Bayesian data analysis - roaches cross-validation demo

Aki Vehtari

First version 2017-01-10. Last modified 2019-12-29.

Setup

Load packages

```
library(rstanarm)
library(brms)
options(mc.cores = parallel::detectCores())
library(loo)
library(ggplot2)
library(bayesplot)
theme_set(bayesplot::theme_default(base_family = "sans"))
```

1 Introduction

This notebook demonstrates cross-validation of simple misspecified model. In this case, cross-validation is useful to detect misspecification.

The example comes from Chapter 8.3 of Gelman and Hill (2007) (http://www.stat.columbia.edu/~gelman/arm/) and the introduction text for the data is from Estimating Generalized Linear Models for Count Data with rstanarm (https://cran.r-project.org/web/packages/rstanarm/vignettes/count.html) by Jonah Gabry and Ben Goodrich.

We want to make inferences about the efficacy of a certain pest management system at reducing the number of roaches in urban apartments. Here is how Gelman and Hill describe the experiment (pg. 161):

> the treatment and control were applied to 160 and 104 apartments, respectively, and the outcome measurement y_i in each apartment i was the number of roaches caught in a set of traps. Different apartments had traps for different numbers of days

In addition to an intercept, the regression predictors for the model are the pre-treatment number of roaches `roach1`, the treatment indicator `treatment`, and a variable indicating whether the apartment is in a building restricted to elderly residents `senior`. Because the number of days for which the roach traps were used is not the same for all apartments in the sample, we include it as an `exposure2` by adding $\ln(u_i))$ to the linear predictor η_i and it can be specified using the `offset` argument to `stan_glm`.

2 Poisson model

Load data

```
data(roaches)
# Roach1 is very skewed and we take a square root
roaches$sqrt_roach1 <- sqrt(roaches$roach1)
```

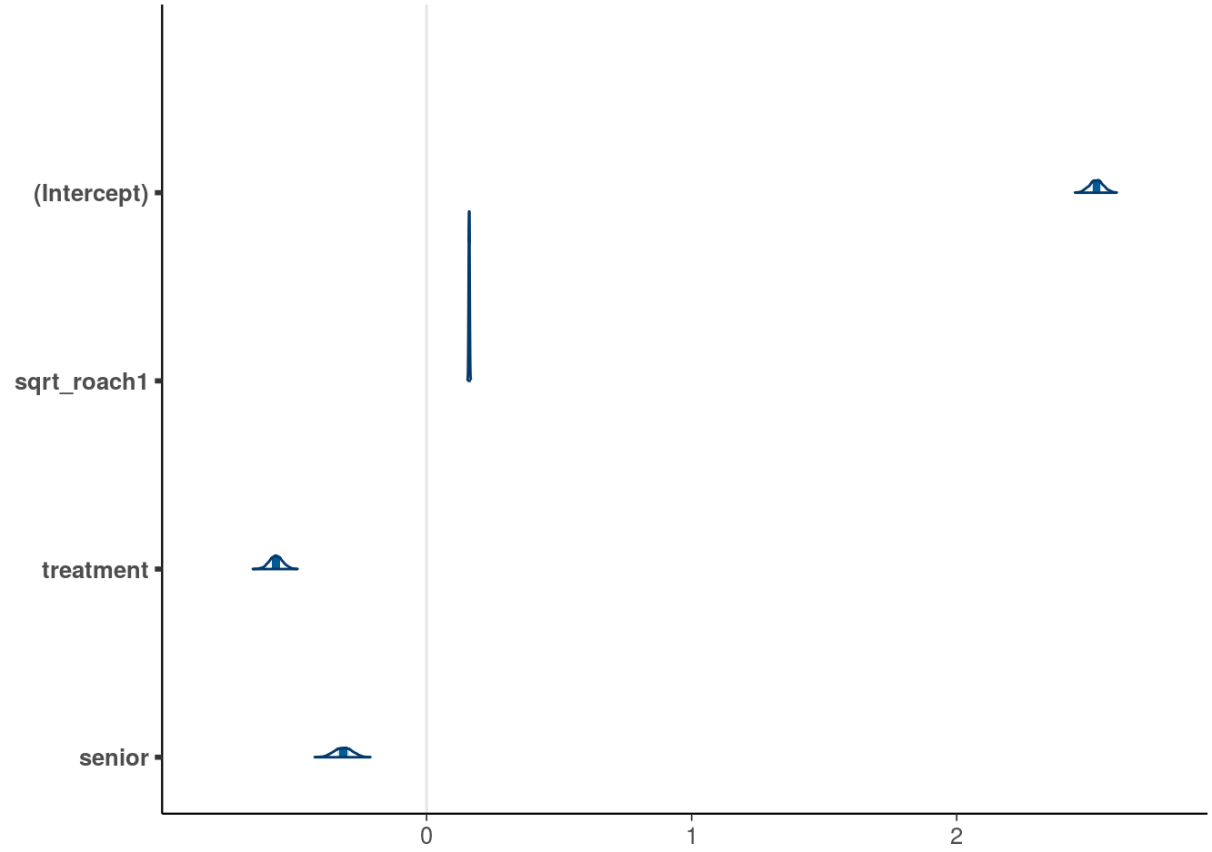
Fit with stan_glm

```
stan_glmp <- stan_glm(y ~ sqrt_roach1 + treatment + senior, offset = log(exposu
re2),
                      data = roaches, family = poisson,
                      prior = normal(0,2.5), prior_intercept = normal(0,5),
                      chains = 4, cores = 1, seed = 170400963, refresh=0)
```

2.1 Analyse posterior

Plot posterior

```
mcmc_areas(as.matrix(stan_glmp), prob_outer = .999)
```



We have all marginals significantly away from zero.

2.2 Cross-validation checking

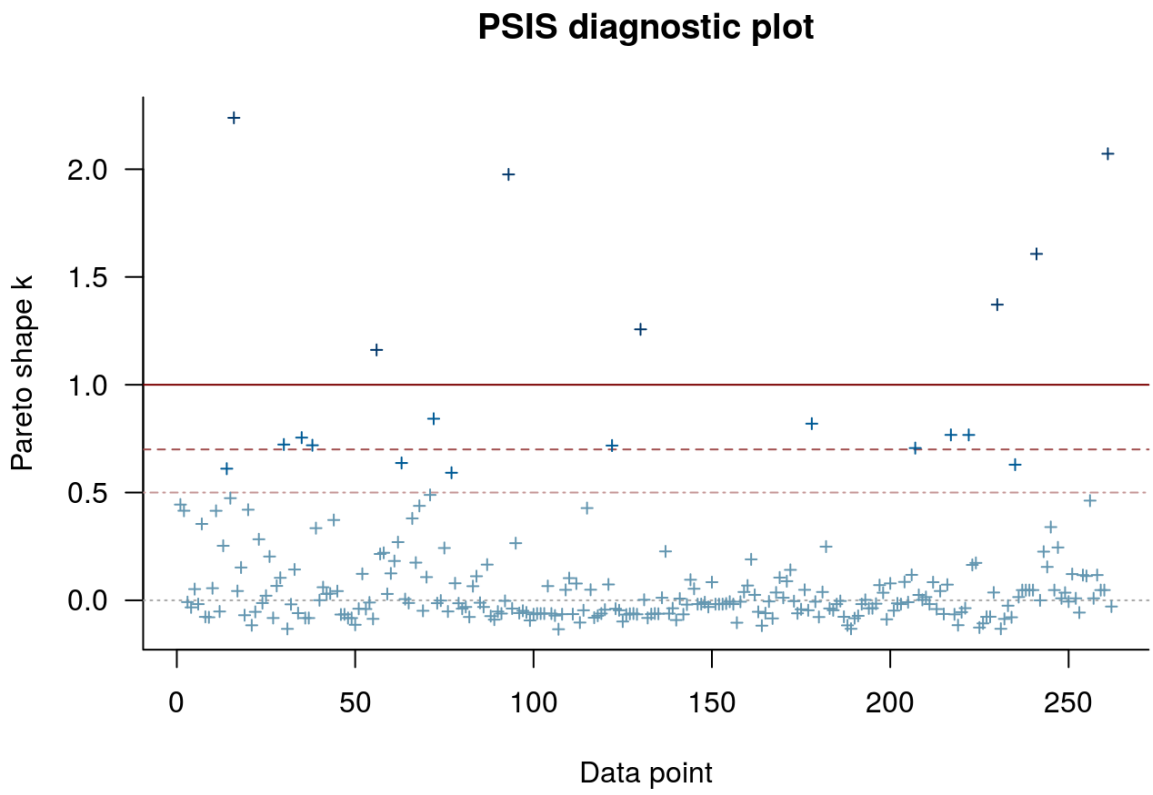
We can use Pareto-smoothed importance sampling leave-one-out cross-validation as model checking tool (Vehtari, Gelman and Gabry, 2017).

```
(loop <- loo(stan_glmp))
```

```
Computed from 4000 by 262 log-likelihood matrix

         Estimate     SE
elpd_loo  -5457.0  694.5
p_loo       251.5   54.3
looic     10914.1 1389.1
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     242   92.4%   234
 (0.5, 0.7]   (ok)         4    1.5%   92
   (0.7, 1]   (bad)        9    3.4%   36
   (1, Inf)   (very bad)   7    2.7%   1
See help('pareto-k-diagnostic') for details.
```

We got serious warnings, let's plot Pareto k values.

```
plot(loop)
```



There are several observations which are highly influential, which indicates potential model misspecification (Vehtari, Gelman and Gabry, 2017).

