

From BRMS to Stan

Mitzi Morris

2025-05-07

Stan Development Team



My Questions for You

- What kinds of models do you use for research? For teaching?
- BRMS
 - How often do you use it: never, sometimes, always?
 - What do you like/dislike/find confusing about BRMS?
- Stan
 - Have you run Stan models through an interface, RStan, CmdStanR, CmdStanPy?
 - Have you written / tried to write or modify Stan models?
 - What do you like/dislike/find confusing about Stan?

Talk Outline

- Brief discussion of BRMS
- Stan from 0 to 60 in 30 minutes
 - Model 1: “Hello World”
 - Model 2: black diamond - varying-slope / varying intercept
- Notebook: github.com/mitzimorris/brms2stan/brms2stan.qmd

BRMS: Bayesian Regression and Multilevelmodeling in Stan



brms

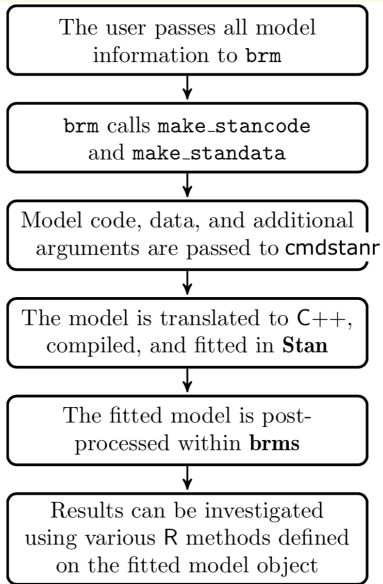
Bayesian regression models using Stan

- Fit Bayesian generalized linear / non-linear multivariate multilevel models using Stan.
- Specify model via lme4-like formula syntax.

```
sleep_mlm_brmsfit <-  
  brm(Reaction ~ Days + (Days|Subject),  
      data = sleepstudy)
```

<https://paul-buerkner.github.io/brms/>

BRMS Processing



BRMS Processing

- Pass all information to BRMS

```
brm(Reaction ~ Days + (Days|Subject), data = sleepstudy)
```

- Get results

```
Family: gaussian
```

```
Links: mu = identity; sigma = identity
```

```
Formula: Reaction ~ Days + (Days | Subject)
```

```
Data: sleepstudy (Number of observations: 180)
```

Multilevel Hyperparameters:

```
~Subject (Number of levels: 18)
```

	Estimate	Est.Error
sd(Intercept)	27.27	7.03
sd(Days)	6.60	1.51
cor(Intercept,Days)	0.07	0.30

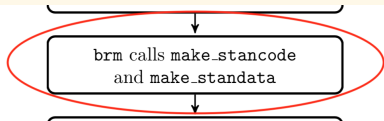
Regression Coefficients:

	Estimate	Est.Error
Intercept	251.31	7.40
Days	10.45	1.70
sigma	25.87	1.52

Reasons to Use BRMS

- Familiar syntax (for R users).
- Handles many common models: IRT, zero-inflated, additive distributional, phylogenetic, missing data imputation.
 - See [Vignettes and Articles](#)
- Sound and robust implementations, e.g.
 - Zero-centers predictors (like RStanARM).
 - Makes varying slope/ varying intercept models easy.
- Large community of users
 - Majority of questions on Stan mailing list are BRMS.
 - Many helpful community experts to answer them!

BRMS Drawbacks



- Generated Stan code is hard to read/modify.
- Priors used may be suboptimal.

See [github.com/stan-dev/stan](https://github.com/stan-dev/stan/wiki/prior-choice-recommendations) wiki page [Prior Choice Recommendations](https://github.com/stan-dev/stan/wiki/prior-choice-recommendations)

A screenshot of a web browser showing the GitHub page for "Prior Choice Recommendations" under the "stan-dev / stan" repository. The browser address bar shows the URL "github.com/stan-dev/stan/wiki/prior-choice-recommendations". The GitHub navigation bar includes links for Code, Issues (133), Pull requests (12), Discussions, Actions, Projects, Wiki (highlighted), Security, and Insights. The page title is "Prior Choice Recommendations" and it notes "Aki Vehtari edited this page last week · 58 revisions". The main content area starts with the heading "5 levels of priors".

