Cellular Impact of Contrast Agents

Angela Ariza de Schellenberger, Eric Barnhill and Shravan Vasishth

2018.05.14

Overview

This study evaluated whether novel cell measurement technologies showed differences in cell behavior between three widely used MRI contrast agents: Gadovist, Magnevist and Dotarem. These three agents are considered to have differing levels of safety and toxicity. We investigated whether we could detect differences between the effects of the contrast agents on two kinds of cells, Monocytes and Neutrophils. To measure differences we used two different measurement techniques: Time-Of-Flight Mass Cytometry (CyTOF) and Realtime Deformation of Cells (RT-DC).

While these methods produce many thousands of output points, the present approach was designed to be compatible with the standard outputs of both measurements, which are summary statistics of means and quantiles. Using po

CyTOF

Time-Of-Flight Mass Cytometry (CyTOF) measures spectral response of different cell types at the cellular level. Here the research question was whether there were differences in performance between three clinically used Gadolinium compounds: Gadovist, Magnevist and Dotarem. As these three compounds have different safety levels, investigating whether the signal levels were also different could aid clinical policy in choosing which compound to use.

Samples were analyzed for each of six cell types. In this pilot study a single subject was analyzed three times at three different concentration levels. In contrast to previous drafts, here Concentration is treated as a continuous variable. Regressing the data against a linear model then becomes easy to interpret: at zero concentration we expect all intercepts to be zero, and we are principally interested in differences in the slopes of the linear fit of contrast agent to concentration. Another way to interpret this "slope" number is the amount of signal at unit concentration. This is a particularly handy interpreter as we have directly measured unit concentration in the data.

Our specific statistical question is whether there is an effect of contrast agent on the signal, independent of trial and cell type. It is clear that results will covary with cell type, but we expect the distribution around the "experiment" parameter to be iid.

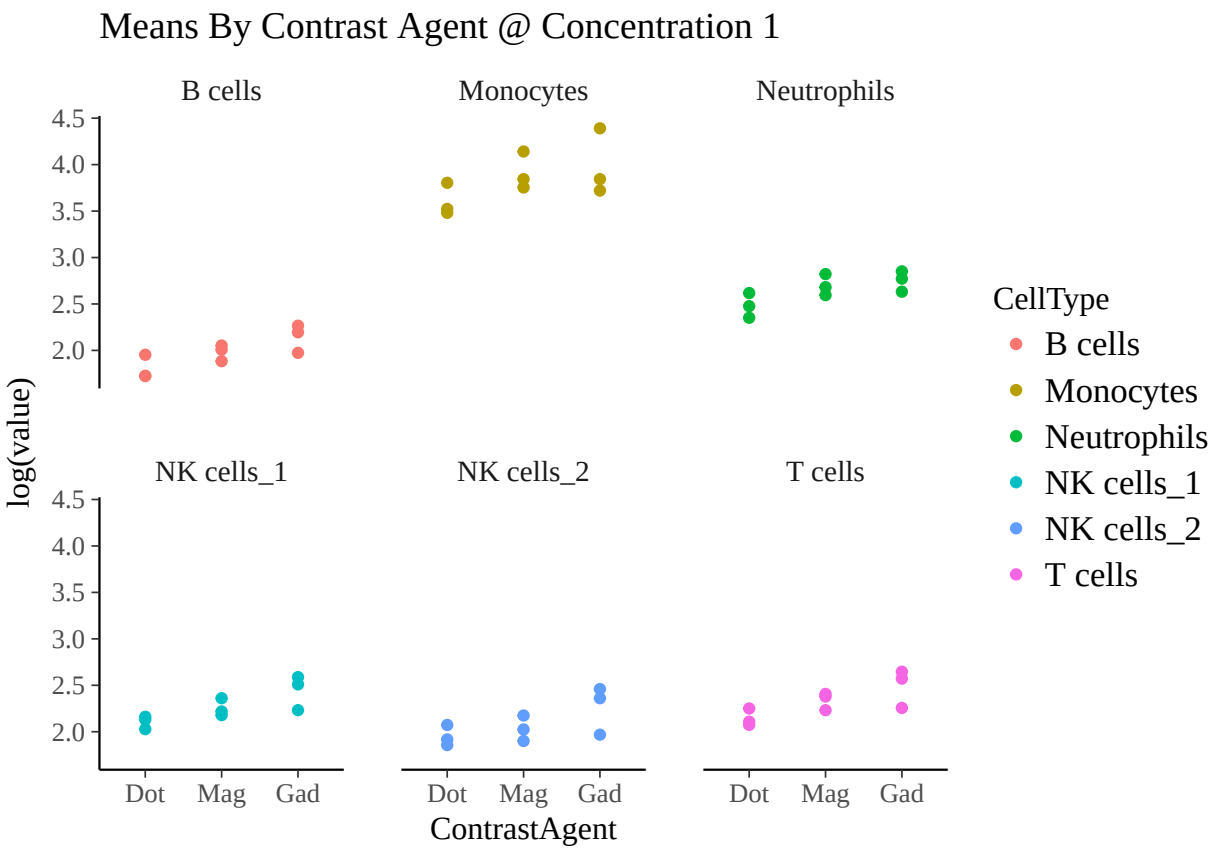
Contrary to a previous draft we do not center or scale the concentration values, as the value of 1 represents the industry standard dosage, making a unit change in the Concentration parameter easy to interpret. Concentration is further treated as a continuous variable with an intercept of 0.

