MLM custom code

Carl F. Falk

2025-08-29

Setup

```
# load packages
library(brms)
```

```
## Loading required package: Rcpp
```

```
## Loading 'brms' package (version 2.21.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
```

```
##
## Attaching package: 'brms'
```

```
## The following object is masked from 'package:stats':
##
##     ar
```

```
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.32.6 (Stan version 2.32.2)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```
library(loo)
```

```
## This is loo version 2.7.0
```

```
## - Online documentation and vignettes at mc-stan.org/loo
```

```
## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' ar
```

```
## - Windows 10 users: loo may be very slow if 'mc.cores' is set in your .Rprofile file (see https://gi
```

```
##
## Attaching package: 'loo'
```

```
## The following object is masked from 'package:rstan':
##
##     loo
```

```
library(posterior)
```

```
## This is posterior version 1.5.0
```

```
##
## Attaching package: 'posterior'
```

```
## The following objects are masked from 'package:rstan':
##
##     ess_bulk, ess_tail
```

```
## The following objects are masked from 'package:stats':
##
##     mad, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     %in%, match
```

```
library(tictoc)

source("fx.R") # function
```

```
## Loading required package: coda
```

```
##
## Attaching package: 'coda'
```

```
## The following object is masked from 'package:rstan':
##
##     traceplot
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 4.4.2
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
```

```
## ## Copyright (C) 2003-2025 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
```

```
## ##
## ## Support provided by the U.S. National Science Foundation
```

```
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
##
## Attaching package: 'MCMCpack'
```

```
## The following objects are masked from 'package:brms':
##
##     ddirichlet, rdirichlet
```

```
##
## Attaching package: 'matrixcalc'
```

```
## The following object is masked from 'package:MCMCpack':
##
```

```
##     vech
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'lme4'
```

```
## The following object is masked from 'package:brms':
##
##     ngrps
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# This was setup from simulation study...
nchains <- 2
ncores <- 2
iter <- 5000
```

Load data

```
dat <- read.csv("example_dataset_77.csv")
```

Setup of models

```
modA <- y ~ 1 + X1 + X2 + (X1|g)
modB <- y ~ 1 + X1 + X2 + (1|g)
modC <- y ~ 1 + X1 + X2 + (X1+X2|g)
modD <- y ~ 1 + X1 + (X1|g)
modE <- y ~ 1 + X1 + X2 + X3 + (X1|g)
```

brms

Estimate Models

```
tic()
modA.brms = brm(modA, data=dat, chains=nchains, iter=iter, silent = 2,
                cores=ncores, refresh = 0)
# stancode(modA.brms) # to look at underlying stan code
toc()
```

```
## 180.2 sec elapsed
```

```
tic()
modB.brms = brm(modB, data=dat, chains=nchains, iter=iter, silent = 2,
```

```
                cores=ncores, refresh = 0)
toc()
```

```
## 126.77 sec elapsed
```

```
tic()
modC.brms = brm(modC, data=dat, chains=nchains, iter=iter, silent = 2,
                cores=ncores, refresh = 0)
toc()
```

```
## 231.67 sec elapsed
```

```
tic()
modD.brms = brm(modD, data=dat, chains=nchains, iter=iter, silent = 2,
                cores=ncores, refresh = 0)
toc()
```

```
## 157.57 sec elapsed
```

```
tic()
modE.brms = brm(modE, data=dat, chains=nchains, iter=iter, silent = 2,
                cores=ncores, refresh = 0)
toc()
```

```
## 154.65 sec elapsed
```

Custom Marginal Likelihood code

Estimate Models

```
tic()
matsA <- mod.mat.mlm(modA, data=dat)
modA.custom <- stan(file='mlmmarg.stan', data = matsA,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 600.21 sec elapsed
```

```
tic()
matsB <- mod.mat.mlm(modB, data=dat)
modB.custom <- stan(file='mlmmarg.stan', data = matsB,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 345.84 sec elapsed
```

```
tic()
matsC <- mod.mat.mlm(modC, data=dat)
modC.custom <- stan(file='mlmmarg.stan', data = matsC,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 617.09 sec elapsed
```

```
tic()
matsD <- mod.mat.mlm(modD, data=dat)
modD.custom <- stan(file='mlmmarg.stan', data = matsD,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 418.19 sec elapsed
```

```
tic()
matsE <- mod.mat.mlm(modE, data=dat)
modE.custom <- stan(file='mlmmarg.stan', data = matsE,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 450.69 sec elapsed
```

Look at Diagnostics

Rhats

```
# error sd, coefficients, random effect covariance matrix
pars <- c("sig", "betas", "Tau")
max(abs(summary(modA.custom, pars=pars)$summary[,"Rhat"]))
```

```
## [1] 1.000005
```

```
max(abs(summary(modB.custom, pars=pars)$summary[,"Rhat"]))
```

```
## [1] 1.000446
```

```
max(abs(summary(modC.custom, pars=pars)$summary[,"Rhat"]))
```

```
## [1] 1.000493
```

```
max(abs(summary(modD.custom, pars=pars)$summary[,"Rhat"]))
```

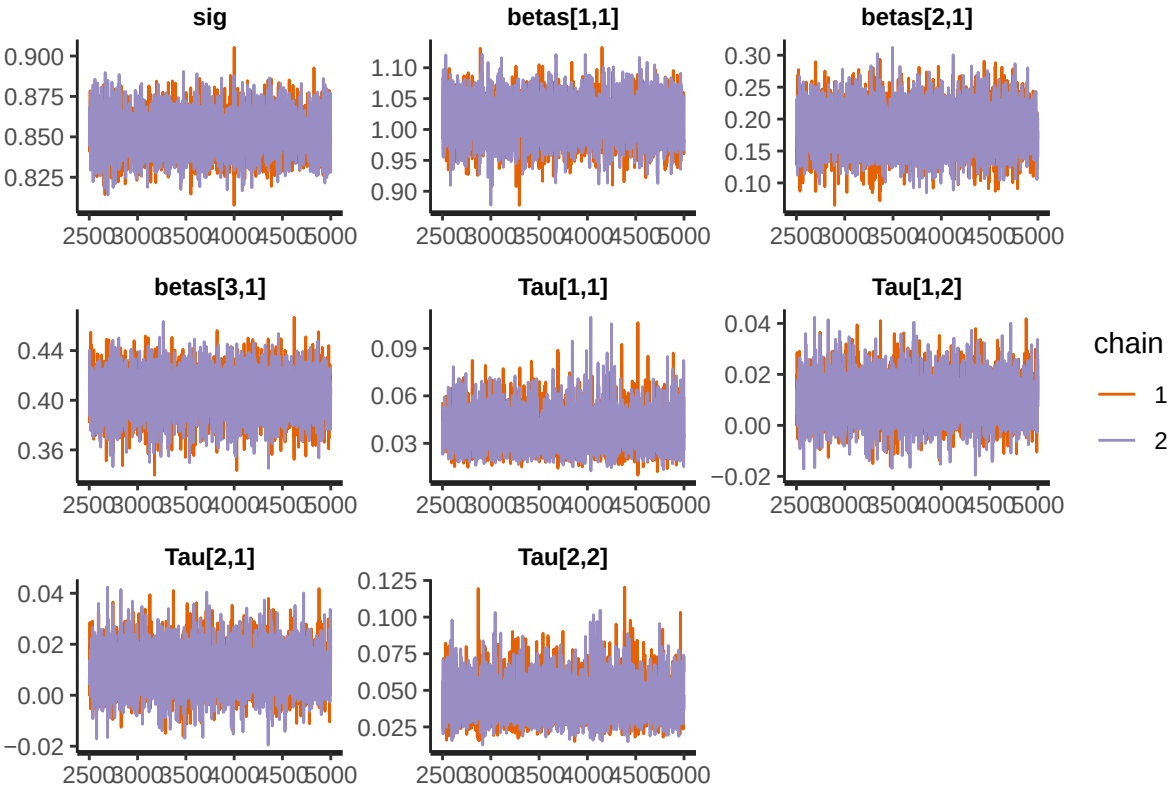
```
## [1] 1.000311
```

```
max(abs(summary(modE.custom, pars=pars)$summary[,"Rhat"]))
```