Before looking in more detail where the problem is or fixing it, let's check what would cross-validation say about relevance of covariates.

We form 3 models by dropping each of the covariates out.

```
stan_glmm1p <- update(stan_glmp, formula = y ~ treatment + senior)
stan_glmm2p <- update(stan_glmp, formula = y ~ sqrt_roach1 + senior)
stan_glmm3p <- update(stan_glmp, formula = y ~ sqrt_roach1 + treatment)
```

Although Pareto k values were very large we can make a quick test with PSIS-LOO (if the comparison would say there is difference, then PSIS-LOO couldn't be trusted and PSIS-LOO+ or k-fold-CV would be needed (see more in Vehtari, Gelman and Gabry, 2017)).

```
loo_compare(loo(stan_glmm1p), loop)
```

```
           elpd_diff se_diff
stan_glmp       0.0       0.0
stan_glmm1p -3001.8     686.9
```

```
loo_compare(loo(stan_glmm2p), loop)
```

```
            elpd_diff se_diff
stan_glmp      0.0       0.0
stan_glmm2p -236.3     201.4
```

```
loo_compare(loo(stan_glmm3p), loop)
```

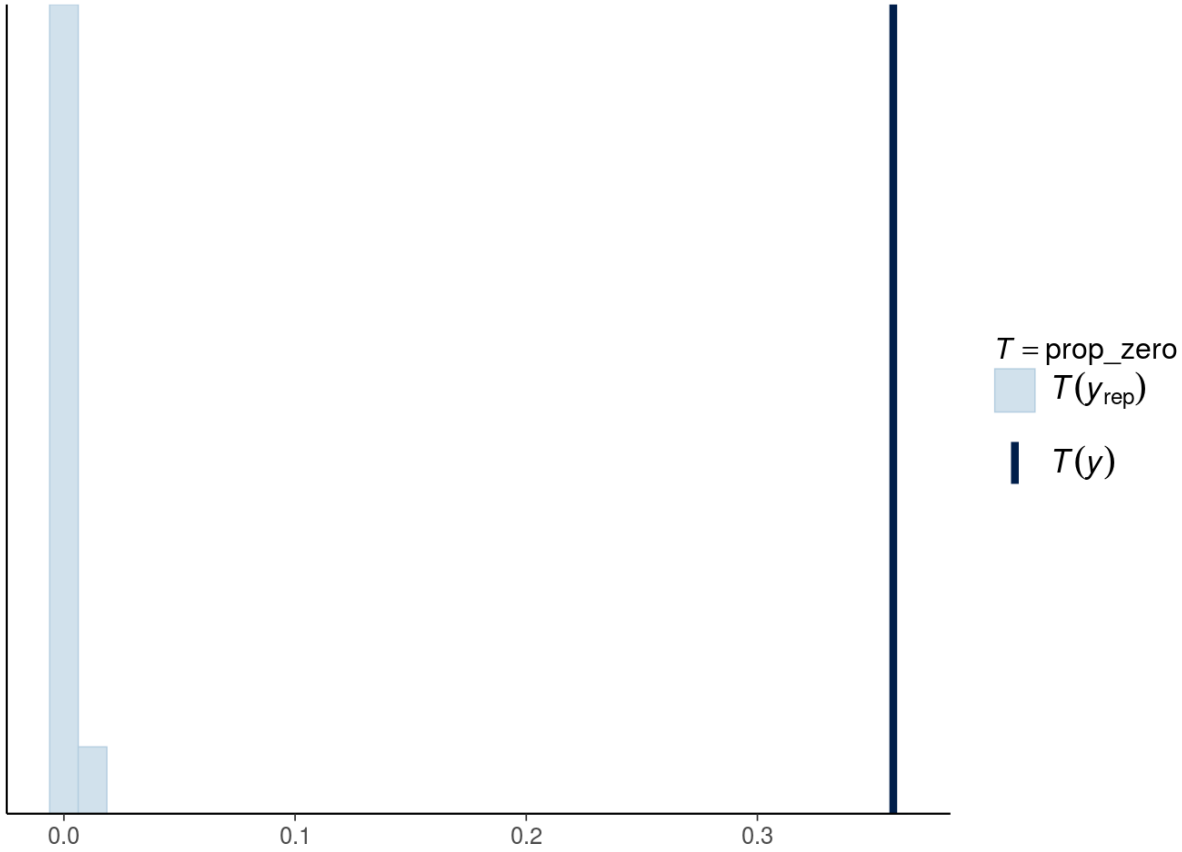
```
            elpd_diff se_diff
stan_glmp     0.0       0.0
stan_glmm3p -18.4      92.3
```

Based on this the roaches covariate would be relevant, but although dropping treatment or senior covariate will make a large change to elpd, the uncertainty is also large and cross-validation states that these covariates are not necessarily relevant! The posterior marginals are conditional on the model, but cross-validation is more cautious by not using any model for the future data distribution.

2.3 Posterior predictive checking

It's also good to remember that in addition of cross-validation, the posterior predictive checks can often detect problems and also provide more information about the reason. Here we test the proportion of zeros predicted by the model and compare them to the observed number of zeros.

```
prop_zero <- function(y) mean(y == 0)
(prop_zero_test1 <- pp_check(stan_glmp, plotfun = "stat", stat = "prop_zero"))
```



3 Negative binomial model

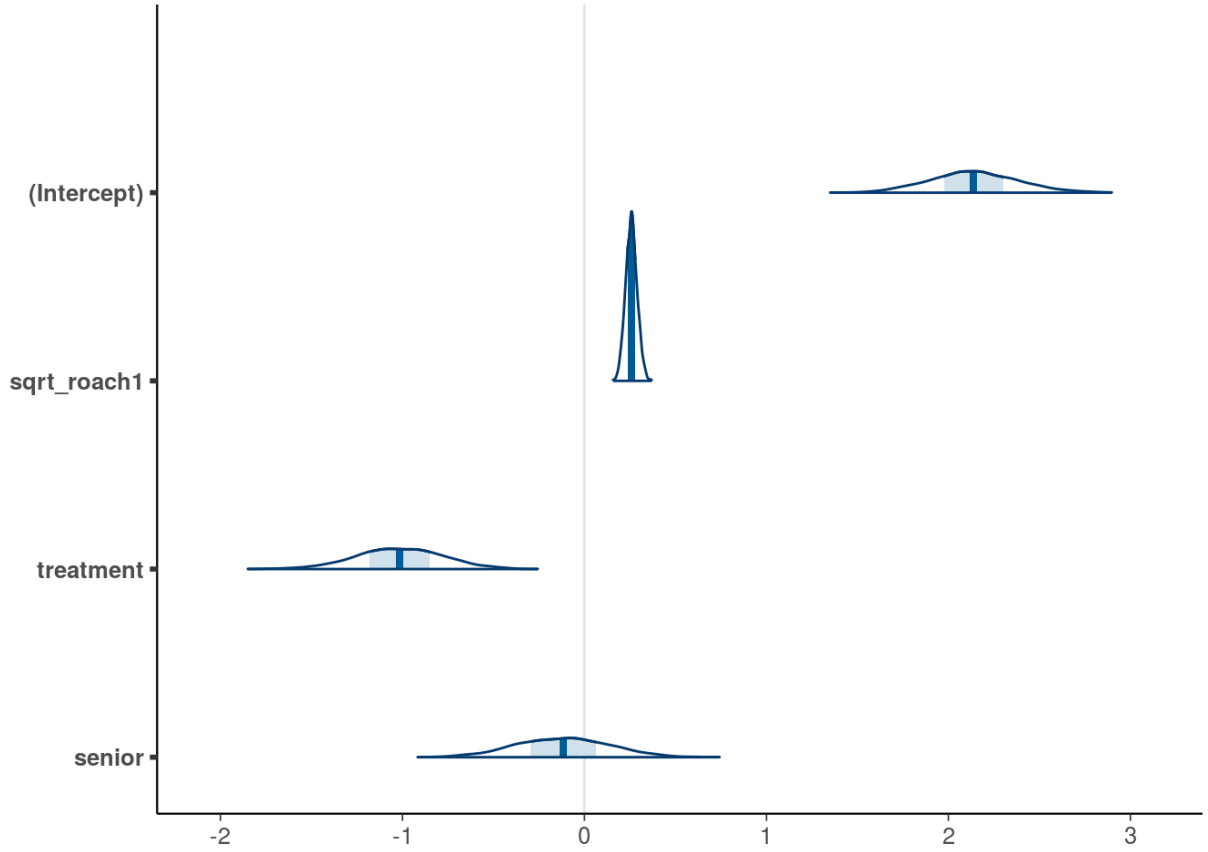
Next we change the Poisson model to a more robust negative binomial model

```
stan_glmnb <- update(stan_glmp, family = neg_binomial_2)
```

