Introduction to the Stan Programming Language

Ethan M. Alt

February 20-21, 2023

BIOS 779

Outline

Introduction to Stan

What is Stan?

- Stan is a probabilistic programming language written in C++.
- Stan is named in honour of Stanislaw Ulam, pioneer of the Monte Carlo method.
- Secretly, Stan also got its name because Andrew Gelman likes the Eminem song by the same name.
- Stan implements gradient-based Markov chain Monte Carlo (MCMC) algorithms called Hamiltonian Monte Carlo (HMC).
  - Gradients are computed using `autodiff` (i.e., numeric gradients). This function can be used outside of Stan for the frequentists out there.

HMC Basics

HMC Basics

- HMC corresponds to an instance of the Metropolis–Hastings algorithm, with a Hamiltonian dynamics evolution simulated using a time-reversible and volume-preserving numerical integrator (typically the leapfrog integrator) to propose a move to a new point in the state space (source: Wikipedia).
- Hamiltonian Monte Carlo reduces the correlation between successive sampled states by proposing moves to distant states which maintain a high probability of acceptance.
- The reduced correlation means fewer Markov chain samples are needed to approximate integrals with respect to the target probability distribution for a given Monte Carlo error.

HMC Basics

- The idea of HMC has been around for a while (Duane et al. 1986).
- The burden of having to supply gradients to each density ultimately prevented the popularization of HMC.
- Stan works by computing numerical gradients, allowing it to serve as a general-purpose MCMC software.

Why Stan?

Why Stan?

- Pros:
  - Sample from difficult (e.g., posterior) densities with little-to-no tuning.
  - Stan code is converted to C++ at the backend, ensuring speed.
  - Samples (usually) have very low autocorrelation.
- Cons:
  - Stan does not take advantage of conjugacy.
  - Stan does not use closed-form derivatives when they exist.
  - Stan is unable to sample discrete parameters (e.g., latent class), but one can sum out the discrete parameters.
  - Full Bayesian nonparametrics are not feasible (approximations are).

Why Stan?

- The Stan syntax is very strict, but relatively easy to master.
- HMC is highly efficient, but complicated to implement on your own.
- Stan interfaces with `R` (and `Python`, `Stata`).
- Stan allows users to sample from posterior densities without writing their own MCMC sampler.

Stan Syntax

- Stan has various frequently and infrequently used program `blocks`.
- The order of the blocks must be as follows:
  1. `functions {}` (e.g., user-defined densities)
  2. `data {}` (e.g., design matrix, hyperparameters for priors)
  3. `transformed data {}` (e.g., standardizing design matrix)
  4. `parameters {}` (e.g., population mean and variance)
  5. `transformed parameters {}` (transformations before sampling).
  6. model {} (computations of the log target density).
  7. `generated quantities {}` computations to be done after sampling (e.g., risk ratio, latent class prediction).

Stan: Bernoulli proportion example

- The below code implements a Bernoulli proportion with a $U(0, 1)$ prior.

```
data {
  int<lower=0> n;                        // number of observations
  array[n] int<lower=0,upper=1> y; // integer array of size n giving Bernoulli responses
}
parameters {
  real<lower=0,upper=1> p;    // Bernoulli proportion
}
model {
  // prior
  p ~ uniform(0, 1);    // not necessary since bounded parameters default to uniform prior
  // likelihood
  for ( i in 1:n )      // avoid looping whenever possible—see next slide
    y[i] ~ bernoulli(p)
}
```

Equivalent (and faster) code

- Declared parameters default to uniform priors (which may be proper or improper depending on bounds).
    - Thus, we do not need to explicitly write `p ~ uniform(0, 1)`, which wastes time.
- Some distributions allow for vectorization. The same code can be written more efficiently as follows:

```
data {
  int<lower=0> n;                        // number of observations
  array[n] int<lower=0,upper=1> y; // integer array of size n giving Bernoulli responses
}
parameters {
  real<lower=0,upper=1> p;    // Bernoulli proportion
}
model {
  // likelihood
  y ~ bernoulli(p);
}
```

Generalizing the code

- It is a good idea to write code as generally as possible.
- E.g., suppose now we wish to elicit a Beta(α, β) prior. We would have to create a whole new program.
- We can write the code more generally as follows:

```
data {
  int<lower=0> n;                        // number of observations
  array[n] int<lower=0,upper=1> y;       // integer array of size n giving Bernoulli responses
  real<lower=0> p_shape1;                // first shape parameter for beta prior on proportion
  real<lower=0> p_shape2;                // second shape parameter for beta prior on proportion
}
parameters {
  real<lower=0, upper=1> p;    // Bernoulli proportion
}
model {
  // prior
  p ~ beta(p_shape1, p_shape2);

  // likelihood
  y ~ bernoulli(p);
}
```

More improvements

- Recall that a likelihood is defined up to a constant of proportionality. Let $s = \sum_{i=1}^n y_i$ denote the sum of the data. Then

$$L(p|\boldsymbol{y}) \propto \prod_{i=1}^n p^{y_i}(1-p)^{1-y_i} = p^s(1-p)^{n-s} \propto \text{Bin}(s|n,p).$$

```
data {
  int<lower=0> n;                    // number of observations
  array[n] int<lower=0,upper=1> y; // integer array of size n giving Bernoulli responses
  real<lower=0> p_shape1;            // first shape parameter for beta prior on proportion
  real<lower=0> p_shape2;            // second shape parameter for beta prior on proportion
}
transformed data {
  int s = sum(y);     // sum of all responses
  int is_not_uniform = (p_shape1 != 1 || p_shape2 != 1) ? 1 : 0;
}
parameters {
  real<lower=0, upper=1> p;    // Bernoulli proportion
}
model {
  // prior
  if (is_not_uniform)
    p ~ beta(p_shape1, p_shape2);

  // likelihood
  s ~ binomial(n, p);
}
```

User-specified densities

- Recall that the conjugate prior for exponential families is given by

$$\pi_{\mathrm{DY}}(\theta|n_0,\mu_0) \propto \exp\left\{n_0\left[\mu_0\theta - b(\theta)\right]\right\} = \exp\left\{n_0\left[\mu_0\theta - \log\left(1+e^{\theta}\right)\right]\right\}$$

```
functions {
  real conjprior_lpdf(real theta, real n0, real mu0) {
    return n0 * (mu0 * theta - log1p_exp(theta));
   }
}
data {
  int<lower=0> n;                       // number of observations
  array[n] int<lower=0,upper=1> y; // integer array of size n giving Bernoulli responses
  real<lower=0> n0;                     // prior sample size for conjugate prior
  real<lower=0,upper=1> mu0;            // prior prediction for E(y)
}
parameters {
  real theta;    // canonical parameter [logit(p)]
}
model {
  // prior
  theta ~ conjprior(n0, mu0);
  // likelihood
  y ~ bernoulli_logit(theta);
}
generated quantities {
  real p = inv_logit(theta);
}
```