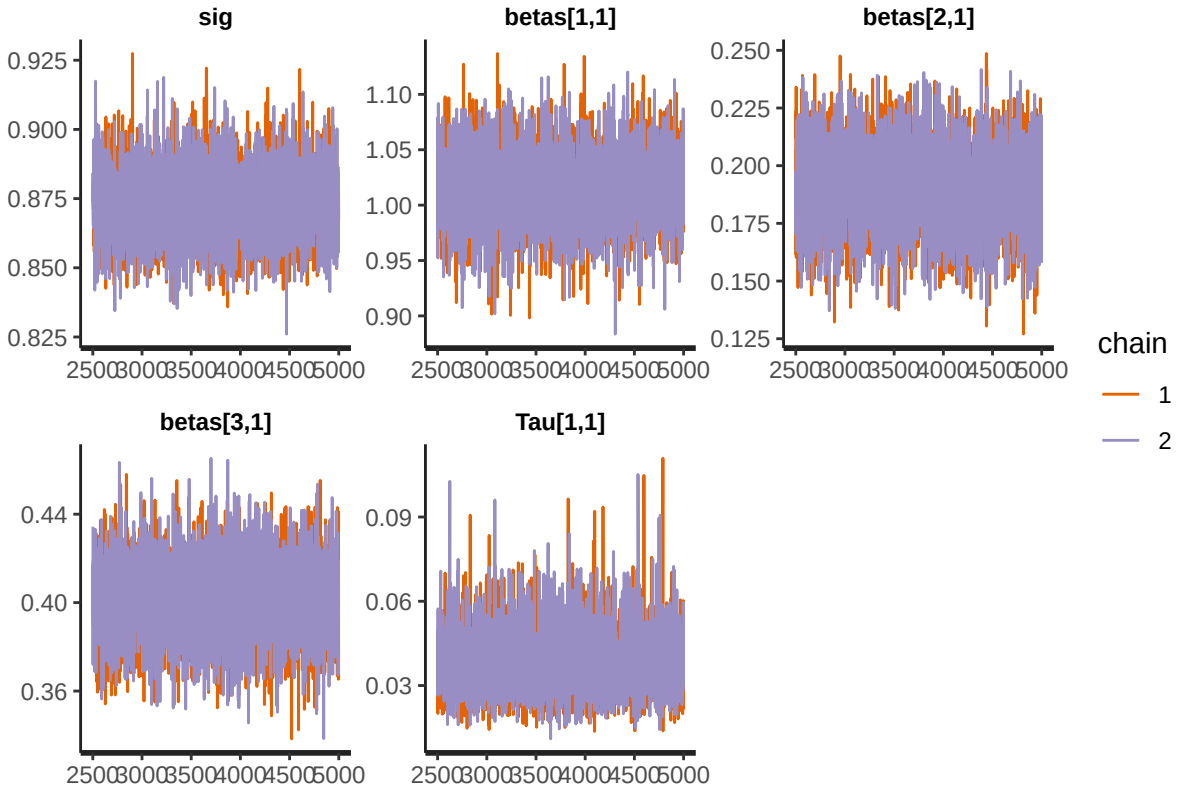
```
## [1] 0.9999194
```

Traceplots

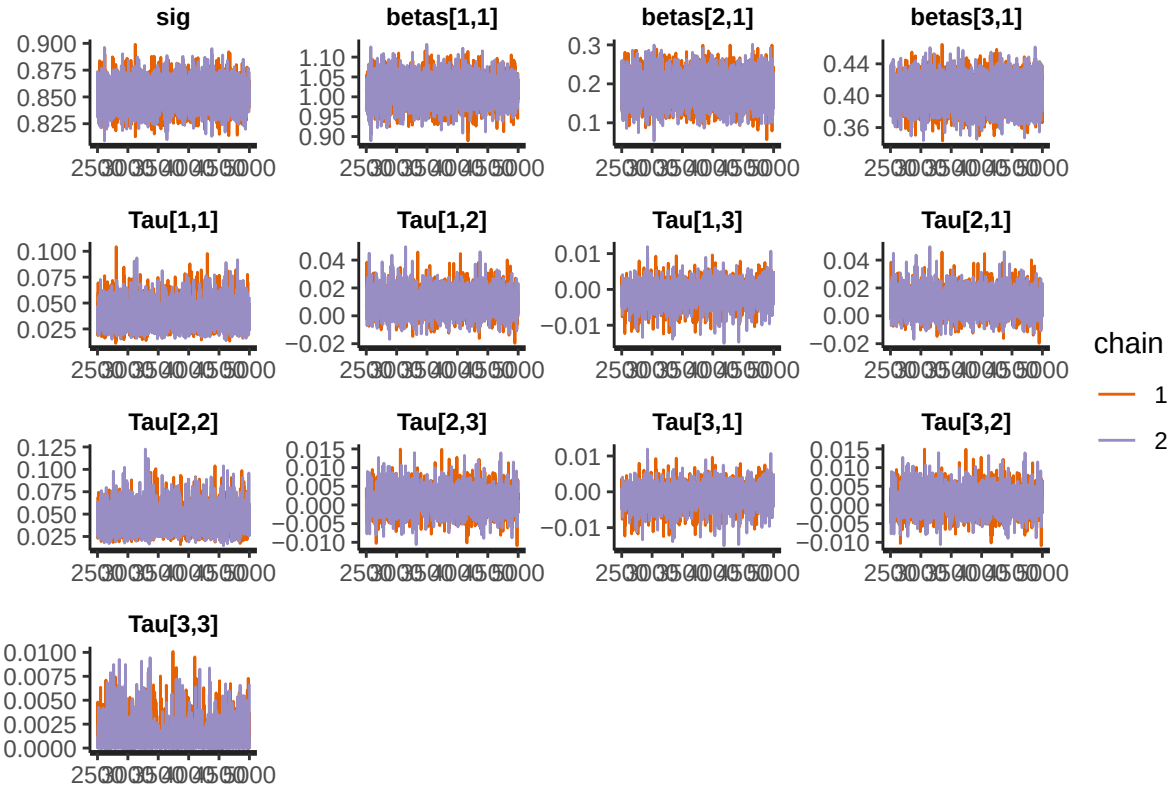
```
rstan::traceplot(modA.custom, pars = pars)
```



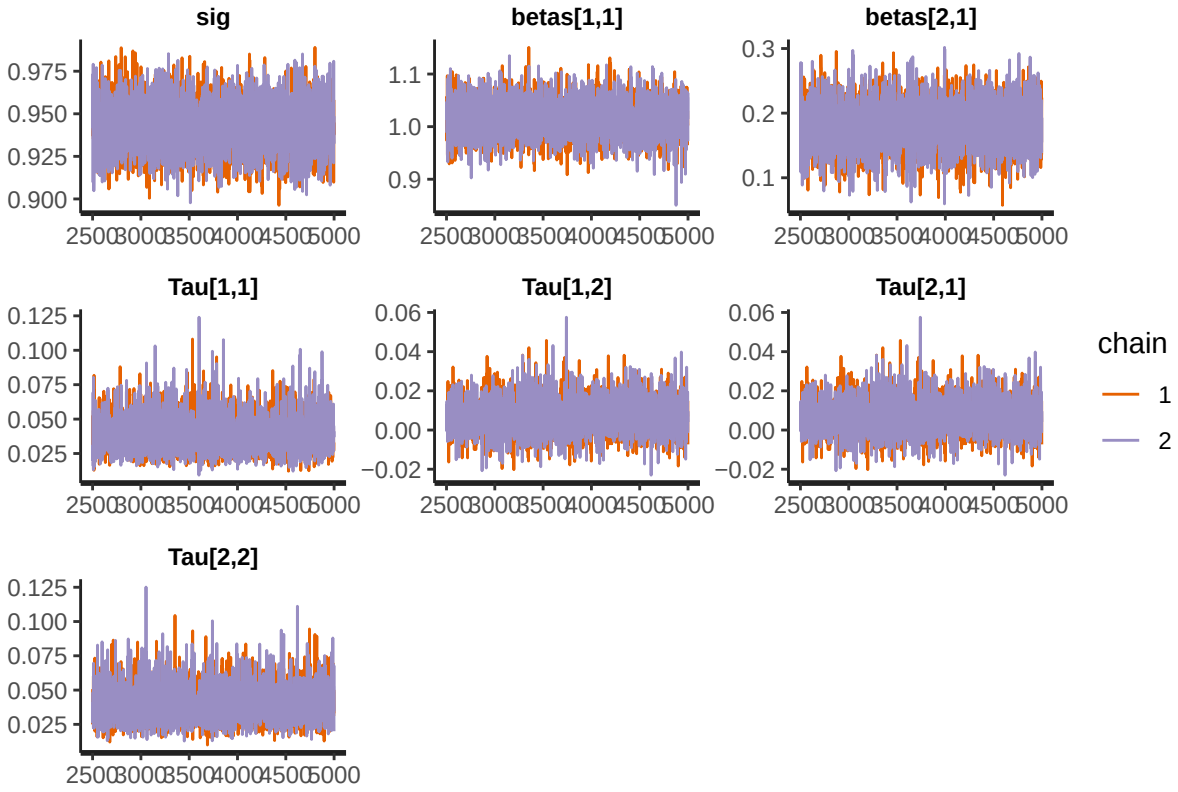

```
rstan::traceplot(modB.custom, pars = pars)
```



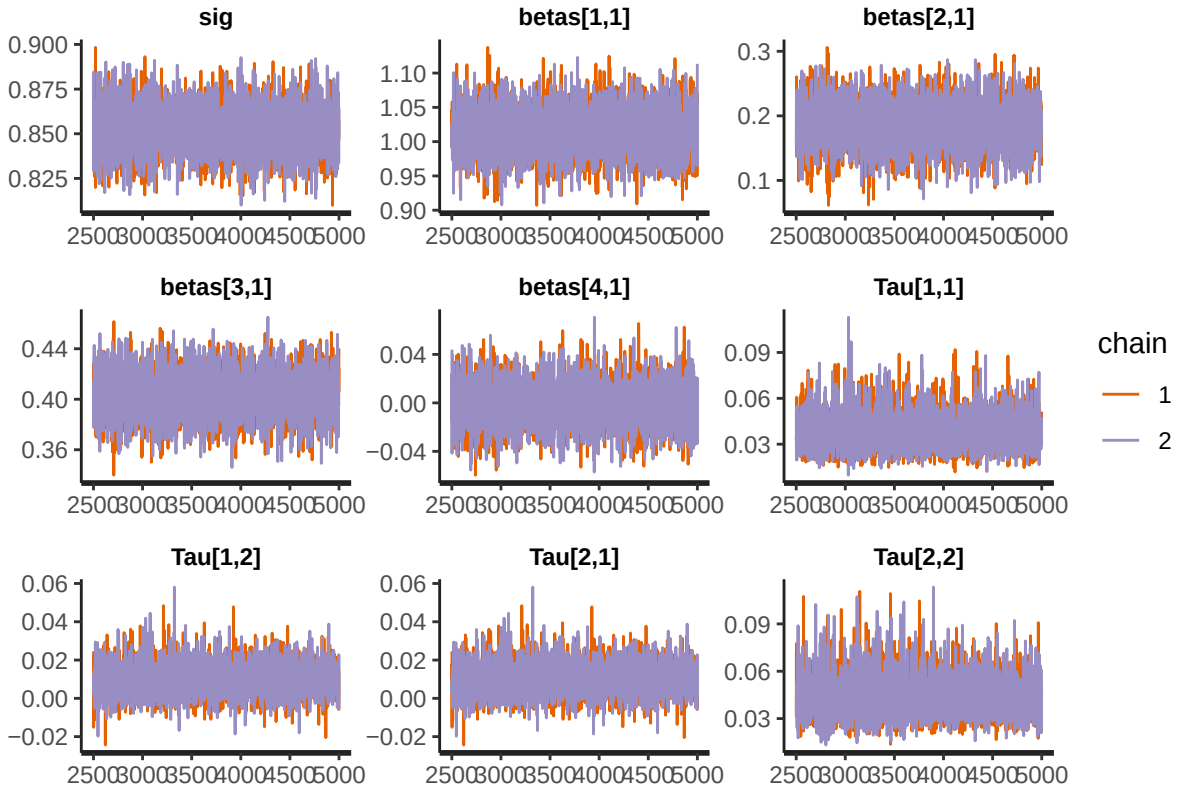

```
rstan::traceplot(modC.custom, pars = pars)
```




```
rstan::traceplot(modD.custom, pars = pars)
```




```
rstan::traceplot(modE.custom, pars = pars)
```



