Homework 7b

homework 7b

1. Review part one of Bayesian Statistics Using Stan

I follows the rstan textbook and review the workflow of the bayesian statistics

```
 pest_data <- readRDS('pest_data.RDS')
str(pest_data)
```

```
## 'data.frame':    120 obs. of  14 variables:
##  $ mus                 : num  0.369 0.359 0.282 0.129 0.452 ...
##  $ building_id         : int  37 37 37 37 37 37 37 37 37 37 ...
##  $ wk_ind              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ date                : Date, format: "2017-01-15" "2017-02-14" ...
##  $ traps               : num  8 8 9 10 11 11 10 10 9 9 ...
##  $ floors              : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ sq_footage_p_floor  : num  5149 5149 5149 5149 5149 ...
##  $ live_in_super       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthly_average_rent: num  3847 3847 3847 3847 3847 ...
##  $ average_tenant_age  : num  53.9 53.9 53.9 53.9 53.9 ...
##  $ age_of_building     : num  47 47 47 47 47 47 47 47 47 47 ...
##  $ total_sq_foot       : num  41192 41192 41192 41192 41192 ...
##  $ month               : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ complaints          : num  1 3 0 1 0 0 4 3 2 2 ...
```

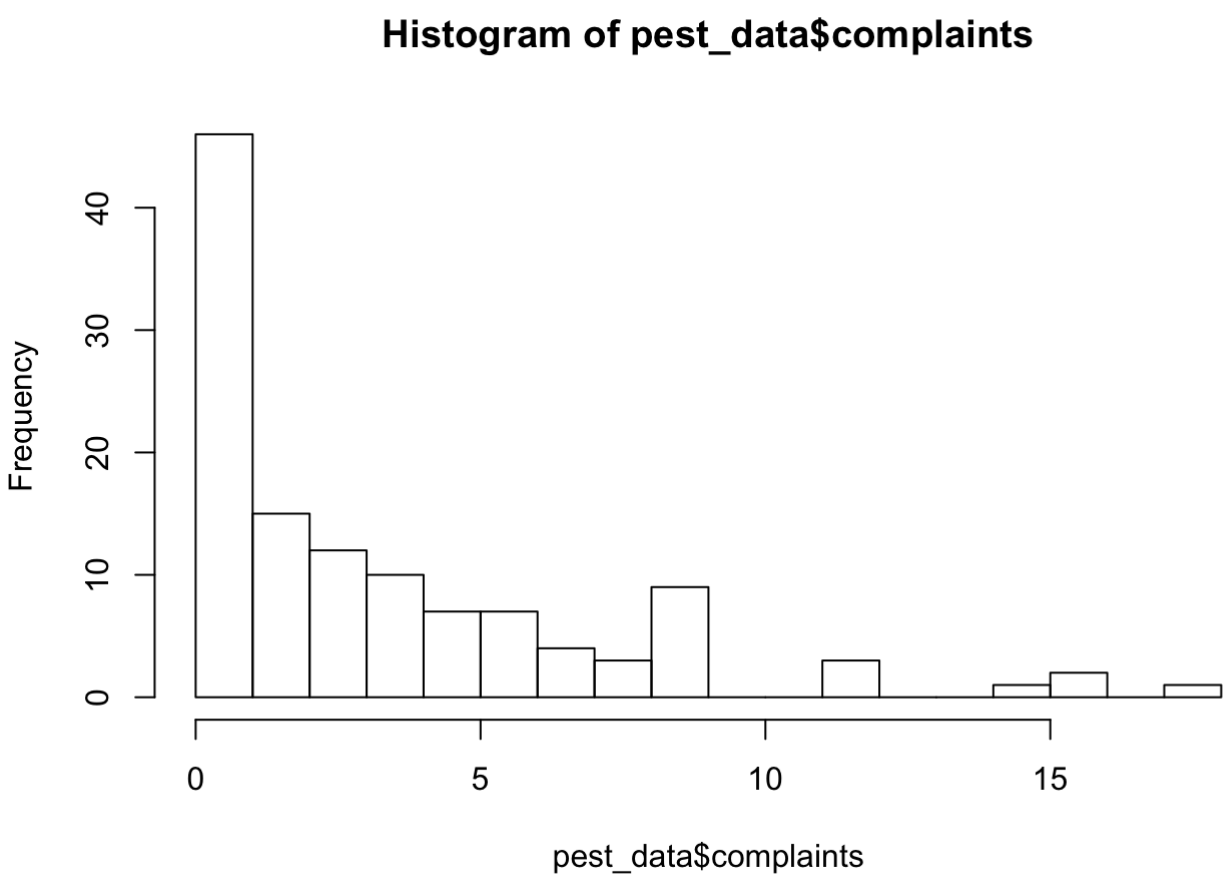
```
N_buildings <- length(unique(pest_data$building_id))
hist(pest_data$complaints,breaks=25)

#  possion regression
## generate the fake data
library(rstan)
```

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

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

```
## rstan (Version 2.17.4, GitRev: 2e1f913d3ca3)
```

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




```
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)

#### For sampling the fake data there is no parameters in the model or even the model bl
ock, can only use the fixed parameter algorithum
#### Here we set the iter =1 since we only need one set of fake parameters otherwise hal
f of the iteration results would be kept
#### the combination of stan_model and sampling is the same as stan()
comp_dgp_simple <- stan_model('simple_possion_regression_dgp.stan')
fitted_model_dgp <- sampling(comp_dgp_simple,
                        data = list(N=nrow(pest_data),mean_traps = mean(pest_data$trap
s)),
                        chains=1,iter=1,algorithm='Fixed_param',seed=123)
```

```
##
## SAMPLING FOR MODEL 'simple_possion_regression_dgp' NOW (CHAIN 1).
## Iteration: 1 / 1 [100%]  (Sampling)
##
##  Elapsed Time: 0 seconds (Warm-up)
##                3.9e-05 seconds (Sampling)
##                3.9e-05 seconds (Total)
```

```
sims_dgp <- extract(fitted_model_dgp)
str(sims_dgp)
```

```
## List of 5
##  $ traps     : num [1, 1:120] 7 5 8 11 9 6 5 6 8 9 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ iterations: NULL
##   .. ..$           : NULL
##  $ complaints: num [1, 1:120] 0 1 0 0 0 0 0 0 1 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ iterations: NULL
##   .. ..$           : NULL
##  $ alpha     : num [1(1d)] 1.29
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
##  $ beta      : num [1(1d)] -0.283
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
##  $ lp__      : num [1(1d)] 0
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
```

```
## fitting the model to the fake data
#### without specifiation, the iter number in the sampling() or stan() is 8000 and the h
alf of simulation would be kept
stan_dat_fake <- list(N= nrow(pest_data),
                      traps = sims_dgp$traps[1,],
                      complaints = sims_dgp$complaints[1,])
str(stan_dat_fake)
```

```
## List of 3
##  $ N         : int 120
##  $ traps     : num [1:120] 7 5 8 11 9 6 5 6 8 9 ...
##  $ complaints: num [1:120] 0 1 0 0 0 0 0 0 1 0 ...
```

```
comp_model_P <- stan_model('bait_possion.stan')
fit_model_P <- sampling(comp_model_P, data = stan_dat_fake, seed = 123)

## assessing parameter recovery
posterior_alpha_beta <- as.matrix(fit_model_P, pars = c('alpha','beta'))
head(posterior_alpha_beta)
```

```
##           parameters
## iterations    alpha       beta
##       [1,] 1.539087 -0.3611280
##       [2,] 1.653324 -0.3215484
##       [3,] 1.702493 -0.4242530
##       [4,] 1.991019 -0.4099450
##       [5,] 1.902890 -0.4239536
##       [6,] 1.969359 -0.3931064
```

```
true_alpha_beta <- c(sims_dgp$alpha, sims_dgp$beta)
library(bayesplot)
```

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

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

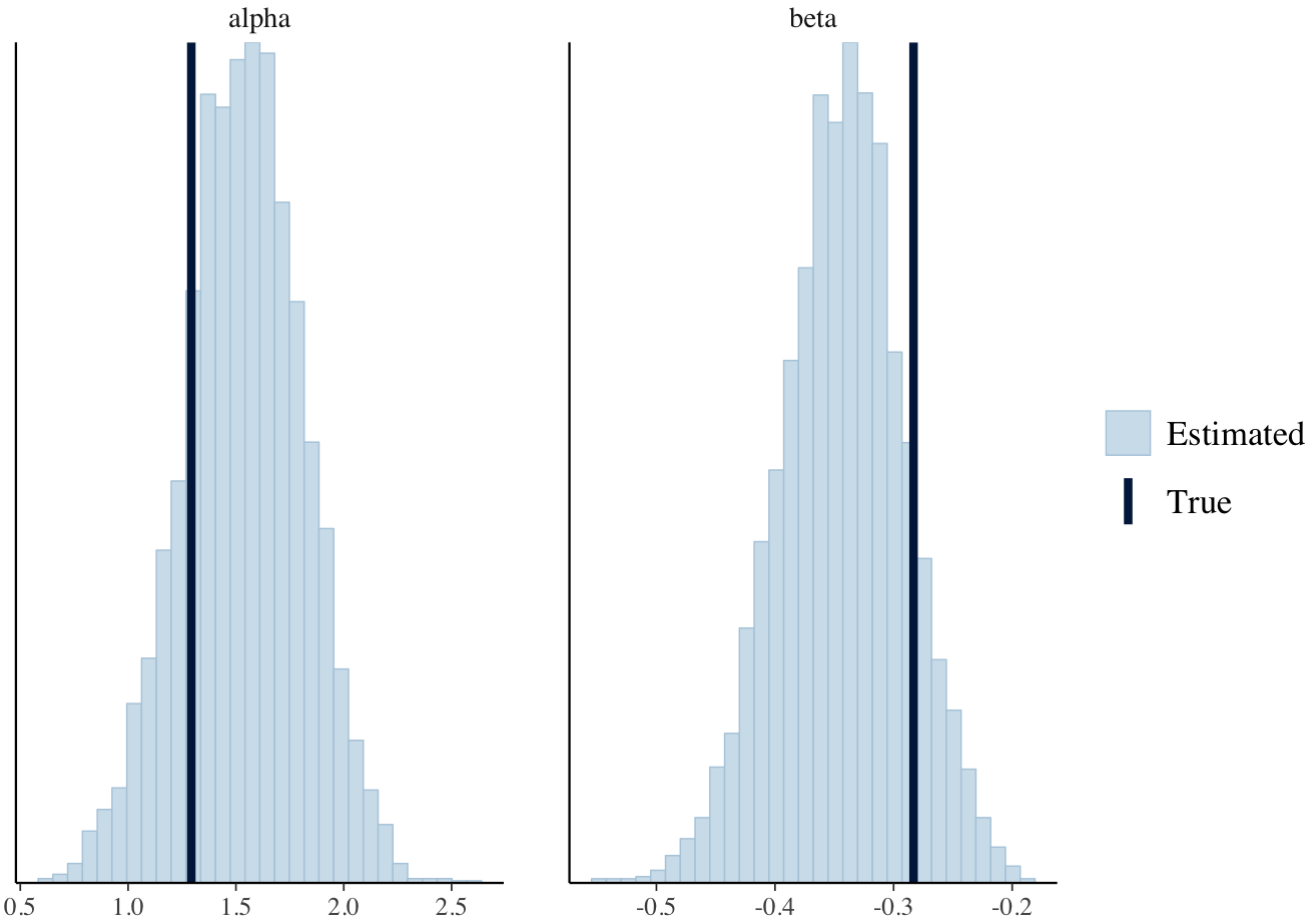
```
## - bayesplot theme set to bayesplot::theme_default()
```

