MCMC examples

```
library(MCMCglmm)  ## older, Gibbs-sampling
library(brms)      ## newest, lme4-like syntax, very flexible, compiled
library(rstanarm)  ## lme4-like syntax, pre-compiled
library(lme4)      ## to get data
options(brms.backend = "cmdstanr")
library(broom.mixed)  ## 'tidy'
library(tidybayes)  ## convenience functions for getting MCMC output in 'tidy' format
library(bayesplot)
library(bayestestR)  ## diagnostics
library(ggplot2); theme_set(theme_bw())
library(shinystan)  ## diagnostics for Stan in a Shiny window
library(tidyverse)  ## general-purpose manipulations
```

- a little more on priors:
  - parameter-expanded priors: $y_j|\mu,\xi_j \sim N(\mu + \alpha\sigma_j, \sigma_j^2)$, $\sigma_j \sim N(0,\sigma_\xi^2)$; $\alpha \sim N(\alpha_0,\sigma_\alpha)$, $\sigma_\alpha \sim$ inverse-Gamma(ν)

```
df(v/alpha.V, df1 = 1, df2 = nu, ncp = (alpha.mu^2)/alpha.V)
2 * dt(sqrt(v)/sqrt(alpha.V), df = nu, ncp = alpha.mu/sqrt(alpha.V))
```

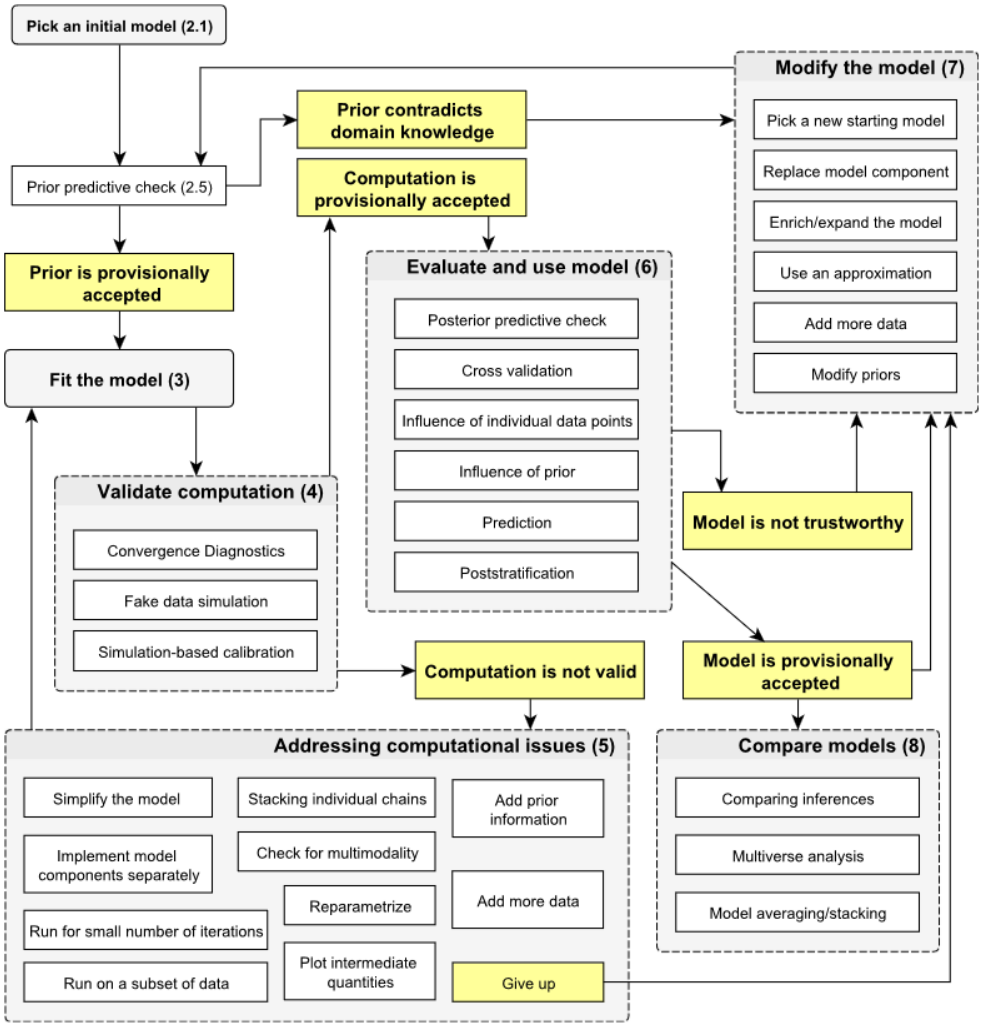
… always set `alpha.mu=0`, can set `V = 1` (or `diag()` in more complex cases) wlog; `sqrt(alpha.V)` (scale) and `nu` are the only relevant parameters

