Exploring `mcmcsamp` results

Ben Bolker

July 2, 2012

1 Introduction

This document is an attempt at a quick (??) exploration of the behavior of `mcmcsamp`, and a comparison against the results from the same model fitted in `MCMCglmm` — which, except for the priors and any coding mistakes on my part, should be fitting the same model For reference, the priors used in `MCMCglmm` (based on the description in `?MCMCglmm` are $N(\mu = 0, \sigma^2 = 10^{10} I)$ for the fixed-effect parameters and Inv-Wishart$(\nu = 0, \Sigma = I)$ for the random-effect parameters.

In most of the cases below I would be surprised if the priors made a big difference, because these examples are generally well-behaved/well-defined cases where I would expect the data to be determining the variance components pretty strongly ...

2 Code

Functions to extract variances for fitted and MCMC variances. The functions will only work sensibly for problems with a single variance component per grouping factor, but these are also the only ones that `mcmcsamp` can handle, so we should be safe ...? These codes started out trying to be simple but ending up getting more complicated as I tried to make them more general ...

```
get_pvars <- function(object) {
    v <- c(unlist(VarCorr(object)),units=sigma(object)^2)
    names(v) <- c(names(getME(object,"theta")),"units")
    v
}

get_mcvars <- function(object,mcmc,mcmcglmm=NULL) {
    ## get pvars, convert to 1-row data frame
    pvars <- as.data.frame(rbind(get_pvars(object)))
    ## get mcmc vars, drop fixed effects, rename
    nfix <- length(fixef(object))
    m0 <- as.data.frame(mcmc,type="varcov")[,-(1:nfix)]
    mmvars <- setNames(m0,names(pvars))
    L <- list(fitted=pvars,mcmc=mmvars)
    if (!is.null(mcmcglmm)) {
```

```
        m1 <- as.data.frame(mcmcglmm$VCV)
        ## re-arrange column names to match lmer
        regexstr <-
"([[:alpha:]_\\(\\)]+)\\.([[:alpha:]_\\(\\)]+)"
        names(m1) <- gsub(regexstr,"\\2.\\1",names(m1))
        f1 <- rbind(colMeans(m1))
        ## combine mcmc and lmer results
        L$mcmc <- rbind(data.frame(method="mcmcsamp",
                                   step=seq(nrow(L$mcmc)),
                                   L$mcmc,check.names=FALSE),
                        data.frame(method="MCMCglmm",

step=seq(nrow(m1)),m1,check.names=FALSE))
        L$fitted <- rbind(data.frame(method="mcmcsamp",
                                     L$fitted,check.names=FALSE),

data.frame(method="MCMCglmm",f1,check.names=FALSE))
    }
    L
}
```

While there certainly could be mistakes in these extraction functions, I have kept them as simple as possible, and the results match those printed by `summary()`, `VarCorr()`, etc., as well as the results of using `getME(.,"theta")` to extract the raw parameters (Cholesky components, which have to be multiplied by `sigma(.)` and squared to get the RE variances).

Code for producing plots etc. is hidden; look at the original `Rnw` file if you want the gory details.

3 Examples

3.1 sleepstudy

```
fm1 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject),
sleepstudy)
vc(fm1)
```

```
##  Groups   Name        Variance Std.Dev.
##  Subject  (Intercept) 627.6    25.05
##  Subject  Days         35.9     5.99
##  Residual             653.6    25.57
```

```
mm1 <- mcmcsamp(fm1,1000)
fm1G <- MCMCglmm(Reaction ~ Days,
                 random=~idh(1+Days):Subject, data=sleepstudy,
                 verbose=FALSE)
```

```
v1 <- get_mcvars(fm1,mm1,fm1G)
```

