binomial_baseline

Jongwoo Choi

2018-12-05

```r
file <-  'core.txt'
data <- read_delim(file = file, delim = '|')

# Sample the data
pct = 1
# pct = 0.1
# pct = 0.01
set.seed(seed = 42)
sample_size = round(pct * nrow(data))
sample <- sample(x = nrow(data), size = sample_size, replace = F)
data = data[sample, ]
## Selecting the relevant columns for the analysis
data_sub <- data %>% dplyr::select(
  state,
  city,
  county,
  zip,
  asset_market_value,
  mar_2_app,
  appraisal_value,
  app_2_inc,
  client_income,
  mar_2_inc,
  age,
  sex_F,
  condition_U,
  y,
  y2)
summary(data_sub)
```

```
    state               city              county               zip       
 Length:30499       Length:30499       Length:30499       Min.   : 1000  
 Class :character   Class :character   Class :character   1st Qu.:32680  
 Mode  :character   Mode  :character   Mode  :character   Median :55295  
                                                          Mean   :54236  
                                                          3rd Qu.:76148  
                                                          Max.   :99900  
 asset_market_value   mar_2_app    appraisal_value     app_2_inc   
 Min.   : 120000    Min.   : 0.9   Min.   :  79521   Min.   :0.00  
 1st Qu.: 350000    1st Qu.: 1.1   1st Qu.: 302908   1st Qu.:0.09  
 Median : 398000    Median : 1.2   Median : 320052   Median :0.10  
 Mean   : 491311    Mean   : 1.3   Mean   : 371064   Mean   :0.10  
 3rd Qu.: 463000    3rd Qu.: 1.4   3rd Qu.: 348289   3rd Qu.:0.11  
 Max.   :4519000    Max.   :21.8   Max.   :1654602   Max.   :0.37  
 client_income    mar_2_inc          age          sex_F       condition_U
 Min.   : 144   Min.   :0.03   Min.   :18   Min.   :0.00   Min.   :0.0  
 1st Qu.: 284   1st Qu.:0.11   1st Qu.:27   1st Qu.:0.00   1st Qu.:0.0  
```

```
 Median : 311   Median :0.12   Median :32   Median :0.00   Median :0.0
 Mean   : 410   Mean   :0.13   Mean   :34   Mean   :0.31   Mean   :0.4
 3rd Qu.: 342   3rd Qu.:0.14   3rd Qu.:40   3rd Qu.:1.00   3rd Qu.:1.0
 Max.   :1887   Max.   :1.09   Max.   :65   Max.   :1.00   Max.   :1.0
       y              y2
 Min.   :0.00   Min.   :0.00
 1st Qu.:0.00   1st Qu.:0.00
 Median :0.00   Median :0.00
 Mean   :0.06   Mean   :0.03
 3rd Qu.:0.00   3rd Qu.:0.00
 Max.   :1.00   Max.   :1.00
```

```r
## Group data by state and define the IDs
state_summary <- data_sub %>%
  dplyr::select(state,
                client_income,
                appraisal_value,
                asset_market_value) %>%
  group_by(state) %>%
  summarize(n_state = n(),
            income_mean_state = mean(client_income),
            appraisal_mean_state = mean(appraisal_value),
            market_mean_state = mean(asset_market_value)) %>%
  arrange(desc(n_state)) %>%
  ungroup()
state_summary$ID_state = seq.int(nrow(state_summary))


## Group data by city and define the IDs
city_summary <- data_sub %>%
  dplyr::select(city, state,
                client_income,
                appraisal_value,
                asset_market_value,
                mar_2_inc,
                mar_2_app,
                app_2_inc,
                age,
                y,
                y2) %>%
  group_by(city, state) %>%
  summarize(n_city = n(),
            income_mean_city = mean(client_income),
            appraisal_mean_city = mean(appraisal_value),
            market_mean_city = mean(asset_market_value),
            mar_2_inc_mean_city = mean(mar_2_inc),
            mar_2_app_mean_city = mean(mar_2_app),
            app_2_inc_mean_city = mean(app_2_inc),
            age_mean_city = mean(age),
            sum_y = sum(y),
            sum_y2 = sum(y2)) %>%
  arrange(desc(n_city)) %>%
  ungroup()
```

```r
## Merge back into data
city_summary <- city_summary %>%
  inner_join(y = state_summary[c('ID_state', 'state')], by = 'state')


## Rescaling
inputs <- city_summary %>%
  mutate(
    market_state_city = (log(market_mean_city) - mean(log(market_mean_city))) /
      sd(log(market_mean_city)),

    income_state_city = (log(appraisal_mean_city) - mean(log(appraisal_mean_city))) /
      sd(log(appraisal_mean_city)),

    appraisal_state_city = (log(appraisal_mean_city) -
                            mean(log(appraisal_mean_city))) /
      sd(log(appraisal_mean_city)),

    mar_2_inc_city = (mar_2_inc_mean_city - mean(mar_2_inc_mean_city)) / sd(mar_2_inc_mean_city),

    app_2_inc_city = (app_2_inc_mean_city - mean(app_2_inc_mean_city)) / sd(app_2_inc_mean_city),

    mar_2_app_city = (mar_2_app_mean_city - mean(mar_2_app_mean_city)) / sd(mar_2_app_mean_city),

    age_city = (age_mean_city - mean(age_mean_city)) / sd(age_mean_city)) %>%
  dplyr::select(
    market_state_city,
    income_state_city,
    appraisal_state_city,
    mar_2_inc_city,
    app_2_inc_city,
    mar_2_app_city,
    age_city,
    ID_state,
    n_city,
    sum_y,
    sum_y2
  )
```

