Bayesian Logistic Regression

July 2024

Importing the data

We begin by loading the required packages:

```r
library(rstan)
library(HDInterval)
library(bayesplot)
library(tidybayes)
library(tidyverse)
library(cowplot)
```

We now import the data files: after storing the path to the working directory (WD) in the `work_dir` object, we can set the WD and load the auxiliary functions:

```r
setwd(work_dir)
```

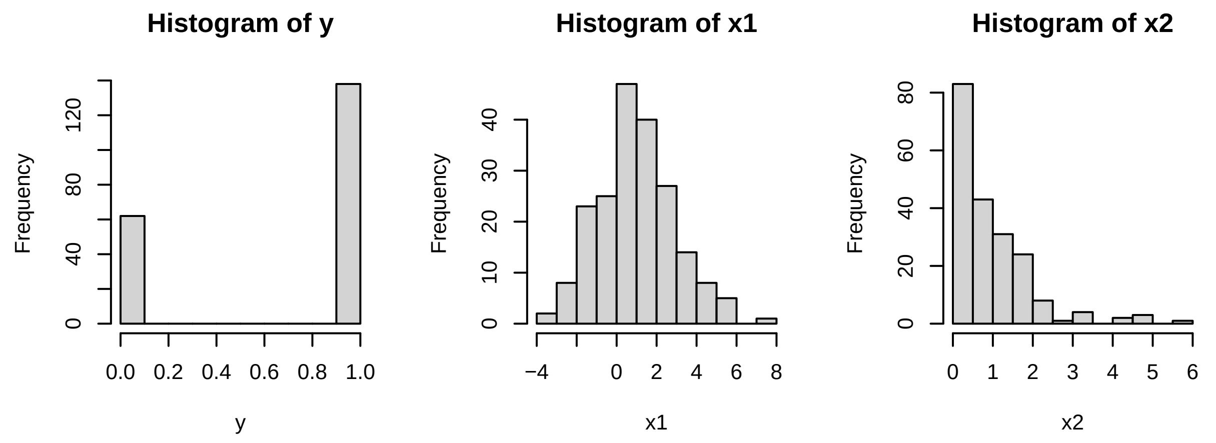
```r
load("logistic_regression_data.RData")
```

```r
source("aux_functions.R")
```

Data Preparation

Let us begin by considering a binary outcome variable `y` and two numeric regressors `x1` and `x2`:

```r
par(mfrow=c(1,3), pty="s")
hist(y)
hist(x1)
hist(x2)
```



```r
par(mfrow=c(1,1), pty="m")
```

Our goal is to estimate a logistic regression model:

$$y_i \sim \text{Bernoulli}(\pi_i)$$

where:

$$\pi_i = \frac{1}{1 + \exp\{\alpha + \beta_1 x_{i,1} + \beta_2 x_{i,2}\}}$$

Therefore, we first need to set priors for α, β_1 and β_2. Unlike linear regression models, however, choosing a distribution for these parameters through simulation checks is not as straightforward because of the non-linear transformation in the inverse-logistic function.

Moreover, whereas non-information or weak information is easy to conceptualized in linear models, how do these assumption translate in binary (or multinational) settings? Intuitively, uninformative or weakly informative priors would keep the chances of $y_i = 1$ or $y_i = 0$ roughly equal, which means $\pi_i \approx 0.5$.

To understand how priors on α, β_1 and β_2 can be calibrated such that $\pi_i \approx 0.5$ across simulations, we begin by addressing what distribution best suits the intercept.

Prior calibration via Prior Predictive Checks

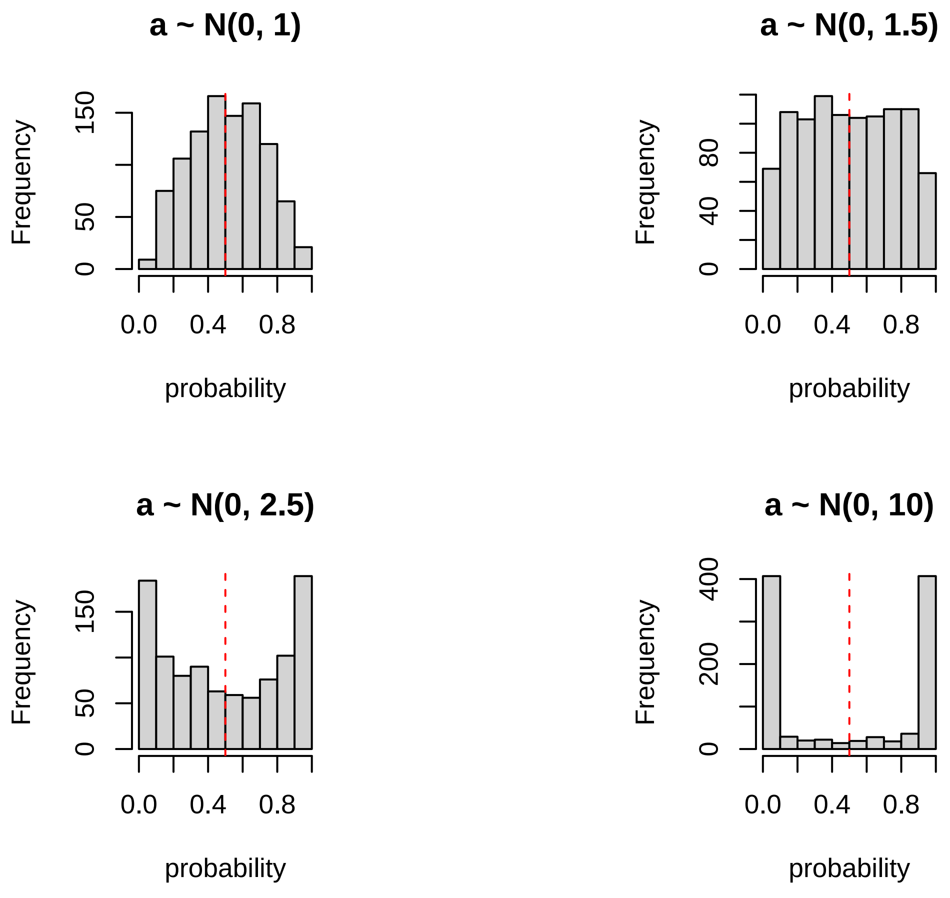
Calibrating α

Let us begin with four simple prior distributions for α:

$$\begin{aligned}\alpha &\sim N(0, 10)\\ \alpha &\sim N(0, 2.5)\\ \alpha &\sim N(0, 1.5)\\ \alpha &\sim N(0, 1)\end{aligned} \tag{1}$$

We simulate $S = 1000$ draws from these distribution, compute $\pi^s = \frac{1}{1+\exp\{\alpha^s\}}$

```r
par(mfrow=c(2,2), pty="s")
for (j in c(1, 1.5, 2.5, 10)) {

  hist(boot::inv.logit(rnorm(1000, 0, j)),
       main = paste0("a ~ N(0, ", j, ")"),
       xlab = "probability")
  abline(v = 0.5, col = "red", lty = 2)

}
```



```r
par(mfrow=c(1,1), pty="m")
```

If we believe that $\pi \approx 0.5$ is a reasonable assumption, then $\alpha \sim N(0, 1)$ is a good uninformative prior for the intercept. Notice how we would have never guessed that such a tight normal would have lead to a weakly informative prior.

Calibrating the slope parameters β_1 and β_2

Understanding what prior to give β_1 and β_2 is a bit more complicated. In fact, not only do the distribution on these parameters will impact π, but also their interaction with the covariate values will play a major role.

To see how this plays out, we begin by simulating probabilities by first setting;

$$\begin{aligned} \alpha &\sim N(0, 1) \\ \beta_1 &\sim N(0, 5) \\ \beta_2 &\sim N(0, 5) \end{aligned} \tag{2}$$

and computing $\pi_i^s = 1/(1 + \exp\{\alpha^s + \beta_1^s x_{i,1} + \beta_2^s x_{i,2}\})$. Unlike our simulation for the sole intercept, here we adopt a slightly different visualization because we need to visualize $S \times N$ generated probabilities.

