Bayesian Uncertainty Modelling in A Mass Cytometry Experiment

Eric Barnhill and Shravan Vasishth

February 19, 2018

Some additional code and analyses by Shravan Vasishth (vasishth at uni-potsdam.de).

Introduction

The goal of this experiment was to estimate robustness and uncertainty of effects in a mass cytometry experiment. In this experiment, the response of contrast agents in six cell types was studied. As the cell types behave biologically differently, a separate statistical model was built for each.

In each model, three experiments were run, in which cell samples were subjected to three differing contrast agents:

- Magnevist
- Dotarem
- Gadevist

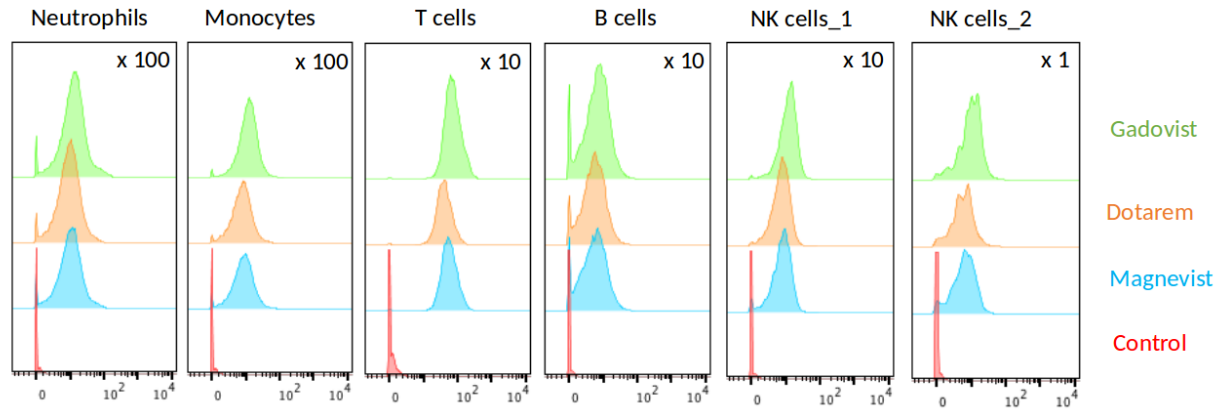
and a control condition. Cells were subjected to these agents at three differing concentrations: 0.1, 0.3, and 1, and the control condition was considered a concentration of 0.0. Thus a fully specified model for the experiment would be:

$$E(Y_{ijk}) = \beta_0 + u_{0j} + w_{0k} + (\beta_1 + u_{1j} + w_{1k})conc_i + \epsilon_i$$

where β_0 is the global intercept, u_{0j} is a random intercept for experiment, w_{0k} is a random intercept for contrast agent, and similarly, β_1, u_{1j} and w_{1k} are adjustments to the slope of the continuous concentration variable *conc*.

The data were acquired with a mass cytometer which provides several thousand measurements for each category. A summary plot from the mass cytometer for one experiment is below:

Figure 2: ^{158}Gd signal on major leukocyte subsets



For this experiment we asked the scientific question: Does the cell signal change with contrast agent, for a given concentration?

Interpretation of this data with a rigorous statistical model posed two major challenges:

- Mass cytometry data is rarely handled in its raw form, which contains millions of samples. Rather, the cytometer outputs four summary statistics: 10% quantile, mean, median, and 90% quantile. To model uncertainty, the underlying distribution of each experimental condition had to be estimated.
- Once the distributions for each condition were estimated, iterative model-building was required to estimate impact of contrast agent.

Data exploration

Data were cleaned and converted to tall format:

```
source('masscyto_clean_gather_dataNEW.R')
mass_cyto <- read.xls(xls="experimental_data.xlsx",
                      header=FALSE, skip=3, nrows=30)[,-(5:10)]
mass_cyto_tall <- clean_gather_data(mass_cyto)
#str(mass_cyto_tall)
#head(mass_cyto_tall)
#summary(mass_cyto_tall)
```

The research question: Does the cell signal change with contrast agent, for a given concentration?

Eric wrote:

"The main experimental question is whether certain of the contrast agents, which are safer, produce signal comparable to other contrast agents which are more dangerous. However, we do also want to investigate whether this statement holds at varying concentrations."

To me (SV), this sounds like we need to know if there is an interaction between ContrastAgent and concentration, but the crucial issue is: is there a specific expectation for how ContrastAgent will affect the dependent variable value? For now, I will assume that there is a natural ordering in level of danger:

- Magnevist is less dangerous than Dotarem
- Dotarem is less dangerous than Gadevist

These comparisons can be changed to reflect reality (if the above ordering is incorrect).

We will use sliding contrasts to reflect the above ordering:

```
library(MASS)
## reversing the signs by multiplying by -1:
comparisons<- -1*round(ginv(contr.sdif(3)))
rownames(comparisons)<-c("MvsD","DvsG")
comparisons
```

```
##      [,1] [,2] [,3]
## MvsD    1   -1    0
## DvsG    0    1   -1
```

Also, we would have to remove control from that particular analysis, because controls have only one level of the concentrations. But the effect of control can be estimated and used as a baseline (to-do).

Experiment needs to be modeled as a random effect because each experiment is actually a patient.

Computing standard deviation of the dependent variable "value" in preparation for measurement error modeling

We can just take logs on the mean and the quantile measurements and then get the measurement error on the log scale:

```
x<-rlnorm(1000,meanlog=1.2,sdlog=1)
qnorm(0.05,mean=1.2,sd=1) # 5% quantile
```

```
## [1] -0.4448536
```

```
qnorm(0.95,mean=1.2,sd=1) # 95% quantile
```

```
## [1] 2.844854
```

```
mean(log(x)) ## recovers MLE of lognormal
```

```
## [1] 1.196182
```

```
sd(log(x))   ## recovers sd of lognormal
```

```
## [1] 1.000142
```

```
log(quantile(x,prob=c(0.05,0.95))) ## recovers 5th and 95th quantile
```

```
##         5%        95%
## -0.4919921  2.7718694
```

So, given the upper 95th percentile, we take the distance between this percentile value and the sample mean, and divide that by 1.64 or so to get an approximate standard deviation. We will use this in the measurement error model.

Modeling effect of concentration

In preparation for the measurement error model, we first
prepare the data so that there is a column for the uncertainty of each mean value in each row of the data frame:

```
## extract means:
means<-subset(mass_cyto_tall,MeasurementType=="Mean")
## extract lower quantiles:
qlow<-subset(mass_cyto_tall,MeasurementType=="pct_05")
## extract upper quantiles:
qhigh<-subset(mass_cyto_tall,MeasurementType=="pct_95")

## log scale difference between upper percentile and mean:
d<-log(qhigh$value)-log(means$value)

means$SD<-d/1.64

## center concentration:
means$cconc<-scale(as.numeric(as.character(means$Concentration)),scale=FALSE)

#head(means)

## rename cell type as numerical values:
means$typ<-as.integer(as.factor(means$CellType))
```