An example using all blocks

```
functions {
  real conjprior_lpdf(real theta, real n0, real mu0) {
    return n0 * (mu0 * theta - log1p_exp(theta));
   }
}
data {
  int<lower=0> n;                       // number of observations
  array[n] int<lower=0,upper=1> y; // integer array of size n giving Bernoulli responses
  real<lower=0> n0;                     // prior sample size for conjugate prior
  real<lower=0,upper=1> mu0;            // prior prediction for E(y)
}
transformed data {
  int s = sum(y);
}
parameters {
  real theta;   // canonical parameter [logit(p)]
}
transformed parameters {
  real p = inv_logit(theta);      // computed at every iteration DURING the sampling scheme
}
model {
  // prior
  theta ~ conjprior(n0, mu0);
  // likelihood
  s ~ binomial(n, p);
}
transformed parameters {
  real odds = p * inv(1 - p);   // computed AFTER all sampling has been done
}
```

A warning on putting priors on transformed parameters

- Suppose I want a normal prior on the log odds $\theta = \log\left[\frac{p}{1-p}\right]$.
- It can be tempting to write

```
functions {
   ...
}
data {
...
}
parameters {
  real p;    // success probability
}                                             ,
transformed parameters {
  real theta = log(p) - log1m(p); // log odds
}
model {
   // prior
   theta ~ std_normal();
   // likelihood
   y ~ bernoulli(p);
}
```

- Does this give us the desired posterior?

A warning on putting priors on transformed parameters

- We began with $p \sim U(0,1)$ in the parameters block.
- The Jacobian of $p = \frac{e^\theta}{1+e^\theta}$ is given by

$$\frac{\partial p}{\partial \theta} = \frac{e^\theta}{(1+e^\theta)^2}.$$

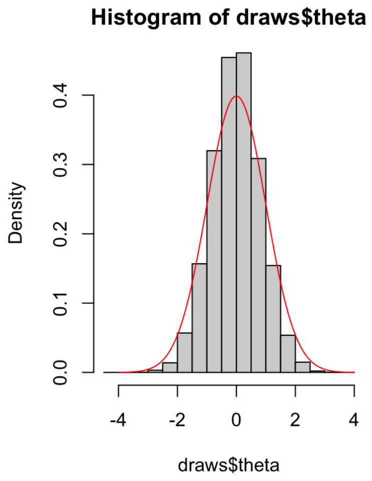
- Thus, the prior we (accidentally) elicited is given by

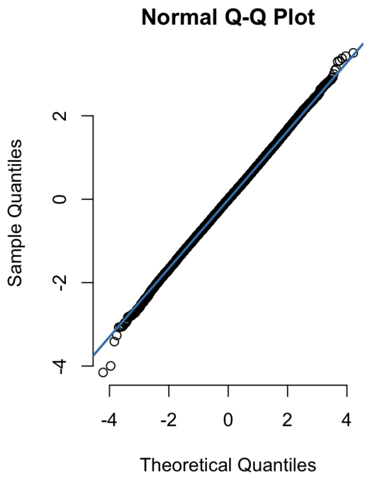
$$\pi(\theta) = \frac{\frac{e^\theta}{(1+e^\theta)^2}\phi(\theta|0,1^2)}{C},$$

  with normalizing constant $C \approx 0.2066$.

- Stan may or may not provide a warning when a Jacobian adjustment is necessary. In this case, it did not.

A warning on putting priors on transformed parameters





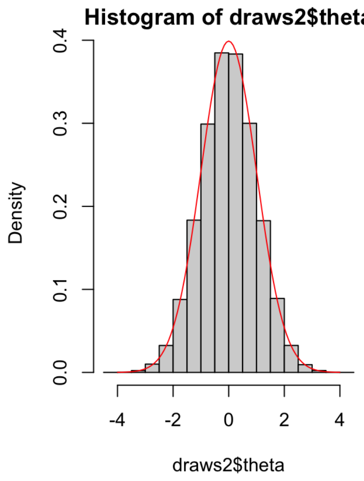
First panel shows histogram and $N(0, 1)$ density overlaid. Second is a QQ plot comparing quantiles of the prior with those from a $N(0, 1)$ density.

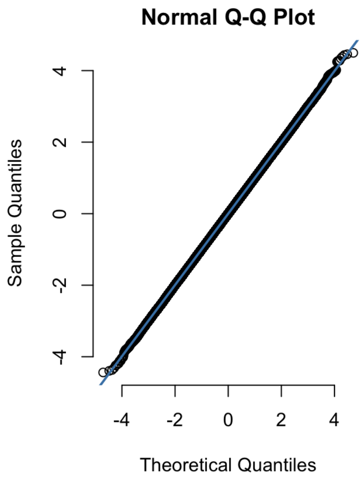
A warning on putting priors on transformed parameters

- If we make the appropriate Jacobian adjustment, we are fine.

```
functions {
   ...
}
data {
...
}
parameters {
  real p;    // success probability
}
transformed parameters {
  real theta = log(p) - log1m(p); // log odds
}
model {
   // prior
   theta ~ std_normal();
   target += -(theta - 2 * log1p_exp(theta));    // jacobian
   // likelihood
   y ~ bernoulli(p);
}
```

A warning on putting priors on transformed parameters





First panel shows histogram and $N(0, 1)$ density overlaid. Second is a QQ plot comparing quantiles of the prior with those from a $N(0, 1)$ density.

