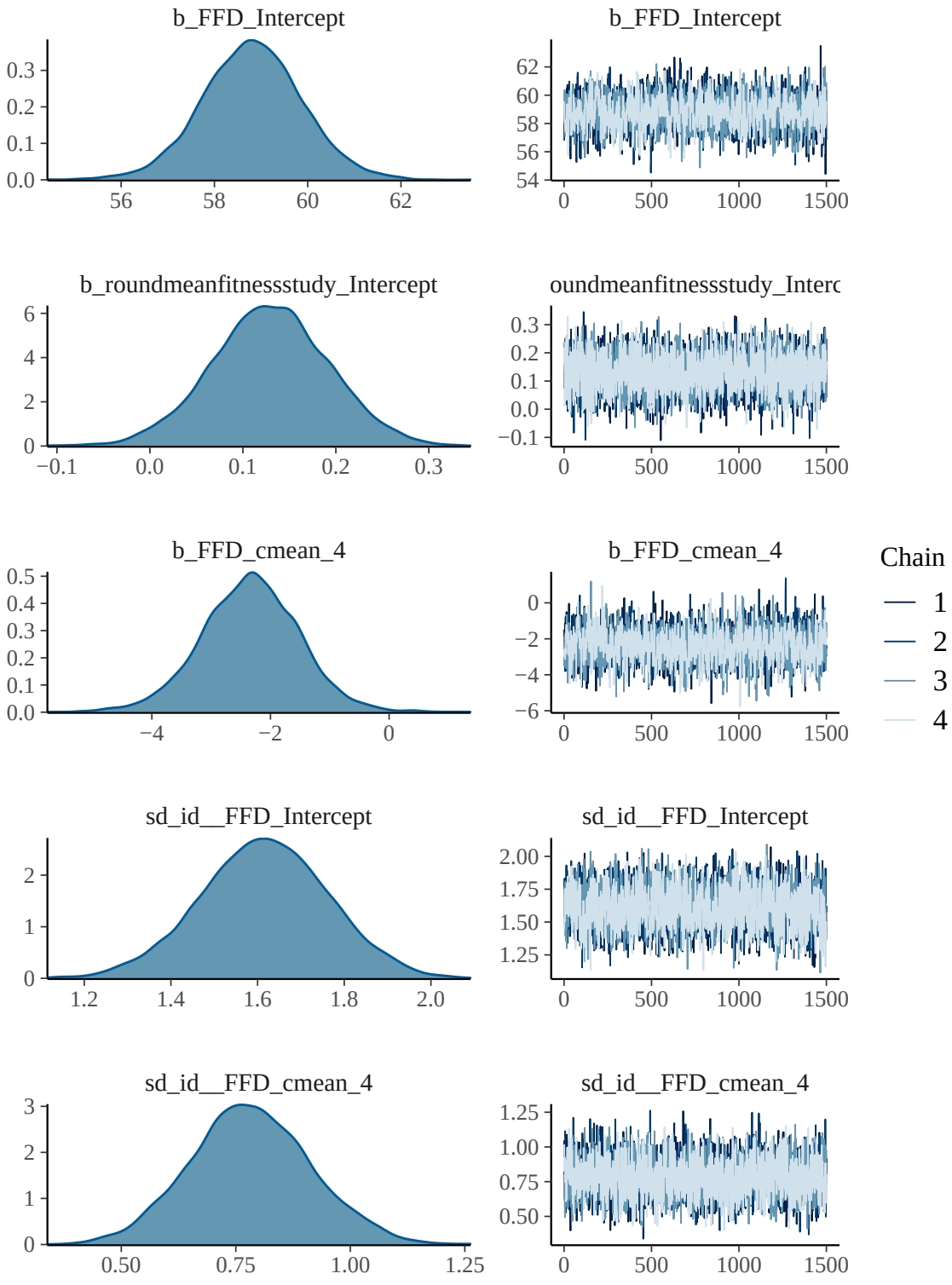
```
summary(bivar3.all.brm.zinb)
```

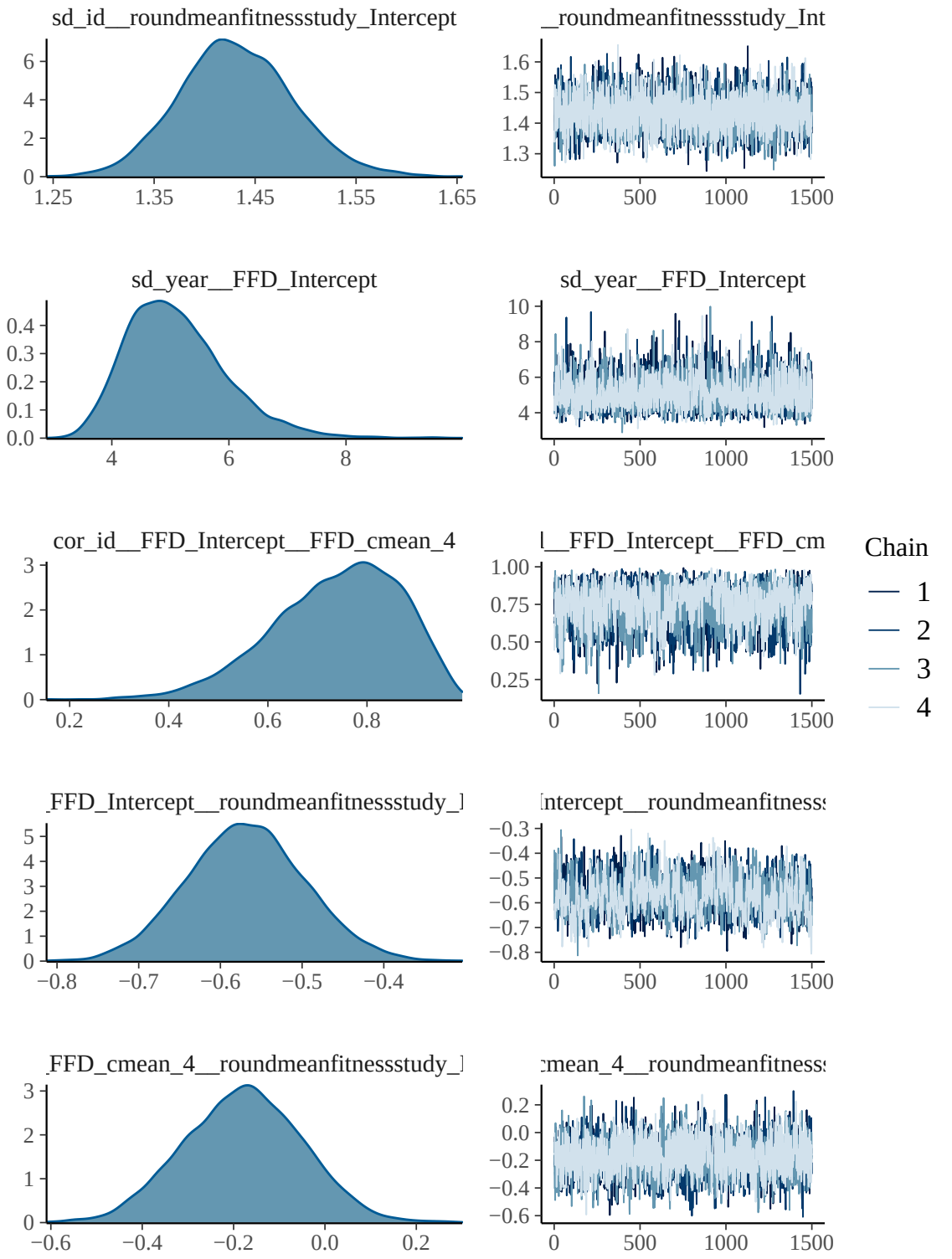
```
##  Family: MV(gaussian, zero_inflated_negbinomial)
##   Links: mu = identity; sigma = identity
##          mu = log; shape = identity; zi = identity
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_study) ~ (1 | ID1 | id)
##    Data: datadef (Number of observations: 2478)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 2;
##          total post-warmup samples = 2000
##
## 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)                                      0.79      0.13     0.53
## sd(roundmeanfitnessstudy_Intercept)                  1.43      0.06     1.33
## cor(FFD_Intercept,FFD_cmean_4)                       0.73      0.12     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)                                    1.89 1.01      833
## sd(FFD_cmean_4)                                      1.06 1.01      960
## sd(roundmeanfitnessstudy_Intercept)                  1.55 1.00     1120
## cor(FFD_Intercept,FFD_cmean_4)                       0.93 1.02      254
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)   -0.44 1.01      296
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)      0.07 1.01      416
##                                                  Tail_ESS
## sd(FFD_Intercept)                                    1480
```

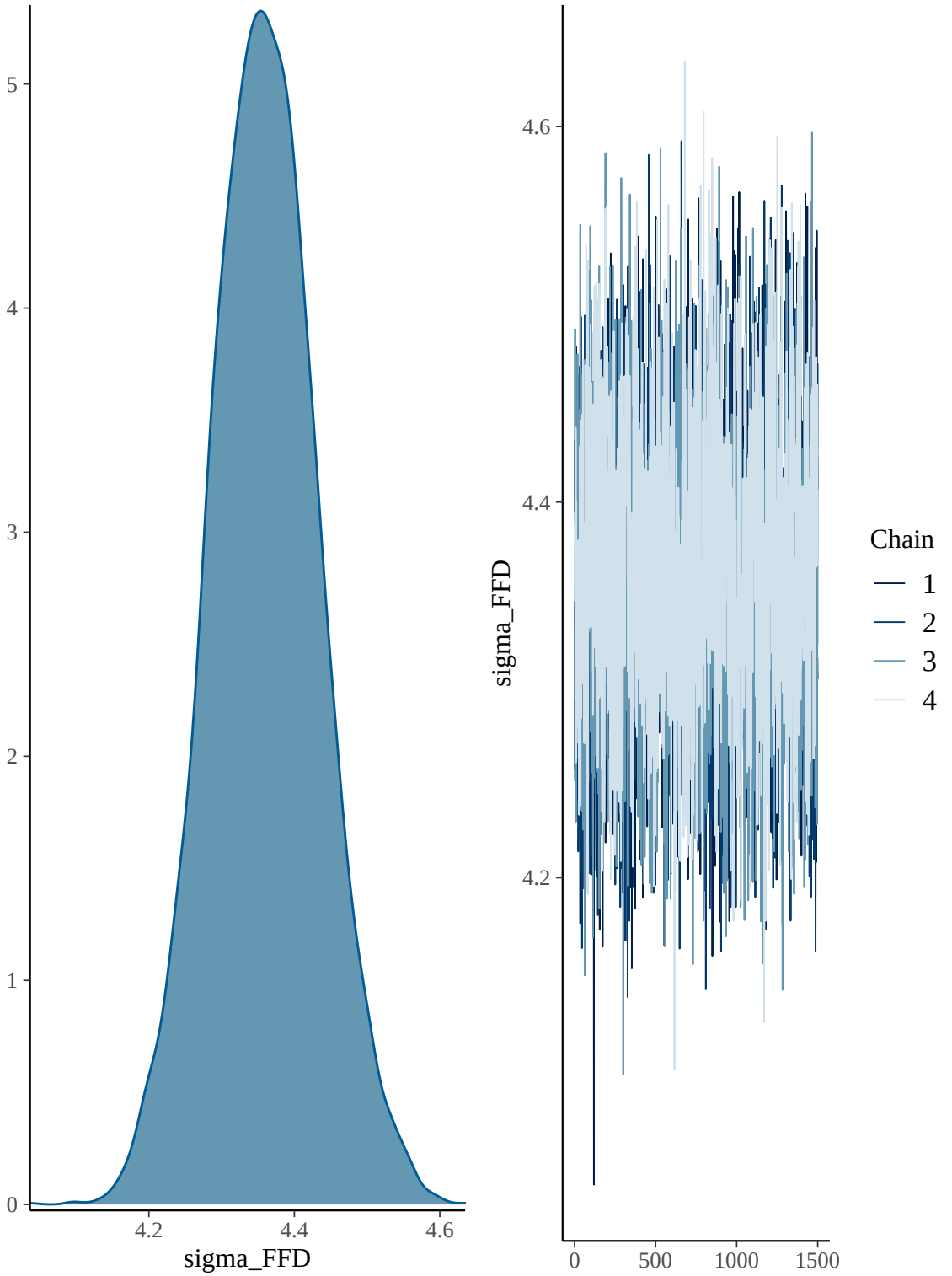
```
## sd(FFD_cmean_4)                                        1643
## sd(roundmeanfitnessstudy_Intercept)                    1322
## cor(FFD_Intercept,FFD_cmean_4)                         1281
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)      606
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)        861
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.12      0.84     3.79     7.03 1.00      912     1307
##
## Population-Level Effects:
##                               Estimate Est.Error l-95% CI u-95% CI Rhat
## FFD_Intercept                    58.75      1.12    56.56    61.05 1.00
## roundmeanfitnessstudy_Intercept     0.13      0.06     0.00     0.25 1.00
## FFD_cmean_4                      -2.30      0.87    -4.02    -0.57 1.00
##                               Bulk_ESS Tail_ESS
## FFD_Intercept                      725      923
## roundmeanfitnessstudy_Intercept     1738     1676
## FFD_cmean_4                        798      849
##
## Family Specific Parameters:
##                             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sigma_FFD                       4.35      0.07     4.21     4.50 1.01      957
## shape_roundmeanfitnessstudy   551.16    170.21   302.39   955.60 1.00     1399
## zi_roundmeanfitnessstudy        0.00      0.00     0.00     0.00 1.00     1711
##                             Tail_ESS
## sigma_FFD                        981
## shape_roundmeanfitnessstudy      867
## zi_roundmeanfitnessstudy        1623
##
## 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

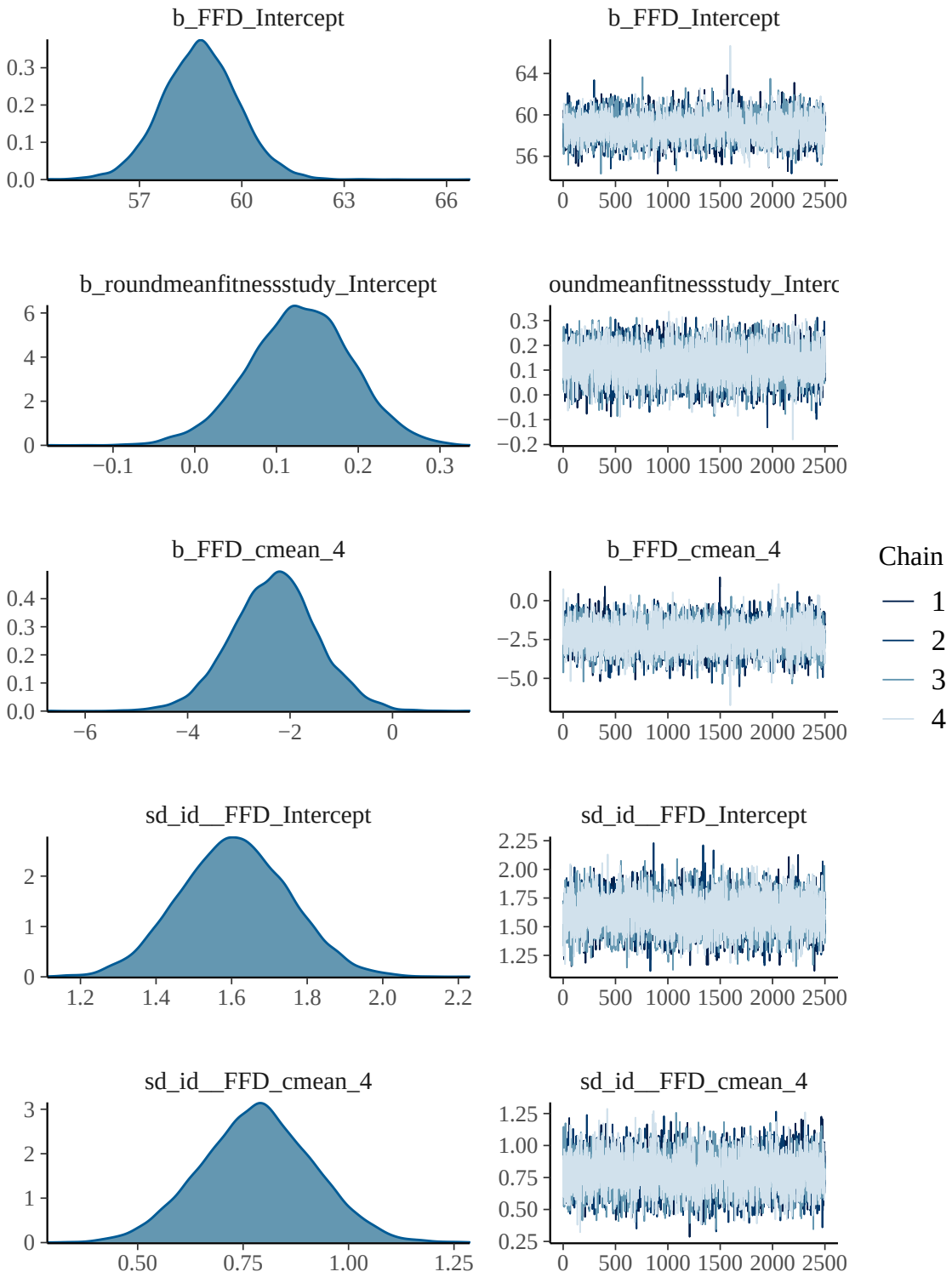
```
plot(bivar3.all.brm.pois)
```

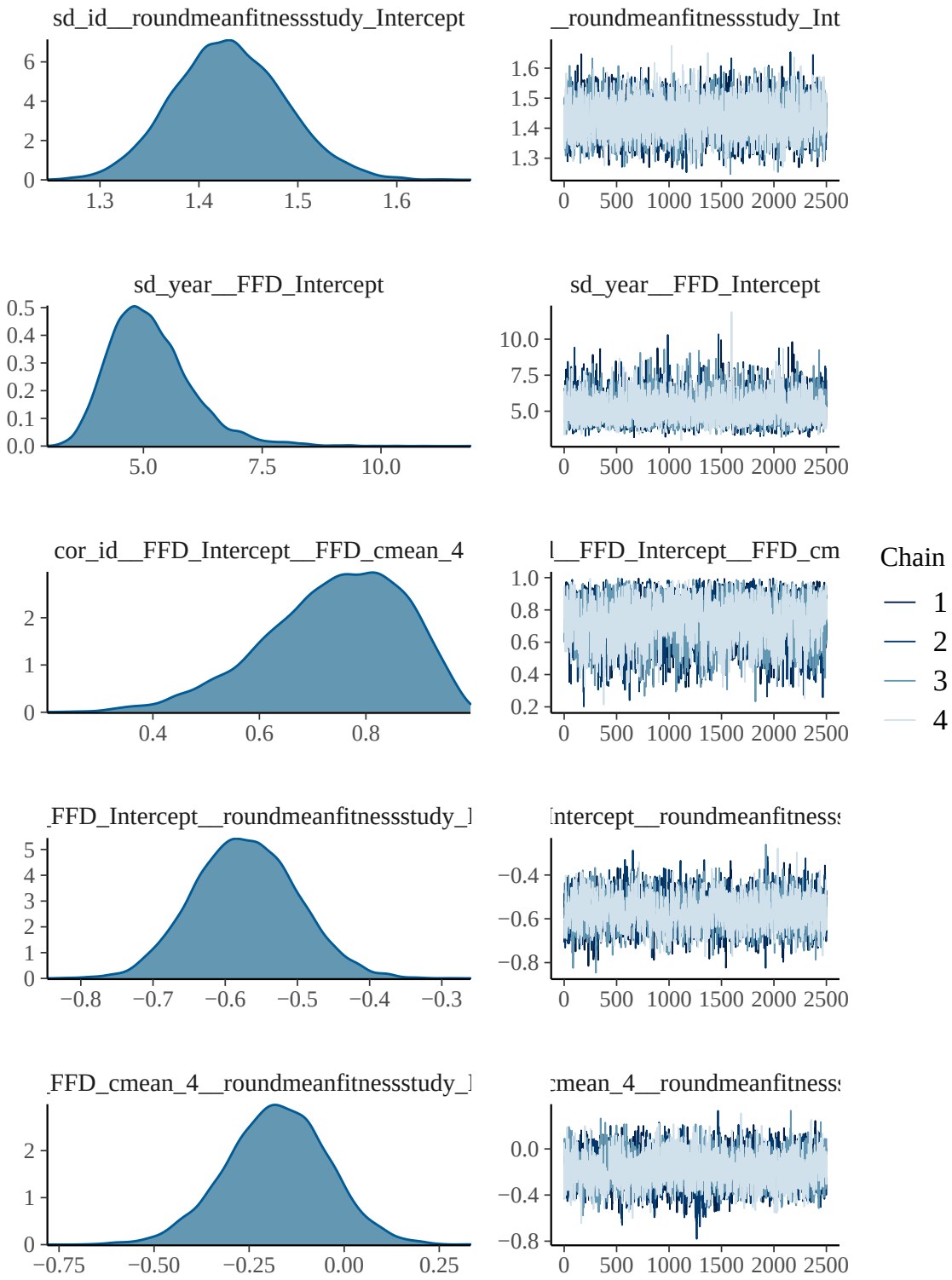

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessstudy_Intercept
oundmeanfitnessstudy_Interc
b_FFD_cmean_4
b_FFD_cmean_4
sd_id__FFD_Intercept
sd_id__FFD_Intercept
sd_id__FFD_cmean_4
sd_id__FFD_cmean_4
Chain
1
2
3
4

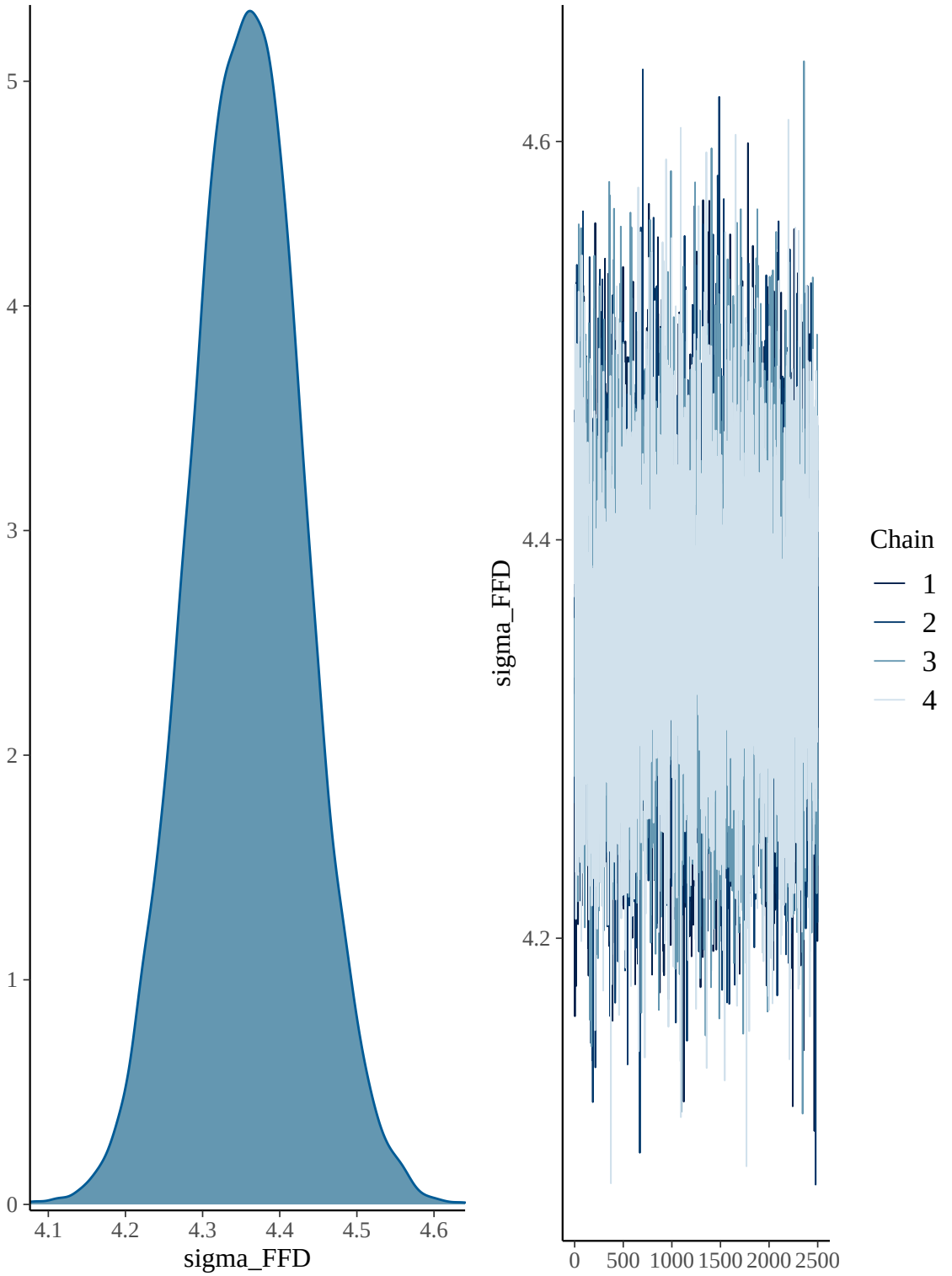

sd_id__roundmeanfitnessstudy_Intercept
__roundmeanfitnessstudy_Int
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
_FFD_Intercept__roundmeanfitnessstudy_
ntercept__roundmeanfitness
FFD_cmean_4__roundmeanfitnessstudy_
cmean_4__roundmeanfitness
Chain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



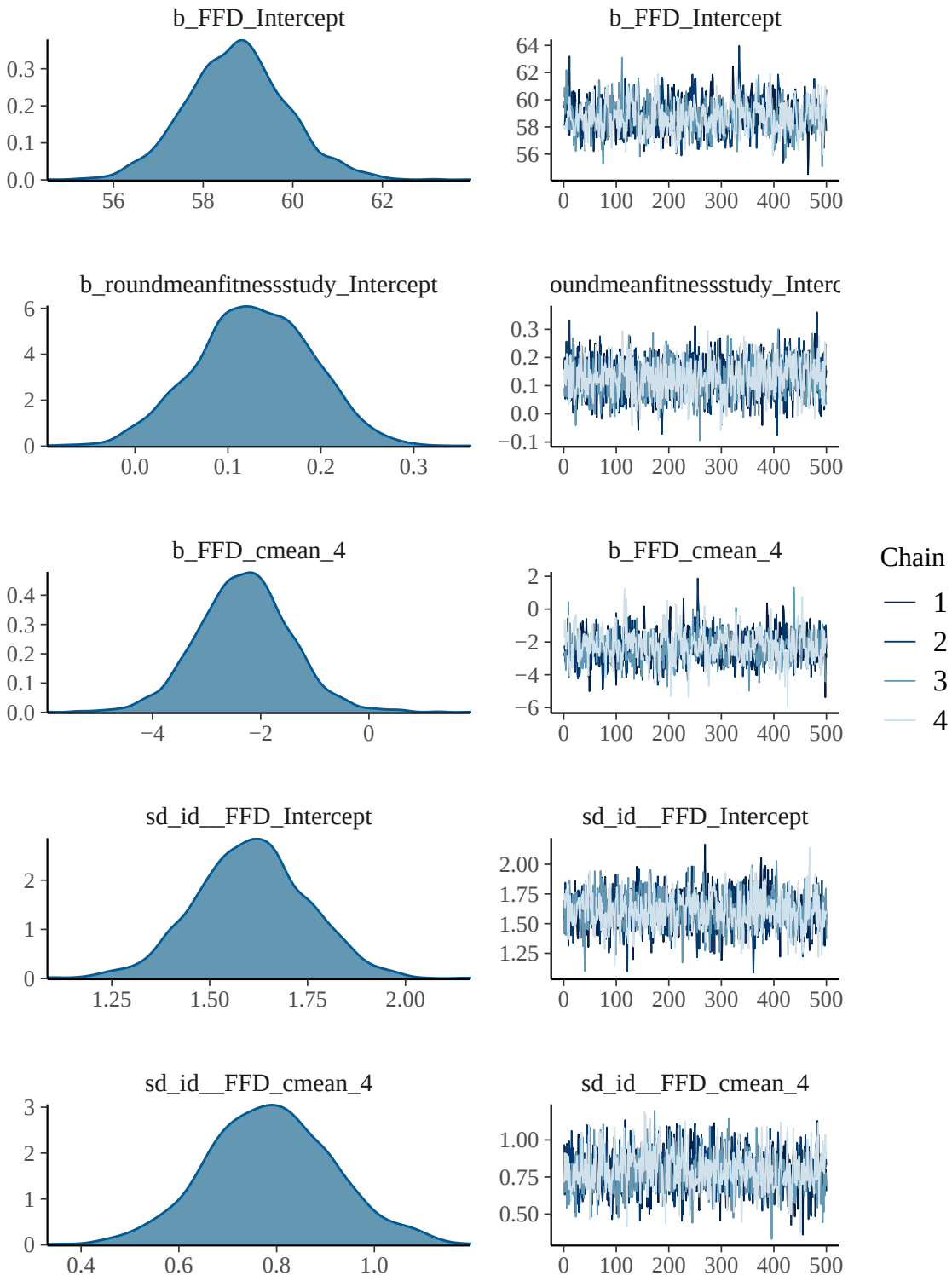