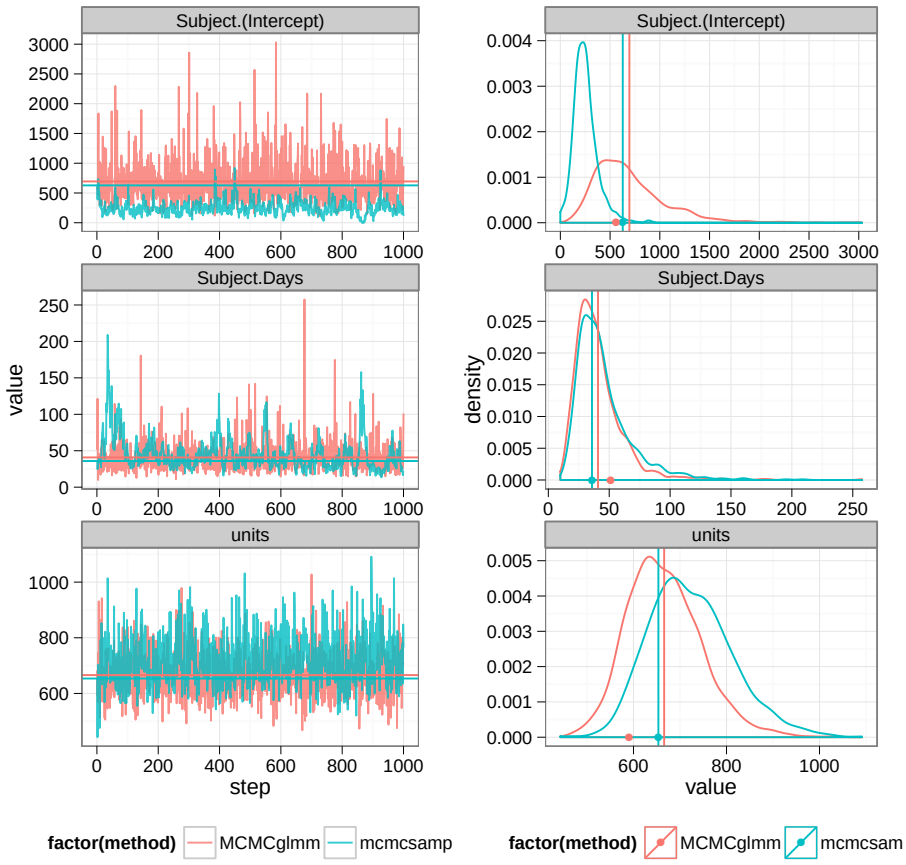
(The `MCMCglmm` expression could be written more simply as `~Subject`: in general I am writing out the fully explicit random effects specification so that the variable names `MCMCglmm` uses match those given by `lmer`.)

In these plots (and those to follow) the left-hand column shows the trace plots, the right the posterior densities; the red vertical and horizontal lines are the fitted values from the original model, the blue points (in all cases identical to the red lines) show the starting points of the MCMC chains. (In the one simulated example below, purple lines indicate the true variances.)

The results for this first (`sleepstudy`) example seem reasonable, although the `mcmcsamp` results for the intercept term (first row) don't match `MCMCglmm` particularly well. (The fitted value from `lmer` does, so the problem seems to be less with `lmer` in general than with what `mcmcsamp` is doing ...)

(first row) is quite close to the edge of its putative posterior distribution.

```
gg1 <- do_plots(v1)
grid.arrange(gg1[[1]],gg1[[2]],ncol=2)
```



We could produce a scatterplot matrix

```
splom(subset(v1$mcmc,select=-c(step,method)),pch=".")
```

but it's not very informative in this case — and I haven't yet gone to the trouble of superimposing the fitted values.

(Hereafter the code is suppressed in the output, since it's practically the same.)

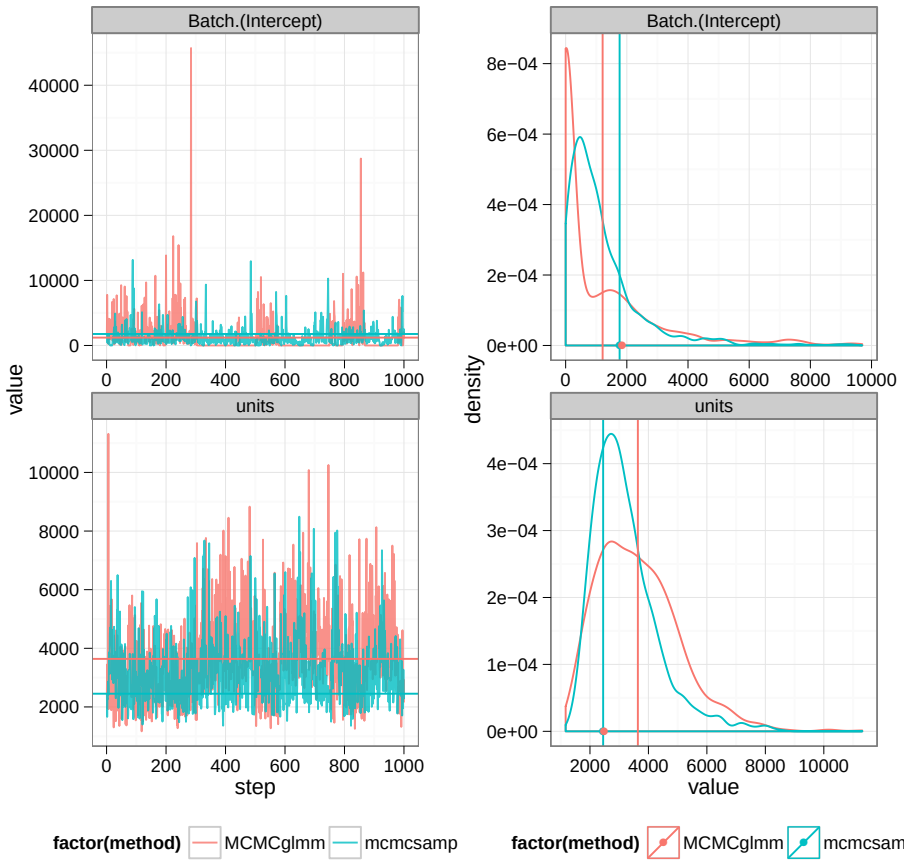
3.2 Dyestuff

```
fm2 <- lmer(Yield ~ 1|Batch, Dyestuff)
fm2M <- MCMCglmm(Yield ~ 1,random=~idh(1):Batch,
data=Dyestuff,verbose=FALSE)
vc(fm2)
##  Groups   Name        Variance Std.Dev.
##  Batch    (Intercept) 1764     42.0
##  Residual             2451     49.5
```