A warning on putting priors on transformed parameters

- In general, best practice is to put priors on parameters declared in the model block whenever possible (99% of the time).
- We could have avoided this headache by reversing what is declared in `parameters` and `transformed parameters`

```
functions {
    ...
}
data {
...
}
parameters {
  real theta;   // success probability
}
transformed parameters {
  real p = inv_logit(theta); // log odds
}
model {
   // prior
   theta ~ std_normal();
   // likelihood
   y ~ bernoulli(p);
}
```

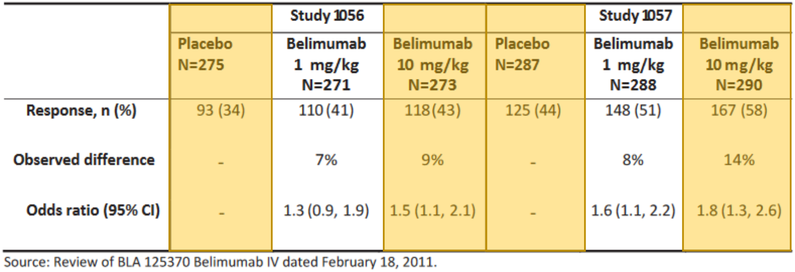
Informative Prior Elicitation in Stan

Historical data

- In clinical trials, it is common to possess prior information (e.g., expert opinion / historical data).
- There are various examples:
  1. Phase II study data could be used in a Phase III study.
  2. Adult study data could be used in a pediatric study.
- Why use prior information?
  1. Ideally, you wouldn't have to.
  2. Rare diseases
  3. Pediatric studies.
  4. Impractical / too expensive.
- In this section, we will implement various informative priors in Stan.

The BLISS Trials

- The BLISS-52 and BLISS-76 trials were two RCTs in adult systemic lupus erythematosus (SLE).
- The FDA approved belimumab for the treatment of adults with active, seropositive SLE (who are already on SOC).



		Study 1056			Study 1057	
	Placebo N=275	Belimumab 1 mg/kg N=271	Belimumab 10 mg/kg N=273	Placebo N=287	Belimumab 1 mg/kg N=288	Belimumab 10 mg/kg N=290
Response, n (%)	93 (34)	110 (41)	118(43)	125 (44)	148 (51)	167 (58)
Observed difference	-	7%	9%	-	8%	14%
Odds ratio (95% CI)	-	1.3 (0.9, 1.9)	1.5 (1.1, 2.1)	-	1.6 (1.1, 2.2)	1.8 (1.3, 2.6)

Source: Review of BLA 125370 Belimumab IV dated February 18, 2011.

BLISS Trials Primary Endpoint Data: SRI Response Rate

The PLUTO Trial

- PLUTO was a Phase II RCT evaluating the efficacy, safety, and pharmacokinetics of IV belimumab vs. placebo plus SOC in childhood-onset SLE in patients aged 5–17 years.
- At 52 weeks, clinical response was observed in $y_{11} = 28$ of $n_{11} = 53$ treated patients and $y_{10} = 17$ out of $n_{10} = 39$ placebo patients.
- $\bar{y}_{11} - \bar{y}_{10} = 0.092$.
- These sample sizes are quite small, making it unlikely to be able to detect a significant effect.
- As the disease progression between adults and children are similar, the clinical review team decided it would be appropriate to borrow information from the adult trials.

Power Prior

Power Prior

- Let $\delta(\boldsymbol{\theta}) = \theta_1 - \theta_0$ be the difference in response probabilities for participants treated with 10 mg/kg belimumab (θ_1) compared to placebo (θ_0).

- We formulate a power prior based on θ_1 and θ_0 noting that we may compute the posterior for any function of them using MCMC, namely $\delta(\boldsymbol{\theta}) = \theta_1 - \theta_0$.

Power Prior

- Specifically, we have

$$\pi(\boldsymbol{\theta}|D_0, \boldsymbol{a}_0) \propto \mathcal{L}(\boldsymbol{\theta}|D_{01})^{a_{01}} \mathcal{L}(\boldsymbol{\theta}|D_{02})^{a_{02}} \pi_0(\boldsymbol{\theta}), \tag{1}$$

where

- $D_{01} = \{(y_{01}, n_{01}) = (\ 93, 275), (y_{11}, n_{11}) = (118, 273)\}$,
- $D_{02} = \{(y_{02}, n_{02}) = (125, 287), (y_{12}, n_{12}) = (167, 290)\}$,
- The likelihood for the s^{th} data set may be written as

$$\mathcal{L}(\boldsymbol{\theta}|D_{0s}) = \theta_0^{y_{0s}}(1-\theta_0)^{n_{0s}-y_{0s}}\theta_1^{y_{1s}}(1-\theta_1)^{n_{1s}-y_{1s}},$$

- $\pi_0(\boldsymbol{\theta}) = \pi_0(\theta_0)\pi_0(\theta_1)$ with $\pi_0(\theta_j) = \text{Beta}(\theta_j|\alpha_0, \beta_0)$, and
- $a_{0s} \in [0, 1]$.

Stan implementation of the power prior for the PLUTO study

- `R` and `Stan` code (in `rmarkdown`) to implement the power prior for this example is available at https://github.com/ethan-alt/IntroBayesianAnalysis/tree/main/Examples

- Click here for an HTML preview.

Normalized power prior

The Normalized Power Prior

- Alternatively, one may treat a_0 as a random variable and assign it a prior distribution. With appropriate normalization, this results in a normalized power prior (Duan et al, 2006 [1]).

- The normalized power prior is given by

$$\begin{aligned}\pi(\boldsymbol{\theta}, a_0 | D_0) &= \frac{1}{c(a_0, D_0)} \mathcal{L}(\boldsymbol{\theta}|D_0)^{a_0} \; \pi_0(\boldsymbol{\theta}) \pi_0(a_0), \\ &= \pi(\boldsymbol{\theta}|D_0, a_0)\pi_0(a_0)\end{aligned}$$

  where

  - $c(a_0, D_0) = \int \mathcal{L}(\boldsymbol{\theta}|D_0)^{a_0} \; \pi_0(\boldsymbol{\theta}) d\boldsymbol{\theta}$, and
  - $\pi_0(a_0)$ is an initial prior for a_0 often taken to be a beta distribution.

Normalized power prior: PLUTO trial

- The power prior with an initial beta prior is given by

$$\pi_{\text{PP}}(\boldsymbol{\theta}|\boldsymbol{D}_0, \boldsymbol{a}_0) \propto \prod_{j=0}^{1} \left\{ \theta_j^{\alpha_{0j}-1}(1-\theta_j)^{\beta_{0j}-1} \prod_{s=1}^{2} \theta_j^{a_{0s}y_{0sj}}(1-\theta_j)^{a_{0s}(n_{0sj}-y_{0sj})} \right\}$$

$$= \prod_{j=0}^{1} \theta_j^{\alpha_{0j}^*-1}(1-\theta_j)^{\beta_{0j}^*-1},$$

  where

  - $\alpha_{0j}^* = \sum_{s=1}^{2} a_{0s}y_{0sj} + \alpha_{0j}$,
  - $\beta_{0j}^* = \sum_{s=1}^{2} a_{0s}(n_{0sj} - y_{0sj}) + \beta_{0j}$.

