Hierarchical Bayesian analysis using Stan - From a binary logit to advanced learning models

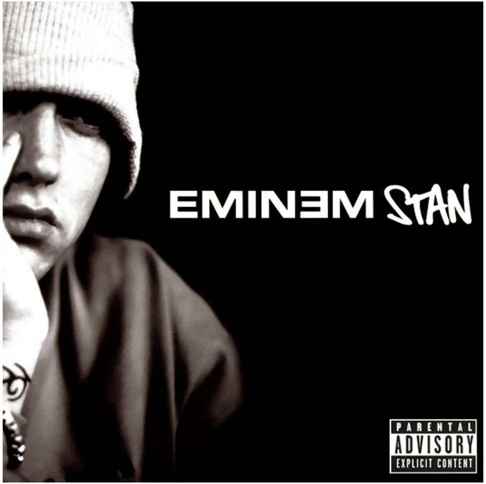
Alina Ferecatu

Rotterdam School of Management,
Erasmus University

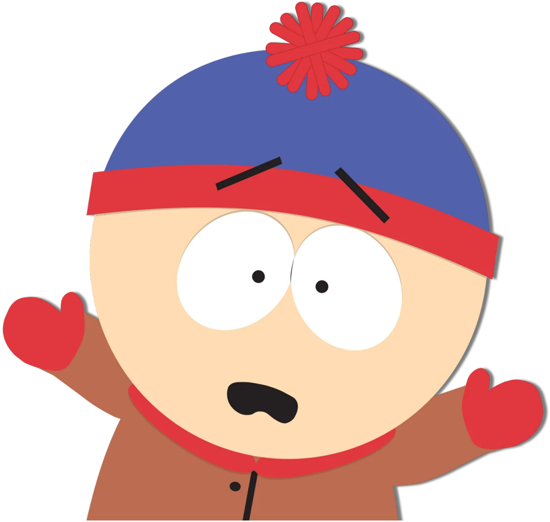
eQMW

November 8, 2022

Stan



Stan

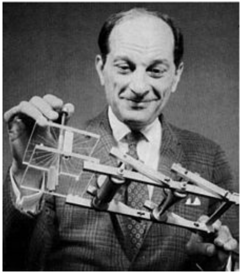


Stan

Probabilistic programming language

Stan

Probabilistic programming language



Stanislaw Ulam
(1909–1984)
Inventor of the Monte Carlo method

Bayes' theorem — quick recap

Bayes' theorem:

$$p(\theta|y,X) \propto p(y|\theta,X) \times p(\theta)$$

where:

- $p(\theta|y,X)$: posterior distribution;
- $p(y|\theta,X)$: model's likelihood;
- $p(\theta)$: prior distribution.

Bayesian workflow

Bayesian workflow

- Exploratory data analysis

Bayesian workflow

- Exploratory data analysis
- Write out a model (full probability model)

Bayesian workflow

- Exploratory data analysis
- Write out a model (full probability model)
- *Prior* predictive checking

Bayesian workflow

- Exploratory data analysis
- Write out a model (full probability model)
- *Prior* predictive checking
- Simulate the model with known parameter values

Bayesian workflow

- Exploratory data analysis
- Write out a model (full probability model)
- *Prior* predictive checking
- Simulate the model with known parameter values
- Model fitting and algorithm diagnostics

Bayesian workflow

- Exploratory data analysis
- Write out a model (full probability model)
- *Prior* predictive checking
- Simulate the model with known parameter values
- Model fitting and algorithm diagnostics
- *Posterior* predictive checking

Bayesian workflow

- Exploratory data analysis
- Write out a model (full probability model)
- *Prior* predictive checking
- Simulate the model with known parameter values
- Model fitting and algorithm diagnostics
- *Posterior* predictive checking
- Model comparison (e.g., via cross-validation)

Stan language

Stan program (with R, Python, Matlab, Julia, Stata, CmdStan interface)