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

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

```
# the plots cannot be knited in the markdown
mcmc_recover_hist(posterior_alpha_beta, true = true_alpha_beta)
```

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

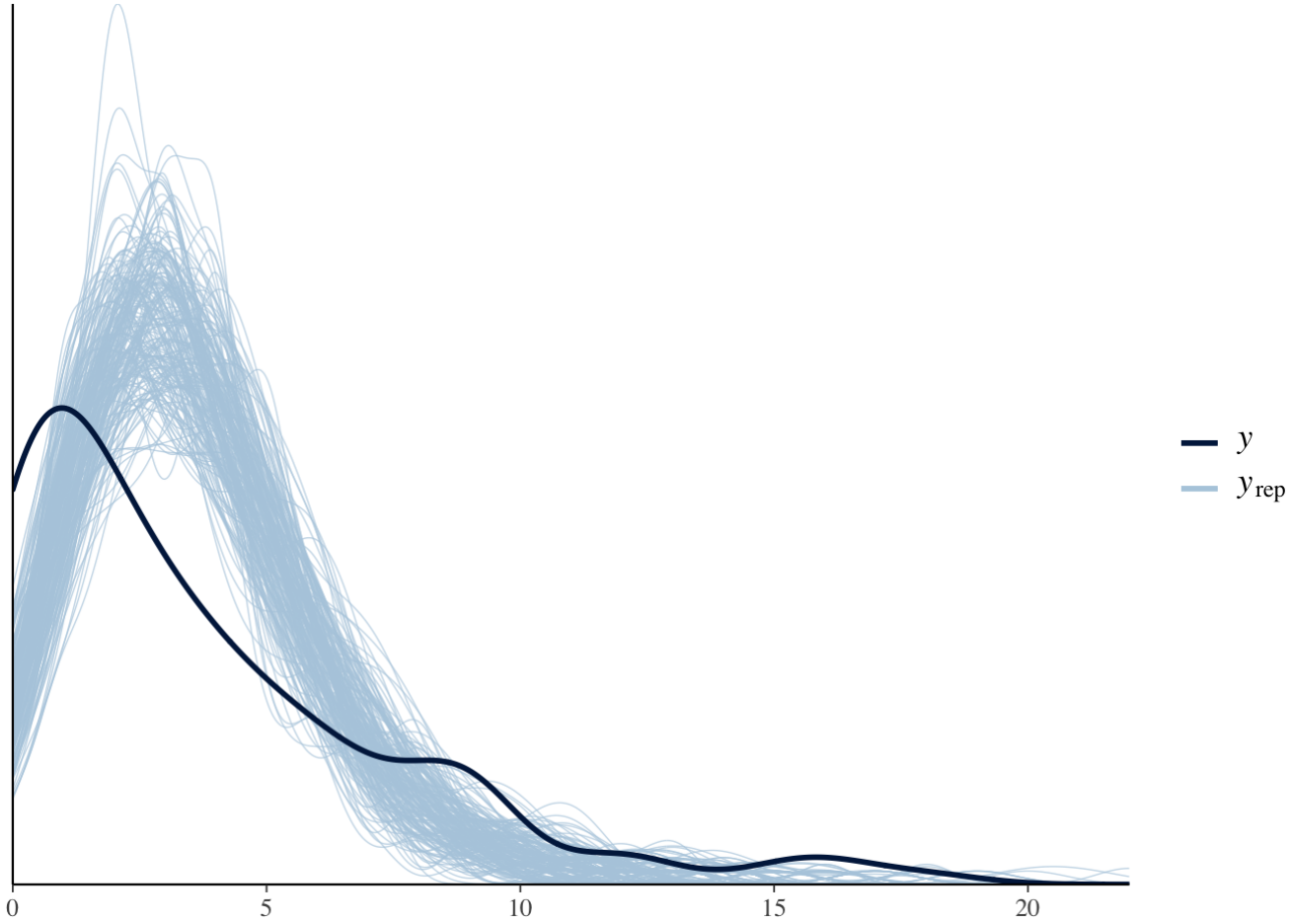


```
y_rep <- as.matrix(fit_model_P,pars = 'y_rep')
#ppc_dens_overlay(y = stan_dat_fake$complaints, yrep = y_rep[1:200, ])
#ppc_rootogram(stan_dat_fake$complaints, yrep = y_rep)
```

```
## fiting with the data supplied to us
stan_dat_simple <- list(N = nrow(pest_data), complaints = pest_data$complaints, traps =
 pest_data$traps)
fit_P_real_data <- sampling(comp_model_P, data = stan_dat_simple)
print(fit_P_real_data, pars = c('alpha','beta'))
```

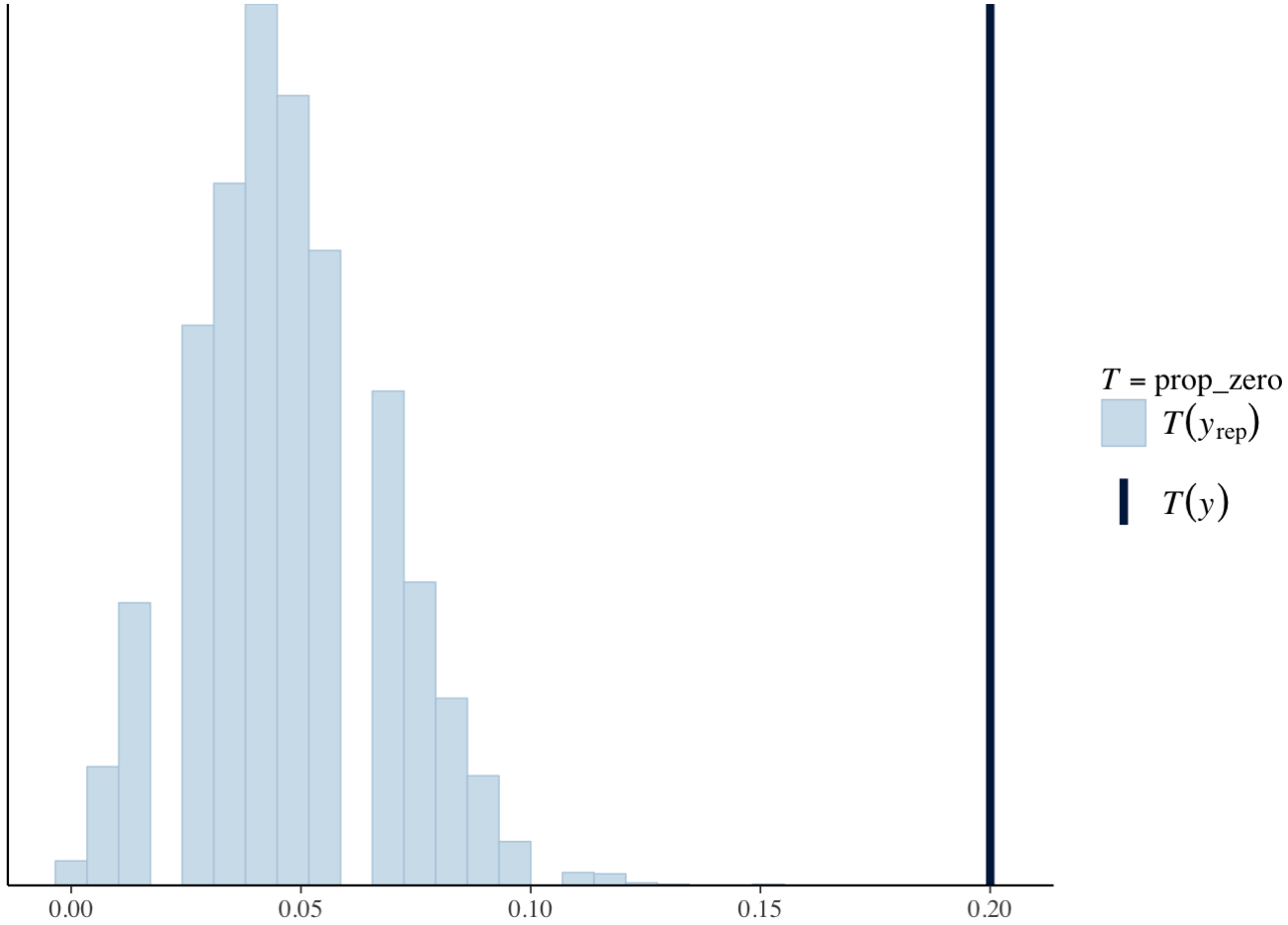
```
## Inference for Stan model: bait_possion.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha  2.58    0.01 0.16  2.27  2.47  2.57  2.68  2.89   838 1.01
## beta  -0.19    0.00 0.02 -0.24 -0.21 -0.19 -0.18 -0.15   852 1.01
##
## Samples were drawn using NUTS(diag_e) at Sun Oct 21 19:15:26 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

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

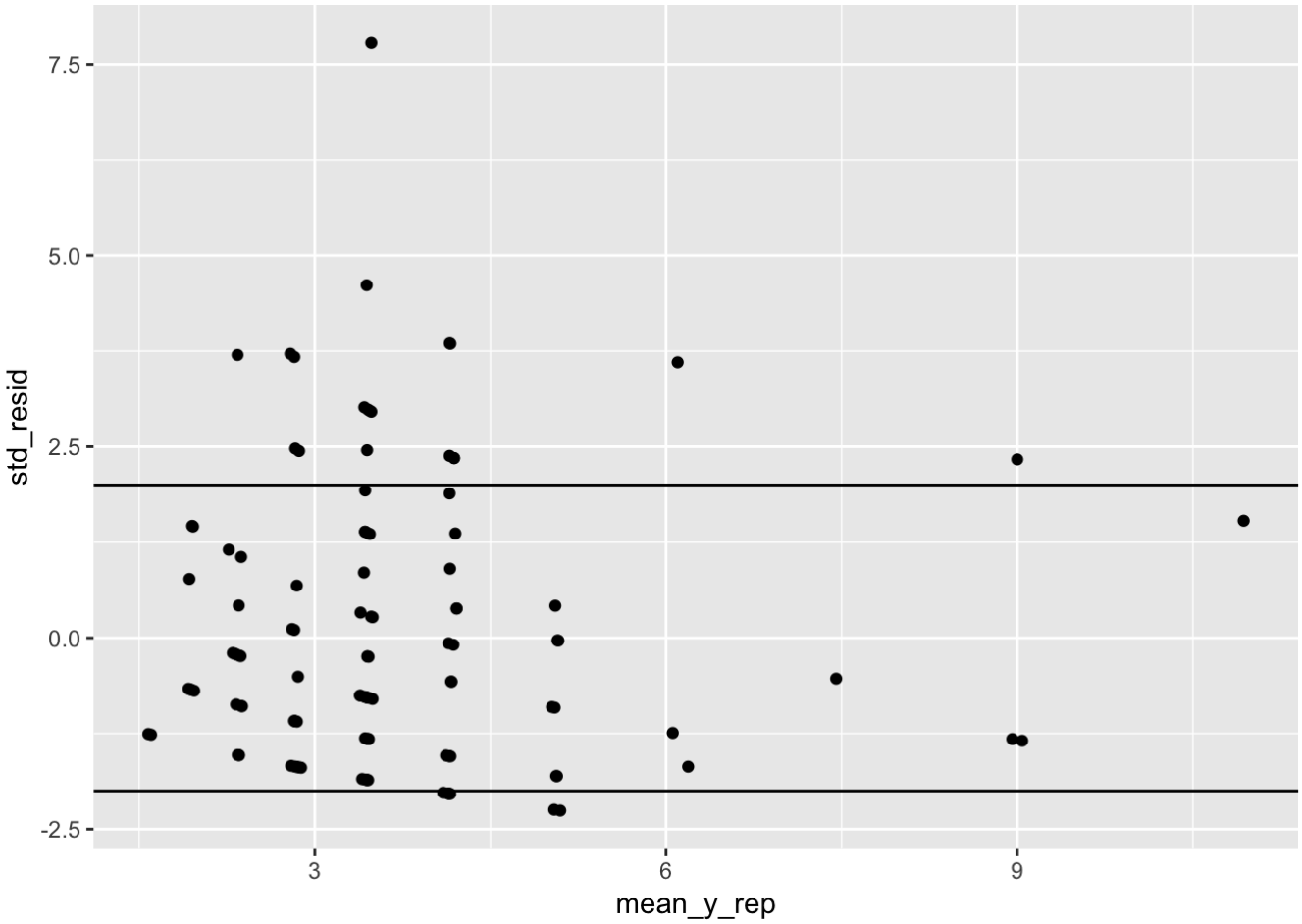