3.1 Analyse posterior

Plot posterior

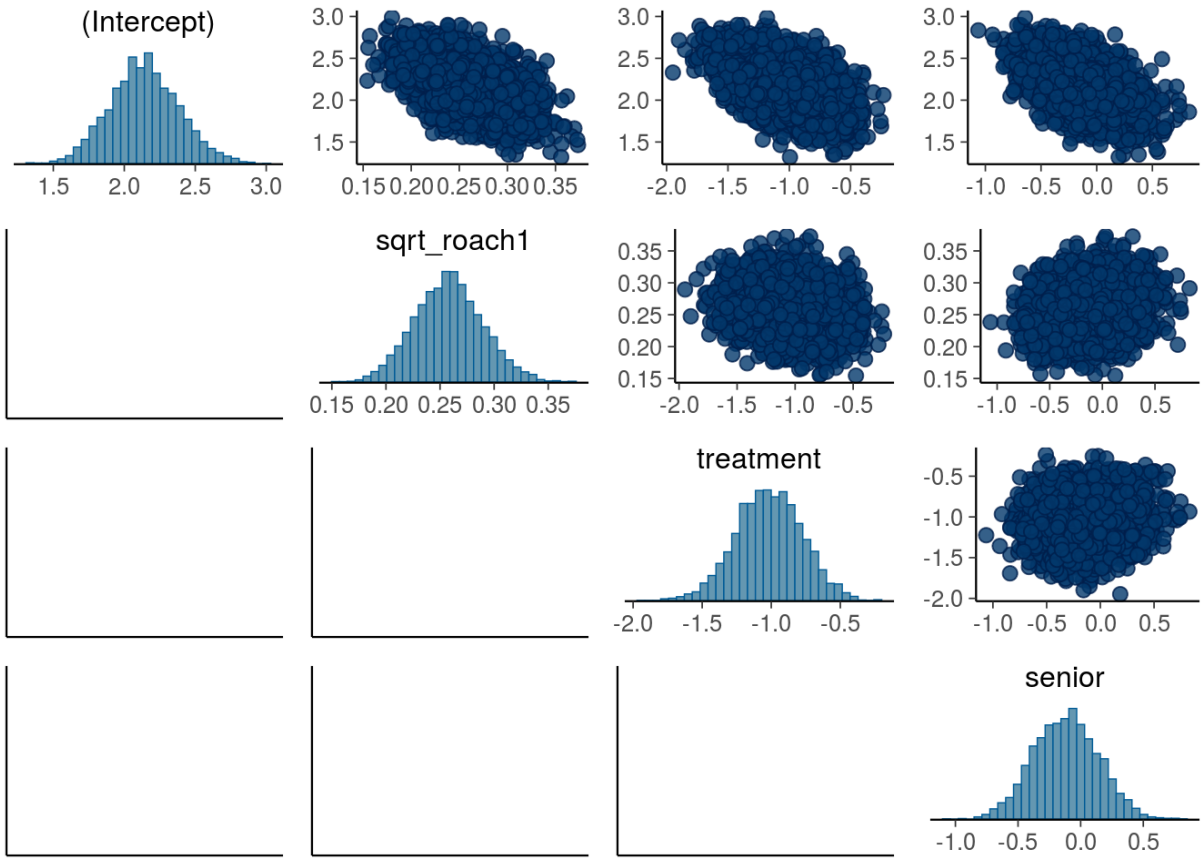
```
mcmc_areas(as.matrix(stan_glmnb), prob_outer = .999,
    pars = c("(Intercept)","sqrt_roach1","treatment","senior"))
```



Treatment effect is much closer to zero, and senior effect has lot of probability mass on both sides of 0. So it matters, which model we use.

We discuss posterior dependencies in more detail in `collinear` notebook, but for reference we plot also here paired marginals.

```
mcmc_pairs(as.matrix(stan_glmnb),pars = c("(Intercept)","sqrt_roach1","treatmen
t","senior"))
```



There are some posterior correlations, but not something which would change our conclusions.

3.2 Cross-validation checking

Let's check PSIS-LOO Pareto k diagnostics

```
(loonb <- loo(stan_glmnb))
```

```
Computed from 4000 by 262 log-likelihood matrix

         Estimate   SE
elpd_loo   -882.1 38.3
p_loo         8.6  3.8
looic      1764.1 76.7
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     260   99.2%   1800
 (0.5, 0.7]   (ok)         1     0.4%   193
   (0.7, 1]   (bad)        0     0.0%   <NA>
   (1, Inf)   (very bad)   1     0.4%   34
See help('pareto-k-diagnostic') for details.
```

All khat's are ok, which indicates that negative-Binomial would be better (for final results it would be good to run PSIS-LOO+). We can also compare Poisson and negative-Binomial.

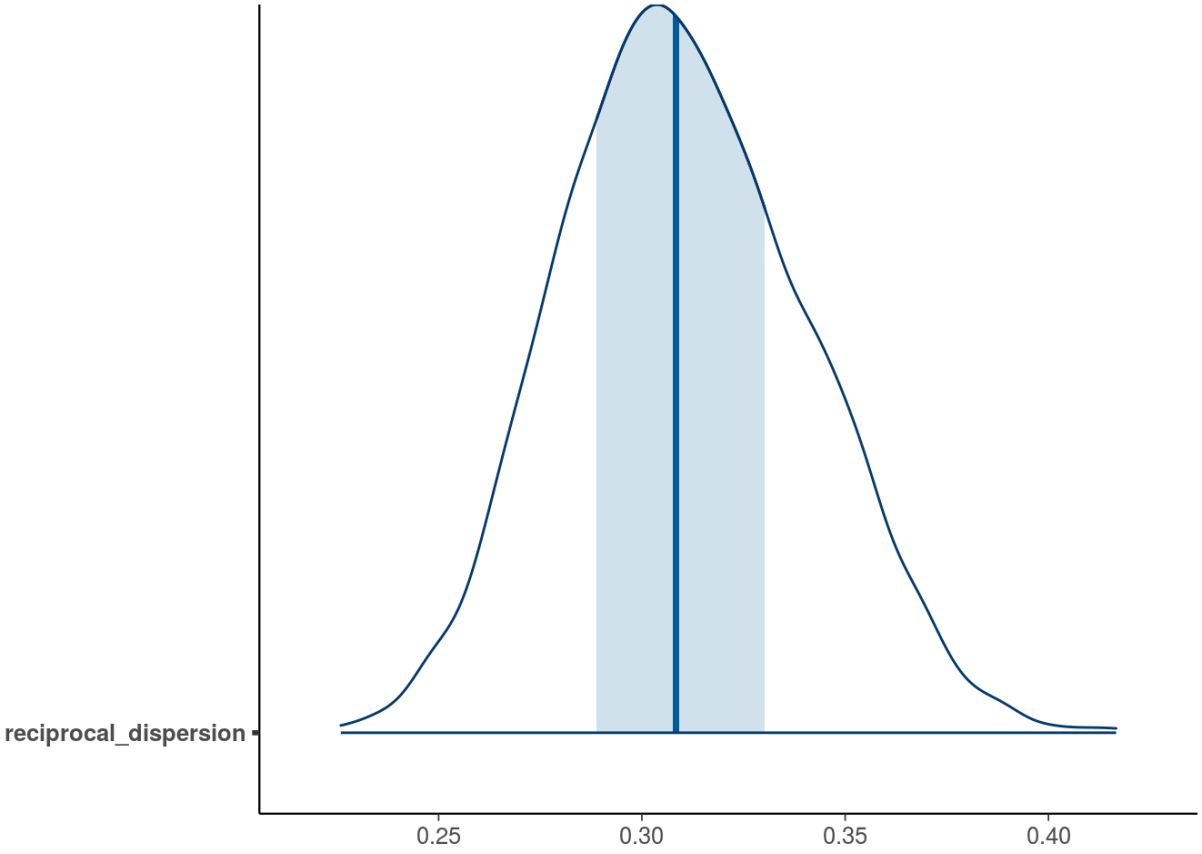
```
loo_compare(loop, loonb)
```

```
           elpd_diff se_diff
stan_glmnb     0.0       0.0
stan_glmp  -4575.0     674.5
```