This one looks OK too (I'm still slightly puzzled that the fitted value doesn't align with the modes of the distribution, but maybe that's a marginal vs. unconditional maximum value issue?)

(Note that I have thrown away extreme values (> 10000) of the batch variance in the density, for easier visualization.)

The `MCMCglmm` results for the batch variance actually look a little wonky — is the problem more poorly constrained than I think?

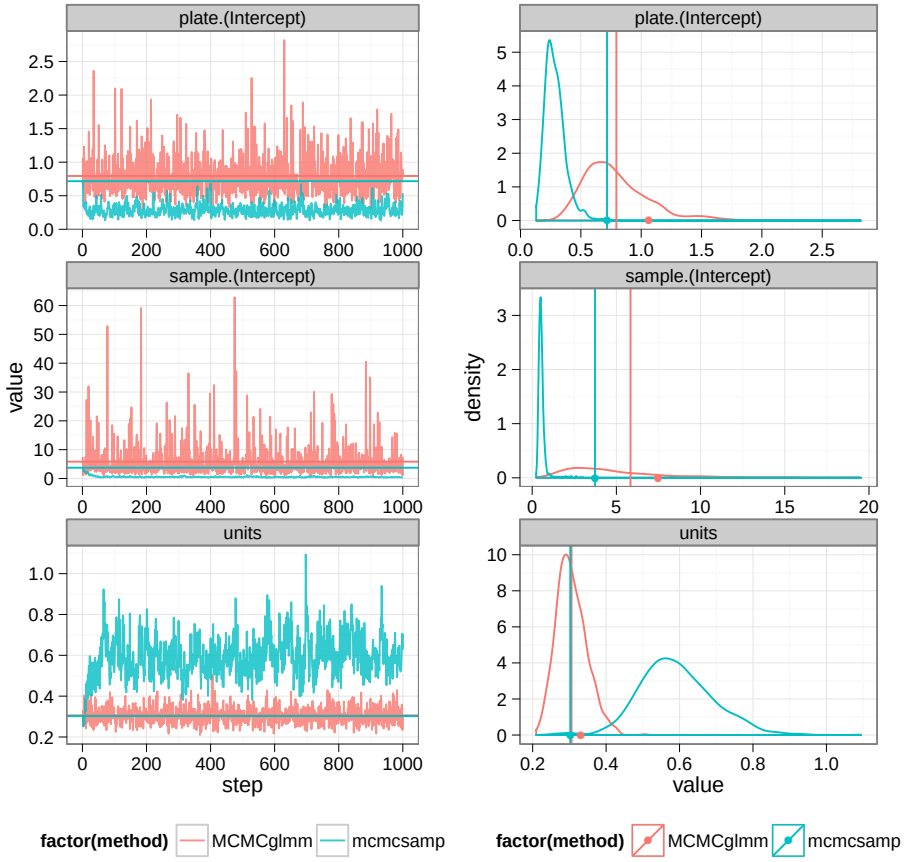


3.3 Penicillin

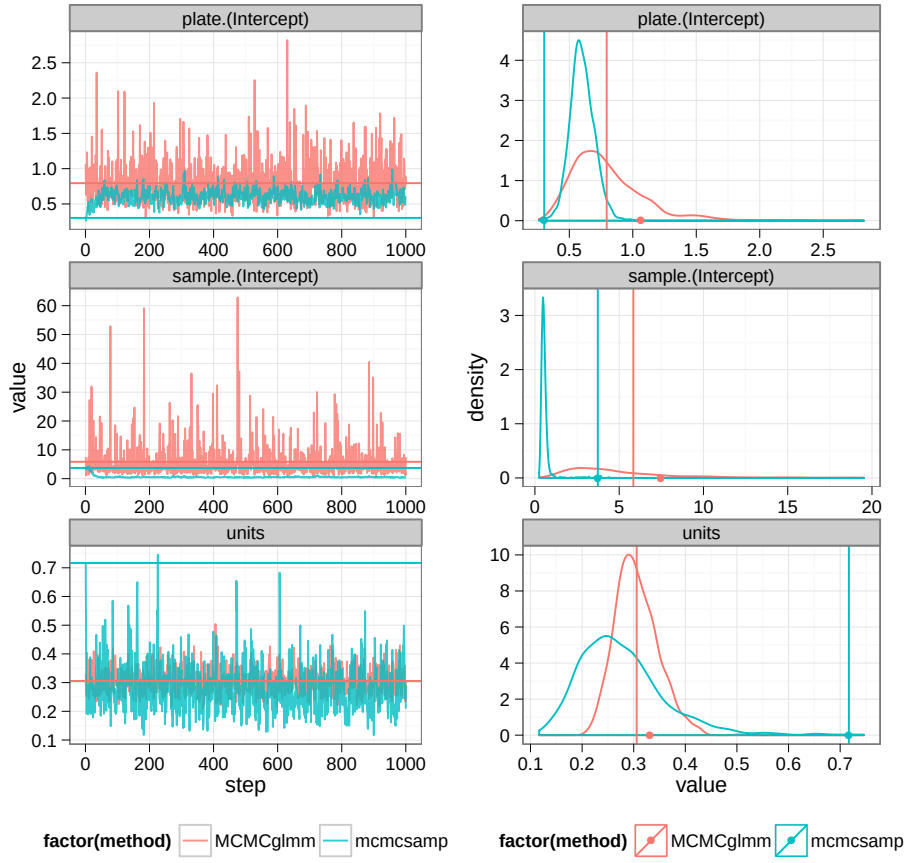
```
fm3 <- lmer(diameter ~ (1|plate) + (1|sample), Penicillin)
fm3M <- MCMCglmm(diameter ~ 1,random=~idh(1):plate+idh(1):sample,
                              data=Penicillin,verbose=FALSE)
vc(fm3)
##  Groups   Name        Variance Std.Dev.
##  plate    (Intercept) 0.717    0.847
##  sample   (Intercept) 3.731    1.932
##  Residual             0.302    0.550
```

Now it gets weird. I could very well be doing something dumb, but I have checked things multiple times and can't see it. It *looks* like the series for the plate intercept and the residual variance could be switched (the