Validation and corrections

Muhammad Yaseen

2026-11-04

Table of contents

1 Abstract **1**

2 Corrections to version 0.3.0 **1**

3 The tables agree with version 0.3.0 **3**

4 The building blocks **3**

5 The compiled sampler and the R reference **4**

6 The stability sampler **5**

7 The information criteria **6**

8 Simulated data **6**

9 The published analysis **8**

10 References **8**

1 Abstract

Version 1.0.0 rewrites the Gibbs sampler of version 0.3.0. This vignette lists what changed and why, and checks the new sampler five ways: on its building blocks, against a plain R version of itself, on its information criteria, on simulated data with known parameters, and on the published analysis (vignette 3).

2 Corrections to version 0.3.0

Function	Version 0.3.0	Now
`bayes_ammi`	singular values computed once and never sampled (`lambdas1` constant)	drawn every iteration from their truncated normal conditional (eq. 7)
`bayes_ammi`	`tau1` held the constant precision of the least-squares fit; σ^2 not returned	`tau1` holds the draws of σ^2
`bayes_ammi`	residuals for σ^2 against the environment of data row j (with `Maiz` always site 1)	residuals of every cell
`bayes_ammi`	within-cell variances summed in place of sums of squares	within-cell sum of squares
`bayes_ammi`	covariance of $\boldsymbol{\tau}$, $\boldsymbol{\delta}$ = square root of the variance; BE for $\boldsymbol{\delta}$	variances; BG for $\boldsymbol{\delta}$
`bayes_ammi`	scores drawn on the full Stiefel manifold, then orthogonalised to $\mathbf{1}$	von Mises-Fisher draws on the constrained spheres (eq. 8-9)
`bayes_ammi`	all $\min(G, E) - 1$ terms; improper prior for σ^2; no burn-in, chains or seed	`.k` (default 2), proper priors, `.burnin`, `.thin`, `.chains`, `.seed`
`biplots`	contours of the singular vectors, fixed bandwidth 0.2; "stable" by bounding box	HPD regions of $\lambda_k^{1/2}\alpha_{ik}$, normal reference bandwidths; region contains the origin
`ge_model`, `ge_ammi`	`lme4::lmer()` for the error variance	closed form (`ge_model()` keeps `lme4`, now suggested)
`Maiz`	help page of another data set	720 rows: site, entry, folds, rep, y
`bayes_ammi`	balanced trials only; one error variance; AMMI only; R loops	missing plots and cells, `.het`, `.model = "GGE"`, DIC and WAIC; compiled sampler

Function	Version 0.3.0	Now
new	none	`bam_predict`, `bam_winners`, `bam_superiority`, `bam_ranks`, `bam_stability`, `bam_compare`, `bam_ppc`, `bam_app`
new	none	`bayes_stability`, `bam_stab_priors`, `bam_indices`, `bam_ammi_indices`, `bam_cite`

3 The tables agree with version 0.3.0

`ge_mean()`, `ge_var()`, `g_eff()`, `e_eff()`, `ge_eff()`, `matrix_k()` and `orthnorm()` return the same numbers as before (test `test-helpers.R`). For example:

```
ge_eff(
    .data = Maiz
  , .y    = y
  , .gen  = entry
  , .env  = site
  )$ge_svd$d[1:3]
```

```
[1] 5922.891 3070.454 2552.407
```

4 The building blocks

The von Mises-Fisher sampler reproduces the mean resultant length $A_p(\kappa) = I_{p/2}(\kappa)/I_{p/2-1}(\kappa)$:

```
set.seed(seed = 7)
p <- 18
vapply(
    X         = c(1, 10, 100)
  , FUN       = function(kap) {
      X <- t(replicate(n = 3000, expr = bayesammi:::.bam_rvmf(v = c(kap,
 ↪  rep(0, p - 1)))))
      c(
          kappa     = kap
        , simulated = mean(X[, 1])
        , theory    = besselI(x = kap, nu = p / 2, expon.scaled = TRUE) /
 ↪  besselI(
```

```
              x            = kap
            , nu           = p / 2 - 1
            , expon.scaled = TRUE
            )
        )
    }
  , FUN.VALUE = numeric(length = 3)
  )
```

```
                [,1]        [,2]         [,3]
kappa     1.00000000 10.0000000 100.0000000
simulated 0.06044483  0.4476345   0.9182527
theory    0.05540201  0.4507702   0.9182146
```