```
prop_zero <- function(x) mean(x == 0)
ppc_stat(y = stan_dat_simple$complaints, yrep = y_rep, stat = "prop_zero")
```

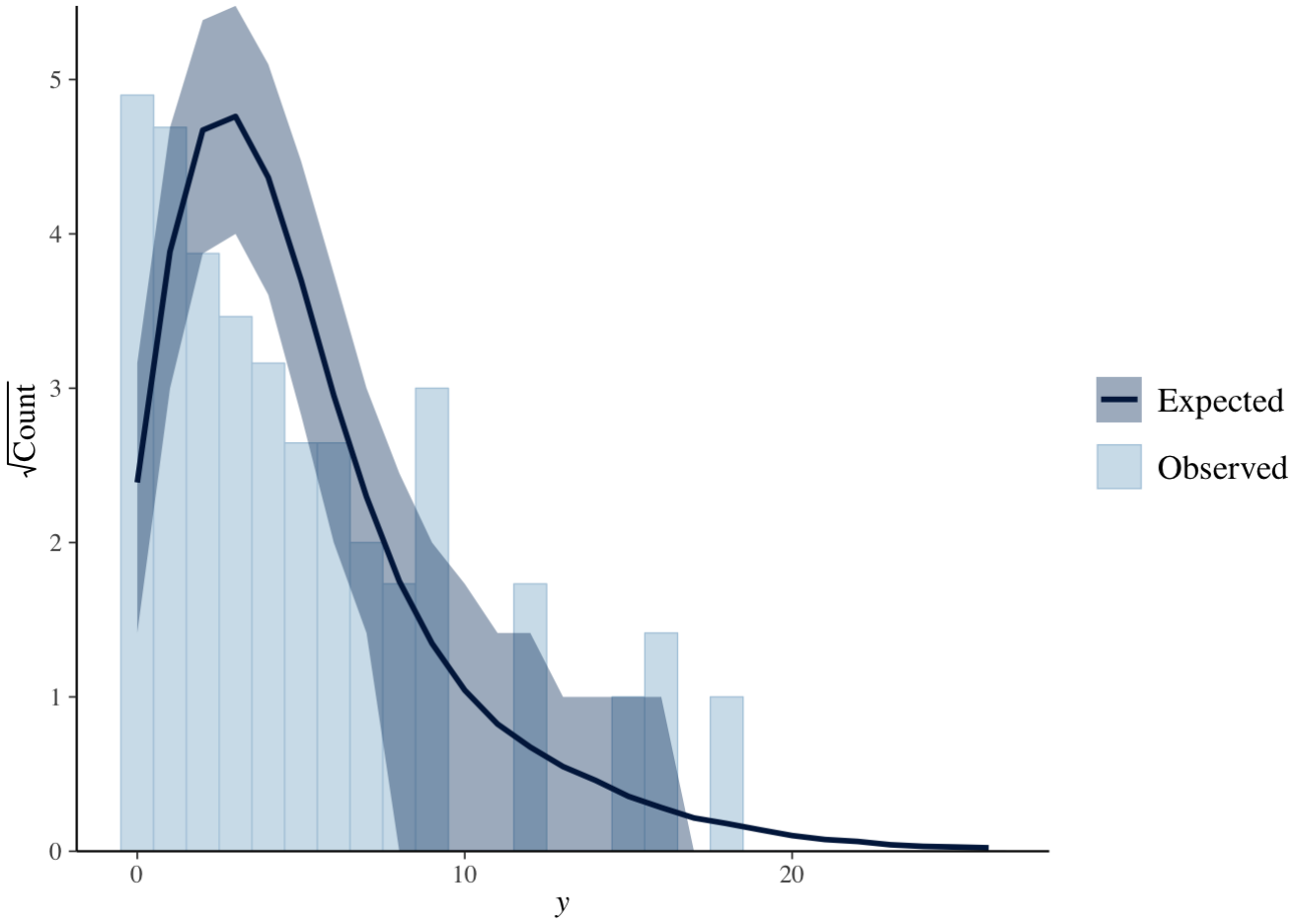
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



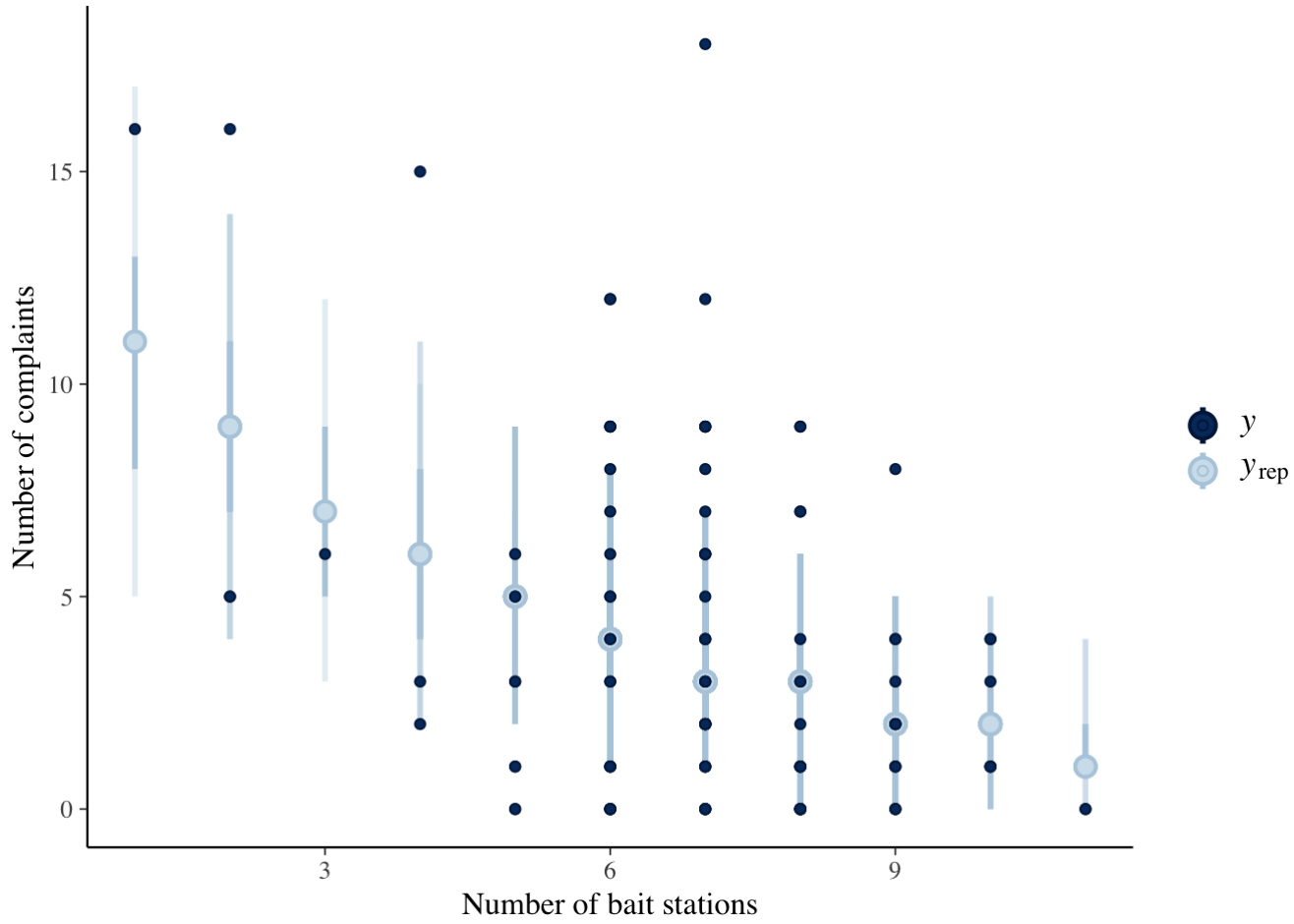