Custom Conditional Likelihood code

```
tic()
matsA <- mod.mat.mlm(modA, data=dat)
modA.custom.cond <- stan(file='mlmcond.stan', data = matsA,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 286.5 sec elapsed
```

```
tic()
matsB <- mod.mat.mlm(modB, data=dat)
modB.custom.cond <- stan(file='mlmcond.stan', data = matsB,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 149.75 sec elapsed
```

```
tic()
matsC <- mod.mat.mlm(modC, data=dat)
modC.custom.cond <- stan(file='mlmcond.stan', data = matsC,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 179.75 sec elapsed
```

```
tic()
matsD <- mod.mat.mlm(modD, data=dat)
modD.custom.cond <- stan(file='mlmcond.stan', data = matsD,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 159.87 sec elapsed
```

```
tic()
matsE <- mod.mat.mlm(modE, data=dat)
modE.custom.cond <- stan(file='mlmcond.stan', data = matsE,
                    chains = nchains, seed=5297,
                    iter = iter, refresh = 0, cores = ncores)
toc()
```

```
## 171.14 sec elapsed
```

Look at Diagnostics

Rhats

```
# error sd, coefficients, random effect covariance matrix
pars <- c("sig", "betas", "Tau")
max(abs(summary(modA.custom.cond, pars=pars)$summary[,"Rhat"]))
```

```
## [1] 1.002459
```

```
max(abs(summary(modB.custom.cond, pars=pars)$summary[,"Rhat"]))
```

```
## [1] 1.000347
```

```
max(abs(summary(modC.custom.cond, pars=pars)$summary[,"Rhat"]))
```

```
## [1] 1.002137
```

```
max(abs(summary(modD.custom.cond, pars=pars)$summary[,"Rhat"]))
```

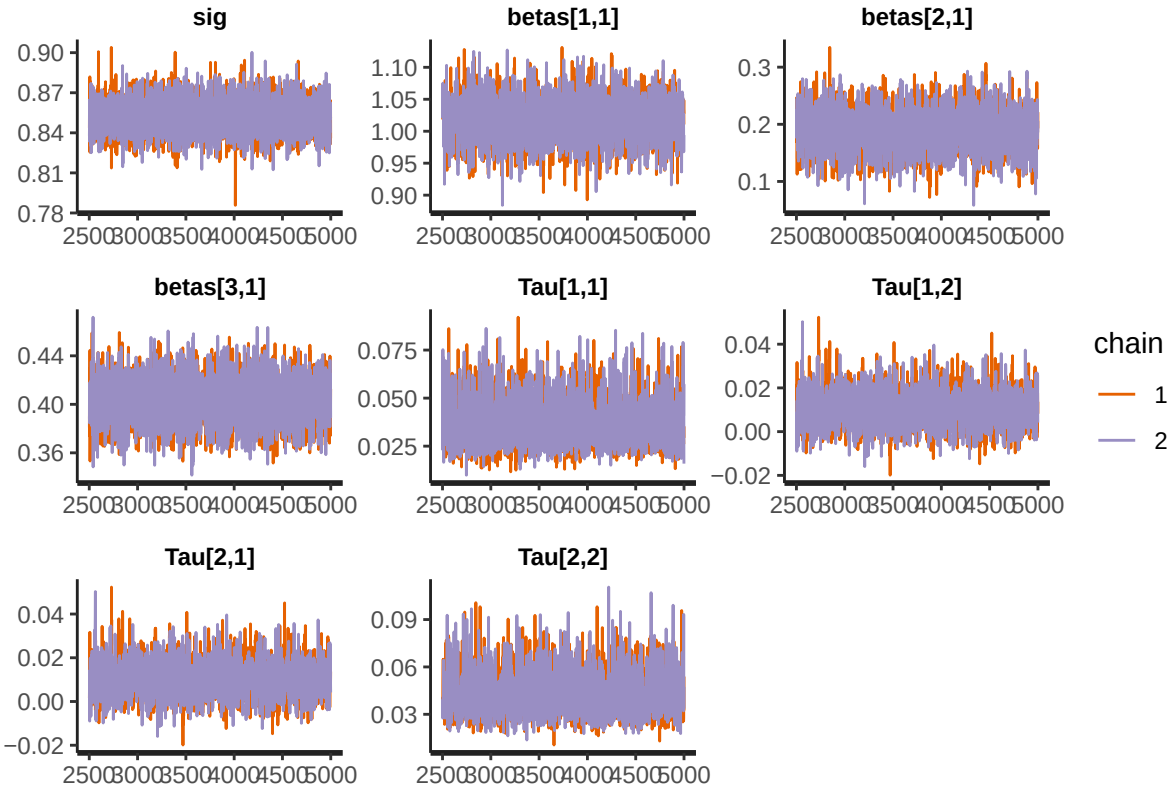
```
## [1] 1.00131
```

```
max(abs(summary(modE.custom.cond, pars=pars)$summary[,"Rhat"]))
```