- declares data and (constrained) parameter variables
- defines log posterior (or penalized likelihood)

Stan inference

- Hamiltonian Monte Carlo for full Bayesian estimation
- Variation inference for approximate Bayes
- Optimization for (penalized) Maximum Likelihood

Stan implementation

A Stan model is comprised of *code blocks*:

- *data:* declares the data.
- *transformed data:* makes transformations of the data, including any restrictions on their values.
- *parameters:* declares the parameters.
- *transformed parameters:* makes transformations or restrictions on the parameters.
- *model:* define the full *probability* model here.
- *generated quantities:* outputs from the model (posterior predictions, forecasts).

```
EWA_stan="
data {
......
}

parameters {
.....
}

transformed parameters {
....
}

model {
.....
}

generated quantities{
......
}
"
```

Hierarchical binary logit example: The data

A choice model of buying decisions given price and promotions:

- 3 product attributes (including intercept), indexed by j
- 500 consumers, indexed by i
- 10 purchase occasions each, indexed by t

Hierarchical binary logit example: The data

A choice model of buying decisions given price and promotions:

- 3 product attributes (including intercept), indexed by j
- 500 consumers, indexed by i
- 10 purchase occasions each, indexed by t

	User_ID	Observation	Y	Intercept	Price	Promotion
1	1	1	1	1	0.4242656054	0.429378508
2	1	2	0	1	0.8425126909	0.274047020
3	1	3	1	1	0.3764421751	0.512474872
4	1	4	1	1	0.9115300649	0.355274114
5	1	5	1	1	0.2585383623	0.127846915
6	1	6	1	1	0.2032627692	0.294366778
7	1	7	1	1	0.2246137869	0.161435480
8	1	8	1	1	0.9146362843	0.506660538
9	1	9	1	1	0.8902309975	0.377715006
10	1	10	1	1	0.2795407828	0.185361522
11	2	1	0	1	0.5262798222	0.636969226
12	2	2	0	1	0.9985261350	0.641892214
13	2	3	0	1	0.9664521548	0.662417121
14	2	4	0	1	0.4240157611	0.409965415
15	2	5	0	1	0.2575837581	0.383950177
16	2	6	1	1	0.9943112358	0.431254078

Hierarchical binary logit example: The data

A choice model of buying decisions given price and promotions:

- 3 product attributes (including intercept), indexed by j
- 500 consumers, indexed by i
- 10 purchase occasions each, indexed by t

	User_ID	Observation	Y	Intercept	Price	Promotion
1	1	1	1	1	0.4242656054	0.429378508
2	1	2	0	1	0.8425126909	0.274047020
3	1	3	1	1	0.3764421751	0.512474872
4	1	4	1	1	0.9115300649	0.355274114
5	1	5	1	1	0.2585383623	0.127846915
6	1	6	1	1	0.2032627692	0.294366778
7	1	7	1	1	0.2246137869	0.161435480
8	1	8	1	1	0.9146362843	0.506660538
9	1	9	1	1	0.8902309975	0.377715006
10	1	10	1	1	0.2795407828	0.185361522
11	2	1	0	1	0.5262798222	0.636969226
12	2	2	0	1	0.9985261350	0.641892214
13	2	3	0	1	0.9664521548	0.662417121
14	2	4	0	1	0.4240157611	0.409965415
15	2	5	0	1	0.2575837581	0.383950177
16	2	6	1	1	0.9943112358	0.431254078

Declare data in *stan*

```
"data {
int<lower=1> nvar; // number of parameters in the logit regression
int<lower=0> N; // number of observations
int<lower=1> nind; // number of individuals
int<lower=0,upper=1> y[N];
int<lower=1,upper=nind> ind[N]; // indicator for individuals
row_vector[nvar] x[N];
}
```

Hierarchical binary logit example: The model

A choice model of buying decisions given price and promotions:

- Buy/ Not buy a product with j product attributes;
- i consumers with t purchase occasions each.