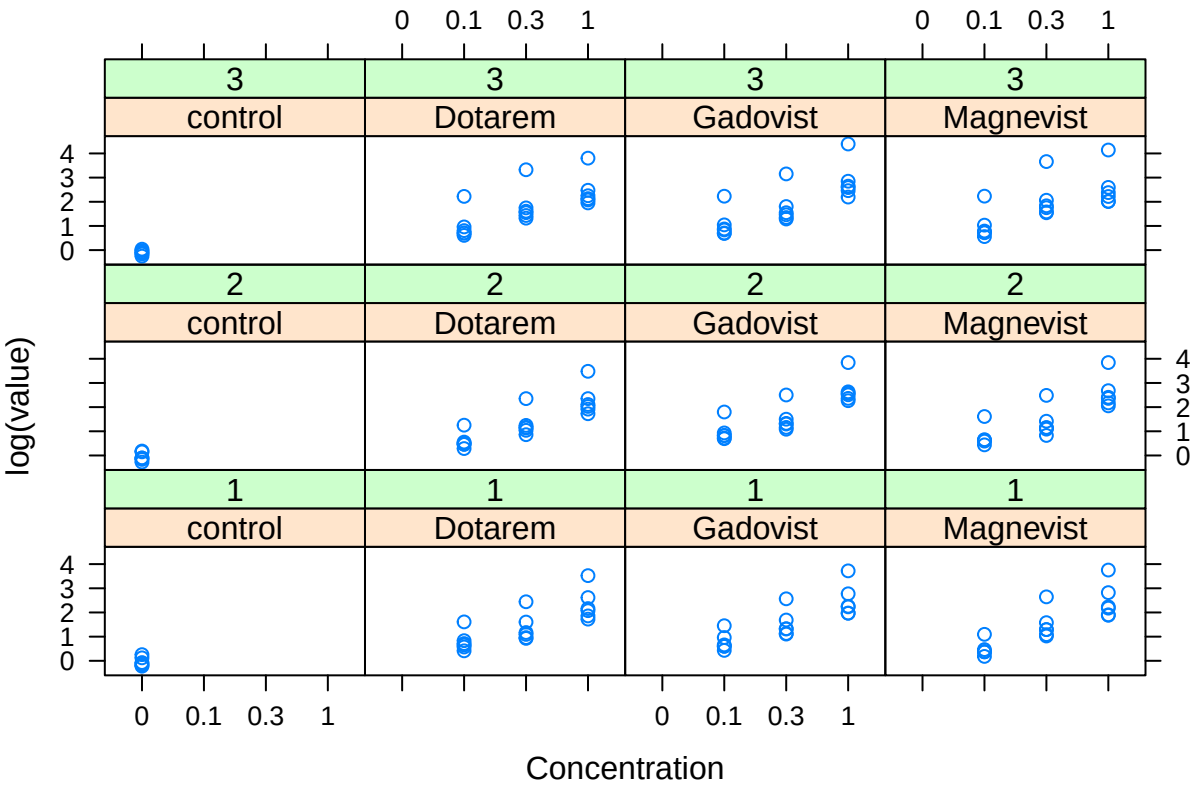
```
## needed later for measurement error model written in Stan:
dat<-list(cconc=as.vector(means$cconc),
     logvalue=log(means$value),
     SD=means$SD,
     typ=means$typ,
     N = dim(means)[1],
     J = length(unique(means$typ))
     )

str(dat)
```

```
## List of 6
##  $ cconc   : num [1:180] -0.42 -0.32 -0.12 0.58 -0.32 -0.12 0.58 -0.32 -0.12 0.58 ...
##  $ logvalue: num [1:180] -0.223 0.464 1.579 2.821 0.833 ...
##  $ SD      : num [1:180] 0.527 0.625 0.611 0.621 0.627 ...
##  $ typ     : int [1:180] 3 3 3 3 3 3 3 3 3 3 ...
##  $ N       : int 180
##  $ J       : int 6
```

The first step is to visualize the effect of concentration by contrast agent and experiment, pooling data from all cell types:

```
library(lattice)
xyplot(log(value)~Concentration|ContrastAgent+Experiment,means)
```



We start by just modeling the effect of concentration on log(value). Concentration is centered and scaled to have standard deviation 1; this has the (small) advantage that the intercept now has the interpretation that it reflects the predicted log(value) when concentration is the average value.

Here is the standard hierarchical linear model of effect of concentration on log value. Cell Types and Experiment are treated as random effects. Here I am making the simplifying assumption that Cell Types and

Experiment are independent—is this reasonable?

```
m1<-lmer(log(value)~cconc+(1+cconc|Experiment)+
           (1+cconc|CellType),
         subset(means,ContrastAgent!="control"))
```

```
## Warning: 'rBind' is deprecated.
##  Since R version 3.2.0, base's rbind() should work fine with S4 objects
```

```
print(summary(m1))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(value) ~ cconc + (1 + cconc | Experiment) + (1 + cconc |
##     CellType)
##    Data: subset(means, ContrastAgent != "control")
##
## REML criterion at convergence: 54.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.9462 -0.8274  0.0046  0.6786  4.0490
##
## Random effects:
##  Groups     Name        Variance Std.Dev. Corr
##  CellType   (Intercept) 0.32794  0.5727
##             cconc       0.04512  0.2124   1.00
##  Experiment (Intercept) 0.03271  0.1809
##             cconc       0.01808  0.1345   -1.00
##  Residual               0.06430  0.2536
## Number of obs: 162, groups:  CellType, 6; Experiment, 3
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   1.5599     0.2568   6.074
## cconc         1.7435     0.1273  13.693
##
## Correlation of Fixed Effects:
##       (Intr)
## cconc 0.368
```

The above frequentist model has some problems. In particular, it cannot estimate the correlations (notice that these are ± 1). I would therefore fit a simpler frequentist model, assuming no correlations between the varying intercepts and slopes:

```
m2<-lmer(log(value)~cconc+(1+cconc||Experiment)+
           (1+cconc||CellType),
         subset(means,ContrastAgent!="control"))

print(summary(m2))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## log(value) ~ cconc + ((1 | Experiment) + (0 + cconc | Experiment)) +
##     ((1 | CellType) + (0 + cconc | CellType))
##    Data: subset(means, ContrastAgent != "control")
##
## REML criterion at convergence: 67.1
```

```
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1918 -0.7972  0.0205  0.6100  3.9187
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  CellType     cconc       0.03443  0.1855
##  CellType.1   (Intercept) 0.33146  0.5757
##  Experiment   cconc       0.01184  0.1088
##  Experiment.1 (Intercept) 0.03170  0.1780
##  Residual                 0.06645  0.2578
## Number of obs: 162, groups:  CellType, 6; Experiment, 3
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   1.5599     0.2573   6.062
## cconc         1.7435     0.1115  15.632
##
## Correlation of Fixed Effects:
##       (Intr)
## cconc -0.004
```

The full model above (m1) can be fit in Stan quite easily. As priors for the fixed effects we choose Cauchy(0,5), to allow them to have extreme values, and for the others, we choose Normal(0,1), which seems reasonable as the dependent variable is on the log scale and the standard deviations are all going to be less than 1. The correlation parameters have as priors the LKJ(2) prior, which downweights the extreme values ± 1.

```
priors<-c(set_prior("cauchy(0,5)",
                    class = "b"),
                      set_prior("cauchy(0,5)", class = "b",coef="cconc"),set_prior("normal(0,1)", class
          set_prior("lkj(2)", class = "cor"))

m1brm<-brm(formula = log(value) ~ cconc+(1+cconc|Experiment)+
           (1+cconc|CellType),
             data = subset(means,ContrastAgent!="control"),
             family = gaussian(),
             prior = priors,
             warmup = 1000,
             iter = 2000,
             chains = 4,
             control = list(adapt_delta = 0.99,max_treedepth=15))
```

```
## Compiling the C++ model
```

```
## Start sampling
```

```
print(summary(m1brm))
```