```
plot(bivar3.all.brm.nb)
```

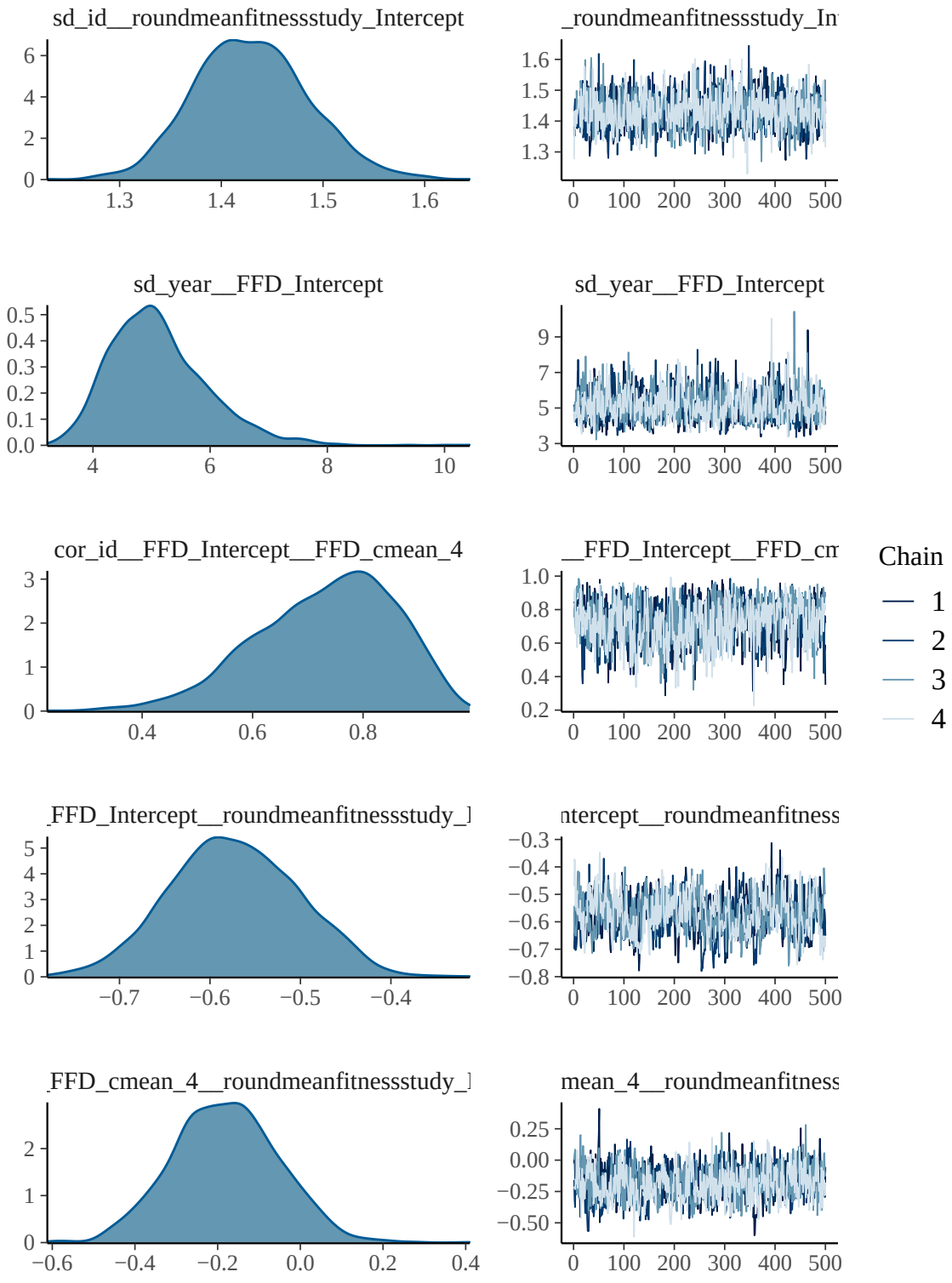

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessstudy_Intercept
oundmeanfitnessstudy_Interc
b_FFD_cmean_4
b_FFD_cmean_4
Chain
1
2
3
4
sd_id__FFD_Intercept
sd_id__FFD_Intercept
sd_id__FFD_cmean_4
sd_id__FFD_cmean_4

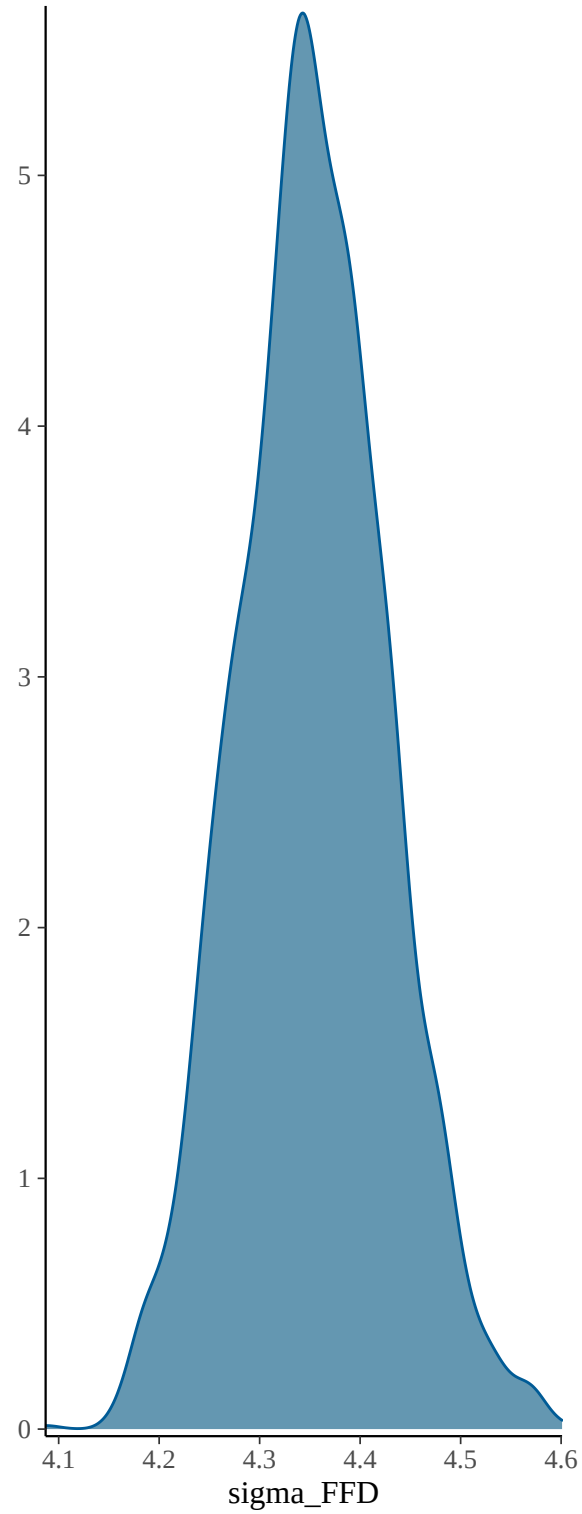

sd_id__roundmeanfitnessstudy_Intercept
__roundmeanfitnessstudy_Int
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
FFD_Intercept__roundmeanfitnessstudy_
ntercept__roundmeanfitness
FFD_cmean_4__roundmeanfitnessstudy_
cmean_4__roundmeanfitness
Chain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

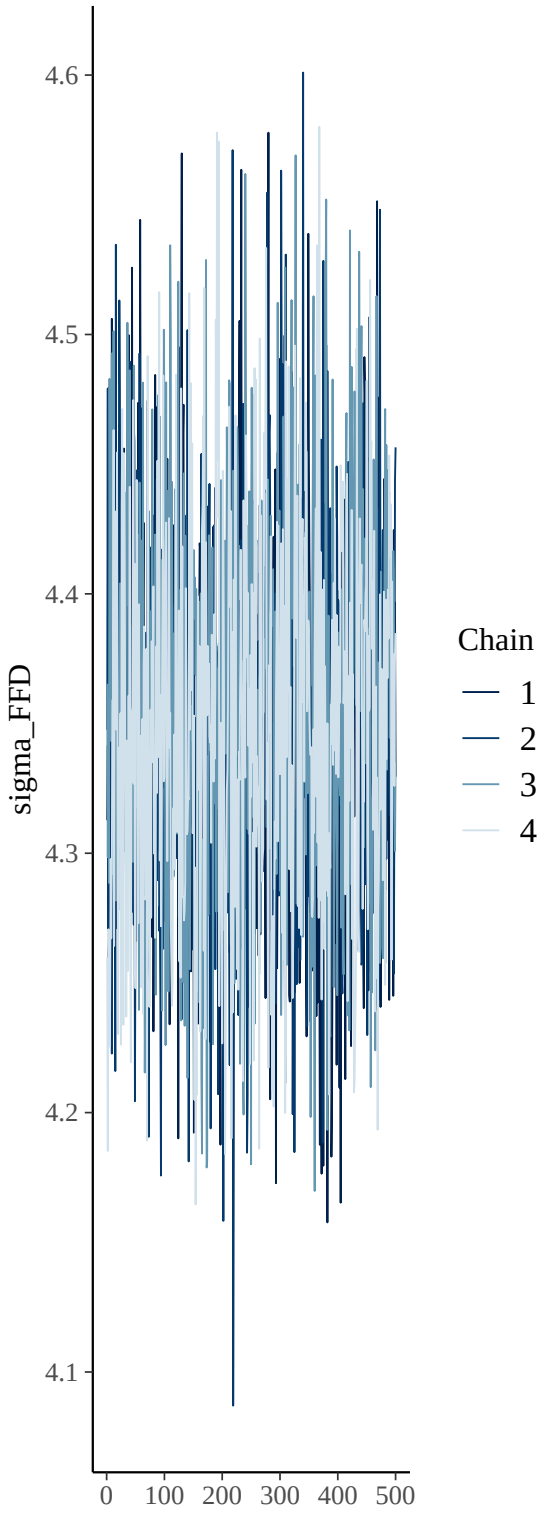



```
plot(bivar3.all.brm.zinb)
```


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

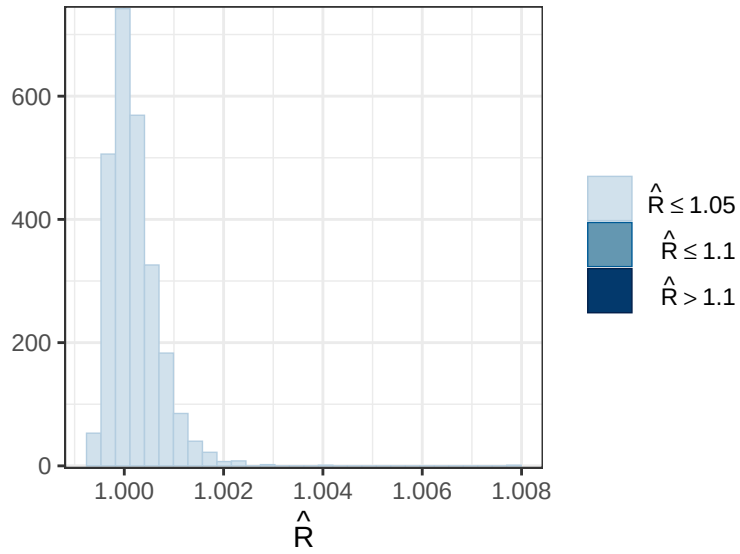

sd_id__roundmeanfitnessstudy_Intercept
_roundmeanfitnessstudy_In
sd_year__FFD_Intercept
sd_year__FFD_Intercept
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
Chain
1
2
3
4
_FFD_Intercept__roundmeanfitnessstudy_
ntercept__roundmeanfitness
_FFD_cmean_4__roundmeanfitnessstudy_
mean_4__roundmeanfitness


5
4
3
2
1
0
4.1
4.2
4.3
4.4
4.5
4.6
sigma_FFD

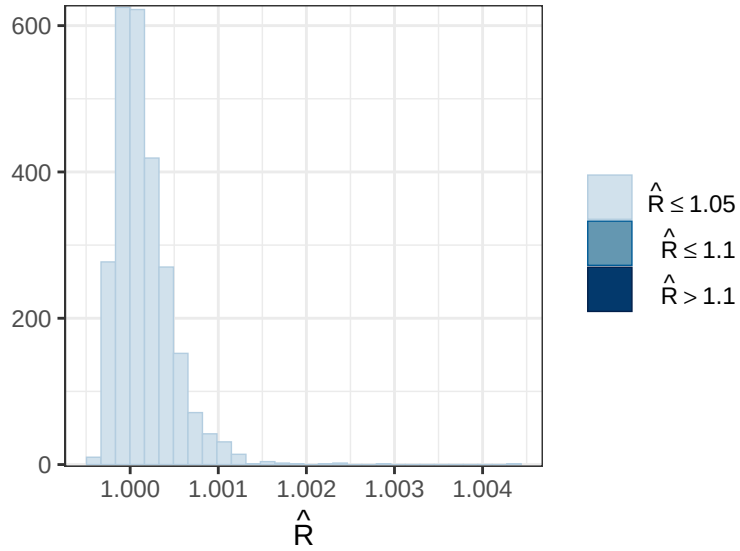

4.6
4.5
4.4
4.3
4.2
4.1
sigma_FFD
0
100
200
300
400
500
Chain
1
2
3
4

Rhat

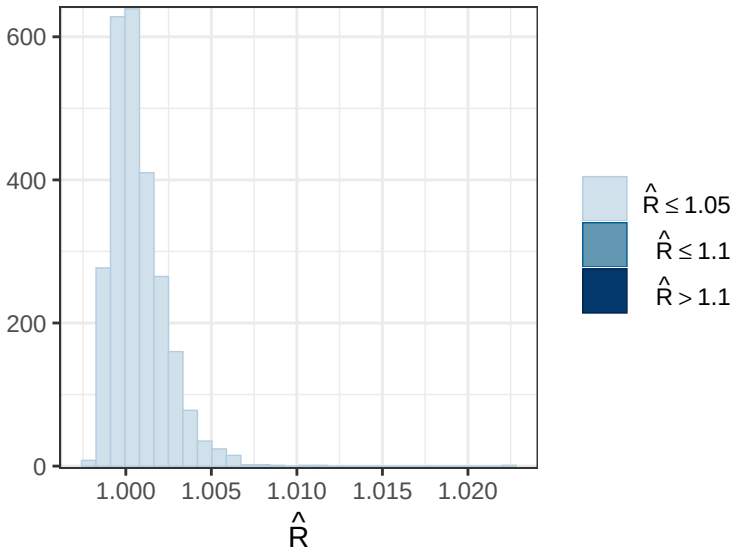
```
mcmc_rhat_hist(rhat(bivar3.all.brm.pois))+theme_bw()
```




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

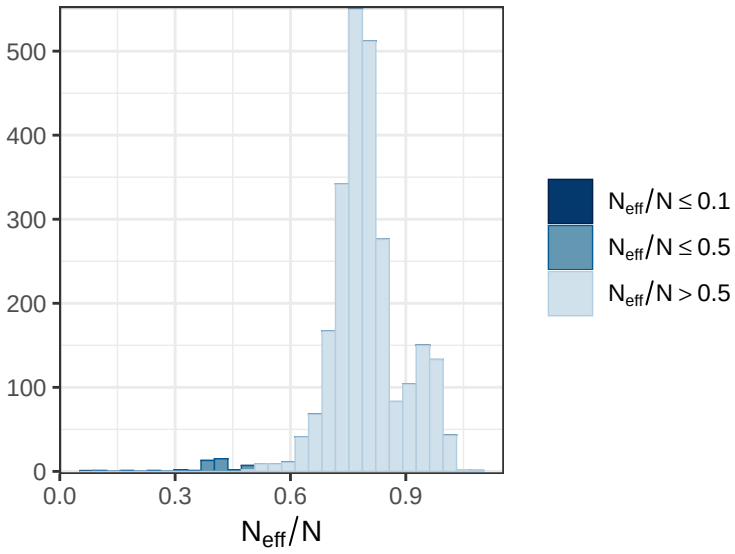



```
mcmc_rhat_hist(rhat(bivar3.all.brm.zinb))+theme_bw()
```

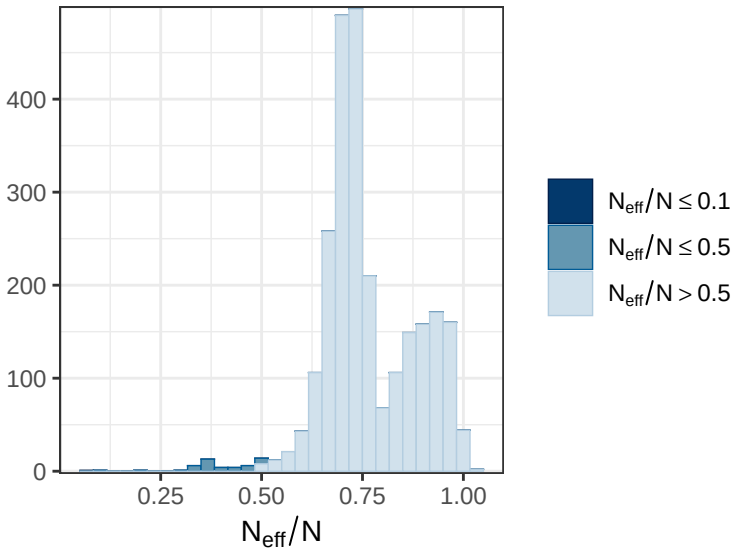


Effective sample size:

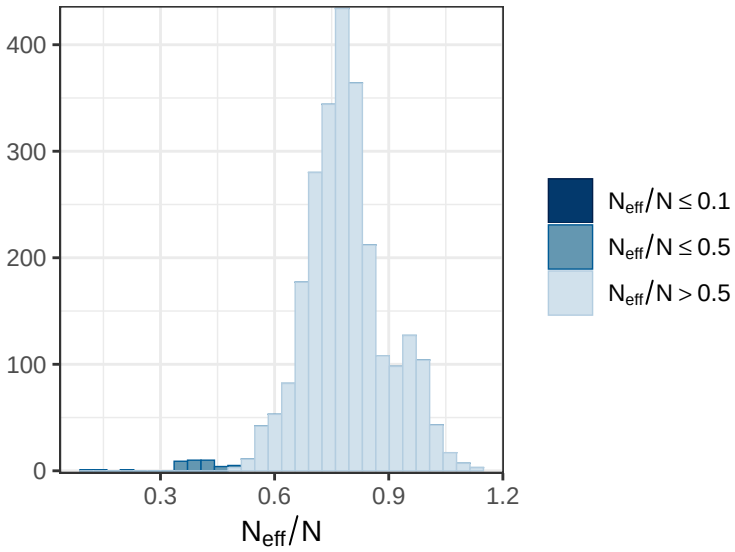
```
mcmc_neff_hist(neff_ratio(bivar3.all.brm.pois))+theme_bw()
```




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




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



Posterior predictive checks:

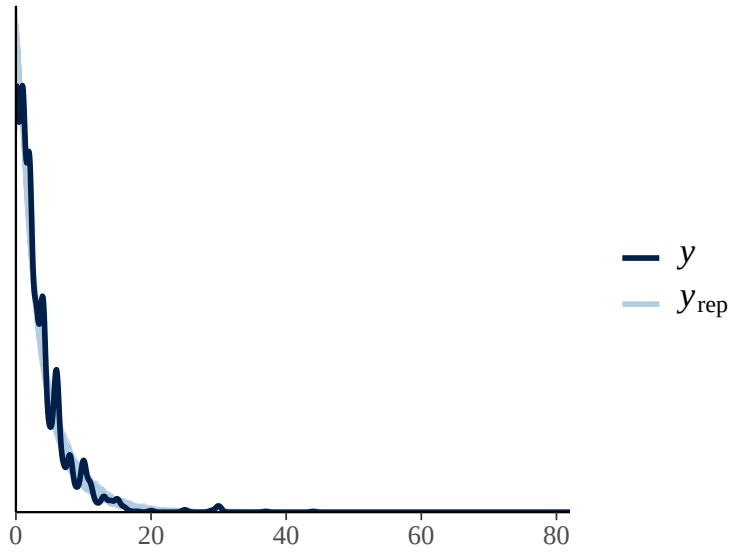
```
y3_fitness<-round(subset(datadef,!is.na(FFD))$mean_fitness_study)
yrep4_fitness_pois<-posterior_predict(bivar3.all.brm.pois,
                                      draws = 500,resp="roundmeanfitnessstudy")
yrep4_FFD_pois<-posterior_predict(bivar3.all.brm.pois,
                                  draws = 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")
yrep4_fitness_zinb<-posterior_predict(bivar3.all.brm.nb,
                                      draws = 500,resp="roundmeanfitnessstudy")
yrep4_FFD_zinb<-posterior_predict(bivar3.all.brm.nb,
```