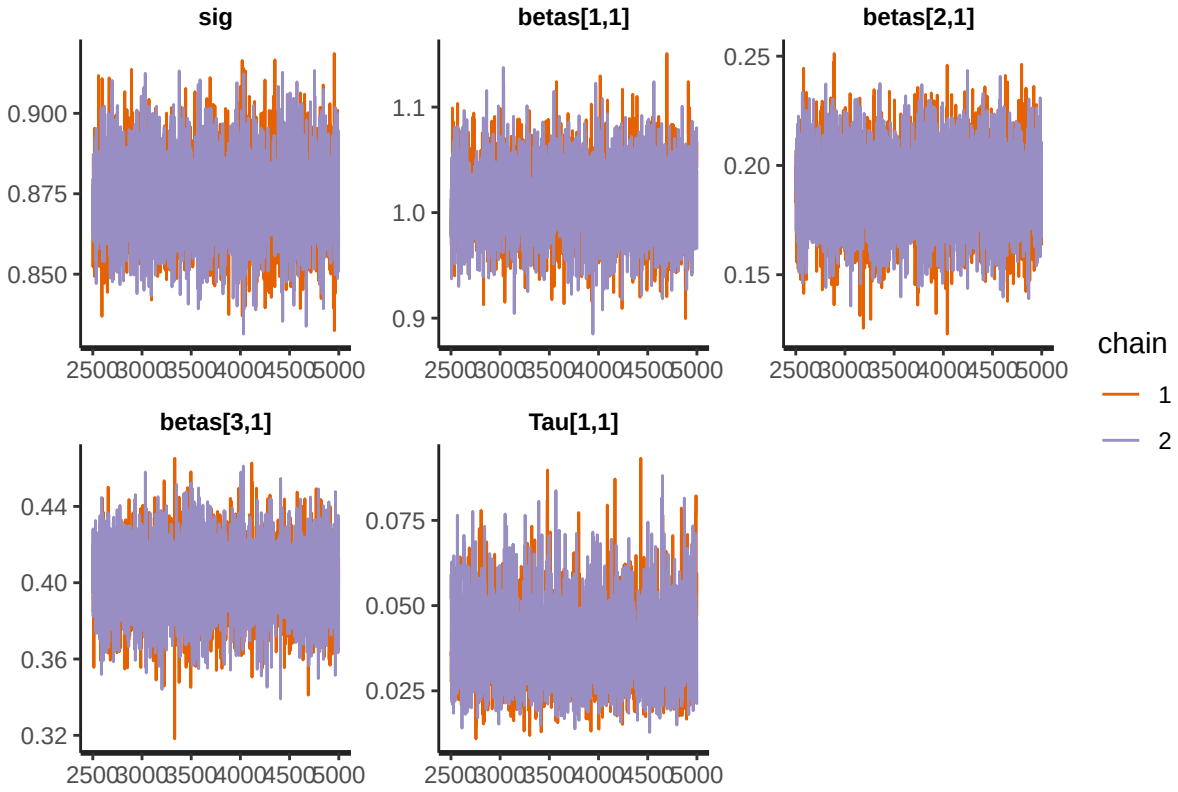
```
## [1] 1.000386
```

Traceplots

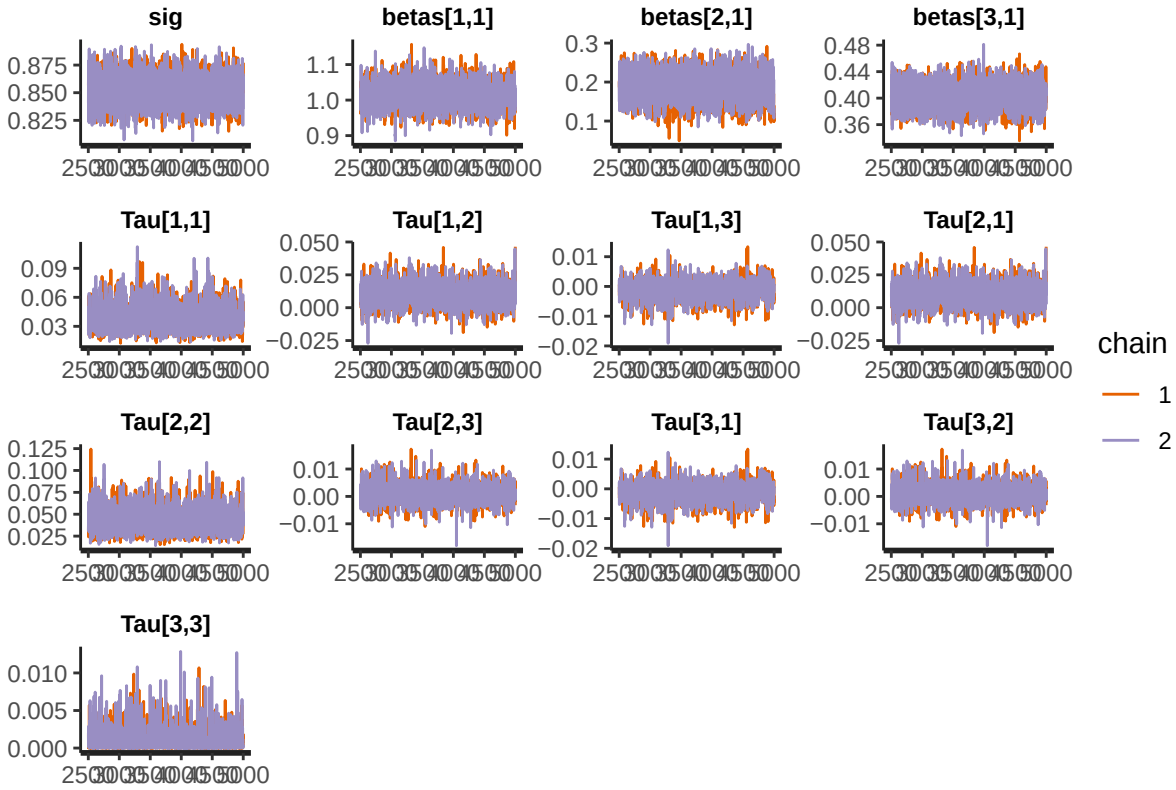
```
rstan::traceplot(modA.custom.cond, pars = pars)
```



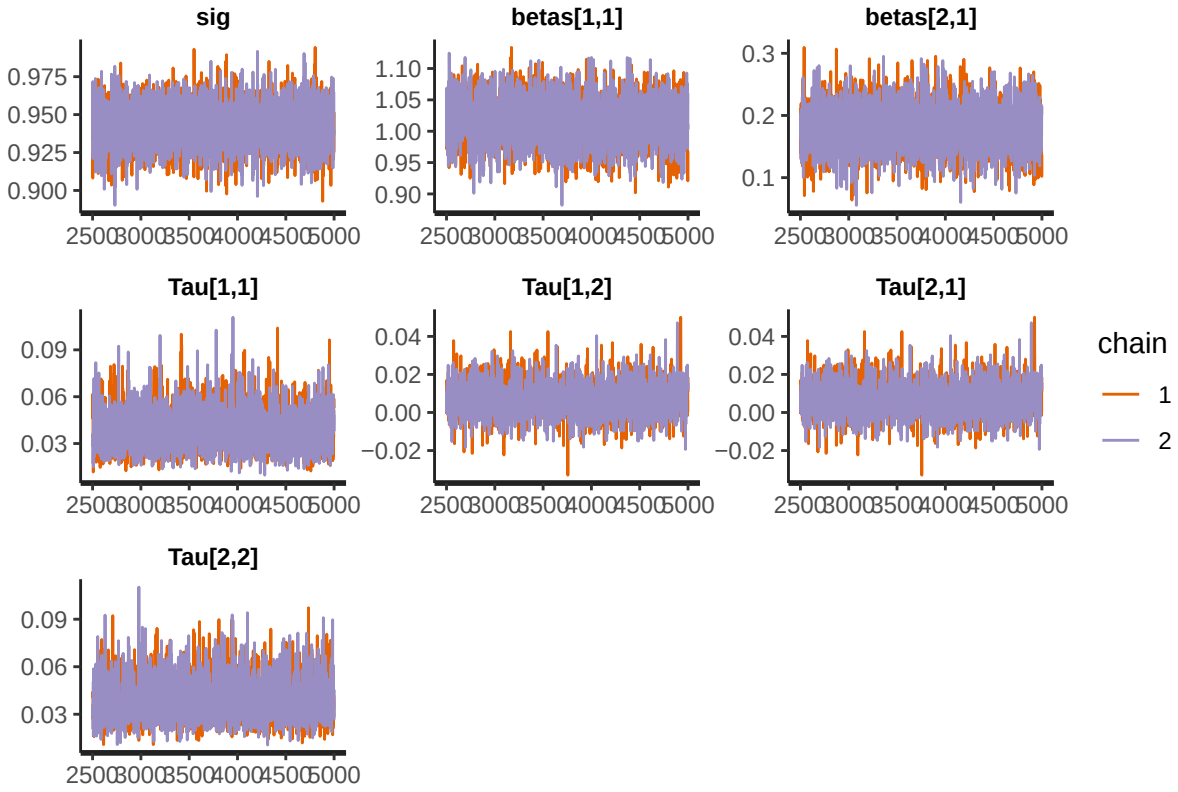

```
rstan::traceplot(modB.custom.cond, pars = pars)
```



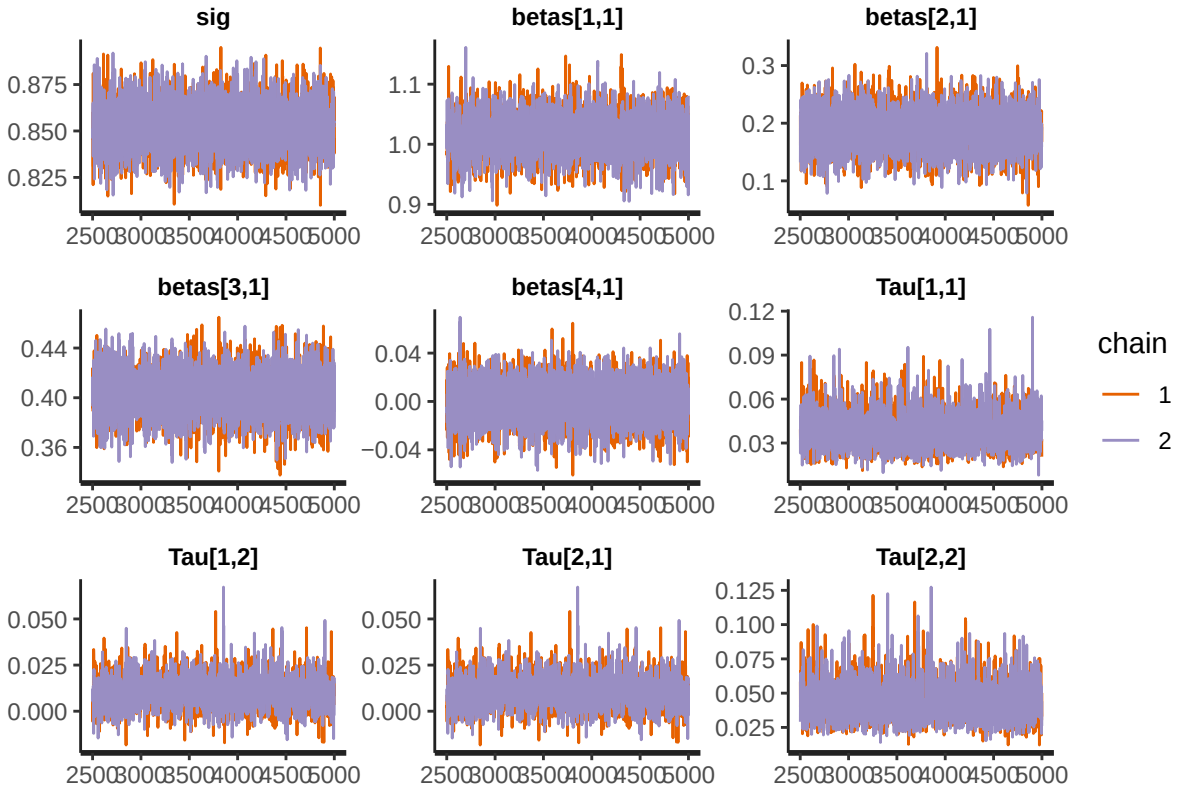

```
rstan::traceplot(modC.custom.cond, pars = pars)
```




```
rstan::traceplot(modD.custom.cond, pars = pars)
```




```
rstan::traceplot(modE.custom.cond, pars = pars)
```