```
mean_y_rep <- colMeans(y_rep)
std_resid <- (stan_dat_simple$complaints - mean_y_rep) / sqrt(mean_y_rep)
qplot(mean_y_rep, std_resid) + hline_at(2) + hline_at(-2)
```



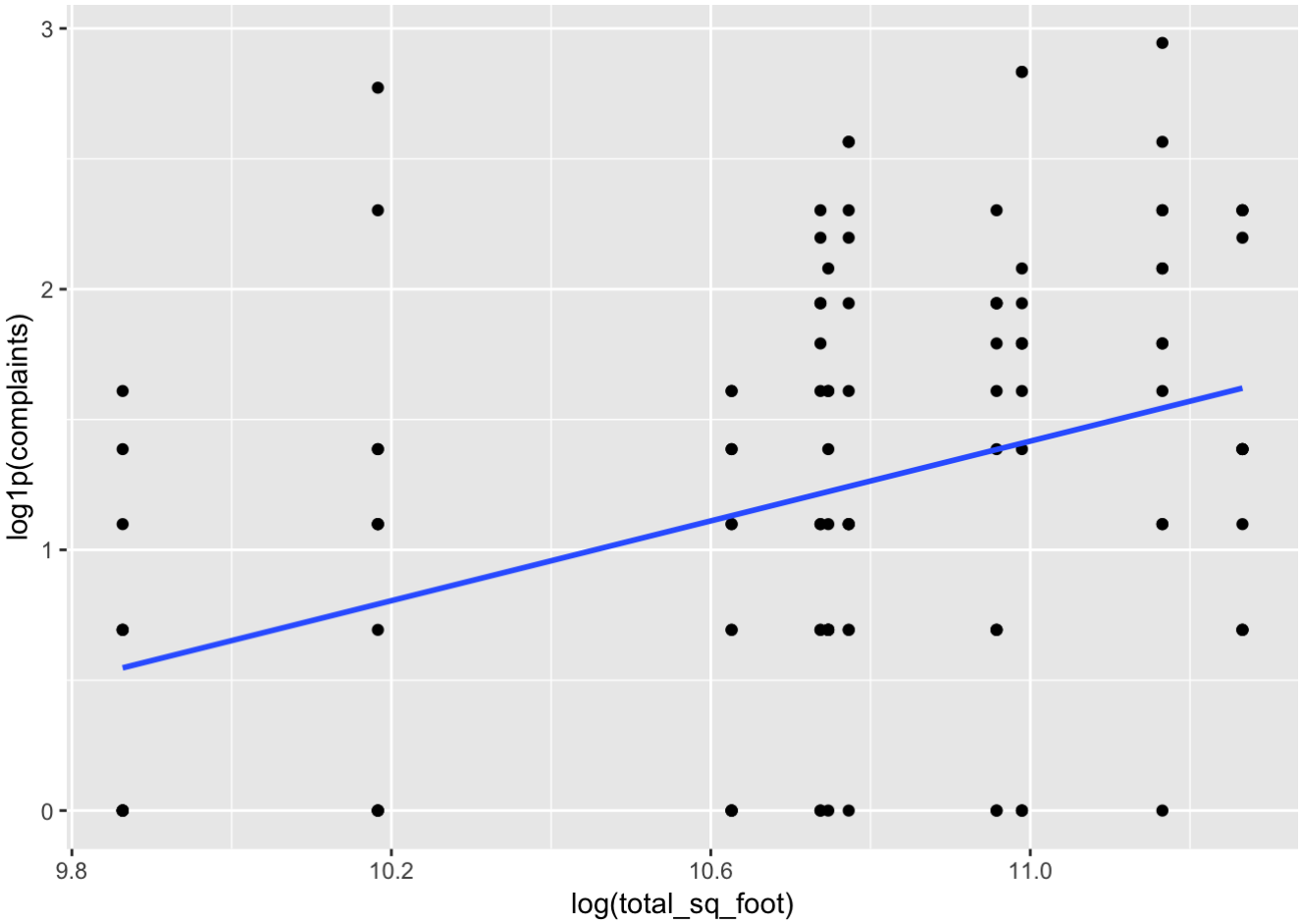

```
ppc_rootogram(stan_dat_simple$complaints, yrep = y_rep)
```




```
ppc_intervals(y = stan_dat_simple$complaints, yrep = y_rep, x = stan_dat_simple$traps) +

  labs(x = "Number of bait stations", y = "Number of complaints")
```




```
## expanding the model: multiple predictiors
ggplot(pest_data, aes(x = log(total_sq_foot), y = log1p(complaints))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

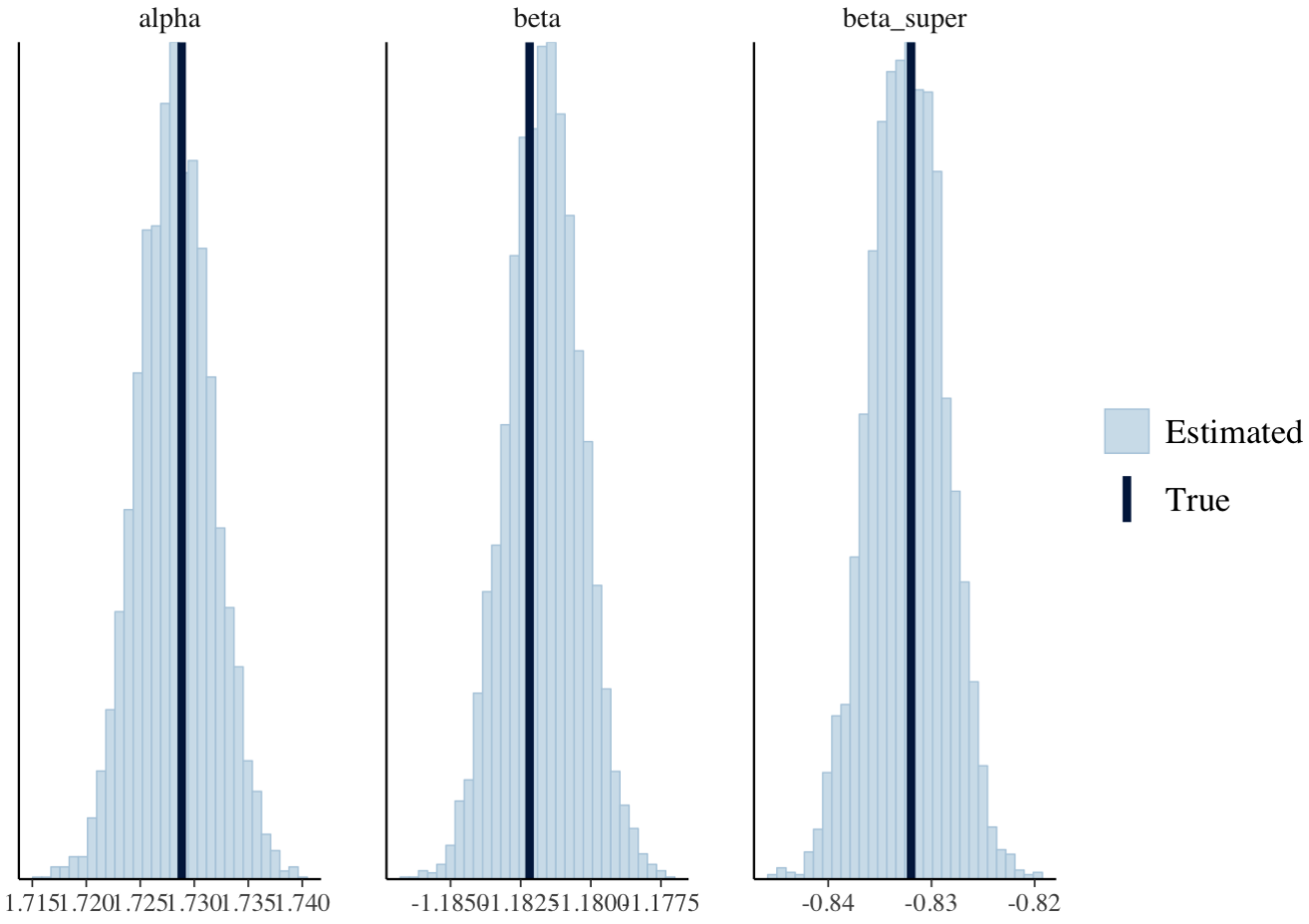



```
stan_dat_simple$log_sq_foot <- log(pest_data$total_sq_foot/1e4)
stan_dat_simple$live_in_super <- pest_data$live_in_super

## simulate fake data with multiple predictors
comp_dgp_multiple <- stan_model('multiple_poisson_regression_dgp.stan')
stan_dat_fake <- list(N=nrow(pest_data),
                      mean_traps = mean(pest_data$traps),
                      mean_live_in_super =mean(pest_data$live_in_super),
                      mean_log_sq_foot =  mean(log(pest_data$total_sq_foot)))
fitted_model_dgp <- sampling(comp_dgp_multiple, data = stan_dat_fake,chains = 1, cores =
 1, iter = 1, algorithm = 'Fixed_param', seed = 123)
```