This time we use the `stan` program to simulate:

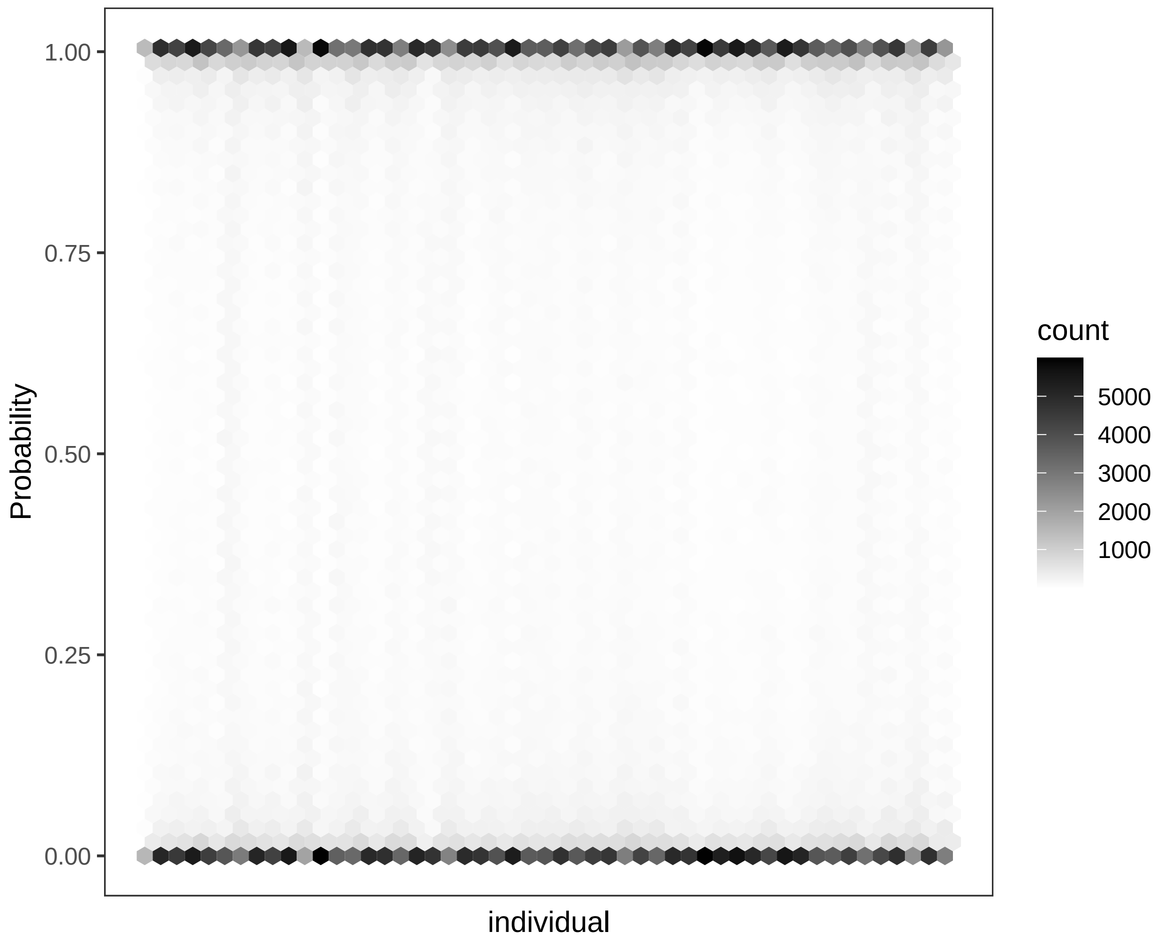
```r
dat_list <- list(
  N = length(y),
  x1 = x1,
  x2 = x2,
  y = y,
  mu_b1 = 0,
  mu_b2 = 0,
  sigma_b1 = 5,
  sigma_b2 = 5,
  mu_alpha = 0,
  sigma_alpha = 1
)

logi_reg <- stan_model("logistic_regression.stan")
logi_reg_prior <- stan_model("logistic_regression_prior.stan")

prior_sample <- sampling(logi_reg_prior,
                         data = dat_list,
                         algorithm = "Fixed_param")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.023 seconds (Sampling)
## Chain 1:                0.023 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
## Chain 2:                0.021 seconds (Sampling)
## Chain 2:                0.021 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.021 seconds (Sampling)
## Chain 3:                0.021 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.021 seconds (Sampling)
## Chain 4:                0.021 seconds (Total)
## Chain 4:
```

```r
prior_sample_fit <- rstan::extract(prior_sample)
prior_sample_fit_pi <- prior_sample %>% spread_draws(prob[condition])
```

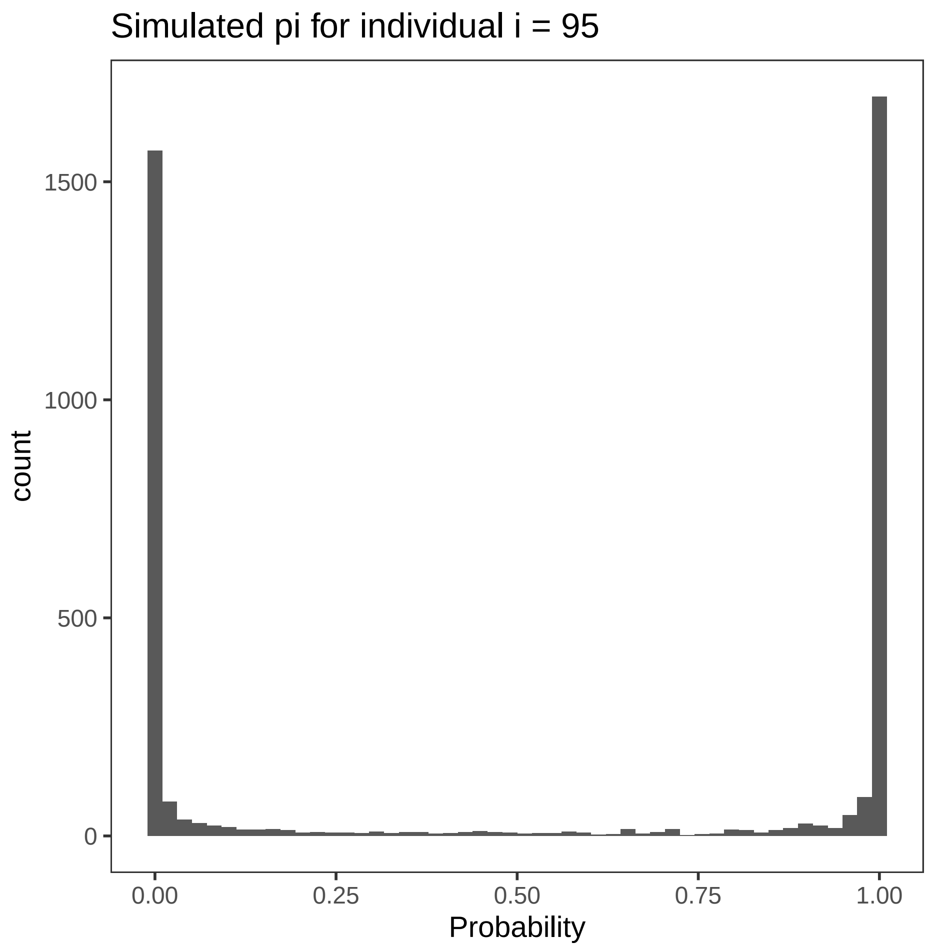
Let us now visualize the simulated π_i^s:

```r
prior_sample_fit_pi %>%
  ggplot(aes(x = condition, y = prob)) +
  geom_hex(bins = 50) +
  scale_fill_gradient(low = "white", high = "black") +
  theme_bw() +
  labs(x = "individual",
       y = "Probability") +
  theme(aspect.ratio = 1,
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        panel.grid = element_blank())
```



You can think of this plot as stacking sequentially (i.e., horizontal axis) a series of histograms like the ones that we have seen above. For example, by randomly picking one individual out of the 200 in out sample, we can visualize thir simulated probability $\tilde{\pi}_i$:

```r
i <- sample(1:200)[1]

prior_sample_fit_pi %>%
  filter(condition==i) %>%
  ggplot(aes(x = prob)) +
  geom_histogram(bins = 50) +
  theme_bw() +
  labs(x = "Probability",
       title = paste0("Simulated pi for individual i = ", i)) +
  theme(aspect.ratio = 1,
        panel.grid = element_blank())
```



Notice how all the probability values concentrate around zero and one, while very few data points fall in the middle part of the plot. In other words, $\beta_1 \sim N(0,5)$ and $\beta_2 \sim N(0,5)$ are very informative in that they automatically set probability values to their extremes.