BRMS Priors

Prior Definitions for brms Models

```
> prior_summary(sleep_mlm_brmsfit)
```

	prior	class	coef	group	resp	dpar	nlpar	lb	ub	source
	(flat)	b								default
	(flat)	b	Days							(vectorized)
student_t(3, 288.7, 59.3)		Intercept								default
lkj_corr_cholesky(1)		L								default
lkj_corr_cholesky(1)		L		Subject						(vectorized)
student_t(3, 0, 59.3)		sd						0		default
student_t(3, 0, 59.3)		sd		Subject				0		(vectorized)
student_t(3, 0, 59.3)		sd	Days	Subject				0		(vectorized)
student_t(3, 0, 59.3)		sd	Intercept	Subject				0		(vectorized)
student_t(3, 0, 59.3)		sigma						0		default

- To change the priors you must understand the model structure.
- If you understand the model structure, you can code it in Stan!

Changing BRMS Priors

```
priors <- c(
  set_prior("normal(250, 50)", class = "Intercept"),
  set_prior("normal(10, 10)", class = "b"),
  set_prior("exponential(1)", class = "sigma"),
  set_prior("exponential(1)", class = "sd"),
  set_prior("lkj_corr_cholesky(2)", class = "cor"),
)
sleep_mlm_brmsfit <- brm(Reaction ~ Days + (Days|Subject),
  data = sleepstudy,
  prior = priors)
```

Easy to get wrong - cf. questions on discourse.mc-stan.org, e.g.: recent discussion

Equally complicated - custom families, custom link functions.

From BRMS to Stan

When model specification in BRMS is long / complicated / not quite possible - try Stan.

- How do you write a Stan program?
- BRMS Formula provides structure of regression and data inputs.
- Mapping BRMS elements to a Stan program
 - `data` block defines all data inputs - outcomes and predictors, plus dimensions.
 - `parameters` block defines all distributional parameters.
 - `model` block specifies the likelihood and priors.

Stan Program Structure

A Stan program consists of one or more named program blocks, strictly ordered.

```
functions {  
  // declare, define functions  
} data {  
  // declare input data  
} transformed data {  
  // transform inputs, define program data  
} parameters {  
  // declare (continuous) parameters  
} transformed parameters {  
  // define derived parameters  
} model {  
  // compute the log joint distribution  
} generated quantities {  
  // define quantities of interest  
}
```

Stan Program Execution

- NUTS-HMC sampler discretizes Hamiltonian dynamics into steps.
 - 1 sampler iteration is comprised of many steps.
 - Good NUTS-HMC sampler intro: [Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo](#)
- data, transformed data blocks - executed once on startup - *cheap*
- parameters -
 - on startup: initialize parameters
 - at every step of inference algorithm: validate constraints
- transformed parameters, model blocks - executed every *step* of the sampler - *expensive*
- generated quantities - executed every *iteration* of the sampler - *less expensive*

Stepwise Model Development: Hello, World!

- A "Hello, World!" program is the name given to the first, simplest possible program written when learning a new programming language.
 - **Pro tip: always start with “Hello, World!”**
- End goal is a efficient and maintainable multi-level model
 - `Reaction ~ Days + (Days|Subject)`
- Initial goal is a simple linear model - ignore difference between subjects
 - BRMS formula: `Reaction ~ Days`
- Simple linear regression: $y \sim \alpha + \beta * x$,
 - where reaction time is y and day is x .