Visualizing the means

Here we plot means at concentration 1 for each contrast agent within each cell type. There seems good evidence of an effect where Dot < Mag and Mag < Gad. We use this sliding contrast coding.

```
means_conc_1 <- subset(means, (Concentration == 1))
means_conc_1$ContrastAgent <- factor(means_conc_1$ContrastAgent, c("Dotarem", "Magnevist", "Gadovist"))
plt <- ggplot(means_conc_1) +
    geom_point(aes(x=ContrastAgent, y=log(value), color=CellType)) +
```

```
    facet_wrap(~ CellType, ncol=3) +
    ggtitle("Means By Contrast Agent @ Concentration 1") +
    scale_x_discrete(labels = c("Dot", "Mag", "Gad"))
print(plt)
```



Normal Means Model

We first model the data using only means, then apply a latent error model.

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

```
priors<-c(set_prior("cauchy(0,10)", class = "b"),
          set_prior("normal(0,1)", class = "sd"),
          set_prior("normal(0,1)", class = "sigma"),
          set_prior("lkj(2)", class = "cor"))

cytof_brm<-brm(formula = value ~
                    Concentration +
                    GvsM+MvsD+
                    Concentration:GvsM +
                    Concentration:MvsD +
                         (1+
```

```
                    Concentration+
                    GvsM+MvsD+
                    Concentration:GvsM +
                    Concentration:MvsD | 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))
```