Hierarchical binary logit example: The model

A choice model of buying decisions given price and promotions:

- Buy/ Not buy a product with j product attributes;
- i consumers with t purchase occasions each.

$$y_{ijt} \sim \mathcal{B}(p_{ijt})$$

Hierarchical binary logit example: The model

A choice model of buying decisions given price and promotions:

- Buy/ Not buy a product with j product attributes;
- i consumers with t purchase occasions each.

$$y_{ijt} \sim \mathcal{B}(p_{ijt})$$

$$logit(p_{ijt}) \sim \mathcal{N}(x_{ijt}\beta_{ij})$$

Hierarchical binary logit example: The model

A choice model of buying decisions given price and promotions:

- Buy/ Not buy a product with j product attributes;
- i consumers with t purchase occasions each.

$$
\begin{aligned}
y_{ijt} &\sim \mathcal{B}(p_{ijt}) \\
logit(p_{ijt}) &\sim \mathcal{N}(x_{ijt}\beta_{ij}) \\
\beta_i &\sim MultiNormal(z_i\delta, \Sigma)
\end{aligned}
$$

Hierarchical binary logit example: The model

A choice model of buying decisions given price and promotions:

- Buy/ Not buy a product with j product attributes;
- i consumers with t purchase occasions each.

$$
\begin{aligned}
y_{ijt} &\sim \mathcal{B}(p_{ijt}) \\
logit(p_{ijt}) &\sim \mathcal{N}(x_{ijt}\beta_{ij}) \\
\beta_i &\sim MultiNormal(z_i\delta, \Sigma) \\
\delta &\sim \mathcal{N}(\delta_0, \sigma)
\end{aligned}
$$

Hierarchical binary logit example: The model

A choice model of buying decisions given price and promotions:

- Buy/ Not buy a product with j product attributes;
- i consumers with t purchase occasions each.

$$
\begin{aligned}
y_{ijt} &\sim \mathcal{B}(p_{ijt}) \\
logit(p_{ijt}) &\sim \mathcal{N}(x_{ijt}\beta_{ij}) \\
\beta_i &\sim MultiNormal(z_i\delta, \Sigma) \\
\delta &\sim \mathcal{N}(\delta_0, \sigma)
\end{aligned}
$$

Non-conjugate prior

$$
\begin{aligned}
\Sigma &= diag(\tau)\,\Omega\,diag(\tau) \\
\tau &\sim \Gamma(a, b) \\
\Omega &\sim LKJcorr(\nu)
\end{aligned}
$$

Hierarchical binary logit example: The model

A choice model of buying decisions given price and promotions:

- Buy/ Not buy a product with j product attributes;
- i consumers with t purchase occasions each.

$$y_{ijt} \sim \mathcal{B}(p_{ijt})$$

$$logit(p_{ijt}) \sim \mathcal{N}(x_{ijt}\beta_{ij})$$

$$\beta_i \sim MultiNormal(z_i\delta, \Sigma)$$

$$\delta \sim \mathcal{N}(\delta_0, \sigma)$$

Non-conjugate prior

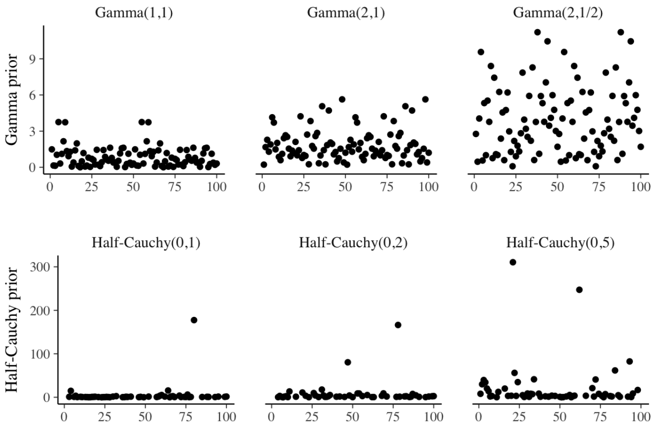
$$\Sigma = diag(\tau)\,\Omega\,diag(\tau)$$