Baseline Model: Binomial

Our baseline model is simple binomial regression model. We give weak cauchy priors on the coefficient parameter β the intercept a. In the city level, we assume that the number of individual records in city i is n_i.

$$a \sim \textsf{Cauchy}(0,\ 10)$$

$$\beta \sim \textsf{Cauchy}(0,\ 2.5)$$

$$y_i \sim \textsf{Binomial}(n_i,\ logit^{-1}(a + \beta \cdot X_i))$$

The binomial baseline model is given by:

```r
baseline_model_binom=stan_model('binomial_baseline_city.stan')
```

```r
print_file('binomial_baseline_city.stan')
```

```
// baseline model: city level
data {
  int<lower=1> N_train;                 // number of record, train
  int<lower=1> N_test;                  // number of record, test
  int<lower=1> D;                       // number of covariates
  matrix[N_train, D] X_train;           // train data
  matrix[N_test, D] X_test;             // test data
  int<lower=1> n_city_train[N_train];   // number of record for city n, train
  int<lower=1> n_city_test[N_test];     // number of record for city n, test
  int<lower=0> y_train[N_train];        // y train
}
parameters {
  real a;                               // include intercept
  vector[D] beta;                       // regression coefficient vector
}
transformed parameters {
  vector[N_train] eta;
  eta = a + X_train*beta;               // probability in binomial regression
}
model {
  a ~ cauchy(0, 10);                    // Cauchy prior
  beta ~ cauchy(0, 2.5);                // Cauchy prior
  y_train ~ binomial_logit(n_city_train, eta);  // binomial model
}
generated quantities{
  int<lower =0> y_rep[N_train];
  int<lower =0> y_rep_cv[N_test];
  for (i in 1:N_train){
    y_rep[i] = binomial_rng(n_city_train[i], inv_logit(eta[i]));
  }
  for (i in 1:N_test){
    y_rep_cv[i] = binomial_rng(n_city_test[i], inv_logit(eta[i]));
  }
}
```

Parameters recovered

We generate the fake data to simulate the model.

```r
a <- rcauchy(1, 0, 10)
beta <- rcauchy(7, 0, 2.5)
X <- inputs %>% dplyr::select(-ID_state,-n_city,-sum_y, -sum_y2)
n_city <- inputs$n_city
N = nrow(X)
D = ncol(X)

y_fake <- c()
for (i in 1:N){
  y_fake[i] <- rbinom(1, n_city[i],
                      invlogit(a + beta %*% as.matrix(X)[i,]))
```