Negative-Binomial model is clearly better than Poisson.

As Poisson is a special case of negative-Binomial, we could have also seen that Poisson is likely by looking at the posterior of the over-dispersion parameter (which gets very small values).

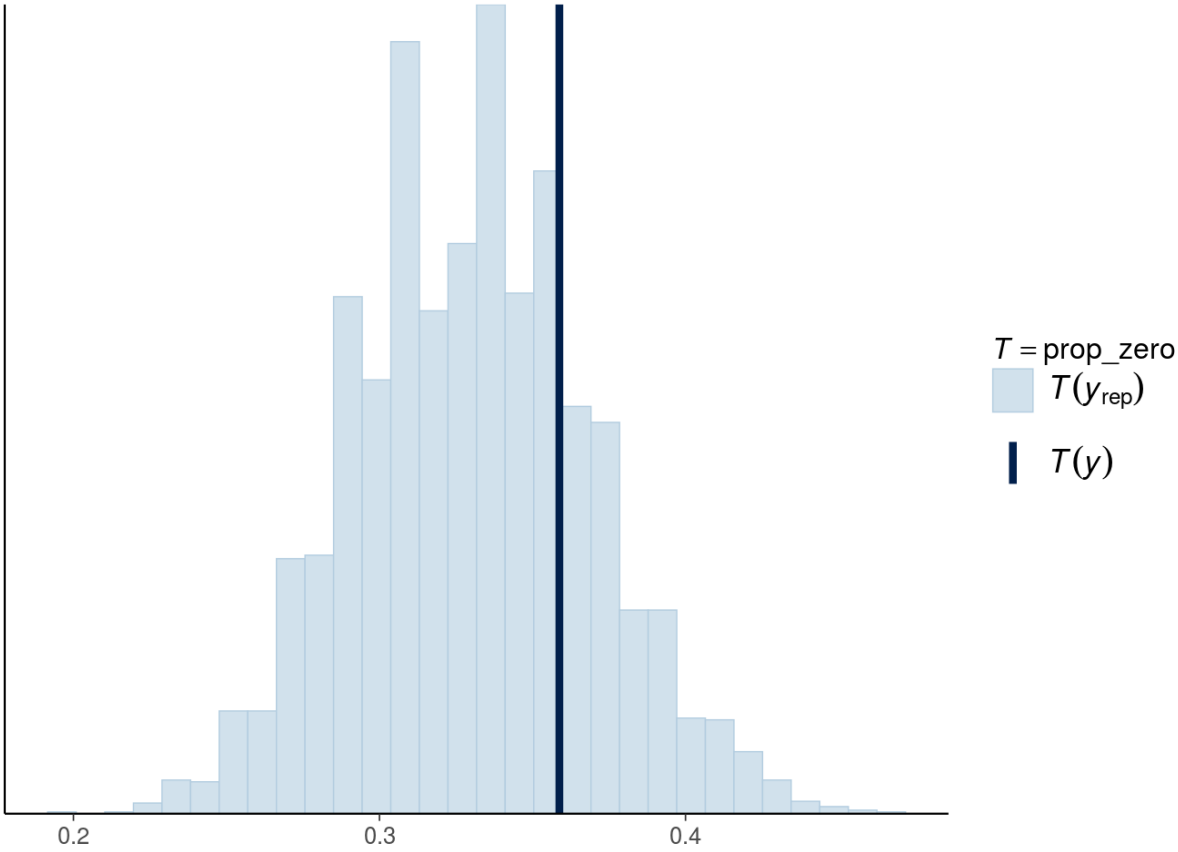
```
mcmc_areas(as.matrix(stan_glmnb), prob_outer = .999,
    pars = c("reciprocal_dispersion"))
```



3.3 Posterior predictive checking

We next use posterior predictive checking to check that the improved model can also predict the proportion of zeros well.

```
(prop_zero_test2 <- pp_check(stan_glmnb, plotfun = "stat", stat = "prop_zero"))
```



The result looks much better than for the Poisson model.

3.4 Predictive relevance of covariates

Let's finally check cross-validation model comparison that it agrees on relevance of covariates

```
stan_glmm1nb <- update(stan_glmm1p, family = neg_binomial_2)
stan_glmm2nb <- update(stan_glmm2p, family = neg_binomial_2)
stan_glmm3nb <- update(stan_glmm3p, family = neg_binomial_2)
```

```
loo_compare(loo(stan_glmm1nb), loonb)
```

```
            elpd_diff se_diff
stan_glmnb     0.0       0.0
stan_glmm1nb -35.1       9.3
```

```
loo_compare(loo(stan_glmm2nb), loonb)
```

```
            elpd_diff se_diff
stan_glmnb     0.0       0.0
stan_glmm2nb -8.2        5.5
```

```
loo_compare(loo(stan_glmm3nb), loonb)
```

```
             elpd_diff se_diff
stan_glmm3nb  0.0       0.0
stan_glmnb   -2.5       2.0
```

Roaches1 has clear effect. Treatment effect is visible in marginal posterior, but as discussed in `betablockers` demo, cross-validation is not good for detecting weak effects. Based on cross-validation senior effect is also not relevant.

Conclusion from the analysis would be then that, treatment is likely to help, but it's difficult to predict the number of roaches given treatment or not.

4 Poisson model with "random effects"

Sometimes overdispersion is modelled by adding "random effects" for each individual. The following example illustrates computational problems in this approach.

```
roaches$id <- 1:dim(roaches)[1]
```