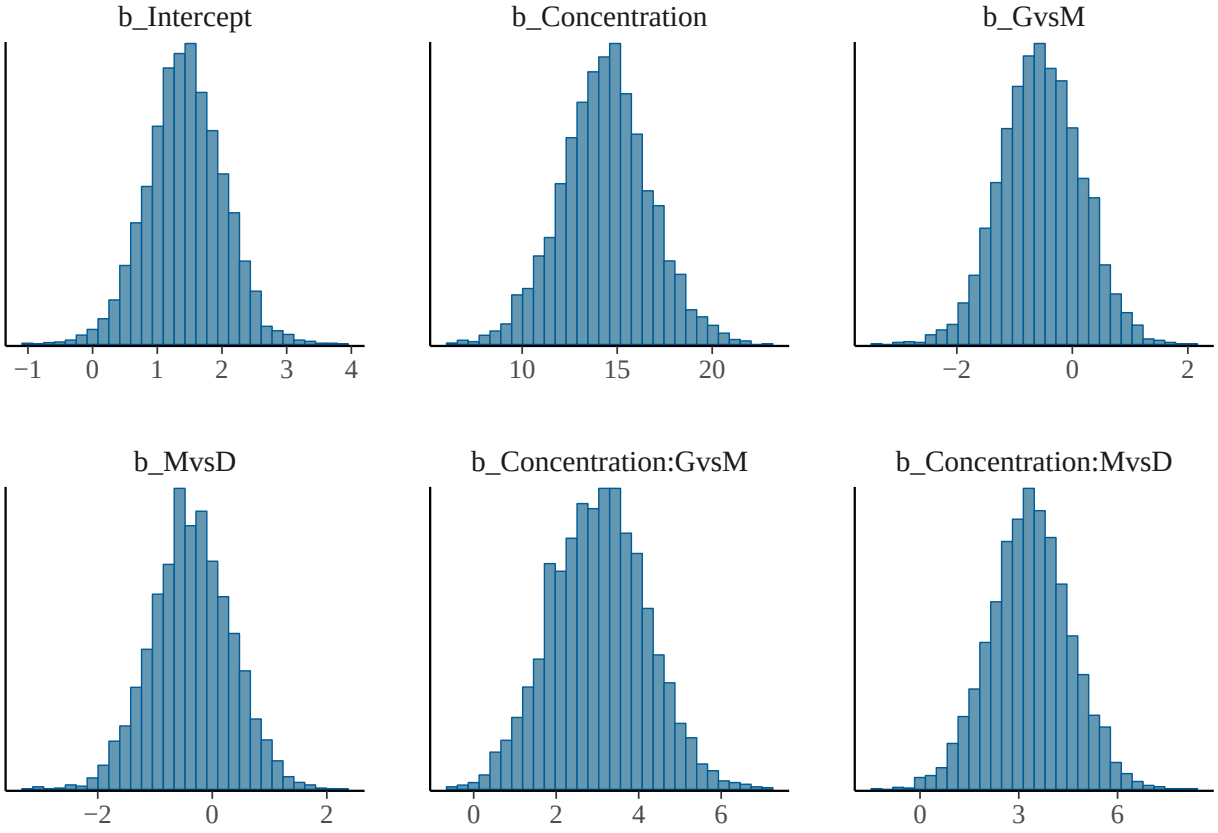
```
## Compiling the C++ model
```

```
## Start sampling
```

Summary of the results:

```
#print(summary(cytof_brm))
stanplot(cytof_brm, type="hist",pars=c("^b"))
```

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



As reported in previous drafts, correlations between the coefficients are spread across a very wide credible interval, and standard deviations within the group parameters reach down to zero.

When Concentration is treated as a continuous variable and the data is regressed against a linear rather than a log-linear model, most of the population-level effects are relatively easy to interpret:

- Intercept we expect to be zero because we expect all samples to show zero signal at zero concentration.

However we have a significant but small positive intercept. I think this has to be interpreted as an indicator of some noise in the data.
- Concentration indicates the mean value at concentration 1 across all cell types. This is not so hard to understand, but doesn't have a useful biological interpretation
- GvsM and MvsD indicate constant offsets by contrast agent. Credible intervals cross through zero. This is the expected result.
- The key contrasts of experimental interest are whether, across a concentration, slope Mag increase > slope Gad increase and diff(Dot) > diff(Mag) (same things). Here we see clear 95% confident effects. Specifically:

```
## [1] "95% HDI, Mag > Gad :"
```

```
##       b_Concentration:GvsM
## lower            0.8155199
## upper            5.3163121
## attr(,"credMass")
## [1] 0.95
```

```
## [1] "95% HDI, Dot > Mag :"
```