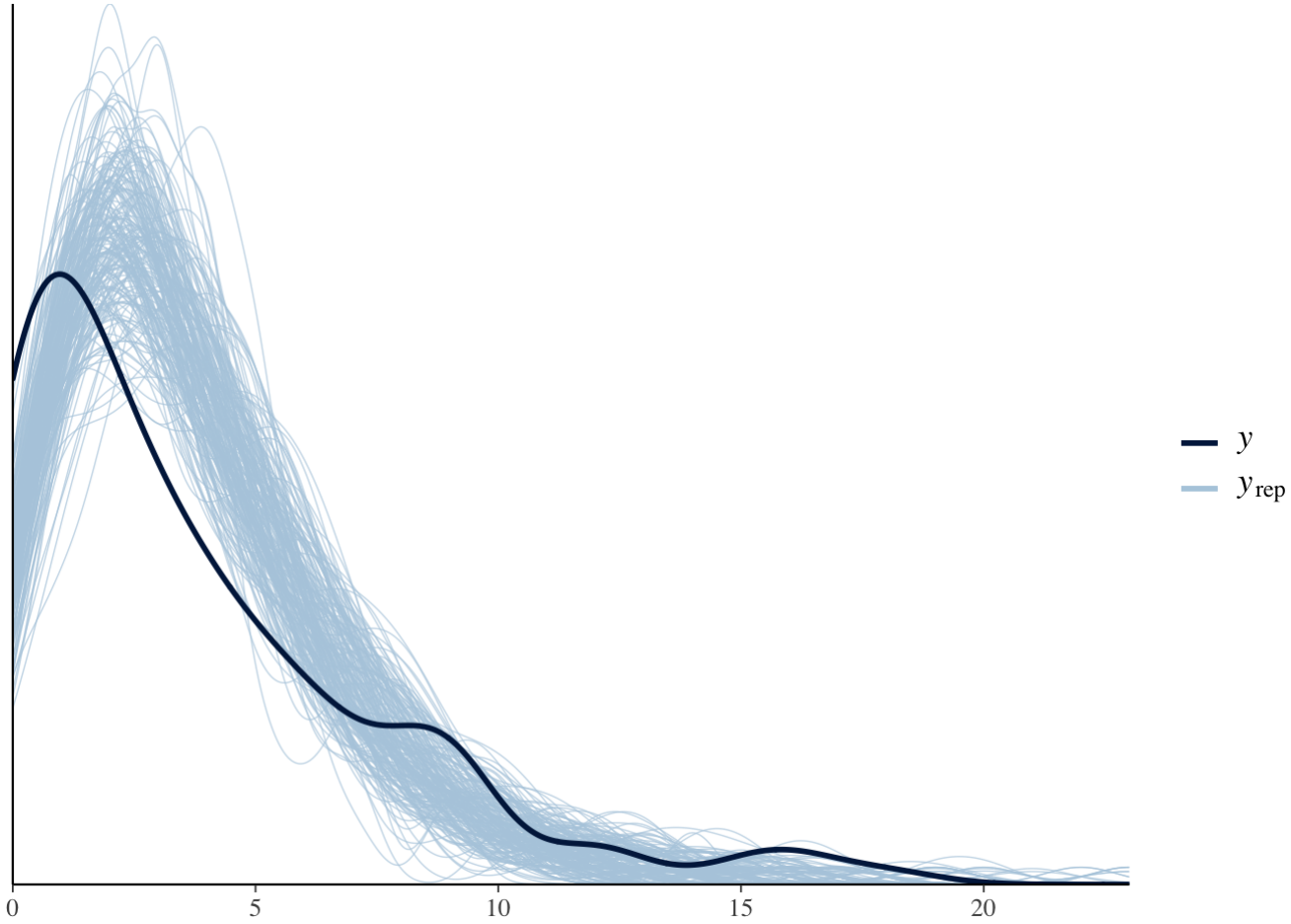
```
##
## SAMPLING FOR MODEL 'multiple_poisson_regression_dgp' NOW (CHAIN 1).
## Iteration: 1 / 1 [100%]  (Sampling)
##
##  Elapsed Time: 0 seconds (Warm-up)
##                5.8e-05 seconds (Sampling)
##                5.8e-05 seconds (Total)
```

```
sims_dgp <- extract(fitted_model_dgp)
stan_dat_fake <- list(N = nrow(pest_data), log_sq_foot = sims_dgp$log_sq_foot[1, ],
  live_in_super = sims_dgp$live_in_super[1, ], traps = sims_dgp$traps[1, ], complaints =
 sims_dgp$complaints[1, ])

comp_model_P_mult <- stan_model('multiple_poisson_regression.stan')
fit_model_P_mult <- sampling(comp_model_P_mult, data = stan_dat_fake, chains = 4, cores
 = 4)

posterior_alpha_beta <- as.matrix(fit_model_P_mult, pars = c('alpha','beta','beta_super'
))
true_alpha_beta <- c(sims_dgp$alpha,sims_dgp$beta,sims_dgp$beta_super)
mcmc_recover_hist(posterior_alpha_beta, true = true_alpha_beta)
```

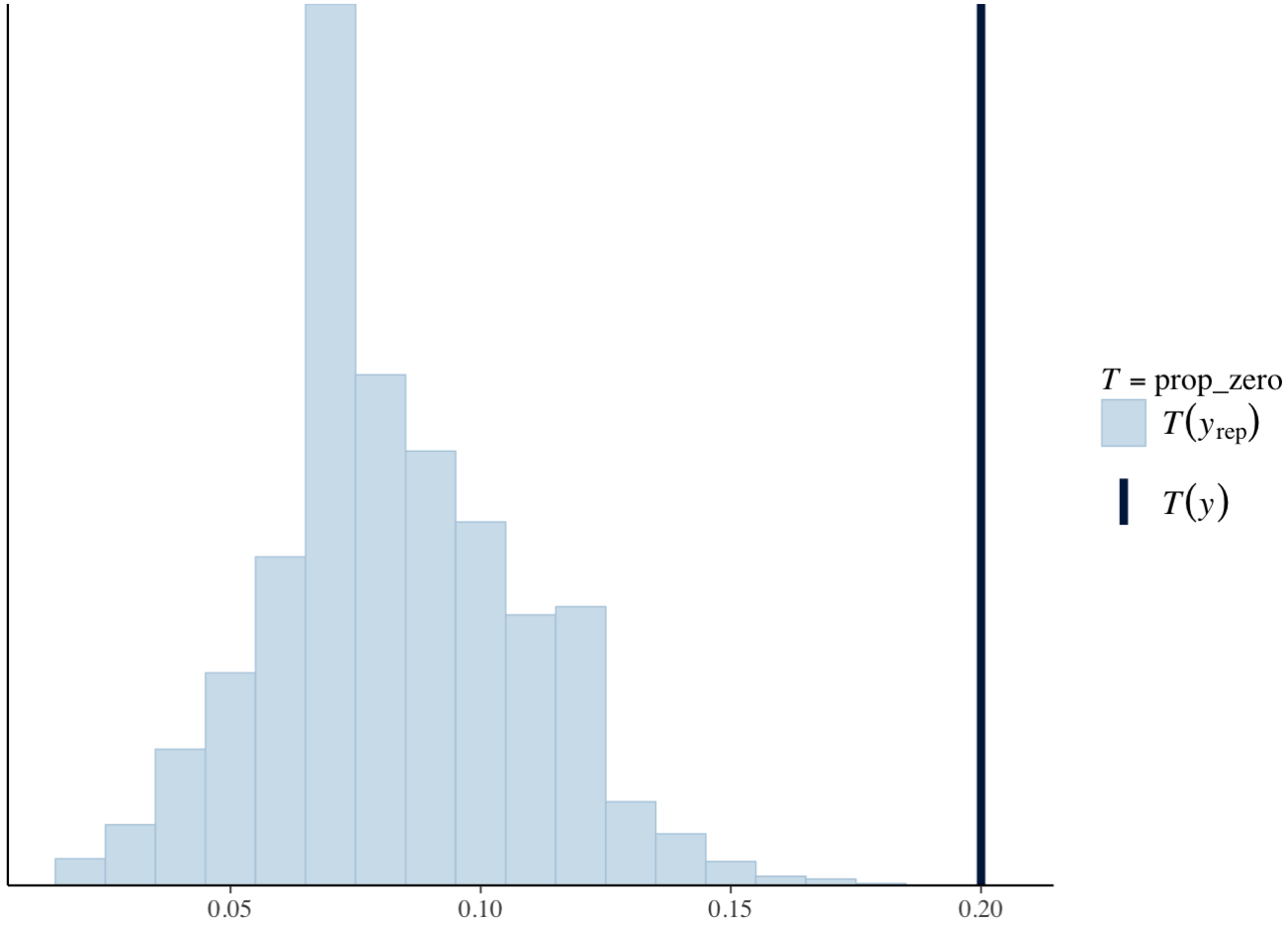
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



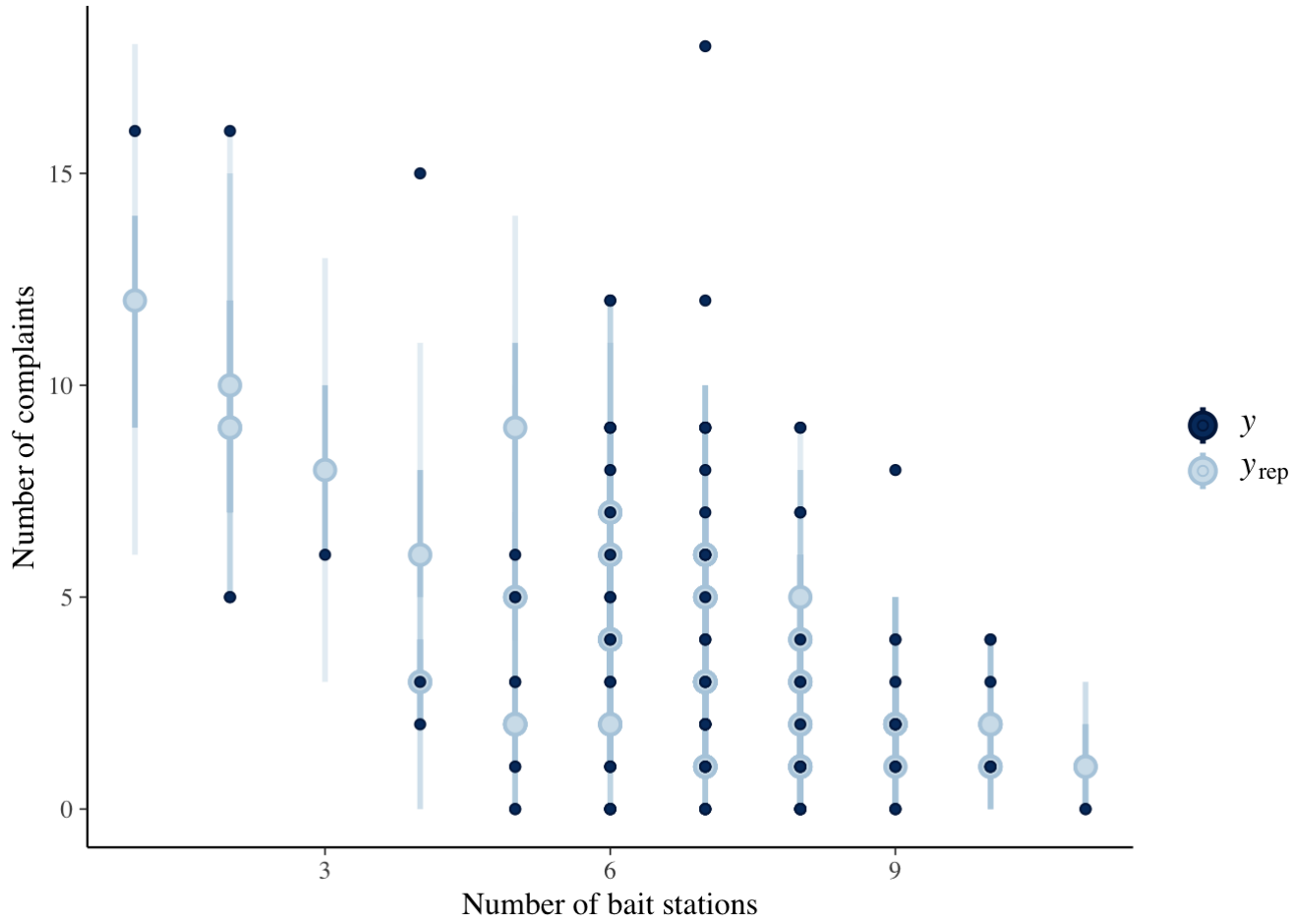

```
## Fit the data given to us
fit_model_P_mult_real <- sampling(comp_model_P_mult, data = stan_dat_simple)
y_rep <- as.matrix(fit_model_P_mult_real, pars = "y_rep")
ppc_dens_overlay(stan_dat_simple$complaints, y_rep[1:200,])
```




```
prop_zero <- function(x) mean(x == 0)
ppc_stat(y = stan_dat_simple$complaints, yrep = y_rep, stat = "prop_zero", binwidth = 0.
01)
```




```
ppc_intervals(y = stan_dat_simple$complaints, yrep = y_rep, x = stan_dat_simple$traps) +

  labs(x = "Number of bait stations", y = "Number of complaints")
```