mean of the plate intercept term MCMC density shown here is about 0.25, which fits pretty well with fitted residual variance; the MCMC density for the residual variance is ≈ 0.6, which fits the fitted plate-intercept variance).

(Restricting sample-intercept variances to < 20 in the density.)



If we switch the units and plate-intercept terms for the `mcmcsamp` things look slightly more reasonable (the `mcmcsamp` gets run again, so it won't be identical).



Double-checking a relatively raw form with less possibility for accidental switching:

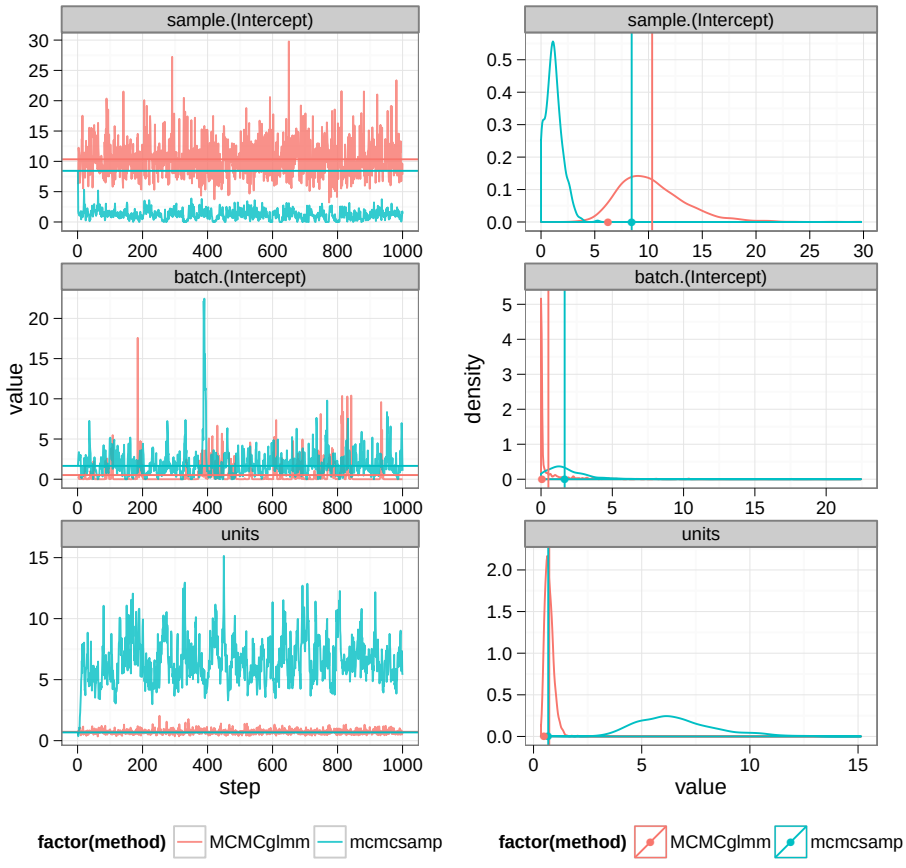
```
summary(as.data.frame(mcmcsamp(fm3,1000),
                      type="varcov"))
##   (Intercept)         V2              V3              V4
##  Min.   :21.6   Min.   :0.130   Min.   :0.239   Min.   :0.252
##  1st Qu.:22.7   1st Qu.:0.227   1st Qu.:0.436   1st Qu.:0.524
##  Median :22.9   Median :0.273   Median :0.513   Median :0.582
##  Mean   :22.9   Mean   :0.286   Mean   :0.567   Mean   :0.592
##  3rd Qu.:23.2   3rd Qu.:0.329   3rd Qu.:0.601   3rd Qu.:0.651
##  Max.   :23.9   Max.   :0.717   Max.   :3.731   Max.   :1.094
```