- The kernel of the power prior is the kernel of a Beta$(\alpha_{0j}^*, \beta_{0j}^*)$ density, so the normalizing constant is $c(a_0, D_0) = B(\alpha_{0j}^*, \beta_{0j}^*)$.
- We elicit a Beta(γ_{0j}, η_{0j}) prior on a_0.

Normalized power prior: PLUTO trial

- R and Stan code (in rmarkdown) to implement the normalized power prior for this example is available at https://github.com/ethan-alt/IntroBayesianAnalysis/blob/main/Examples/PLUTO_normalizedPowe

- Click here for an HTML preview.

Bayesian Hierarchical Model (BHM)

Bayesian Hierarchical Model (BHM)

- For the BHM, we consider a logistic regression model parameterization given by

  $$\text{logit}\left[P(Y_{hi} = 1 | z_{hi})\right] = \beta_{0h} + \beta_{1h} z_{hi},$$

  for participant $i = 1, ..., n_h$ from study $h = 1, 2, 3$.

- Here z_{hi} is an indicator for whether participant i was randomized to receive belimumab.

- We consider $h = 1$ to correspond to the PLUTO (pediatric) data and $h = 2$ and $h = 3$ to correspond to the BLISS-52 and BLISS-76 (adult) data, respectively.

Bayesian Hierarchical Model

- The hierarchical prior for the intercepts is given as follows.

$$\begin{aligned} \beta_{0h}|\beta_0, \tau_0 &\sim \text{Normal}\,(\beta_0, \text{sd} = \tau_0) \\ \beta_0 &\sim \text{Normal}\,(0, \text{sd} = 3) \\ \tau_0 &\sim \text{Half-Cauchy}\,(0, 1) \end{aligned}$$

- The hierarchical prior for the treatment effects is given as follows.

$$\begin{aligned} \beta_{1h}|\beta_1, \tau_1 &\sim \text{Normal}\,(\beta_1, \text{sd} = \tau_1) \\ \beta_1 &\sim \text{Normal}\,(0, \text{sd} = 3) \\ \tau_1 &\sim \text{Half-Cauchy}\,(0, 1) \end{aligned}$$

- The non-conjugate hyper-priors for τ_0 and τ_1 are proper, but noninformative ($\text{Var}(\tau_j) = \infty$).
- The Half-Cauchy prior was suggested by Gelman [2].

Bayesian Hierarchical Model

- The hierarchical parameters τ_0 and τ_1 quantify the degree to which the three data sets agree (between-study heterogeneity).
- For example, if $\Pr(\tau_1 < 1/3|\boldsymbol{y})$ is large, then the treatment effects are likely to be within 1 standard deviation of each other.
- Special choices of a_0 in the power prior and particular priors in the BHM yield equivalent results [3].
- In general, it is difficult to ascertain the level of borrowing of the prior.
- The choice of priors must be justified to regulators, which can sometimes be difficult.

Bayesian Hierarchical Model: PLUTO Example

- R and Stan code for the implementation of the BHM for the PLUTO study is available at
  PLUTO_bhm.Rmd

- Rendered HTML version.

Robust Mixture Priors

Robust Mixture Priors

- Robust mixture priors combine one or more informative priors with a vague prior via a mixture distribution:

$$\pi(\theta) = \sum_{i=1}^{I} \gamma_{Ii} f_{Ii}(\theta|\eta_i) + \gamma_v f_v(\theta|\eta_v), \quad \sum_{i=1}^{I} \gamma_{Ii} + \gamma_v = 1.$$

- Note that each f_{Ii} and f_v must be a properly normalized density.
- In the context of historical data, the f_{Ii}'s may be chosen as a normal approximation to the posterior of θ for each historical data set.
- The vague prior could be, e.g., $\theta \sim N(0, 10^2)$ for $\theta \in (-\infty, \infty)$ or a uniform prior if θ is bounded.

Robust Mixture Priors: PLUTO Example

- To obtain a robust mixture prior, we conduct separate analysis of the BLISS-52 and BLISS-76 trials via maximum likelihood:

Data set	Intercept	Treatment Effect
BLISS-52	$-0.6714(0.1275)$	$0.3987\ (0.1766)$
BLISS-76	$-0.2593(0.1190)$	$0.5651(0.1682)$

MLE of historical data sets (SE in parentheses).

- We elicit a multivariate normal priors with mean = MLE, covariance = inverse Fisher information.
  - BLISS-52: $f_{I1}(\boldsymbol{\beta}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, $\boldsymbol{\Sigma}_1 = \begin{pmatrix} 0.0162 & -0.0162 \\ -0.0162 & 0.0312 \end{pmatrix}$
  - BLISS-76: $f_{I2}(\boldsymbol{\beta}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, $\boldsymbol{\Sigma}_2 = \begin{pmatrix} 0.0142 & -0.0142 \\ -0.0142 & 0.0283 \end{pmatrix}$

Robust Mixture Priors: PLUTO Example

- For robustification, we mix these priors with a $N_2(\mathbf{0}, 100^2 \boldsymbol{I}_2)$ vague prior.
- The implemented prior is thus:

$$\pi_{\text{RM}}(\beta|D_0) = \gamma_{I1} N_2(\beta|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \gamma_{I2} N_2(\beta|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) + \pi_v N_2(\beta|\mathbf{0}, 10^2 \boldsymbol{I})$$

- It is unstable to implement mixtures in this way.
- Let $M = \max_{1 \le k \le K}\{\log \gamma_k + \log f_k\}$. Note that

$$\log\left(\sum_{k=1}^{K} \gamma_k f_k\right) = \log\left(\sum_{k=1}^{K} e^{\log \gamma_k + \log f_k}\right) = M + \log\left(\sum_{k=1}^{K} e^{\log \gamma_k + \log f_k - M}\right),$$

- This is much more stable, and is implemented in Stan as `log_sum_exp(vector x)`

Robust Mixture Priors: PLUTO Example

- We choose $\gamma_{I1} = 0.3$, $\gamma_{I2} = 0.3$, $\gamma_v = 0.4$.
- The code is available in `/Examples/PLUTO_robustMixture.qmd`
- HTML rendering .