```
                                draws = 500,resp="FFD")
# matrices of draws from the posterior predictive distribution
```

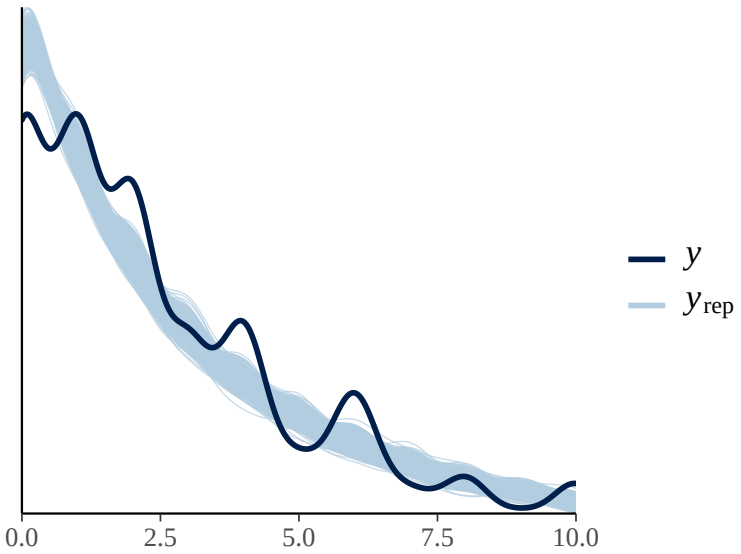
Each row of the matrix is a draw from the posterior predictive distribution, i.e. a vector with one element for each of the data points in y.

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

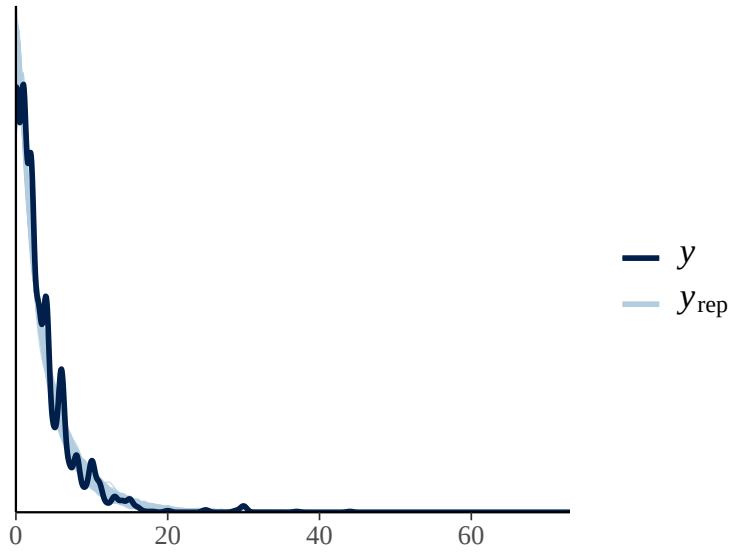
```
ppc_dens_overlay(y3_fitness, yrep4_fitness_pois[1:500,])
```



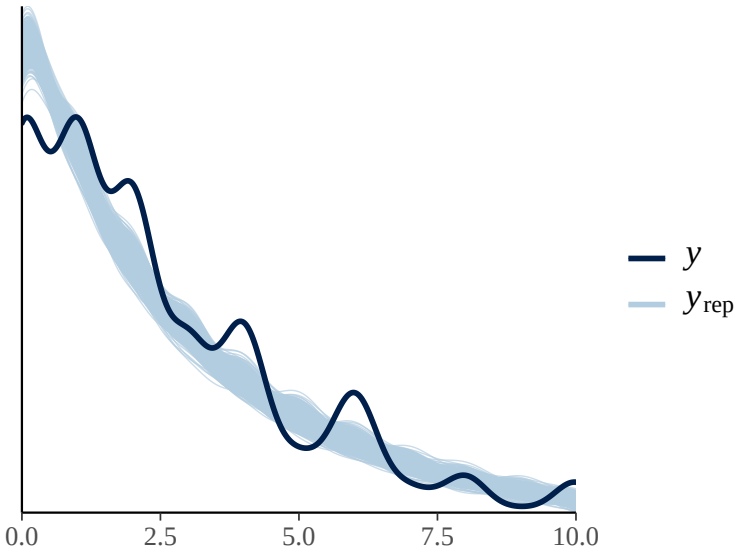

```
ppc_dens_overlay(y3_fitness, yrep4_fitness_pois[1:500,])+xlim(0, 10)
```



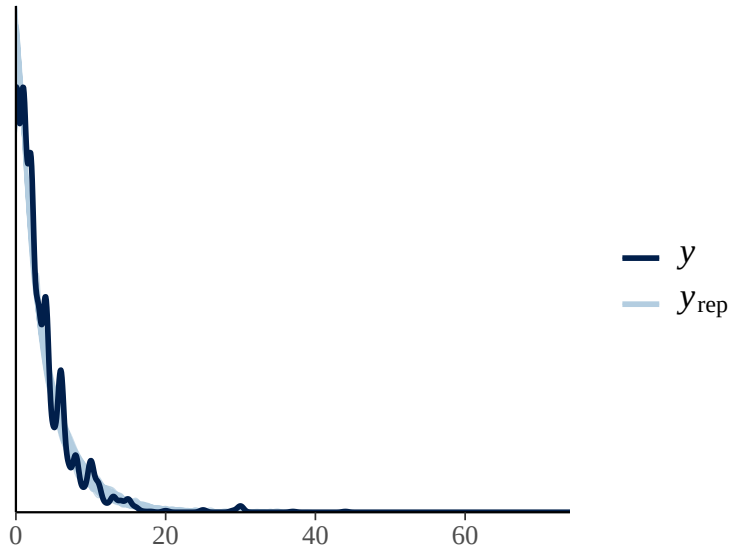

```
ppc_dens_overlay(y3_fitness, yrep4_fitness_nb[1:500,])
```



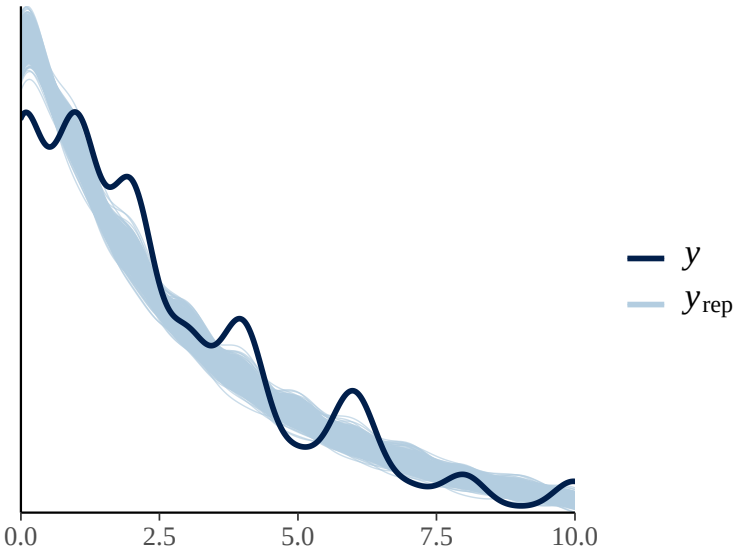

```
ppc_dens_overlay(y3_fitness, yrep4_fitness_nb[1:500,])+xlim(0,10)
```



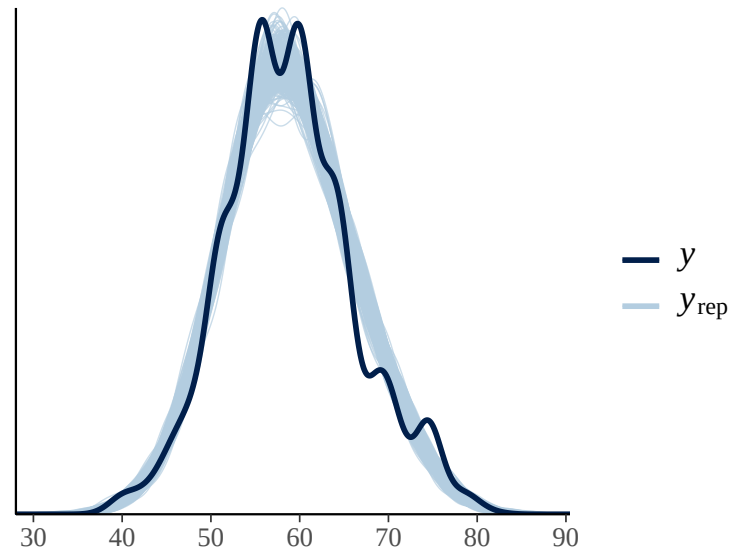

```
ppc_dens_overlay(y3_fitness, yrep4_fitness_zinb[1:500,])
```



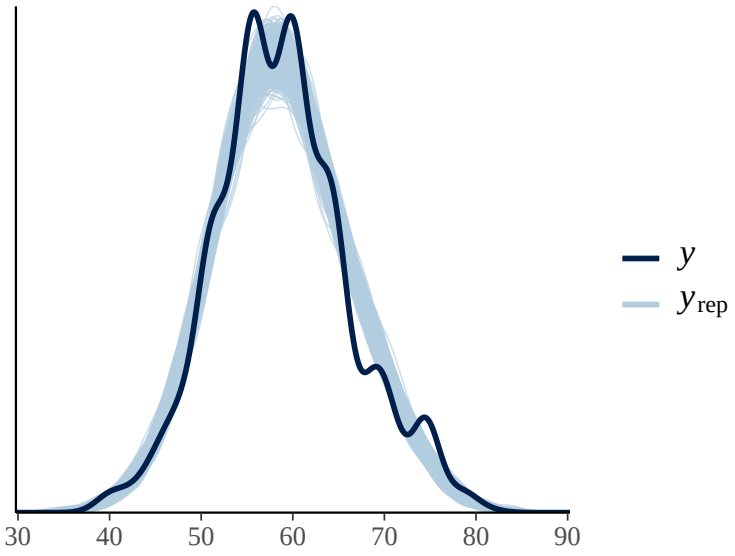

```
ppc_dens_overlay(y3_fitness, yrep4_fitness_zinb[1:500,])+xlim(0,10)
```



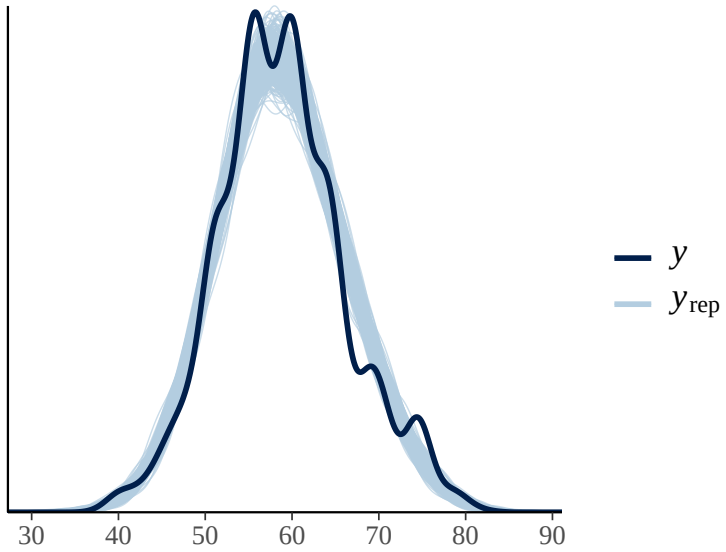

```
ppc_dens_overlay(y2_FFD, yrep4_FFD_pois[1:500,])
```




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



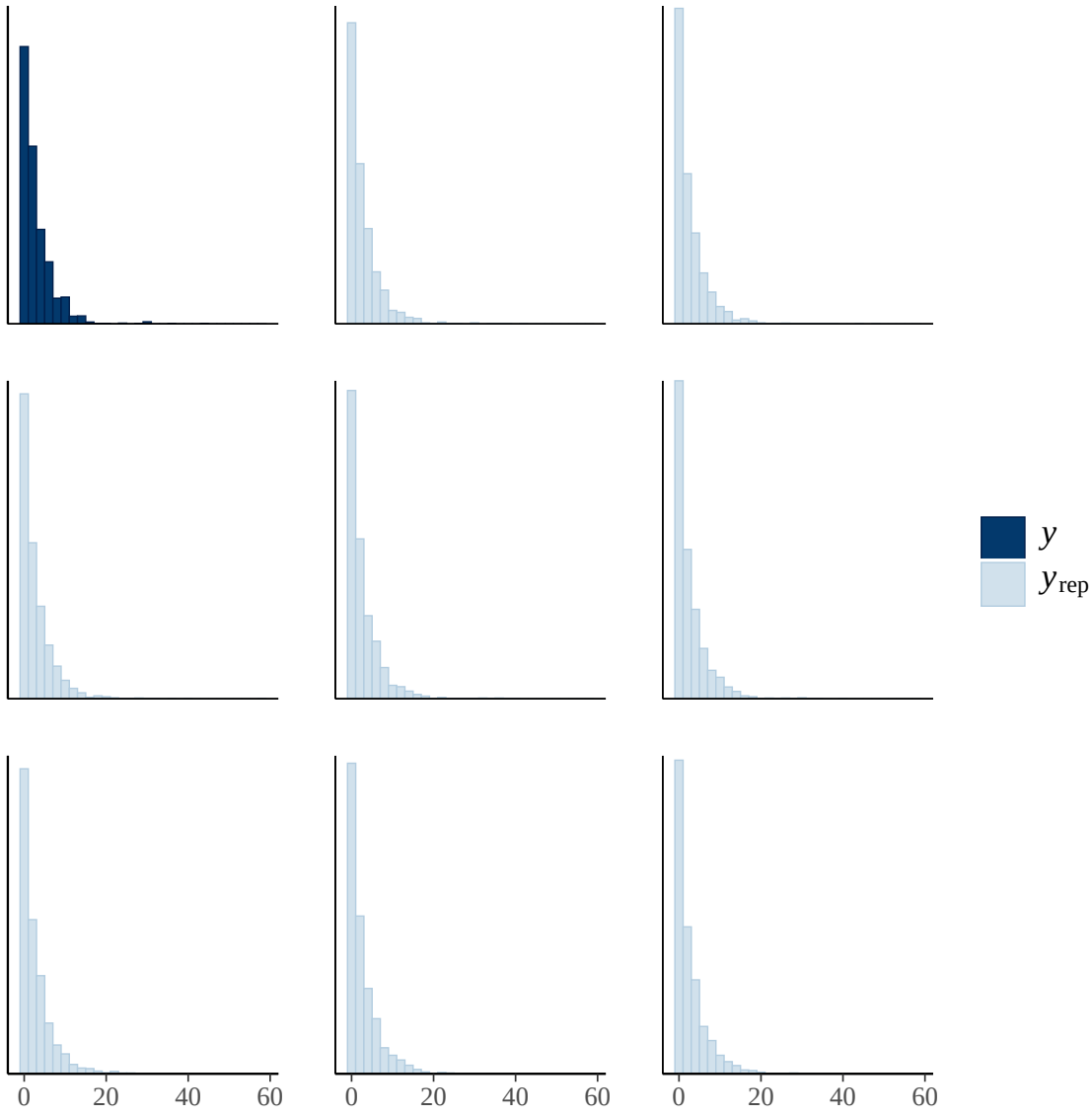

```
ppc_dens_overlay(y2_FFD, yrep4_FFD_zinb[1:500,])
```



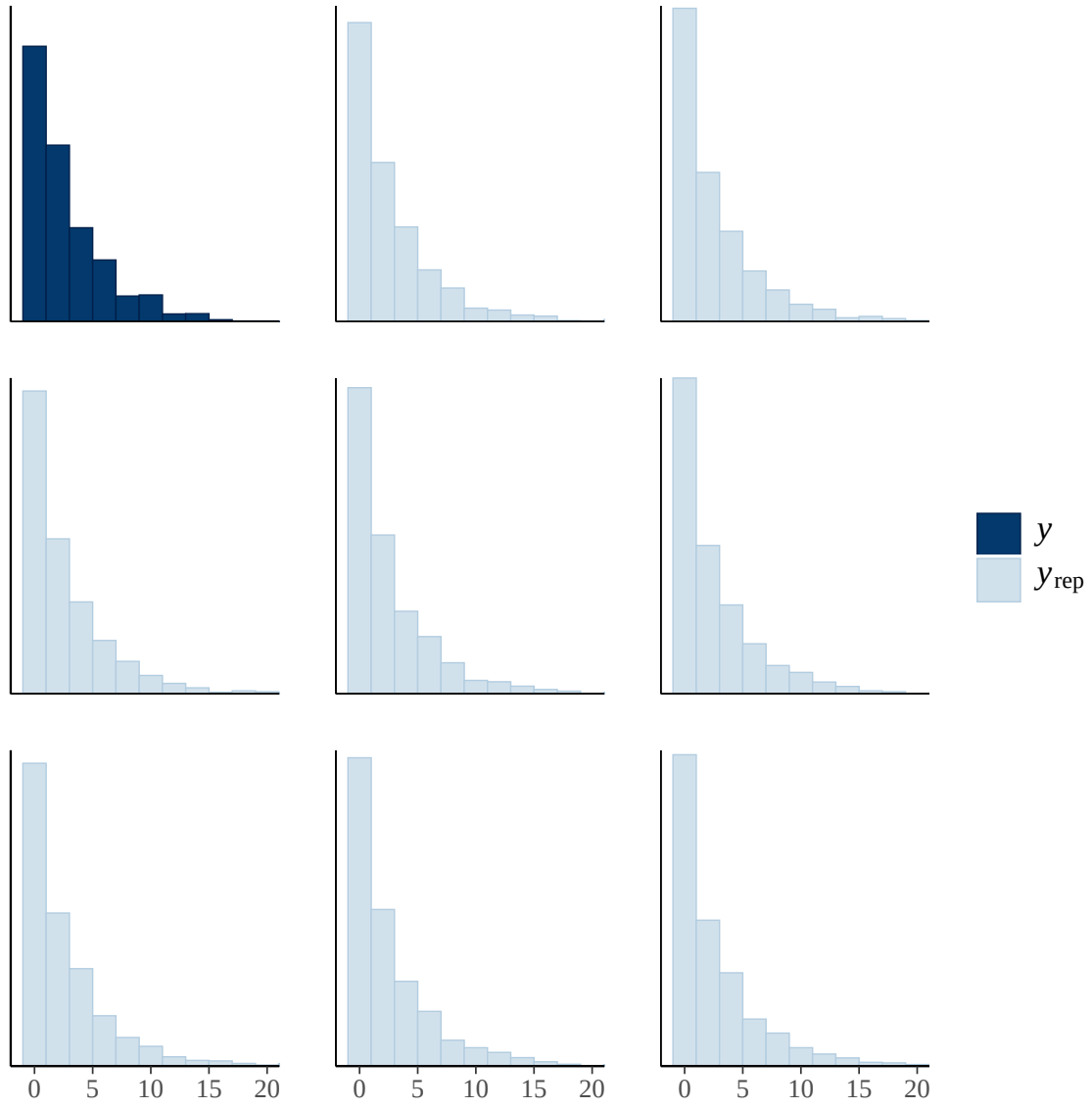
Poisson and negative binomial look pretty similar.

Separate histograms of y and some of the yrep datasets

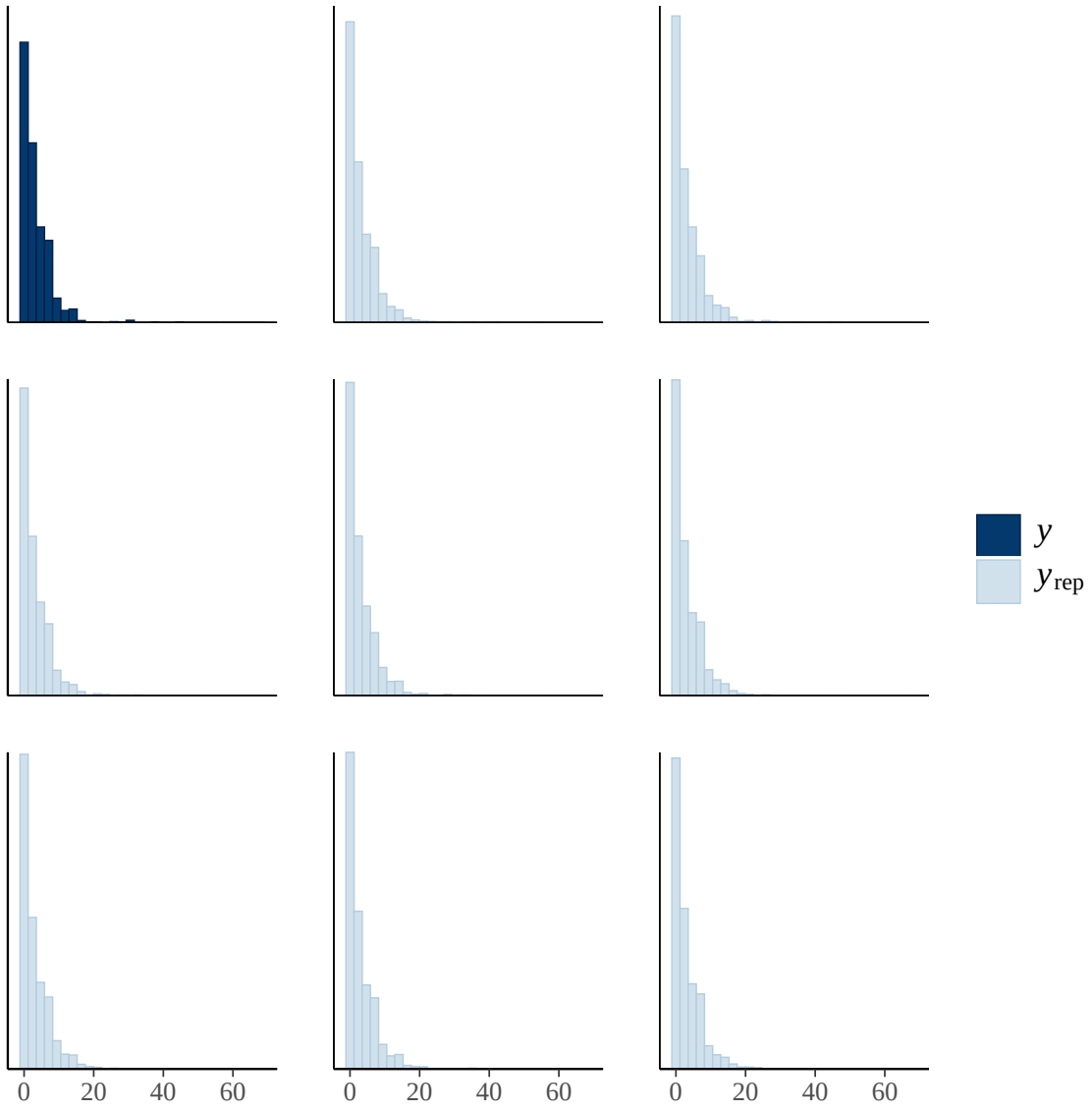
```
ppc_hist(y3_fitness, yrep4_fitness_pois[1:8, ])
```



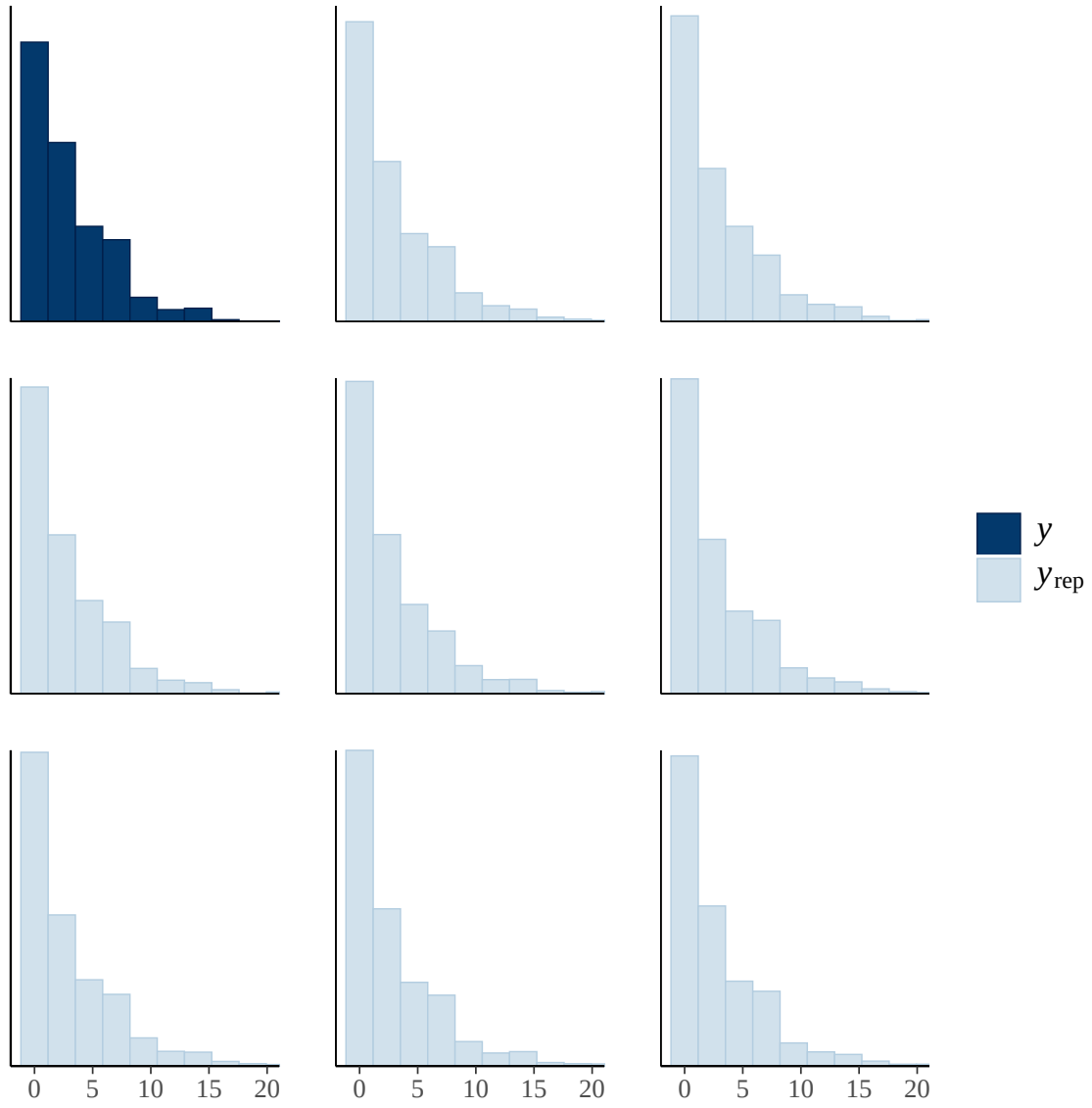