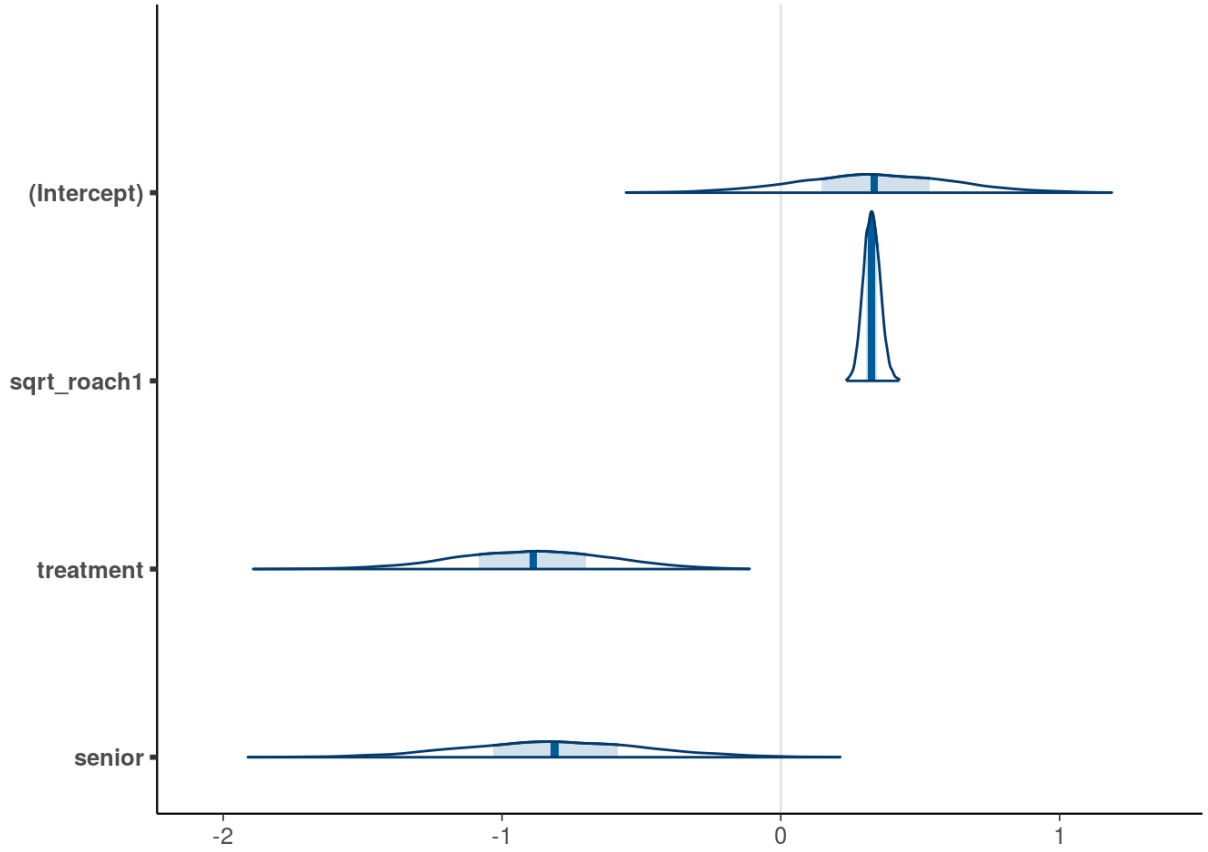
Fit with stan_glm

```
stan_glmpr <- stan_glmer(y ~ sqrt_roach1 + treatment + senior + (1 | id),
                         offset = log(exposure2),
                         data = roaches, family = poisson,
                         prior = normal(0,2.5), prior_intercept = normal(0,5),
                         chains = 4, cores = 4, iter = 4000,
                         seed = 170400963, refresh=0)
```

4.1 Analyse posterior

Plot posterior

```
mcmc_areas(as.matrix(stan_glmpr), prob_outer = .999,
    pars = c("(Intercept)","sqrt_roach1","treatment","senior"))
```



The marginals are similar as with negative-binomial model except the marginal for senior effect is clearly away from zero.

4.2 Cross-validation checking

Let's check PSIS-LOO.

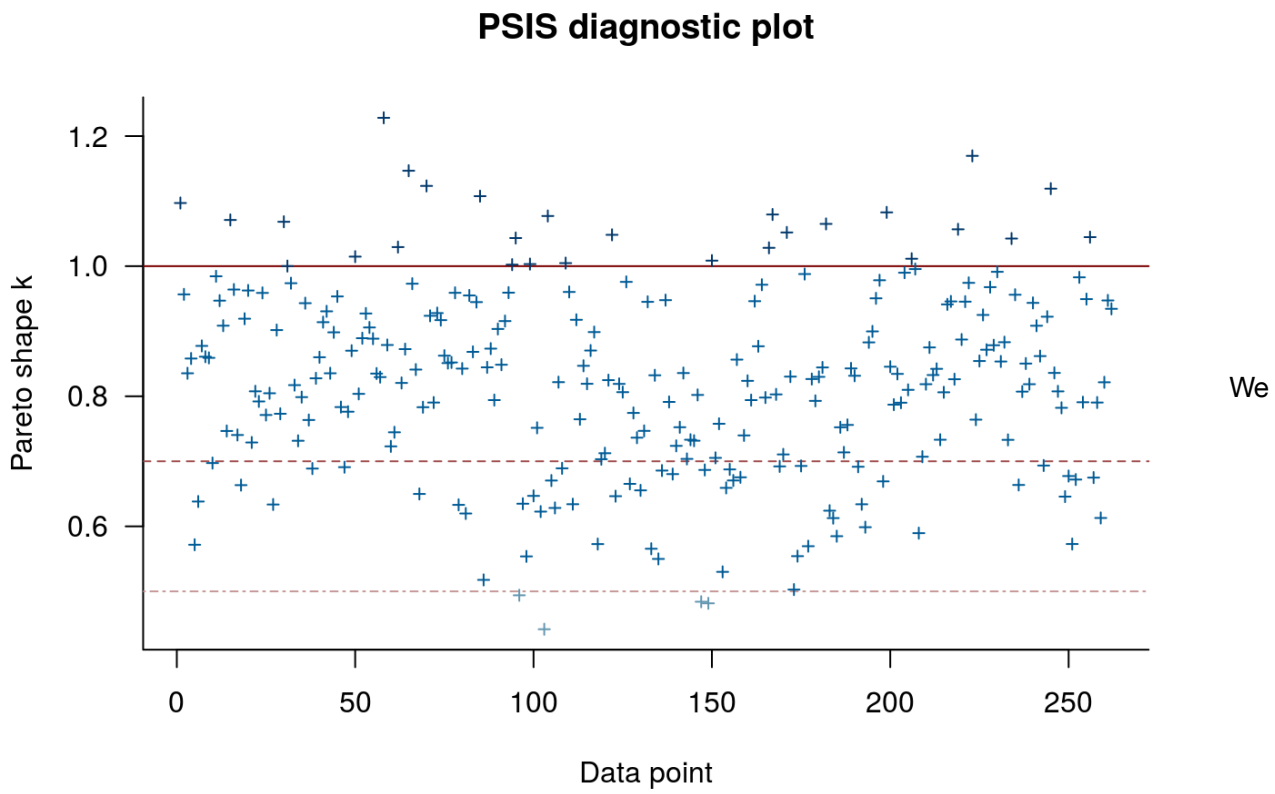
```
(loopr <- loo(stan_glmpr))
```

```
Computed from 8000 by 262 log-likelihood matrix

         Estimate   SE
elpd_loo   -636.7 24.4
p_loo       172.4  4.9
looic      1273.4 48.7
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)       4    1.5%   620
 (0.5, 0.7]   (ok)        54   20.6%   121
   (0.7, 1]   (bad)      176   67.2%   16
   (1, Inf)   (very bad)  28   10.7%   4
See help('pareto-k-diagnostic') for details.
```

We got serious warnings, which in this case is due to having a “random effect” parameter for each observations. Removing one observation changes the posterior for that random effect so much that importance sampling fails (even with Pareto smoothing). Note that WAIC would fail due to the same reason.

We can also plot Pareto k values.

```
plot(loopr)
```



We have a very large number of high k values which indicates very flexible model.

While importance sampling in PSIS-LOO can fail for “random effect” model, we can use K-fold-CV instead to re-fit the model 10 times, each time leaving out 10% of the observations. This shows that cross-validation itself is not infeasible for “random effect” models.

```
(kcvpr <- kfold(stan_glmpr, K=10))
```

```
Based on 10-fold cross-validation

           Estimate   SE
elpd_kfold   -879.6 38.2
p_kfold          NA   NA
kfoldic      1759.3 76.3
```

loo package allows comparing PSIS-LOO and K-fold-CV results

```
loo_compare(loonb, kcvpr)
```

There is not much difference, and this difference could also be explained by K-fold-CV using only 90% of observations for the posteriors, while PSIS-LOO is using 99.6% of observations for the posteriors. We can check this by running K-fold-CV also for negative-binomial model.

```
(kcvnb <- kfold(stan_glmnb, K=10))
```

```
Based on 10-fold cross-validation

           Estimate   SE
elpd_kfold   -883.6 38.8
p_kfold          NA   NA
kfoldic      1767.2 77.5
```

```
loo_compare(kcvnb, kcvpr)
```

```
           elpd_diff se_diff
stan_glmpr  0.0       0.0
stan_glmnb -3.9       8.7
```

When both models are assessed using K-fold-CV, the difference in predictive performance is very small. The models can still have different predictive distributions.

