Binomial logit t simulation Stan implementation

Divan A. Burger, Sean van der Merwe, and Emmanuel Lesaffre

2025-05-16

Table of contents

Introduction

This document provides introductory information and example code of how one might use Stan to build and fit a Binomial Generalized Linear Mixed Effects Model, but with additional Student-t innovations on the linear predictor.

```
options(scipen = 12)
library(tidyverse)
library(knitr)
```

Fixed effects Stan model

```
library(rstan)
mycores <- max(1,floor(parallel::detectCores(logical = FALSE)*0.8))
options(mc.cores = mycores)
rstan_options(auto_write = TRUE)
```

The model being used is a variation on the standard binomial regression model, but with a Bayesian implementation using objective priors and a Student-t distribution.

```
// This Stan block defines a binomial t regression model, by Sean van der Merwe, UFS
data {
  int<lower = 1> n_total;                          // number of observations in total
  int<lower = 1> m;                          // binomial maximum parameter
  int<lower = 0, upper = m> y[n_total];    // observations
  int<lower = 1> n_var;                 // number of explanatory variables, including intercep
  matrix[n_total, n_var] X;         // explanatory matrix, starting with intercept(1s)
}
// The parameters of the model
parameters {
  real<lower = 0> sigma;                    // error scale parameter
  real<lower = 2> nu;                       // error degrees of freedom
  vector[n_var] beta;                       // coefficients
  vector[n_total] alpha;                    // intermediate (nuisance) parameter
}
model {
  y ~ binomial_logit(m, alpha);             // likelihood
  alpha ~ student_t(nu, X*beta, sigma);
  target += log(nu) - 3*log(nu + 1) - log(sigma);   // priors
}
```

Base sample generation

```
rbin_t <- function(n, m, mu, s, nu) {
  t_values <- rt(n, nu)*s + mu
  rbinom(n, m, plogis(t_values))
}

r_bin_t_data <- function(p) {
```

```
  X <- cbind(matrix(1, p$n, 1), rnorm(p$n))
  Z <- rnorm(p$n_sbj, 0, p$tau)
  sbj <- sample(1:p$n_sbj, p$n, TRUE)
  y <- rbin_t(p$n, p$m, X %*% p$beta + Z[sbj], p$s, p$nu)
  list(n_total = p$n, m = p$m, y = y, n_var = length(p$beta), X = X, n_sbj = p$n_sbj, sbj = s
}
```

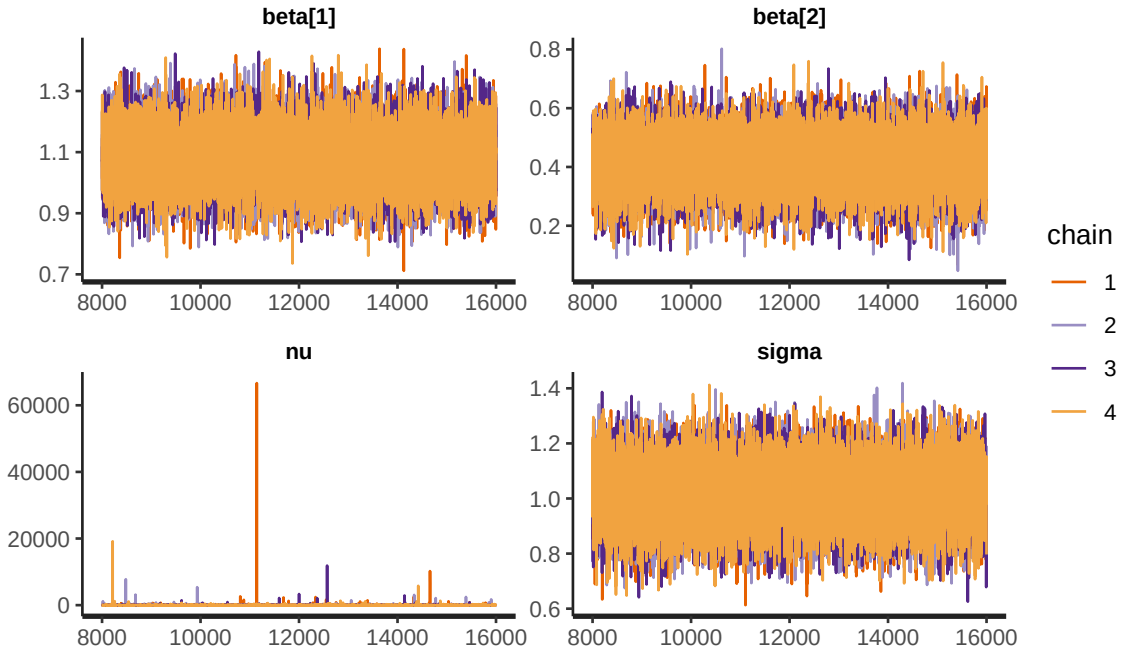
```
params <- list(
  n = 200,
  m = 30,
  beta = c(1, 0.5),
  s = 1,
  nu = 6,
  tau = 0,
  n_sbj = 1
)
smpl <- r_bin_t_data(params)
```