Commensurate Priors

Commensurate Priors

- The commensurate prior (CP) (Hobbs et al, 2012) [4] is developed for single historical data set settings.
- The CP assumes

$$\pi_{\mathrm{CP}}(\boldsymbol{\beta}, \boldsymbol{\beta}_0 | D_0) \propto \left[\prod_{j=1}^{J} N_p(\beta_j | \beta_{0j}, \tau_j^{-1}) \right] \mathcal{L}(\boldsymbol{\beta}_0 | D_0) \pi_0(\boldsymbol{\beta}_0)$$

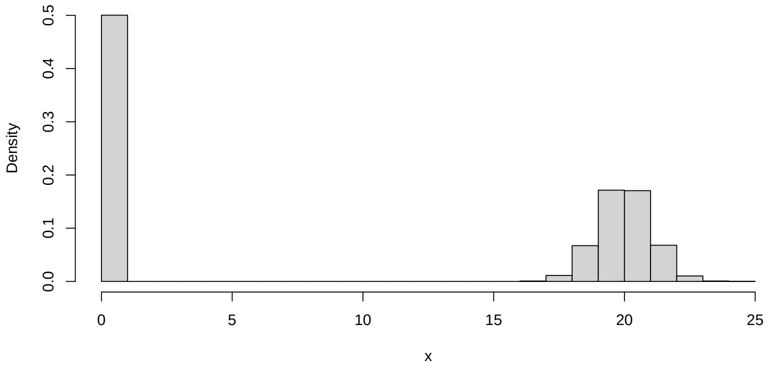
- Similar to the BHM, the CP does not assume that the regression coefficients are the same.
- The precision parameters τ_j's measure the commensurability of the current and historical data sets for each parameter.
- The authors recommend independent spike and slab priors on τ_j's, e.g.,

$$\pi(\tau_j) = \gamma N(\tau_j | 20, 1) + (1 - \gamma) U(\tau_j | 0.10, 0.50)$$

- γ may be elicited or given a hyperprior.

Commensurate Priors

- We plot the marginal prior for τ_j when $\gamma \sim U(0,1)$.



Marginal prior for τ_j when $\tau_j|\gamma$ is spike and slab and $\gamma \sim U(0,1)$.

Commensurate Priors: PLUTO Example

- We specify $\gamma \sim U(0, 1)$.
- The implementation is in `Examples/PLUTO_commensurate.Rmd`.
- HTML rendering.

Robust MAP Prior

- The Robust MAP prior is essentially a robust mixture prior of the form

$$\pi_{\mathrm{rMAP}}(\theta|D_0) = (1-\gamma)\pi_{\mathrm{MAP}}(\theta|D_0) + \gamma\pi_{\mathrm{v}}(\theta),$$

where π_v is a "vague" (i.e., noninformative) prior and $\gamma \in (0,1)$. The prior π_{MAP} is the meta-analytic predictive prior, which is the prior for the current data parameters induced by the hierarchical model

$$\pi_{\mathrm{MAP}}(\theta) = \int\int \phi(\theta|\mu,\tau^{-1})\pi(\mu,\tau|D_0)d\mu d\tau,$$

where

$$\pi(\mu,\tau|D_0) \propto \int \left[\prod_{j=1}^{J} L(\theta_{0j}|D_{0j})\pi(\theta_{0j}|\mu,\tau^{-1})\right]\pi(\mu,\tau)d\boldsymbol{\theta}_0$$

Robust MAP Prior

- In most cases, π_{MAP} is not available in closed form.
- We may construct a finite mixture model approximation via the following algorithm:
  1. Sample from the prior
  $$\pi(\mu, \tau^{-1}|D_0) \propto \pi(\mu, \tau^{-1}) \prod_{j=1}^{J} L(\theta_{0j}|D_{0j})\pi(\theta_{0j}|\mu, \tau^{-1})$$
  2. Sample $\theta|D_0, \mu, \tau \sim N(\mu, \tau)$
  3. Fit a finite mixture model to the samples in (2) to obtain
  $$\hat{\pi}_{\text{MAP}}(\theta) = \sum_{k=1}^{K} \alpha_k \phi(\theta|\mu_k, \sigma_k^2), \quad \sum_{k=1}^{K} \alpha_k = 1$$

Robust MAP Prior

- If we have a vector of parameters $\boldsymbol{\theta}$, we may sample from a mixture of multivariate normals

$$\hat{\pi}_{\mathsf{MAP}}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k \phi_p(\boldsymbol{\theta}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

- Obviously, this makes the robust MAP tedious to implement.
- The authors suggest to minimize the K-L divergence between the MAP prior and the approximation.
- We instead minimize the BIC associated with the mixture approximation, which is much easier to implement.

PLUTO: Robust MAP Prior

- The R and Stan code is available at /Examples/PLUTO_rmap.rmd

- Rendered HTML version

Efficient Parameterizations

Efficient Parameterizations

- For some models (e.g., GLMMs with many random effects), the convergence can be quite slow.

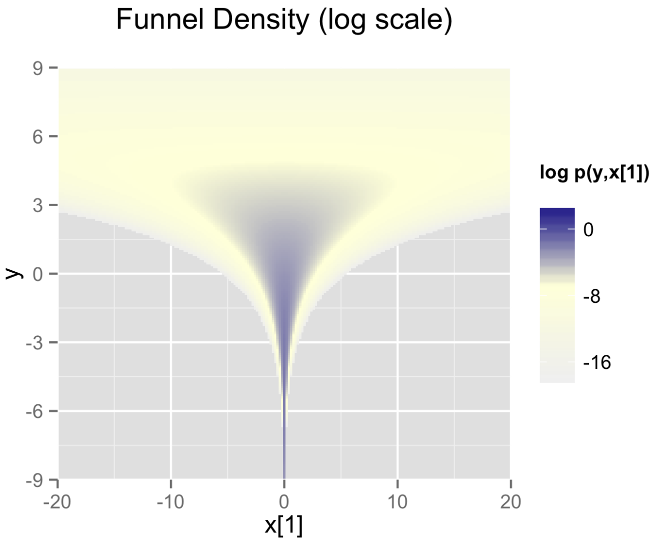
- The developers at Stan recommend using non-centered parameterizations (NCP) for complicated posterior geometries.