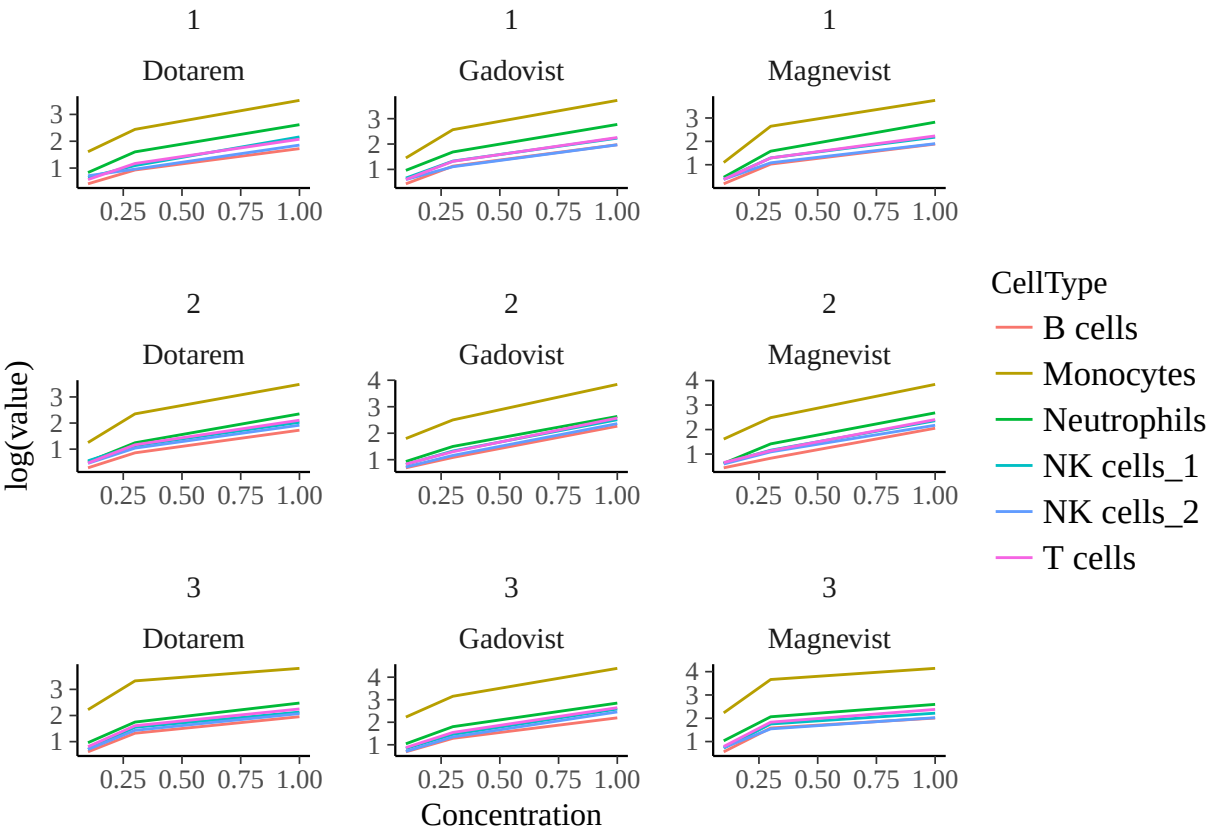
```
##       b_Concentration:MvsD
## lower             1.058749
## upper             5.838136
## attr(,"credMass")
## [1] 0.95
```

We can conclude with over 95% confidence that in this study, Dotarem contained more signal per unit of contrast than Magnevist, and Magnevist contained more signal per unit of contrast than Gadovist.