```
ppc_hist(y3_fitness, yrep4_fitness_pois[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```



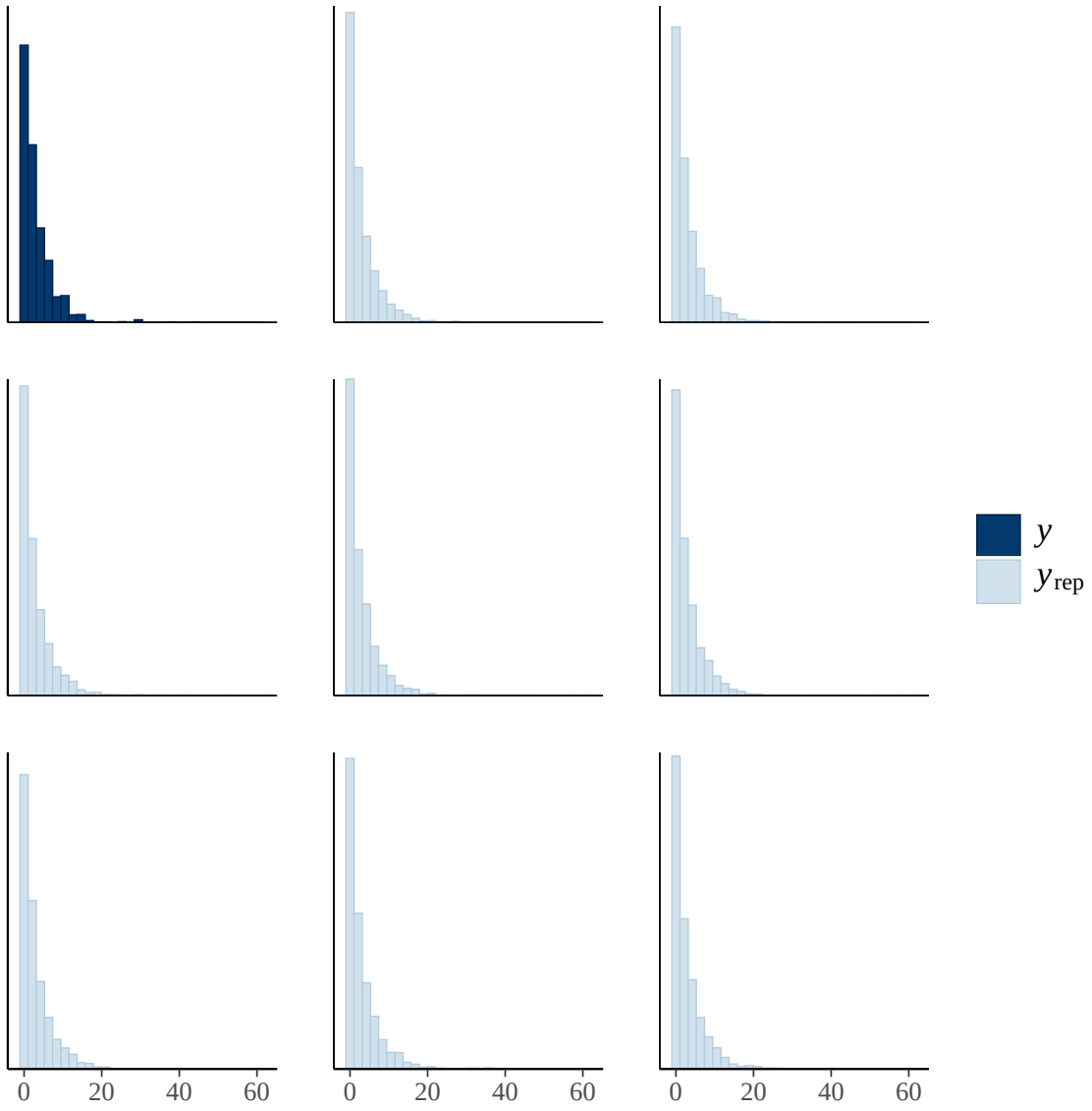

```
ppc_hist(y3_fitness, yrep4_fitness_nb[1:8, ])
```



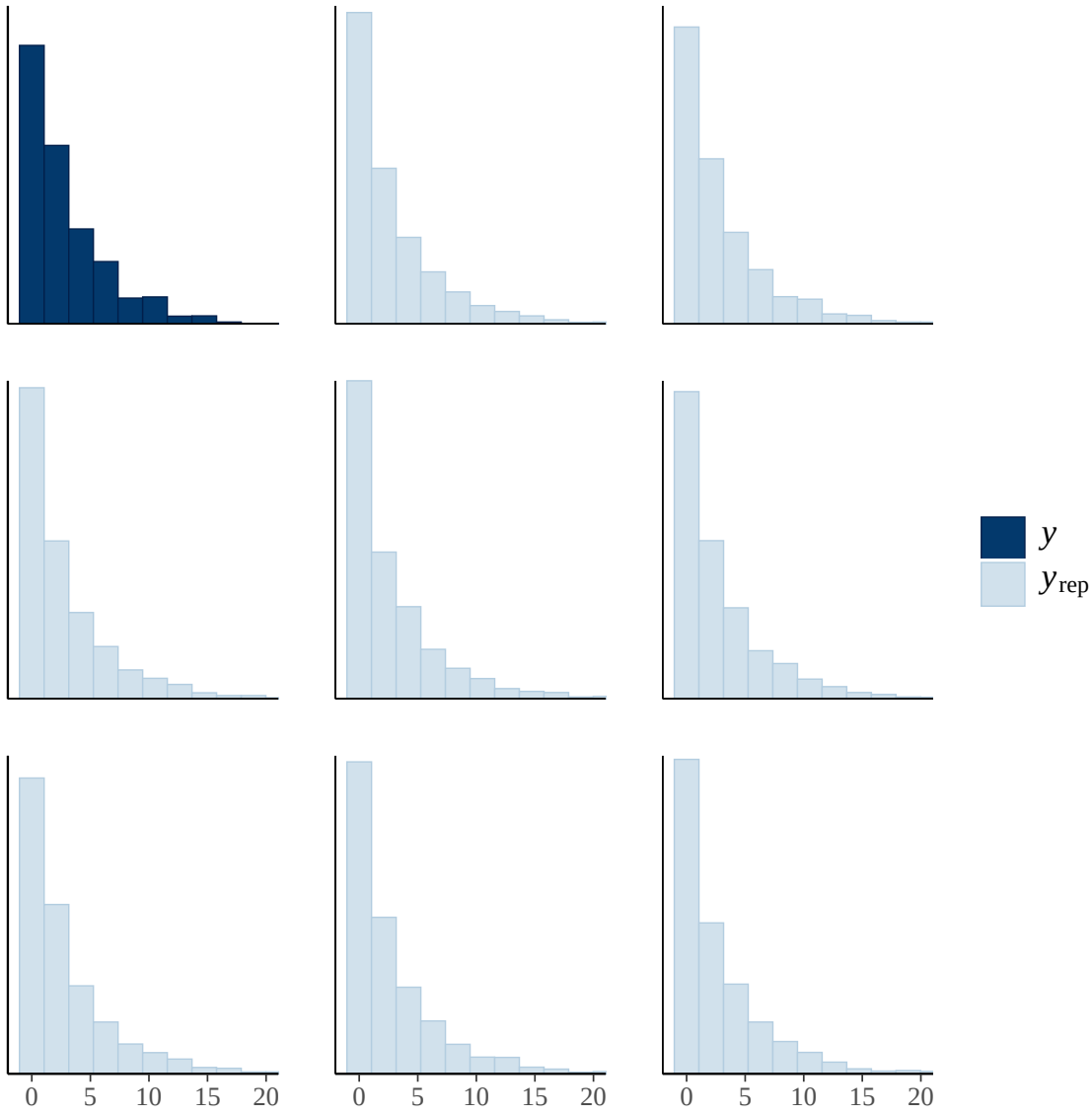

```
ppc_hist(y3_fitness, yrep4_fitness_nb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```




```
ppc_hist(y3_fitness, yrep4_fitness_zinb[1:8, ])
```

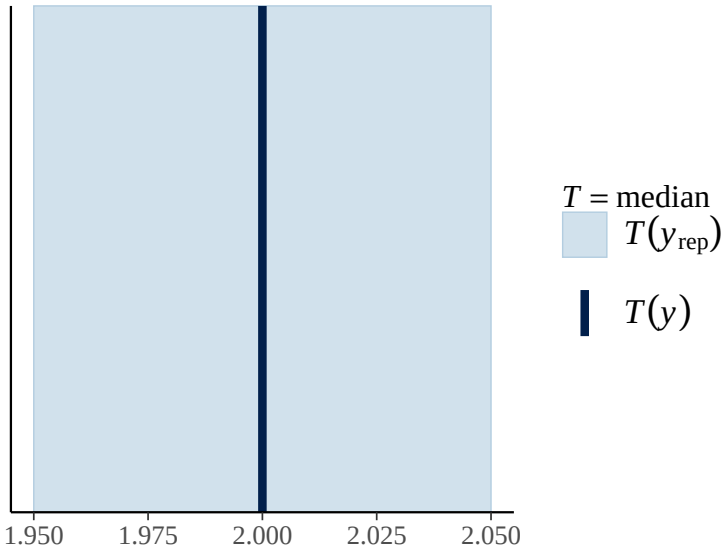



```
ppc_hist(y3_fitness, yrep4_fitness_zinb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```

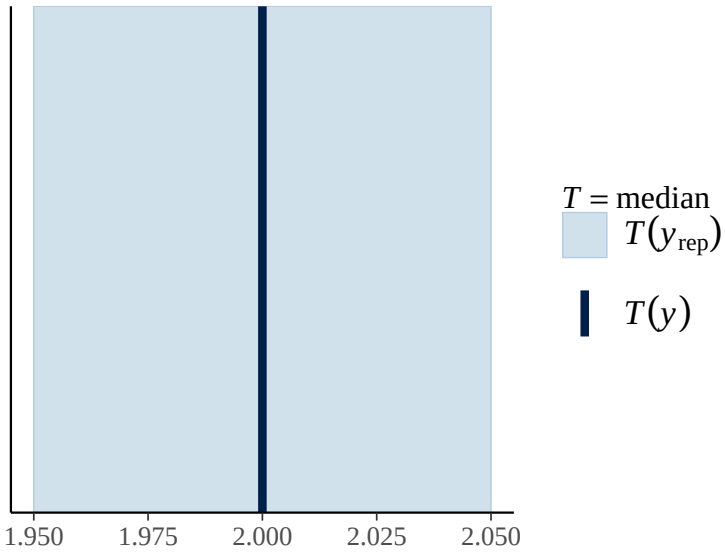


Here, negative binomial looks a bit better for fitness.

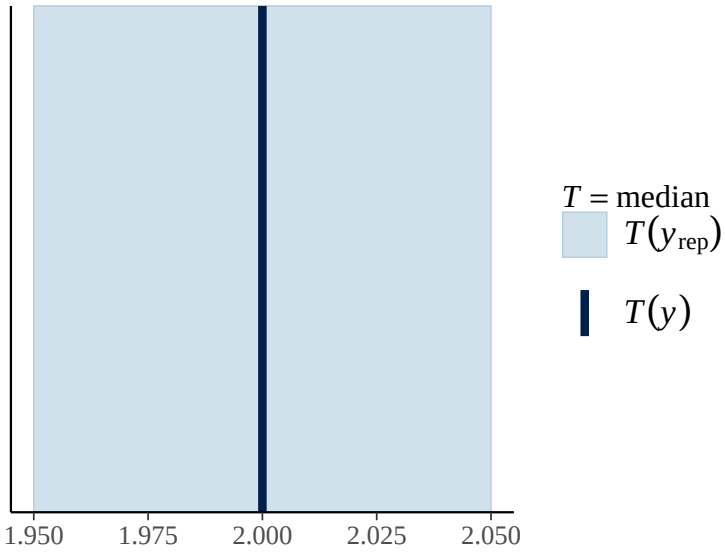
```
ppc_stat(y3_fitness, yrep4_fitness_pois,stat="median")
```



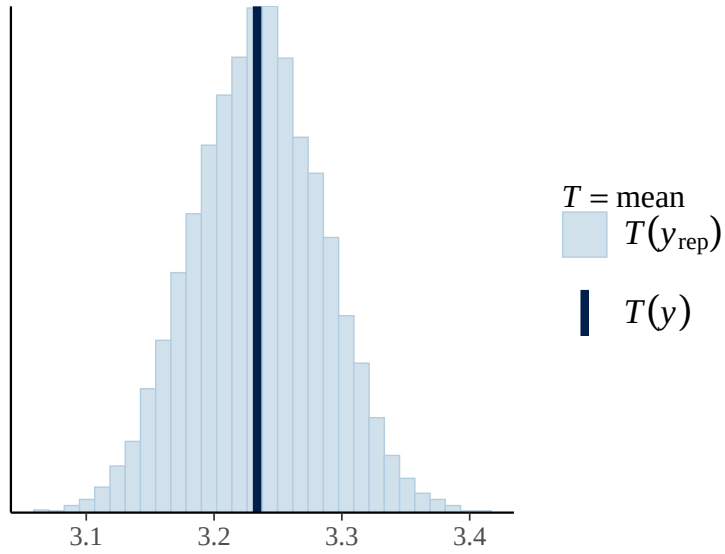

```
ppc_stat(y3_fitness, yrep4_fitness_nb,stat="median")
```



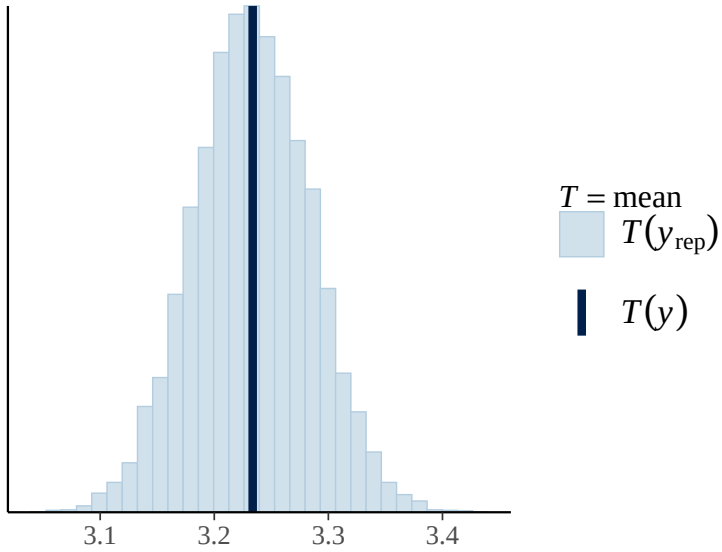

```
ppc_stat(y3_fitness, yrep4_fitness_zinb,stat="median")
```



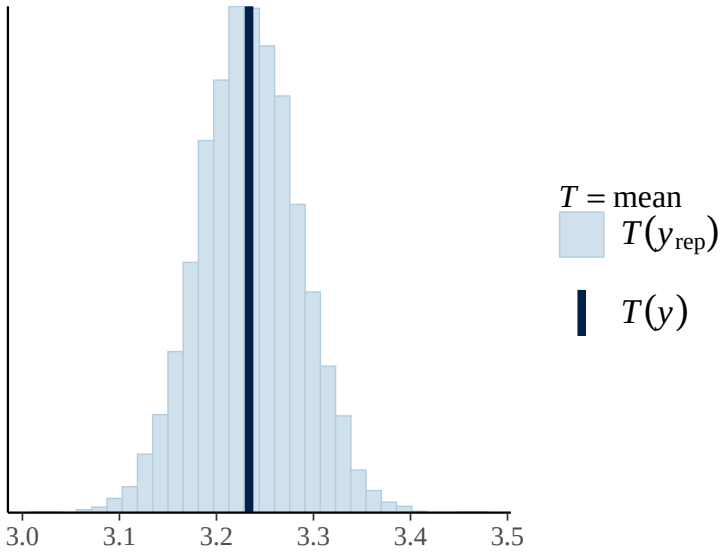

```
ppc_stat(y3_fitness, yrep4_fitness_pois,stat="mean")
```



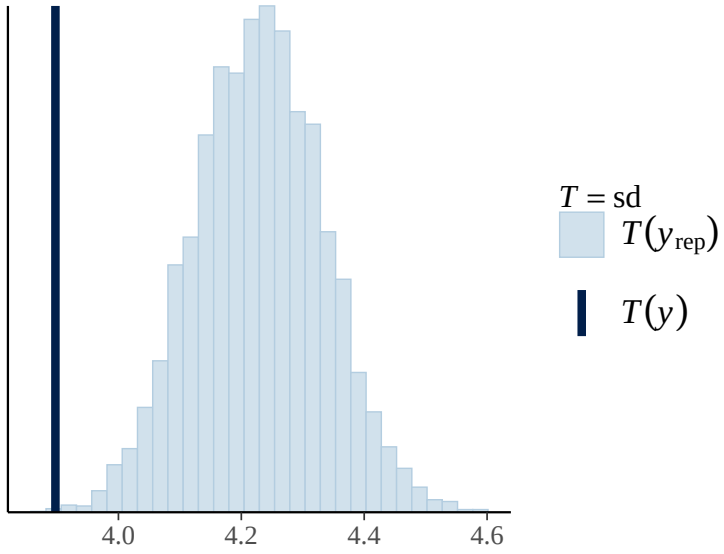

```
ppc_stat(y3_fitness, yrep4_fitness_nb,stat="mean")
```



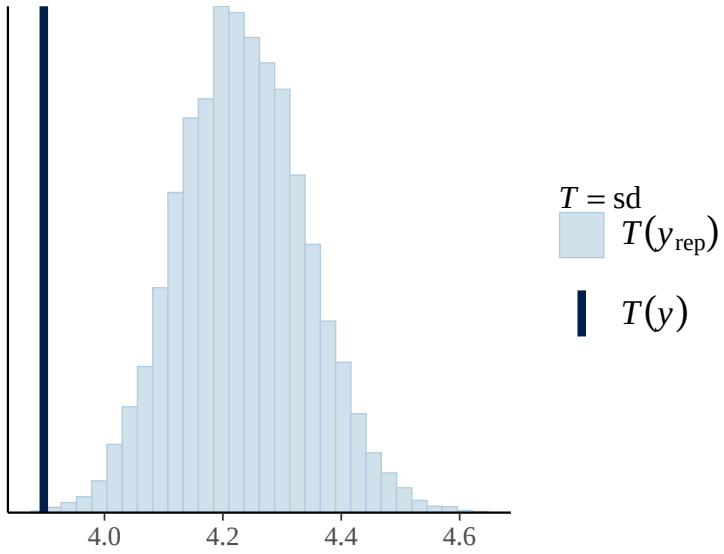

```
ppc_stat(y3_fitness, yrep4_fitness_zinb,stat="mean")
```