- Example: Neal's funnel:

$$f(y, \boldsymbol{x}) = \phi(y|0, 3) \times \prod_{i=1}^{9} \phi(x_i|0, e^{y/2}).$$

Funnel

Efficient Parameterizations: Funnel



Efficient Parameterizations: Funnel

- It is trivial to implement this density in Stan.

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

Efficient Parameterizations: Funnel

- When the model is expressed this way, Stan has trouble sampling from the neck of the funnel, where y is negative and thus x is constrained to be near 0.

- This is due to the fact that the density's scale changes with y, so that a step size that works well in the body will be too large for the neck, and a step size that works in the neck will be inefficient in the body.

- A similar behavior is observed with hierarchical models with random variance components, e.g., a GLMM:

$$E(y|b, \boldsymbol{\beta}, \boldsymbol{x}) = \beta_0 + b + \boldsymbol{x}'\boldsymbol{\beta}, \quad b \sim N(0, \sigma_b^2),$$

  with some prior on σ_b^2.

Efficient Parameterizations: Funnel

- The model can be converted to the following more efficient form.

```
parameters {
  real y_raw;
  vector[9] x_raw;
}
transformed parameters {
  real y;
  vector[9] x;

  y = 3.0 * y_raw;
  x = exp(y/2) * x_raw;
}
model {
  y_raw ~ std_normal(); // implies y ~ normal(0, 3)
  x_raw ~ std_normal(); // implies x ~ normal(0, exp(y/2))
}
```

- In this second model, the parameters `x_raw` and `y_raw` are sampled as independent standard normals, which is easy for Stan. These are then transformed into samples from the funnel.

Cauchy

Efficient Parameterizations: Cauchy

- Stan has difficulty sampling from heavy-tailed distributions, such as the t family of distributions with small df (e.g., Cauchy).
- The practical problem is that tail of the Cauchy requires a relatively large step size compared to the trunk.
    - With a small step size, the No-U-Turn sampler requires many steps when starting in the tail of the distribution
    - With a large step size, there will be too much rejection in the central portion of the distribution.
- This problem may be mitigated by defining the Cauchy-distributed variable as the transform of a uniformly distributed variable using the Cauchy inverse cumulative distribution function.

Efficient Parameterizations: Cauchy

- The density function, CDF, and quantile function for the Cauchy are given by

$$f(x|\mu,\sigma) = \frac{1}{\pi\sigma\left[1+\left(\frac{x-\mu}{\sigma}\right)^2\right]},$$
$$F(x|\mu,\sigma) = \frac{1}{\pi}\text{arctan}\left(\frac{x-\mu}{\sigma}\right)+\frac{1}{2},$$
$$F^{-1}(x|\mu,\sigma) = \mu+\sigma\tan\left(\pi\left[x-\frac{1}{2}\right]\right),$$

- Recall the (very useful) fact that if $U \sim U(0,1)$ and F is a continuous CDF with inverse F^{-1}, then $Y = F^{-1}(U) \sim F$
- Hence, we can sample from the Cauchy(μ,σ) distribution via

$$U \sim U(0,1), \quad Y = \mu+\sigma\tan\left(\pi\left[U-\frac{1}{2}\right]\right)$$

Efficient Parameterizations: Cauchy

- Note that $U^* = \pi\left(U - \frac{1}{2}\right) \sim U\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
- Hence, the previous scheme reduces to

$$U^* \sim U\left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \quad Y = \mu + \sigma \tan\left(U^*\right)$$

- A half-Cauchy random variable can be similarly defined (**exercise**).
- Note that the half-Cauchy family is very useful for prior elicitation for variance parameters as it is proper but noninformative.

Efficient Parameterizations: Cauchy

- Consider a Stan program involving a Cauchy-distributed parameter `beta`.

```
parameters {
  real beta;
  // ...
}
model {
  beta ~ cauchy(mu, tau);
  // ...
```

- This declaration of beta as a parameter may be replaced with a transformed parameter `beta` defined in terms of a uniform-distributed parameter `beta_unif`.

```
parameters {
  // beta_unif ~ U(-pi/2, pi/2)
  real<lower=-pi() / 2, upper=pi() / 2> beta_unif;
  // ...
}
transformed parameters {
  real beta= mu + tau * tan(beta_unif);  // beta ~ cauchy(mu, tau)
}
```

Student t

Efficient Parameterizations: Student t

- Student t distributions with heavy tails (e.g., Cauchy) can be difficult to sample via MCMC.

- Recall that if $X \sim t(\nu, 0, 1)$, the density of X can be expressed as a scale mixture in the normal-gamma family, i.e.,

$$f_X(x|\nu) = \int \phi(x|0, \tau^{-1}) \times f_\Gamma\left(\frac{\nu}{2}, \frac{\nu}{2}\right) d\tau$$

- Said differently, if $X|\tau \sim N(0, \tau^{-1})$, and $\tau \sim \text{Gamma}(\nu/2, \nu/2)$, then the marginal distribution of X is $t(\nu, 0, 1)$.

Efficient Parameterizations: Student *t*

- We can go one step further. Suppose $\alpha \sim N(0, 1)$ and let $\beta = \alpha\tau^{-1/2}$. Then $\beta|\alpha, \tau \sim N(0, \tau^{-1})$.
- Suppose $E(y|x) = \beta x$.
- If we parameterize $\beta \sim t(\nu, 0, 1)$, we have a heavy-tailed prior on β.
- Conversely, if we sample $\tau \sim \Gamma(\nu/2, \nu/2)$, $\alpha \sim N(0, 1)$, and set $\beta = \alpha\tau$, we put a marginal *t* prior on beta, but we sample from distributions with light tails.
- These approaches are often called data augmentation approaches (or variable augmentation).

Efficient Parameterizations: Student *t*

- To summarize, we replace

```
parameters {
  real<lower=0> nu;
  real beta;
  // ...
}
model {
  beta ~ student_t(nu, 0, 1);
  // ...
}
```

with

```
parameters {
  real<lower=0> nu;
  real<lower=0> tau;
  real alpha;
  // ...
}
transformed parameters {
  real beta;
  beta = alpha / sqrt(tau);
  // ...
}
model {
  real half_nu = 0.5 * nu;
  tau ~ gamma(half_nu, half_nu);
  alpha ~ std_normal();
  // ...
}
```

Hierarchical Models and Non-Centered Parameterizations

Hierarchical Models and Non-Centered Parameterizations

- Applied Bayesian modeling involves complex geometries and interactions that are not known analytically.
- Nevertheless, reparameterization can still be effective for separating parameters.
- We discuss the differences between centered and non-centered parameterizations.