effective sample size

- number of samples, corrected for autocorrelation
- ESS may be > sample size! (e.g. *antithetic sampling*)
- efficiency of a sampler is not (samples/time), but (effective samples/time)
- effective sample size >1000 for both tail and bulk quantities (Vehtari et al. 2021)

Bayesian workflow

Gelman et al. (2020)



simulation-based calibration

Talts et al. (2020)

default priors/prior predictive simulations:

- `rstanarm` default priors

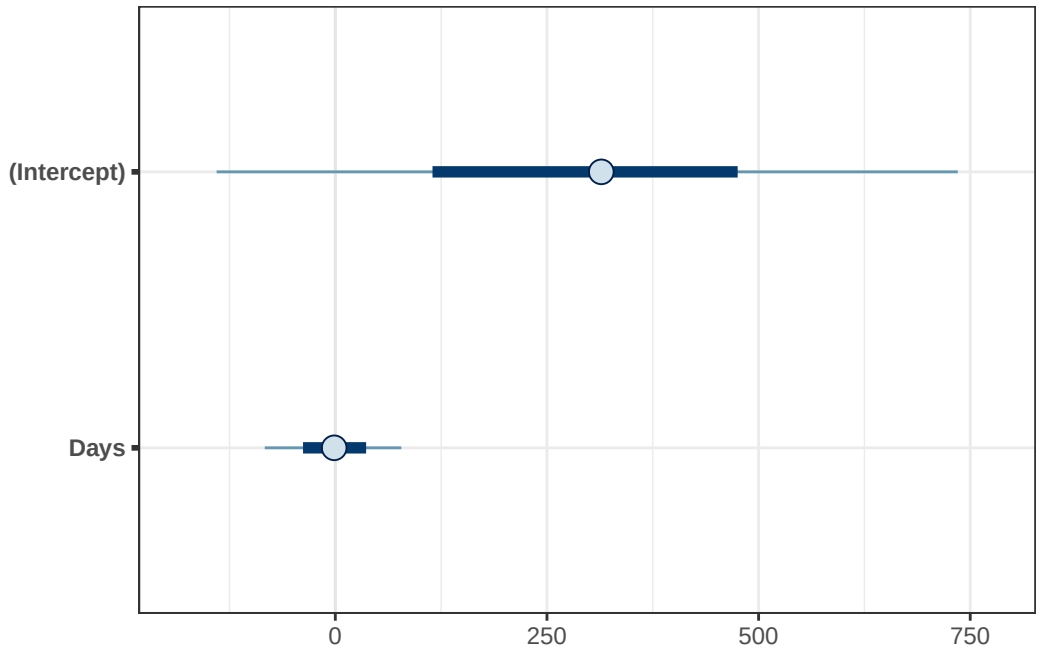
Using the good old `sleepstudy` example:

```
priorpred <- stan_lmer(Reaction ~ Days + (Days|Subject),
                       prior_PD = TRUE, data = sleepstudy, chains = 1,
                       seed = 101,
                       refresh = 0)
```