5 The compiled sampler and the R reference

The sampler runs in C++. The package keeps a plain R version of the balanced, common-variance sampler, written directly from the formulas of vignette 2. Both take their random numbers from R in the same order, so from the same seed they must return the same draws:

```
ns <- asNamespace(ns = "bayesammi")
cd <-
  ns$.bam_prep(
      "check"
    , Maiz
    , quote(y)
    , quote(entry)
    , quote(site)
    , quote(rep)
    , environment()
    , allow_missing = TRUE
    )
ls <- ns$.bam_ls(Y = cd$Y, K = 2)
pr <-
  bam_priors(
      mu_mean     = 0
    , mu_var      = 1e+8
    , tau_mean    = 0
    , tau_var     = 1e+8
    , delta_mean  = 0
    , delta_var   = 1e+8
    , lambda_mean = 0
    , lambda_var  = 1e+8
    , nu          = 1
    , s2          = NULL
    )
pr$s2 <- (cd$ssw + 4 * ls$rss) / 600
```

```
st <- ns$.bam_start(ls, pr$s2, 1)
set.seed(seed = 5)
t_r <- system.time(r <- ns$.bam_chain(cd$Y, 4, cd$ssw, 2, pr, st,
 ↪  list(alpha = ls$alpha), 500, 1:500))
set.seed(seed = 5)
t_c <- system.time(cc <- ns$.bam_run_chain(cd, 2, FALSE, FALSE, pr, st,
 ↪  ls$alpha, 500, rep(1L, 500), TRUE, FALSE))
c(scalars = max(abs(cc$scalar - r$scalar)), alpha = max(abs(cc$alpha -
 ↪  r$alpha)), gamma = max(abs(cc$gamma - r$gamma)))
```

```
     scalars        alpha        gamma
1.047738e-09 1.788847e-14 1.250822e-14
```

```
c(R = t_r[["elapsed"]], compiled = t_c[["elapsed"]])
```

```
       R compiled
   0.464    0.064
```

The largest difference is at the level of rounding error, and the compiled sampler is 7 times faster here.

6 The stability sampler

The sampler of `bayes_stability()` has a plain R version too, and the two must give the same draws for all three models:

```
vapply(
    X         = 0:2
  , FUN       = function(code) {
      ls <- ns$.bam_stab_ls(cd, code)
      pr <-
        ns$.bam_stab_fill_priors(
            bam_stab_priors(
                mu_var = NULL
              , g_var  = NULL
              , h_var  = NULL
              , b_var  = 100
              , nu     = 1
              , s2     = NULL
              , nu_d   = 1
              , s2_d   = NULL
              )
          , cd
          , ls
          )
      st <- ns$.bam_stab_start(ls, 1L, code)
      set.seed(seed = 9)
      r  <- ns$.bam_stab_chain(cd$S, cd$Q, cd$N * 1, code, pr, st, 200,
 ↪  rep(1L, 200))
```

```
    set.seed(seed = 9)
    cc <-
      ns$.bam_stab_cpp(
          cd$S
        , cd$Q
        , cd$N * 1
        , code
        , pr
        , st
        , 200L
        , rep(1L, 200)
        , cd$data$y
        , cd$data$cell - 1L
        , FALSE
        )
    max(abs(cc$mu - r$mu), abs(cc$g - r$g), abs(cc$b - r$b), abs(cc$d
     ↪  - r$d))
  }
, FUN.VALUE = numeric(length = 1)
)
```

```
[1] 2.000888e-11 1.455192e-11 1.477929e-11
```

The test `test-stability.R` also recovers known slopes and deviation variances from simulated trials with missing plots and an empty cell.