How can we go about this? We cloud try by setting tighter priors:

$$\begin{aligned} \alpha &\sim N(0,1) \\ \beta_1 &\sim N(0,1) \\ \beta_2 &\sim N(0,1) \end{aligned} \tag{3}$$

```
dat_list <- list(
  N = length(y),
  x1 = x1,
  x2 = x2,
  y = y,
  mu_b1 = 0,
  mu_b2 = 0,
  sigma_b1 = 1,
  sigma_b2 = 1,
  mu_alpha = 0,
  sigma_alpha = 1
)
```

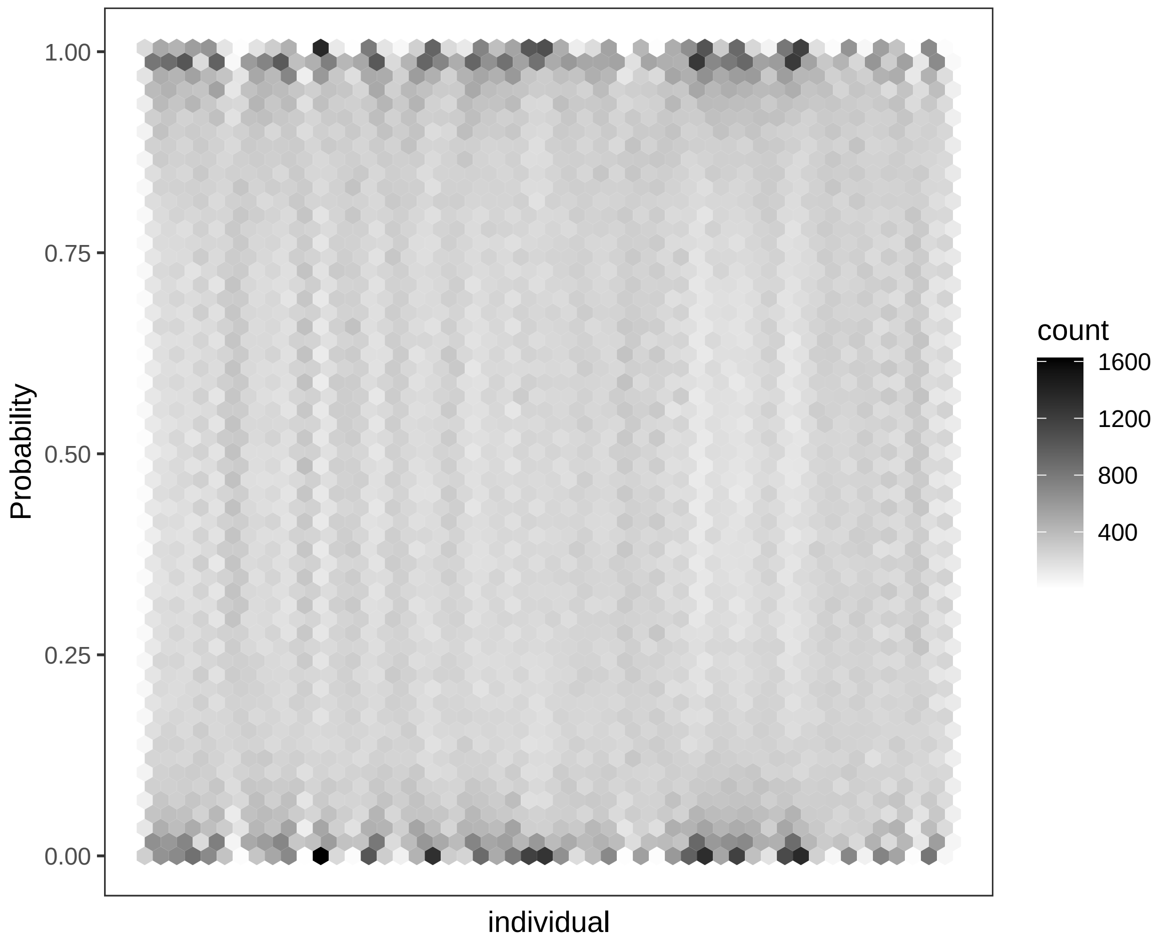
```r
prior_sample <- sampling(logi_reg_prior,
                         data = dat_list,
                         algorithm = "Fixed_param")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.021 seconds (Sampling)
## Chain 1:                0.021 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
## Chain 2:                0.021 seconds (Sampling)
## Chain 2:                0.021 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.021 seconds (Sampling)
## Chain 3:                0.021 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.021 seconds (Sampling)
## Chain 4:                0.021 seconds (Total)
## Chain 4:
```

```r
prior_sample_fit <- rstan::extract(prior_sample)
prior_sample_fit_pi <- prior_sample %>% spread_draws(prob[condition])
```

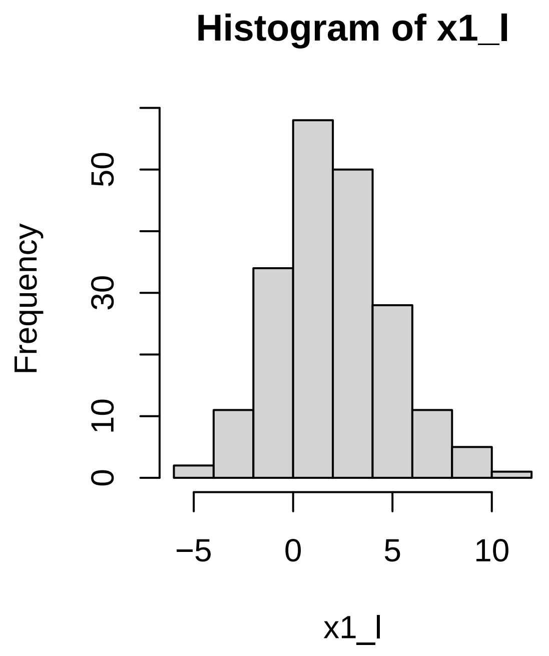
Let us now visualize the simulated π_i^s:

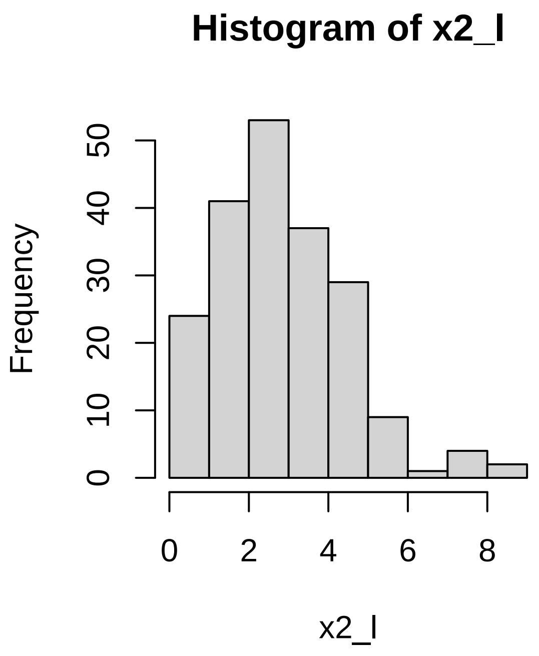
```r
prior_sample_fit_pi %>%
  ggplot(aes(x = condition, y = prob)) +
  geom_hex(bins = 50) +
  scale_fill_gradient(low = "white", high = "black") +
  theme_bw() +
  labs(x = "individual",
       y = "Probability") +
  theme(aspect.ratio = 1,
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        panel.grid = element_blank())
```



This looks slightly better, but many probabilities still lie in the $[0, 1]$ area (black areas around the edges). Moreover, these prior settings are conditional on the distribution of x_1 and x_2, meaning that different configuration of these variables might invalidate the results displayed above. For example, let us repeat this exercise with two covariates with larger support:

```r
set.seed(123)
x1_l <- rnorm(200, 2, 3)
x2_l <- rgamma(200, 3, 1)

par(mfrow=c(1,2), pty="s")
hist(x1_l)
hist(x2_l)
```