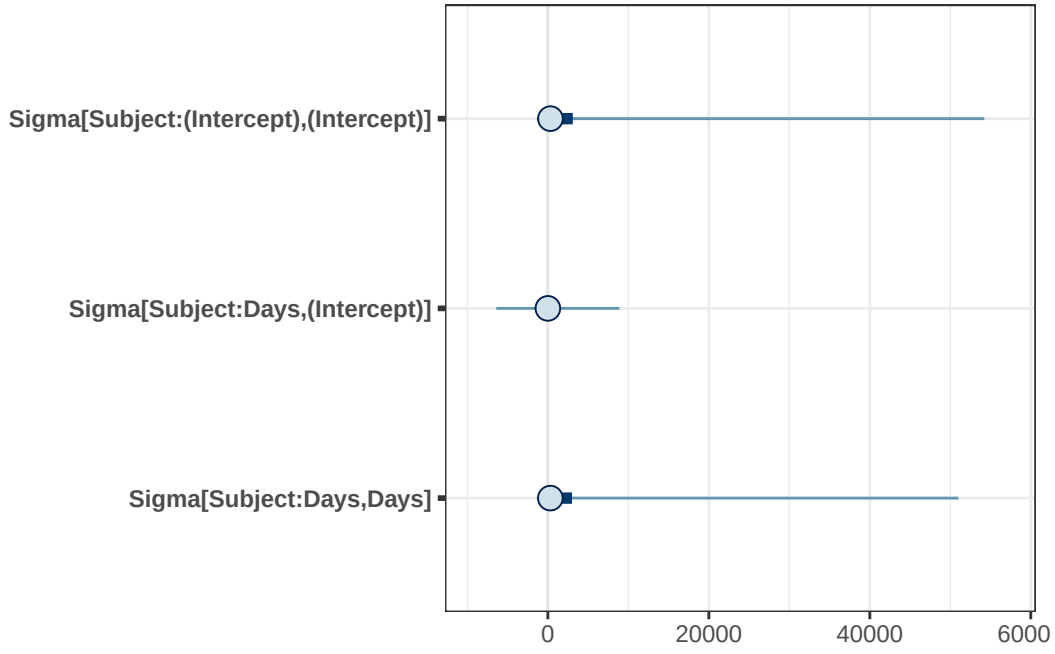
```
prior_summary(priorpred)
```

```
Priors for model 'priorpred'
------
Intercept (after predictors centered)
  Specified prior:
    ~ normal(location = 299, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 299, scale = 141)

Coefficients
  Specified prior:
    ~ normal(location = 0, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 0, scale = 49)

Auxiliary (sigma)
  Specified prior:
    ~ exponential(rate = 1)
  Adjusted prior:
    ~ exponential(rate = 0.018)

Covariance
 ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
------
See help('prior_summary.stanreg') for more details
```

```
plot(priorpred, pars = c("(Intercept)", "Days"))
```




```
plot(priorpred, regex_pars = "Sigma")
```



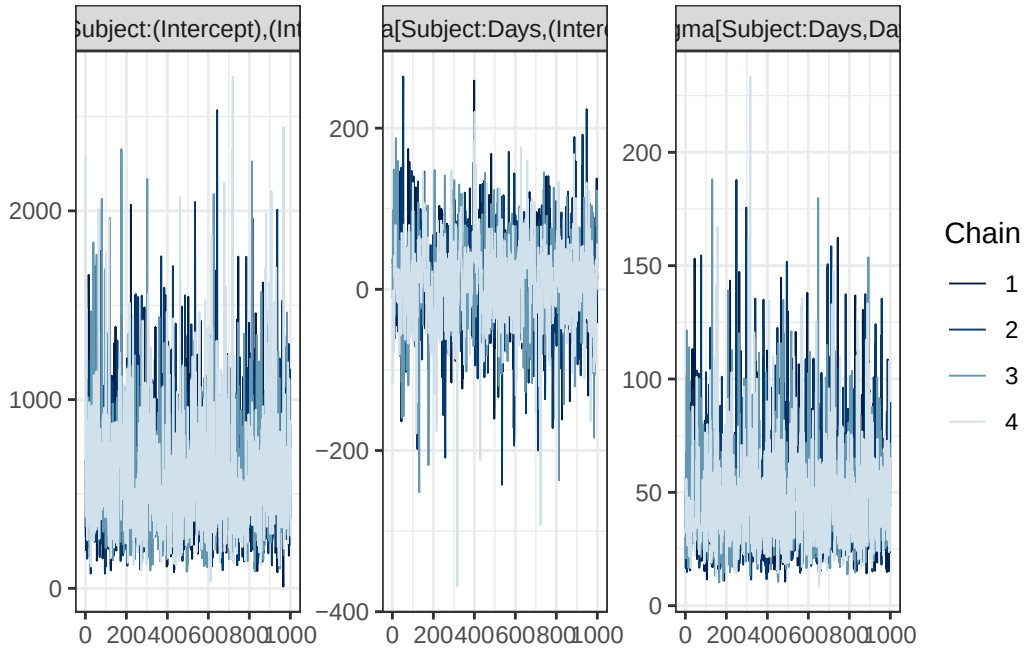

```
stanfit <- stan_lmer(Reaction ~ Days + (Days|Subject),
                     data = sleepstudy, chains = 4)
```

```
print(bayestestR::diagnostic_posterior(stanfit),      digits = 4)
```