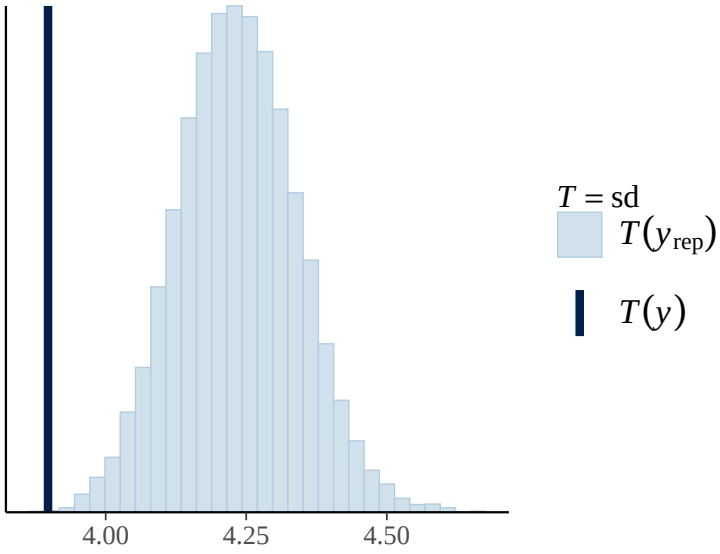

```
ppc_stat(y3_fitness, yrep4_fitness_pois,stat="sd")
```



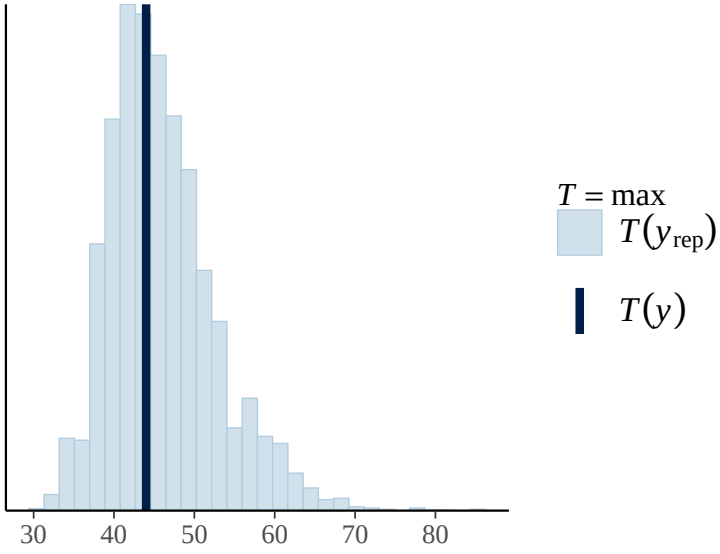

```
ppc_stat(y3_fitness, yrep4_fitness_nb,stat="sd")
```



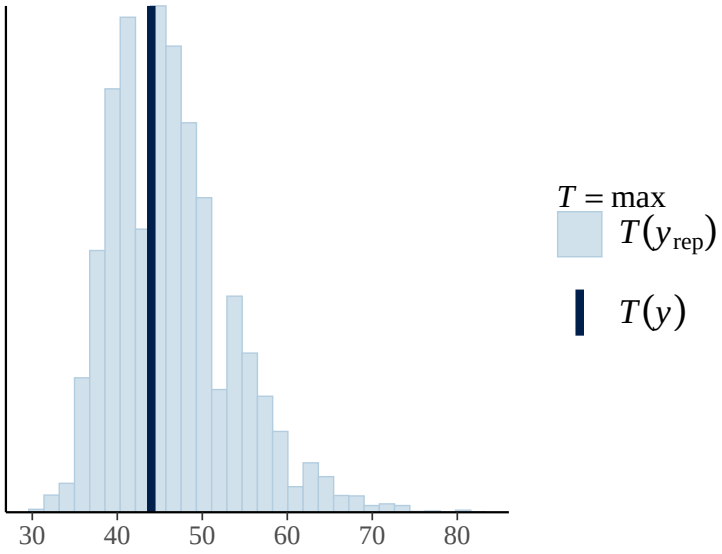

```
ppc_stat(y3_fitness, yrep4_fitness_zinb,stat="sd")
```



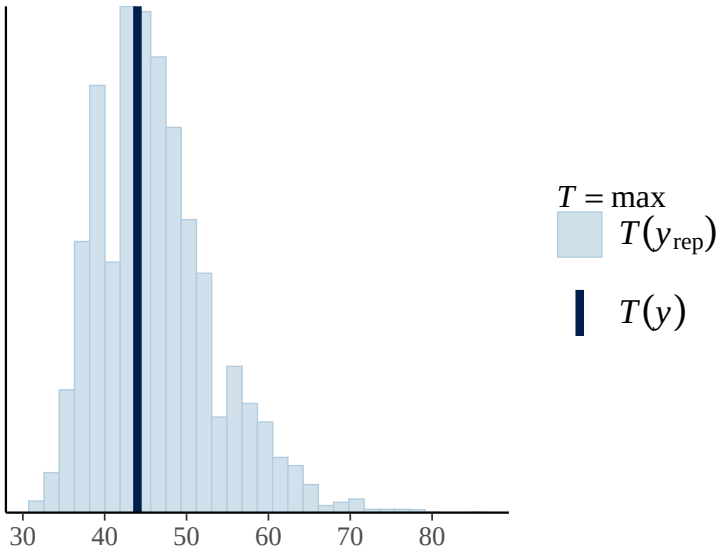

```
ppc_stat(y3_fitness, yrep4_fitness_pois,stat="max")
```




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

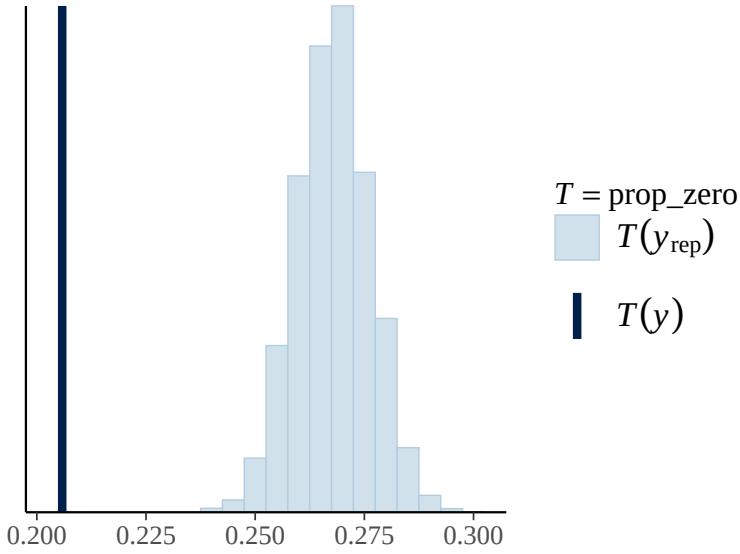



```
ppc_stat(y3_fitness, yrep4_fitness_zinb,stat="max")
```

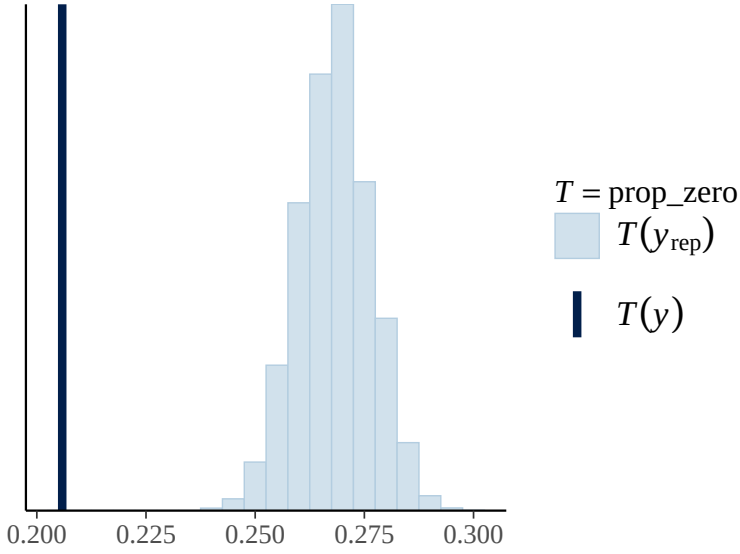


Look at the distribution of the proportion of zeros over the replicated datasets from the posterior predictive distribution in yrep and compare to the proportion of observed zeros in y.

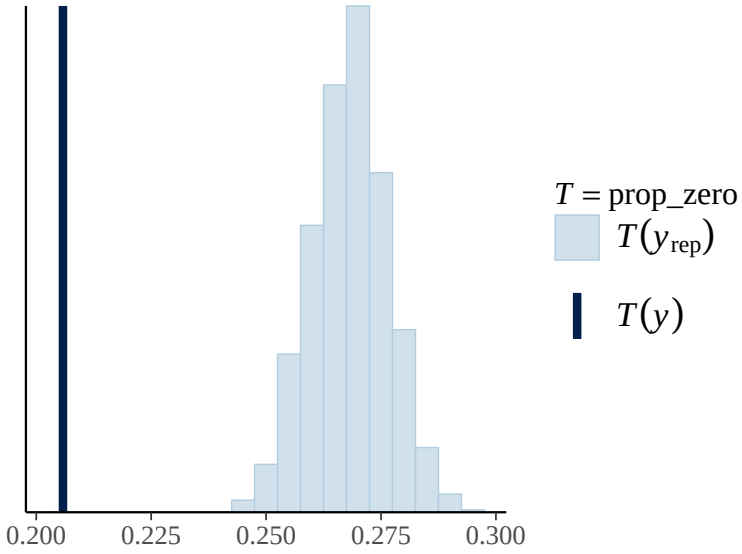
```
ppc_stat(y3_fitness, yrep4_fitness_pois, stat = "prop_zero", binwidth = 0.005)
```




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




```
ppc_stat(y3_fitness, yrep4_fitness_zinb, 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
```

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

```
##                           Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                    0.6472059 0.009183956 0.6288124 0.6644935
## R2roundmeanfitnessstudy 0.9391691 0.006549219 0.9256104 0.9490675
```

Very similar.

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 = 6000)

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
##   -1.2981427   -0.2130341
```

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

```
##                  lower       upper
## S_intercepts -1.708835 -0.94237491
## S_slopes     -0.468725  0.08862541
## 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
}
```

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

```
## beta_intercepts     beta_slopes
##      -0.7503894       0.6763191
```

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

```
##                       lower      upper
## beta_intercepts -1.9536517 -0.3719021
## beta_slopes     -0.2164703  3.4826545
## attr(,"Probability")
## [1] 0.95
```

```r
# 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 = 6000)

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
##   -1.2784093   -0.1637342
```

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

```
##                  lower      upper
## S_intercepts -1.713212 -0.9400165
## S_slopes     -0.483439  0.1052165
## 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
##      -0.7456015       0.6408855
```

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

```
##                       lower      upper
## beta_intercepts -2.1026519 -0.3155972
## beta_slopes     -0.1637603  4.1730462
## attr(,"Probability")
## [1] 0.95
```

```
# Extract posterior samples
bivar3.all.brm.zinb_post <- posterior_samples(bivar3.all.brm.zinb)
bivar3.all.brm.zinb_post <- as.mcmc(bivar3.all.brm.zinb_post)
#head(bivar3.all.brm.zinb_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.gmat11 <- sample.gmat1(bivar3.all.brm.zinb_post, replicates = 2000)
```

```
sgmat11 <- lapply(sampled.gmat11, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat11 <- unname(sapply(sgmat11, '[[', 1)) #Change to matrix

sgmat11 <- t(sgmat11)

P.modelBV_RR11 <- sgmat11
P.modelBV_RR11.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR11.mode[k] <- posterior.mode(mcmc(sgmat11[,k]))

# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR11 <- sgmat11[,c(3,6)]
colnames(S.modelBV_RR11) <- c("S_intercepts", "S_slopes")
S.modelBV_RR11.mode <- P.modelBV_RR11.mode[1:2, 3]

posterior.mode(mcmc(S.modelBV_RR11))
```

Negative binomial model with zero-inflation

```
## S_intercepts     S_slopes
##   -1.3029084   -0.1848154
```

```
HPDinterval(mcmc(S.modelBV_RR11))
```

```
##                   lower       upper
## S_intercepts -1.7050227 -0.94528357
## S_slopes     -0.4617111  0.09342776
## 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_RR11 <- matrix(NA, nrow(S.modelBV_RR11) ,2)

for (i in 1:nrow(S.modelBV_RR11)) {
  P3_11 <- 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_11[k] <- P.modelBV_RR11[i, k] }
  P2_11 <- P3_11[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S11 <- P3_11[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR11[i,] <- solve(P2_11) %*% S11   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR11) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR11))
```

```
## beta_intercepts     beta_slopes
##      -0.8623064       0.7121471
```

```
HPDinterval(mcmc(beta_post_RR11))
```

```
##                       lower      upper
## beta_intercepts -1.7935330 -0.3987866
## beta_slopes     -0.1606909  3.0218059
## 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))
```

Got warnings:

Due to these warnings, results of bivar4.all.brm.pois are probably not reliable! I have been trying to increase max_treedepth and running more iterations, but this takes extremely long to run (a couple of days with these parameters). I am not sure it makes sense to try increasing max_treedepth and iterations even more, or if we should just stay with the negative binomial distribution, which works fine and gives no warnings.

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

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)
```

```
print(bivar4.all.brm.nb,digits=3)
```

```
##  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)                                     1.651     0.148    1.365
## sd(FFD_cmean_4)                                       0.801     0.132    0.541
## sd(roundmeanfitnessstudy_Intercept)                   1.199     0.050    1.107
## cor(FFD_Intercept,FFD_cmean_4)                        0.688     0.131    0.407
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)   -0.466     0.075   -0.606
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)      0.053     0.138   -0.214
##                                                    u-95% CI  Rhat Bulk_ESS
## sd(FFD_Intercept)                                     1.936 1.000     3171
## sd(FFD_cmean_4)                                       1.062 1.002     2065
## sd(roundmeanfitnessstudy_Intercept)                   1.302 1.001     3097
## cor(FFD_Intercept,FFD_cmean_4)                        0.905 1.005     1008
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)   -0.311 1.005     1350
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)      0.323 1.004     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)    5.223     0.860    3.840    7.158 1.000     3348     4518
##
## Population-Level Effects:
##                                              Estimate Est.Error l-95% CI
## FFD_Intercept                                  58.630     1.106   56.428
## roundmeanfitnessstudy_Intercept                 0.132     0.057    0.016
## FFD_cmean_4                                    -2.423     0.832   -4.054
## roundmeanfitnessstudy_cn_shoot_vol_mean_sqrt    0.054     0.004    0.046
##                                              u-95% CI  Rhat Bulk_ESS Tail_ESS
## FFD_Intercept                                  60.835 1.001     2350     3538
## roundmeanfitnessstudy_Intercept                 0.242 1.000     4767     5088
## FFD_cmean_4                                    -0.797 1.000     2863     4181
## roundmeanfitnessstudy_cn_shoot_vol_mean_sqrt    0.063 1.000     4886     5162
##
## Family Specific Parameters:
##                             Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS
## sigma_FFD                      4.356     0.074    4.209    4.502 1.001     3138
## shape_roundmeanfitnessstudy  551.333   168.601  288.203  944.637 1.001     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).
```

Negative binomial distribution with zero-inflation

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

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

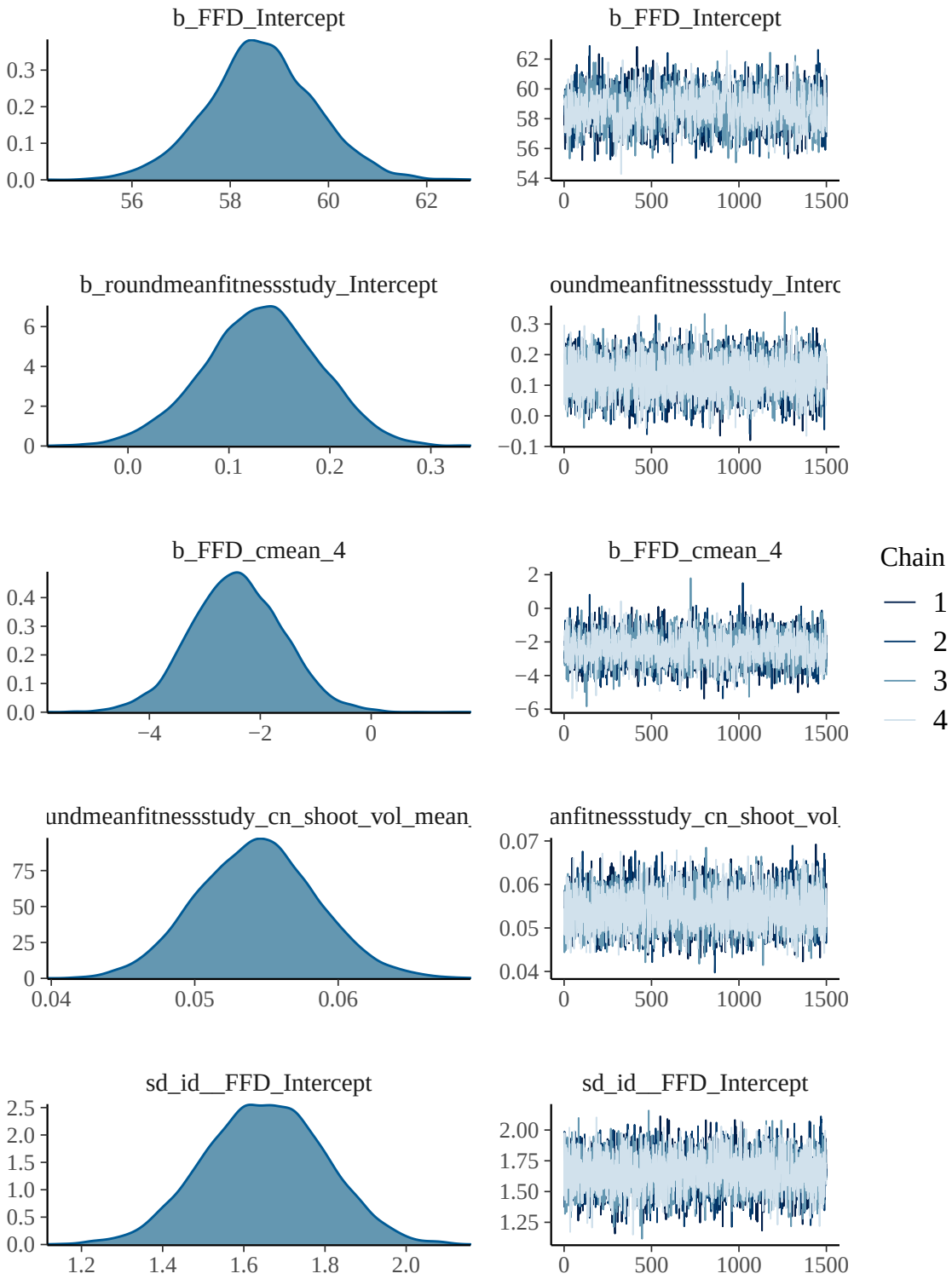
```
##  Family: MV(gaussian, zero_inflated_negbinomial)
##   Links: mu = identity; sigma = identity
##          mu = log; shape = identity; zi = identity
## Formula: FFD ~ cmean_4 + (cmean_4 | ID1 | id) + (1 | year)
##          round(mean_fitness_study) ~ cn_shoot_vol_mean_sqrt + (1 | ID1 | id)
##    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     1.35
## 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)                       0.68      0.13     0.40
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)  -0.47      0.08    -0.61
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)     0.05      0.14    -0.23
##                                                  u-95% CI Rhat Bulk_ESS
## sd(FFD_Intercept)                                    1.94 1.00     3075
## sd(FFD_cmean_4)                                      1.07 1.01      959
## sd(roundmeanfitnessstudy_Intercept)                  1.30 1.00     2922
## cor(FFD_Intercept,FFD_cmean_4)                       0.90 1.02      209
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)  -0.32 1.00     1115
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)     0.32 1.01      362
##                                                  Tail_ESS
## sd(FFD_Intercept)                                    4380
## sd(FFD_cmean_4)                                      3100
## sd(roundmeanfitnessstudy_Intercept)                  4812
## cor(FFD_Intercept,FFD_cmean_4)                       1956
## cor(FFD_Intercept,roundmeanfitnessstudy_Intercept)    2402
## cor(FFD_cmean_4,roundmeanfitnessstudy_Intercept)      1127
##
## ~year (Number of levels: 22)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(FFD_Intercept)     5.27      0.87     3.88     7.23 1.00     2628     3974
##
## Population-Level Effects:
##                                            Estimate Est.Error l-95% CI
## FFD_Intercept                                 58.64      1.12    56.42
## roundmeanfitnessstudy_Intercept                0.13      0.06     0.02
## FFD_cmean_4                                   -2.40      0.84    -4.07
## roundmeanfitnessstudy_cn_shoot_vol_mean_sqrt   0.05      0.00     0.05
```

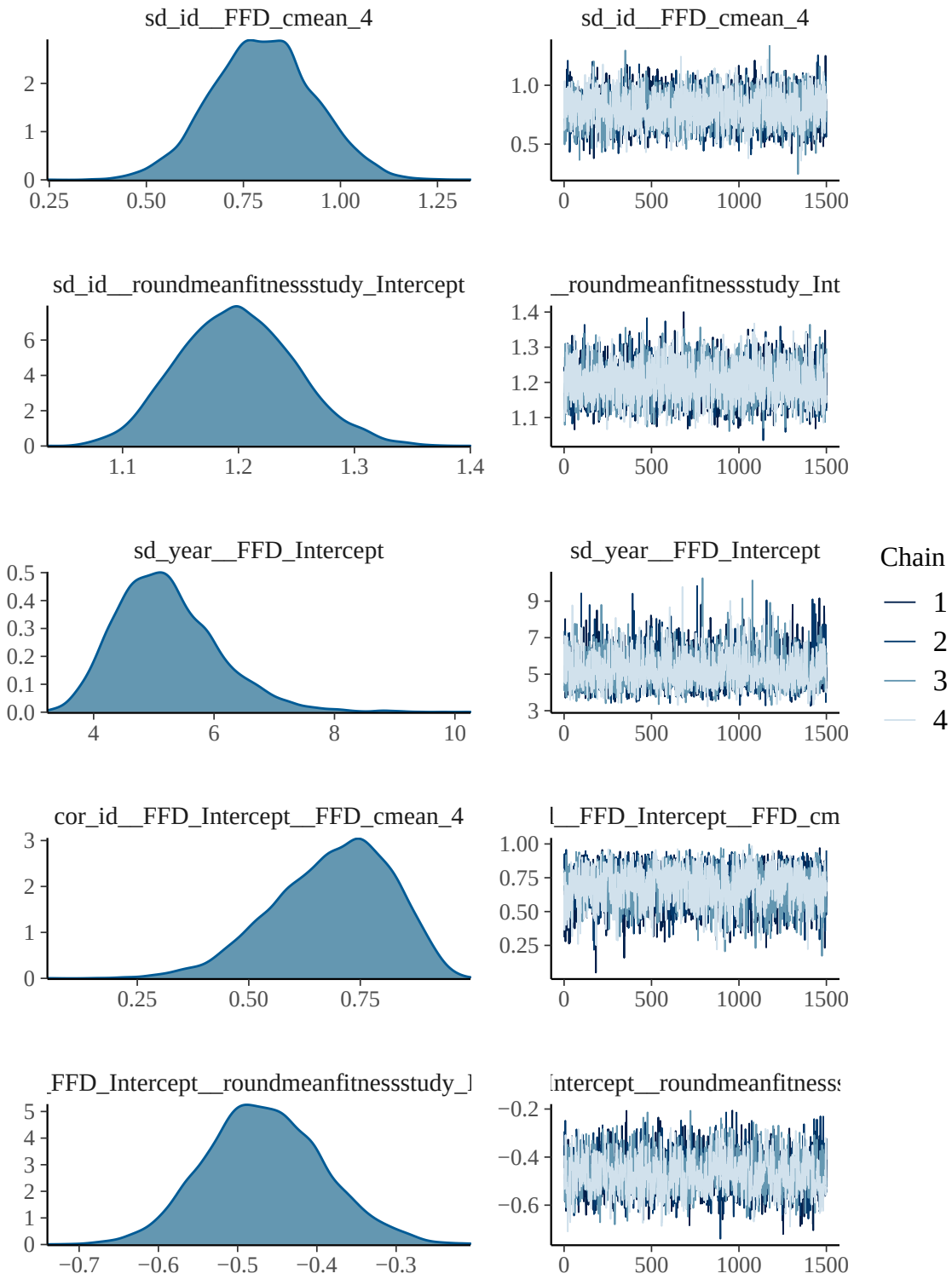
```
##                                             u-95% CI Rhat Bulk_ESS Tail_ESS
## FFD_Intercept                                  60.88 1.00     1711     2576
## roundmeanfitnessstudy_Intercept                 0.24 1.00     4502     5478
## FFD_cmean_4                                    -0.74 1.00     2331     3661
## roundmeanfitnessstudy_cn_shoot_vol_mean_sqrt    0.06 1.00     4697     4836
##
## Family Specific Parameters:
##                             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sigma_FFD                       4.35      0.07     4.21     4.50 1.00     1543
## shape_roundmeanfitnessstudy   556.96    169.77   296.31   945.63 1.00     5127
## zi_roundmeanfitnessstudy        0.00      0.00     0.00     0.00 1.00     4725
##                             Tail_ESS
## sigma_FFD                       4927
## shape_roundmeanfitnessstudy     4948
## zi_roundmeanfitnessstudy        4154
##
## 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).
```