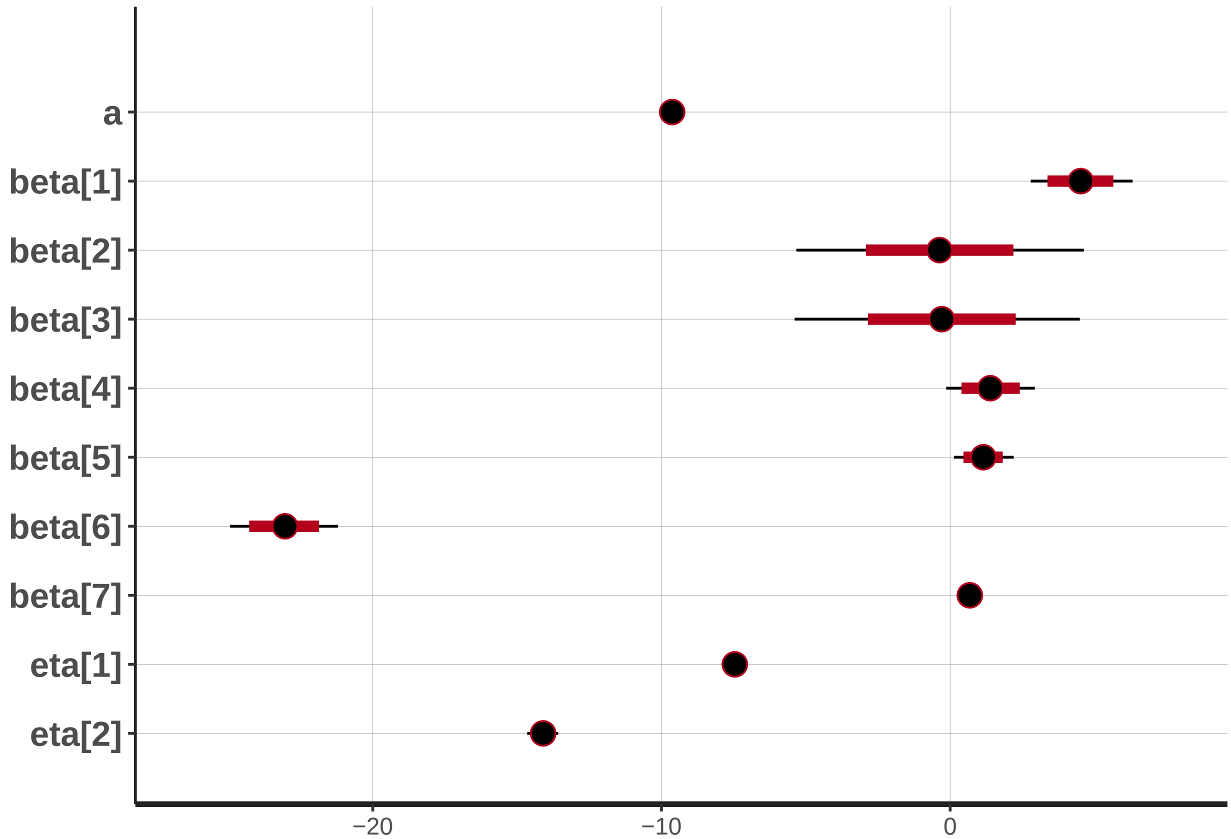
```
}

fake_baseline_data <- list(N_train=N, N_test=N, D=D,
                           X_train=X, X_test=X,
                           n_city_train = n_city,
                           n_city_test = n_city,
                           y_train = y_fake)
```

```
fit_fake <- sampling(baseline_model_binom,
                     data=fake_baseline_data, seed=1234)
saveRDS(fit_fake, file = 'baseline_fit_fake.rds')
```

```
fit_fake <- readRDS(file = 'baseline_fit_fake.rds')
print(fit_fake, pars = c('a', 'beta'),
      digits = 2, probs = c(0.025, 0.5, 0.975))
```