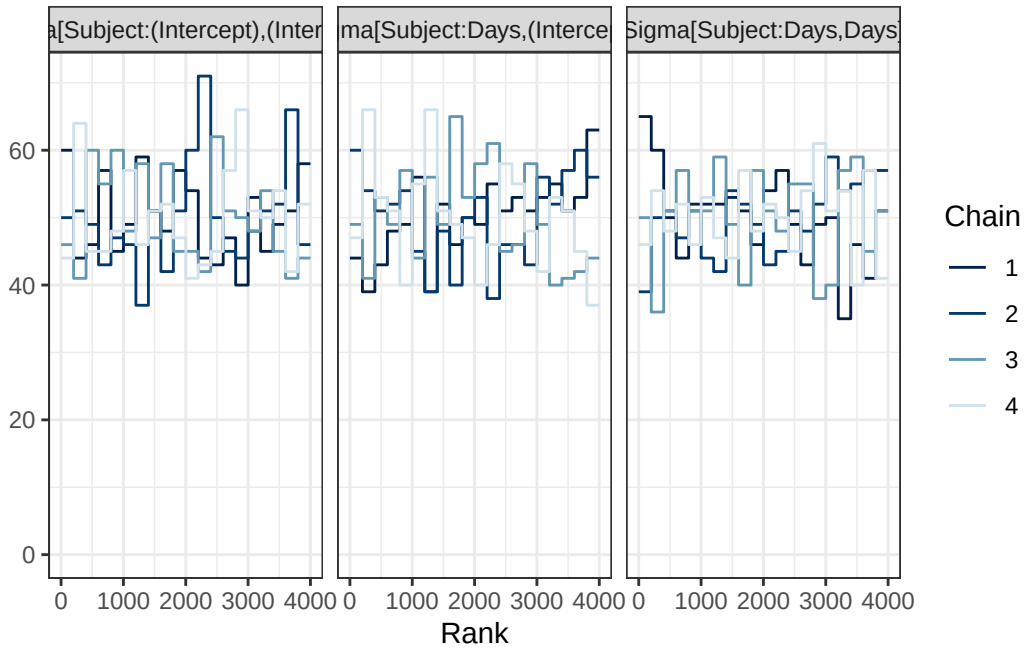
```
     Parameter  Rhat  ESS    MCSE
1  (Intercept) 1.001 2142 0.14337
38        Days 1.002 1353 0.04635
```

```
launch_shinystan(stanfit)
```

```
mcmc_trace(stanfit, regex_pars= "Sigma")
```




```
mcmc_rank_overlay(stanfit, regex_pars= "Sigma")
```



- MCMC diagnostics
  - trace plots, improved trace plots
  - R-hat Vehtari et al. (2021)
  - divergences (HMC only)

See http://bbolker.github.io/bbmisc/bayes/examples.html

doing stuff with the results

```
tidy(stanfit, effects=c("fixed", "ran_pars"), conf.int = TRUE)
```

```
# A tibble: 6 x 6
  term                          estimate std.error conf.low conf.high group
  <chr>                            <dbl>     <dbl>    <dbl>     <dbl> <chr>
1 (Intercept)                     251.        6.35   240.       262.  <NA>
2 Days                             10.4       1.63     7.48      13.1 <NA>
3 sd_(Intercept).Subject           24.2      NA       NA         NA   Subject
4 sd_Days.Subject                   6.88     NA       NA         NA   Subject
5 cor_(Intercept).Days.Subject      0.0735   NA       NA         NA   Subject
6 sd_Observation.Residual          26.0      NA       NA         NA   Residual
```