Posterior simulation

```
fit1 <- bin_t_model |> sampling(smpl,
                                pars = c('alpha'), include = FALSE,
                                iter = 16000, chains = mycores)
```

Evaluation

```
fit1 |> traceplot(pars = c('beta', 'nu', 'sigma'))
```




```
summary(fit1, pars = c('beta', 'nu', 'sigma'))$summary
```

```
              mean      se_mean            sd      2.5%       25%       50%
beta[1]  1.0837243 0.0004171409    0.08845227 0.9118969 1.0237860 1.0826017
beta[2]  0.4146311 0.0003937648    0.08508309 0.2463720 0.3576294 0.4148821
nu      21.2434332 2.6894982704 420.69732200 2.8279368 4.8078668 7.0211200
sigma    1.0126803 0.0008333114    0.10448535 0.8105439 0.9414000 1.0123514
              75%      97.5%    n_eff      Rhat
beta[1]  1.1435417  1.2576679 44962.71 0.9999565
beta[2]  0.4724247  0.5811322 46688.80 0.9999097
nu      11.6098155 66.7767855 24467.91 0.9999715
sigma    1.0836418  1.2167056 15721.58 1.0000329
```

Stan model with random intercept only

```
library(rstan)
mycores <- max(1,floor(parallel::detectCores(logical = FALSE)*0.8))
options(mc.cores = mycores)
rstan_options(auto_write = TRUE)
```

The model being used is a variation on the standard binomial regression model, but with a Bayesian implementation using objective priors and a Student-t distribution.

```
// This Stan block defines a binomial t regression model, by Sean van der Merwe, UFS
data {
  int<lower = 1> n_total;                        // number of observations in total
  int<lower = 1> m;                          // binomial maximum parameter
  int<lower = 0, upper = m> y[n_total];    // observations
  int<lower = 1> n_var;                 // number of explanatory variables, including intercep
  matrix[n_total, n_var] X;         // explanatory matrix, starting with intercept(1s)
  int<lower = 2> n_sbj;             // number of subjects
  int<lower = 1, upper = n_sbj> sbj[n_total]; // subject membership number
}
// The parameters of the model
parameters {
  real<lower = 0> sigma;                    // error scale parameter
  real<lower = 2> nu;                       // error degrees of freedom
  vector[n_var] beta;                       // coefficients
  vector[n_total] alpha;                    // intermediate (nuisance) parameter
  vector[n_sbj] sbj_int;                    // subject intercept
  real<lower = 0> tau_sbj;                  // between subject standard deviation
}
transformed parameters {
  vector[n_total] mu;
  mu = X*beta + sbj_int[sbj];
}
model {
  y ~ binomial_logit(m, alpha);             // likelihood
  alpha ~ student_t(nu, mu, sigma);
  sbj_int ~ normal(0, tau_sbj);
  target += log(nu) - 3*log(nu + 1) - log(sigma) - log(tau_sbj);   // priors
}
```

```
params <- list(
  n = 200,
  m = 30,
  beta = c(1, 0.5),
  s = 1,
  nu = 6,
  tau = 0.5,
  n_sbj = 10
)
smpl <- r_bin_t_data(params)
```