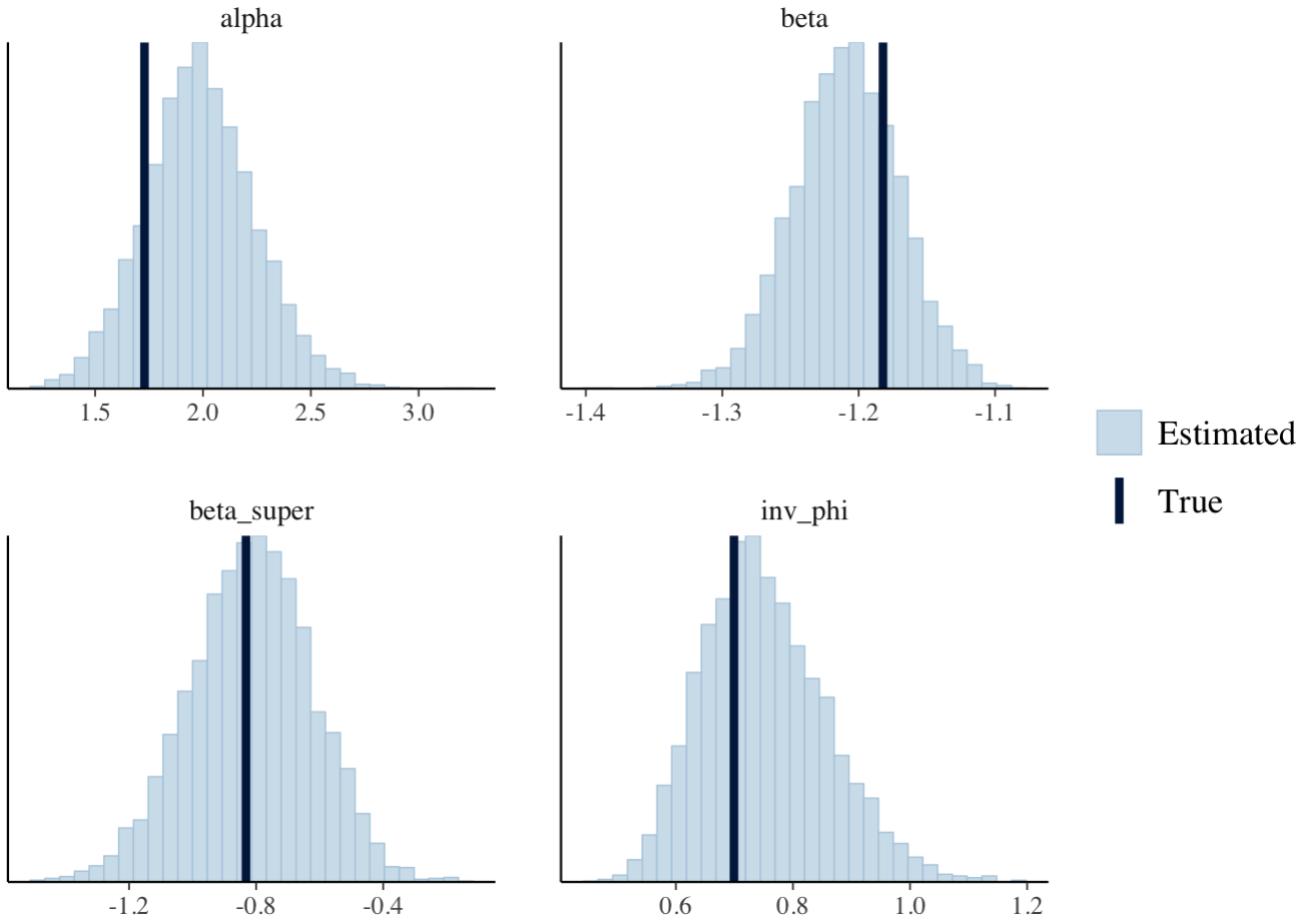



```
## Modeling count data with the negative binomial distribution
comp_dgp_multiple_NB <- stan_model('multiple_NB_regression_dgp.stan')
stan_dat_fake <- list(N=nrow(pest_data),
                      mean_traps = mean(pest_data$traps),
                      mean_live_in_super =mean(pest_data$live_in_super),
                      mean_log_sq_foot =  mean(log(pest_data$total_sq_foot)))
fitted_model_dgp_NB <- sampling(comp_dgp_multiple_NB, data = stan_dat_fake,
  chains = 1, cores = 1, iter = 1, algorithm = 'Fixed_param', seed = 123)
```

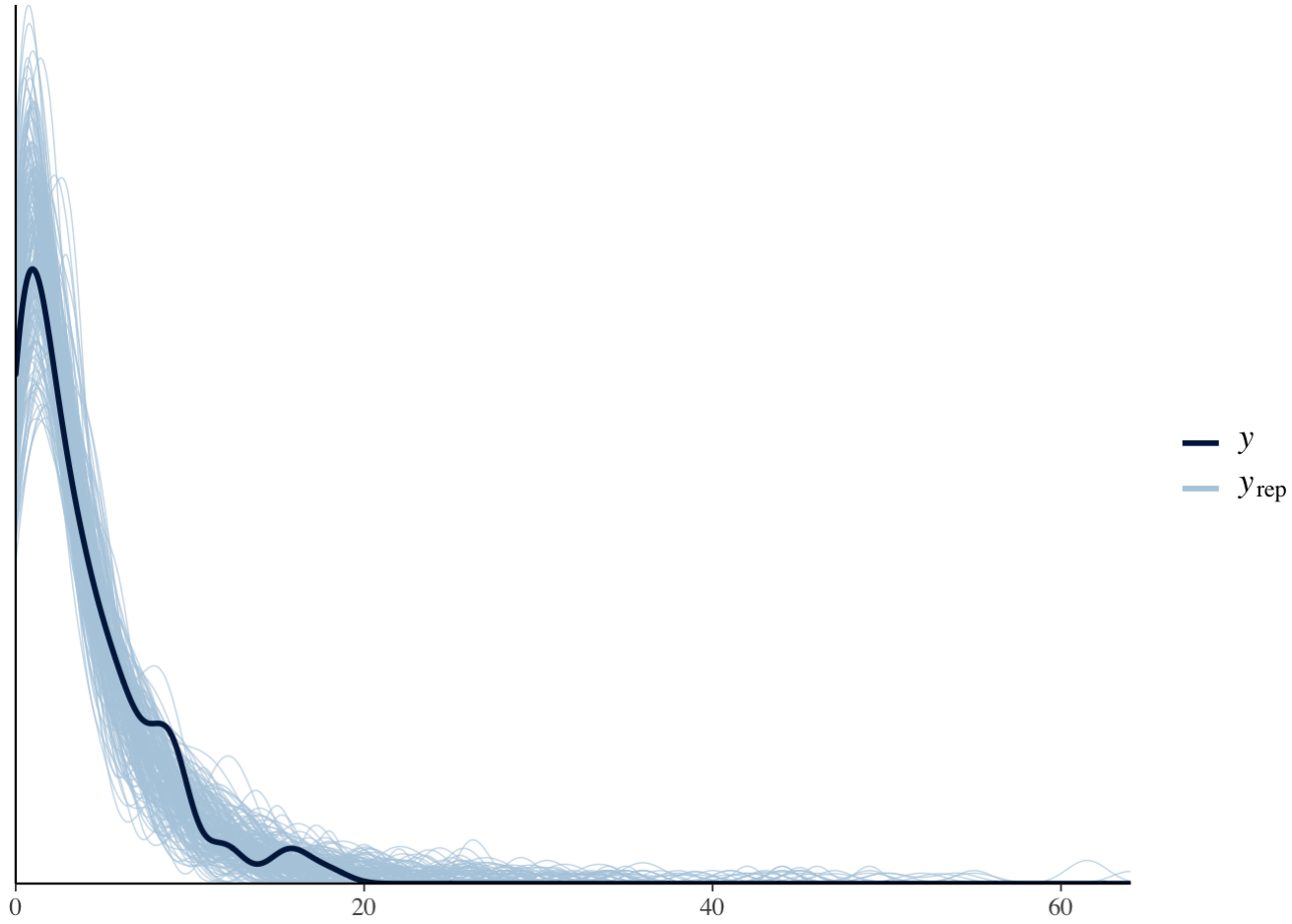
```
##
## SAMPLING FOR MODEL 'multiple_NB_regression_dgp' NOW (CHAIN 1).
## Iteration: 1 / 1 [100%]  (Sampling)
##
##  Elapsed Time: 0 seconds (Warm-up)
##                6.6e-05 seconds (Sampling)
##                6.6e-05 seconds (Total)
```

```
sims_dgp_NB <- extract(fitted_model_dgp_NB)

stan_dat_fake_NB <- list(N = nrow(pest_data),
                         log_sq_foot = sims_dgp_NB$log_sq_foot[1, ],
                         live_in_super = sims_dgp_NB$live_in_super[1, ],
                         traps = sims_dgp_NB$traps[1, ],
                         complaints = sims_dgp_NB$complaints[1, ]
)

comp_model_NB <- stan_model('multiple_NB_regression.stan')

fitted_model_NB <- sampling(comp_model_NB, data = stan_dat_fake_NB,
                            chains = 4, cores = 4)

posterior_alpha_beta_NB <- as.matrix(fitted_model_NB, pars = c('alpha', 'beta', 'beta_su
per', 'inv_phi'))
true_alpha_beta_NB <- c(sims_dgp_NB$alpha, sims_dgp_NB$beta, sims_dgp_NB$beta_super, sim
s_dgp_NB$inv_phi)
mcmc_recover_hist(posterior_alpha_beta_NB, true = true_alpha_beta_NB)
```