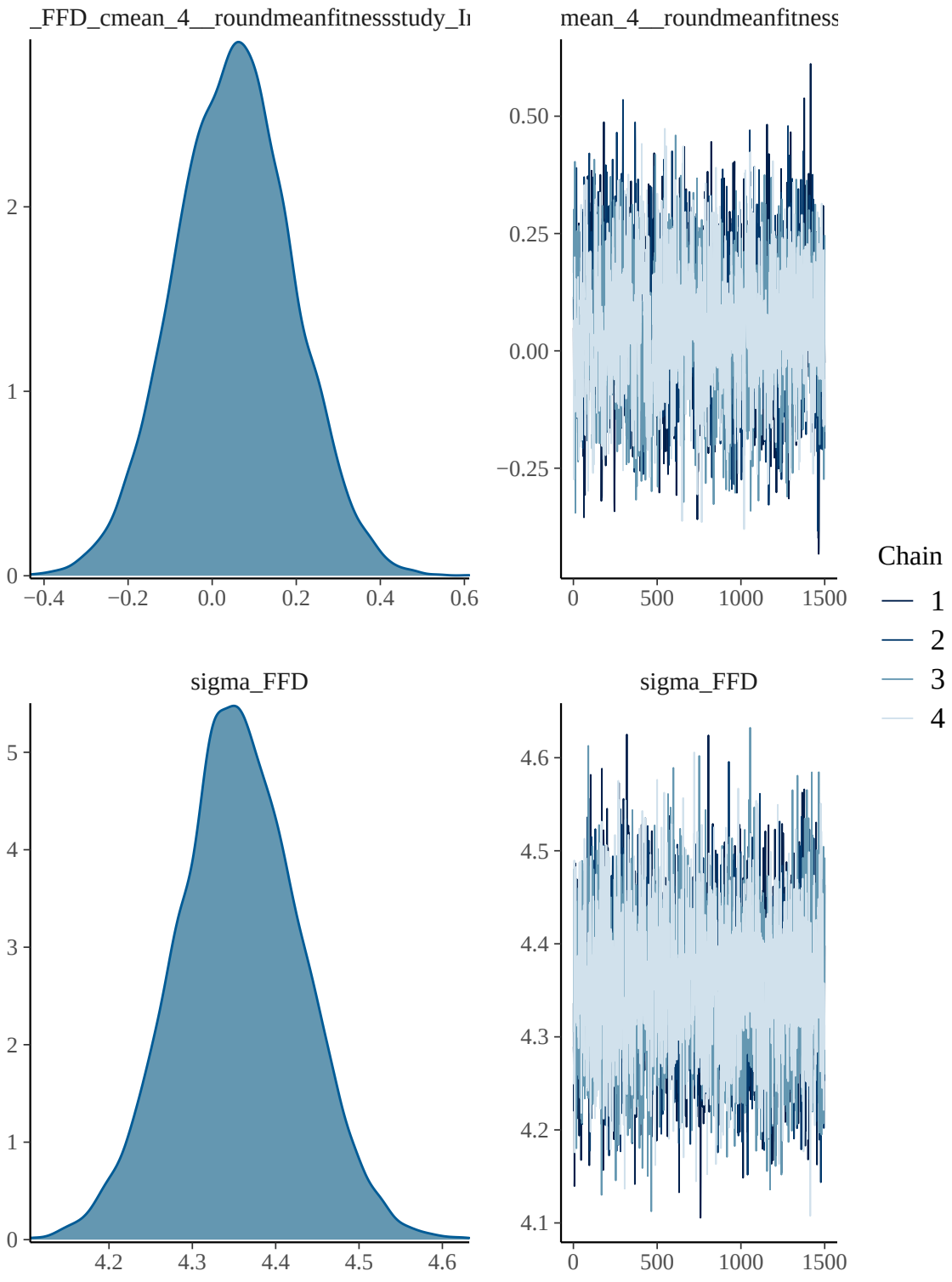
Model evaluation: Compare models

Looking only at negative binomial model by now, as Poisson model gave warnings.

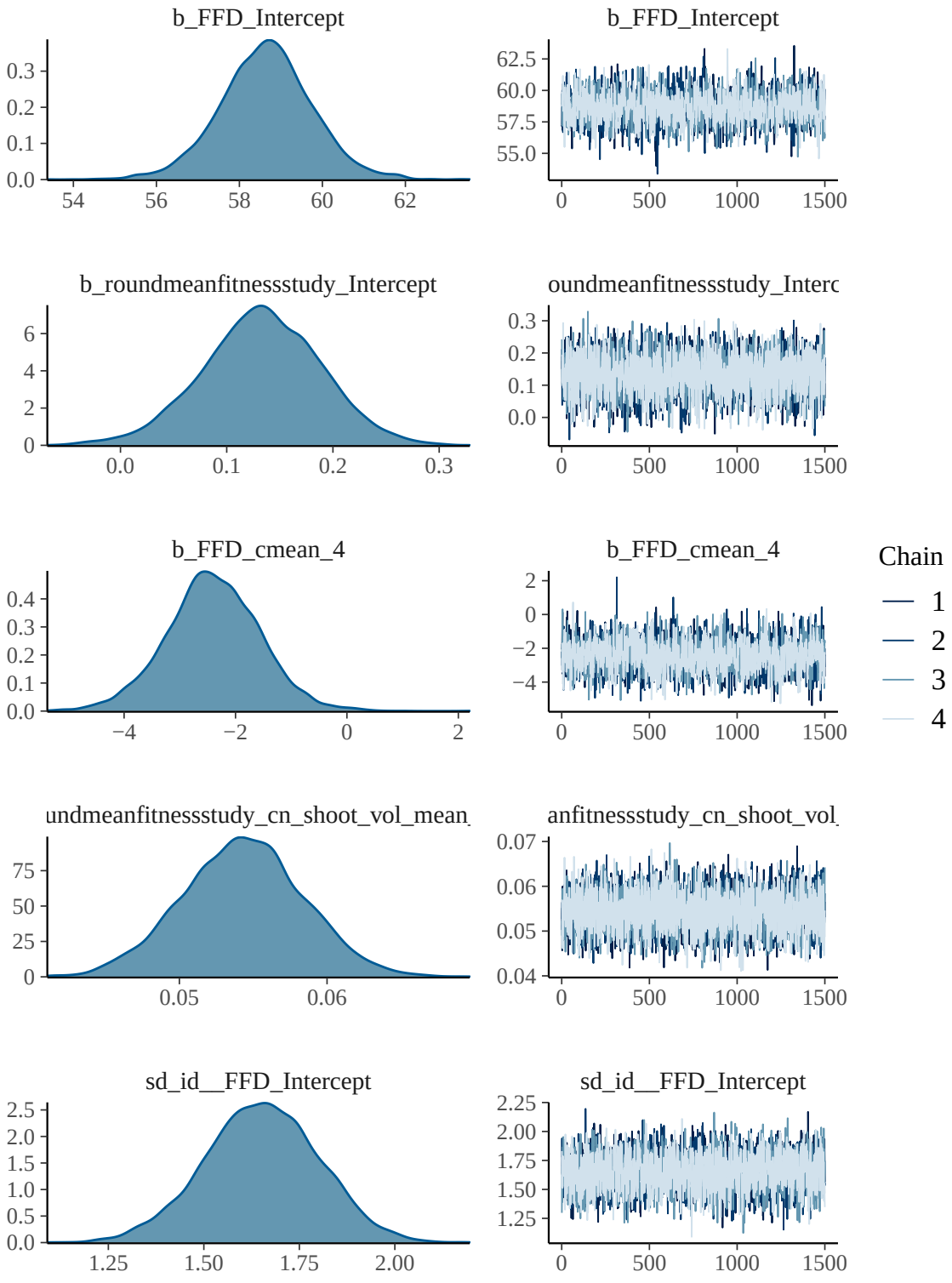
```
#plot(bivar4.all.brm.pois)
plot(bivar4.all.brm.nb)
```

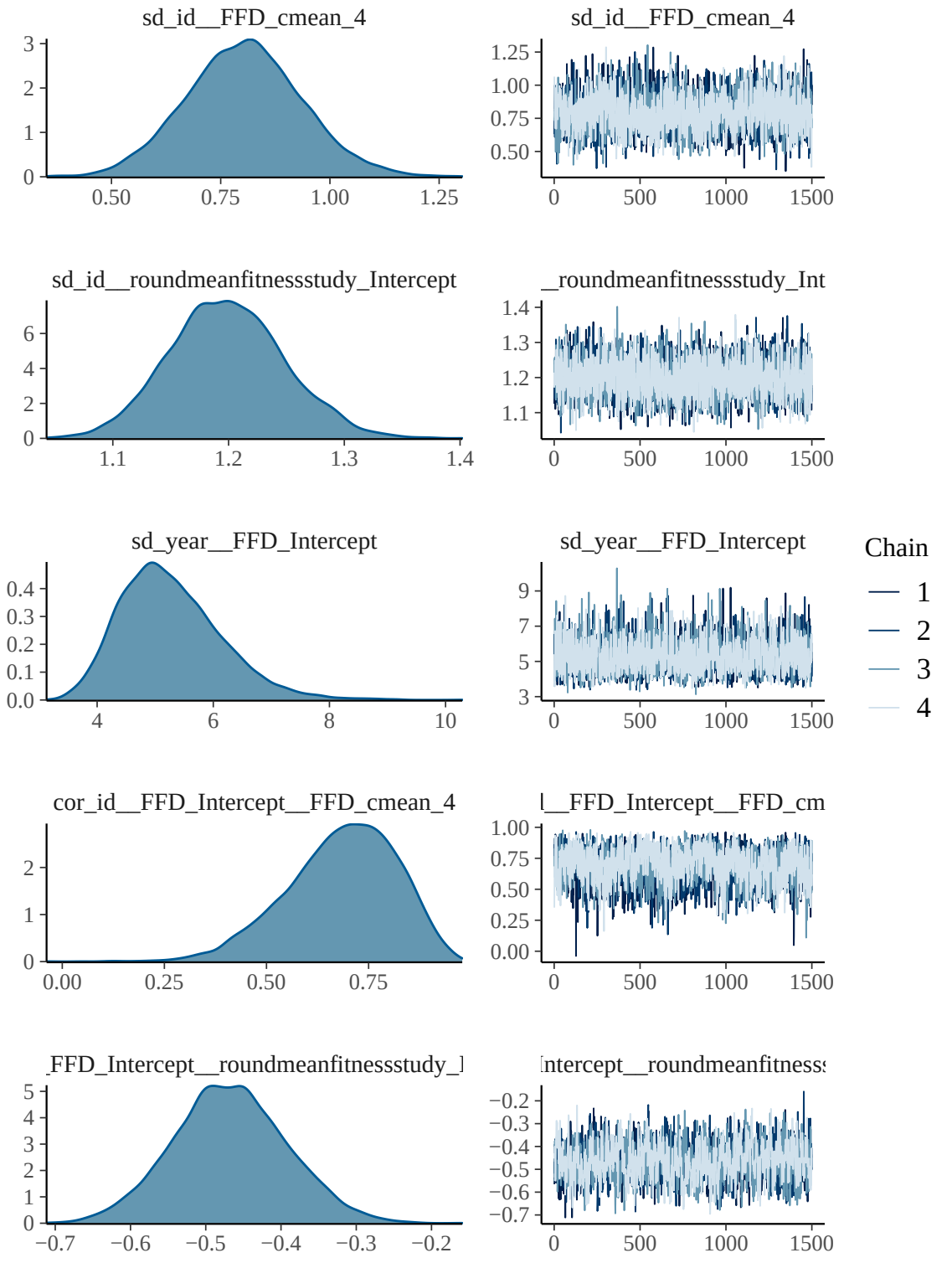

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessstudy_Intercept
oundmeanfitnessstudy_Interc
b_FFD_cmean_4
b_FFD_cmean_4
undmeanfitnessstudy_cn_shoot_vol_mean
anfitnessstudy_cn_shoot_vol
sd_id__FFD_Intercept
sd_id__FFD_Intercept
Chain
1
2
3
4

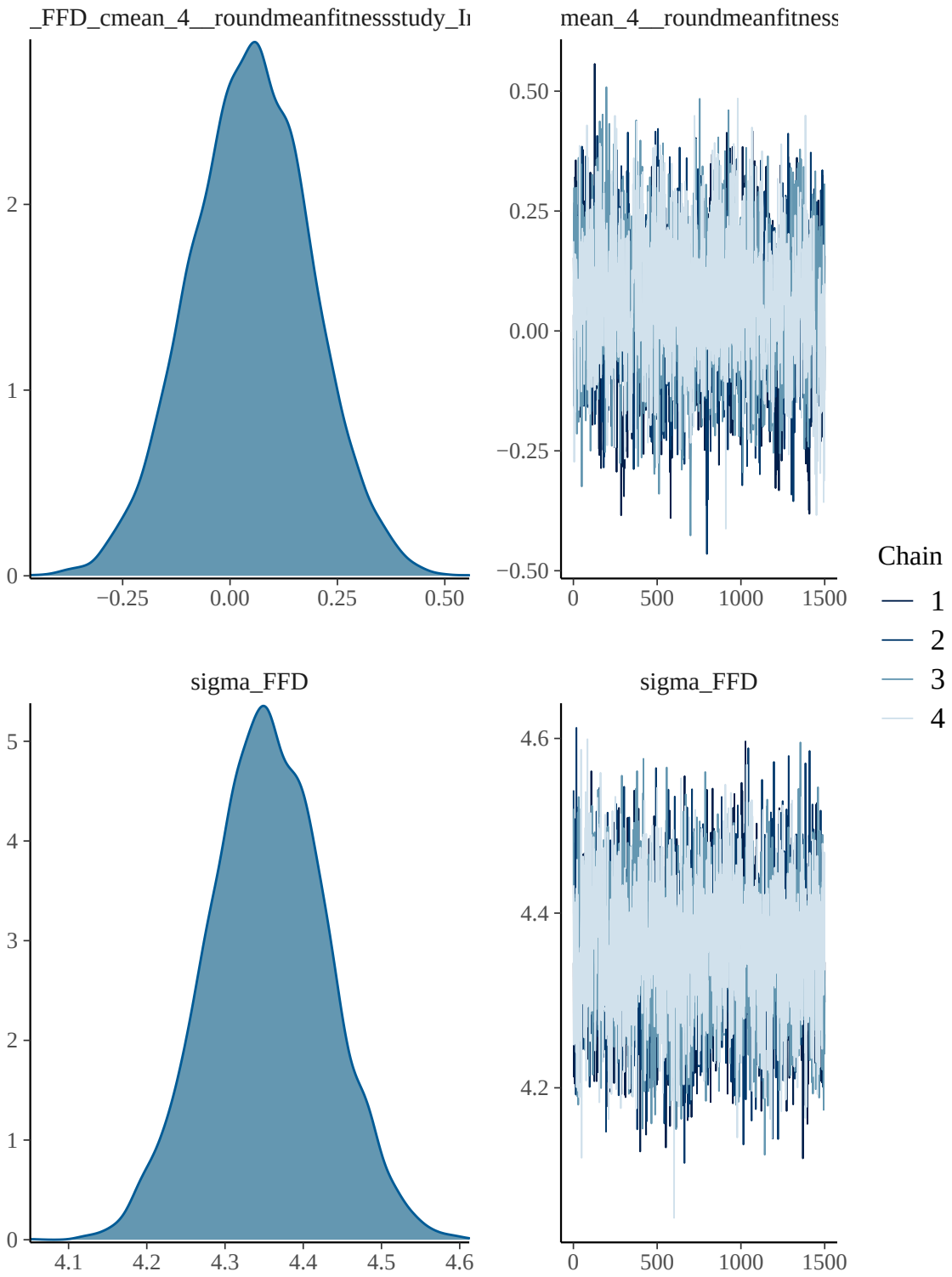

sd_id__FFD_cmean_4
sd_id__FFD_cmean_4
sd_id__roundmeanfitnessstudy_Intercept
__roundmeanfitnessstudy_Int
sd_year__FFD_Intercept
sd_year__FFD_Intercept
Chain
1
2
3
4
cor_id__FFD_Intercept__FFD_cmean_4
__FFD_Intercept__FFD_cm
_FFD_Intercept__roundmeanfitnessstudy_
ntercept__roundmeanfitness




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

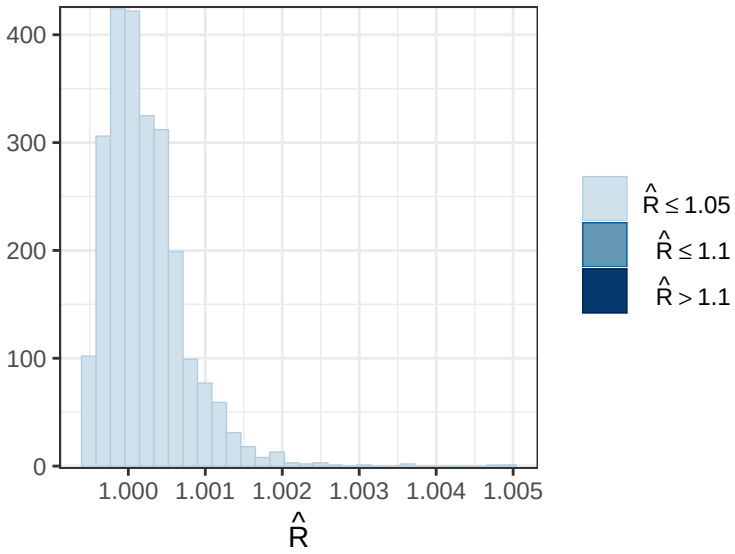

b_FFD_Intercept
b_FFD_Intercept
b_roundmeanfitnessstudy_Intercept
oundmeanfitnessstudy_Interc
b_FFD_cmean_4
b_FFD_cmean_4
undmeanfitnessstudy_cn_shoot_vol_mean
anfitnessstudy_cn_shoot_vol
sd_id__FFD_Intercept
sd_id__FFD_Intercept
Chain
1
2
3
4


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

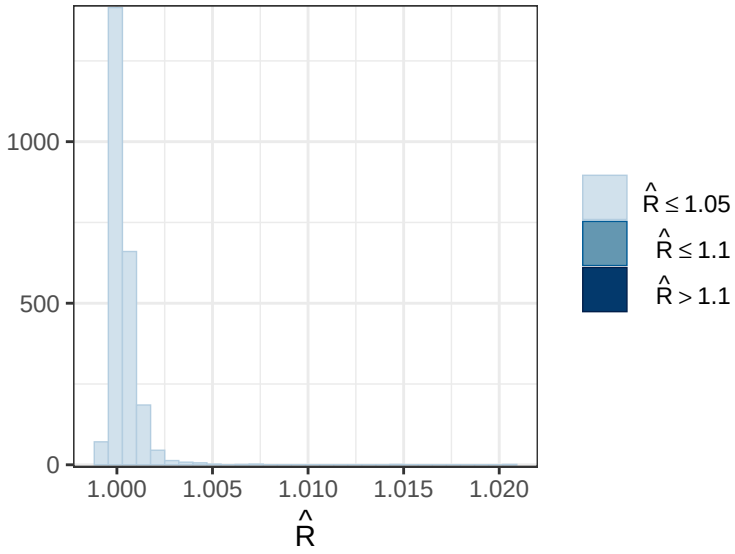


Rhat

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

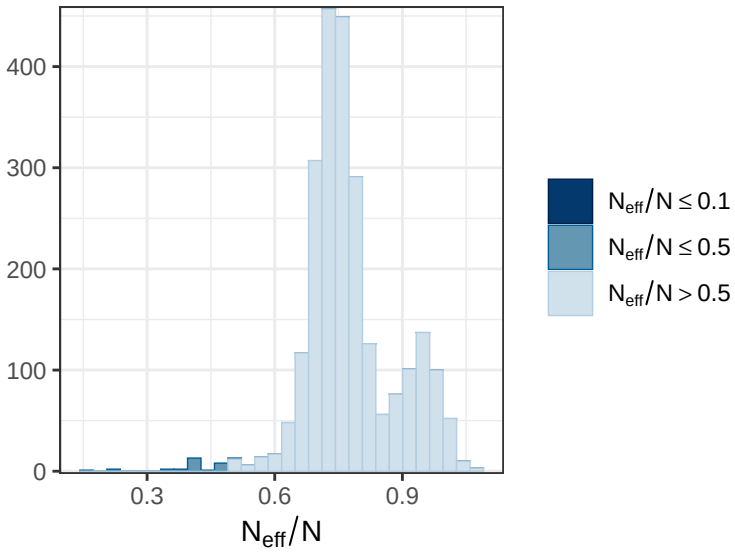



```
mcmc_rhat_hist(rhat(bivar4.all.brm.zinb))+theme_bw()
```

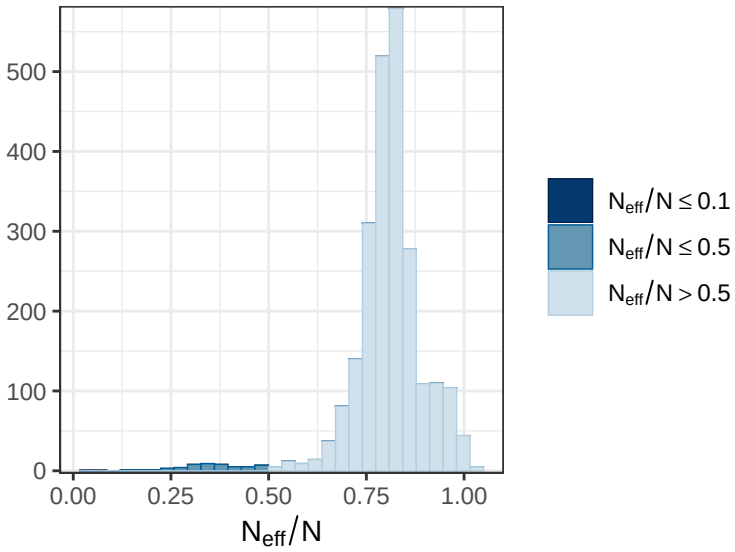


Effective sample size:

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




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



Posterior predictive checks:

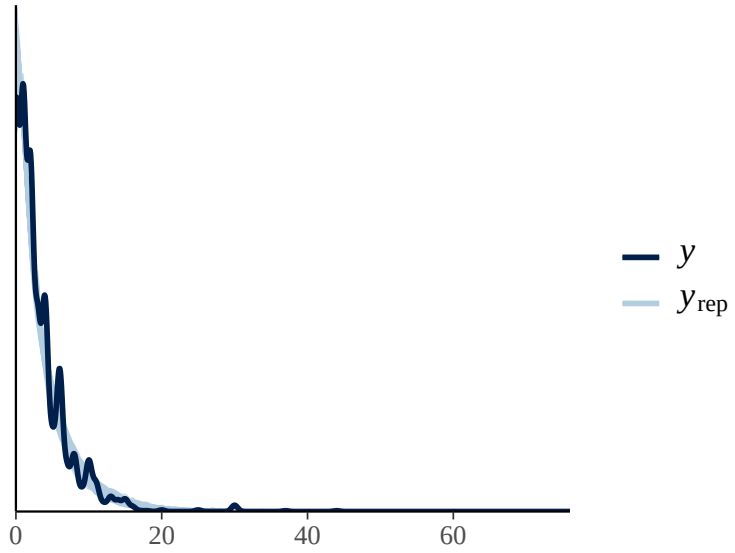
```
y3_FFD<-subset(datadef,!is.na(FFD)&
                            !is.na(cn_shoot_vol_mean_sqrt))$FFD
y4_fitness<-round(subset(datadef,!is.na(FFD)&
                            !is.na(cn_shoot_vol_mean_sqrt))$mean_fitness_study)
#yrep5_fitness_pois<-posterior_predict(bivar4.all.brm.pois,
#                                      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")
```

```
yrep5_fitness_zinb<-posterior_predict(bivar4.all.brm.zinb,
                                      draws = 500,resp="roundmeanfitnessstudy")
yrep5_FFD_zinb<-posterior_predict(bivar4.all.brm.zinb,
                                  draws = 500,resp="FFD")
# matrices of draws from the posterior predictive distribution
```

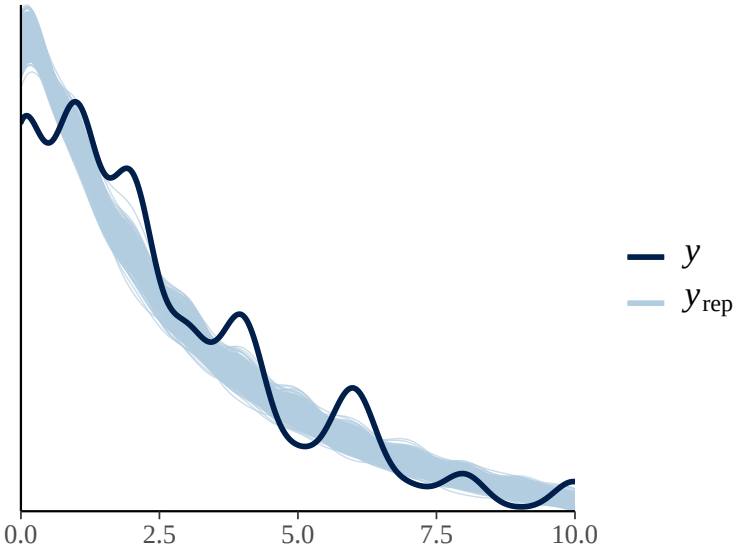
Each row of the matrix is a draw from the posterior predictive distribution, i.e. a vector with one element for each of the data points in y.

