binomial_baseline

Jongwoo Choi

2018-12-02

```
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
  )


## Train / Test split
set.seed(seed = 1234)
pct_train = 0.8
sample_size = round(pct_train * nrow(inputs))
sample <- sample(x = nrow(inputs), size = sample_size, replace = F)

## Allocate train
y = inputs$sum_y
y2 = inputs$sum_y2

inputs_train = inputs[sample, ]
y_train = y[sample]
```

```r
## Allocate test
inputs_test = inputs[-sample, ]
y_test = y[-sample]


## Inputs for STAN
X_train = inputs_train %>% 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)

n_city_train = inputs_train$n_city
n_city_test = inputs_test$n_city

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)
```

Baseline Model: Binomial

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

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

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

$$y_c \sim \mathsf{Binomial}(N_c,\ logit^{-1}(a + \beta \cdot X_c))$$

The binomial baseline model is given by:

```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 {
  // regression coefficient vector
  real a; // include intercept
  vector[D] beta;
```

```
}
transformed parameters {
  vector[N_train] eta;
  eta = a + X_train*beta;
}
model {
  a ~ normal(0, 5);
  beta ~ normal(0, 5);
  y_train ~ binomial_logit(n_city_train, eta);
}
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.

```
a <- rcauchy(1, 0, 10)
beta <- rcauchy(7, 0, 2.5)

y_fake <- c()
for (i in 1:N_train){
  y_fake[i] <- rbinom(1, n_city_train[i],
                      invlogit(a + beta %*% as.matrix(X_train)[i,]))
}

fake_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_fake)
```

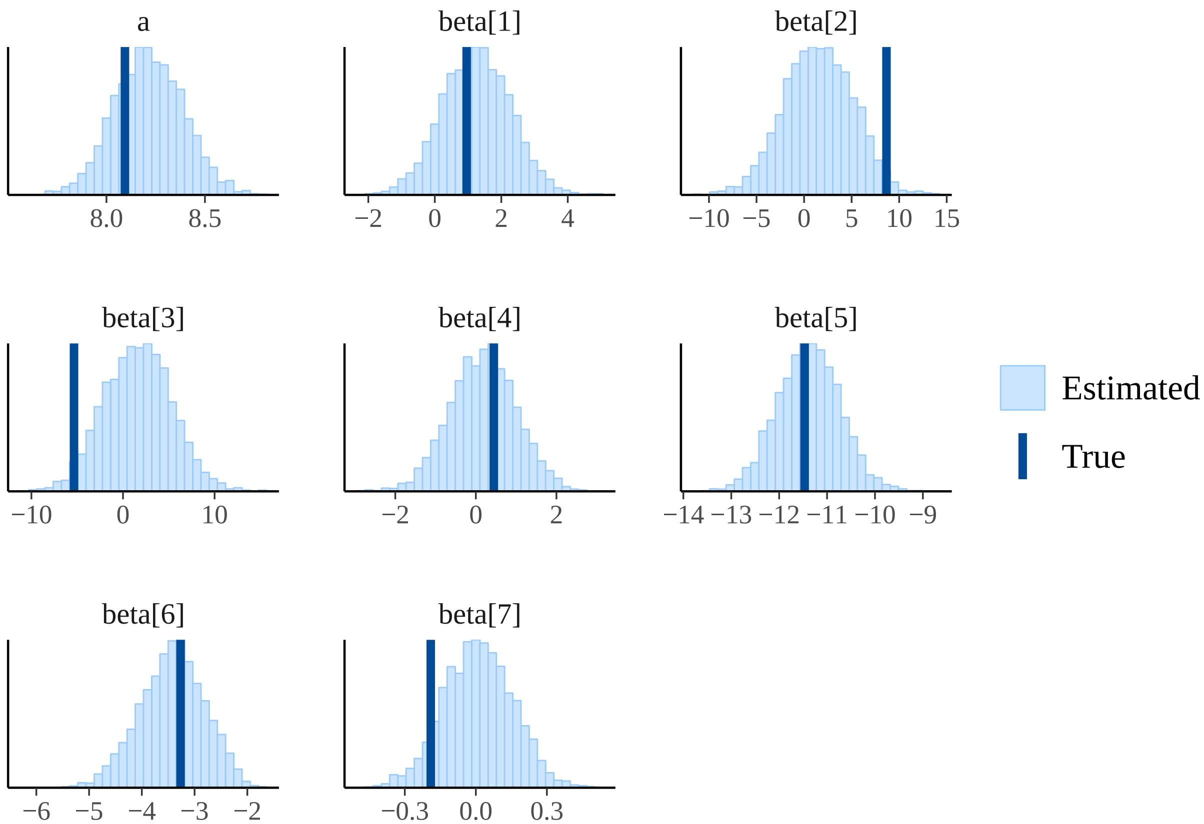
```
baseline_model_binom=stan_model('binomial_baseline_city.stan')
fit_fake <- sampling(baseline_model_binom, data=fake_baseline_data, seed=1234)
```