$$\tau \sim \Gamma(a, b)$$

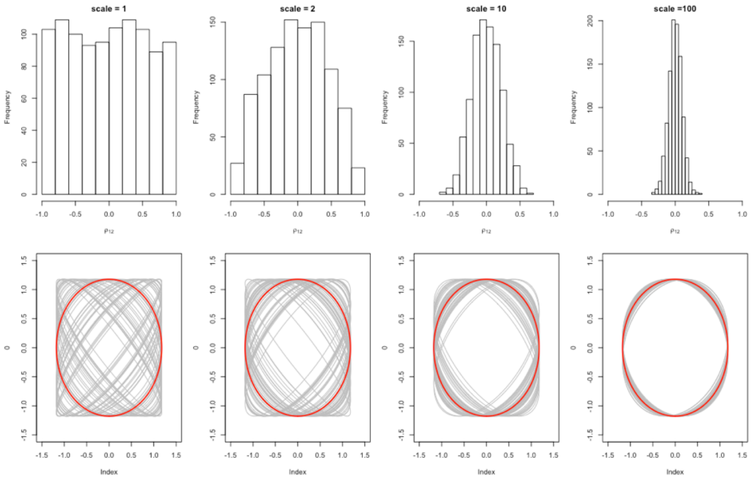
$$\Omega \sim LKJcorr(\nu)$$

```
hierarchical_binlogit_fullcov="data {
....
}

parameters {
vector[nvar] delta;
vector<lower=0>[nvar] tau;
vector[nvar] beta[nind];
corr_matrix[nvar] Omega; // Vbeta - prior correlation
}

model {
to_vector(delta) ~ normal(0, 5);
to_vector(tau) ~ gamma(2, 0.5);
Omega ~ lkj_corr(2);

for (h in 1:nind)
beta[h]~multi_normal(delta, quad_form_diag(Omega, tau));

for (n in 1:N)
y[n] ~ bernoulli_logit(x[n] * beta[ind[n]]);
}

generated quantities {
corr_matrix[nvar] Omega_corr;
int z[N];
real log_lik[N];

Omega_corr=Omega;
for (n in 1:N){
z[n] = bernoulli_logit_rng(x[n] * beta[ind[n]]);
log_lik[n]= bernoulli_logit_lpmf(y[n]|x[n] * beta[ind[n]]);
}

}
"
```

Choice of prior distribution of the variance components



Choice of the LKJ prior distribution



Checking model fit

- Lack of mixing.
- Stationarity.
- Autocorrelation.
- Divergent transitions.

Checking model fit

- Lack of mixing.
- Stationarity.
- Autocorrelation.
- Divergent transitions.

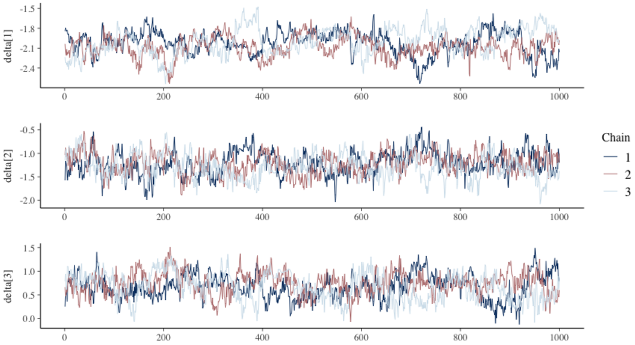


Figure 1: Traceplot of δ parameters (after burnin), using package *bayesplot*

Summary statistics

Effective sample size and convergence properties

	mean	se_mean	sd	2.5%	97.5%	n_eff	Rhat
delta[1]	-2.0226636	0.01805745	0.1661046	-2.3656187	-1.7061194	84.61558	1.034883
delta[2]	-1.2509968	0.02373062	0.2545043	-1.7638505	-0.7585503	115.01969	1.019059
delta[3]	0.6711088	0.01967228	0.2450191	0.1780753	1.1415407	155.12809	1.019018

