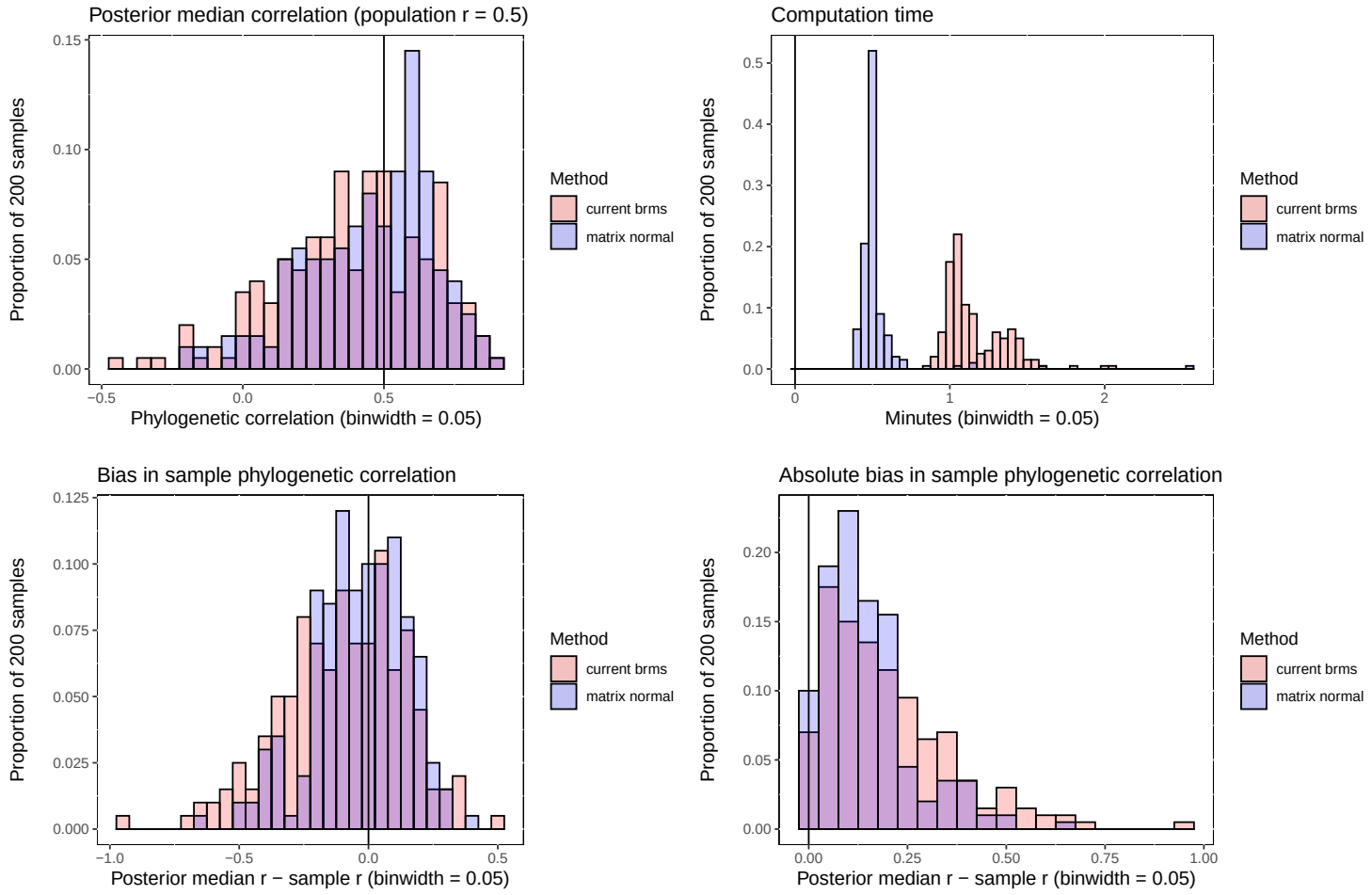
Matrix normal comparison with current brms

Jordan S. Martin

6/19/2021

Results comparing a bivariate phylogenetic correlation estimated with current development (2.15.9) brms code and updated brms code using a matrix normal parameterization across 200 simulated datasets (n = 100).



Code for running the simulation.