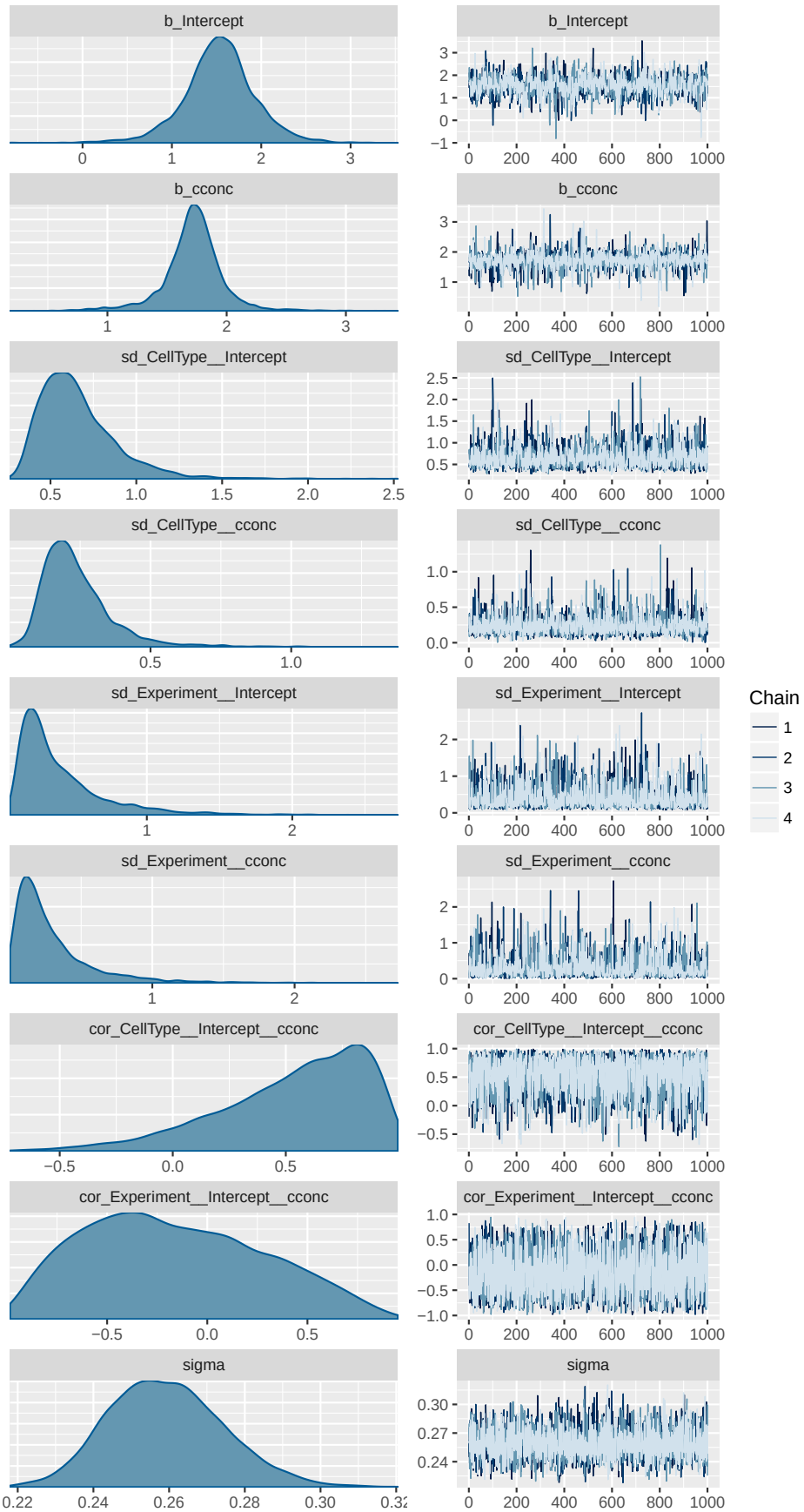
```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: log(value) ~ cconc + (1 + cconc | Experiment) + (1 + cconc | CellType)
##    Data: subset(means, ContrastAgent != "control") (Number of observations: 162)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##     ICs: LOO = NA; WAIC = NA; R2 = NA
```

```
##
## Group-Level Effects:
## ~CellType (Number of levels: 6)
##                     Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sd(Intercept)           0.66      0.23     0.36     1.22       1369 1.00
## sd(cconc)               0.24      0.13     0.08     0.56       1648 1.00
## cor(Intercept,cconc)    0.53      0.32    -0.24     0.95       2952 1.00
##
## ~Experiment (Number of levels: 3)
##                     Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sd(Intercept)           0.40      0.31     0.11     1.26       1417 1.00
## sd(cconc)               0.29      0.29     0.02     1.11       1350 1.00
## cor(Intercept,cconc)   -0.16      0.44    -0.87     0.71       4000 1.00
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept     1.53      0.40     0.71     2.33       1192 1.00
## cconc         1.72      0.26     1.14     2.23       1605 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     0.26      0.02     0.23     0.29       4000 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Notice some important differences between the frequentist model (FM) and the Bayesian model (BM): The FM fails to estimate the intercept-slope correlations; the BM is able to estimate the correlations, but also shows very high uncertainty in the posterior distributions of these correlations. The BM also provides 95% credible intervals for each parameter; the FM only does this for the fixed effects parameters (the intercept and the effect of concentration).

Plotting the posteriors:

```
plot(m1brm,N=9)
```


b_Intercept
b_cconc
sd_CellType__Intercept
sd_CellType__cconc
sd_Experiment__Intercept
sd_Experiment__cconc
cor_CellType__Intercept__cconc
cor_Experiment__Intercept__cconc
sigma
Chain
1
2
3
4

Modeling the effect of contrast agent and their interactions with concentration

Because of the complexity of the model, in the frequentist model I will not attempt to fit correlations in the frequentist model:

```
## hand-coded sliding contrasts:
means$MvsD<-ifelse(means$ContrastAgent=="Magnevist",1,
           ifelse(means$ContrastAgent=="Dotarem",-1,0))
means$DvsG<-ifelse(means$ContrastAgent=="Dotarem",1,
           ifelse(means$ContrastAgent=="Gadovist",-1,0))

m3<-lmer(log(value)~ cconc +MvsD+DvsG+cconc:MvsD + cconc:DvsG+(1+cconc +MvsD+DvsG+cconc:MvsD + cconc:Dvs
                 subset(means,ContrastAgent!="control"))
print(summary(m3))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(value) ~ cconc + MvsD + DvsG + cconc:MvsD + cconc:DvsG +
##     ((1 | Experiment) + (0 + cconc | Experiment) + (0 + MvsD |
##         Experiment) + (0 + DvsG | Experiment) + (0 + cconc:MvsD |
##         Experiment) + (0 + cconc:DvsG | Experiment)) + ((1 |
##     CellType) + (0 + cconc | CellType) + (0 + MvsD | CellType) +
##     (0 + DvsG | CellType) + (0 + cconc:MvsD | CellType) + (0 +
##     cconc:DvsG | CellType))
##    Data: subset(means, ContrastAgent != "control")
##
## REML criterion at convergence: 64.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2662 -0.5969 -0.0337  0.5395  4.2093
##
## Random effects:
##  Groups       Name        Variance  Std.Dev.
##  CellType     cconc:DvsG  3.494e-18 1.869e-09
##  CellType.1   cconc:MvsD  0.000e+00 0.000e+00
##  CellType.2   DvsG        0.000e+00 0.000e+00
##  CellType.3   MvsD        0.000e+00 0.000e+00
##  CellType.4   cconc       3.636e-02 1.907e-01
##  CellType.5   (Intercept) 3.313e-01 5.756e-01
##  Experiment   cconc:DvsG  0.000e+00 0.000e+00
##  Experiment.1 cconc:MvsD  5.535e-16 2.353e-08
##  Experiment.2 DvsG        2.032e-03 4.508e-02
##  Experiment.3 MvsD        2.067e-04 1.438e-02
##  Experiment.4 cconc       1.291e-02 1.136e-01
##  Experiment.5 (Intercept) 3.197e-02 1.788e-01
##  Residual                 5.841e-02 2.417e-01
## Number of obs: 162, groups:  CellType, 6; Experiment, 3
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.559926   0.257373   6.061
## cconc        1.743527   0.113075  15.419
## MvsD        -0.003335   0.028293  -0.118
## DvsG        -0.089993   0.037536  -2.398
## cconc:MvsD   0.059804   0.069592   0.859
```