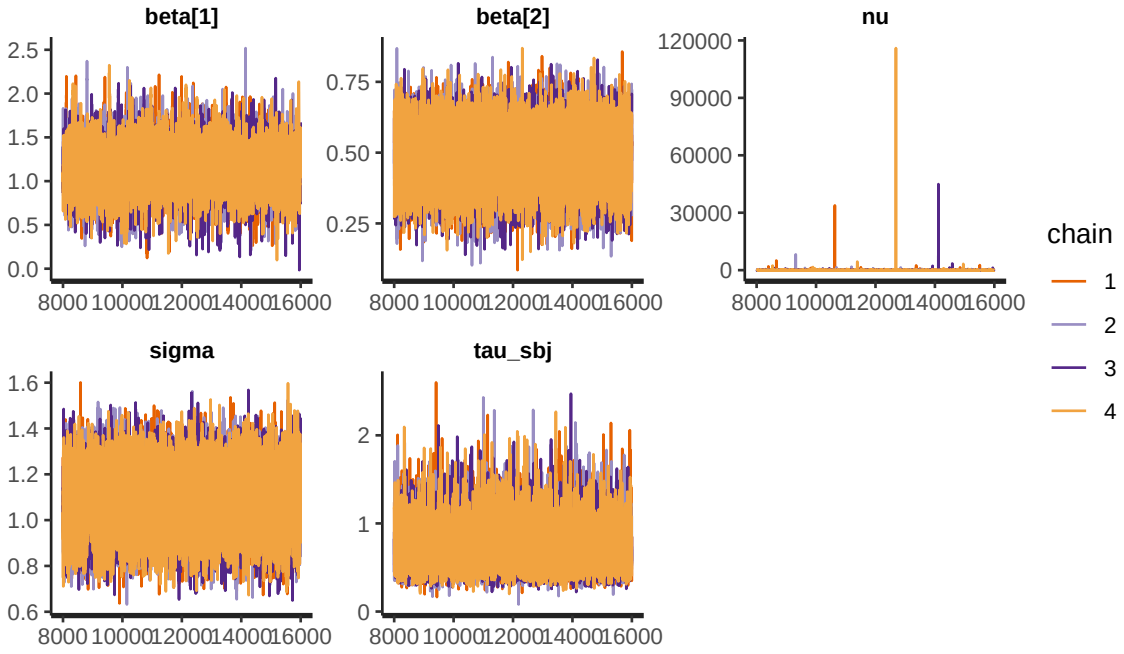
Posterior simulation

```
fit <- bin_t_model_rint |> sampling(smpl,
                                    pars = c('alpha', 'mu'), include = FALSE,
                                    iter = 16000, chains = mycores)
```

```
fit |> saveRDS("C:/temp/Simulations/bin_t_rint_stan_simulation.rds")
```

Evaluation

```
fit |> traceplot(pars = c('beta', 'nu', 'sigma', 'tau_sbj'))
```




```
summary(fit, pars = c('beta', 'nu', 'sigma', 'tau_sbj'))$summary
```

```
              mean      se_mean           sd      2.5%       25%       50%
beta[1]  1.1563283 0.0033044074   0.25408431 0.6426838 0.9994006 1.1551933
beta[2]  0.4795249 0.0005656486   0.09608752 0.2920067 0.4144875 0.4788924
nu      20.3874365 4.6968852213 741.15214832 2.2806730 3.5659077 5.0828092
sigma    1.0727852 0.0014779195   0.13251228 0.8228467 0.9798467 1.0704100
tau_sbj  0.7141516 0.0018555174   0.23057496 0.3764993 0.5545284 0.6766158
              75%      97.5%     n_eff      Rhat
beta[1] 1.3145870  1.6596277  5912.464 1.0008166
beta[2] 0.5443574  0.6677377 28856.293 1.0000394
nu      8.4196195 53.4823127 24899.742 0.9999957
sigma   1.1633928  1.3368114  8039.161 1.0003988
tau_sbj 0.8312491  1.2759265 15441.669 1.0000642
```

