Matrix form of data and parameter product in `stan`

Miao Cai miao.cai@slu.edu

2019-07-09

Contents

1 Model setting

A simple random intercept model. Let us assume that there are three predictor variables x_1, x_2, x_3, and **10 groups** ($d(i) = 1, 2, \cdots, 10$):

$$
\begin{aligned}
y_i &= \beta_{0,d(i)} + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon_i \\
\epsilon_i &\sim N(0, \sigma^2) \\
\beta_{0,d(i)} &\sim N(\mu_0, \sigma_0^2)
\end{aligned}
$$

The goal here is to write this $\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ term in a matrix multiplication form in `Stan`. Here we assume the predictor variables were generated from the following distribution:

$$
\begin{aligned}
x_1 &\sim N(3, 10) \\
x_2 &\sim \text{Gamma}(10, 10) \\
x_3 &\sim \text{Poisson}(10)
\end{aligned}
$$

The parameter settings are:

$$
\begin{aligned}
\text{Hyperparameters:} &\quad \mu_0 = 2, \sigma_0 = 5 \\
\text{Fixed parameters:} &\quad \beta_1 = 2.5, \beta_2 = 0.5, \beta_3 = -1.3 \\
\text{Variance parameter:} &\quad \sigma = 2
\end{aligned}
$$

2 Data simulation

```
set.seed(123)
# Simulation setting
d = 10 # total number of groups
D = rpois(d, 100) # sample size in each group
n = sum(D) # total sample size

# Hyper-parameters
mu0 = 2; sigma0 = 5

# Parameters
B = c(2.5, 0.5, -1.3); sigma = 2

# Random intercepts
b0 = rnorm(d, mu0, sigma0)
B0 = rep(b0, D)

# Predictor variables
id = rep(1:d, D)
x1 = rnorm(n, 3, 10)
x2 = rgamma(n, 10, 10)
x3 = rpois(n, 10)
epsilon = rnorm(n, 0, sigma)
X = matrix(c(x1, x2, x3), nrow = n)

Y = B0 + X %*% B + epsilon
df = data.frame(cbind(id, Y, X))
names(df) = c("id", "y", paste0("x", 1:3))
```

The random intercepts are:

```
df_b0 = data.frame(t(b0))
names(df_b0) = paste0("b0_", 1:d)
knitr::kable(df_b0, digits = 3)
```

b0_1	b0_2	b0_3	b0_4	b0_5	b0_6	b0_7	b0_8	b0_9	b0_10
4.004	2.553	-0.779	10.935	4.489	-7.833	5.507	-0.364	-3.339	0.91

3 Estimation with lme

```
pacman::p_load(lme4, rstanarm, brms, rstan, broom)

fit0 <- lme4::lmer(y ~ x1 + x2 + x3 + (1 | id), data = df)

broom::tidy(fit0) %>%
  knitr::kable(digits = 3)
```

```
## Warning in bind_rows_(x, .id): binding factor and character vector,
## coercing into character vector
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

term	estimate	std.error	statistic	group
(Intercept)	1.722	1.701	1.013	fixed
x1	2.507	0.006	395.025	fixed
x2	0.342	0.202	1.691	fixed
x3	-1.302	0.021	-63.400	fixed
sd_(Intercept).id	5.297	NA	NA	id
sd_Observation.Residual	1.997	NA	NA	Residual

4 Esitmation with rstanarm

```
f_rstanarm = stan_lmer(y ~ x1 + x2 + x3 + (1 | id),
                       data = df,
                       chains = 1, iter = 1000, refresh = 0)
```

4.1 Parameter estimates

```
broom::tidy(f_rstanarm, parameters = "non-varying") %>%
  knitr::kable(digits = 3)
```

term	estimate	std.error
(Intercept)	2.193	1.547
x1	2.506	0.007
x2	0.334	0.191
x3	-1.304	0.022

```
broom::tidy(f_rstanarm, parameters = "varying") %>%
  knitr::kable(digits = 3)
```

level	group	term	estimate	std.error
1	id	(Intercept)	2.118	1.563
2	id	(Intercept)	0.481	1.503
3	id	(Intercept)	-3.343	1.579
4	id	(Intercept)	9.002	1.561
5	id	(Intercept)	2.740	1.580
6	id	(Intercept)	-10.057	1.563
7	id	(Intercept)	3.447	1.502
8	id	(Intercept)	-2.332	1.577
9	id	(Intercept)	-5.379	1.624
10	id	(Intercept)	-1.153	1.568

```
broom::tidy(f_rstanarm, parameters = "hierarchical") %>%
  knitr::kable(digits = 3)
```

term	group	estimate
sd_(Intercept).id	id	5.200
sd_Observation.Residual	Residual	2.003

```
broom::tidy(f_rstanarm, parameters = "auxiliary") %>%
  knitr::kable(digits = 3)
```

term	estimate	std.error
sigma	2.003	0.045
mean_PPD	-2.915	0.085

4.2 Default priors

```r
prior_summary(f_rstanarm)
```

```
## Priors for model 'f_rstanarm'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##      **adjusted scale = 260.09
##
## Coefficients
##  ~ normal(location = [0,0,0], scale = [2.5,2.5,2.5])
##      **adjusted scale = [  6.49,206.81, 20.84]
##
## Auxiliary (sigma)
##  ~ exponential(rate = 1)
##      **adjusted scale = 26.01 (adjusted rate = 1/adjusted scale)
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details
```

5 Esitmation with brms

```
f_brms = brm(y ~ x1 + x2 + x3 + (1 | id),
             data = df, family = gaussian(),
             chains = 1, iter = 1000, refresh = 0)
```