```
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         8.22    0.00 0.18  7.88  8.21  8.56  2810    1
beta[1]   1.25    0.02 1.02 -0.76  1.23  3.24  2469    1
beta[2]   1.54    0.07 3.67 -5.63  1.53  8.47  2417    1
```

```
beta[3]   1.63    0.08 3.71  -5.42   1.66   8.89  2403    1
beta[4]   0.18    0.02 0.82  -1.43   0.20   1.77  1902    1
beta[5] -11.38    0.02 0.66 -12.67 -11.37 -10.11  1777    1
beta[6]  -3.43    0.01 0.62  -4.68  -3.40  -2.29  2227    1
beta[7]   0.01    0.00 0.15  -0.28   0.01   0.29  3561    1

Samples were drawn using NUTS(diag_e) at Sun Dec  2 20:51:37 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).
```

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



Fit real data

Now we fit the real data with this model.

```
#baseline_model_binom=stan_model('binomial_baseline_city.stan')
fit1 <- sampling(baseline_model_binom, data=baseline_data, seed=1234)
```

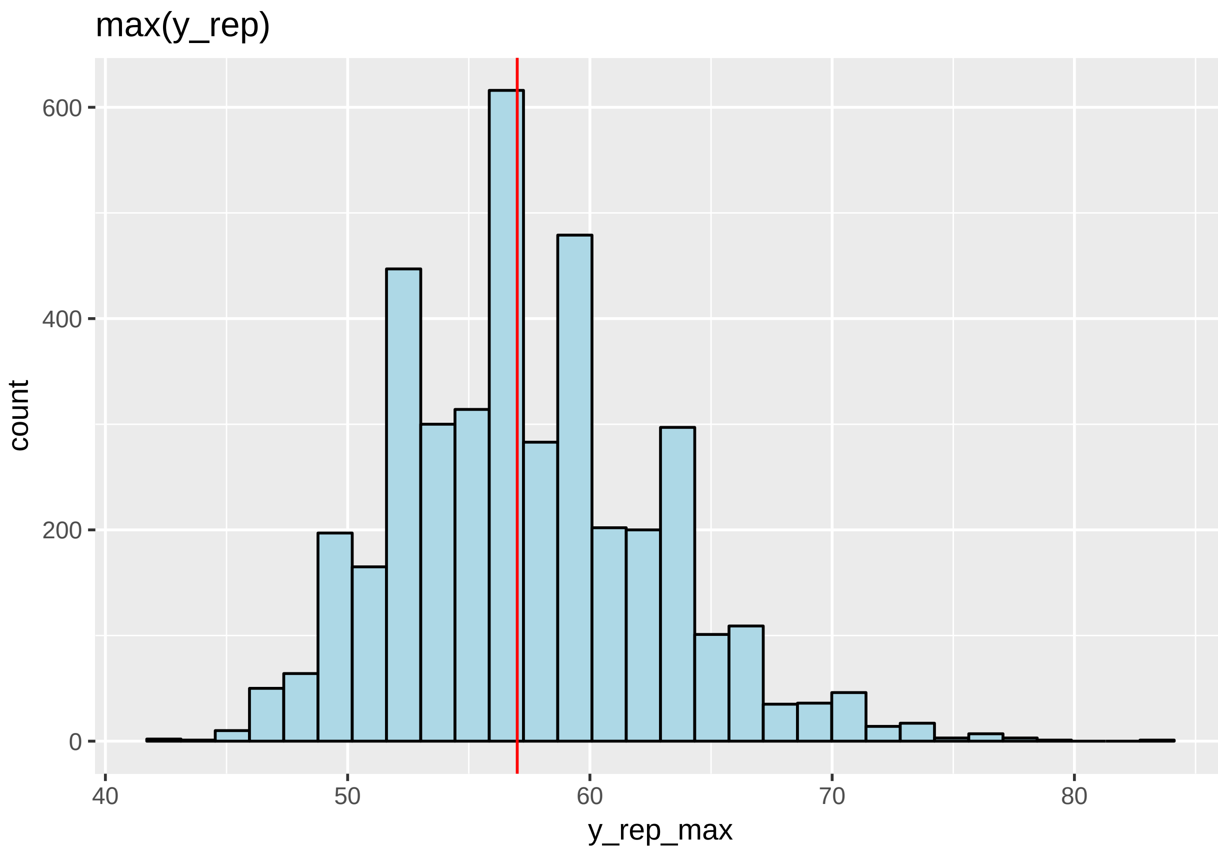
```
print(fit1, pars=c('a', 'beta', 'lp__'),
      digits = 2, probs = c(0.025, 0.5, 0.975))
```