Compare to `brms`

Parameter estimates

Some code could automate comparisons between parameters..

```
# Note Tau is not directly estimated, but sds and correlation matrix of random effects is
pars <- c("gssig", "gsmat", "betas","sig")

summary(modA.custom, pars = pars)$summary
```

```
##                 mean      se_mean           sd       2.5%       25%       50%
## gssig[1]   0.1918110 3.080043e-04 2.810884e-02  0.1418738 0.1721204 0.1900512
## gssig[2]   0.2049995 3.239594e-04 3.030804e-02  0.1513372 0.1834301 0.2033433
## gsmat[1,1] 1.0000000          NaN 0.000000e+00  1.0000000 1.0000000 1.0000000
## gsmat[1,2] 0.2484226 2.118150e-03 1.850621e-01 -0.1337348 0.1232743 0.2586771
## gsmat[2,1] 0.2484226 2.118150e-03 1.850621e-01 -0.1337348 0.1232743 0.2586771
## gsmat[2,2] 1.0000000 1.271269e-18 9.195014e-17  1.0000000 1.0000000 1.0000000
## betas[1,1] 1.0151835 3.529104e-04 3.154864e-02  0.9532490 0.9946503 1.0144379
## betas[2,1] 0.1870510 3.994190e-04 3.396616e-02  0.1195645 0.1645170 0.1869748
## betas[3,1] 0.4039262 1.910961e-04 1.706209e-02  0.3702468 0.3922990 0.4040235
## sig        0.8520069 1.518920e-04 1.209477e-02  0.8286582 0.8439104 0.8519799
##                 75%     97.5%    n_eff      Rhat
## gssig[1]   0.2097149 0.2537442 8328.608 1.0000714
## gssig[2]   0.2239143 0.2705220 8752.549 0.9996923
## gsmat[1,1] 1.0000000 1.0000000      NaN       NaN
```

```
## gsmat[1,2] 0.3802265 0.5919233 7633.466 0.9998811
## gsmat[2,1] 0.3802265 0.5919233 7633.466 0.9998811
## gsmat[2,2] 1.0000000 1.0000000 5231.548 0.9995999
## betas[1,1] 1.0359007 1.0780544 7991.574 0.9997140
## betas[2,1] 0.2102328 0.2535111 7231.616 0.9998718
## betas[3,1] 0.4157825 0.4371860 7971.881 0.9998050
## sig        0.8599656 0.8762982 6340.527 0.9999245
```

```
summary(modA.custom.cond, pars = pars)$summary
```

```
##                 mean      se_mean           sd       2.5%       25%       50%
## gssig[1]   0.1917614 5.708049e-04 2.783647e-02  0.1422070 0.1718994 0.1904965
## gssig[2]   0.2050996 6.471360e-04 2.977677e-02  0.1523955 0.1843445 0.2032047
## gsmat[1,1] 1.0000000          NaN 0.000000e+00  1.0000000 1.0000000 1.0000000
## gsmat[1,2] 0.2615447 5.032553e-03 1.826167e-01 -0.1154052 0.1387897 0.2692926
## gsmat[2,1] 0.2615447 5.032553e-03 1.826167e-01 -0.1154052 0.1387897 0.2692926
## gsmat[2,2] 1.0000000 1.351782e-18 9.104739e-17  1.0000000 1.0000000 1.0000000
## betas[1,1] 1.0159249 7.210667e-04 3.230296e-02  0.9541794 0.9943023 1.0152046
## betas[2,1] 0.1873588 7.601275e-04 3.409776e-02  0.1210169 0.1643743 0.1869247
## betas[3,1] 0.4036757 1.787176e-04 1.733937e-02  0.3679687 0.3922464 0.4036457
## sig        0.8522053 1.397209e-04 1.240695e-02  0.8283015 0.8439205 0.8519354
##                 75%     97.5%    n_eff      Rhat
## gssig[1]   0.2092188 0.2523644 2378.225 0.9997012
## gssig[2]   0.2241427 0.2688319 2117.211 1.0005606
## gsmat[1,1] 1.0000000 1.0000000      NaN       NaN
## gsmat[1,2] 0.3933291 0.5922740 1316.753 0.9998251
## gsmat[2,1] 0.3933291 0.5922740 1316.753 0.9998251
## gsmat[2,2] 1.0000000 1.0000000 4536.512 0.9995999
## betas[1,1] 1.0377332 1.0800986 2006.937 1.0024589
## betas[2,1] 0.2097332 0.2571362 2012.235 0.9996596
## betas[3,1] 0.4153987 0.4374996 9413.083 0.9997361
## sig        0.8603520 0.8765842 7885.099 0.9996684
```

```
summary(modA.brms)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ 1 + X1 + X2 + (X1 | g)
##    Data: dat (Number of observations: 2500)
##   Draws: 2 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 5000
##
## Multilevel Hyperparameters:
## ~g (Number of levels: 50)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)         0.19      0.03     0.14     0.25 1.00     2360     3586
## sd(X1)                0.20      0.03     0.15     0.27 1.00     2125     2771
## cor(Intercept,X1)     0.25      0.18    -0.11     0.58 1.00     1408     2617
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.01      0.03     0.95     1.08 1.00     2601     3390
## X1            0.19      0.03     0.12     0.25 1.00     2747     3230
## X2            0.40      0.02     0.37     0.44 1.00    11113     3931
##
```

```
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.85      0.01     0.83     0.88 1.00     9904     3174
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
summary(modB.custom, pars = pars)$summary
```

```
##                 mean      se_mean         sd      2.5%       25%       50%
## gssig[1]   0.1923681 0.0003690778 0.02786288 0.1422447 0.1735016 0.1907334
## gsmat[1,1] 1.0000000          NaN 0.00000000 1.0000000 1.0000000 1.0000000
## betas[1,1] 1.0130818 0.0004613252 0.03263846 0.9496920 0.9914550 1.0130321
## betas[2,1] 0.1893812 0.0002292444 0.01786855 0.1541005 0.1772779 0.1890396
## betas[3,1] 0.4010111 0.0002196821 0.01749074 0.3664365 0.3892013 0.4013143
## sig        0.8736130 0.0001618714 0.01258730 0.8492467 0.8652348 0.8734514
##                 75%     97.5%    n_eff      Rhat
## gssig[1]   0.2090517 0.2531057 5699.232 1.0002389
## gsmat[1,1] 1.0000000 1.0000000      NaN       NaN
## betas[1,1] 1.0344937 1.0786197 5005.471 1.0000738
## betas[2,1] 0.2014761 0.2239965 6075.491 0.9997895
## betas[3,1] 0.4124569 0.4347344 6339.090 0.9998775
## sig        0.8818100 0.8991470 6046.789 1.0004460
```

```
summary(modB.custom.cond, pars = pars)$summary
```

```
##                 mean      se_mean         sd      2.5%       25%       50%
## gssig[1]   0.1922894 0.0006734092 0.02791011 0.1425325 0.1729303 0.1905776
## gsmat[1,1] 1.0000000          NaN 0.00000000 1.0000000 1.0000000 1.0000000
## betas[1,1] 1.0121623 0.0006811161 0.03305573 0.9467564 0.9902695 1.0121083
## betas[2,1] 0.1894188 0.0002134748 0.01801588 0.1540873 0.1769402 0.1896585
## betas[3,1] 0.4008553 0.0002147105 0.01782444 0.3659216 0.3888636 0.4007410
## sig        0.8734694 0.0001510031 0.01244701 0.8500514 0.8649370 0.8732035
##                 75%     97.5%    n_eff      Rhat
## gssig[1]   0.2101924 0.2517036 1717.770 1.0000110
## gsmat[1,1] 1.0000000 1.0000000      NaN       NaN
## betas[1,1] 1.0341945 1.0764556 2355.327 0.9996959
## betas[2,1] 0.2019651 0.2240854 7122.264 1.0003473
## betas[3,1] 0.4126570 0.4363902 6891.678 0.9999336
## sig        0.8817166 0.8992709 6794.511 1.0001442
```

```
summary(modB.brms)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ 1 + X1 + X2 + (1 | g)
##    Data: dat (Number of observations: 2500)
##   Draws: 2 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 5000
##
## Multilevel Hyperparameters:
## ~g (Number of levels: 50)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.19      0.03     0.14     0.25 1.00     1862     2802
##
```

```
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.01      0.03     0.95     1.08 1.00     2026     2572
## X1            0.19      0.02     0.15     0.23 1.00     6979     3825
## X2            0.40      0.02     0.37     0.43 1.00     7243     3477
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.87      0.01     0.85     0.90 1.00     6606     3548
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
summary(modC.custom, pars = pars)$summary
```

```
##                   mean      se_mean           sd          2.5%          25%
## gssig[1]    0.19160290 3.313331e-04 2.897480e-02  0.1401200106  0.171458762
## gssig[2]    0.20565102 3.823812e-04 3.000967e-02  0.1515489798  0.184771660
## gssig[3]    0.02332682 2.615383e-04 1.765683e-02  0.0009513602  0.009441122
## gsmat[1,1]  1.00000000          NaN 0.000000e+00  1.0000000000  1.000000000
## gsmat[1,2]  0.23936578 2.267083e-03 1.807565e-01 -0.1299656368  0.115940309
## gsmat[1,3] -0.11973518 5.504373e-03 4.735594e-01 -0.9008676107 -0.504083418
## gsmat[2,1]  0.23936578 2.267083e-03 1.807565e-01 -0.1299656368  0.115940309
## gsmat[2,2]  1.00000000 1.301942e-18 9.161434e-17  1.0000000000  1.000000000
## gsmat[2,3]  0.09839165 5.842529e-03 4.712396e-01 -0.8168947751 -0.254230038
## gsmat[3,1] -0.11973518 5.504373e-03 4.735594e-01 -0.9008676107 -0.504083418
## gsmat[3,2]  0.09839165 5.842529e-03 4.712396e-01 -0.8168947751 -0.254230038
## gsmat[3,3]  1.00000000 2.074288e-18 8.544524e-17  1.0000000000  1.000000000
## betas[1,1]  1.01449142 4.051494e-04 3.231456e-02  0.9515671688  0.992450492
## betas[2,1]  0.18730244 4.294914e-04 3.493284e-02  0.1180532339  0.164203593
## betas[3,1]  0.40381828 2.105561e-04 1.738987e-02  0.3692833294  0.392414084
## sig         0.85194345 1.459069e-04 1.240023e-02  0.8278588918  0.843579485
##                   50%        75%      97.5%    n_eff      Rhat
## gssig[1]    0.18930240 0.21017484 0.25350858 7647.354 0.9998031
## gssig[2]    0.20359782 0.22406288 0.27189200 6159.271 0.9999291
## gssig[3]    0.01977891 0.03297286 0.06628364 4557.793 1.0000400
## gsmat[1,1]  1.00000000 1.00000000 1.00000000      NaN       NaN
## gsmat[1,2]  0.24480165 0.36771556 0.57529230 6357.017 0.9999002
## gsmat[1,3] -0.15818707 0.23515061 0.82700745 7401.730 1.0001531
## gsmat[2,1]  0.24480165 0.36771556 0.57529230 6357.017 0.9999002
## gsmat[2,2]  1.00000000 1.00000000 1.00000000 4951.575 0.9995999
## gsmat[2,3]  0.12676637 0.47321536 0.89376744 6505.516 1.0001128
## gsmat[3,1] -0.15818707 0.23515061 0.82700745 7401.730 1.0001531
## gsmat[3,2]  0.12676637 0.47321536 0.89376744 6505.516 1.0001128
## gsmat[3,3]  1.00000000 1.00000000 1.00000000 1696.828 0.9995999
## betas[1,1]  1.01456233 1.03652346 1.07865477 6361.597 0.9996710
## betas[2,1]  0.18756873 0.21026382 0.25797154 6615.440 0.9999144
## betas[3,1]  0.40387583 0.41533281 0.43792436 6821.142 1.0000721
## sig         0.85170172 0.86023466 0.87635550 7222.842 1.0000742
```

```
summary(modC.custom.cond, pars = pars)$summary
```

```
##                   mean      se_mean           sd         2.5%         25%
## gssig[1]    0.19214914 6.730741e-04 2.907744e-02  0.142298262  0.17167740
```

```