Now that we've seen that based on robust K-fold-CV there is not much difference between negative-binomial and random-effect-Poisson models, we can also check how bad the comparison would have been with PSIS-LOO.

```
loo_compare(loonb, loopr)
```

```
           elpd_diff se_diff
stan_glmpr    0.0       0.0
stan_glmnb -245.3      17.4
```

If we would have ignored Pareto-k warnings, we would have mistakenly assumed that random effect model is much better. Note that WAIC is (as usual) even worse

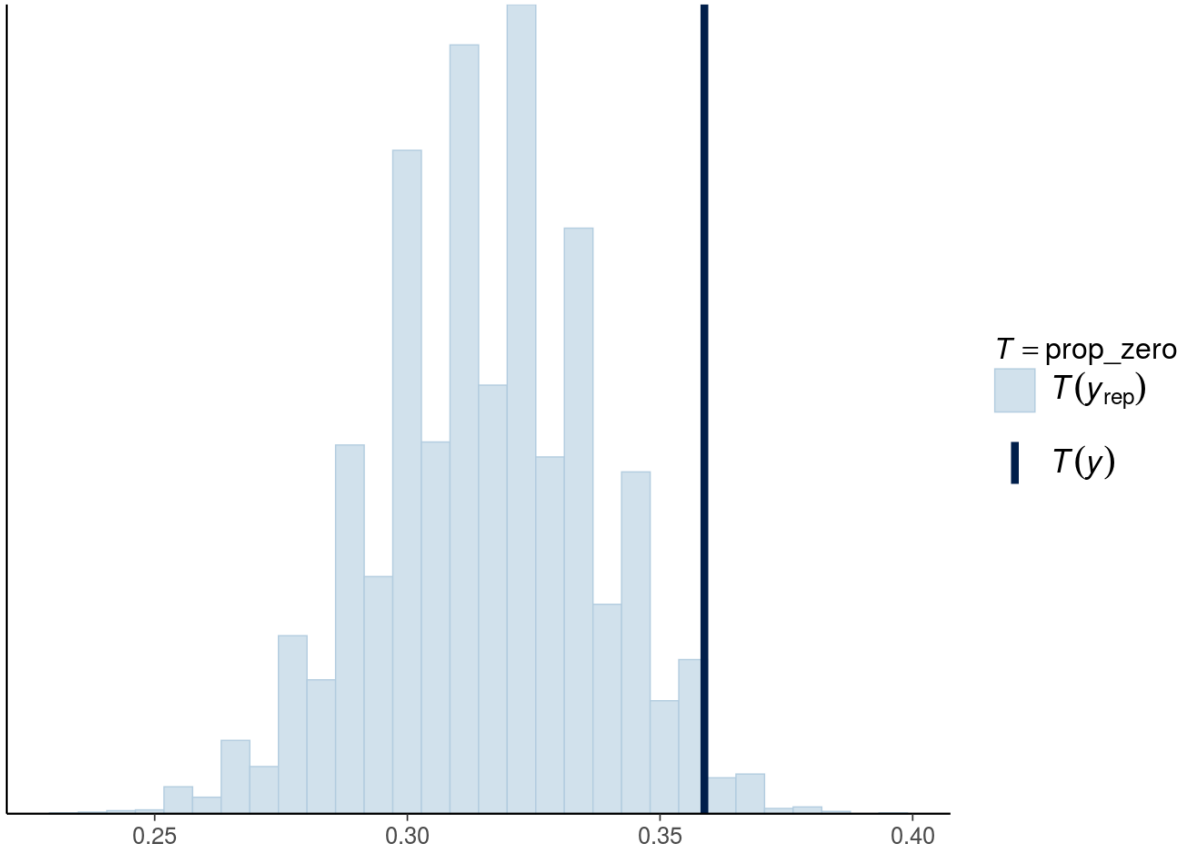
```
loo_compare(waic(stan_glmnb), waic(stan_glmpr))
```

```
           elpd_diff se_diff
stan_glmpr    0.0       0.0
stan_glmnb -309.8      18.3
```

4.3 Posterior predictive checking

We do posterior predictive checking also for Poisson with "random effects" model.

```
(prop_zero_test3 <- pp_check(stan_glmpr, plotfun = "stat", stat = "prop_zero"))
```



The proportion of zeros in posterior predictive simulations from Poisson with “random effects” model is different (less variation) than from negative-binomial model. The models are similar in that way that are modeling over-dispersion, but the model for the over-dispersion is different.

5 Zero-inflated negative-binomial model

As the proportion of zeros is quite high in the data, it is worthwhile to test also a zero-inflated negative-binomial model, which is a mixture of two models - logistic regression to model the proportion of extra zero counts - negative-binomial model

We switch to brms as rstanarm doesn’t support zero-inflated negative-binomial model

```
brm_glmznb <- brm(bf(y ~ sqrt_roach1 + treatment + senior + offset(log(exposure
2)),
                  zi ~ sqrt_roach1 + treatment + senior + offset(log(exposure
2))),
                  family=zero_inflated_negbinomial(), data=roaches,
                  prior=set_prior("normal(0,1)"), seed=170400963, refresh=500)
```

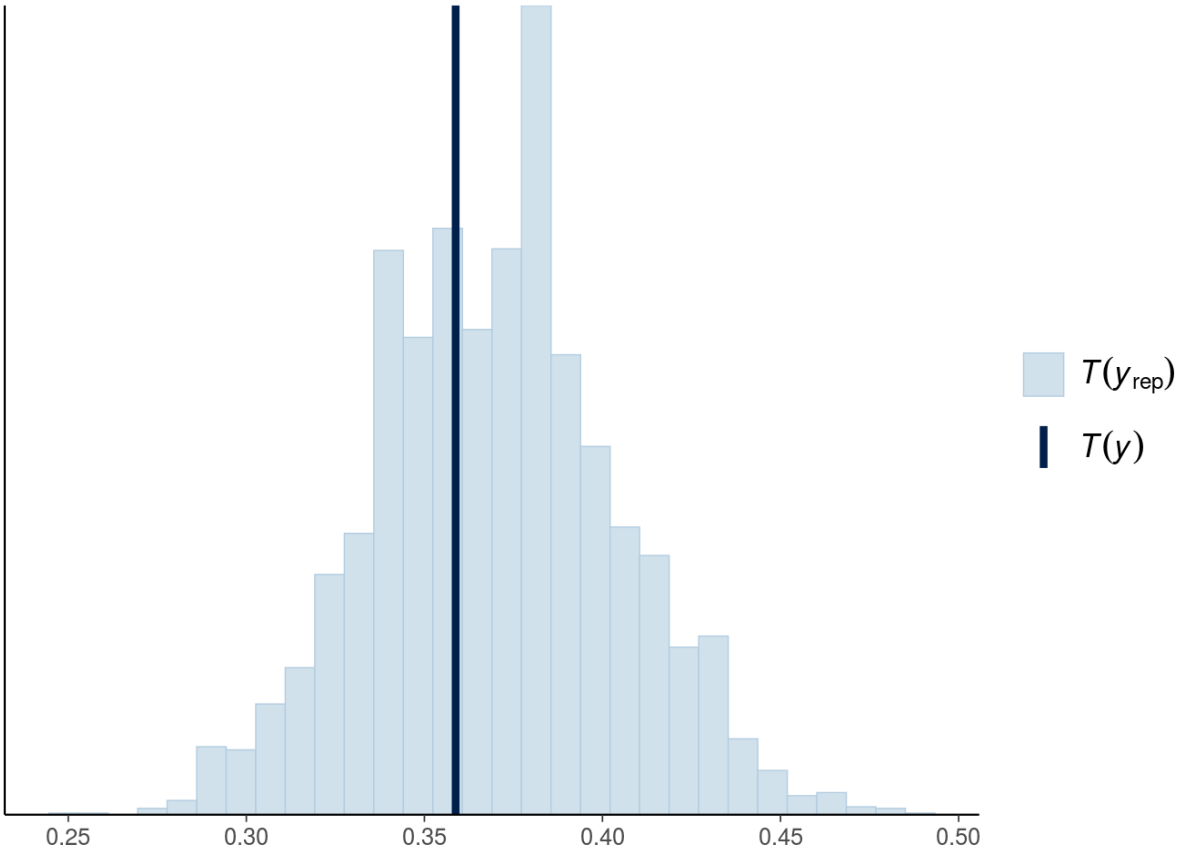
```
looznb <- loo(brm_glmznb)
loo_compare(loonb, looznb)
```

```
           elpd_diff se_diff
brm_glmznb   0.0       0.0
stan_glmnb -23.2       7.0
```

Based on loo, zero-inflated negative-binomial is clearly better.