```
## cconc:DvsG  -0.093508   0.069592  -1.344
##
## Correlation of Fixed Effects:
##            (Intr) cconc  MvsD   DvsG   ccn:MD
## cconc      -0.004
## MvsD        0.000  0.000
## DvsG        0.000  0.000  0.344
## cconc:MvsD  0.000  0.000 -0.115 -0.043
## cconc:DvsG  0.000  0.000 -0.057 -0.087  0.500
```

The vertical bars in the random effects are expressing the assumption that the correlation between intercepts and slopes in random effects is assumed to be 0.

Stan version implemented in brms. Here we are fitting the maximal model, because we can (due to the regularization that the priors provide on the parameters).

```
m3brm<-brm(formula = log(value)~ cconc +MvsD+DvsG+cconc:MvsD + cconc:DvsG+
             (1+cconc +MvsD+DvsG+cconc:MvsD + cconc:DvsG|Experiment)+
             (1+cconc +MvsD+DvsG+cconc:MvsD + cconc:DvsG|CellType),
           data = subset(means,ContrastAgent!="control"),
           family = gaussian(),
           prior = priors,
           warmup = 1000,
           iter = 2000,
           chains = 4,
           control = list(adapt_delta = 0.99,max_treedepth=15))
```

```
## Compiling the C++ model
```

```
## Start sampling
```

```
## Warning: There were 1 divergent transitions after warmup. Increasing adapt_delta above 0.99 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
print(summary(m3brm))
```

```
## Warning: There were 1 divergent transitions after warmup. Increasing adapt_delta above 0.99 may help
## See http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: log(value) ~ cconc + MvsD + DvsG + cconc:MvsD + cconc:DvsG + (1 + cconc + MvsD + DvsG + cco
##    Data: subset(means, ContrastAgent != "control") (Number of observations: 162)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##     ICs: LOO = NA; WAIC = NA; R2 = NA
##
## Group-Level Effects:
## ~CellType (Number of levels: 6)
##                             Estimate Est.Error l-95% CI u-95% CI Eff.Sample
## sd(Intercept)                   0.70      0.25     0.38     1.33       1451
## sd(cconc)                       0.25      0.13     0.07     0.57       1881
## sd(MvsD)                        0.03      0.03     0.00     0.12       4000
## sd(DvsG)                        0.03      0.03     0.00     0.11       4000
## sd(cconc:MvsD)                  0.08      0.08     0.00     0.30       4000
## sd(cconc:DvsG)                  0.08      0.08     0.00     0.29       4000
```

```
## cor(Intercept,cconc)           0.33      0.30    -0.31     0.81       4000
## cor(Intercept,MvsD)            0.05      0.33    -0.59     0.67       4000
## cor(cconc,MvsD)                0.04      0.33    -0.60     0.64       4000
## cor(Intercept,DvsG)            0.00      0.33    -0.62     0.63       4000
## cor(cconc,DvsG)               -0.00      0.33    -0.63     0.62       4000
## cor(MvsD,DvsG)                 0.03      0.34    -0.62     0.65       4000
## cor(Intercept,cconc:MvsD)      0.03      0.34    -0.61     0.66       4000
## cor(cconc,cconc:MvsD)          0.03      0.33    -0.60     0.67       4000
## cor(MvsD,cconc:MvsD)           0.02      0.34    -0.62     0.66       4000
## cor(DvsG,cconc:MvsD)           0.00      0.33    -0.61     0.64       4000
## cor(Intercept,cconc:DvsG)     -0.00      0.33    -0.63     0.61       4000
## cor(cconc,cconc:DvsG)          0.00      0.33    -0.62     0.62       4000
## cor(MvsD,cconc:DvsG)           0.00      0.34    -0.64     0.64       4000
## cor(DvsG,cconc:DvsG)          -0.01      0.34    -0.64     0.64       4000
## cor(cconc:MvsD,cconc:DvsG)     0.03      0.33    -0.61     0.65       3270
##                            Rhat
## sd(Intercept)              1.00
## sd(cconc)                  1.00
## sd(MvsD)                   1.00
## sd(DvsG)                   1.00
## sd(cconc:MvsD)             1.00
## sd(cconc:DvsG)             1.00
## cor(Intercept,cconc)       1.00
## cor(Intercept,MvsD)        1.00
## cor(cconc,MvsD)            1.00
## cor(Intercept,DvsG)        1.00
## cor(cconc,DvsG)            1.00
## cor(MvsD,DvsG)             1.00
## cor(Intercept,cconc:MvsD)  1.00
## cor(cconc,cconc:MvsD)      1.00
## cor(MvsD,cconc:MvsD)       1.00
## cor(DvsG,cconc:MvsD)       1.00
## cor(Intercept,cconc:DvsG)  1.00
## cor(cconc,cconc:DvsG)      1.00
## cor(MvsD,cconc:DvsG)       1.00
## cor(DvsG,cconc:DvsG)       1.00
## cor(cconc:MvsD,cconc:DvsG) 1.00
##
## ~Experiment (Number of levels: 3)
##                            Estimate Est.Error l-95% CI u-95% CI Eff.Sample
## sd(Intercept)                  0.42      0.31     0.11     1.27       1772
## sd(cconc)                      0.33      0.29     0.03     1.14       1768
## sd(MvsD)                       0.16      0.21     0.00     0.79        957
## sd(DvsG)                       0.17      0.21     0.01     0.78       1185
## sd(cconc:MvsD)                 0.30      0.32     0.01     1.19       1231
## sd(cconc:DvsG)                 0.31      0.32     0.01     1.19       1591
## cor(Intercept,cconc)          -0.10      0.33    -0.69     0.57       4000
## cor(Intercept,MvsD)            0.04      0.33    -0.60     0.65       4000
## cor(cconc,MvsD)               -0.02      0.33    -0.64     0.61       4000
## cor(Intercept,DvsG)            0.05      0.33    -0.57     0.66       4000
## cor(cconc,DvsG)               -0.05      0.34    -0.68     0.61       4000
## cor(MvsD,DvsG)                -0.00      0.34    -0.64     0.63       4000
## cor(Intercept,cconc:MvsD)     -0.07      0.33    -0.68     0.56       4000
## cor(cconc,cconc:MvsD)          0.04      0.33    -0.59     0.65       4000
```