```
Inference for Stan model: binomial_baseline_city.
4 chains, each with iter=2000; warmup=1000; thin=1;
```

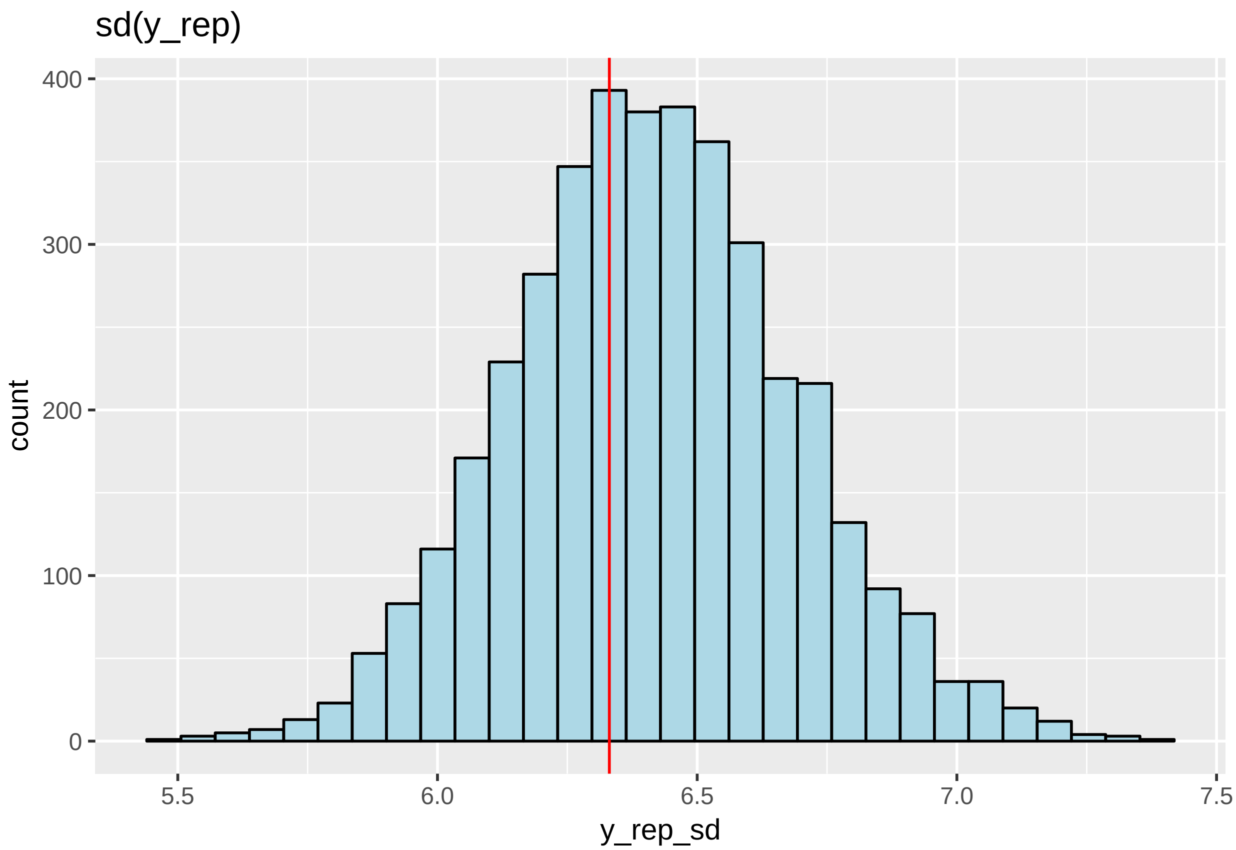
```
post-warmup draws per chain=1000, total post-warmup draws=4000.

             mean se_mean   sd     2.5%      50%      98% n_eff Rhat
a           -2.72    0.00 0.04    -2.81    -2.72    -2.64  2685    1
beta[1]     -0.63    0.01 0.41    -1.44    -0.64     0.17  2178    1
beta[2]      0.33    0.08 3.61    -6.56     0.34     7.39  2147    1
beta[3]      0.35    0.08 3.59    -6.68     0.35     7.24  2156    1
beta[4]      0.33    0.01 0.25    -0.16     0.33     0.84  1837    1
beta[5]     -0.25    0.00 0.21    -0.65    -0.24     0.17  1810    1
beta[6]      0.01    0.00 0.22    -0.43     0.01     0.43  2490    1
beta[7]     -0.07    0.00 0.09    -0.25    -0.07     0.12  3412    1
lp__     -5487.86    0.05 2.01 -5492.65 -5487.54 -5484.90  1586    1