¿¿ why don't we get confidence intervals ?? Do it by hand ...

```
(as_draws(stanfit)
    |> tidyr::pivot_longer(everything())
    |> group_by(name)
    |> summarise(estimate = median(value),
                 lwr = quantile(value, 0.025),
                 upr = quantile(value, 0.975))
    |> filter(!stringr::str_detect(name, "^b\\["))
)
```

```
# A tibble: 9 x 4
  name                                    estimate    lwr    upr
  <chr>                                      <dbl>  <dbl>  <dbl>
1 (Intercept)                                251.  238.    264.
2 .chain                                       2.5   1       4
3 .draw                                     2000.  101.   3900.
4 .iteration                                 500.   26.0   975.
5 Days                                        10.4   6.92   13.6
6 Sigma[Subject:(Intercept),(Intercept)]     515.  161.   1376.
7 Sigma[Subject:Days,(Intercept)]             15.1 -99.4   109.
8 Sigma[Subject:Days,Days]                    42.3  18.0   107.
9 sigma                                       25.9  23.1    29.4
```

```
form1 <- Reaction ~ Days + (Days|Subject)
get_prior(form1, sleepstudy)
```

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

```
     default
 (vectorized)
     default
     default
 (vectorized)
 (vectorized)
 (vectorized)
     default
```

```
b_prior <- c(set_prior("normal(200, 50)", "Intercept"),
             set_prior("normal(0, 10)", "b"),
             set_prior("normal(0, 1)", "sigma")
             )
```

```
b <- brm(form1, sleepstudy,
         prior = b_prior,
         seed = 101,              ## reproducibility
         sample_prior = 'only',   ## for prior predictive sim
         chains = 1, iter = 500,  ## very short sample for convenience
         silent = 2, refresh = 0  ## be vewy vewy quiet ...
         )
p_df <- sleepstudy |> tidybayes::add_predicted_draws(b)
```

'spaghetti plot' of prior preds

```
gg0 <- ggplot(p_df,aes(Days, .prediction, group=interaction(Subject,.draw))) +
        geom_line(alpha = 0.1)
```

```
b_prior4 <- c(set_prior("normal(200, 5)", "Intercept"),
              set_prior("normal(0, 2)", "b"),
              set_prior("normal(0, 1)", "sd"),
              set_prior("normal(0, 1)", "sigma")
              )
cc <-capture.output(
    suppressMessages(
    b_reg <- brm(form1, sleepstudy,
             prior = b_prior4,
             seed = 101,
             init = 0,
             control = list(adapt_delta = 0.95)
             ))
)
```

I've used `suppressMessages` to get rid of a lot of messages like

> Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of the following issue: Exception: normal_id_glm_lpdf: Scale vector is inf, but must be positive finite! (in '/tmp/RtmpSSmixI/model-6899b70c2b466.stan', line 74, column 4 to column 55) If this warning occurs sporadically, such as for highly constrained variable types like covariance matrices, then the sampler is fine, but if this warning occurs often then your model may be either severely ill-conditioned or misspecified.

Suppressing all messages is generally a bad idea (it might suppress other messages that you do want to see), but there's no obvious way to suppress just these messages *when they occur in the warmup phase*, which seems to be a harmless case.

From the Stan forums:

> This is common and not a problem, the algorithm explores a large range of values in the warm-up phase and often triggers numerical problems that go away.

```
print(bayestestR::diagnostic_posterior(b_reg),      digits = 4)
```

```
    Parameter   Rhat  ESS    MCSE
1      b_Days 0.9999 3998 0.01643
2 b_Intercept 1.0001 2952 0.05493
```

```
## debug(MCMCglmm:::priorformat)
m <- MCMCglmm(Reaction ~ Days, random = ~us(1+Days):Subject,
              data = sleepstudy,
              verbose=FALSE,
              prior = list(G=list(G1=list(V=diag(2), nu = 0.1))))
broom.mixed::tidy(m)
```