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

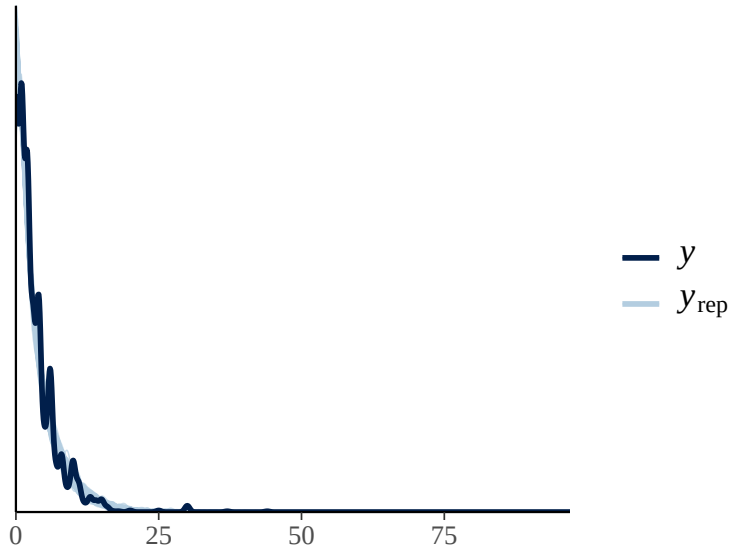
```
#ppc_dens_overlay(y4_fitness, yrep5_fitness_pois[1:500,])
#ppc_dens_overlay(y4_fitness, yrep5_fitness_pois[1:500,])+xlim(0, 10)
ppc_dens_overlay(y4_fitness, yrep5_fitness_nb[1:500,])
```



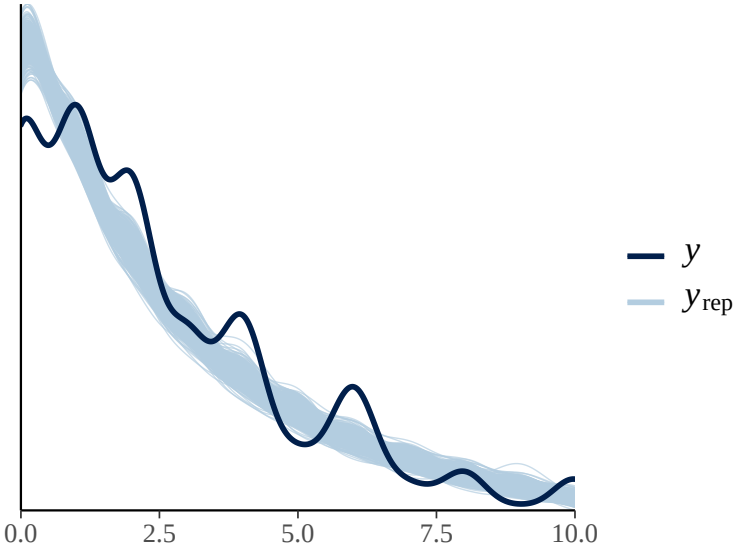

```
ppc_dens_overlay(y4_fitness, yrep5_fitness_nb[1:500,])+xlim(0,10)
```



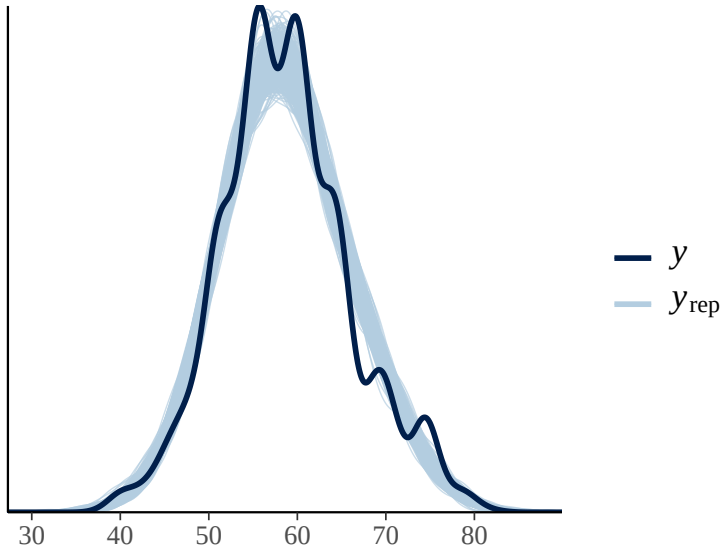
```
ppc_dens_overlay(y4_fitness, yrep5_fitness_zinb[1:500,])
```



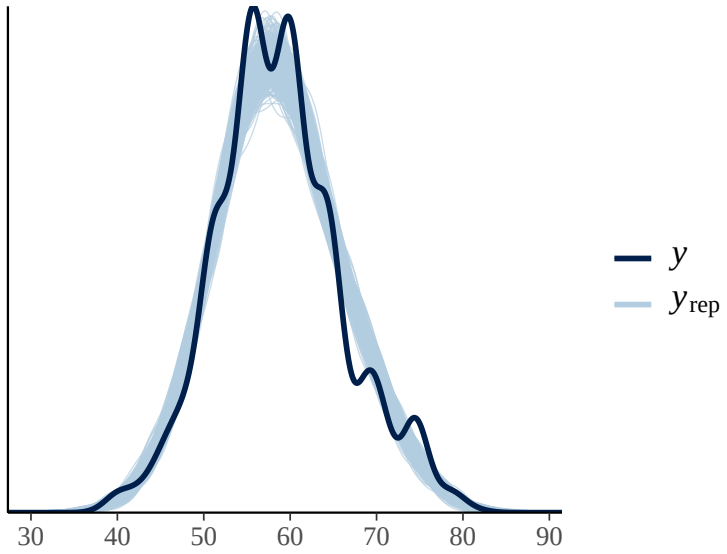

```
ppc_dens_overlay(y4_fitness, yrep5_fitness_zinb[1:500,])+xlim(0,10)
```




```
#ppc_dens_overlay(y3_FFD, yrep5_FFD_pois[1:500,])
ppc_dens_overlay(y3_FFD, yrep5_FFD_nb[1:500,])
```

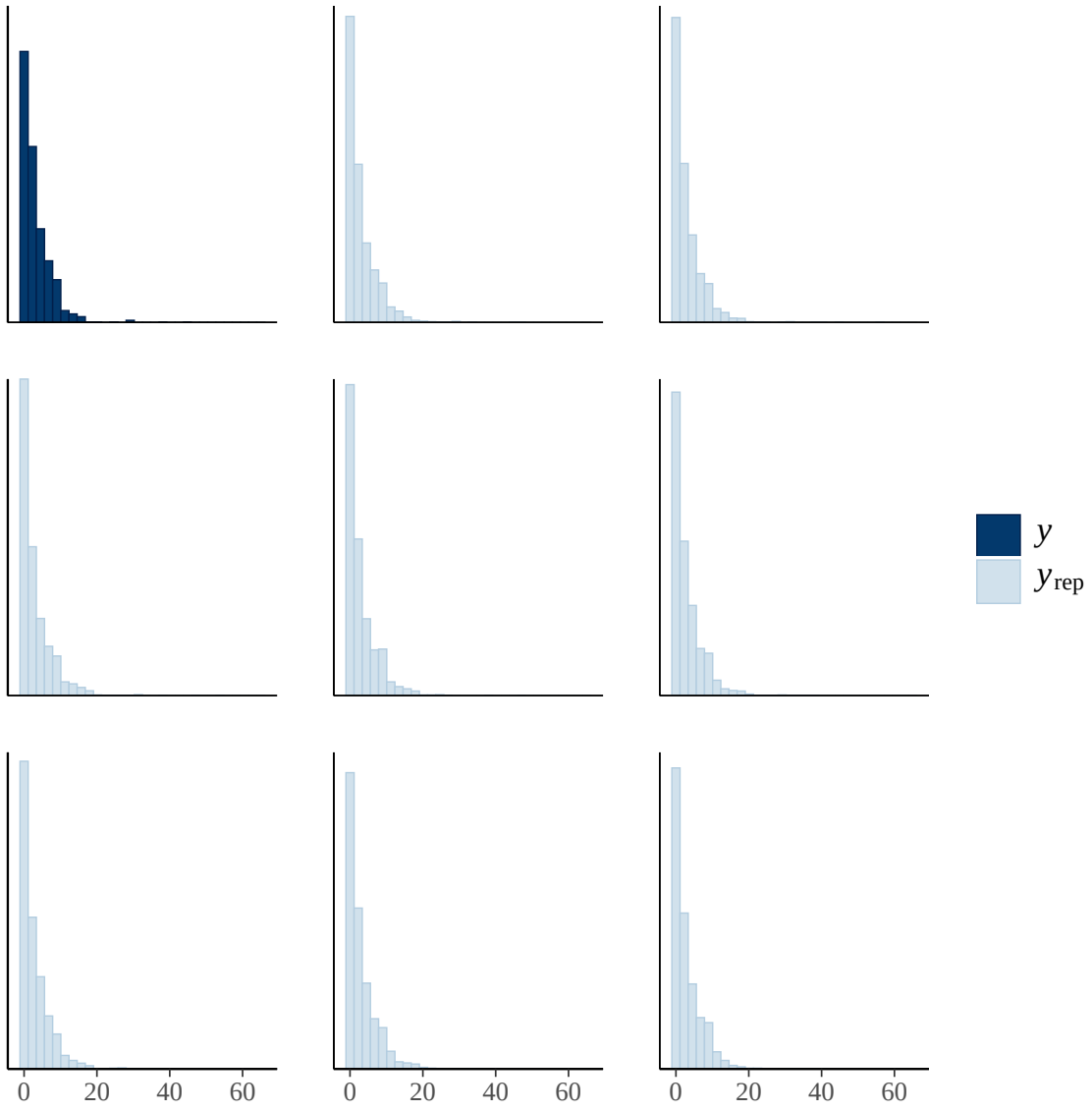



```
ppc_dens_overlay(y3_FFD, yrep5_FFD_zinb[1:500,])
```

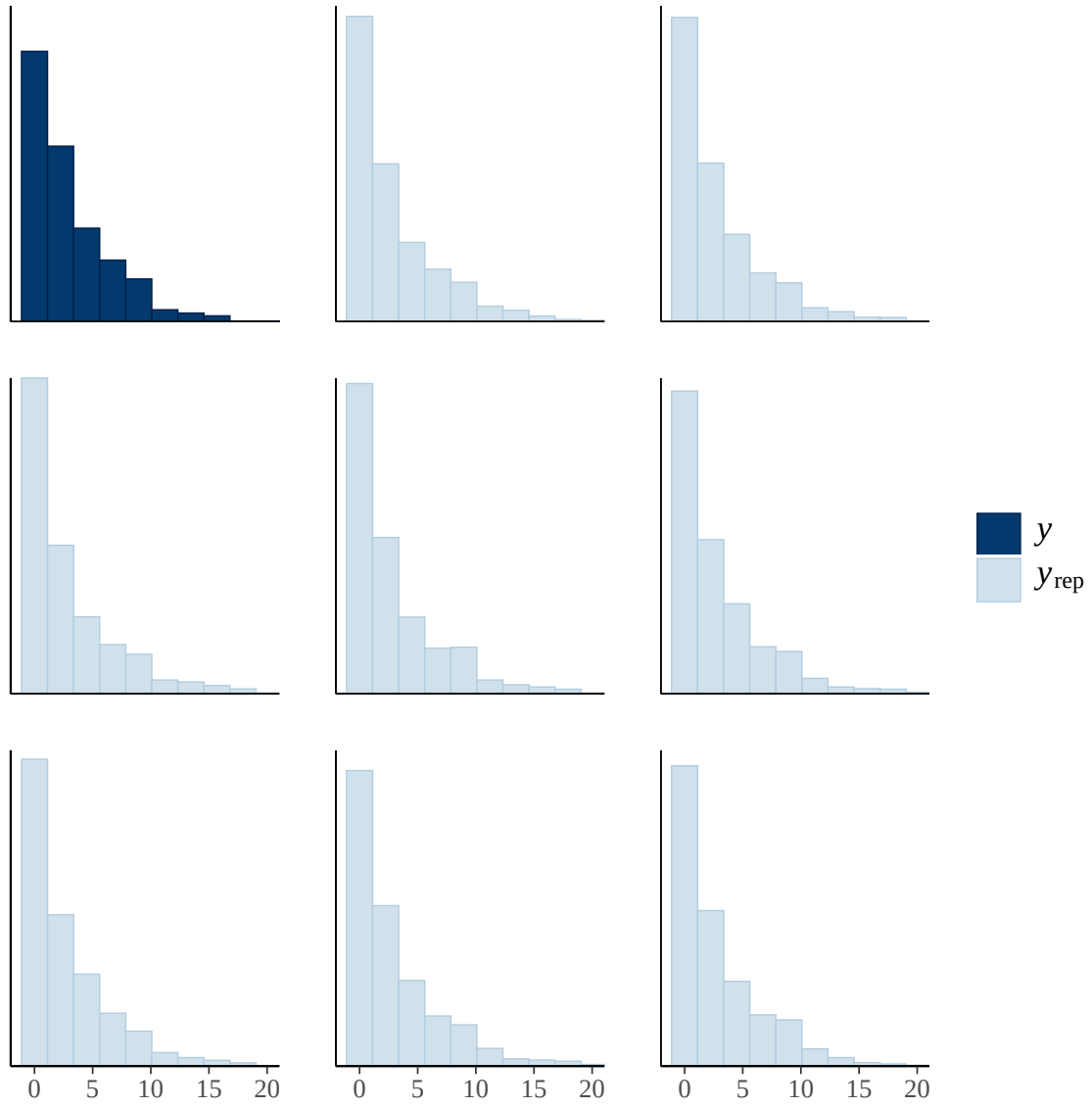


Separate histograms of y and some of the yrep datasets

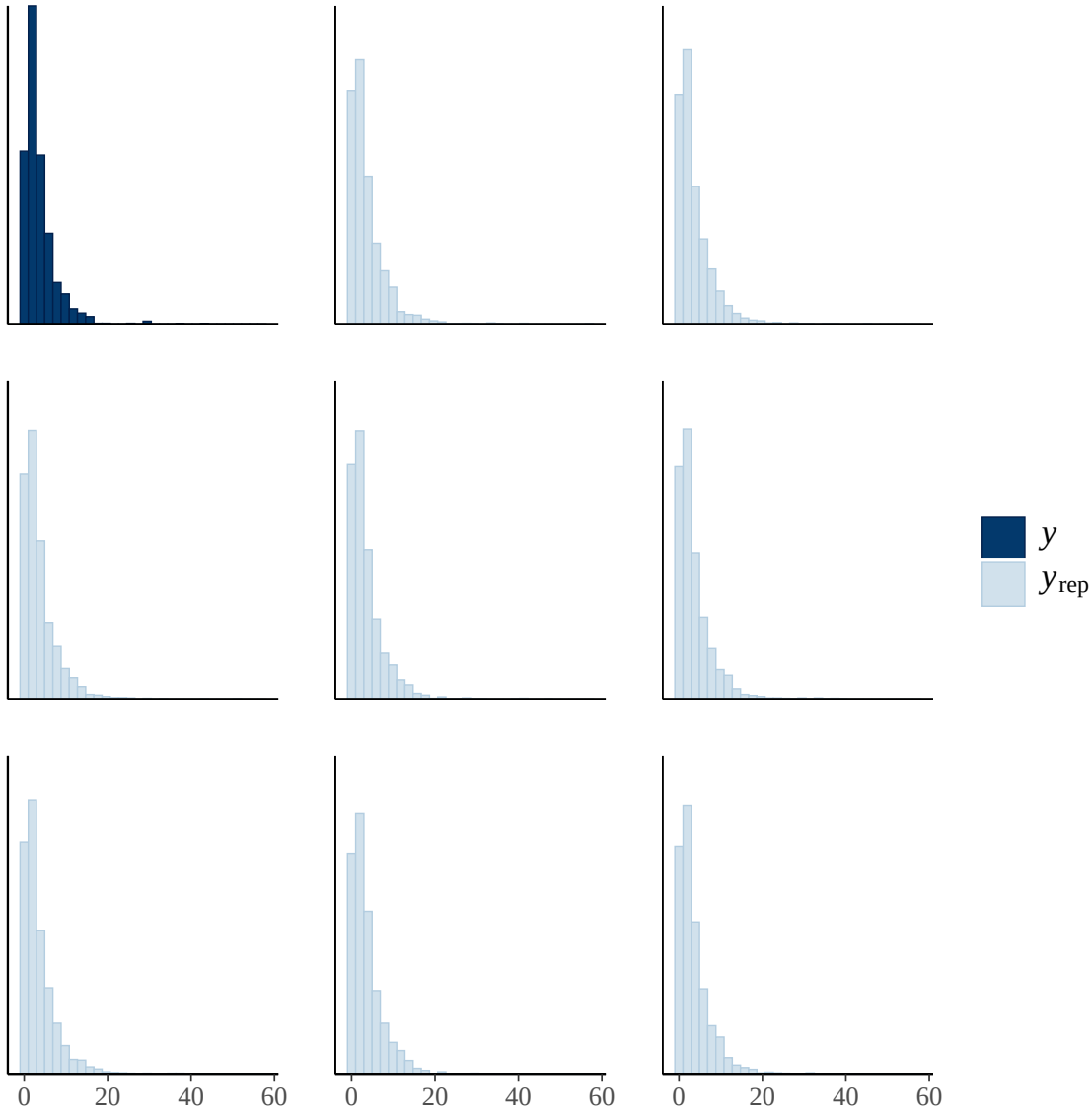
```
#ppc_hist(y4_fitness, yrep5_fitness_pois[1:8, ])
#ppc_hist(y4_fitness, yrep5_fitness_pois[1:8, ])+
#  coord_cartesian(xlim = c(-1, 20))
ppc_hist(y4_fitness, yrep5_fitness_nb[1:8, ])
```