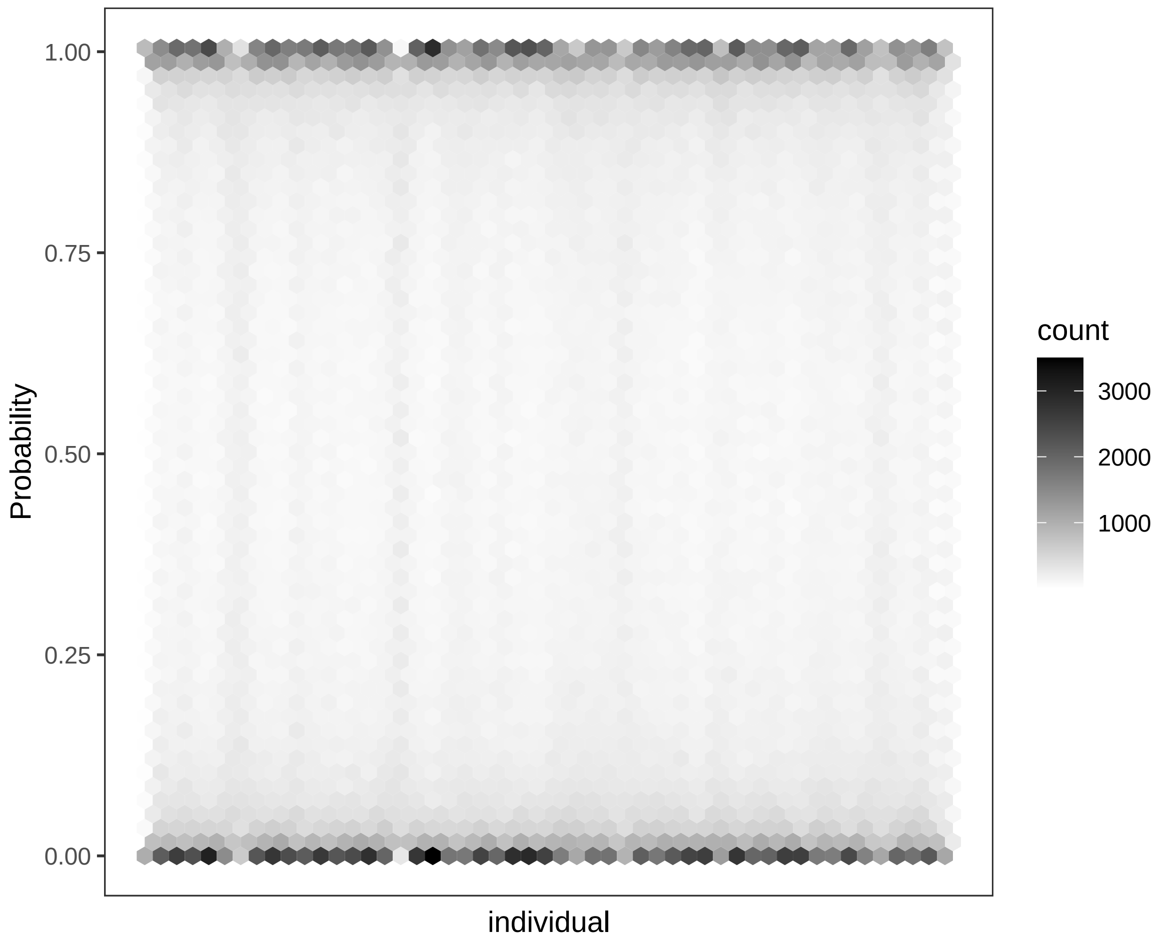


```r
par(mfrow=c(1,1), pty="m")
```

```r
dat_list_l <- list(
  N = length(y),
  x1 = x1_l,
  x2 = x2_l,
  y = y,
  mu_b1 = 0,
  mu_b2 = 0,
  sigma_b1 = 1,
  sigma_b2 = 1,
  mu_alpha = 0,
  sigma_alpha = 1
)

prior_sample_l <- sampling(logi_reg_prior,
                           data = dat_list_l,
                           algorithm = "Fixed_param")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.021 seconds (Sampling)
## Chain 1:                0.021 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
## Chain 2:                0.02 seconds (Sampling)
## Chain 2:                0.02 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.019 seconds (Sampling)
## Chain 3:                0.019 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.021 seconds (Sampling)
## Chain 4:                0.021 seconds (Total)
## Chain 4:
```

```
prior_sample_fit_l <- rstan::extract(prior_sample_l)
prior_sample_fit_pi_l <- prior_sample_l %>% spread_draws(prob[condition])

prior_sample_fit_pi_l %>%
  ggplot(aes(x = condition, y = prob)) +
  geom_hex(bins = 50) +
  scale_fill_gradient(low = "white", high = "black") +
  theme_bw() +
  labs(x = "individual",
       y = "Probability") +
  theme(aspect.ratio = 1,
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        panel.grid = element_blank())
```



Even with the a tightener distribution on β_1 and β_2 we are back to the original problem.

The importance of standardizing

To make the scale of x_1 and x_2 irrelevant when placing priors on β_1 and β_2, one can *scale* the regressors to have mean zero and standard deviation 0.5, i.e.,

$$x_{i,p}^{sc} = \frac{x_{i,p} - \bar{x}_p}{sd(x_p)} \times 0.5$$

We have already performed this operation when studying default priors in Normal regression models. In logistic regression, however, standardizing is fundamental because predicted probabilities are highly sensitive to large covariate values. To see how standardization gets rid of this issue, let us begin by scaling the initial covariates and simulate probability values:

```
x1_sc <- as.vector(scale(x1)*0.5)
x2_sc <- as.vector(scale(x2)*0.5)

dat_list <- list(
  N = length(y),
  x1 = x1_sc,
```

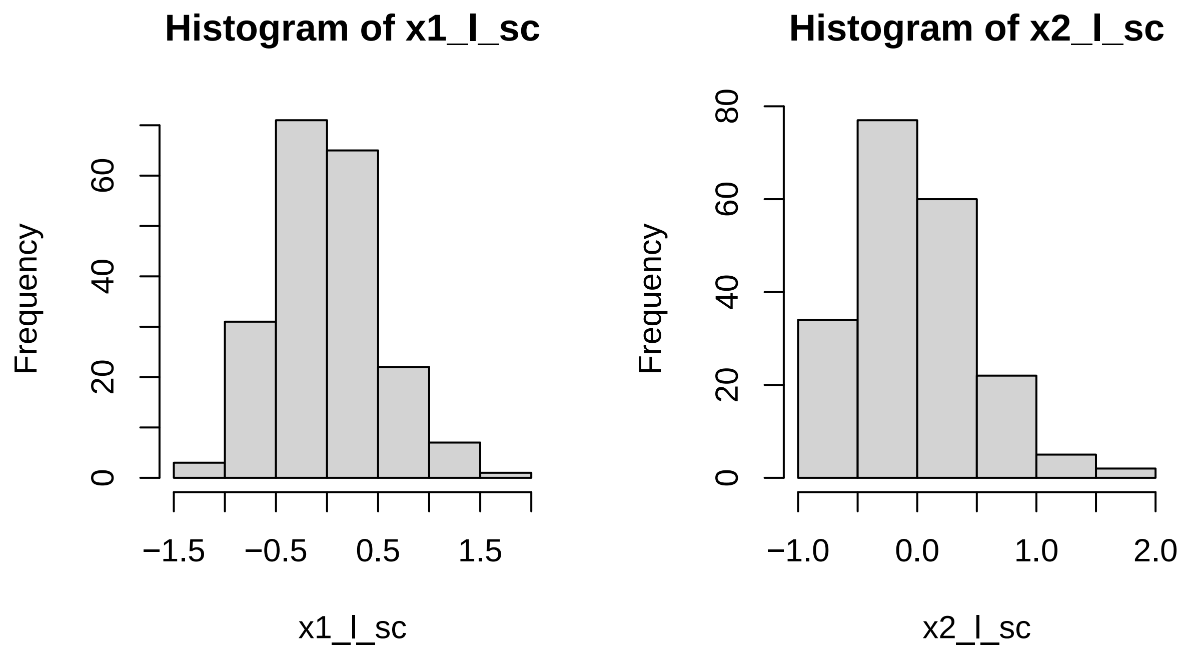
```r
  x2 = x2_sc,
  y = y,
  mu_b1 = 0,
  mu_b2 = 0,
  sigma_b1 = 1,
  sigma_b2 = 1,
  mu_alpha = 0,
  sigma_alpha = 1
)

prior_sample <- sampling(logi_reg_prior,
                         data = dat_list,
                         algorithm = "Fixed_param")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.021 seconds (Sampling)
## Chain 1:                0.021 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
## Chain 2:                0.02 seconds (Sampling)
## Chain 2:                0.02 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.021 seconds (Sampling)
## Chain 3:                0.021 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.021 seconds (Sampling)
## Chain 4:                0.021 seconds (Total)
## Chain 4:
```

```r
prior_sample_fit <- rstan::extract(prior_sample)
prior_sample_fit_pi <- prior_sample %>% spread_draws(prob[condition])
```