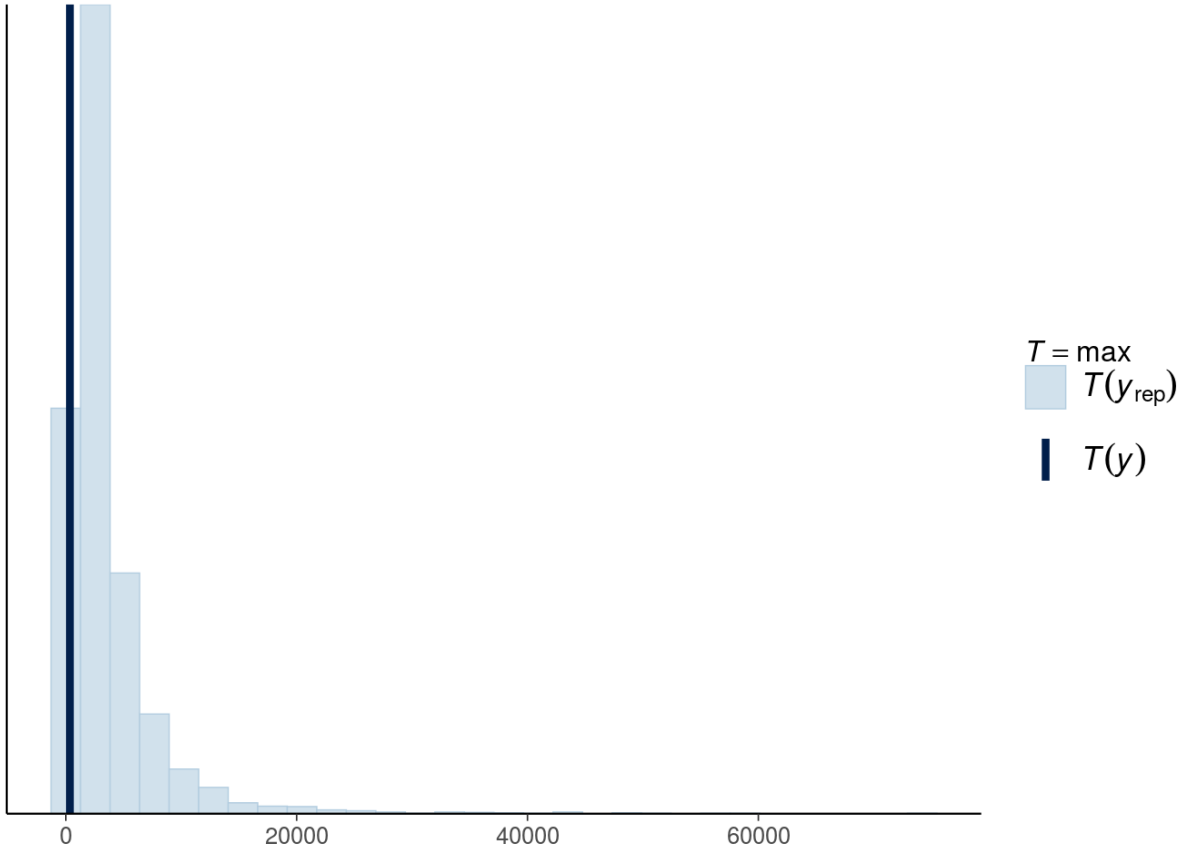
Posterior predictive checking provides further evidence that zero-inflated negative-binomial is better.

```
yrepnzb <- posterior_predict(brm_glmznb)
(prop_zero_test4 <- ppc_stat(y=roaches$y, yrepnzb, stat=function(y) mean(y==0
)))
```



Proportion of zeros is similar, but has more variation compared to negative-binomial model. However there is much bigger difference in max count test statistic. We first check the max count PPC for negative-binomial model.

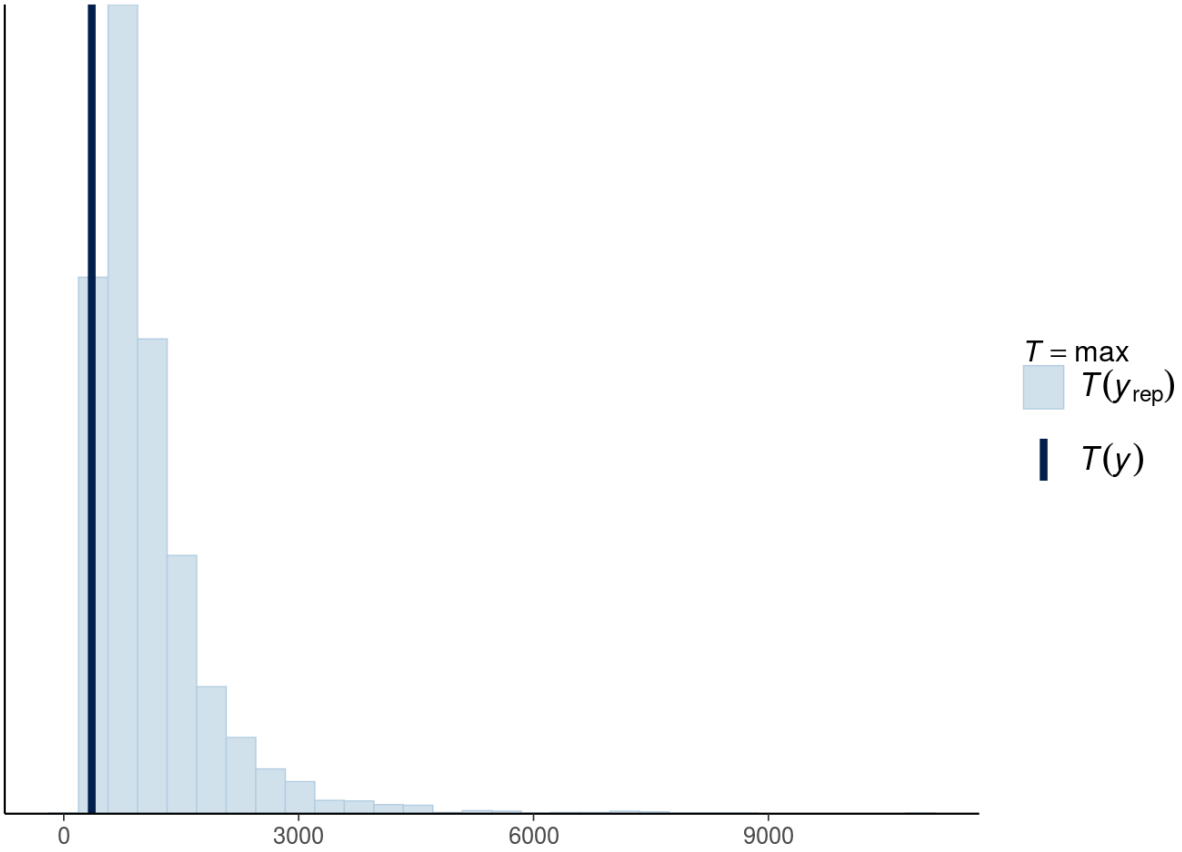
```
(max_test_nb <- pp_check(stan_glmnb, plotfun = "stat", stat = "max"))
```



which shows that negative-binomial model is predicting often 10-100 larger roach counts than in the data (40,000 roaches in one trap is a lot).