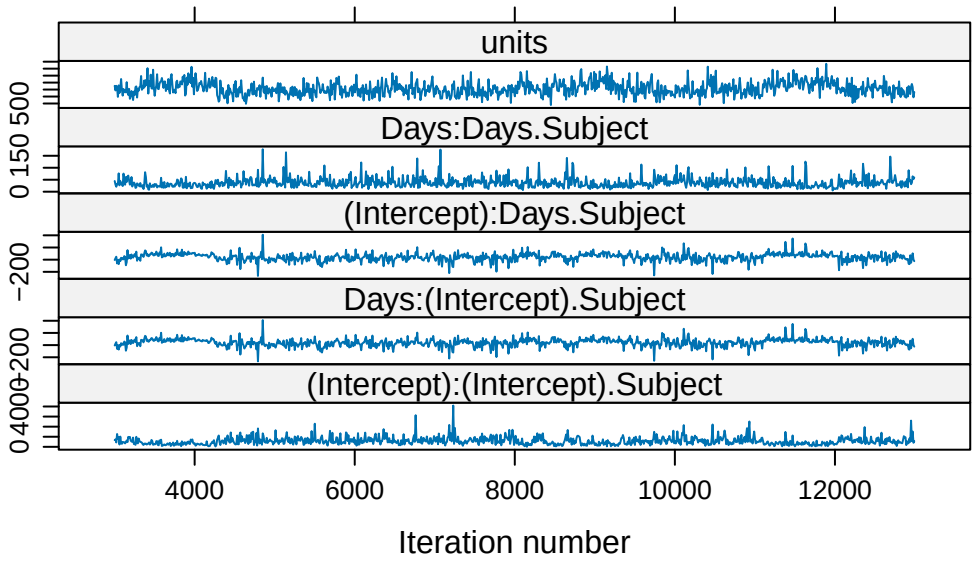
```
# A tibble: 6 x 5
  effect   group    term                  estimate std.error
  <chr>    <chr>    <chr>                    <dbl>     <dbl>
1 fixed    <NA>     (Intercept)              251.       6.97
2 fixed    <NA>     Days                      10.5      1.63
3 ran_pars Subject  var__(Intercept)         586.     384.
4 ran_pars Subject  cov__(Intercept).Days     39.1     60.9
5 ran_pars Subject  var__Days                 37.3     17.8
6 ran_pars Residual var__Observation         705.     102.
```

```
try(MCMCglmm(Reaction ~ Days, random = ~us(1+Days):Subject,
               data = sleepstudy,
               verbose=FALSE,
               prior = list(G=list(G1=list(V=diag(2), nu = 0.1,
                                           alpha.mu = 0, alpha.V = diag(2))))))
```

```
Error in priorformat(if (NOpriorG) { :
  alpha.mu is the wrong length for some prior$G/prior$R elements
```

```
m2 <- MCMCglmm(Reaction ~ Days, random = ~us(1+Days):Subject,
               data = sleepstudy,
               verbose=FALSE,
               prior = list(G=list(G1=list(V=diag(2), nu = 0.1,
                                           alpha.mu = rep(0,2),
                                           alpha.V = diag(2)))))
```

```
lattice::xyplot(m2$VCV)
```



Run longer (and thin)? Strengthen prior?

to do

- test silencing of `brms` messages
- improve `tidy` for `rstanarm`

- better ways to get draws
- prior pred sims for `MCMCglmm`? (examples of parameter-expansion)
- SBC examples?
- figure out compilation caching for `brms`?
- contact Hadfield about `MCMCglmm` tweaks

Gelman, Andrew, Aki Vehtari, Daniel Simpson, Charles C. Margossian, Bob Carpenter, Yuling Yao, Lauren Kennedy, Jonah Gabry, Paul-Christian Bürkner, and Martin Modrák. 2020. "Bayesian Workflow." *arXiv:2011.01808 [Stat]*, November. http://arxiv.org/abs/2011.01808.

Talts, Sean, Michael Betancourt, Daniel Simpson, Aki Vehtari, and Andrew Gelman. 2020. "Validating Bayesian Inference Algorithms with Simulation-Based Calibration." *arXiv:1804.06788 [Stat]*, October. http://arxiv.org/abs/1804.06788.

Vehtari, Aki, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. 2021. "Rank-Normalization, Folding, and Localization: An Improved R-hat for Assessing Convergence of MCMC (with Discussion)." *Bayesian Analysis* 16 (2): 667–718. https://doi.org/10.1214/20-BA1221.