```
## Compiling the C++ model
```

```
## Start sampling
```

5.1 Parameter estimates

```
broom::tidy(f_brms) %>%
  knitr::kable(digits = 3)
```

term	estimate	std.error	lower	upper
b_Intercept	1.363	2.118	-1.790	5.013
b_x1	2.506	0.007	2.496	2.517
b_x2	0.350	0.196	0.030	0.667
b_x3	-1.303	0.021	-1.338	-1.271
sd_id__Intercept	6.300	1.907	4.024	9.956
sigma	2.002	0.041	1.937	2.065
r_id[1,Intercept]	2.951	2.121	-0.744	6.237
r_id[2,Intercept]	1.303	2.118	-2.265	4.480
r_id[3,Intercept]	-2.545	2.124	-6.281	0.651
r_id[4,Intercept]	9.817	2.108	6.093	13.017
r_id[5,Intercept]	3.550	2.117	-0.087	6.779
r_id[6,Intercept]	-9.233	2.124	-12.923	-6.057
r_id[7,Intercept]	4.268	2.121	0.588	7.542
r_id[8,Intercept]	-1.521	2.117	-5.273	1.785
r_id[9,Intercept]	-4.623	2.124	-8.451	-1.438
r_id[10,Intercept]	-0.387	2.113	-3.958	2.731
lp__	-2121.149	3.842	-2127.760	-2115.838

5.2 Default priors

```
prior_summary(f_brms)
```

```
##                    prior     class      coef group resp dpar nlpar bound
## 1                                b
## 2                                b        x1
## 3                                b        x2
## 4                                b        x3
## 5 student_t(3, -3, 25) Intercept
## 6  student_t(3, 0, 25)          sd
## 7                               sd               id
## 8                               sd Intercept     id
## 9  student_t(3, 0, 25)     sigma
```

5.3 Return stan code using brms::stancode()

```r
stancode(f_brms)
```

```
// generated with brms 2.8.0
functions {
}
data {
  int<lower=1> N;  // number of observations
  vector[N] Y;  // response variable
  int<lower=1> K;  // number of population-level effects
  matrix[N, K] X;  // population-level design matrix
  // data for group-level effects of ID 1
  int<lower=1> N_1;
  int<lower=1> M_1;
  int<lower=1> J_1[N];
  vector[N] Z_1_1;
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
  int Kc = K - 1;
  matrix[N, K - 1] Xc;  // centered version of X
  vector[K - 1] 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;  // population-level effects
  real temp_Intercept;  // temporary intercept
  real<lower=0> sigma;  // residual SD
  vector<lower=0>[M_1] sd_1;  // group-level standard deviations
  vector[N_1] z_1[M_1];  // unscaled group-level effects
}
transformed parameters {
  // group-level effects
  vector[N_1] r_1_1 = (sd_1[1] * (z_1[1]));
}
model {
  vector[N] mu = temp_Intercept + Xc * b;
  for (n in 1:N) {
    mu[n] += r_1_1[J_1[n]] * Z_1_1[n];
  }
  // priors including all constants
  target += student_t_lpdf(temp_Intercept | 3, -3, 25);
  target += student_t_lpdf(sigma | 3, 0, 25)
    - 1 * student_t_lccdf(0 | 3, 0, 25);
  target += student_t_lpdf(sd_1 | 3, 0, 25)
    - 1 * student_t_lccdf(0 | 3, 0, 25);
  target += normal_lpdf(z_1[1] | 0, 1);
  // likelihood including all constants
  if (!prior_only) {
    target += normal_lpdf(Y | mu, sigma);
  }
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = temp_Intercept - dot_product(means_X, b);
}
```

5.4 Return stan code using brms::make_stancode()

```r
make_stancode(y ~ x1 + x2 + x3 + (1 | id), data = df, family = gaussian())
```

```
// generated with brms 2.8.0
functions {
}
data {
  int<lower=1> N;  // number of observations
  vector[N] Y;  // response variable
  int<lower=1> K;  // number of population-level effects
  matrix[N, K] X;  // population-level design matrix
  // data for group-level effects of ID 1
  int<lower=1> N_1;
  int<lower=1> M_1;
  int<lower=1> J_1[N];
  vector[N] Z_1_1;
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
  int Kc = K - 1;
  matrix[N, K - 1] Xc;  // centered version of X
  vector[K - 1] 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;  // population-level effects
  real temp_Intercept;  // temporary intercept
  real<lower=0> sigma;  // residual SD
  vector<lower=0>[M_1] sd_1;  // group-level standard deviations
  vector[N_1] z_1[M_1];  // unscaled group-level effects
}
transformed parameters {
  // group-level effects
  vector[N_1] r_1_1 = (sd_1[1] * (z_1[1]));
}
model {
  vector[N] mu = temp_Intercept + Xc * b;
  for (n in 1:N) {
    mu[n] += r_1_1[J_1[n]] * Z_1_1[n];
  }
  // priors including all constants
  target += student_t_lpdf(temp_Intercept | 3, -3, 25);
  target += student_t_lpdf(sigma | 3, 0, 25)
    - 1 * student_t_lccdf(0 | 3, 0, 25);
  target += student_t_lpdf(sd_1 | 3, 0, 25)
    - 1 * student_t_lccdf(0 | 3, 0, 25);
  target += normal_lpdf(z_1[1] | 0, 1);
  // likelihood including all constants
  if (!prior_only) {
    target += normal_lpdf(Y | mu, sigma);
  }
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = temp_Intercept - dot_product(means_X, b);
}
```

6 Self-define R code