```
## cor(MvsD,cconc:MvsD)          -0.03      0.35    -0.69     0.62       4000
## cor(DvsG,cconc:MvsD)          -0.01      0.34    -0.65     0.64       4000
## cor(Intercept,cconc:DvsG)     -0.08      0.34    -0.69     0.57       4000
## cor(cconc,cconc:DvsG)          0.04      0.33    -0.60     0.67       4000
## cor(MvsD,cconc:DvsG)          -0.03      0.35    -0.69     0.63       4000
## cor(DvsG,cconc:DvsG)          -0.02      0.34    -0.68     0.63       4000
## cor(cconc:MvsD,cconc:DvsG)     0.05      0.33    -0.58     0.67       4000
##                            Rhat
## sd(Intercept)              1.00
## sd(cconc)                  1.00
## sd(MvsD)                   1.01
## sd(DvsG)                   1.00
## sd(cconc:MvsD)             1.01
## sd(cconc:DvsG)             1.00
## cor(Intercept,cconc)       1.00
## cor(Intercept,MvsD)        1.00
## cor(cconc,MvsD)            1.00
## cor(Intercept,DvsG)        1.00
## cor(cconc,DvsG)            1.00
## cor(MvsD,DvsG)             1.00
## cor(Intercept,cconc:MvsD)  1.00
## cor(cconc,cconc:MvsD)      1.00
## cor(MvsD,cconc:MvsD)       1.00
## cor(DvsG,cconc:MvsD)       1.00
## cor(Intercept,cconc:DvsG)  1.00
## cor(cconc,cconc:DvsG)      1.00
## cor(MvsD,cconc:DvsG)       1.00
## cor(DvsG,cconc:DvsG)       1.00
## cor(cconc:MvsD,cconc:DvsG) 1.00
##
## Population-Level Effects:
##            Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept      1.55      0.44     0.61     2.38       1047 1.00
## cconc          1.73      0.27     1.18     2.28       1669 1.00
## MvsD           0.00      0.15    -0.28     0.31        826 1.00
## DvsG          -0.09      0.15    -0.35     0.22       1288 1.00
## cconc:MvsD     0.05      0.27    -0.53     0.54        966 1.00
## cconc:DvsG    -0.09      0.27    -0.60     0.50       1254 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     0.25      0.02     0.22     0.28       4000 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Measurement error on log(value): the effect of concentration

I switch to Stan instead of brms because this way I am very sure of the underlying generative process. With brms I am not sure if it is working correctly (I am working with the developer to fix the bug in brms).

We start with a varying intercepts-only model.

```
Mconc <- stan(file = "StanModels/MeasErrVarInt.stan",
                    data = dat,
                    iter = 4000,
                    chains = 4,
              control = list(adapt_delta = 0.99,
max_treedepth = 15))
```

```
## In file included from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config.hp
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/math/tool
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/src/st
##                  from file43f9432e269e.cpp:8:
## /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config/compiler/gcc.hpp:186:0:
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
```

```
## Warning: There were 16 divergent transitions after warmup. Increasing adapt_delta above 0.99 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: There were 14 transitions after warmup that exceeded the maximum treedepth. Increase max_tre
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. Se
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

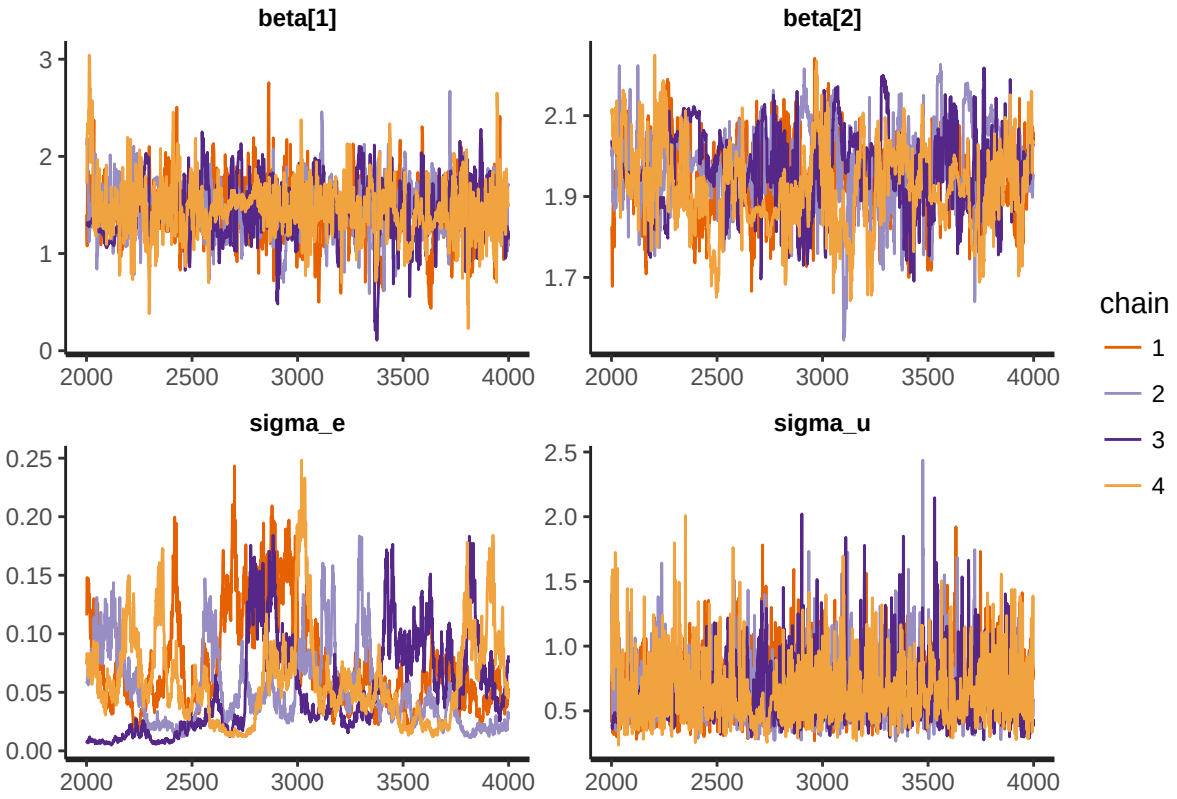
```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
print(Mconc,pars=c("beta","sigma_e","sigma_u"))
```

```
## Inference for Stan model: MeasErrVarInt.
## 4 chains, each with iter=4000; warmup=2000; thin=1;
## post-warmup draws per chain=2000, total post-warmup draws=8000.
##
##         mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## beta[1] 1.47    0.01 0.28 0.87 1.30 1.48 1.64  2.01   520 1.02
## beta[2] 1.95    0.01 0.10 1.75 1.89 1.95 2.02  2.14   177 1.03
## sigma_e 0.06    0.01 0.04 0.01 0.03 0.05 0.08  0.16    40 1.09
## sigma_u 0.66    0.01 0.23 0.35 0.50 0.62 0.78  1.24  1145 1.00
##
## Samples were drawn using NUTS(diag_e) at Mon May 14 17:23:30 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Some minor sampling problems are apparent in the trace plots, but these can be fixed quickly with reparameterization (in any case, they become irrelevant in the maximal model below).

```
traceplot(Mconc,pars=c("beta","sigma_e","sigma_u"))
```