The other evidence that I haven't gotten something backward is that the MCMC chains do *start* from the correct value that matches the fitted result — and what's up with the sample intercept ???

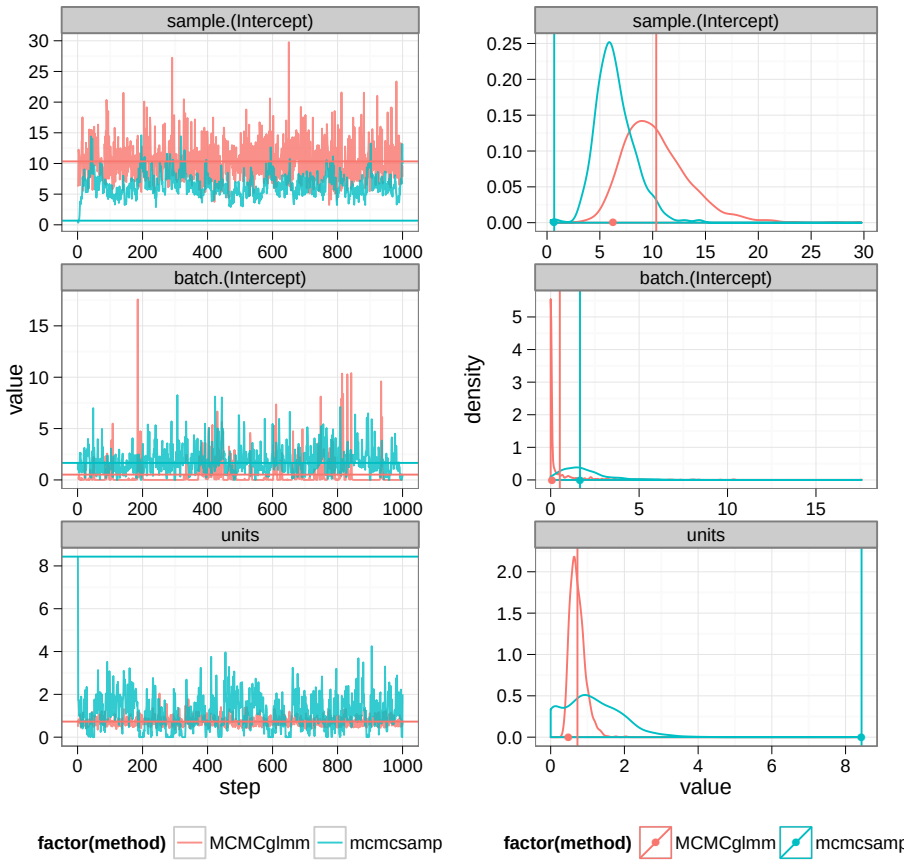
3.4 Pastes

```
fm4 <- lmer(strength ~ (1|batch) + (1|sample), Pastes)
fm4M <- MCMCglmm(strength ~ 1,
                 random=~idh(1):batch + idh(1):sample,
                 data=Pastes,verbose=FALSE)
vc(fm4)
##  Groups   Name        Variance Std.Dev.
##  sample   (Intercept) 8.434    2.904
##  batch    (Intercept) 1.657    1.287
##  Residual             0.678    0.823
```

Here, again, it seems as though the first and third components might be switched. The `MCMCglmm` fit for the batch intercept isn't mixing very well, so it's hard to see the density plot, but the horizontal line in the trace plot indicates here that `mcmcsamp` is actually doing a good job ...



Try it with variables #1 and #3 switched ...