```
############################################################
#generate Stan code for brms model with Kronecker product
############################################################

#simulate scaling matrix
A = rethinking::rlkjcorr(1, 100, 1)

#simulate correlated phylogenetic effects
r_G = 0.5 #phylo correlation
v_G = 0.5 #phylo variance

G_cor <- matrix(c(1,r_G,r_G,1), nrow=2, ncol=2)
G_sd  <- c(sqrt(v_G),sqrt(v_G))
G <- diag(G_sd) %*% G_cor %*% diag(G_sd)
Kron.prod <- G %x% A
P <- matrix(mvtnorm::rmvnorm(1, mean=rep(0,nrow(A)*2), sigma=Kron.prod), ncol = 2)
cor(P)

#Gaussian responses
v_res = 0.5 #residual variance (assume independent errors)
t1 = 0 + P[,1] + rnorm(nrow(A), 0, sqrt(v_res))
t2 = 0 + P[,2] + rnorm(nrow(A), 0, sqrt(v_res))

library(brms); library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

df = data.frame(t1,t2, phylo = seq(1:nrow(A)))
rownames(A) = df$phylo

write(make_stancode(formula = bf(t1 ~ 0 + (1|G| gr(phylo, cov = A))) +
                      bf(t2 ~ 0 + (1|G| gr(phylo, cov = A))),
                    data = df, data2 = list(A = A),
                    prior = c(prior("exponential(1)", class = "sd", resp = "t1"),
                              prior("exponential(1)", class = "sigma", resp = "t1"),
                              prior("exponential(1)", class = "sd", resp = "t2"),
                              prior("exponential(1)", class = "sigma", resp = "t2"),
                              prior("lkj(1)", class = "cor"))), "m1.stan")

m1 = stan_model("m1.stan")
```

```
##############################################
#modify brms code w/ matrix normal sampling
##############################################

#removed function block w/ Kronecker product function
write(
  "// generated with brms 2.15.9
    data {
      int<lower=1> N;  // total number of observations
      int<lower=1> N_t1;  // number of observations
      vector[N_t1] Y_t1;  // response variable
      int<lower=1> N_t2;  // number of observations
      vector[N_t2] Y_t2;  // response variable
      int<lower=1> nresp;  // number of responses
      int nrescor;  // number of residual correlations
      // data for group-level effects of ID 1
      int<lower=1> N_1;  // number of grouping levels
      int<lower=1> M_1;  // number of coefficients per level
      int<lower=1> J_1_t1[N_t1];  // grouping indicator per observation
      int<lower=1> J_1_t2[N_t2];  // grouping indicator per observation
      matrix[N_1, N_1] Lcov_1;  // cholesky factor of known covariance matrix
      // group-level predictor values
      vector[N_t1] Z_1_t1_1;
      vector[N_t2] Z_1_t2_2;
      int<lower=1> NC_1;  // number of group-level correlations
      int prior_only;  // should the likelihood be ignored?
    }
    transformed data {
      vector[nresp] Y[N];  // response array
      for (n in 1:N) {
        Y[n] = transpose([Y_t1[n], Y_t2[n]]);
      }
    }
    parameters {
      real<lower=0> sigma_t1;  // dispersion parameter
      real<lower=0> sigma_t2;  // dispersion parameter
      cholesky_factor_corr[nresp] Lrescor;  // parameters for multivariate linear models
      vector<lower=0>[M_1] sd_1;  // group-level standard deviations
      matrix[M_1, N_1] z_1;  // standardized group-level effects
      cholesky_factor_corr[M_1] L_1;  // cholesky factor of correlation matrix
    }
    transformed parameters {
      matrix[N_1, M_1] r_1;  // actual group-level effects
      // using vectors speeds up indexing in loops
      vector[N_1] r_1_t1_1;
      vector[N_1] r_1_t2_2;
      // compute actual group-level effects
      r_1 = Lcov_1 * z_1' * diag_pre_multiply(sd_1, L_1)';
      r_1_t1_1 = r_1[, 1];
      r_1_t2_2 = r_1[, 2];
    }
    model {