Summary statistics

Effective sample size and convergence properties

	mean	se_mean	sd	2.5%	97.5%	n_eff	Rhat
delta[1]	-2.0226636	0.01805745	0.1661046	-2.3656187	-1.7061194	84.61558	1.034883
delta[2]	-1.2509968	0.02373062	0.2545043	-1.7638505	-0.7585503	115.01969	1.019059
delta[3]	0.6711088	0.01967228	0.2450191	0.1780753	1.1415407	155.12809	1.019018

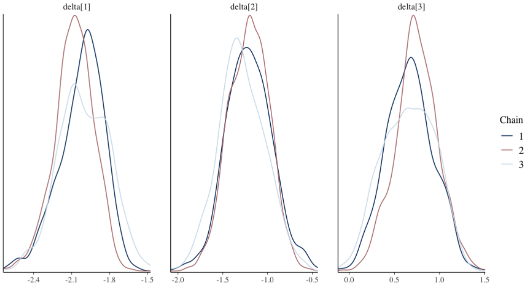


Figure 2: Density plot of δ parameters (after burnin), using package *bayesplot*

Noncentered (Re)Parameterization - the "Matt Trick"

Consider a model with a diagonal variance-covariance matrix

- Assume our intercept model: $\beta_i \sim \mathcal{N}(\delta, \tau)$
- We can decompose that into: $\mathcal{N}(\delta, \sigma) = \delta + \tau\mathcal{N}(0, 1)$
- The trick applies to other distributions in the location-scale family
- The transformation:
    1. declare α_i in the parameters block and β_i in the transformed parameters block
    2. draw $\alpha_i \sim \mathcal{N}(0, 1)$ & $\tau \sim \Gamma(a, b)$
    3. compute $\beta_i = \delta + \tau\alpha_i$

Noncentered (Re)Parameterization - the multivariate case

- Assume our full model: $\beta_{\mathbf{i}} \sim MultiNormal(\delta, \Sigma)$
  - If Σ_{jj} is small, then β_{ij} needs to fall into a small range, the sampler needs a small step size
  - If Σ_{jj} is large, then β_{ij} can fall into a wide range, the sampler needs a large step size or lots of small steps
- The transformation:
  1. declare α_i in the parameters block and β_i in the transformed parameters block.
  2. draw $\alpha_i \sim \mathcal{N}(0,1)$ & $\tau \sim \Gamma(a,b)$.
  3. compute $\beta_{\mathbf{i}} = \delta + \tau \mathbf{L} \alpha_{\mathbf{i}} \sim \mathcal{N}(\delta, \tau^2 \mathbf{L}\mathbf{L}^T)$,
  4. where $\tau \mathbf{L}$ is the Cholesky factor of $\Sigma = \tau^2 \mathbf{L}\mathbf{L}^T$, and τ is the standard deviation of the errors.