Next, we fit a maximal model with varying intercepts for cell type:

```
MconcMaximal <- stan(file = "StanModels/MEVarIntMaximal.stan",
                     data = dat,
                     iter = 2000,
                     chains = 4,
                control = list(adapt_delta = 0.99,
max_treedepth = 15))
```

```
## In file included from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config.hp
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/math/tool
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/src/st
##                  from file43f97c33bc06.cpp:8:
## /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config/compiler/gcc.hpp:186:0:
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition

## Warning: There were 31 divergent transitions after warmup. Increasing adapt_delta above 0.99 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. Se
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

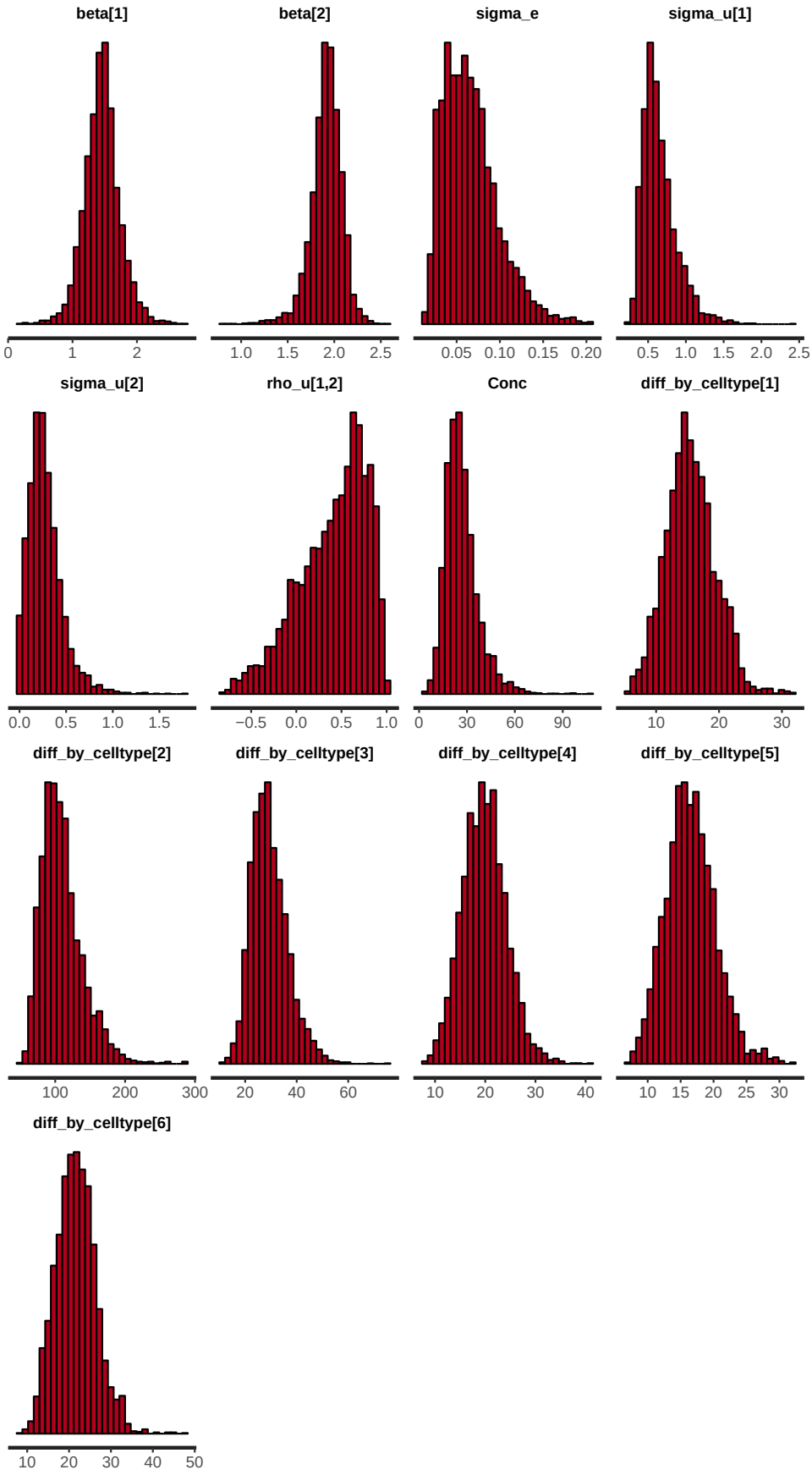
```
print(MconcMaximal,pars=c("beta","sigma_e","sigma_u","Conc","diff_by_celltype"))
```

```
## Inference for Stan model: MEVarIntMaximal.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                        mean se_mean    sd  2.5%   25%    50%    75%  97.5%
## beta[1]                1.46    0.02  0.28  0.92  1.29   1.46   1.61   2.04
## beta[2]                1.91    0.01  0.17  1.54  1.83   1.93   2.02   2.21
## sigma_e                0.07    0.01  0.03  0.02  0.04   0.06   0.08   0.14
## sigma_u[1]             0.65    0.01  0.23  0.35  0.49   0.61   0.76   1.22
## sigma_u[2]             0.27    0.01  0.18  0.02  0.15   0.24   0.36   0.73
## Conc                  26.55    0.57 10.31 11.31 19.61  24.88  31.45  51.78
## diff_by_celltype[1]   15.50    0.22  3.92  8.44 12.87  15.29  17.97  23.14
## diff_by_celltype[2]  109.58    1.75 29.51 67.21 88.86 104.89 124.52 178.63
## diff_by_celltype[3]   29.44    0.48  7.09 18.25 24.42  28.56  33.63  45.92
## diff_by_celltype[4]   19.80    0.27  4.33 11.58 16.85  19.70  22.55  28.50
## diff_by_celltype[5]   16.53    0.26  3.75  9.78 14.05  16.31  18.85  24.42
## diff_by_celltype[6]   21.63    0.32  4.75 13.15 18.38  21.52  24.55  31.98
##                      n_eff Rhat
## beta[1]                339 1.01
## beta[2]                255 1.01
## sigma_e                 34 1.25
## sigma_u[1]             602 1.00
## sigma_u[2]             185 1.01
## Conc                   330 1.01
## diff_by_celltype[1]    325 1.01
## diff_by_celltype[2]    284 1.01
## diff_by_celltype[3]    219 1.01
## diff_by_celltype[4]    248 1.00
## diff_by_celltype[5]    205 1.01
## diff_by_celltype[6]    217 1.02
##
## Samples were drawn using NUTS(diag_e) at Mon May 14 17:25:11 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

We can plot the results as shown below:

```
stan_hist(MconcMaximal,pars=c("beta","sigma_e","sigma_u","rho_u[1,2]","Conc","diff_by_celltype"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


beta[1]
beta[2]
sigma_e
sigma_u[1]
sigma_u[2]
rho_u[1,2]
Conc
diff_by_celltype[1]
diff_by_celltype[2]
diff_by_celltype[3]
diff_by_celltype[4]
diff_by_celltype[5]
diff_by_celltype[6]

The effects by cell type are also shown, and the mapping from cell type by number id (1-6) to actual cell type names can be inferred from:

```
xtabs(~typ+CellType,means)
```

```
##    CellType
## typ B cells Monocytes Neutrophils NK cells_1 NK cells_2 T cells
##   1      30         0           0          0          0       0
##   2       0        30           0          0          0       0
##   3       0         0          30          0          0       0
##   4       0         0           0         30          0       0
##   5       0         0           0          0         30       0
##   6       0         0           0          0          0      30
```

Also, notice that the correlation parameter has a *lot* of uncertainty; basically any value between -1 and +1 is possible, and the extreme values are downweighted due to the LKJ prior on the correlation.

Next, we will fit the full measurement error model, with main effects and interactions of concentration and contrast agent.