Initial Stan Model sleep_simple.stan

```
data {  
  int<lower=0> N;  
  vector[N] day;  
  vector[N] y;  // reaction time  
}  
parameters {  
  real alpha;  // intercept  
  real b_day;  // slope  
  real<lower=0> sigma; // residual standard deviation  
}  
model {  
  y ~ normal(alpha + day * b_day, sigma);  
  alpha ~ normal(250, 50);  // informed prior for human reaction times in ms  
  b_day ~ normal(10, 10);  // weakly informed prior for per-day effect  
  sigma ~ normal(0, 10);  // very weakly informative prior  
}
```

Improved Stan Model sleep_simple.stan - Mean-center Data

This decreases the correlation between the intercept and slope.
(BRMS, RStanArm do this automatically.)

```
data {  
  int<lower=0> N;   vector[N] day;   vector[N] y;  
}  
transformed data {  
  real day_mean = mean(day);  
  vector[N] day_centered = day - day_mean;    // mean-center data  
}  
parameters {  
  real alpha;  real b_day;  real<lower=0> sigma;  
}  
model {  
  y ~ normal(alpha + day_centered * b_day, sigma);  
  alpha ~ normal(250, 50); b_day ~ normal(10, 10); sigma ~ normal(0, 10);  
}  
generated quantities {  
  real b_intercept = alpha - b_day * day_mean;    // recover intercept  
}
```


BRMS stancode for Reaction ~ Days

```
> stancode(sleep_simple_brmsfit)
// generated with brms 2.22.0
functions {
}
data {
  int<lower=1> N; // total 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
  int<lower=1> Kc; // number of population-level effects
  int prior_only; // should the likelihood be ignored?
}
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
}
transformed parameters {
  real lprior = 0; // prior contributions to the log posterior
  lprior += normal_lpdf(b | 10, 10);
  lprior += normal_lpdf(Intercept | 250, 50);
  lprior += normal_lpdf(sigma | 0, 10)
    - 1 * normal_lccdf(0 | 0, 10);
}
model {
  if (!prior_only) { // likelihood including constants
    target += normal_id_glm_lpdf(Y | Xc, Intercept, b, sigma);
  }
  target += lprior; // priors including constants
}
generated quantities { // actual population-level intercepts
  real b_Intercept = Intercept - dot_product(means_X, b);
}
```

Multilevel Models

- Account for the structure in the data.
 - Individual observations are drawn from one or more common populations.
 - Estimate population-level mean, variance as well as individual-level mean, variance.
 - Population level parameters inform priors for individual level parameters.
- Provide **partial pooling** of information.
 - The hierarchical prior controls the pooling between levels.
 - Similar data across levels → Low hierarchical variance → strong pooling.
When hierarchical variance approaches 0, complete pooling.
 - Dissimilar data across levels → High hierarchical variance → weak pooling.
When hierarchical variance approaches ∞ , no pooling.

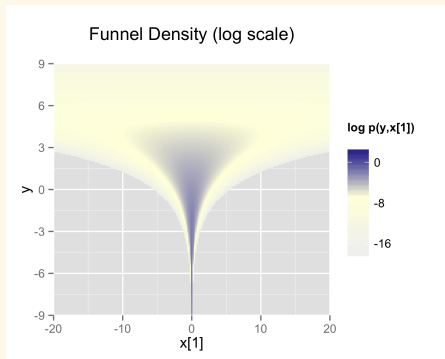
Multilevel models

- Expand Stan model:
 - A per-subject parameter vector β (`b_subj`) is *multivariate normal*
 - `b_subj ~ multi_normal(mu_subj, sigma_subj)`
 - `mu_subj` is vector, of length number of subjects,
`sigma_subj` is covariance matrix.
- *Problems*
 - Stan distribution `multi_normal` - requires inverting covariance matrix at every evaluation - computationally expensive.
 - Two sources of variance: `sigma` and hierarchical variance `sigma_subj`
difficult to estimate from small number of observations per group.
Without sufficient data → **funnel distribution**
- **Solution:** Reparameterization, following code example in Stan User's Guide section *Hierarchical models and the non-centered parameterization*

Multilevel models and the Funnel

Neal's Funnel: extreme example of a challenging hierarchical prior

$$p(y, x) = \text{normal}(y \mid 0, 3) \times \prod_{n=1}^9 \text{normal}(x_n \mid 0, \exp(y/2)).$$



To explore neck of the funnel:

- small steps on x -axis, large steps on y -axis.