We next repeat the same exercise using scaled values for `x1_l` and `x2_l`:

```r
x1_l_sc <- as.vector(scale(x1_l)*0.5)
x2_l_sc <- as.vector(scale(x2_l)*0.5)

par(mfrow=c(1,2), pty="s")
hist(x1_l_sc)
hist(x2_l_sc)
```



```r
par(mfrow=c(1,1), pty="m")
```

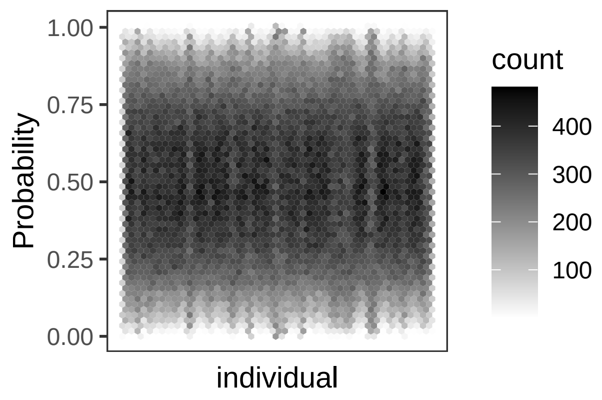
```r
dat_list_l <- list(
  N = length(y),
  x1 = x1_l_sc,
  x2 = x2_l_sc,
  y = y,
  mu_b1 = 0,
  mu_b2 = 0,
  sigma_b1 = 1,
  sigma_b2 = 1,
  mu_alpha = 0,
  sigma_alpha = 1
)

prior_sample_l <- sampling(logi_reg_prior,
                           data = dat_list_l,
                           algorithm = "Fixed_param")
```

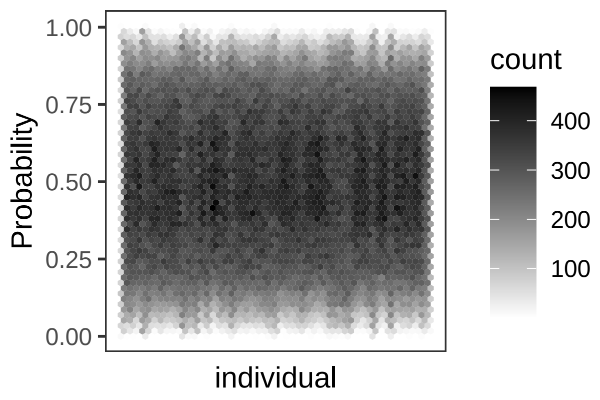
```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.021 seconds (Sampling)
## Chain 1:                0.021 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
## Chain 2:                0.022 seconds (Sampling)
## Chain 2:                0.022 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.021 seconds (Sampling)
## Chain 3:                0.021 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.021 seconds (Sampling)
## Chain 4:                0.021 seconds (Total)
## Chain 4:
```

```
prior_sample_fit_l <- rstan::extract(prior_sample_l)
prior_sample_fit_pi_l <- prior_sample_l %>% spread_draws(prob[condition])
```

We can finally visualize the simulated π_i for both configurations:

```
prior_sample_fit_pi %>%
  ggplot(aes(x = condition, y = prob)) +
  geom_hex(bins = 50) +
  scale_fill_gradient(low = "white", high = "black") +
  theme_bw() +
  labs(x = "individual",
       y = "Probability") +
  theme(aspect.ratio = 1,
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        panel.grid = element_blank()) -> PrPD_point

prior_sample_fit_pi_l %>%
  ggplot(aes(x = condition, y = prob)) +
  geom_hex(bins = 50) +
  scale_fill_gradient(low = "white", high = "black") +
  theme_bw() +
  labs(x = "individual",
       y = "Probability") +
  theme(aspect.ratio = 1,
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        panel.grid = element_blank()) -> PrPD_point_l

plot_grid(PrPD_point, PrPD_point_l)
```

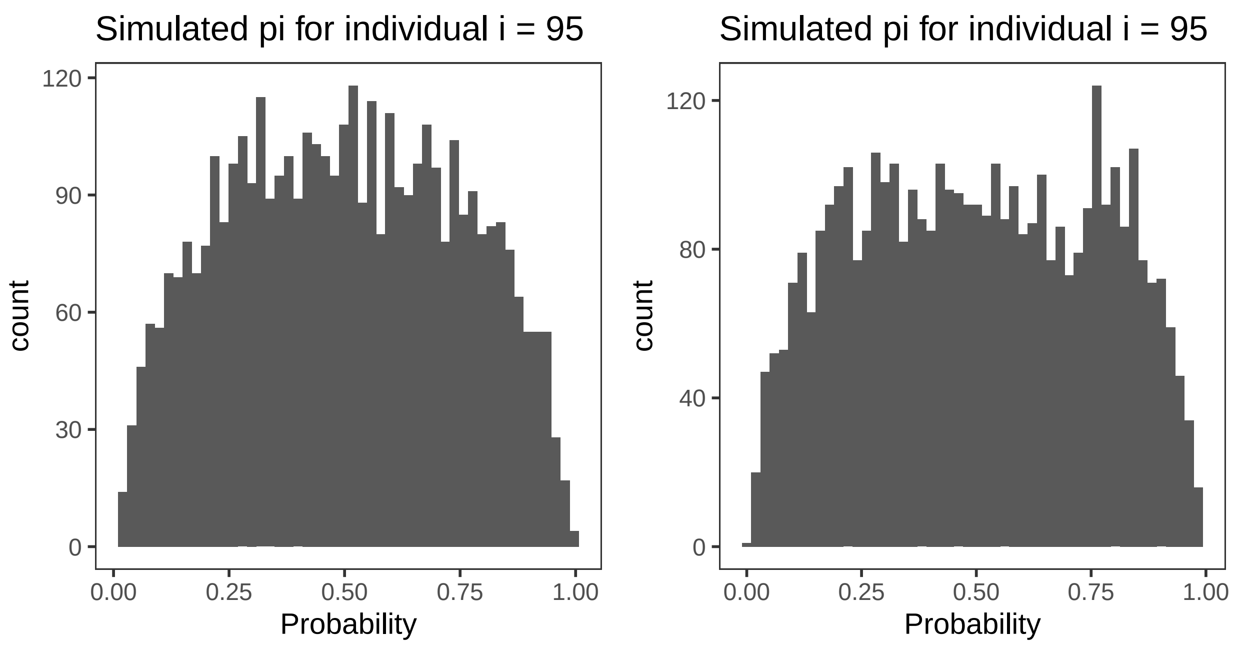




The prior distribution of π_i now concentrates around values close to 0.5, as intended, regardless of the covariates' support:

```r
prior_sample_fit_pi %>%
  filter(condition==i) %>%
  ggplot(aes(x = prob)) +
  geom_histogram(bins = 50) +
  theme_bw() +
  labs(x = "Probability",
       title = paste0("Simulated pi for individual i = ", i)) +
  theme(aspect.ratio = 1,
        panel.grid = element_blank()) -> PrPD_i

prior_sample_fit_pi_l %>%
  filter(condition==i) %>%
  ggplot(aes(x = prob)) +
  geom_histogram(bins = 50) +
  theme_bw() +
  labs(x = "Probability",
       title = paste0("Simulated pi for individual i = ", i)) +
  theme(aspect.ratio = 1,
        panel.grid = element_blank())  -> PrPD_i_l
```

```
plot_grid(PrPD_i, PrPD_i_l)
```



Estimation

Now that we have figured out a reasonable weakly informative prior configuration for α, β_1 and β_2, we can fit the model using HMC:

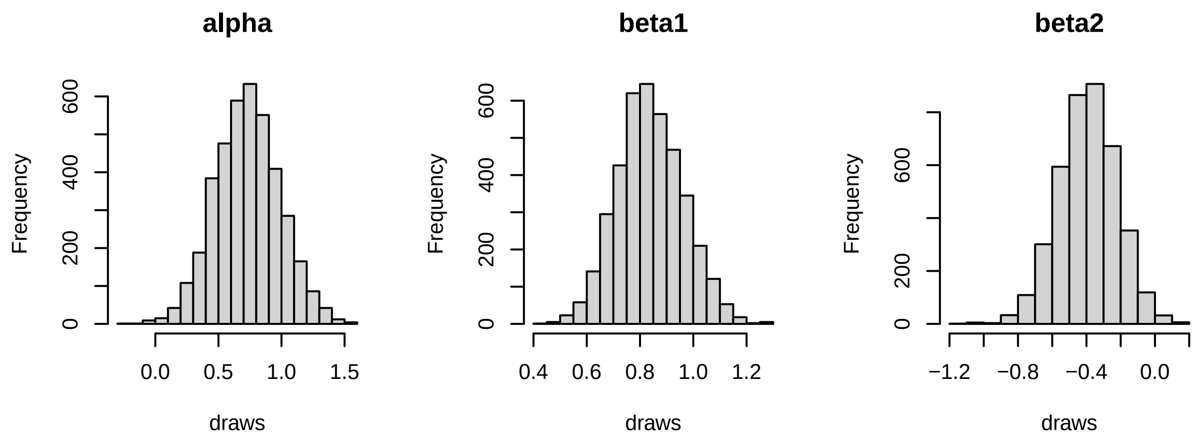
```
fit <- sampling(logi_reg,
                        data = dat_list,
                        chains = 4,
                        cores = 4)
sample_fit <- rstan::extract(fit)
sample_fit_pi <- fit %>% spread_draws(prob[condition])
```

Since x_1 and x_2 have been scaled, we need to map the estimated coefficients back to the original scale of the data:

```
alpha <- sample_fit$alpha - (sample_fit$beta1 * (mean(x1)*0.5)/sd(x1)) - (sample_fit$beta2 * (mean(x2)*
beta1 <- sample_fit$beta1 * (0.5/sd(x1))
beta2 <- sample_fit$beta2 * (0.5/sd(x2))
```

Now we can visualize their marginal posterior:

```r
par(mfrow=c(1,3), pty="s")
hist(alpha, main = "alpha", xlab = "draws")
hist(beta1, main = "beta1", xlab = "draws")
hist(beta2, main = "beta2", xlab = "draws")
```



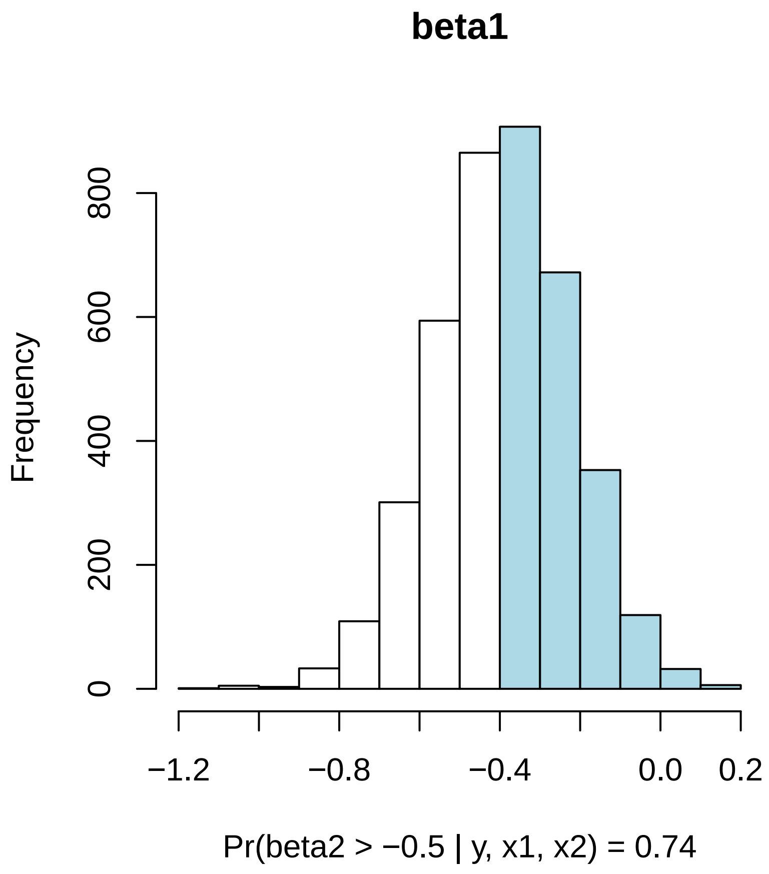
```r
par(mfrow=c(1,1), pty="m")
```

Calculating probabilities and Credible Intervals (CrI)

Similarly to what we have seen for Normal linear models, we can use the marginal posterior distribution on α, β_1 and β_2 to calculate probabilities and CrI. For example, suppose that we want to know that is the probability of β_2 being larger than, say -0.5, we can obtain this value by averaging the count of posterior parameter values $I(\beta_j^s > 1)$:

```r
thr <- -0.5
prob <- mean(beta2>thr)

h_data <- hist(beta2, plot = F)
h_area <- cut(h_data$breaks, c(-Inf, thr, Inf))
```

```r
par(pty="s")
plot(h_data, col = c("white", "lightblue")[h_area], main = "beta1",
     xlab = paste0("Pr(beta2 > ", thr, " | y, x1, x2) = ", round(prob, 2)))
```



```r
par(pty="m")
```

To calculate CrI, we simply take the corresponding quantile values:

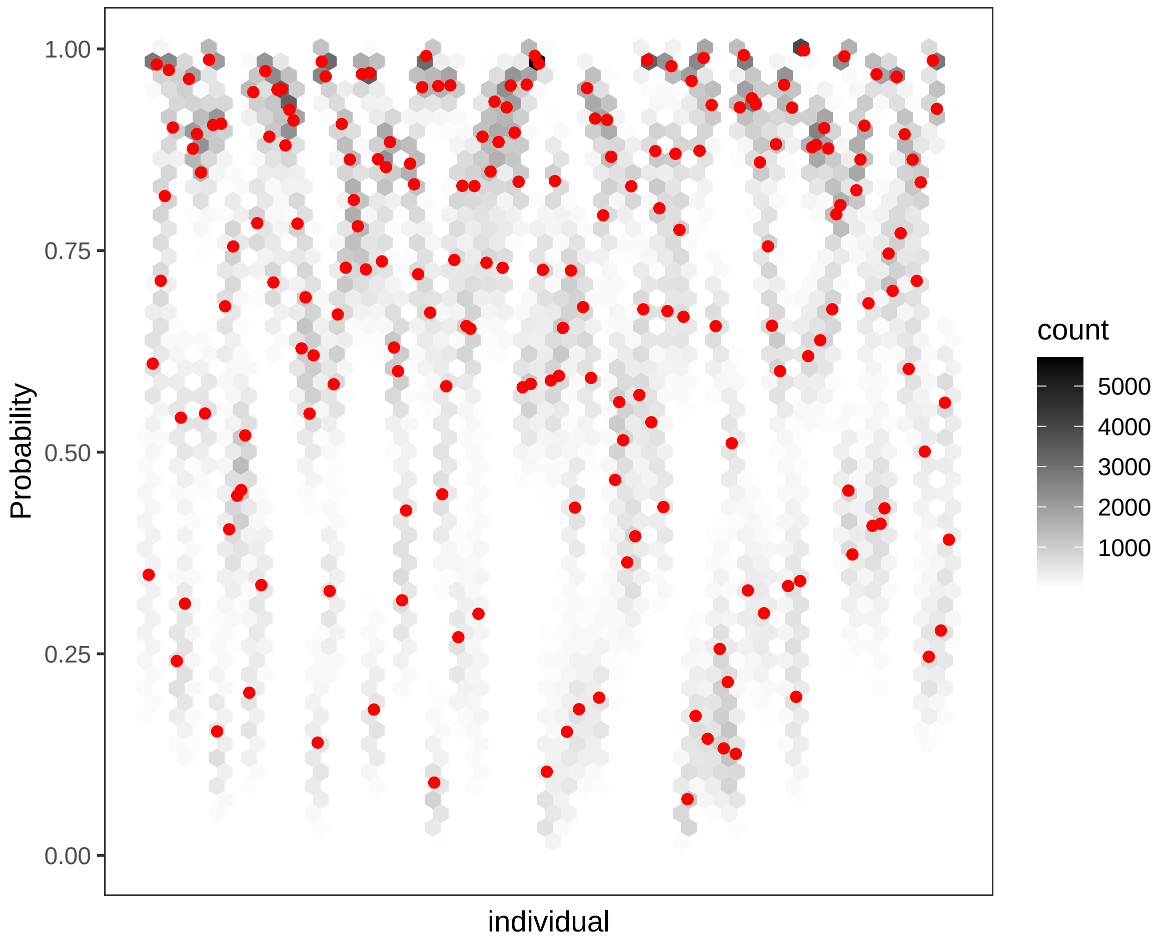
```r
rbind(quantile(alpha, c(.025, .975)),
      quantile(beta1, c(.025, .975)),
      quantile(beta2, c(.025, .975)))
```

```
##            2.5%       97.5%
## [1,]  0.2387637  1.23988830
## [2,]  0.6060762  1.08339555
## [3,] -0.7375202 -0.06968921
```