Noncentered (Re)Parameterization - the implementation

$$y_{ijt} \sim \mathcal{B}(p_{ijt})$$

$$logit(p_{ijt}) \sim \mathcal{N}(x_{ijt}\beta_{ij})$$

$$\beta_{\mathbf{i}} = \delta + \tau \mathbf{L} \alpha_{\mathbf{i}}$$

$$\alpha_i \sim \mathcal{N}(0, 1)$$

$$\delta \sim \mathcal{N}(\delta_0, \sigma)$$

Non-conjugate prior

$$\tau \sim \Gamma(a, b)$$

$$\Omega \sim LKJcorr(\nu)$$

```
hierarchical_binlogit_fullcov_noncentered="data {
.....
}

parameters {
matrix[nvar, nind] alpha; // nvar*H parameter matrix
row_vector[nvar] delta;
vector<lower=0>[nvar] tau;
cholesky_factor_corr[nvar] L_Omega;
}


transformed parameters{
row_vector[nvar] beta[nind];
matrix[nind,nvar] Vbeta_reparametrized;
Vbeta_reparametrized = (diag_pre_multiply(tau, L_Omega)*alpha)';

for (h in 1:nind)
beta[h]=delta+Vbeta_reparametrized[h];
}

model {
L_Omega~lkj_corr_cholesky(2);
to_vector(delta) ~ normal(0, 5);
to_vector(tau) ~ gamma(2, 0.5);
to_vector(alpha)~ normal(0,1);

for (n in 1:N)
y[n] ~ bernoulli_logit(beta[ind[n]]*x[n]);
}

generated quantities {
corr_matrix[nvar] Omega;
int z[N];
real log_lik[N];

Omega=L_Omega*L_Omega';

for (n in 1:N){
z[n] = bernoulli_logit_rng(beta[ind[n]]*x[n]);
log_lik[n]= bernoulli_logit_lpmf(y[n]|beta[ind[n]]*x[n]);
}

}
"
```

Traceplots of model parameters

The noncentered reparametrization helps tremendously

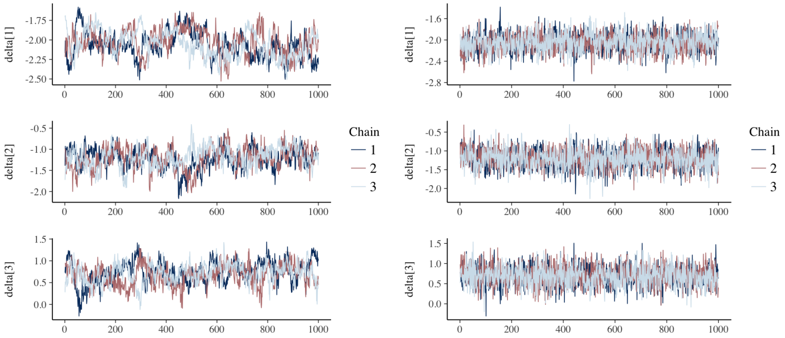
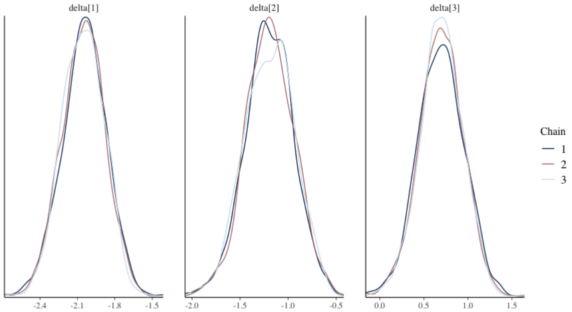


Figure 3: Traceplot of δ parameters (after burnin), estimated via the centered vs. noncentered parametrization

Summary statistics

Effective sample size and convergence properties

```
$summary
                mean     se_mean        sd       2.5%      97.5%    n_eff      Rhat
delta[1] -2.0508997 0.005138191 0.1701534 -2.3861202 -1.7288254 1096.632 1.0009678
delta[2] -1.1951041 0.006205318 0.2535605 -1.7052461 -0.7068838 1669.688 0.9998616
delta[3]  0.6966904 0.006432588 0.2470562  0.2011148  1.1660746 1475.095 1.0001598
```