```
means$cconcMvsD <- means$cconc*means$MvsD
means$cconcDvsG <- means$cconc*means$DvsG

dat2<-list(cconc=as.vector(means$cconc),
     logvalue=log(means$value),
     MvsD=means$MvsD,
     DvsG=means$DvsG,
     cconcMvsD = as.vector(means$cconcMvsD),
     cconcDvsG = as.vector(means$cconcDvsG),
     SD=means$SD,
     typ=means$typ,
     N = dim(means)[1],
     J = length(unique(means$typ))
     )
```

```
MconcMaximal2 <- stan(file = "StanModels/MEVarIntMaximal2.stan",
                      data = dat2,
                      iter = 2000,
                      chains = 4,
              control = list(adapt_delta = 0.99,
max_treedepth = 15))
```

```
## In file included from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config.hp
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/math/tool
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/m
##                  from /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/src/st
##                  from file43f95d8a504e.cpp:8:
## /home/ericbarnhill/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config/compiler/gcc.hpp:186:0:
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
```

```
## Warning: There were 53 divergent transitions after warmup. Increasing adapt_delta above 0.99 may help
```

```
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. S
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

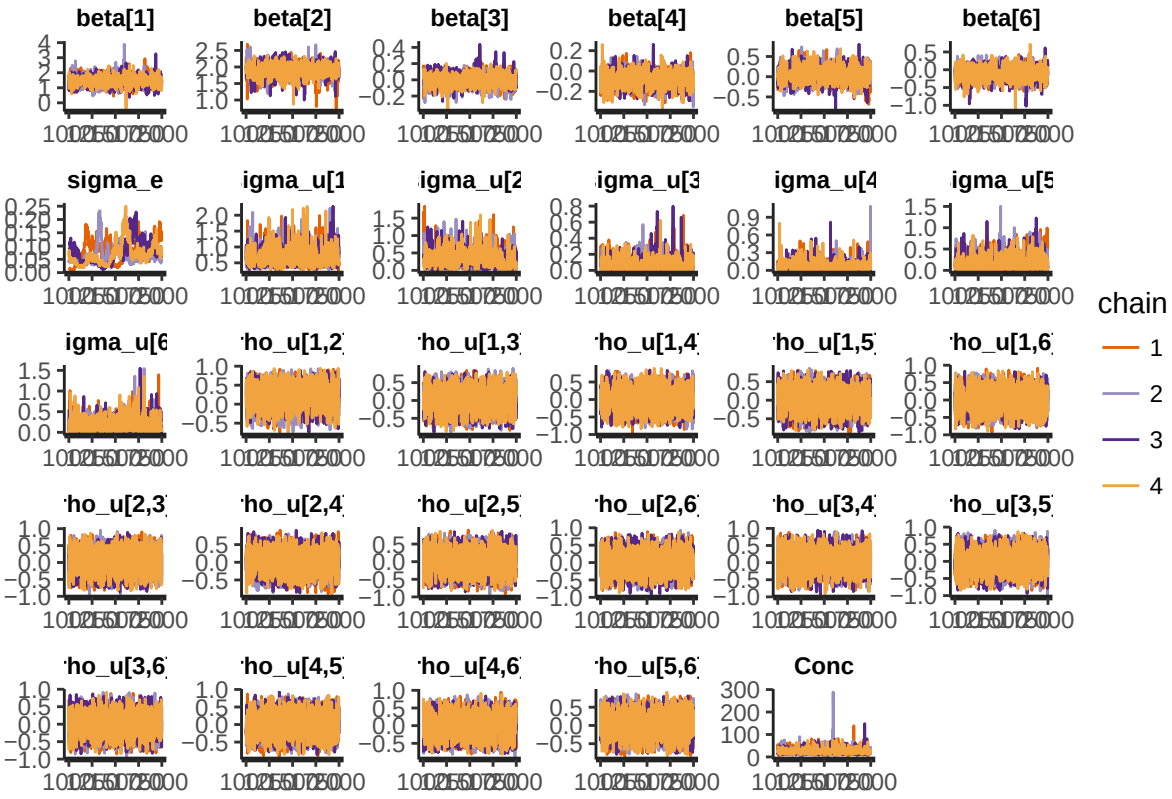
```
print(MconcMaximal2,pars=c("beta","sigma_e","sigma_u","Conc","rho_u"))
```

```
## Inference for Stan model: MEVarIntMaximal2.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##              mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1]      1.47    0.01  0.31  0.88  1.28  1.46  1.64  2.11  1658 1.00
## beta[2]      1.91    0.00  0.18  1.53  1.81  1.91  2.02  2.23  1909 1.00
## beta[3]      0.00    0.00  0.07 -0.13 -0.05  0.00  0.04  0.14  1016 1.00
## beta[4]     -0.08    0.00  0.07 -0.23 -0.13 -0.08 -0.04  0.05  1088 1.00
## beta[5]      0.06    0.00  0.18 -0.29 -0.06  0.05  0.17  0.40  1478 1.00
## beta[6]     -0.10    0.00  0.18 -0.45 -0.21 -0.10  0.01  0.23  1528 1.00
## sigma_e      0.07    0.01  0.04  0.02  0.04  0.06  0.09  0.17    38 1.16
## sigma_u[1]   0.68    0.01  0.25  0.36  0.51  0.63  0.80  1.31  1747 1.00
## sigma_u[2]   0.30    0.01  0.20  0.03  0.17  0.27  0.40  0.81   727 1.01
## sigma_u[3]   0.07    0.00  0.07  0.00  0.02  0.05  0.09  0.24  1399 1.00
## sigma_u[4]   0.07    0.00  0.07  0.00  0.02  0.05  0.09  0.23  1278 1.00
## sigma_u[5]   0.17    0.00  0.15  0.01  0.06  0.13  0.23  0.55  1270 1.00
## sigma_u[6]   0.17    0.00  0.15  0.01  0.06  0.13  0.23  0.54  1337 1.01
## Conc        26.98    0.37 12.82 11.06 19.43 25.06 31.89 55.34  1181 1.00
## rho_u[1,1]   1.00    0.00  0.00  1.00  1.00  1.00  1.00  1.00  4000  NaN
## rho_u[1,2]   0.25    0.00  0.31 -0.43  0.05  0.28  0.48  0.77  4000 1.00
## rho_u[1,3]   0.03    0.01  0.33 -0.61 -0.21  0.03  0.27  0.64  4000 1.00
## rho_u[1,4]   0.00    0.01  0.33 -0.62 -0.26 -0.01  0.24  0.62  4000 1.00
## rho_u[1,5]   0.01    0.01  0.33 -0.62 -0.22  0.02  0.25  0.63  4000 1.00
## rho_u[1,6]  -0.01    0.01  0.33 -0.63 -0.25 -0.01  0.24  0.61  4000 1.00
## rho_u[2,1]   0.25    0.00  0.31 -0.43  0.05  0.28  0.48  0.77  4000 1.00
## rho_u[2,2]   1.00    0.00  0.00  1.00  1.00  1.00  1.00  1.00  3627 1.00
## rho_u[2,3]   0.01    0.01  0.34 -0.65 -0.23  0.01  0.26  0.65  4000 1.00
## rho_u[2,4]   0.01    0.01  0.33 -0.63 -0.24  0.00  0.25  0.63  4000 1.00
## rho_u[2,5]   0.01    0.01  0.34 -0.63 -0.24  0.02  0.26  0.64  4000 1.00
## rho_u[2,6]   0.01    0.01  0.34 -0.62 -0.24  0.00  0.25  0.64  4000 1.00
## rho_u[3,1]   0.03    0.01  0.33 -0.61 -0.21  0.03  0.27  0.64  4000 1.00
## rho_u[3,2]   0.01    0.01  0.34 -0.65 -0.23  0.01  0.26  0.65  4000 1.00
## rho_u[3,3]   1.00    0.00  0.00  1.00  1.00  1.00  1.00  1.00  4000 1.00
## rho_u[3,4]   0.04    0.01  0.34 -0.61 -0.21  0.05  0.28  0.67  3207 1.00
## rho_u[3,5]  -0.01    0.01  0.33 -0.63 -0.25 -0.01  0.24  0.65  4000 1.00
## rho_u[3,6]   0.00    0.01  0.34 -0.64 -0.24  0.00  0.24  0.65  4000 1.00
## rho_u[4,1]   0.00    0.01  0.33 -0.62 -0.26 -0.01  0.24  0.62  4000 1.00
## rho_u[4,2]   0.01    0.01  0.33 -0.63 -0.24  0.00  0.25  0.63  4000 1.00
## rho_u[4,3]   0.04    0.01  0.34 -0.61 -0.21  0.05  0.28  0.67  3207 1.00
## rho_u[4,4]   1.00    0.00  0.00  1.00  1.00  1.00  1.00  1.00  4000 1.00
## rho_u[4,5]   0.00    0.01  0.34 -0.64 -0.25  0.01  0.25  0.63  3431 1.00
## rho_u[4,6]  -0.01    0.01  0.33 -0.64 -0.25 -0.02  0.23  0.63  2760 1.00
## rho_u[5,1]   0.01    0.01  0.33 -0.62 -0.22  0.02  0.25  0.63  4000 1.00
## rho_u[5,2]   0.01    0.01  0.34 -0.63 -0.24  0.02  0.26  0.64  4000 1.00
## rho_u[5,3]  -0.01    0.01  0.33 -0.63 -0.25 -0.01  0.24  0.65  4000 1.00
## rho_u[5,4]   0.00    0.01  0.34 -0.64 -0.25  0.01  0.25  0.63  3431 1.00
```

```
## rho_u[5,5]  1.00    0.00  0.00  1.00  1.00  1.00  1.00  1.00  3218 1.00
## rho_u[5,6]  0.03    0.01  0.34 -0.64 -0.22  0.04  0.27  0.63  2724 1.00
## rho_u[6,1] -0.01    0.01  0.33 -0.63 -0.25 -0.01  0.24  0.61  4000 1.00
## rho_u[6,2]  0.01    0.01  0.34 -0.62 -0.24  0.00  0.25  0.64  4000 1.00
## rho_u[6,3]  0.00    0.01  0.34 -0.64 -0.24  0.00  0.24  0.65  4000 1.00
## rho_u[6,4] -0.01    0.01  0.33 -0.64 -0.25 -0.02  0.23  0.63  2760 1.00
## rho_u[6,5]  0.03    0.01  0.34 -0.64 -0.22  0.04  0.27  0.63  2724 1.00
## rho_u[6,6]  1.00    0.00  0.00  1.00  1.00  1.00  1.00  1.00  3848 1.00
##
## Samples were drawn using NUTS(diag_e) at Mon May 14 17:42:20 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

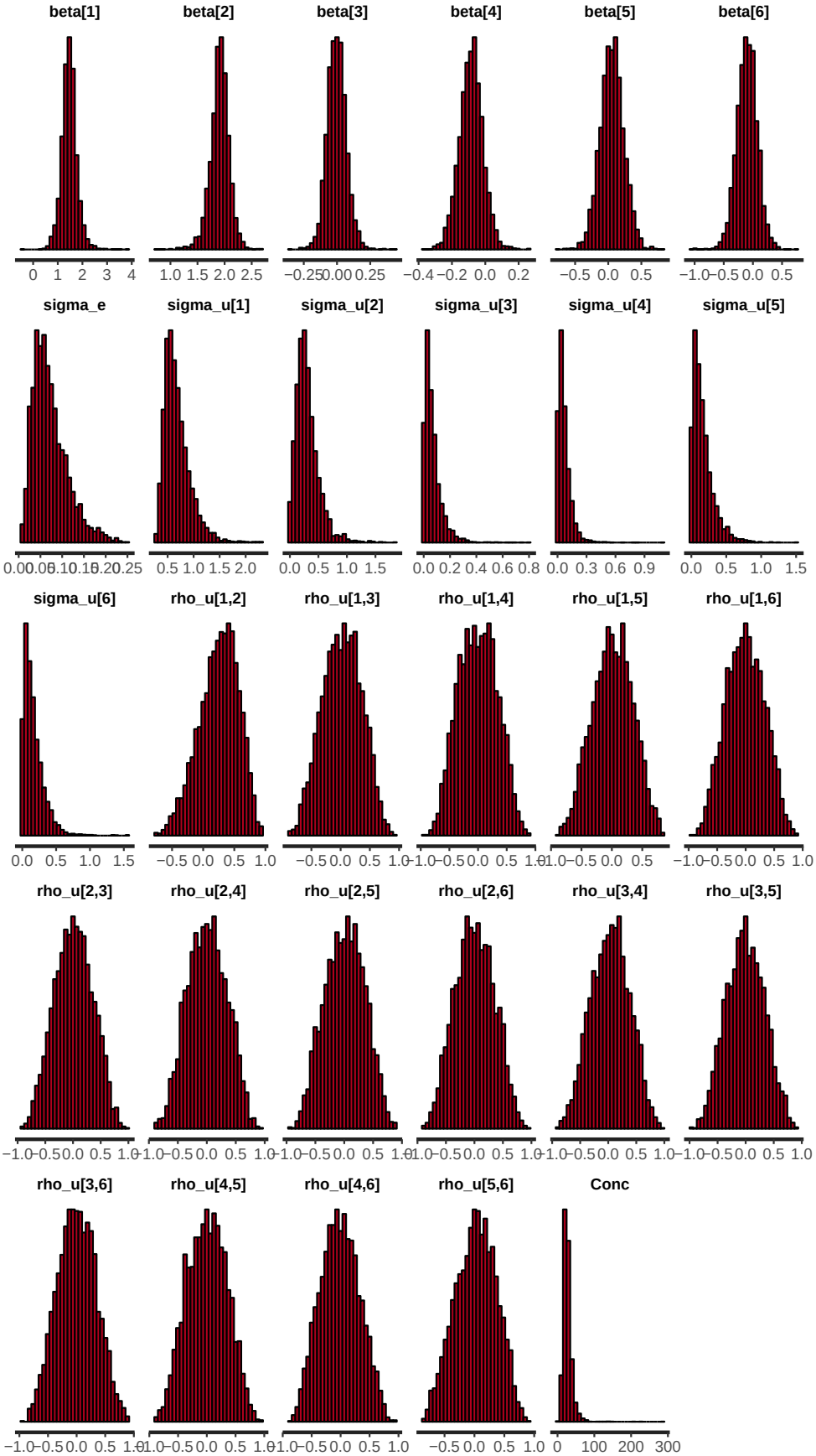
The traceplot shows that the residual error needs to be reparameterized (to-do), but all other parameters are converging nicely.

```
traceplot(MconcMaximal2,pars=c("beta","sigma_e","sigma_u","rho_u[1,2]","rho_u[1,3]","rho_u[1,4]","rho_u
```




```
stan_hist(MconcMaximal2,pars=c("beta","sigma_e","sigma_u","rho_u[1,2]","rho_u[1,3]","rho_u[1,4]","rho_u
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


beta[1]
beta[2]
beta[3]
beta[4]
beta[5]
beta[6]
sigma_e
sigma_u[1]
sigma_u[2]
sigma_u[3]
sigma_u[4]
sigma_u[5]
sigma_u[6]
rho_u[1,2]
rho_u[1,3]
rho_u[1,4]
rho_u[1,5]
rho_u[1,6]
rho_u[2,3]
rho_u[2,4]
rho_u[2,5]
rho_u[2,6]
rho_u[3,4]
rho_u[3,5]
rho_u[3,6]
rho_u[4,5]
rho_u[4,6]
rho_u[5,6]
Conc