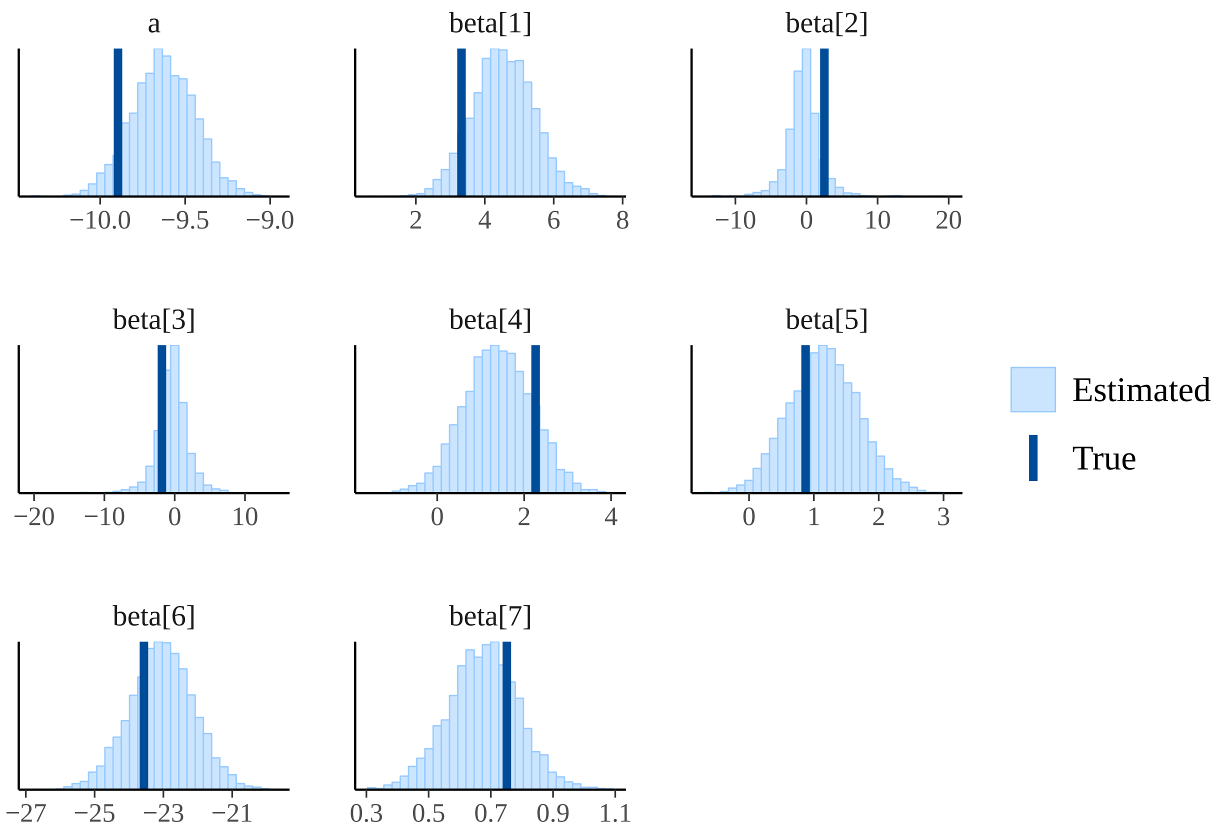
```
Inference for Stan model: binomial_baseline_city.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

          mean se_mean   sd   2.5%    50%   98% n_eff Rhat
a        -9.63    0.00 0.19  -9.99  -9.63  -9.3  2317    1
beta[1]   4.53    0.02 0.90   2.79   4.52   6.3  2545    1
beta[2]  -0.33    0.09 2.55  -5.32  -0.37   4.7   837    1
beta[3]  -0.33    0.09 2.53  -5.45  -0.29   4.5   846    1
beta[4]   1.40    0.02 0.78  -0.13   1.39   2.9  1831    1
beta[5]   1.15    0.01 0.53   0.13   1.15   2.2  1802    1
beta[6] -23.05    0.02 0.94 -24.94 -23.04 -21.2  1496    1
beta[7]   0.68    0.00 0.11   0.45   0.68   0.9  4000    1

Samples were drawn using NUTS(diag_e) at Wed Dec  5 20:20:24 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).
```

```
plot(fit_fake)
```



The following plot shows how well parameters recovered.

```
sims_fake <- as.matrix(fit_fake)
true <- c(a, beta)
color_scheme_set("brightblue")
mcmc_recover_hist(sims_fake[, 1:8], true)
```



We see that most β's didn't recovered well except β_6 and β_7.