Model assessment: Posterior Predictive Checks (PoPC)

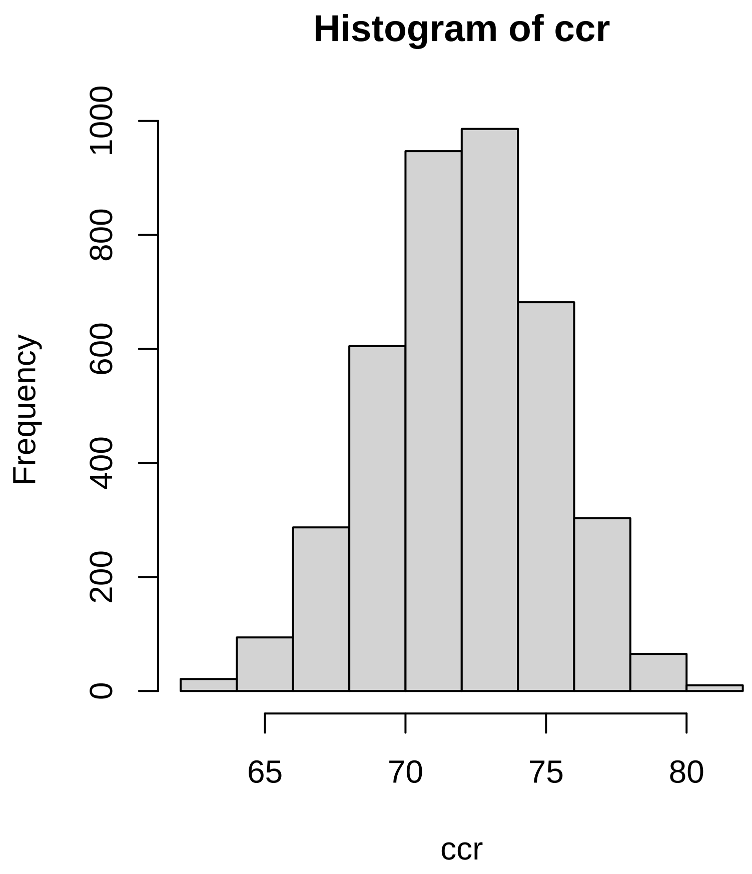
Checking model fit through the PoPC is also similar to what we have seen for linear regression. First, let's have a look at the posterior predicted probabilities:

```r
sample_fit_pi %>%
  ggplot(aes(x = condition, y = prob)) +
  geom_hex(bins = 50) +
  geom_point(data = sample_fit_pi %>%
               group_by(condition) %>%
               summarise(m_prob = mean(prob)),
             aes(x = condition, y = m_prob),
             col = "red") +
  scale_fill_gradient(low = "white", high = "black") +
  theme_bw() +
  labs(x = "individual",
       y = "Probability") +
  theme(aspect.ratio = 1,
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        panel.grid = element_blank())
```



While this plot is nice, it does not say much about how well the model fitted the data, unless we knew the exact probability values that generated the data. Therefore, in logistic regression studies, one can use different metrics such as the correct classification rate (CCR). For each posterior sample $\tilde{y}^s$, we count the number of y_i that the model correctly predicted to one or zero and divide the amount by N. We finally visualize the distribution of the CCR for each $s \in 1, \ldots, S$:

```r
ccr <- numeric(length = nrow(sample_fit$y_pred))
for(s in 1:nrow(sample_fit$y_pred)) {

  ccr[s] <- sum(diag(table(sample_fit$y_pred[s,], y)))/sum(table(sample_fit$y_pred[s,], y))*100

}

par(pty="s")
hist(ccr)
```