Adding random slopes

```
// This Stan block defines a binomial t regression model, adapted from
// code generated by the brms package
functions {
 /* compute correlated group-level effects
  * Args:
  *   z: matrix of unscaled group-level effects
  *   SD: vector of standard deviation parameters
  *   L: cholesky factor correlation matrix
  * Returns:
  *   matrix of scaled group-level effects
  */
  matrix scale_r_cor(matrix z, vector SD, matrix L) {
    // r is stored in another dimension order than z
    return transpose(diag_pre_multiply(SD, L) * z);
  }
}
data {
  int<lower=1> N;  // total number of observations
  int<lower = 1> m;   // binomial maximum parameter
  int<lower = 0, upper = m> Y[N];  // response variable
  int<lower=1> K;  // number of population-level effects
  matrix[N, K] X;  // population-level design matrix
  int<lower=1> Kc;  // number of population-level effects after centering
  // data for group-level effects of ID 1
  int<lower=1> N_1;  // number of grouping levels
```

```
  int<lower=1> M_1;  // number of coefficients per level
  array[N] int<lower=1> J_1;  // grouping indicator per observation
  // group-level predictor values
  vector[N] Z_1_1;
  vector[N] Z_1_2;
  int<lower=1> NC_1;  // number of group-level correlations
}
transformed data {
  matrix[N, Kc] Xc;  // centered version of X without an intercept
  vector[Kc] means_X;  // column means of X before centering
  for (i in 2:K) {
    means_X[i - 1] = mean(X[, i]);
    Xc[, i - 1] = X[, i] - means_X[i - 1];
  }
}
parameters {
  vector[Kc] b;  // regression coefficients
  real Intercept;  // temporary intercept for centered predictors
  real<lower=0> sigma;  // dispersion parameter
  vector<lower=0>[M_1] sd_1;  // group-level standard deviations
  matrix[M_1, N_1] z_1;  // standardized group-level effects
  cholesky_factor_corr[M_1] L_1;  // cholesky factor of correlation matrix
  real<lower=1> nu;  // degrees of freedom or shape
  vector[N] alpha;                    // intermediate (nuisance) parameter
}
transformed parameters {
  matrix[N_1, M_1] r_1;  // actual group-level effects
  // using vectors speeds up indexing in loops
  vector[N_1] r_1_1;
  vector[N_1] r_1_2;
  real lprior = 0;  // prior contributions to the log posterior
  // compute actual group-level effects
  r_1 = scale_r_cor(z_1, sd_1, L_1);
  r_1_1 = r_1[, 1];
  r_1_2 = r_1[, 2];
  lprior += student_t_lpdf(Intercept | 3, 23, 5.9);
  lprior += student_t_lpdf(sigma | 3, 0, 5.9)
    - 1 * student_t_lccdf(0 | 3, 0, 5.9);
  lprior += gamma_lpdf(nu | 2, 0.1)
    - 1 * gamma_lccdf(1 | 2, 0.1);
  lprior += student_t_lpdf(sd_1 | 3, 0, 5.9)
    - 2 * student_t_lccdf(0 | 3, 0, 5.9);
```