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

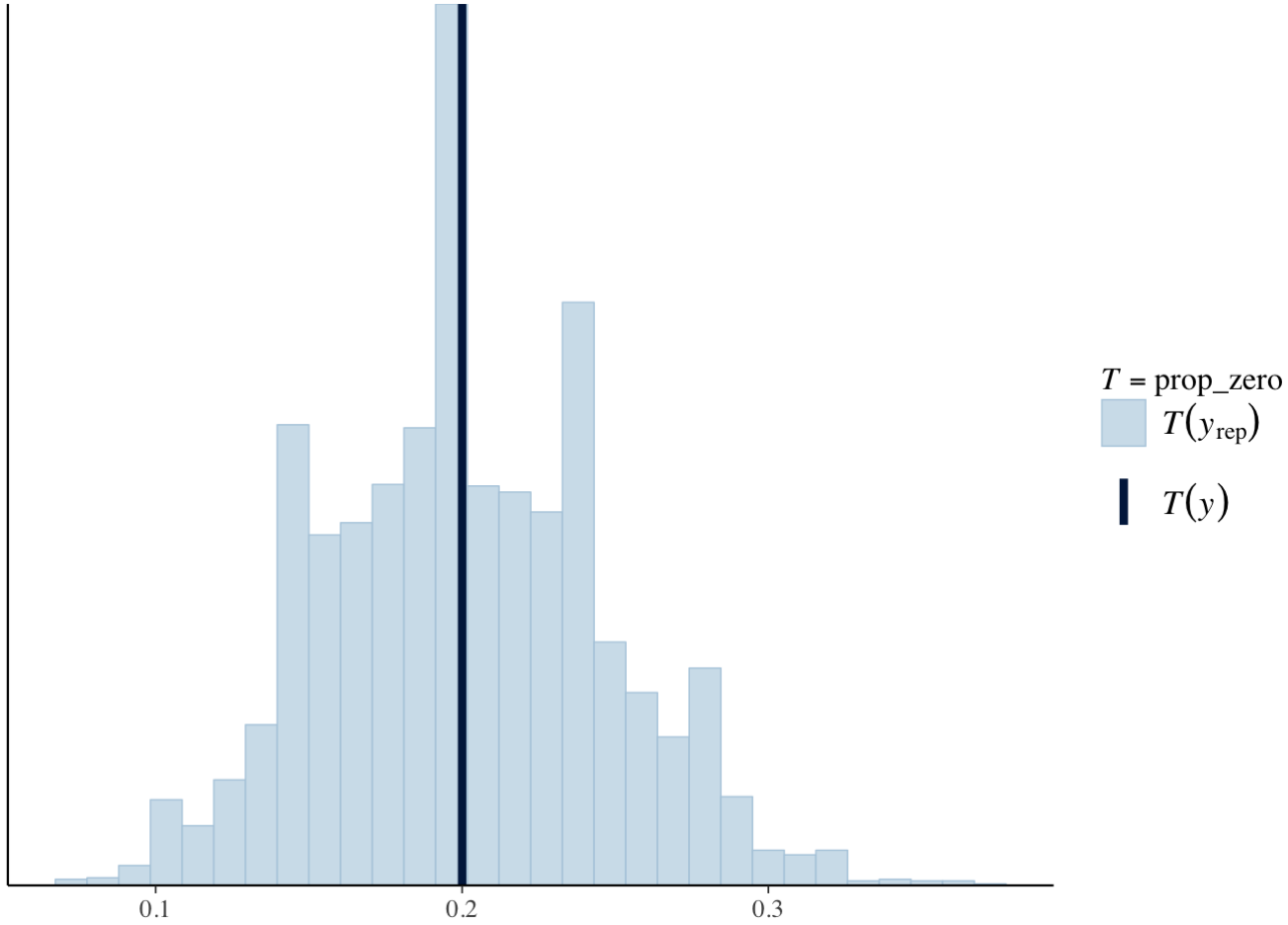


```
## fiting to the given data and checking the fit
fitted_model_NB <- sampling(comp_model_NB, data = stan_dat_simple)
sims_NB <- extract(fitted_model_NB)

y_rep <- sims_NB$y_rep
ppc_dens_overlay(stan_dat_simple$complaints, y_rep[1:200,])
```

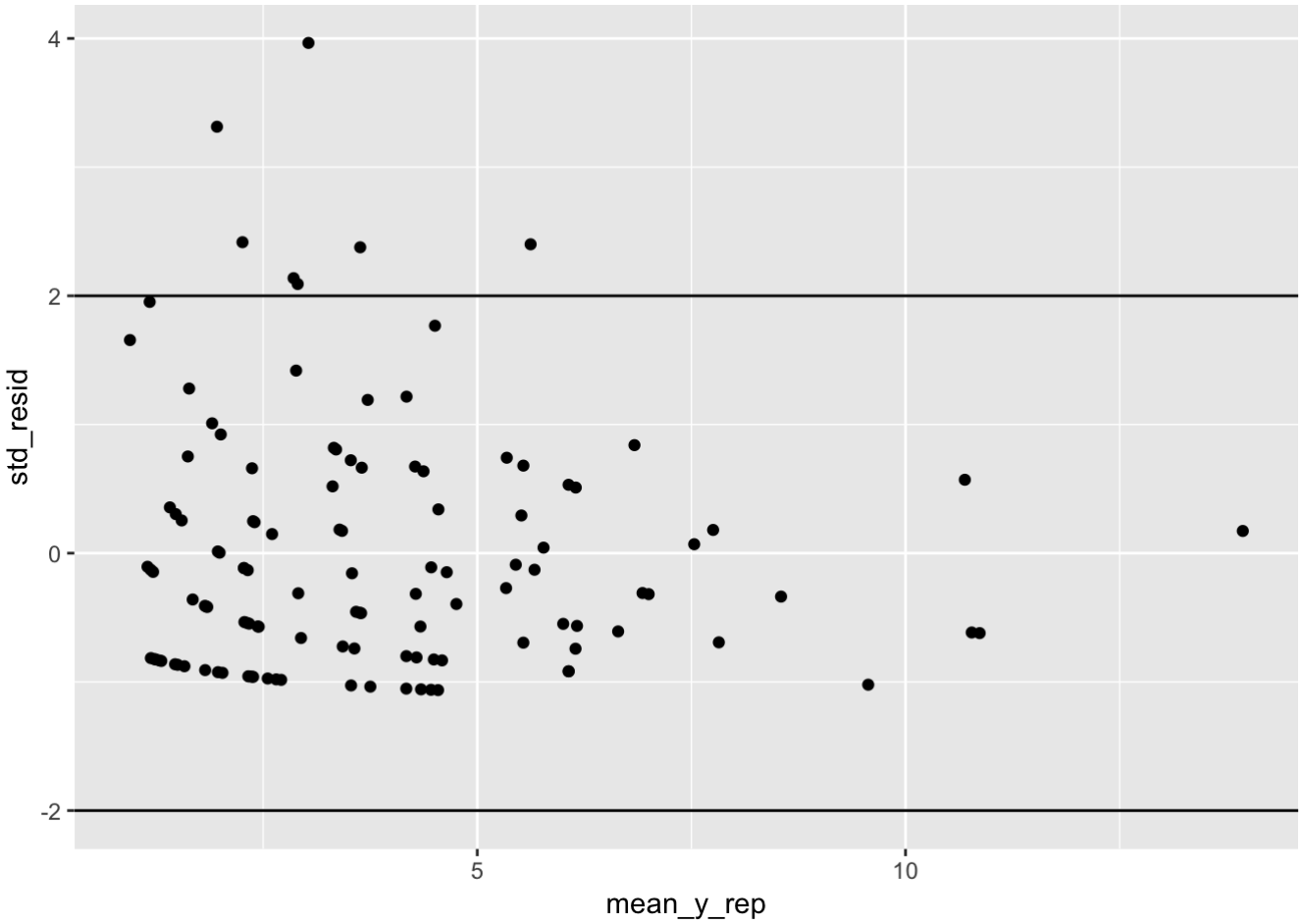



```
ppc_stat(y = stan_dat_simple$complaints, yrep = y_rep, stat = "prop_zero")
```

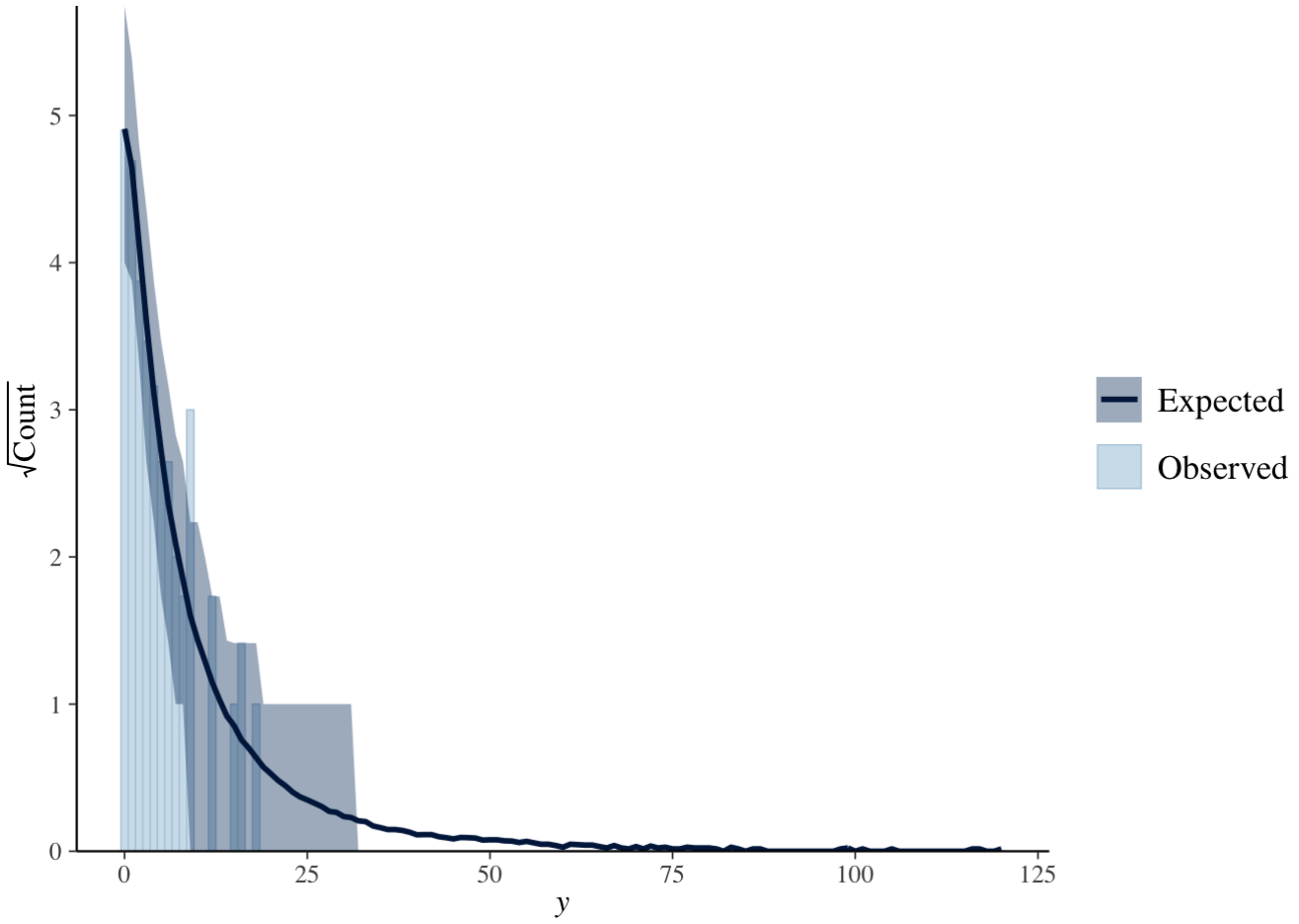
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