Centered Parameterizations

- Consider a hierarchical linear model as follows:

$$\begin{aligned} y|\boldsymbol{\beta}, \sigma^2, x &\sim N(\boldsymbol{x}'\boldsymbol{\beta}, \sigma^2), \\ \boldsymbol{\beta}|\mu_\beta, \sigma_\beta &\sim N(\mu_\beta \boldsymbol{J_p}, \sigma_\beta^2 \boldsymbol{I_p}), \end{aligned}$$

  with hyperprior $\pi(\mu_\beta, \sigma_\beta)$.

- The values of β, μ_β, and σ_β will be highly correlated in the posterior.
- The extremity of the correlation depends on the amount of data, with Neal's funnel being the extreme with no data.
- In these cases, the non-centered parameterization, discussed in the sequel, is preferable; when there is a lot of data, the centered parameterization is more efficient.

Non-Centered Parameterizations

- Sometimes, the group-level effects do not constrain the hierarchical distribution tightly (e.g., when there are not many groups or when the inter-group variation is high).

- In such cases, hierarchical models can be made much more efficient by shifting the data's correlation with the parameters to the hyperparameters.

- In extreme cases, it can become necessary to use a non-centered parameterization to achieve convergence.

Non-Centered Parameterizations

```
parameters {
  vector[p] beta_raw;
  real mu_beta;
  real<lower=0> sigma_beta;
}
transformed parameters {
  vector[p] beta;
  beta = mu_beta + sigma_beta * beta_raw;  // implies: beta ~ normal(mu_beta, sigma_beta)
}
model {
  beta_raw ~ std_normal();
  // ...
}
```

Non-Centered Parameterizations

- Reparameterization of hierarchical models is not limited to the normal distribution, although the normal distribution is the best candidate for doing so.
- In general, any distribution of parameters in the location-scale family is a good candidate for reparameterization (e.g., normal, t-distributions, uniform, logistic, Laplace, extreme value).
- If $X \sim D(l, s)$ and we can write $X = l + sY$ where $Y \sim D(0, 1)$, then the distribution D is in the location-scale family.

Multivariate Reparameterizations

Multivariate Reparameterizations

- The benefits of reparameterization are not limited to univariate distributions.
- A parameter with a multivariate normal prior distribution is also an excellent candidate for reparameterization.
- Recall that if $\boldsymbol{y} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then we can write

$$\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{LZ}, \quad \boldsymbol{z} \sim N(\boldsymbol{0}, \boldsymbol{I}),$$

where $\boldsymbol{\Sigma} = \boldsymbol{LL}'$ is the Cholesky decomposition of $\boldsymbol{\Sigma}$.

Multivariate Reparameterizations

- Suppose we wish to elicit the prior $\boldsymbol{\beta} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The centered parameterization is given as

```
data {
  int<lower=2> K;
  vector[K] mu;
  cov_matrix[K] Sigma;
  // ...
}
parameters {
  vector[K] beta;
  // ...
}
model {
  beta ~ multi_normal(mu, Sigma);
  // ...
}
```

Multivariate Reparameterizations

- Suppose we wish to elicit the prior $\boldsymbol{\beta} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The non-centered parameterization is given as

```
data {
  int<lower=2> K;
  vector[K] mu;
  cov_matrix[K] Sigma;
  // ...
}
transformed data {
  matrix[K, K] L = cholesky_decompose(Sigma);
}
parameters {
  vector[K] alpha;
  // ...
}
transformed parameters {
  vector[K] beta = mu + L * alpha;
}
model {
  alpha ~ std_normal();   // implies: beta ~ multi_normal(mu, Sigma)
  // ...
}
```

Multivariate Reparameterizations

- The non-centered reparameterization is more efficient for two reasons.

  1. It reduces dependence among the elements of α.
  2. It avoids the need to invert $\mathbf{\Sigma}$ every time `multi_normal` is evaluated.

- The non-centered approach can be extended to other multivariate distributions that can be conceptualized as contaminations of the multivariate normal, e.g., the multivariate t and the skew multivariate normal distribution.

Multivariate Reparameterizations: Wishart

- If $\boldsymbol{W} \sim \text{Wishart}(\nu, \boldsymbol{S})$, we can write $\boldsymbol{W} = \boldsymbol{LAL'A'} = (\boldsymbol{LA})(\boldsymbol{LA})'$, where $\boldsymbol{S} = \boldsymbol{LL'}$ and

$$\boldsymbol{A} = \begin{pmatrix} \sqrt{c}_1 & 0 & \cdots & 0 \\ z_{21} & \sqrt{c}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ z_{K1} & z_{K2} & \cdots & \sqrt{c}_K \end{pmatrix},$$

  where $c_k \sim \chi^2(\nu - k + 1)$ and $z_{km} \sim N(0, 1)$.

Multivariate Reparameterizations: Wishart

```
data {
  int<lower=1> N;
  int<lower=1> K;
  int<lower=K + 2> nu
  matrix[K, K] L; // Cholesky factor of scale matrix
  vector[K] mu;
  matrix[N, K] y;
  // ...
}
parameters {
  vector<lower=0>[K] c;
  vector[0.5 * K * (K - 1)] z;
  // ...
}
model {
  matrix[K, K] A;
  int count = 1;
  for (j in 1:(K - 1)) {
    for (i in (j + 1):K) {
      A[i, j] = z[count];
      count += 1;
    }
    for (i in 1:(j - 1)) {
      A[i, j] = 0.0;
    }
    A[j, j] = sqrt(c[j]);
  }
  for (i in 1:(K - 1)) {
    A[i, K] = 0;
  }
  A[K, K] = sqrt(c[K]);

  for (i in 1:K) {
    c[i] ~ chi_square(nu - i + 1);
  }

  z ~ std_normal();
  // implies: L * A * A' * L' ~ wishart(nu, L * L')
  y ~ multi_normal_cholesky(mu, L * A);
  // ...
}
```

Multivariate Reparameterizations

- This reparameterization is more efficient for three reasons.

    1. It reduces dependence among the elements of $\boldsymbol{z}$.
    2. It avoids the need to invert the covariance matrix $\boldsymbol{W}$ every time `wishart` is evaluated.
    3. If $\boldsymbol{W}$ is to be used with a multivariate normal distribution, you can pass $\boldsymbol{LA}$ to the more efficient `multi_normal_cholesky` function, rather than passing $\boldsymbol{W}$ to `multi_normal`.