## gssig[2]    0.20445448 6.803969e-04 2.950764e-02  0.152877679  0.18347172
## gssig[3]    0.02367172 3.489747e-04 1.785130e-02  0.000837383  0.00943531
## gsmat[1,1]  1.00000000          NaN 0.000000e+00  1.000000000  1.00000000
## gsmat[1,2]  0.24195410 5.044851e-03 1.807986e-01 -0.131923742  0.11723144
## gsmat[1,3] -0.12034850 6.507067e-03 4.597356e-01 -0.878255710 -0.47990076
## gsmat[2,1]  0.24195410 5.044851e-03 1.807986e-01 -0.131923742  0.11723144
## gsmat[2,2]  1.00000000 1.269140e-18 9.108801e-17  1.000000000  1.00000000
## gsmat[2,3]  0.10299380 6.994985e-03 4.729334e-01 -0.828721219 -0.24591166
## gsmat[3,1] -0.12034850 6.507067e-03 4.597356e-01 -0.878255710 -0.47990076
## gsmat[3,2]  0.10299380 6.994985e-03 4.729334e-01 -0.828721219 -0.24591166
## gsmat[3,3]  1.00000000 8.972476e-19 6.065311e-17  1.000000000  1.00000000
## betas[1,1]  1.01433997 8.295802e-04 3.254999e-02  0.951528582  0.99280894
## betas[2,1]  0.18657261 8.449635e-04 3.392476e-02  0.119475336  0.16359012
## betas[3,1]  0.40392262 2.178400e-04 1.792986e-02  0.368567611  0.39140375
## sig         0.85210301 1.606109e-04 1.242186e-02  0.828115581  0.84370768
##                   50%        75%      97.5%    n_eff      Rhat
## gssig[1]    0.18950242 0.21051196 0.25497265 1866.322 1.0009214
## gssig[2]    0.20192920 0.22321936 0.26696559 1880.807 1.0008805
## gssig[3]    0.02027037 0.03382843 0.06503983 2616.687 1.0007433
## gsmat[1,1]  1.00000000 1.00000000 1.00000000      NaN       NaN
## gsmat[1,2]  0.25111754 0.37284176 0.56568546 1284.380 0.9999281
## gsmat[1,3] -0.16190907 0.20802457 0.79833055 4991.669 1.0008697
## gsmat[2,1]  0.25111754 0.37284176 0.56568546 1284.380 0.9999281
## gsmat[2,2]  1.00000000 1.00000000 1.00000000 5151.138 0.9995999
## gsmat[2,3]  0.13935178 0.47998879 0.88808065 4571.159 0.9997500
## gsmat[3,1] -0.16190907 0.20802457 0.79833055 4991.669 1.0008697
## gsmat[3,2]  0.13935178 0.47998879 0.88808065 4571.159 0.9997500
## gsmat[3,3]  1.00000000 1.00000000 1.00000000 4569.635 0.9995999
## betas[1,1]  1.01406604 1.03565631 1.07829590 1539.519 1.0006541
## betas[2,1]  0.18631058 0.20921032 0.25298182 1611.972 1.0021369
## betas[3,1]  0.40400965 0.41583230 0.43813461 6774.525 0.9997167
## sig         0.85187168 0.86006417 0.87721235 5981.682 0.9999569
```

```
summary(modC.brms)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ 1 + X1 + X2 + (X1 + X2 | g)
##    Data: dat (Number of observations: 2500)
##   Draws: 2 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 5000
##
## Multilevel Hyperparameters:
## ~g (Number of levels: 50)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)         0.19      0.03     0.14     0.25 1.00     2148     3192
## sd(X1)                0.21      0.03     0.15     0.27 1.00     2040     2963
## sd(X2)                0.02      0.02     0.00     0.07 1.00     2628     2166
## cor(Intercept,X1)     0.24      0.18    -0.15     0.57 1.00     1152     2248
## cor(Intercept,X2)    -0.12      0.47    -0.90     0.79 1.00     6834     3428
## cor(X1,X2)            0.11      0.47    -0.80     0.89 1.00     6455     3919
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.01      0.03     0.95     1.08 1.00     1882     2908
```

```
## X1            0.19      0.03     0.12     0.26 1.00     1895     2140
## X2            0.40      0.02     0.37     0.44 1.00     7569     3878
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.85      0.01     0.83     0.88 1.00     7281     3068
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
summary(modD.custom, pars = pars)$summary
```

```
##                 mean      se_mean           sd       2.5%        25%       50%
## gssig[1]   0.1935204 4.130046e-04 3.090245e-02  0.1392984 0.17193187 0.1916898
## gssig[2]   0.1958051 3.898703e-04 3.075364e-02  0.1418536 0.17419381 0.1937933
## gsmat[1,1] 1.0000000          NaN 0.000000e+00  1.0000000 1.00000000 1.0000000
## gsmat[1,2] 0.1809829 2.630157e-03 1.985634e-01 -0.2254724 0.04764889 0.1875613
## gsmat[2,1] 0.1809829 2.630157e-03 1.985634e-01 -0.2254724 0.04764889 0.1875613
## gsmat[2,2] 1.0000000 1.313643e-18 9.123677e-17  1.0000000 1.00000000 1.0000000
## betas[1,1] 1.0160345 4.410142e-04 3.303103e-02  0.9523405 0.99330338 1.0160469
## betas[2,1] 0.1801277 4.616105e-04 3.381872e-02  0.1137230 0.15815957 0.1800260
## sig        0.9424056 1.582922e-04 1.371087e-02  0.9157070 0.93301131 0.9421362
##                 75%     97.5%    n_eff      Rhat
## gssig[1]   0.2126310 0.2605958 5598.556 0.9998961
## gssig[2]   0.2147029 0.2619235 6222.326 1.0002845
## gsmat[1,1] 1.0000000 1.0000000      NaN       NaN
## gsmat[1,2] 0.3197420 0.5525350 5699.475 0.9997332
## gsmat[2,1] 0.3197420 0.5525350 5699.475 0.9997332
## gsmat[2,2] 1.0000000 1.0000000 4823.755 0.9995999
## betas[1,1] 1.0381456 1.0807423 5609.692 0.9997793
## betas[2,1] 0.2023987 0.2469757 5367.389 1.0002954
## sig        0.9518284 0.9694847 7502.583 1.0000681
```

```
summary(modD.custom.cond, pars = pars)$summary
```

```
##                 mean      se_mean           sd       2.5%        25%       50%
## gssig[1]   0.1927912 6.183227e-04 3.006365e-02  0.1385658 0.17205747 0.1910379
## gssig[2]   0.1959378 6.386475e-04 3.067985e-02  0.1403043 0.17471799 0.1944403
## gsmat[1,1] 1.0000000          NaN 0.000000e+00  1.0000000 1.00000000 1.0000000
## gsmat[1,2] 0.1929143 4.892967e-03 1.986138e-01 -0.2147432 0.05866578 0.1997407
## gsmat[2,1] 0.1929143 4.892967e-03 1.986138e-01 -0.2147432 0.05866578 0.1997407
## gsmat[2,2] 1.0000000 1.283064e-18 9.073542e-17  1.0000000 1.00000000 1.0000000
## betas[1,1] 1.0155747 6.859032e-04 3.340000e-02  0.9483559 0.99357811 1.0151354
## betas[2,1] 0.1809350 5.776368e-04 3.400896e-02  0.1147238 0.15816750 0.1811421
## sig        0.9421652 1.459965e-04 1.393512e-02  0.9157080 0.93225269 0.9422138
##                 75%     97.5%    n_eff      Rhat
## gssig[1]   0.2122468 0.2557692 2364.030 1.0003723
## gssig[2]   0.2155238 0.2606005 2307.725 1.0014114
## gsmat[1,1] 1.0000000 1.0000000      NaN       NaN
## gsmat[1,2] 0.3358380 0.5584623 1647.685 1.0001474
## gsmat[2,1] 0.3358380 0.5584623 1647.685 1.0001474
## gsmat[2,2] 1.0000000 1.0000000 5001.006 0.9995999
## betas[1,1] 1.0379409 1.0813420 2371.196 1.0011947
## betas[2,1] 0.2030142 0.2480804 3466.387 1.0004660
```

```
## sig        0.9515949 0.9691680 9110.387 0.9997178
```

```
summary(modD.brms)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ 1 + X1 + (X1 | g)
##    Data: dat (Number of observations: 2500)
##   Draws: 2 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 5000
##
## Multilevel Hyperparameters:
## ~g (Number of levels: 50)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)         0.19      0.03     0.14     0.26 1.00     2371     3063
## sd(X1)                0.20      0.03     0.14     0.26 1.00     2208     3490
## cor(Intercept,X1)     0.18      0.20    -0.23     0.55 1.00     1717     2596
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.02      0.03     0.95     1.08 1.00     3403     3809
## X1            0.18      0.03     0.11     0.25 1.00     3560     3670
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.94      0.01     0.92     0.97 1.00     9833     3647
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
summary(modE.custom, pars = pars)$summary
```

```
##                    mean      se_mean           sd        2.5%         25%
## gssig[1]    0.191584225 3.134064e-04 2.817121e-02  0.14165413  0.17182018
## gssig[2]    0.205219533 3.129574e-04 2.940422e-02  0.15388628  0.18452991
## gsmat[1,1]  1.000000000          NaN 0.000000e+00  1.00000000  1.00000000
## gsmat[1,2]  0.247339082 2.018841e-03 1.812911e-01 -0.12111311  0.12388931
## gsmat[2,1]  0.247339082 2.018841e-03 1.812911e-01 -0.12111311  0.12388931
## gsmat[2,2]  1.000000000 1.257807e-18 9.165470e-17  1.00000000  1.00000000
## betas[1,1]  1.014983726 3.711169e-04 3.258476e-02  0.95299170  0.99325273
## betas[2,1]  0.186362834 3.857017e-04 3.374497e-02  0.11819030  0.16419798
## betas[3,1]  0.403853970 1.803638e-04 1.727701e-02  0.36920740  0.39259458
## betas[4,1] -0.001029122 1.916765e-04 1.729309e-02 -0.03535518 -0.01296458
## sig         0.852348714 1.251941e-04 1.239611e-02  0.82807000  0.84424498
##                     50%        75%      97.5%    n_eff      Rhat
## gssig[1]    0.1898274520 0.20896368 0.25308192 8079.697 0.9997889
## gssig[2]    0.2030220708 0.22351691 0.26848477 8827.727 0.9996936
## gsmat[1,1]  1.0000000000 1.00000000 1.00000000      NaN       NaN
## gsmat[1,2]  0.2555585836 0.37636408 0.58139844 8063.973 0.9997594
## gsmat[2,1]  0.2555585836 0.37636408 0.58139844 8063.973 0.9997594
## gsmat[2,2]  1.0000000000 1.00000000 1.00000000 5309.840 0.9995999
## betas[1,1]  1.0145571484 1.03720499 1.08009665 7709.169 0.9996685
## betas[2,1]  0.1866371130 0.20899630 0.25254896 7654.467 0.9999194
## betas[3,1]  0.4038066291 0.41541954 0.43780493 9175.678 0.9996951
```

```
## betas[4,1] -0.0009773671 0.01064513 0.03306578 8139.682 0.9999137
## sig         0.8522041555 0.86052695 0.87693732 9803.996 0.9996905
```

```
summary(modE.custom.cond, pars = pars)$summary
```

```
##                     mean      se_mean           sd        2.5%         25%
## gssig[1]    0.1922497909 5.878509e-04 2.853238e-02  0.14087655  0.17260126
## gssig[2]    0.2054511746 6.728583e-04 3.006053e-02  0.15307039  0.18444785
## gsmat[1,1]  1.0000000000          NaN 0.000000e+00  1.00000000  1.00000000
## gsmat[1,2]  0.2520563312 4.591635e-03 1.860165e-01 -0.13136699  0.12775814
## gsmat[2,1]  0.2520563312 4.591635e-03 1.860165e-01 -0.13136699  0.12775814
## gsmat[2,2]  1.0000000000 1.381783e-18 9.104739e-17  1.00000000  1.00000000
## betas[1,1]  1.0155099452 6.790466e-04 3.306841e-02  0.95057689  0.99320109
## betas[2,1]  0.1882400793 6.519954e-04 3.399363e-02  0.12263950  0.16623551
## betas[3,1]  0.4034706732 1.692261e-04 1.740901e-02  0.36905682  0.39228388
## betas[4,1] -0.0007688253 1.689383e-04 1.691220e-02 -0.03453865 -0.01223937
## sig         0.8522343439 1.209930e-04 1.221243e-02  0.82933382  0.84381928
##                     50%        75%      97.5%     n_eff      Rhat
## gssig[1]    0.1901421430 0.20988566 0.25326256  2355.817 1.0001158
## gssig[2]    0.2031737811 0.22409647 0.26995233  1995.933 1.0004933
## gsmat[1,1]  1.0000000000 1.00000000 1.00000000       NaN       NaN
## gsmat[1,2]  0.2617509046 0.38430188 0.58476762  1641.226 0.9997707
## gsmat[2,1]  0.2617509046 0.38430188 0.58476762  1641.226 0.9997707
## gsmat[2,2]  1.0000000000 1.00000000 1.00000000  4341.656 0.9995999
## betas[1,1]  1.0153087408 1.03722314 1.07973751  2371.523 1.0002524
## betas[2,1]  0.1874117749 0.21065831 0.25522049  2718.354 1.0001173
## betas[3,1]  0.4031768067 0.41533354 0.43701037 10583.120 0.9996807
## betas[4,1] -0.0007379099 0.01080002 0.03116623 10021.763 0.9996639
## sig         0.8520701542 0.86061600 0.87659987 10187.880 0.9996518
```

```
summary(modE.brms)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ 1 + X1 + X2 + X3 + (X1 | g)
##    Data: dat (Number of observations: 2500)
##   Draws: 2 chains, each with iter = 5000; warmup = 2500; thin = 1;
##          total post-warmup draws = 5000
##
## Multilevel Hyperparameters:
## ~g (Number of levels: 50)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)         0.19      0.03     0.14     0.25 1.00     2415     3667
## sd(X1)                0.21      0.03     0.15     0.27 1.00     2308     3385
## cor(Intercept,X1)     0.25      0.19    -0.13     0.59 1.00     1397     2664
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.01      0.03     0.95     1.08 1.00     2206     3119
## X1            0.19      0.03     0.12     0.26 1.00     2476     3184
## X2            0.40      0.02     0.37     0.44 1.00    11530     3367
## X3           -0.00      0.02    -0.04     0.03 1.00    11151     3111
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
```

```
## sigma     0.85      0.01     0.83     0.88 1.00    10483     3810
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Fit stats