To explore mouth of the funnel:

- large steps on x -axis, small steps on y -axis.

But stepsize is same for all axes;
cannot adequately sample either.

Prior distribution; no data.

Funnel Example: Stan Implementations

Centered parameterization

- “Natural” parameterization.

```
parameters {  
  real y;  
  vector[9] x;  
}  
  
model {  
  y ~ normal(0, 3);  
  x ~ normal(0, exp(y/2));  
}
```

Note: minimal model, no predictors

- variable y == hierarchical variance
- variable x == group-level covariate

Non-centered parameterization

- Parameters block: declare standardized parameters.
- Transformed parameters: declare *variables* add offset (location), multiply by scale.

```
parameters {  
  real y_raw;  
  vector[9] x_raw;  
}  
  
transformed parameters {  
  // offset is 0, just multiply by scale  
  real y = 3.0 * y_raw;  
  vector[9] x = exp(y/2) * x_raw;  
}  
  
model {  
  y_raw ~ std_normal(); // y ~ normal(0, 3)  
  x_raw ~ std_normal(); // x ~ normal(0, exp(y/2))  
}
```

Stan's Affine Transform

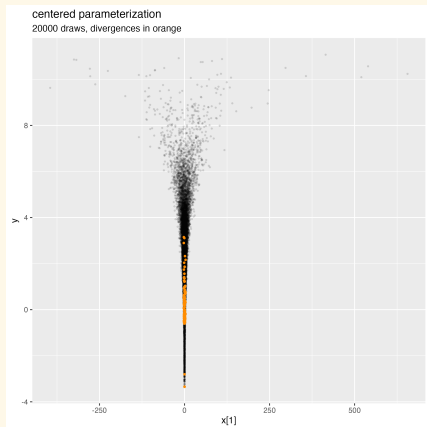
Stan's [Affinely Transformed Scalar](#) can be applied element-wise to vector x to facilitate the [non-centered parameterization](#)

```
parameters {  
  real<multiplier = 3.0> y;  
  vector<multiplier = exp(y/2)>[9] x;  
}  
  
model {  
  y ~ normal(0, 3)  
  x ~ normal(0, exp(y/2))  
}
```

Compare Funnel Fits

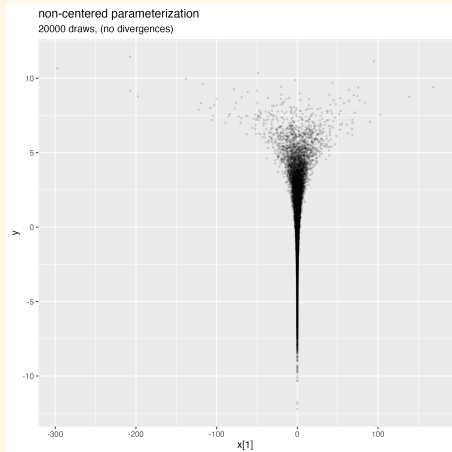
Centered parameterization

- Cannot explore neck of funnel
many divergences (in orange)



Non-centered parameterization

- Explores further; no divergences



BRMS to Stan

- “Hello, World!” model - formula `Reaction ~ 1 + Days`
 - coded as distribution statement:
`y ~ normal(alpha + day_centered * b_day, sigma);`
- Target model - formula `Reaction ~ 1 + Days + (1 + Days | Subject)`
 - Global intercept and slope for Days
 - Subject-specific random intercepts and slopes
- Correlation between random effects: prior on subject effect is a multivariate normal with mean vector μ (mu) and covariance matrix Σ (Sigma).
- See Gelman & Hill 2007, chapter 11, 12, 13 or [this blogpost](#)
- [Multivariate reparameterization](#)