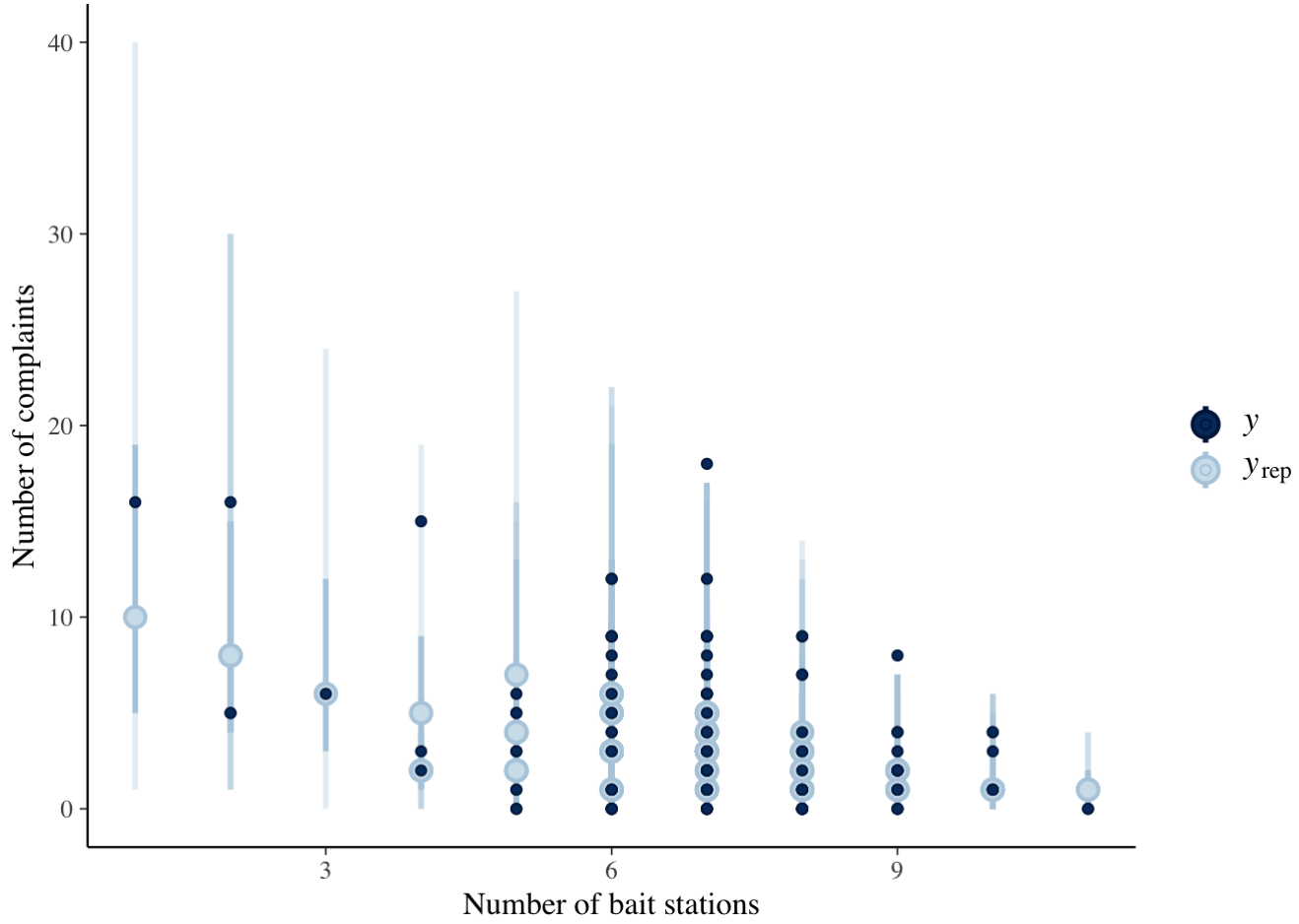

```
mean_inv_phi <- mean(sims_NB$inv_phi)
mean_y_rep <- colMeans(y_rep)
std_resid <- (stan_dat_simple$complaints - mean_y_rep) / sqrt(mean_y_rep + mean_y_rep^2*
mean_inv_phi)
qplot(mean_y_rep, std_resid) + hline_at(2) + hline_at(-2)
```



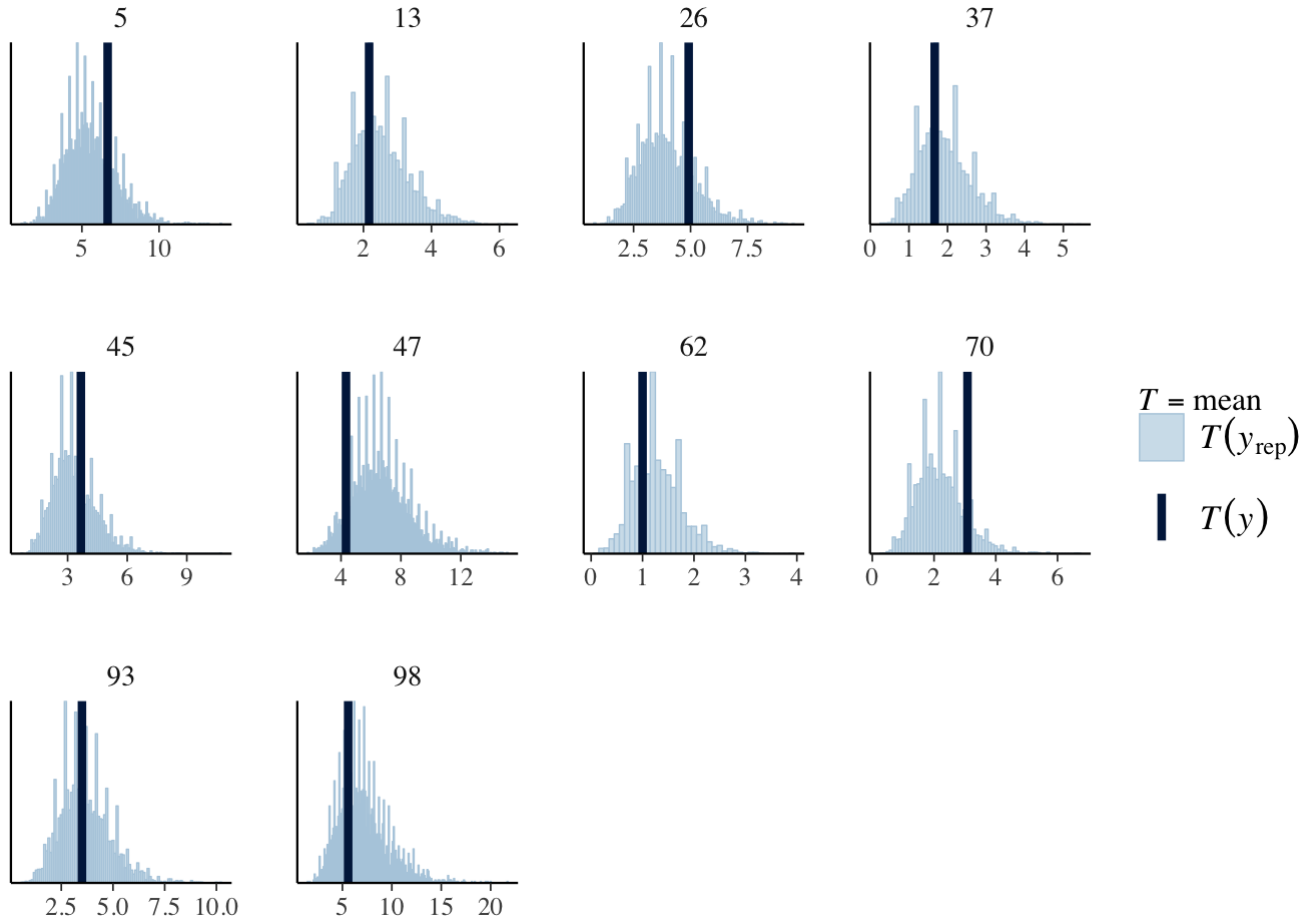

```
ppc_rootogram(stan_dat_simple$complaints, yrep = y_rep)
```




```
ppc_intervals(y = stan_dat_simple$complaints, yrep = y_rep, x = stan_dat_simple$traps) +

  labs(x = "Number of bait stations", y = "Number of complaints")
```




```
ppc_stat_grouped(y = stan_dat_simple$complaints, yrep = y_rep, group = pest_data$buildin
g_id,
  stat = 'mean', binwidth = 0.1)
```



2. Write down a two sentence plan for your final project

A data set about customers' information (including sex, income, address and so on) as well as log records of website visit would be analyzed with the hierarchical model would be used to predicte the potential (latent variable) and its risk of treatment (latent variable). The data is from a start up company and they are online shopping website.

Since we have log visit record, time-varying effects and structured priors many need to be discussed.

3. What sort of Bayesian modeling do you plan on performing?

hierarchical modeling based on multivariable regression.

4. How might you check your model? (e.g. posterior predictive checks c.f. Chapter 6 of BDA)

First, graphical posterior predictive checks would be done. plots including residual plots and other statistical inference plots would be drawn to get the big picture of the check.

Second, the posterior predictive checks would be done. The basic statistics including mean, standard deviation, max value and min value would be choosen. And we probabily would choose other quantiles for differenct model questions like check whether it is indepent.

Second, marginal predictive checks would be done.

5. How might you evaluate your model? (e.g. cross-validation c.f. Chapter 7 of BDA)

First, the cross-validation would be done. Based on the computation cost, we may choose leave one out cross-validation or others like 10 forld cross-validation.

Second, we would try to use the WAIC which is under the bayesian framework and try to compare the results with the cross-validation and discuss it.

6. How might you collect data? (e.g. ensuring ignorability)

The data have been collected from the company directly. But there would have some missing data problem. These data are observational data we need to use as many covariable as possible to make the design as random as possible.