7 The information criteria

WAIC computed during the run agrees with WAIC computed afterwards by **loo** from the pointwise log-likelihood of the kept draws (test `test-extensions.R`, to 10^{-8}). DIC uses the plug-in deviance at the posterior means.

8 Simulated data

Data are simulated from the model with known parameters, and the 95% HPD intervals should cover them:

```
set.seed(seed = 8)
G <- 8
E <- 12
B <- 3
A  <- qr.Q(qr(cbind(1, matrix(data = rnorm(G * 2), nrow = G))))[, 2:3]
Gm <- qr.Q(qr(cbind(1, matrix(data = rnorm(E * 2), nrow = E))))[, 2:3]
tau <- as.vector(scale(x = rnorm(n = G, mean = 0, sd = 6), scale =
 ↪  FALSE))
del <- as.vector(scale(x = rnorm(n = E, mean = 0, sd = 10), scale =
 ↪  FALSE))
```

```
M <- 100 + outer(X = tau, Y = del, FUN = "+") + A %*% (c(80, 40) * t(Gm))
sim <- expand.grid(rep = 1:B, gen = 1:G, env = 1:E)
sim$y <- M[cbind(sim$gen, sim$env)] + rnorm(n = nrow(sim), mean = 0, sd =
 ↪  5)
fs <-
  bayes_ammi(
      .data   = sim
    , .y      = y
    , .gen    = gen
    , .env    = env
    , .rep    = rep
    , .nIter  = 3000
    , .k      = 2
    , .burnin = 500
    , .thin   = 1
    , .chains = 2
    , .priors = bam_priors(
          mu_mean     = 0
        , mu_var      = 1e+8
        , tau_mean    = 0
        , tau_var     = 1e+8
        , delta_mean  = 0
        , delta_var   = 1e+8
        , lambda_mean = 0
        , lambda_var  = 1e+8
        , nu          = 1
        , s2          = NULL
        )
    , .level  = 0.95
    , .seed   = 9
    , .align  = TRUE
    , .model  = "AMMI"
    , .het    = FALSE
    , .waic   = TRUE
    )
s <- fs$summary
c(tau_covered   = mean(s[block == "tau", lower] <= tau & tau <= s[block
 ↪  == "tau", upper]),
  delta_covered = mean(s[block == "delta", lower] <= del & del <= s[block
   ↪  == "delta", upper]))
```

```
  tau_covered delta_covered
        0.875         1.000
```

```
s[block %in% c("sigma2", "lambda"), list(parameter, truth = c(25, 80,
 ↪  40), mean, lower, upper)]
```

```
   parameter truth     mean    lower    upper
1:    sigma2    25 27.22189 22.25513 32.24195
```

```
2: lambda[1]    80 80.55573 74.65189 86.69097
3: lambda[2]    40 41.83578 35.73567 47.56598
```

9 The published analysis

Vignette 3 compares all 33 parameters of Table 1 and the 58 first scores of Table B1 of Crossa et al. (2011). Every posterior mean is within a quarter of a posterior standard deviation of the published one, and the test `test-crossa2011.R` repeats this comparison.

```
bam_reference(
    source   = "Crossa"
  , evaluate = TRUE
  , level    = 1
  )[, list(quantity, value, computed, agrees)]
```

```
                                            quantity value computed agrees
1:           least-squares lambda_1 of the maize data  5923 5922.891   TRUE
2:           least-squares lambda_2 of the maize data  3070 3070.454   TRUE
3: grand mean of the maize data (posterior mean 4858)  4858 4858.138   TRUE
```

10 References

Crossa, J., Perez-Elizalde, S., Jarquin, D., Cotes, J. M., Viele, K., Liu, G. and Cornelius, P. L. (2011). Bayesian estimation of the additive main effects and multiplicative interaction model. *Crop Science*, 51(4), 1458–1469. doi:10.2135/cropsci2010.06.0343

Wood, A. T. A. (1994). Simulation of the von Mises Fisher distribution. *Communications in Statistics - Simulation and Computation*, 23(1), 157–164.

Mardia, K. V. and Jupp, P. E. (2000). *Directional Statistics*. Wiley, Chichester.