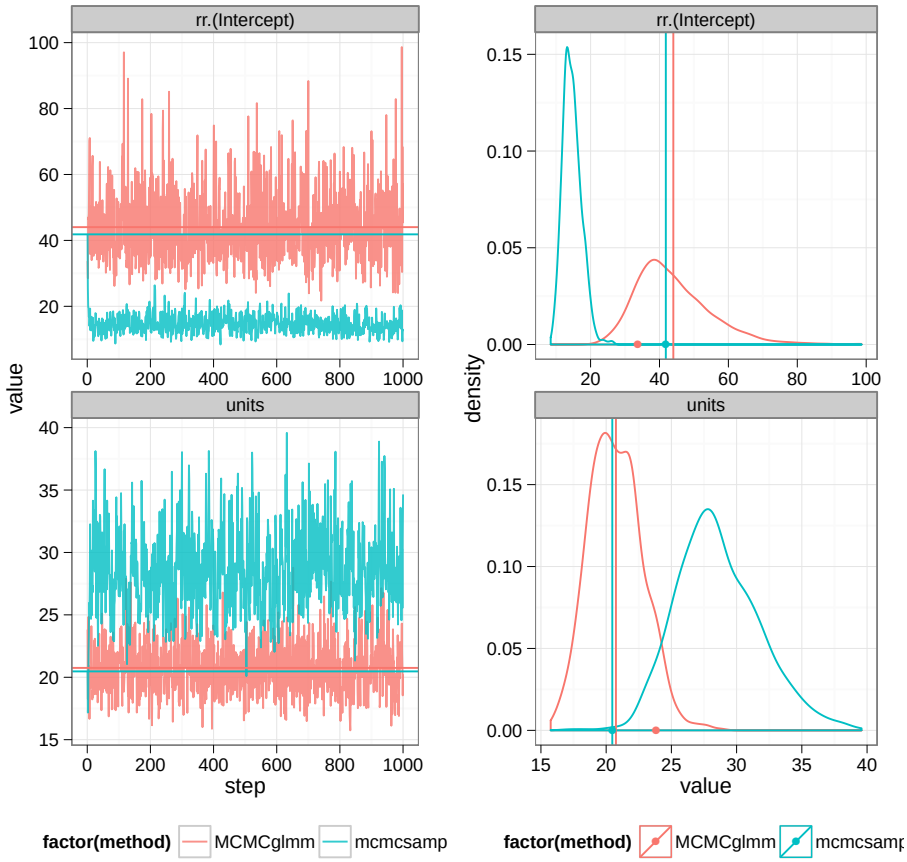
Comparing direct calculation:

```
colMeans(as.data.frame(mcmcsamp(fm3,1000),
                       type="varcov"))[-1]
##     V2     V3     V4
## 0.2737 0.5953 0.5922
```

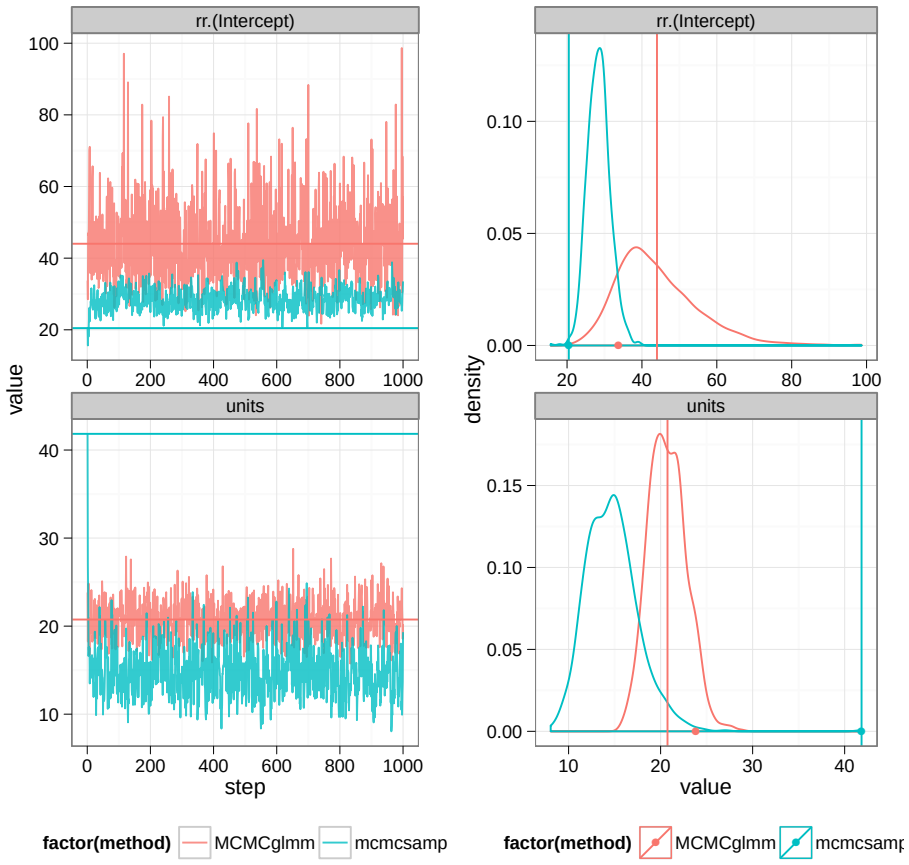
3.5 cake

```
cake <- transform(cake,rr=interaction(recipe,replicate))
fm5 <- lmer(angle ~ recipe * temperature + (1|rr), cake)
fm5M <- MCMCglmm(angle ~ recipe * temperature,
                 random=~idh(1):rr,
                 data=cake,verbose=FALSE)
vc(fm5)
##  Groups   Name        Variance Std.Dev.
##  rr       (Intercept) 41.8     6.47
```

```
##  Residual                20.5     4.52
```