Fit real data

Now we fit the real data with this model.

```
y = inputs$sum_y
#y2 = inputs$sum_y2

baseline_data = list(N_train=N, N_test=N, D=D,
                     X_train=X, X_test=X,
                     n_city_train = n_city,
                     n_city_test = n_city,
                     y_train = y)
```

```
fit1 <- sampling(baseline_model_binom,
                 data=baseline_data, seed=1234)
saveRDS(fit1, file = 'baseline_fit1.rds')
```

The following shows the posterior mean and sd of our parameters.

```
fit1 <- readRDS(file = 'baseline_fit1.rds')
print(fit1, pars=c('a', 'beta', 'lp__'),
      digits = 2, probs = c(0.025, 0.5, 0.975))
```

```
Inference for Stan model: binomial_baseline_city.
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%      50%      98% n_eff Rhat
a          -2.68    0.00 0.04    -2.76    -2.68    -2.61  2891    1
beta[1]    -0.69    0.01 0.36    -1.39    -0.68     0.01  2346    1
beta[2]     0.45    0.05 2.34    -4.45     0.33     5.47  1859    1
beta[3]     0.29    0.05 2.33    -4.67     0.39     5.13  1844    1
beta[4]     0.47    0.01 0.22     0.03     0.48     0.90  1985    1
beta[5]    -0.44    0.00 0.18    -0.78    -0.44    -0.09  1996    1
beta[6]    -0.14    0.00 0.20    -0.55    -0.14     0.24  2305    1
beta[7]    -0.08    0.00 0.08    -0.24    -0.08     0.08  3162    1
lp__    -7252.01    0.06 2.11 -7256.99 -7251.65 -7248.94  1267    1

Samples were drawn using NUTS(diag_e) at Wed Dec  5 20:26:39 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).
```

The posterior mean of intercept a is -2.68 with standard deviation 0.04. We can also see the posterior mean and standard deviation of coefficient parameter β from the above table. We see that β_2 and β_3 have larger standard deviation compare to others.

PPC

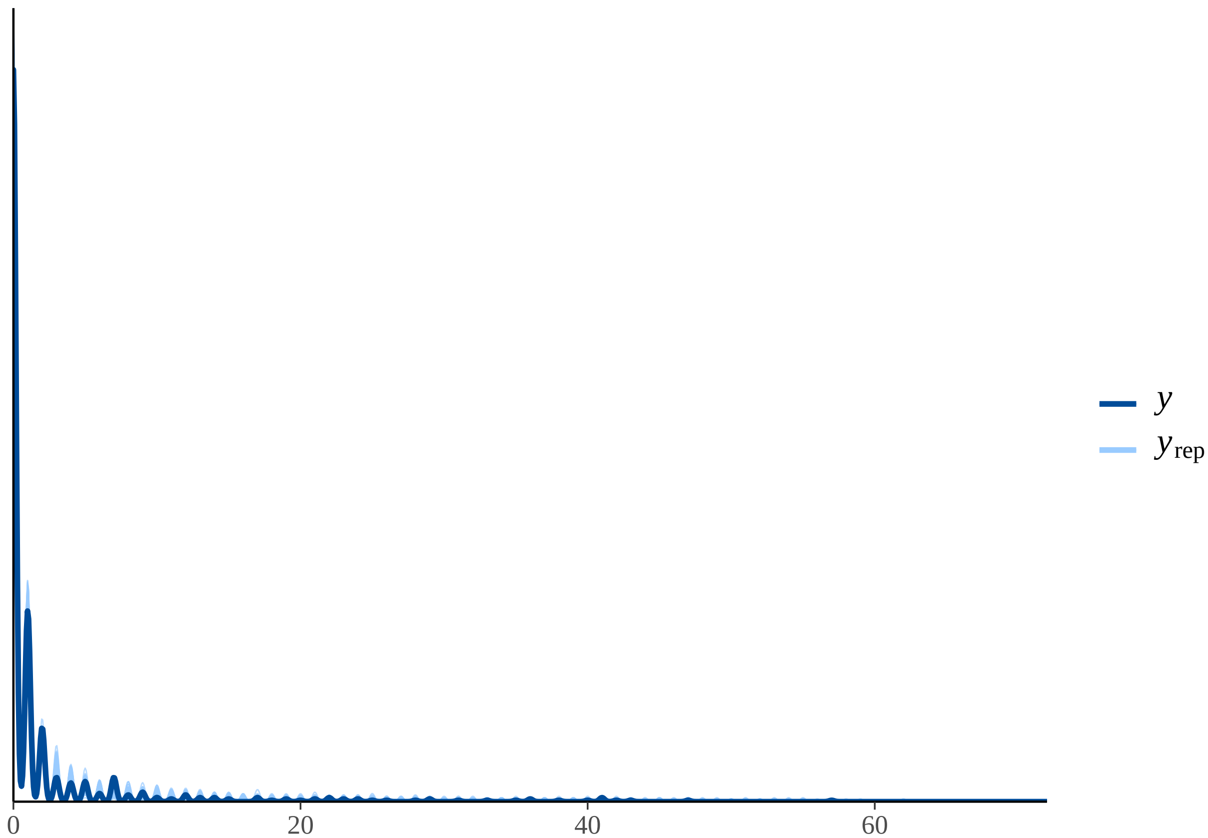
In stan model, we generated y_rep of city level using `binomial_rng` function in stan.

```
'transformed parameters {
  vector[N_train] eta;
  eta = a + X_train*beta;               // probability in binomial regression
}

generated quantities{
  int<lower =0> y_rep[N_train];
  for (i in 1:N_train){
    y_rep[i] = binomial_rng(n_city_train[i], inv_logit(eta[i]));
  }
}'
```

```
[1] "transformed parameters {\n  vector[N_train] eta;\n  eta = a + X_train*beta;               // probal
```