Specifying the Likelihood

- $\text{Reaction} \sim 1 + \text{Day} + (1 + \text{Subject} \mid \text{Day})$
- Create design matrix x with column 1 for group-level intercept term

```
transformed data {  
  matrix[N, 2] x;  
  x[ , 1] = rep_vector(1, N);  
  x[ , 2] = day;  
}
```

- Add group-level term to regression formula.

```
vector[N] eta = b_intercept + b_day * day + rows_dot_product(x, beta[ subj, ] );  
y ~ normal(eta, sigma);
```

Multi-variate reparameterization

Centered parameterization

```
data {  
  int K; // num predictors + intercept  
  int J; // num groups  
}  
  
parameters {  
  vector[K, J] beta; // group covariates  
  vector[K] mu; // location  
  cov_matrix[K] Sigma; // scale  
  // ...  
}  
  
model {  
  // hierarchical prior  
  for (j in 1:J) {  
    beta[, j] ~ multi_normal(mu, Sigma);  
  }  
  // ...  
}
```

Non-centered parameterization

```
data {  
  int K; int J;  
}  
  
parameters {  
  vector<lower=0>[K] tau;  
  cholesky_factor_corr[K] L_Omega;  
  matrix[K, J] beta_std;  
}  
  
transformed parameters {  
  matrix[J, K] beta =  
    (diag_pre_multiply(tau, L_Omega) * beta_std)';  
}  
  
model {  
  // non-centered priors  
  tau ~ exponential(1);  
  L_Omega ~ lkj_corr_cholesky(K);  
  to_vector(beta_std) ~ std_normal();  
}
```

Parameters, transformed parameters, model blocks

```
parameters {  
  real b_intercept; real b_day; real<lower=0> sigma;  
  
  vector<lower=0>[2] tau; cholesky_factor_corr[2] L_Omega;  
  matrix[2, J] beta_std;  
}  
transformed parameters {  
  // random effects matrix scaled, transposed (centered at 0)  
  matrix[J, 2] beta = (diag_pre_multiply(tau, L_Omega) * beta_std)';  
}  
model {  
  vector[N] eta = b_intercept + b_day * day + rows_dot_product(x, beta[ subj, ]);  
  y ~ normal(eta, sigma);  
  
  b_intercept ~ normal(250, 50); b_day ~ normal(10, 10); sigma ~ exponential(1);  
  
  tau ~ exponential(1); L_Omega ~ lkj_corr_cholesky(2);  
  to_vector(beta_std) ~ std_normal();  
}
```

Recovering the Quantities of Interest

```
generated quantities {  
  // match BRMS outputs  
  real sd_intercept = tau[1];  
  real sd_day = tau[2];  
  
  // Reconstruct correlation matrix  
  matrix[2, 2] Omega;  
  Omega = multiply_lower_tri_self_transpose(L_Omega);  
  real cor_intercept_day = Omega[1, 2];  
  
  // Posterior likelihood and posterior predictive y-replicates  
  vector[N] y_rep;  vector[N] log_lik;  
  { // don't save to output  
    vector[N] eta = b_intercept + b_day * day + rows_dot_product(x, beta[ subj, ] );  
    y_rep = to_vector(normal_rng(eta, sigma));  
    for (n in 1:N) {  
      log_lik[n] = normal_lpdf(y[n] | eta[n], sigma);  
    }  
  }  
}
```

Notebook

- github.com/mitzimirris/brms2stan/brms2stan.qmd
- Let's run this in RStudio

References

Stan User's Guide:

- Efficiency Tuning, [Hierarchical models and the non-centered parameterization](#)
- Efficiency Tuning, [Multivariate reparameterization](#)
- Regression, [Multivariate regression example](#)
- Regression, [Optimization through Cholesky Factorization](#)

Andrew Gelman's Blog

- Gelman blogpost [varying-slope/varying-intercept models](#)
- Bob Carpenter's response [Varying slopes and intercepts in Stan: still painful in 2024](#)

Many Thanks!

Questions???