Way off again.
Switching variables doesn't really fix it:



3.6 A constructed example

From Laurent Stephane:

```
set.seed(666)
sims <- function(I, J, sigmab0, sigmaw0){
    Mu <- rnorm(I, mean=0, sd=sigmab0)
    y <- c(sapply(Mu, function(mu) rnorm(J, mu, sigmaw0)))
    data.frame(y=y, group=gl(I,J))
}

I <- 20  # number of groups
J <- 20  # number of repeats per group
sigmab0 <- 1 # between standard deviation
sigmaw0 <- 2  # within standard deviation
dat <- sims(I, J, sigmab0, sigmaw0)
```

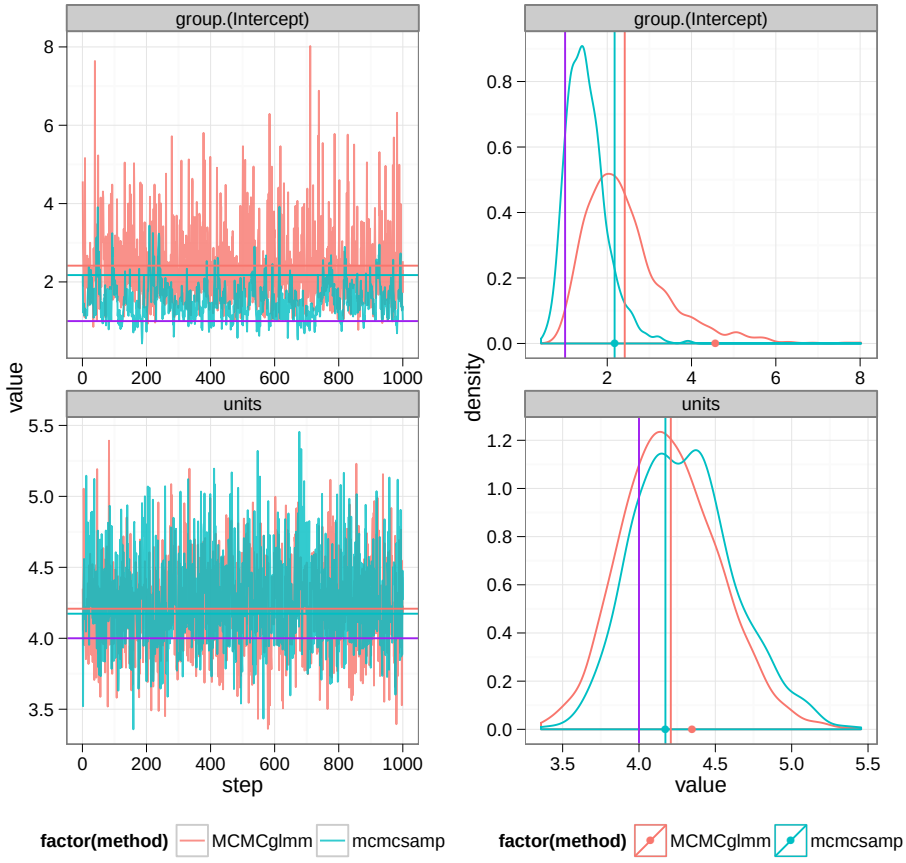
How typical is this example/how much among-realization variability is there?