The following shows the posterior predictive check.

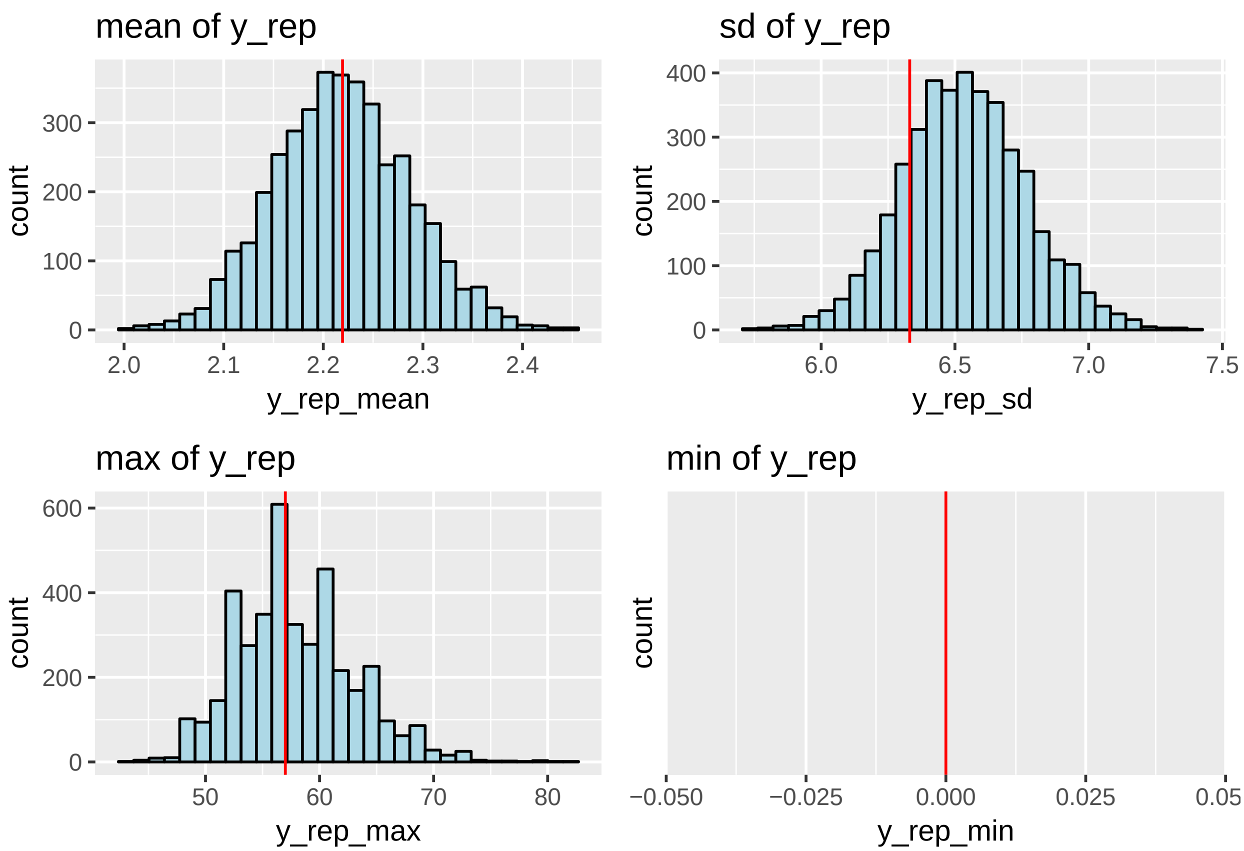
```
y_rep <- as.matrix(fit1, pars = "y_rep")
ppc_dens_overlay(y = y, y_rep[1:200,])
```