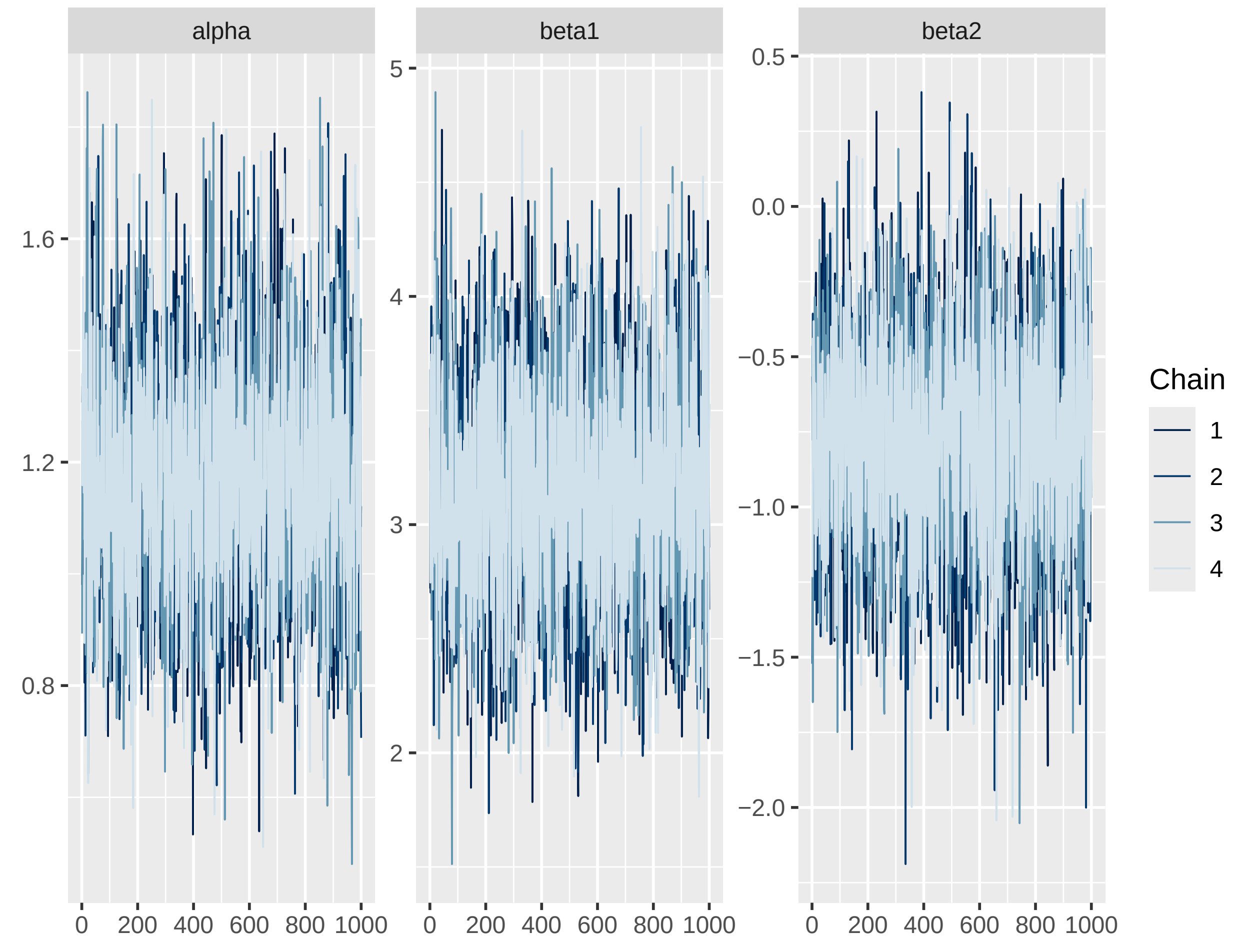


```r
par(pty="m")
```

Convergence Diagnostics

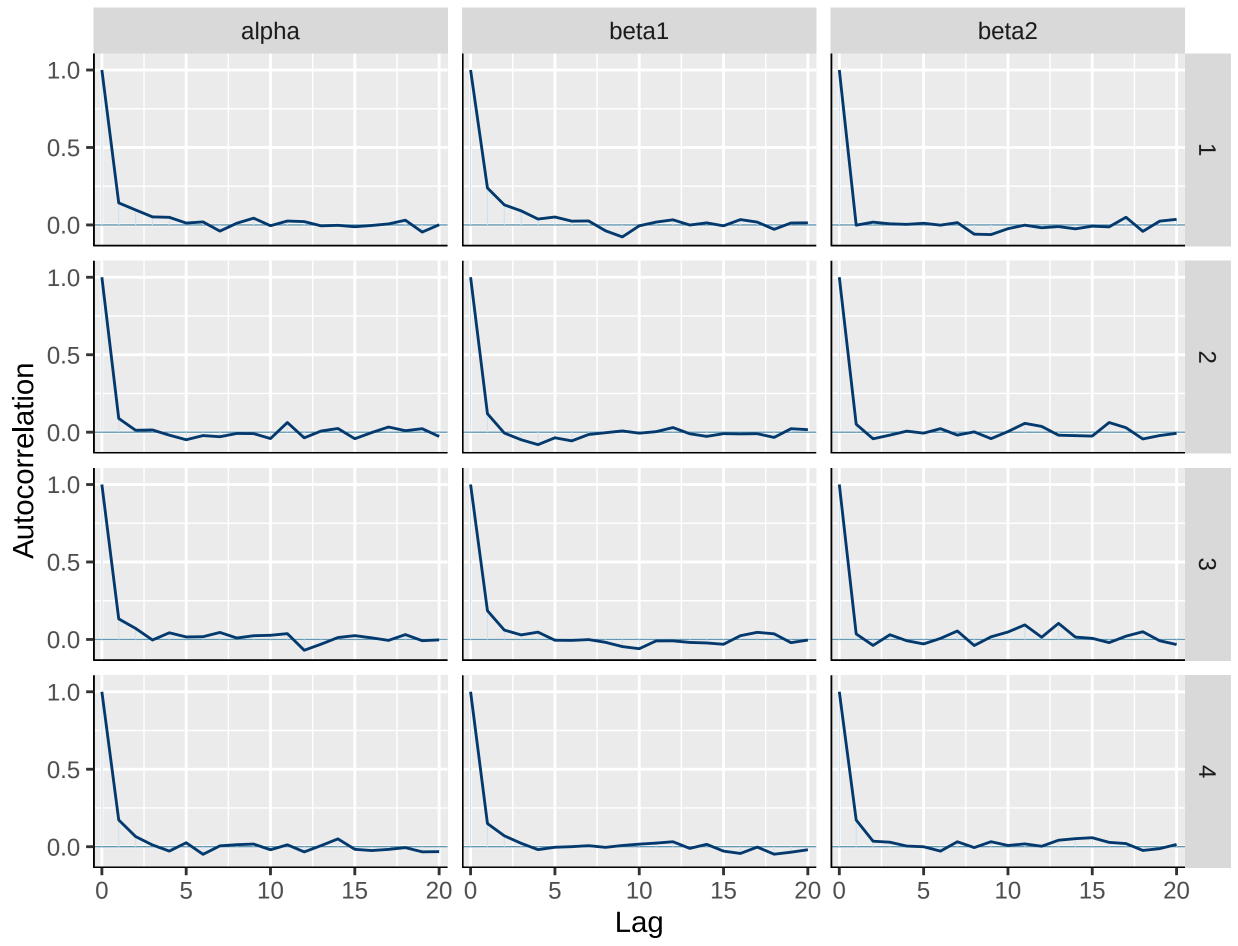
Trace Plots

```r
params <- c("alpha", "beta1", "beta2")
mcmc_trace(fit, pars = params)
```



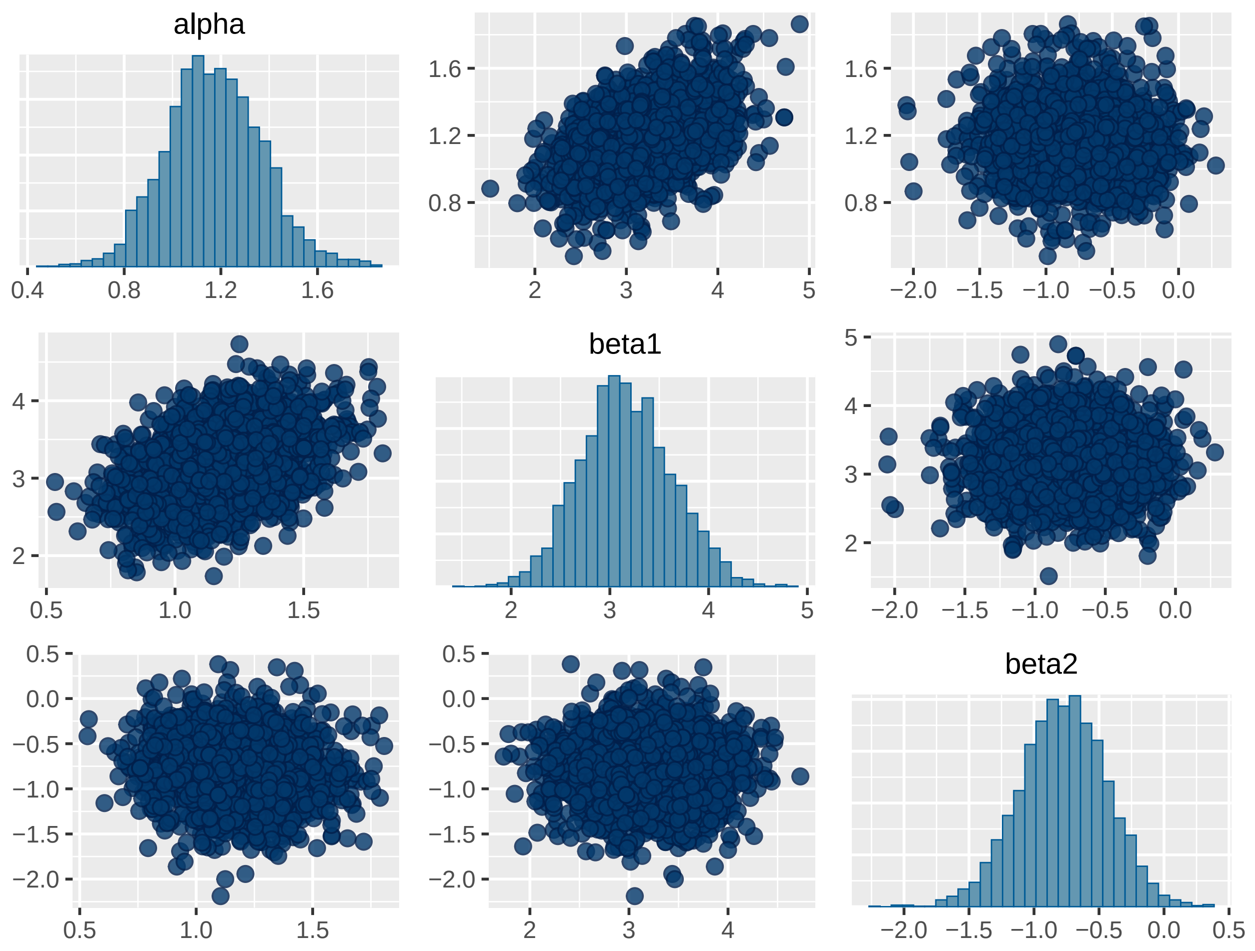
Autocorrelation Plots

```r
mcmc_acf(fit, pars = params)
```



Pair Plots

```
mcmc_pairs(fit, pars = params)
```



Effective Sample Size and R-measure

```
neff_ratio(fit, pars = params)
```

```
##     alpha     beta1     beta2
## 0.6902563 0.6526136 0.8766036
```

```
rhat(fit, pars = params)
```

```
##     alpha     beta1     beta2
## 0.9997749 1.0005347 1.0002567
```