Log-log means model

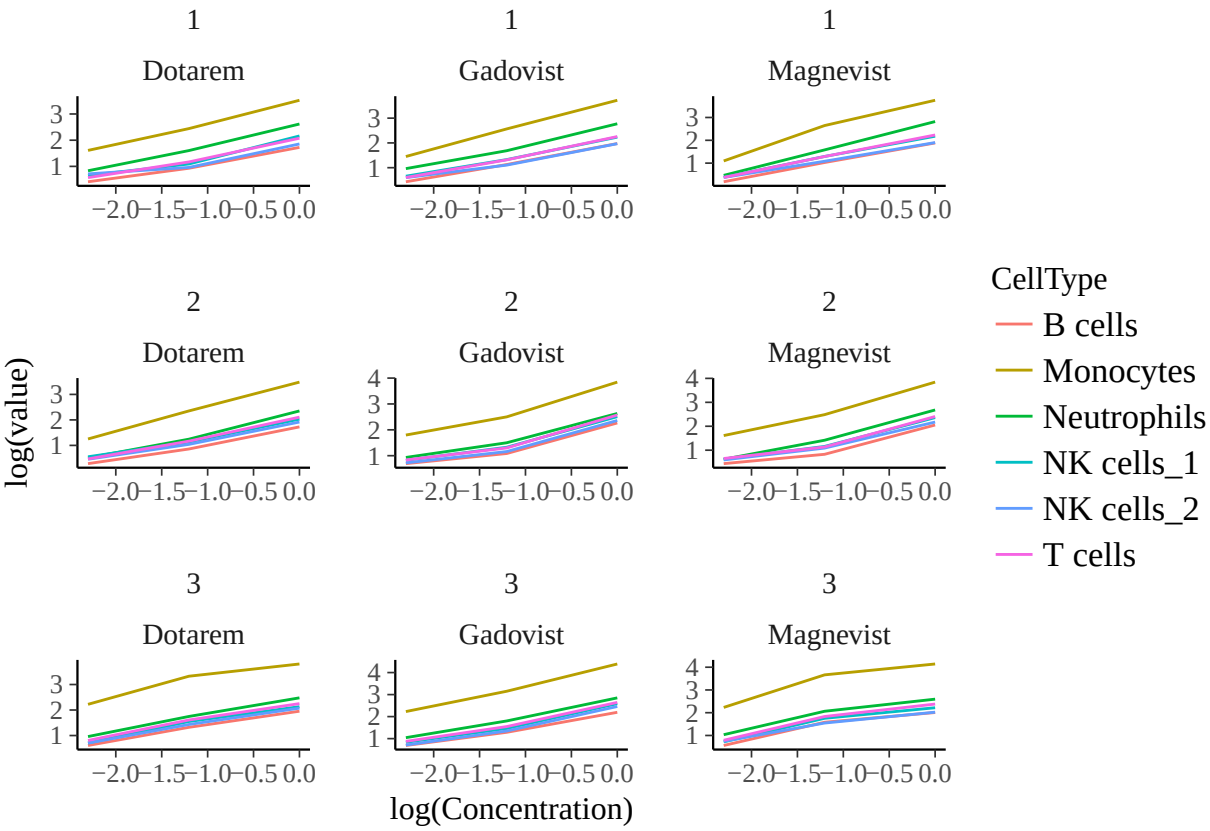
While the above model left coefficients easy to interpret, the distributions of the CyTOF data are better modelled by a lognormal distribution. However, the log means are then not linear in the concentration:

```
ggplot(subset(means, ContrastAgent != 'control')) +
    geom_line(aes(x=Concentration, y=log(value), group=CellType, color=CellType)) +
    facet_wrap(~ Experiment + ContrastAgent, scales='free')
```



However, modelling log signal against log concentration appears to be roughly linear [to do – Shravan, can we justfiy this better?):

```
ggplot(subset(means, ContrastAgent != 'control')) +
    geom_line(aes(x=log(Concentration), y=log(value), group=CellType, color=CellType)) +
    facet_wrap(~ Experiment + ContrastAgent, scales='free')
```



We can thus run the same model on the log scale. Here we make a key change in the model. Instead of modelling the intercepts as zero and the change as occurring in the *slope* of the data, we model the slope as uniform and the change as occurring in the *intercept* of the data. Consequently we will interpret change in *intercept* as an effect. We also move to a more fully specified model:

```
# FOR LOG MODELS

priors<-c(set_prior("cauchy(0,10)", class = "b"),
          set_prior("normal(0,1)", class = "sigma"))

means_log = means
means_log$value = log(means_log$value)
means_log$Concentration = log(means_log$Concentration)

cytof_brm_log<-brm(formula = value ~
             Concentration +
             GvsM+
             MvsD+
             Concentration:GvsM +
             Concentration:MvsD +
                 (1 +
                 Concentration +
                 GvsM+
                 MvsD+
                 Concentration:GvsM +
                 Concentration:MvsD
                 | CellType),
             data = subset(means_log,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
```