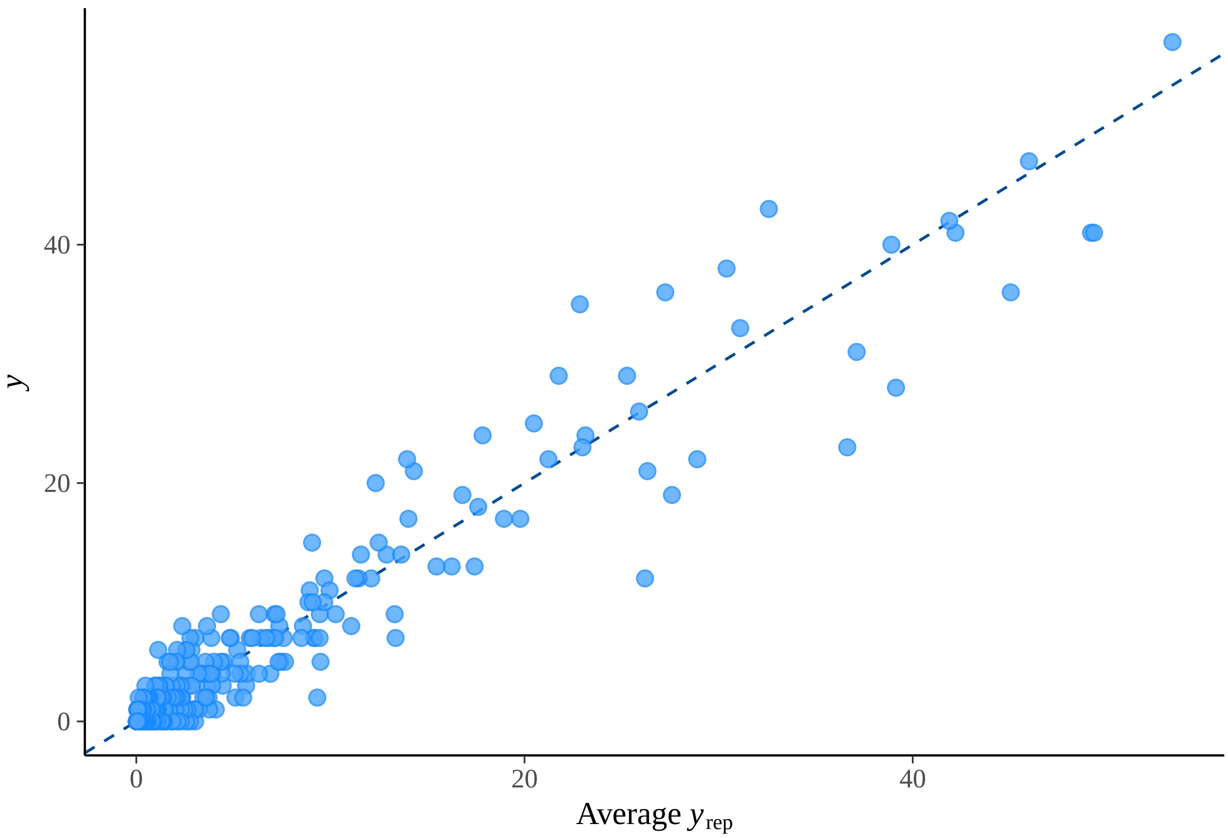
We see that the new y fit well even though there were some gaps.

```
sims <- rstan::extract(fit1)

df <- data.frame(y_rep_mean = apply(X=sims$y_rep, MARGIN = 1, FUN = mean))
meangg <- ggplot(df, aes(x=y_rep_mean)) +
  geom_histogram(fill='lightblue', color='black') +
  geom_vline(xintercept = mean(y), color='red') +
  ggtitle('mean of y_rep')

df <- data.frame(y_rep_sd = apply(X = sims$y_rep, MARGIN = 1, FUN = sd))
sdgg <- ggplot(df, aes(x=y_rep_sd)) +
  geom_histogram(fill='lightblue', color='black') +
  geom_vline(xintercept = sd(y), color='red') +
  ggtitle('sd of y_rep')

df <- data.frame(y_rep_max = apply(X = sims$y_rep, MARGIN = 1, FUN = max))
maxgg <- ggplot(df, aes(x=y_rep_max)) +
  geom_histogram(fill='lightblue',color='black') +
  geom_vline(xintercept = max(y), color='red') +
  ggtitle('max of y_rep')

df <- data.frame(y_rep_min = apply(X = sims$y_rep, MARGIN = 1, FUN = min))
mingg <- ggplot(df, aes(x=y_rep_min)) +
  geom_histogram(fill='lightblue',color='black') +
  geom_vline(xintercept = min(y), color='red') +
  ggtitle('min of y_rep')
```

```r
gridExtra::grid.arrange(meangg, sdgg, maxgg, mingg,
                        layout_matrix = rbind(c(1, 2),
                                              c(3, 4)))
```




```r
# Scatterplot of two test statistics
ppc_scatter_avg(y = y, yrep = y_rep)
```