      // likelihood including constants
      if (!prior_only) {
        // initialize linear predictor term
        vector[N_t1] mu_t1 = rep_vector(0.0, N_t1);
        // initialize linear predictor term
        vector[N_t2] mu_t2 = rep_vector(0.0, N_t2);
```

```
      // multivariate predictor array
      vector[nresp] Mu[N];
      vector[nresp] sigma = transpose([sigma_t1, sigma_t2]);
      // cholesky factor of residual covariance matrix
      matrix[nresp, nresp] LSigma = diag_pre_multiply(sigma, Lrescor);
      for (n in 1:N_t1) {
        // add more terms to the linear predictor
        mu_t1[n] += r_1_t1_1[J_1_t1[n]] * Z_1_t1_1[n];
      }
      for (n in 1:N_t2) {
        // add more terms to the linear predictor
        mu_t2[n] += r_1_t2_2[J_1_t2[n]] * Z_1_t2_2[n];
      }
      // combine univariate parameters
      for (n in 1:N) {
        Mu[n] = transpose([mu_t1[n], mu_t2[n]]);
      }
      target += multi_normal_cholesky_lpdf(Y | Mu, LSigma);
    }
    // priors including constants
    target += exponential_lpdf(sigma_t1 | 1);
    target += exponential_lpdf(sigma_t2 | 1);
    target += lkj_corr_cholesky_lpdf(Lrescor | 1);
    target += exponential_lpdf(sd_1 | 1);
    target += std_normal_lpdf(to_vector(z_1));
    target += lkj_corr_cholesky_lpdf(L_1 | 1);
  }
  generated quantities {
    // residual correlations
    corr_matrix[nresp] Rescor = multiply_lower_tri_self_transpose(Lrescor);
    vector<lower=-1,upper=1>[nrescor] rescor;
    // compute group-level correlations
    corr_matrix[M_1] Cor_1 = multiply_lower_tri_self_transpose(L_1);
    vector<lower=-1,upper=1>[NC_1] cor_1;
    // extract upper diagonal of correlation matrix
    for (k in 1:nresp) {
      for (j in 1:(k - 1)) {
        rescor[choose(k - 1, 2) + j] = Rescor[j, k];
      }
    }
    // extract upper diagonal of correlation matrix
    for (k in 1:M_1) {
      for (j in 1:(k - 1)) {
        cor_1[choose(k - 1, 2) + j] = Cor_1[j, k];
      }
    }
  }", "m2.stan")

m2 = stan_model("m2.stan")
```

```
#########################################################################
#compare estimation bias in phylogenetic correlation between approaches
#########################################################################
run = 200 #number of random samples
med_brm = data.frame(n = seq(1:run), time = NA, cor = NA, cor_pop_bias = NA, cor_emp_bias = NA)
med_mtn = data.frame(n = seq(1:run), time = NA, cor = NA, cor_pop_bias = NA, cor_emp_bias = NA)

counter = 0 #track progress
for(i in 1:run){
  ########################
  #sim data
  ########################
  #simulate scaling matrix
  A = rethinking::rlkjcorr(1, 100, 1)

  #simulate correlated phylogenetic effects
  r_G = 0.5 #true population phylo correlation
  v_G = 0.5 #phylo variance
  G_cor <- matrix(c(1,r_G,r_G,1), nrow=2, ncol=2)
  G_sd  <- c(sqrt(v_G),sqrt(v_G))
  G <- diag(G_sd) %*% G_cor %*% diag(G_sd)
  Kron.prod <- G %x% A
  P <- matrix(mvtnorm::rmvnorm(1, mean=rep(0,nrow(A)*2), sigma=Kron.prod), ncol = 2)
  emp_cor = cor(P)[2,1] #true sample correlation

  #Gaussian responses
  v_res = 0.5 #residual variance (assume independent errors)
  t1 = 0 + P[,1] + rnorm(nrow(A), 0, sqrt(v_res))
  t2 = 0 + P[,2] + rnorm(nrow(A), 0, sqrt(v_res))

  df = data.frame(t1,t2, phylo = seq(1:nrow(A)))
  rownames(A) = df$phylo

  stan_data = make_standata(formula = bf(t1 ~ 0 + (1|G| gr(phylo, cov = A))) +
                              bf(t2 ~ 0 + (1|G| gr(phylo, cov = A))),
                            data = df, data2 = list(A = A))
  ########################
  #current brms model
  ########################
  start_time <- Sys.time() #time model
  mod1 <- sampling(m1, data= stan_data, init = 0, iter = 2000, warmup = 1000)