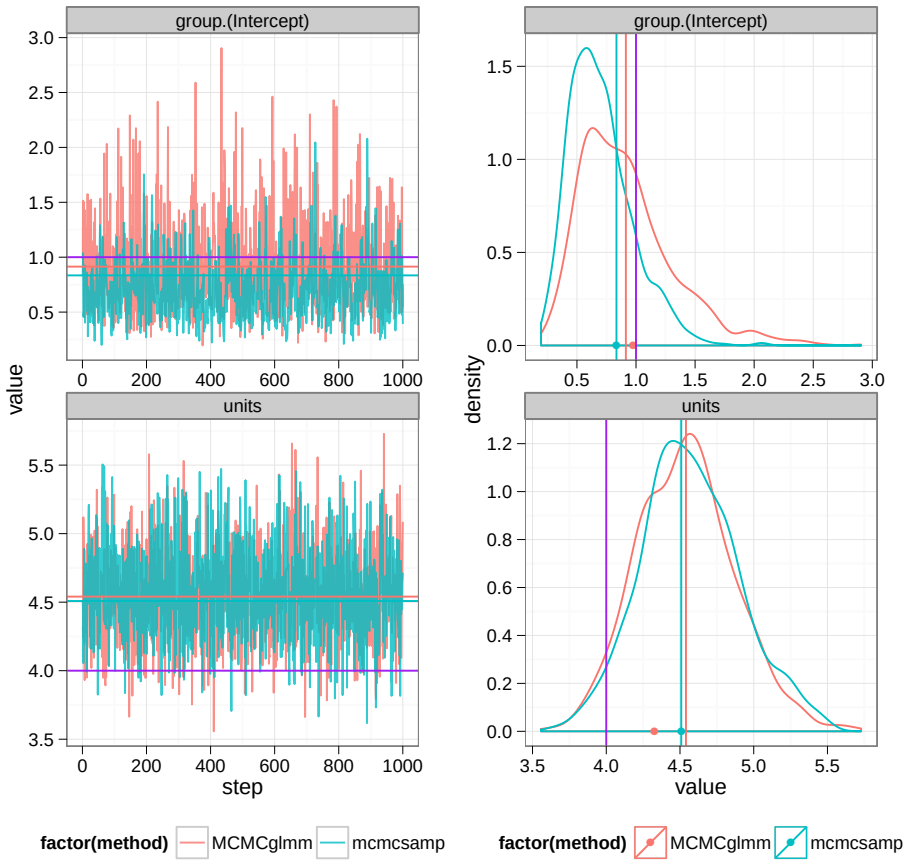
Trying another example ...

```
fm5 <- lmer(y~(1|group),data=dat)
fm5M <- MCMCglmm(y~1,
                 random=~idh(1):group,
                 data=dat,
                 verbose=FALSE)
vc(fm5)
##  Groups   Name        Variance Std.Dev.
##  group    (Intercept) 2.17     1.47
##  Residual             4.17     2.04
```

In this case the results look not-too-unreasonable — although who knows what would happen if I used two random effects?

```
tvals <- data.frame(variable=factor(names(get_pvars(fm5))),
                    value=c(1,4),
                    method=rep("mcmcsamp",2),
                    stringsAsFactors=FALSE)
do_all(fm5,MCMCglmmfit=fm5M,
       truevals=tvals)
```


group.(Intercept)
group.(Intercept)
units
units
value
density
step
value
factor(method) MCMCglmm mcmcsamp
factor(method) MCMCglmm mcmcsamp



To follow this up I should really run a large number of simulations and show the distribution of point estimates/distribution of HPD intervals or quantiles/coverage ...

4 The bottom line

It's still not clear to me whether the apparent differences between the fitted values from `lmer` and (1) the `mcmcsamp` results and (2) the `MCMCglmm` results are due to (a combination of):

- a bug in the `mcmcsamp` code that switches the order of the theta (random effects) parameters in some cases?
- differences in priors between `lmer` and `MCMCglmm`
- differences in the Bayesian answer (mean/HPD interval of marginal posterior distribution) and MLE answer
- per-realization variability (i.e. for the real data, the answer might look

different for a different realization of the same system; for simulations, we can do this precisely

5 Session info

```
sessionInfo()
## R Under development (unstable) (2012-05-12 r59340)
## Platform: i686-pc-linux-gnu (32-bit)
##
## locale:
##  [1] LC_CTYPE=en_CA.utf8       LC_NUMERIC=C
##  [3] LC_TIME=en_CA.utf8        LC_COLLATE=en_CA.utf8
##  [5] LC_MONETARY=en_CA.utf8    LC_MESSAGES=en_CA.utf8
##  [7] LC_PAPER=C                LC_NAME=C
##  [9] LC_ADDRESS=C              LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_CA.utf8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] MCMCglmm_2.16      corpcor_1.6.3      ape_3.0-3
##  [4] coda_0.14-7        tensorA_0.36       gridExtra_0.9
##  [7] lme4.0_0.999999-1  Matrix_1.0-6       lattice_0.20-6
## [10] knitr_0.5.10       RColorBrewer_1.0-5 igraph_0.5.5-4
## [13] ggplot2_0.9.1      reshape2_1.2.1     plyr_1.7.1
##
## loaded via a namespace (and not attached):
##  [1] codetools_0.2-8   colorspace_1.2-0  dichromat_1.2-4
##  [4] digest_0.5.2      evaluate_0.4.2    formatR_0.4
##  [7] gee_4.13-18       highlight_0.3.2   labeling_0.2
## [10] lme4_0.99999911-0 MASS_7.3-18       memoise_0.1
## [13] minqa_1.2.1       munsell_0.3       nlme_3.1-103
## [16] parser_0.0-14     proto_0.3-9.2     Rcpp_0.9.10
## [19] scales_0.2.1      splines_2.16.0    stats4_2.16.0
## [22] stringr_0.6       tools_2.16.0
```