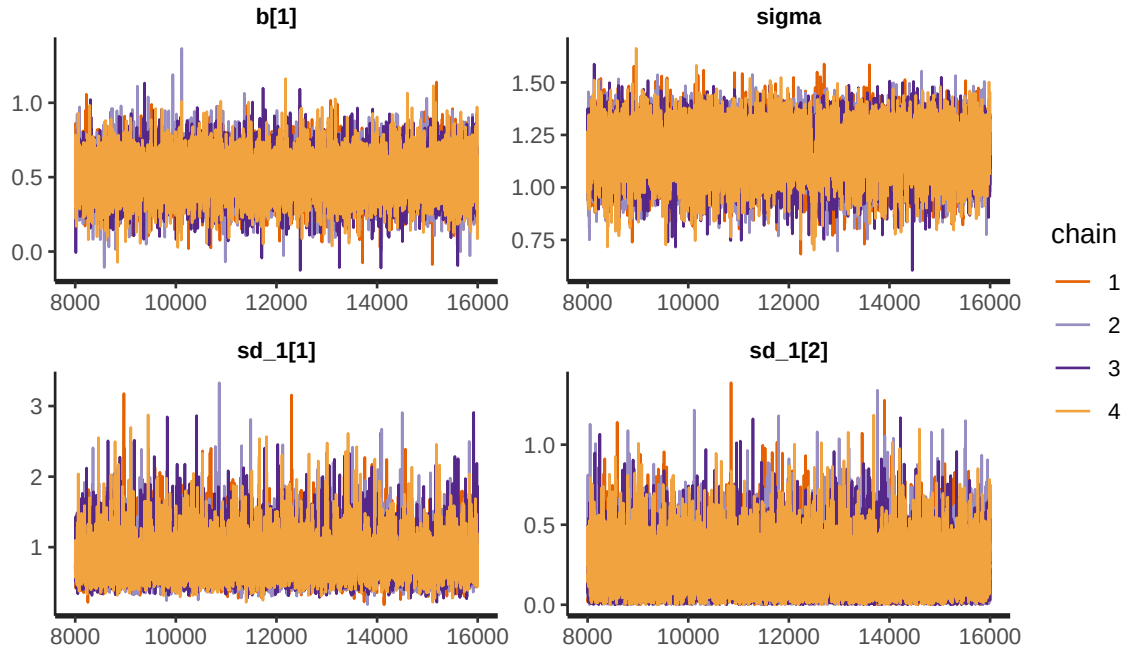
```
  lprior += lkj_corr_cholesky_lpdf(L_1 | 1);
}
model {
  // initialize linear predictor term
  vector[N] mu = rep_vector(0.0, N);
  mu += Intercept + Xc * b;
  for (n in 1:N) {
    // add more terms to the linear predictor
    mu[n] += r_1_1[J_1[n]] * Z_1_1[n] + r_1_2[J_1[n]] * Z_1_2[n];
  }
  Y ~ binomial_logit(m, alpha);            // likelihood
  alpha ~ student_t(nu, mu, sigma);
  target += lprior;
  target += std_normal_lpdf(to_vector(z_1));
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = Intercept - dot_product(means_X, b);
  // compute group-level correlations
  corr_matrix[M_1] Cor_1 = multiply_lower_tri_self_transpose(L_1);
  vector<lower=-1,upper=1>[NC_1] cor_1;
  // extract upper diagonal of correlation matrix
  for (k in 1:M_1) {
    for (j in 1:(k - 1)) {
      cor_1[choose(k - 1, 2) + j] = Cor_1[j, k];
    }
  }
}
```

```
stan_data <- list(N = smpl$n_total, m = smpl$m, Y = smpl$y, K = smpl$n_var,
                  X = smpl$X, Kc = smpl$n_var-1, N_1 = smpl$n_sbj, M_1 = smpl$n_var,
                  J_1 = smpl$sbj, Z_1_1 = smpl$X[,1], Z_1_2 = smpl$X[,2],
                  NC_1 = smpl$n_var - 1)
```

```
fit <- bin_t_model_rslope |> sampling(stan_data,
                                pars = c('alpha', 'mu'), include = FALSE,
                                iter = 16000, chains = mycores)
```

Evaluation

```
fit |> traceplot(pars = c('b', 'sigma', 'sd_1'))
```




```
summary(fit, pars = c('b', 'sigma', 'sd_1'))$summary
```

```
             mean      se_mean         sd       2.5%       25%       50%
b[1]    0.5099606 0.0009652586 0.1314722 0.25265999 0.4259583 0.5088327
sigma   1.1693370 0.0011346330 0.1137405 0.93038802 1.0980840 1.1729207
sd_1[1] 0.8392523 0.0029414558 0.2864831 0.43299844 0.6423364 0.7879782
sd_1[2] 0.2236565 0.0014262953 0.1572308 0.01102975 0.1049054 0.1990957
              75%     97.5%     n_eff      Rhat
b[1]    0.5920380 0.7757086 18551.572 0.9999823
sigma   1.2463073 1.3810868 10048.924 1.0002982
sd_1[1] 0.9791324 1.5317442  9485.785 1.0003488
sd_1[2] 0.3100315 0.5984530 12152.247 1.0001754
```