  med_brm[i,"time"] = Sys.time() - start_time
  post1 = extract(mod1)
  med_brm[i,"cor"] = median(post1$cor_1) #phylo correlation
  med_brm[i,"cor_pop_bias"] = median(post1$cor_1) - r_G
  med_brm[i,"cor_emp_bias"] = median(post1$cor_1) - emp_cor

  ########################
  #matrix normal model
  ########################
  start_time = Sys.time()
  mod2 <- sampling(m2, data= stan_data, init = 0, iter = 2000, warmup = 1000)
  med_mtn[i,"time"] = Sys.time() - start_time
  post2 = extract(mod2)
  med_mtn[i,"cor"] = median(post2$cor_1) #phylo correlation
  med_mtn[i,"cor_pop_bias"] = median(post2$cor_1) - r_G
  med_mtn[i,"cor_emp_bias"] = median(post2$cor_1) - emp_cor
```

```
  saveRDS(med_brm,"med_brm.RDS")
  saveRDS(med_mtn,"med_mtn.RDS")

  counter <- counter + 1;
  print(paste(counter/run*100, "% has been processed"))

}

med_brm = readRDS("med_brm.RDS")
med_mtn = readRDS("med_mtn.RDS")

d = data.frame(diff = med_brm$cor - med_mtn$cor)

library(ggplot2)

#object for ggplot
med_brm$type = "current brms"
med_mtn$type = "matrix normal"
ldf = rbind(med_brm, med_mtn)


#compare corr
p1 =
ggplot(ldf, aes(cor, fill = type)) +
  geom_histogram(aes(y=..count../run), color = "black", position='identity',
                 binwidth=0.05, alpha = 0.20)+
  scale_fill_manual(values = c("red", "blue"), name = "Method")+
  geom_vline(xintercept=0.5)+
  ggtitle("Posterior median correlation (population r = 0.5)")+
  xlab("Phylogenetic correlation (binwidth = 0.05)")+
  ylab("Proportion of 200 samples\n")+
  theme(panel.background = element_rect(fill='white', colour='black'),
        axis.title = element_text(size = 12))

#compare computational time
ldf$min = ifelse(ldf$time>10, ldf$time/60, ldf$time)  #change seconds to minutes

p2 =
  ggplot(ldf, aes(min, fill = type)) +
  geom_histogram(aes(y=..count../run), color = "black", position='identity',
                 binwidth=0.05, alpha = 0.20)+
  scale_fill_manual(values = c("red", "blue"), name = "Method")+
  geom_vline(xintercept=0)+
  ggtitle("Computation time")+
  xlab("Minutes (binwidth = 0.05)")+
  ylab("Proportion of 200 samples\n")+
  theme(panel.background = element_rect(fill='white', colour='black'),
        axis.title = element_text(size = 12))

#sample bias
p3 =
  ggplot(ldf, aes(cor_emp_bias, fill = type)) +
    geom_histogram(aes(y=..count../run), color = "black", position='identity',
                   binwidth=0.05, alpha = 0.20)+
    scale_fill_manual(values = c("red", "blue"), name = "Method")+
    geom_vline(xintercept=0)+
    ggtitle("Bias in sample phylogenetic correlation")+
    xlab("Posterior median r - sample r (binwidth = 0.05)")+
```

```
    ylab("Proportion of 200 samples\n")+
    theme(panel.background = element_rect(fill='white', colour='black'),
          axis.title = element_text(size = 12))

  #sample bias
p4 =
  ggplot(ldf, aes(abs(cor_emp_bias), fill = type)) +
    geom_histogram(aes(y=..count../run), color = "black", position='identity',
                   binwidth=0.05, alpha = 0.20)+
    scale_fill_manual(values = c("red", "blue"), name = "Method")+
    geom_vline(xintercept=0)+
    ggtitle("Absolute bias in sample phylogenetic correlation")+
    xlab("Posterior median r - sample r (binwidth = 0.05)")+
    ylab("Proportion of 200 samples\n")+
    theme(panel.background = element_rect(fill='white', colour='black'),
          axis.title = element_text(size = 12))


library(cowplot)

p = plot_grid(p1, p2, p3,  p4, ncol = 2)
  #plot_grid(p1, p2, p3,  p5, p4, p6, ncol = 2)
save_plot(p, filename= "matrix normal brms comparison.png", base_height = 8, base_width = 12)
save_plot(p, filename= "matrix normal brms comparison.pdf", base_height = 8, base_width = 12)
```