Scattor plot looks linear but it is not perfect.

Evaluation (RMSE)

We evaluate our training model with RMSE using 5 fold cross validation.

```
## K fold CV
set.seed(1234)
splited_inputs <- split(inputs, sample(rep(1:5, 176)))

for (i in 1:5){
  a <- c(1,2,3,4,5)[-i]
  inputs_test = splited_inputs[[i]]
  inputs_train = rbind(splited_inputs[[a[1]]],
                       splited_inputs[[a[2]]],
                       splited_inputs[[a[3]]],
                       splited_inputs[[a[4]]])

  y_train = inputs_train$sum_y
  y_test = inputs_test$sum_y

  ## Inputs for STAN
  n_city_train = inputs_train$n_city
  n_city_test = inputs_test$n_city


  X_train = inputs_train %>%
```

```r
    dplyr::select(-ID_state,-n_city,-sum_y, -sum_y2)
  X_test = inputs_test %>%
    dplyr::select(-ID_state,-n_city,-sum_y, -sum_y2)

  N_train = nrow(X_train)
  N_test = nrow(X_test)

  D = ncol(X_train)

  baseline_data = list(N_train=N_train, N_test=N_test, D=D,
                       X_train=X_train, X_test=X_test,
                       n_city_train = n_city_train,
                       n_city_test = n_city_test,
                       y_train = y_train)
  fit_cv <- sampling(baseline_model_binom, data=baseline_data, seed=1234)
  name = paste('model1_cv', as.character(i), '.rds', sep = "")
  saveRDS(fit_cv, file = name)
}
```

```r
rmse <- c()
for (i in 1:5){
  name = paste('model1_cv', as.character(i),'.rds',sep = "")
  fit_cv <- readRDS(file = name)
  sims_cv <- rstan::extract(fit_cv)
  y_hat <- apply(X = sims_cv$y_rep_cv, MARGIN = 2, FUN = median)

  test_df = data.frame(ID_state = inputs_test$ID_state,
                       y_test = y_test,
                       y_hat = y_hat)

  test_df <- test_df %>%
    summarize(y_sum_test = sum(y_test),
              y_sum_hat = sum(y_hat)) %>%
    arrange(desc(y_sum_test)) %>%
    ungroup()

  #mse_baseline = mean((test_df$y_sum_test) ** 2)
  rmse[i] = sqrt(mean((test_df$y_sum_hat - test_df$y_sum_test) ** 2))
}

average_RMSE <- mean(rmse)
sd_RMSE <- sd(rmse)

cat('The average of RMSE of baseline model is: ', average_RMSE)
```

```
The average of RMSE of baseline model is:  75
```

```r
cat('The standard deviation of RMSE of baseline model is: ', sd_RMSE)
```

```
The standard deviation of RMSE of baseline model is:  34
```