- If $\boldsymbol{W} \sim \text{Wishart}(\nu, \boldsymbol{S})$, then $\boldsymbol{W}^{-1} \sim \text{Inv-Wishart}(\nu, \boldsymbol{S}^{-1})$.
- Since $\boldsymbol{W} = \boldsymbol{LAA}'\boldsymbol{L}'$, we have $\boldsymbol{W}^{-1} = \boldsymbol{L}^{-1}\boldsymbol{A}^{-1}\boldsymbol{A}^{-T}\boldsymbol{L}^{-T}$.

Runtime warnings and convergence problems

Runtime warnings and convergence problems

- As your models become more complicated, you are more likely to run into convergence issues.
- A big advantage of Stan is that it employs a range of diagnostics to let you notice many potential problems with your model.
- Stan is conservative and throws warnings for anything suspicious.

When can warnings be ignored

- In most cases, warnings indicate a problem with the model, rather than the sampler (e.g., improper posterior density, unidentifiable parameters).
- This does not mean that every time you see a warning the model estimates are meaningless, but when you see warnings you should not trust your estimates without first understanding what the warnings mean.

Divergent transitions

Divergent transitions after warmup

- In order to approximate the exact solution of the Hamiltonian dynamics we need to choose a step size governing how far we move each time we evolve the system forward.
- For hard problems the curvature of the posterior can vary a lot and there can be features of the target distribution that are too small.
- For example, divergences are likely if the log posterior density is not continuously differentiable (e.g., triangle distribution).
- Consequently the sampler misses those features and returns biased estimates. Fortunately, this mismatch of scales manifests as divergences which provide a practical diagnostic.

Divergent transitions after warmup

- Even a small number of divergences after warmup cannot be safely ignored if completely reliable inference is desired.

- There are also cases when a small number divergences without any pattern in their locations can be verified to be unimportant, but this cannot be safely assumed without a careful investigation of the model.

Divergent transitions after warmup

- Consider the hierarchical model (8 schools example)

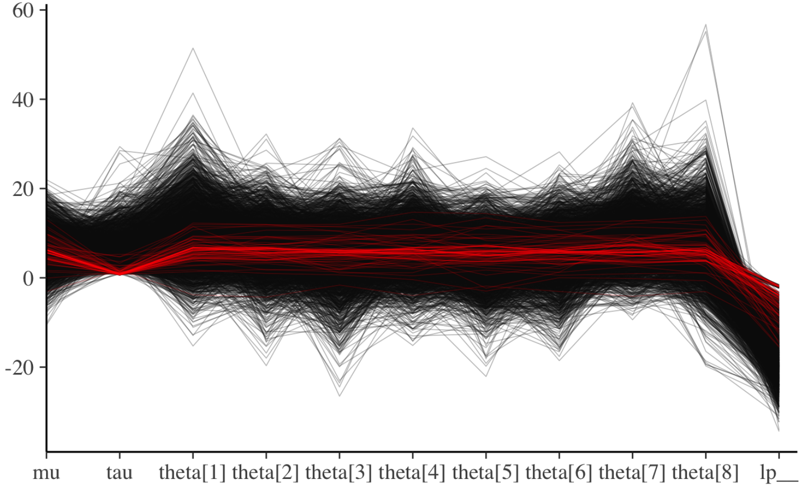
$$\begin{aligned}
y_j &\sim N(\theta_j, \sigma_j^2), \\
\theta_j &\sim N(\mu, \tau^2), \\
\mu &\sim N(0, 10^2), \\
\tau &\sim \text{Half-Cauchy}(0, 10)
\end{aligned}$$

- We run with both the centered and non-centered parameterizations.
- The centered parameterization gave the warning

```
Warning:  There were 85 divergent transitions after
warmup.  See
https://mc-stan.org/misc/warnings.htmldivergent-transitio
to find out why this is a problem and how to eliminate
them.
```

Divergent transitions after warmup

- The mcmc_parcoord function in the bayesplot package shows one line per iteration, connecting the parameter values at this iteration.



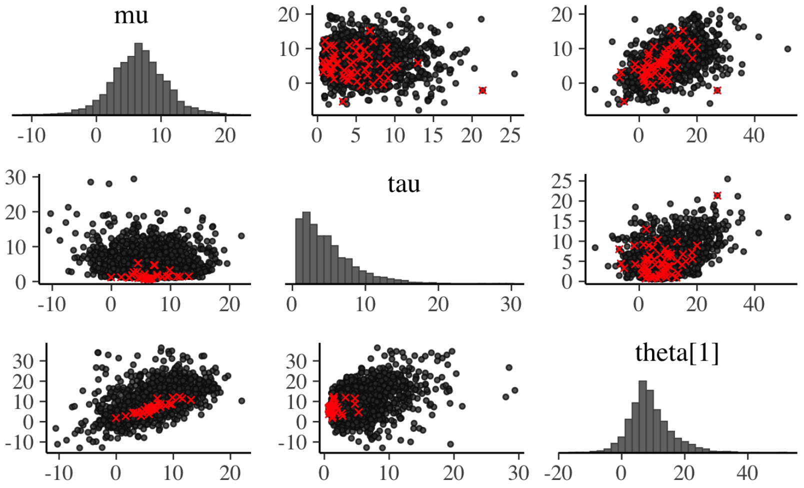
Divergent transitions after warmup

- The divergences in the centered parameterization happen exclusively when τ, the hierarchical standard deviation, goes near zero and thus the values of the θ's are essentially fixed.

- This makes τ immediately suspect.

- The `mcmc_pairs` plot can also be used when the number of parameters is not huge to more easily identify the problem.

Divergent transitions after warmup



Selected References

Selected References I

[1] Yuyan Duan, Keying Ye, and Eric P. Smith. Evaluating water quality using power priors to incorporate historical information. *Environmetrics*, 17:95–106, 2006.

[2] Andrew Gelman. Prior distributions for variance parameters in hierarchical models (comment on article by browne and draper). *Bayesian analysis*, 1(3):515–534, 2006.

[3] Ming-Hui Chen and Joseph G Ibrahim. The relationship between the power prior and hierarchical models. *Bayesian Analysis*, 1(3):551–574, 2006.

[4] Brian P. Hobbs, Daniel J. Sargent, and Bradley P. Carlin. Commensurate priors for incorporating historical information in clinical trials using general and generalized linear models. *Bayesian Analysis*, 7:639–674, 2012.