```
loo(modA.custom)
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3212.0 39.1
## p_loo         6.3  0.8
## looic      6424.1 78.2
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume MCMC draws (r_eff in [1.0, 1.7]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modA.custom.cond)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3185.0 36.5
## p_loo        73.2  2.8
## looic      6370.0 73.0
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.4, 2.4]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modA.brms)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3184.7 36.5
## p_loo        72.9  2.8
## looic      6369.3 73.0
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.4, 2.6]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modB.custom)
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3242.3 40.4
## p_loo         7.1  0.9
## looic      6484.7 80.8
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.9, 1.2]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modB.custom.cond)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3228.6 37.2
## p_loo        39.2  1.2
## looic      6457.2 74.3
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.6, 2.0]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modB.brms)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3228.2 37.1
## p_loo        38.9  1.2
## looic      6456.4 74.3
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.6, 1.8]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modC.custom)
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3212.8 39.0
## p_loo         6.8  0.9
```

```
## looic      6425.5 78.1
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.9, 1.4]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modC.custom.cond)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3186.0 36.5
## p_loo        75.6  2.9
## looic      6372.0 73.0
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.4, 2.1]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modC.brms)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3186.2 36.5
## p_loo        75.9  2.9
## looic      6372.4 72.9
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.4, 2.3]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modD.custom)
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3454.9 34.9
## p_loo         5.4  0.7
## looic      6909.8 69.8
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.9, 1.4]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modD.custom.cond)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3433.1 35.8
## p_loo        68.0  2.5
## looic      6866.2 71.6
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.4, 2.5]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modD.brms)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3433.0 35.8
## p_loo        68.1  2.5
## looic      6866.0 71.5
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.3, 2.4]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modE.custom)
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3213.0 39.1
## p_loo         7.2  0.8
## looic      6426.0 78.1
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume MCMC draws (r_eff in [1.1, 1.8]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modE.custom.cond)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3185.2 36.5
## p_loo        73.4  2.8
```

```
## looic      6370.3 72.9
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.4, 2.4]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(modE.brms)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo  -3185.6 36.5
## p_loo        73.6  2.8
## looic      6371.1 72.9
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.4, 2.5]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
waic(extract_log_lik(modA.custom))
```

```
## Warning:
## 1 (2.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3212.0 39.1
## p_waic         6.3  0.8
## waic        6424.0 78.2
##
## 1 (2.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(extract_log_lik(modA.custom.cond))
```

```
## Warning:
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3184.7 36.5
## p_waic        72.9  2.8
## waic        6369.5 72.9
##
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(modA.brms)
```

```
## Warning:
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3184.4 36.5
## p_waic        72.6  2.8
## waic        6368.8 72.9
##
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(extract_log_lik(modB.custom))
```

```
## Warning:
## 2 (4.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3242.3 40.4
## p_waic         7.1  0.9
## waic        6484.7 80.8
##
## 2 (4.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(extract_log_lik(modB.custom.cond))
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3228.5 37.2
## p_waic        39.2  1.2
## waic        6457.1 74.3
```

```
waic(modB.brms)
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3228.1 37.1
## p_waic        38.8  1.2
## waic        6456.3 74.3
```

```
waic(extract_log_lik(modC.custom))
```

```
## Warning:
## 1 (2.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3212.7 39.0
## p_waic         6.8  0.9
## waic        6425.5 78.1
##
```

```
## 1 (2.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(extract_log_lik(modC.custom.cond))
```

```
## Warning:
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3185.7 36.5
## p_waic        75.4  2.8
## waic        6371.4 72.9
##
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(modC.brms)
```

```
## Warning:
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3185.9 36.5
## p_waic        75.6  2.9
## waic        6371.8 72.9
##
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(extract_log_lik(modD.custom))
```

```
## Warning:
## 2 (4.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3454.9 34.9
## p_waic         5.3  0.7
## waic        6909.7 69.8
##
## 2 (4.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(extract_log_lik(modD.custom.cond))
```

```
## Warning:
## 5 (0.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3432.9 35.8
## p_waic        67.7  2.5
## waic        6865.8 71.5
```

```
##
## 5 (0.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(modD.brms)
```

```
## Warning:
## 5 (0.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3432.8 35.8
## p_waic        67.9  2.5
## waic        6865.6 71.5
##
## 5 (0.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(extract_log_lik(modE.custom))
```

```
## Warning:
## 1 (2.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 50 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3212.9 39.1
## p_waic         7.1  0.8
## waic        6425.9 78.1
##
## 1 (2.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(extract_log_lik(modE.custom.cond))
```

```
## Warning:
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3184.9 36.5
## p_waic        73.1  2.7
## waic        6369.8 72.9
##
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(modE.brms)
```

```
## Warning:
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 5000 by 2500 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -3185.3 36.4
## p_waic        73.4  2.8
```

```
## waic        6370.6 72.9
##
## 7 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