Samples were drawn using NUTS(diag_e) at Sun Dec  2 20:53:59 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).
```

PPC

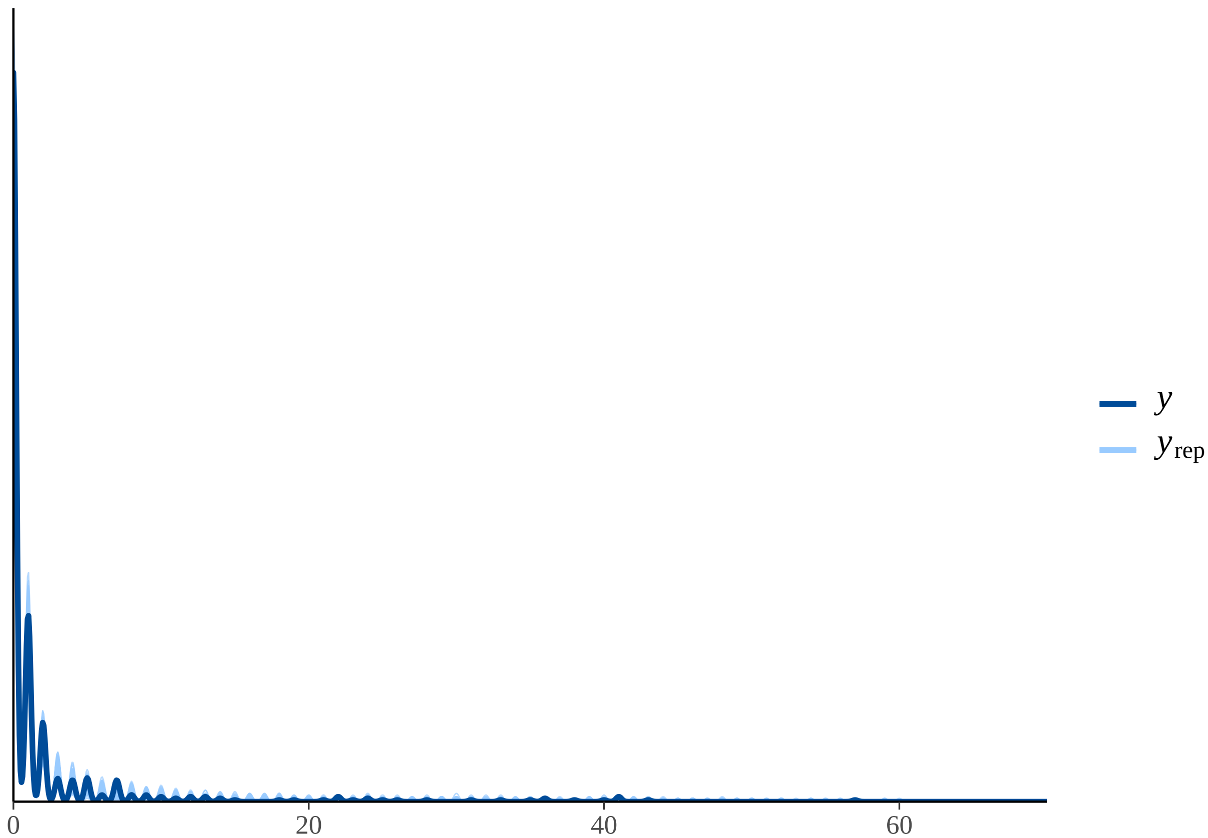
```r
sims <- rstan::extract(fit1)
y_max <- apply(X = sims$y_rep, MARGIN = 1, FUN = max)

df <- data.frame(y_rep_max = y_max)
ggplot(df, aes(x=y_rep_max)) +
  geom_histogram(fill='lightblue',
                 color='black') +
  geom_vline(xintercept = max(y), color='red') +
  ggtitle('max(y_rep)')
```




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




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



MSE

```
# a_median <- median(sims$a)
# beta_median <- apply(X = sims$beta, MARGIN = 2, FUN = median)
#
# y_hat <- N_test * invlogit(a_median + as.matrix(X_test)%*%beta_median)
# mode <- function(x) {
#   ux <- unique(x)
#   ux[which.max(tabulate(match(x, ux)))]
# }

y_hat <- apply(X = sims$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()

accuracy_baseline = mean(abs(test_df$y_sum_test) ** 2)
accuracy = mean(abs(test_df$y_sum_hat - test_df$y_sum_test) ** 2)
```

```r
cat('\nBaseline MSE: ', accuracy_baseline * 100)
```

```
Baseline MSE:  2.3e+07
```

```r
cat('\nLogistic MSE: ', accuracy * 100)
```

```
Logistic MSE:  291600
```