Individual level parameters

Most parameters are within the 95% highest density intervals

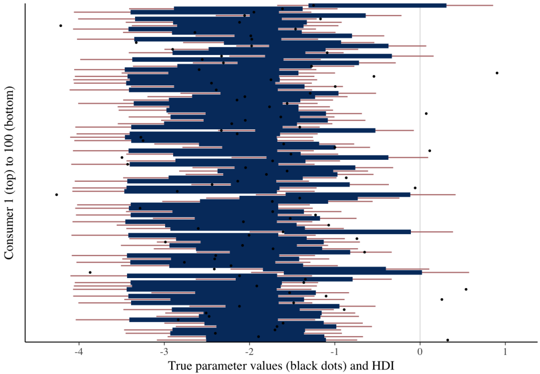


Figure 5: True values (black dots) and the 80% and 95% highest density intervals for the intercept, for the first 100 consumers

Model checking

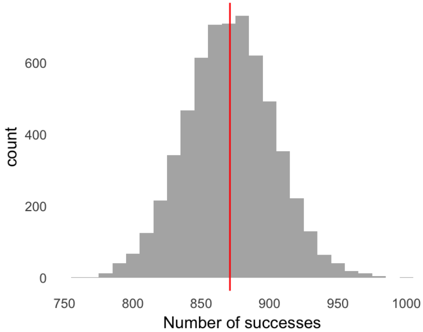


Figure 6: Number of successes: posterior replications vs. true value

Model checking

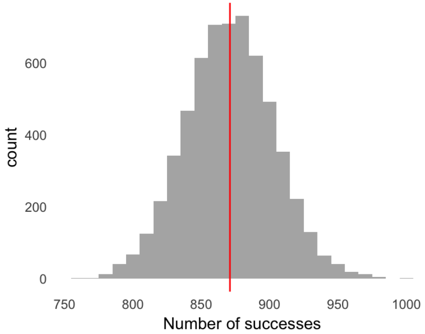


Figure 6: Number of successes: posterior replications vs. true value

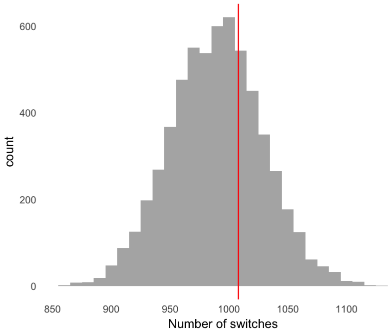


Figure 7: Switches between buying/ not buying: posterior replications vs. true value

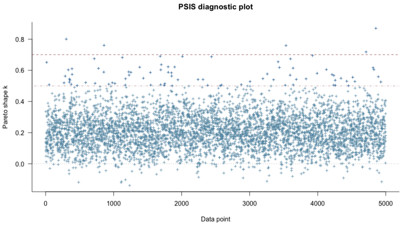
Compute hit rates and MSEs based on posterior replications

Model comparison

Likelihood-based measures: *Leave-one-out cross-validation*

Table 1: Model comparison based on LOO-CV, using package *loo*

	Variance model		*Full covariance model*		*NCP model*	
	Estimate	SE	Estimate	SE	Estimate	SE
elpd_loo	-1874.3	43.1	-1871.9	43.2	-1871.2	43
p_loo	398.6	12.5	364.4	11.8	363.7	11.8
looic	3748.6	86.2	3743.7	86.4	3742.4	86.5



Stan resources

Stan ecosystem

- lang, math library (C++)
- interfaces and tools (R, Python, many more)
- documentation (example model repo, user guide & reference manual, case studies, R package vignettes)
- online community (Stan Forums on Discourse)

Libraries implementing Stan

- rstanarm: complex hierarchical models
- hBayesDM: behavioral decision making models (stan codes on GitHub)
- bayesplot: data visualization

More Stan resources

StanCon 2018 talks: Link

Books:

- Bayesian Data Analysis: aka the Bible:)
  Link
- Bayesian Statistics using Stan
  Link
- Statistical Rethinking
  Link

github.com/alinafere/eQMW_stan_tutorial