Stuff

```
devtools::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.4.0 (2024-04-24 ucrt)
##  os       Windows 10 x64 (build 19045)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_Canada.utf8
##  ctype    English_Canada.utf8
##  tz       America/Toronto
##  date     2025-08-29
##  pandoc   3.4 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown)
##
## - Packages -------------------------------------------------------------------
##  ! package        * version    date (UTC) lib source
##    abind            1.4-5      2016-07-21 [1] CRAN (R 4.4.0)
##    backports        1.5.0      2024-05-23 [1] CRAN (R 4.4.0)
##    bayesplot        1.11.1     2024-02-15 [1] CRAN (R 4.4.0)
##    boot             1.3-30     2024-02-26 [2] CRAN (R 4.4.0)
##    bridgesampling   1.1-2      2021-04-16 [1] CRAN (R 4.4.0)
##    brms           * 2.21.0     2024-03-20 [1] CRAN (R 4.4.0)
##    Brobdingnag      1.2-9      2022-10-19 [1] CRAN (R 4.4.0)
##    cachem           1.1.0      2024-05-16 [1] CRAN (R 4.4.0)
##    checkmate        2.3.1      2023-12-04 [1] CRAN (R 4.4.0)
##    cli              3.6.2      2023-12-11 [1] CRAN (R 4.4.0)
##    coda           * 0.19-4.1   2024-01-31 [1] CRAN (R 4.4.0)
##    codetools        0.2-20     2024-03-31 [2] CRAN (R 4.4.0)
##    colorspace       2.1-0      2023-01-23 [1] CRAN (R 4.4.0)
##    curl             5.2.1      2024-03-01 [1] CRAN (R 4.4.0)
##    devtools         2.4.5      2022-10-11 [1] CRAN (R 4.4.0)
##    digest           0.6.35     2024-03-11 [1] CRAN (R 4.4.0)
##    distributional   0.4.0      2024-02-07 [1] CRAN (R 4.4.0)
##    dplyr          * 1.1.4      2023-11-17 [1] CRAN (R 4.4.0)
##    ellipsis         0.3.2      2021-04-29 [1] CRAN (R 4.4.0)
##    emmeans          1.10.7     2025-01-31 [1] CRAN (R 4.4.2)
##    estimability     1.5.1      2024-05-12 [1] CRAN (R 4.4.2)
##    evaluate         1.0.1      2024-10-10 [1] CRAN (R 4.4.0)
##    fansi            1.0.6      2023-12-08 [1] CRAN (R 4.4.0)
##    farver           2.1.2      2024-05-13 [1] CRAN (R 4.4.0)
##    fastmap          1.2.0      2024-05-15 [1] CRAN (R 4.4.0)
##    fs               1.6.4      2024-04-25 [1] CRAN (R 4.4.0)
##    generics         0.1.3      2022-07-05 [1] CRAN (R 4.4.2)
##    ggplot2          3.5.2      2025-04-09 [1] CRAN (R 4.4.3)
##    glue             1.7.0      2024-01-09 [1] CRAN (R 4.4.0)
##    gridExtra        2.3        2017-09-09 [1] CRAN (R 4.4.0)
```

```
##    gtable           0.3.5      2024-04-22 [1] CRAN (R 4.4.0)
##    highr            0.11       2024-05-26 [1] CRAN (R 4.4.0)
##    htmltools        0.5.8.1    2024-04-04 [1] CRAN (R 4.4.0)
##    htmlwidgets      1.6.4      2023-12-06 [1] CRAN (R 4.4.0)
##    httpuv           1.6.15     2024-03-26 [1] CRAN (R 4.4.0)
##    inline           0.3.19     2021-05-31 [1] CRAN (R 4.4.0)
##    jsonlite         1.8.8      2023-12-04 [1] CRAN (R 4.4.0)
##    knitr            1.48       2024-07-07 [1] CRAN (R 4.4.0)
##    labeling         0.4.3      2023-08-29 [1] CRAN (R 4.4.0)
##    later            1.3.2      2023-12-06 [1] CRAN (R 4.4.0)
##    lattice          0.22-6     2024-03-20 [2] CRAN (R 4.4.0)
##    lifecycle        1.0.4      2023-11-07 [1] CRAN (R 4.4.0)
##    lme4           * 1.1-35.3   2024-04-16 [1] CRAN (R 4.4.0)
##    loo            * 2.7.0      2024-02-24 [1] CRAN (R 4.4.0)
##    magrittr         2.0.3      2022-03-30 [1] CRAN (R 4.4.0)
##    MASS           * 7.3-64     2025-01-04 [1] CRAN (R 4.4.2)
##    Matrix         * 1.7-0      2024-03-22 [2] CRAN (R 4.4.0)
##    matrixcalc     * 1.0-6      2022-09-14 [1] CRAN (R 4.4.0)
##    MatrixModels     0.5-3      2023-11-06 [1] CRAN (R 4.4.0)
##    matrixStats      1.3.0      2024-04-11 [1] CRAN (R 4.4.0)
##    mcmc             0.9-8      2023-11-16 [1] CRAN (R 4.4.0)
##    MCMCpack       * 1.7-0      2024-01-18 [1] CRAN (R 4.4.0)
##    memoise          2.0.1      2021-11-26 [1] CRAN (R 4.4.0)
##    mime             0.12       2021-09-28 [1] CRAN (R 4.4.0)
##    miniUI           0.1.1.1    2018-05-18 [1] CRAN (R 4.4.0)
##    minqa            1.2.7      2024-05-20 [1] CRAN (R 4.4.0)
##    mnormt         * 2.1.1      2022-09-26 [1] CRAN (R 4.4.0)
##    multcomp         1.4-26     2024-07-18 [1] CRAN (R 4.4.2)
##    munsell          0.5.1      2024-04-01 [1] CRAN (R 4.4.0)
##    mvtnorm        * 1.2-5      2024-05-21 [1] CRAN (R 4.4.0)
##    nlme             3.1-164    2023-11-27 [2] CRAN (R 4.4.0)
##    nloptr           2.0.3      2022-05-26 [1] CRAN (R 4.4.0)
##    numDeriv       * 2016.8-1.1 2019-06-06 [1] CRAN (R 4.4.0)
##    pillar           1.9.0      2023-03-22 [1] CRAN (R 4.4.0)
##    pkgbuild         1.4.4      2024-03-17 [1] CRAN (R 4.4.0)
##    pkgconfig        2.0.3      2019-09-22 [1] CRAN (R 4.4.0)
##    pkgload          1.3.4      2024-01-16 [1] CRAN (R 4.4.0)
##    plyr             1.8.9      2023-10-02 [1] CRAN (R 4.4.0)
##    posterior      * 1.5.0      2023-10-31 [1] CRAN (R 4.4.0)
##    profvis          0.3.8      2023-05-02 [1] CRAN (R 4.4.0)
##    promises         1.3.0      2024-04-05 [1] CRAN (R 4.4.0)
##    purrr            1.0.2      2023-08-10 [1] CRAN (R 4.4.0)
##    quantreg         5.98       2024-05-26 [1] CRAN (R 4.4.0)
##    QuickJSR         1.2.0      2024-05-31 [1] CRAN (R 4.4.0)
##    R6               2.5.1      2021-08-19 [1] CRAN (R 4.4.0)
##    Rcpp           * 1.0.12     2024-01-09 [1] CRAN (R 4.4.0)
##  D RcppParallel     5.1.7      2023-02-27 [1] CRAN (R 4.4.0)
##    remotes          2.5.0      2024-03-17 [1] CRAN (R 4.4.0)
##    reshape2         1.4.4      2020-04-09 [1] CRAN (R 4.4.0)
##    rlang            1.1.4      2024-06-04 [1] CRAN (R 4.4.0)
##    rmarkdown        2.29       2024-11-04 [1] CRAN (R 4.4.2)
##    rstan          * 2.32.6     2024-03-05 [1] CRAN (R 4.4.0)
##    rstantools       2.4.0      2024-01-31 [1] CRAN (R 4.4.0)
##    rstudioapi       0.16.0     2024-03-24 [1] CRAN (R 4.4.0)
```

```
##    sandwich         3.1-1      2024-09-15 [1] CRAN (R 4.4.2)
##    scales           1.3.0      2023-11-28 [1] CRAN (R 4.4.0)
##    sessioninfo      1.2.2      2021-12-06 [1] CRAN (R 4.4.0)
##    shiny            1.8.1.1    2024-04-02 [1] CRAN (R 4.4.0)
##    SparseM          1.83       2024-05-30 [1] CRAN (R 4.4.0)
##    StanHeaders    * 2.32.9     2024-05-29 [1] CRAN (R 4.4.0)
##    stringi          1.8.4      2024-05-06 [1] CRAN (R 4.4.0)
##    stringr          1.5.1      2023-11-14 [1] CRAN (R 4.4.0)
##    survival         3.5-8      2024-02-14 [2] CRAN (R 4.4.0)
##    tensorA          0.36.2.1   2023-12-13 [1] CRAN (R 4.4.0)
##    TH.data          1.1-3      2025-01-17 [1] CRAN (R 4.4.0)
##    tibble           3.2.1      2023-03-20 [1] CRAN (R 4.4.0)
##    tictoc         * 1.2.1      2024-03-18 [1] CRAN (R 4.4.0)
##    tidyselect       1.2.1      2024-03-11 [1] CRAN (R 4.4.0)
##    urlchecker       1.0.1      2021-11-30 [1] CRAN (R 4.4.0)
##    usethis          2.2.3      2024-02-19 [1] CRAN (R 4.4.0)
##    utf8             1.2.4      2023-10-22 [1] CRAN (R 4.4.0)
##    V8               5.0.0      2024-08-16 [1] CRAN (R 4.4.1)
##    vctrs            0.6.5      2023-12-01 [1] CRAN (R 4.4.0)
##    withr            3.0.0      2024-01-16 [1] CRAN (R 4.4.0)
##    xfun             0.48       2024-10-03 [1] CRAN (R 4.4.0)
##    xtable           1.8-4      2019-04-21 [1] CRAN (R 4.4.0)
##    yaml             2.3.8      2023-12-11 [1] CRAN (R 4.4.0)
##    zoo              1.8-12     2023-04-13 [1] CRAN (R 4.4.0)
##
##  [1] C:/Users/Carl F Falk/AppData/Local/R/win-library/4.4
##  [2] C:/Program Files/R/R-4.4.0/library
##
##  D -- DLL MD5 mismatch, broken installation.
##
## ------------------------------------------------------------------------------
```