The max count PPC for zero-inflated negative-binomial model

```
(max_test_znb <- ppc_stat(y=roaches$y, yrepnzb, stat="max"))
```

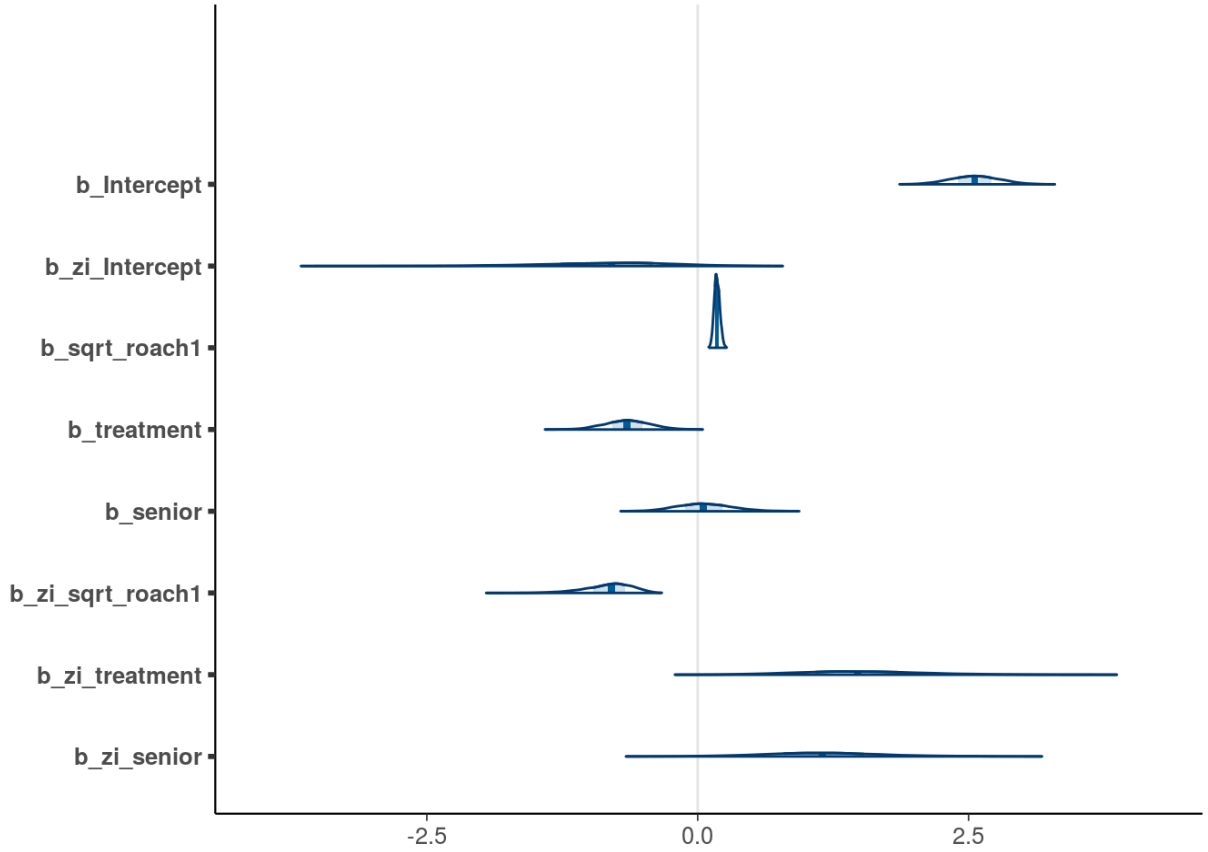


is much better, although still the max counts can be 10 times bigger than the max count in the data.

5.1 Analyse posterior

Plot posterior

```
mcmc_areas(as.matrix(brm_glmznb)[,1:8], prob_outer = .999)
```



The marginals for negative-binomial part are similar to marginals in the plain negative-binomial model. The marginal effects for the logistic part have opposite sign as the logistic part is modelling the extra zeros.

5.2 Cross-validation checking

5.3 Predictive relevance of covariates

Let's finally check cross-validation model comparison to see whether improved model has effect on the predictive performance comparison.

```
brm_glmm1znb <- brm(bf(y ~ treatment + senior + offset(log(exposure2)),
                   zi ~ treatment + senior + offset(log(exposure2))),
                   family=zero_inflated_negbinomial(), data=roaches,
                   prior=set_prior("normal(0,1)"), seed=170400963, refresh=500)
brm_glmm2znb <- brm(bf(y ~ sqrt_roach1 + senior + offset(log(exposure2)),
                   zi ~ sqrt_roach1 + senior + offset(log(exposure2))),
                   family=zero_inflated_negbinomial(), data=roaches,
                   prior=set_prior("normal(0,1)"), seed=170400963, refresh=500)
brm_glmm3znb <- brm(bf(y ~ sqrt_roach1 + treatment + offset(log(exposure2)),
                   zi ~ sqrt_roach1 + treatment + offset(log(exposure2))),
                   family=zero_inflated_negbinomial(), data=roaches,
                   prior=set_prior("normal(0,1)"), seed=170400963, refresh=500)
```

```
loo_compare(loo(brm_glmm1znb),loozn b)
```

```
             elpd_diff se_diff
brm_glmznb     0.0       0.0
brm_glmm1znb -57.8      10.2
```

```
loo_compare(loo(brm_glmm2znb),looznb)
```

```
             elpd_diff se_diff
brm_glmznb    0.0       0.0
brm_glmm2znb -8.7       5.0
```

```
loo_compare(loo(brm_glmm3znb),looznb)
```

```
             elpd_diff se_diff
brm_glmm3znb  0.0       0.0
brm_glmznb   -0.9       2.5
```

Roaches1 has clear effect. Treatment effect is visible in marginal posterior, but as discussed in `betablockers` demo, cross-validation is not that good for detecting weak effects.

References

Vehtari, A., Gelman, A. and Gabry, J. (2017) 'Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC', *Statistics and Computing*, 27(5), pp. 1413–1432. doi: 10.1007/s11222-016-9696-4 (https://doi.org/10.1007/s11222-016-9696-4).

Licenses

Original Computing Environment

```
sessionInfo()
```

```
R version 3.5.1 (2018-07-02)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.6 LTS

Matrix products: default
BLAS: /usr/lib/libblas/libblas.so.3.6.0
LAPACK: /usr/lib/lapack/liblapack.so.3.6.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.utf8         LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=fi_FI.utf8        LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] bayesplot_1.7.1 ggplot2_3.2.1   loo_2.1.0.9000  brms_2.10.0
[5] rstanarm_2.19.3 Rcpp_1.0.3

loaded via a namespace (and not attached):
 [1] nlme_3.1-142         matrixStats_0.55.0   xts_0.11-2
 [4] threejs_0.3.1        rstan_2.19.2         tools_3.5.1
 [7] backports_1.1.5      R6_2.4.1             DT_0.10
[10] lazyeval_0.2.2       colorspace_1.4-1     withr_2.1.2
[13] tidyselect_0.2.5     gridExtra_2.3        prettyunits_1.0.2
[16] processx_3.4.1       Brobdingnag_1.2-6    compiler_3.5.1
[19] cli_1.1.0            shinyjs_1.0          labeling_0.3
[22] colourpicker_1.0     scales_1.1.0         dygraphs_1.1.1.6
[25] ggridges_0.5.1       callr_3.3.2          stringr_1.4.0
[28] digest_0.6.23        StanHeaders_2.19.0   minqa_1.2.4
[31] rmarkdown_1.18       base64enc_0.1-3      pkgconfig_2.0.3
[34] htmltools_0.4.0      lme4_1.1-21          fastmap_1.0.1
[37] htmlwidgets_1.5.1    rlang_0.4.2          shiny_1.4.0
[40] farver_2.0.1         zoo_1.8-6            crosstalk_1.0.0
[43] gtools_3.8.1         dplyr_0.8.3          inline_0.3.15
[46] magrittr_1.5         Matrix_1.2-18        munsell_0.5.0
[49] abind_1.4-5          lifecycle_0.1.0      stringi_1.4.3
[52] yaml_2.2.0           MASS_7.3-51.4        pkgbuild_1.0.6
[55] plyr_1.8.4           grid_3.5.1           parallel_3.5.1
[58] promises_1.1.0       crayon_1.3.4         miniUI_0.1.1.1
[61] lattice_0.20-38      splines_3.5.1        knitr_1.26
[64] ps_1.3.0             pillar_1.4.2         igraph_1.2.4.2
[67] boot_1.3-23          markdown_1.1         shinystan_2.5.0
[70] reshape2_1.4.3       codetools_0.2-16     stats4_3.5.1
[73] rstantools_2.0.0     glue_1.3.1           evaluate_0.14
[76] nloptr_1.2.1         httpuv_1.5.2         gtable_0.3.0
[79] purrr_0.3.3          assertthat_0.2.1     xfun_0.11
[82] mime_0.7             xtable_1.8-4         coda_0.19-3
```

```
[85] later_1.0.0          rsconnect_0.8.15     survival_3.1-8
[88] tibble_2.1.3         shinythemes_1.1.2    bridgesampling_0.7-2
```