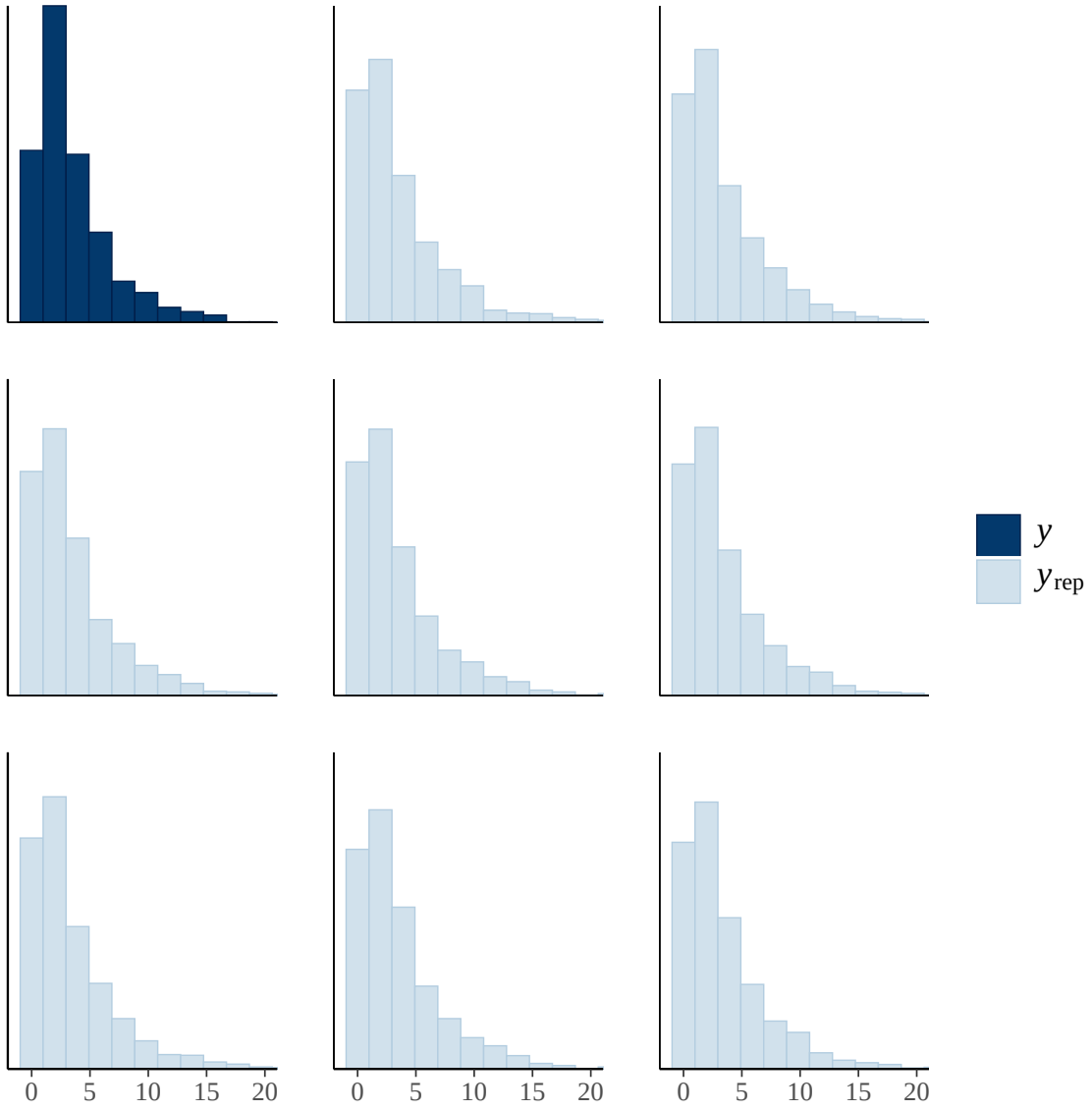

```
ppc_hist(y4_fitness, yrep5_fitness_nb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```



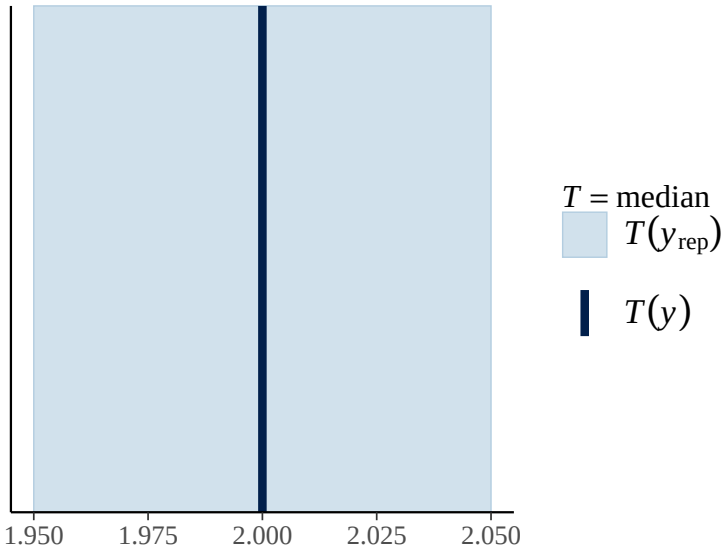

```
ppc_hist(y4_fitness, yrep5_fitness_zinb[1:8, ])
```



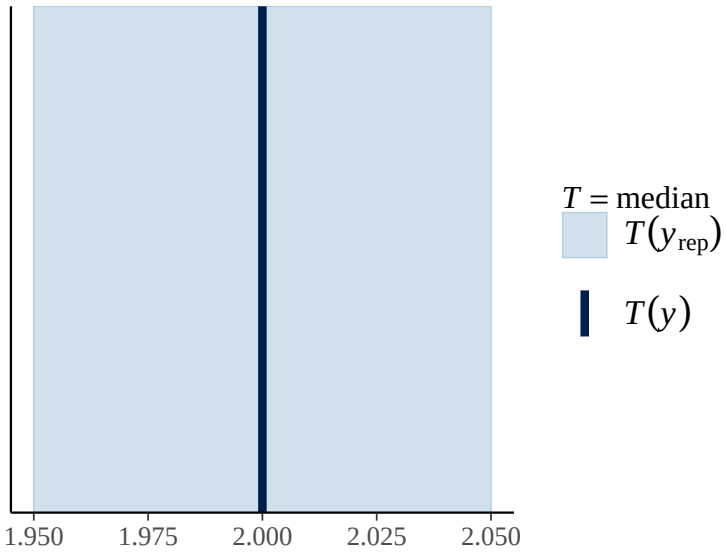

```
ppc_hist(y4_fitness, yrep5_fitness_zinb[1:8, ])+
  coord_cartesian(xlim = c(-1, 20))
```



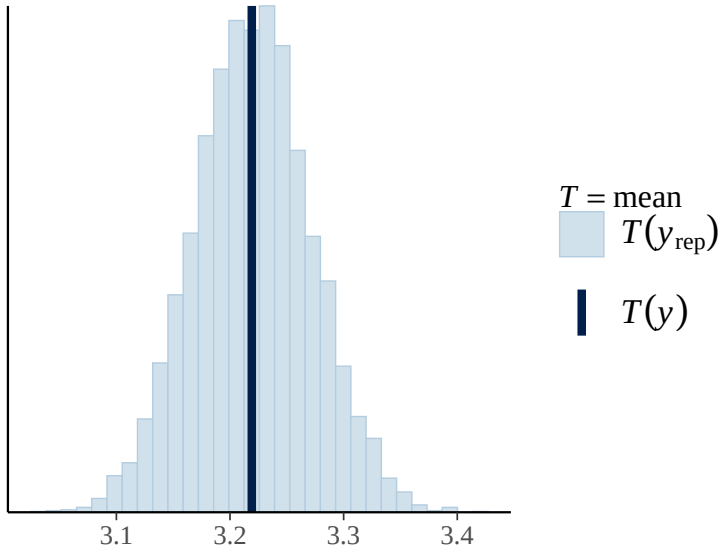

```
#ppc_stat(y4_fitness, yrep5_fitness_pois,stat="median")
ppc_stat(y4_fitness, yrep5_fitness_nb,stat="median")
```



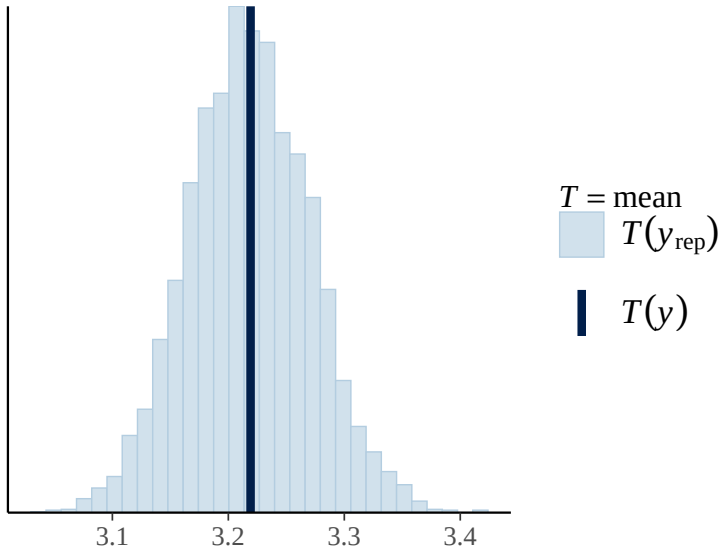

```
ppc_stat(y4_fitness, yrep5_fitness_zinb,stat="median")
```



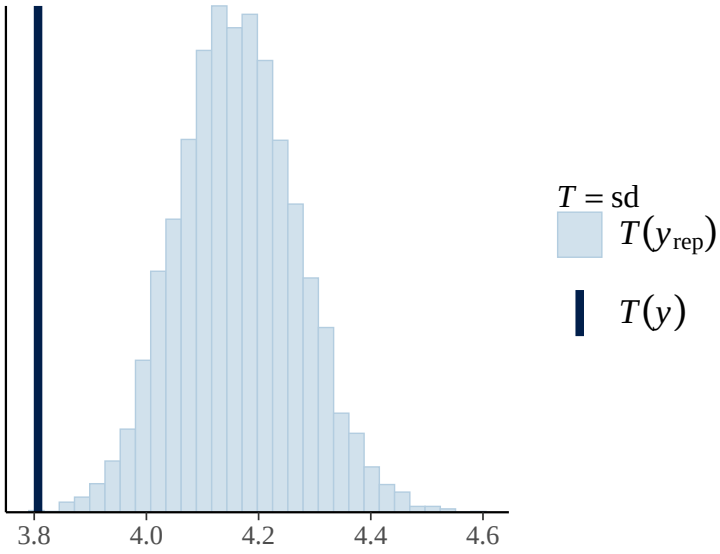

```
#ppc_stat(y4_fitness, yrep5_fitness_pois,stat="mean")
ppc_stat(y4_fitness, yrep5_fitness_nb,stat="mean")
```



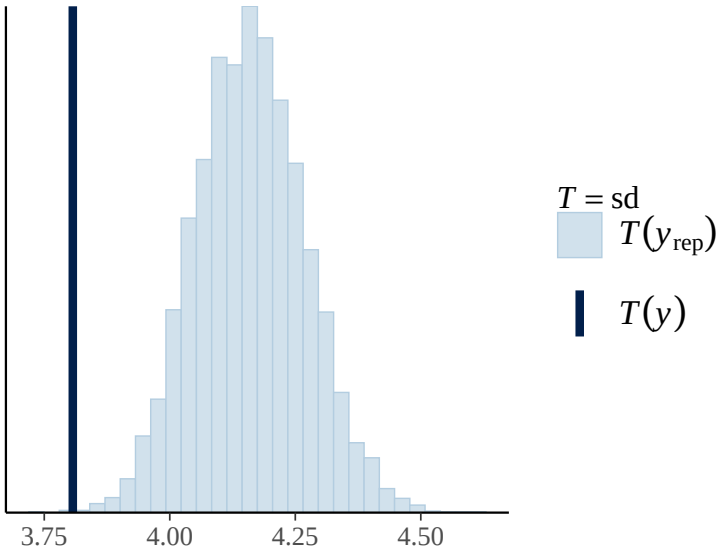

```
ppc_stat(y4_fitness, yrep5_fitness_zinb,stat="mean")
```



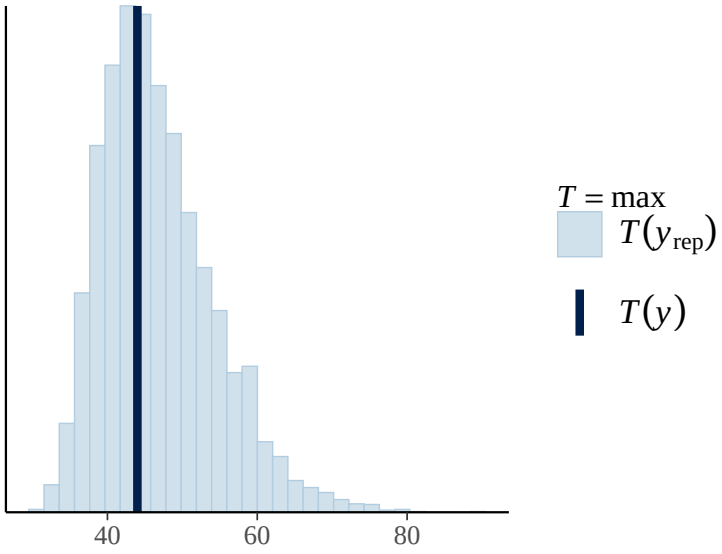

```
#ppc_stat(y4_fitness, yrep5_fitness_pois,stat="sd")
ppc_stat(y4_fitness, yrep5_fitness_nb,stat="sd")
```



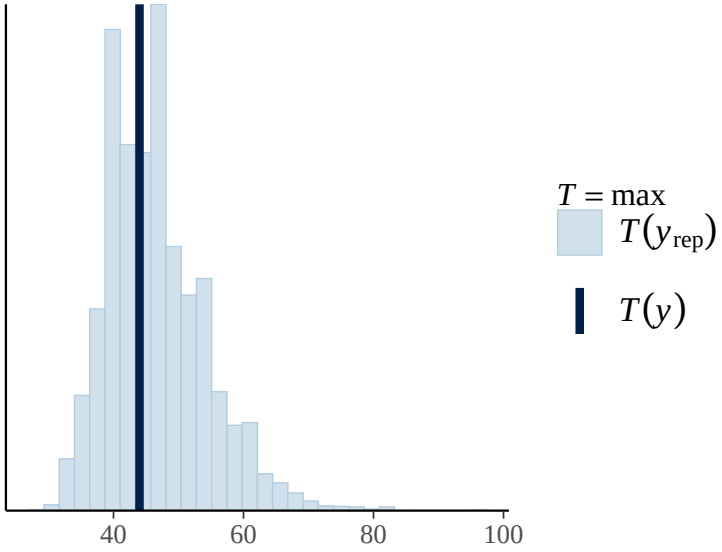

```
ppc_stat(y4_fitness, yrep5_fitness_zinb,stat="sd")
```




```
#ppc_stat(y4_fitness, yrep5_fitness_pois,stat="max")
ppc_stat(y4_fitness, yrep5_fitness_nb,stat="max")
```

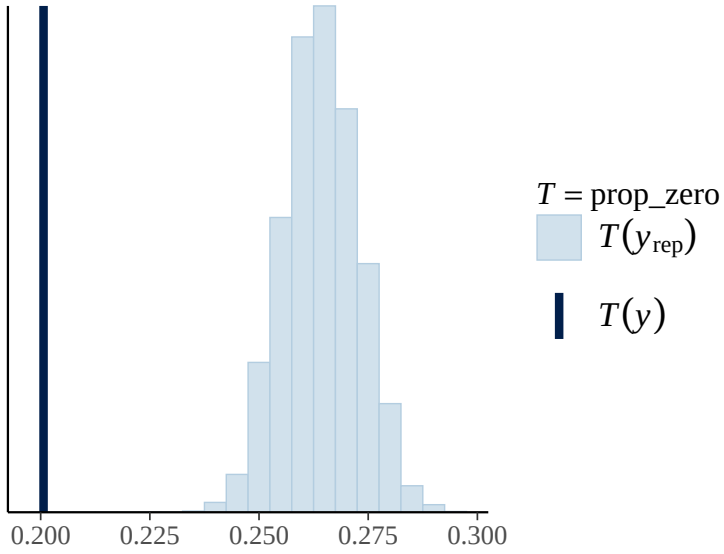



```
ppc_stat(y4_fitness, yrep5_fitness_zinb,stat="max")
```

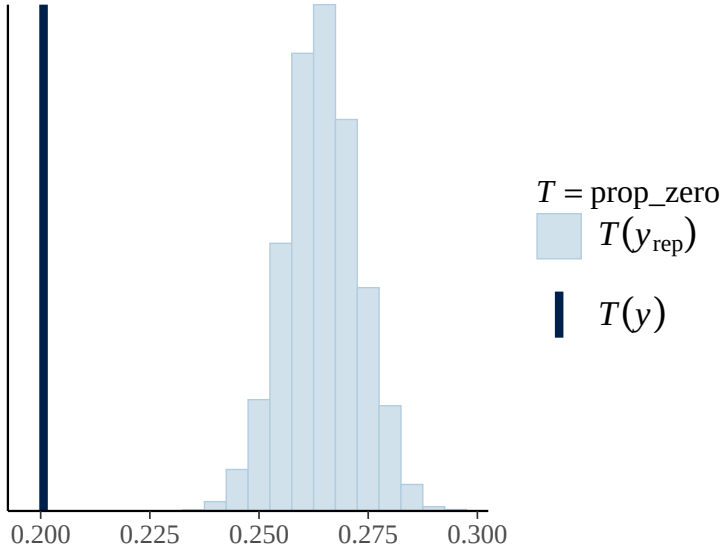


Look at the distribution of the proportion of zeros over the replicated datasets from the posterior predictive distribution in yrep and compare to the proportion of observed zeros in y.

```
#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)
```




```
ppc_stat(y4_fitness, yrep5_fitness_zinb, stat = "prop_zero", binwidth = 0.005)
```



Measure of fit: Bayesian R2

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

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

```
##                           Estimate   Est.Error      Q2.5     Q97.5
## R2FFD                    0.6481119 0.009535382 0.6291737 0.6666585
## R2roundmeanfitnessstudy 0.9406760 0.005673043 0.9287982 0.9504883
```

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 = 6000)
#
# 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))
# HPDinterval(mcmc(S.modelBV_RR7))
#
# # 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))
# HPDinterval(mcmc(beta_post_RR7))
```

Poisson model (not run due to warnings)

```
# 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 = 6000)

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
##  -0.92640366   0.09358836
```

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

```
##                   lower      upper
## S_intercepts -1.2582552 -0.5901391
## S_slopes     -0.2000891  0.3224275
## 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
##      -0.6544561       0.7454171
```

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

```
##                       lower      upper
## beta_intercepts -1.3765965 -0.3294188
## beta_slopes      0.1781614  2.6749181
## attr(,"Probability")
## [1] 0.95
```

```
# Extract posterior samples
bivar4.all.brm.zinb_post <- posterior_samples(bivar4.all.brm.zinb)
bivar4.all.brm.zinb_post <- as.mcmc(bivar4.all.brm.zinb_post)
#head(bivar4.all.brm.zinb_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.gmat12 <- sample.gmat2(bivar4.all.brm.zinb_post, replicates = 6000)

sgmat12 <- lapply(sampled.gmat12, `[`, c('gmat')) #Get list 'gmat' from each list
sgmat12 <- unname(sapply(sgmat12, '[[', 1)) #Change to matrix

sgmat12 <- t(sgmat12)

P.modelBV_RR12 <- sgmat12
P.modelBV_RR12.mode <- matrix(1:9, nrow = 3)
for (k in 1:9) P.modelBV_RR12.mode[k] <- posterior.mode(mcmc(sgmat12[,k]))

# Extract selection *differentials* (i.e. covariances) for intercept and slope:
# and calculate posterior mode and credible intervals for each
S.modelBV_RR12 <- sgmat12[,c(3,6)]
colnames(S.modelBV_RR12) <- c("S_intercepts", "S_slopes")
S.modelBV_RR12.mode <- P.modelBV_RR12.mode[1:2, 3]

posterior.mode(mcmc(S.modelBV_RR12))
```

Negative binomial model with zero-inflation

```
## S_intercepts     S_slopes
##  -0.86957837   0.04857219
```

```
HPDinterval(mcmc(S.modelBV_RR12))
```

```
##                    lower      upper
## S_intercepts -1.2501098 -0.5831169
## S_slopes     -0.1991307  0.3200217
## 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_RR12 <- matrix(NA, nrow(S.modelBV_RR12) ,2)

for (i in 1:nrow(S.modelBV_RR12)) {
  P3_12 <- 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_12[k] <- P.modelBV_RR12[i, k] }
  P2_12 <- P3_12[1:2, 1:2]   # 2x2 matrix of just trait intercept & slope var-cov
  S12 <- P3_12[1:2, 3]   # selection differentials on traits (last column of P3)
  beta_post_RR12[i,] <- solve(P2_12) %*% S12   # selection gradients beta = P^-1 * S
}

colnames(beta_post_RR12) <- c("beta_intercepts", "beta_slopes")
posterior.mode(mcmc(beta_post_RR12))
```

```
## beta_intercepts     beta_slopes
##      -0.5876033       0.8737708
```

```
HPDinterval(mcmc(beta_post_RR12))
```

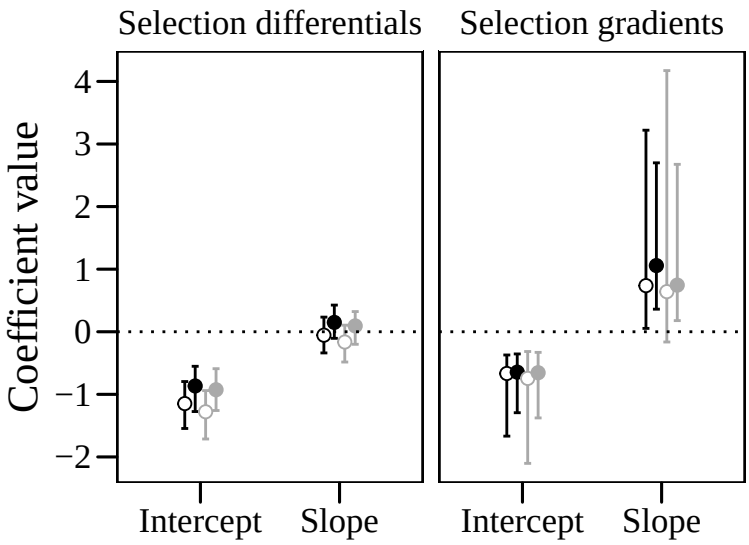
```
##                      lower     upper
## beta_intercepts -1.3933510 -0.297682
## beta_slopes      0.1188439  2.822056
## attr(,"Probability")
## [1] 0.95
```

Figure 2: Selection differentials and gradients

```
selcoefs<-rbind(
  # 1. mean_fitness_fl, no condition variable
  rbind(cbind(data.frame(coef=posterior.mode(mcmc(S.modelBV_RR2))),
              data.frame(HPDinterval(mcmc(S.modelBV_RR2)))),
        cbind(data.frame(coef=posterior.mode(mcmc(beta_post_RR2))),
              data.frame(HPDinterval(mcmc(beta_post_RR2))))%>%
    mutate(fitness_meas="mean_fitness_fl",condition="no",
           type=c("Selection differentials","Selection differentials",
```

```
                "Selection gradients","Selection gradients"),
         param=c("Intercept","Slope","Intercept","Slope")),
  # 2. mean_fitness_fl, with shoot volume
  rbind(cbind(data.frame(coef=posterior.mode(mcmc(S.modelBV_RR4))),
              data.frame(HPDinterval(mcmc(S.modelBV_RR4)))),
        cbind(data.frame(coef=posterior.mode(mcmc(beta_post_RR4))),
              data.frame(HPDinterval(mcmc(beta_post_RR4))))%>%
    mutate(fitness_meas="mean_fitness_fl",condition="yes",
         type=c("Selection differentials","Selection differentials",
                "Selection gradients","Selection gradients"),
         param=c("Intercept","Slope","Intercept","Slope")),
  # 3. mean_fitness_study, no condition variable
  rbind(cbind(data.frame(coef=posterior.mode(mcmc(S.modelBV_RR6))),
              data.frame(HPDinterval(mcmc(S.modelBV_RR6)))),
        cbind(data.frame(coef=posterior.mode(mcmc(beta_post_RR6))),
              data.frame(HPDinterval(mcmc(beta_post_RR6))))%>%
    mutate(fitness_meas="mean_fitness_study",condition="no",
         type=c("Selection differentials","Selection differentials",
                "Selection gradients","Selection gradients"),
         param=c("Intercept","Slope","Intercept","Slope")),
  # 4. mean_fitness_study, with shoot volume
  rbind(cbind(data.frame(coef=posterior.mode(mcmc(S.modelBV_RR8))),
              data.frame(HPDinterval(mcmc(S.modelBV_RR8)))),
        cbind(data.frame(coef=posterior.mode(mcmc(beta_post_RR8))),
              data.frame(HPDinterval(mcmc(beta_post_RR8))))%>%
    mutate(fitness_meas="mean_fitness_study",condition="yes",
         type=c("Selection differentials","Selection differentials",
                "Selection gradients","Selection gradients"),
         param=c("Intercept","Slope","Intercept","Slope"))
  )
```

```
ggplot(selcoefs,
       aes(x=param,y=coef,color=fitness_meas,fill=condition,shape=condition))+
  geom_errorbar(aes(ymin=lower, ymax=upper),
                width=.2,position=position_dodge(.3))+
  geom_point(size=2,position=position_dodge(.3))+
  facet_wrap(~type)+
  geom_hline(yintercept=0,lty=3)+
  scale_shape_manual(values=c(21,19))+
  scale_color_manual(values=c("black","darkgrey"))+
  scale_fill_manual(values=c("white","black"))+
  scale_y_continuous(breaks=c(-2,-1,0,1,2,3,4))+
  my_theme()+xlab(NULL)+ylab("Coefficient value")
```




```
# black: mean_fitness_fl, grey: mean_fitness_study
# hollow circles: no condition, filled circles: condition
ggsave(filename="output/figures/fig2.tiff",
       device="tiff",width=16,height=10,units="cm",dpi=300,compression="lzw")
```

Save large objects as .RData file