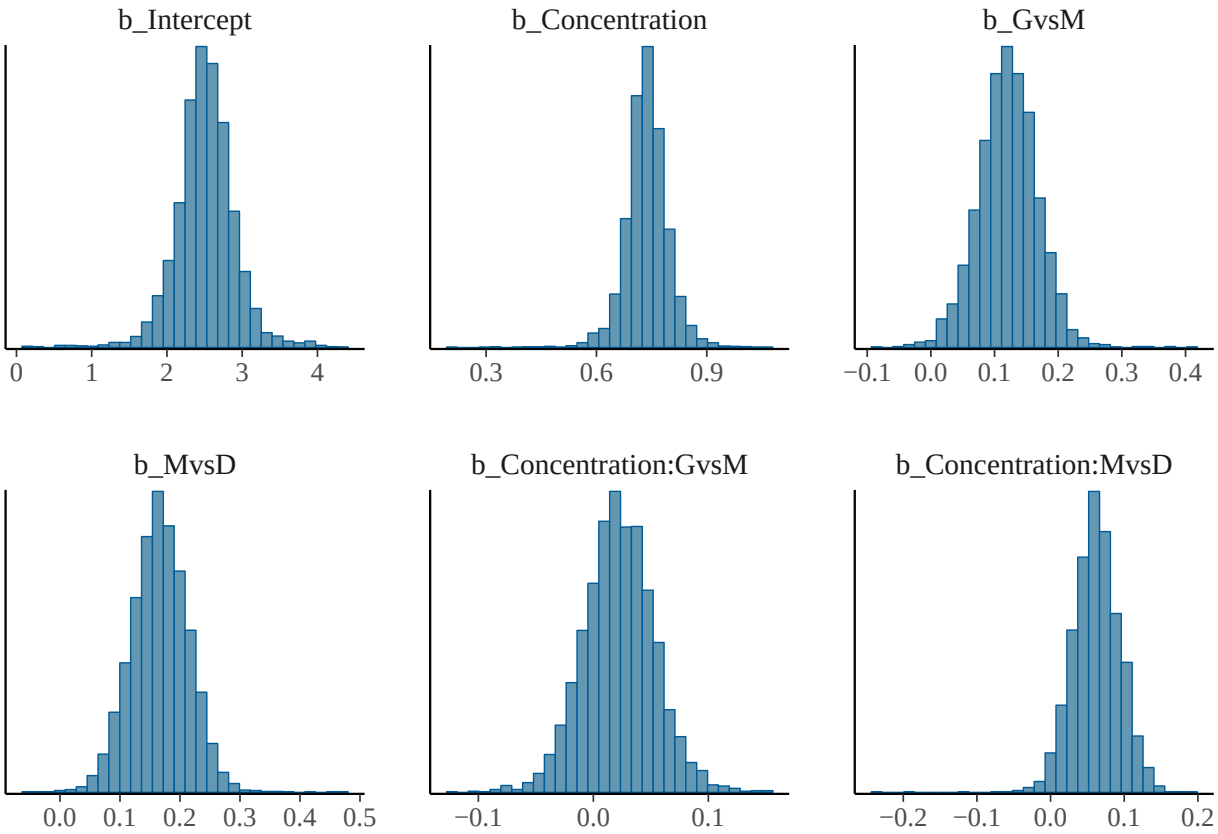
```
##
## SAMPLING FOR MODEL 'gaussian brms-model' NOW (CHAIN 1).
##
## Gradient evaluation took 9.7e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.97 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 16.4722 seconds (Warm-up)
##                12.2538 seconds (Sampling)
##                28.726 seconds (Total)
##
##
## SAMPLING FOR MODEL 'gaussian brms-model' NOW (CHAIN 2).
##
## Gradient evaluation took 5.3e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.53 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
```

```
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 16.1039 seconds (Warm-up)
##                9.81675 seconds (Sampling)
##                25.9207 seconds (Total)
##
##
## SAMPLING FOR MODEL 'gaussian brms-model' NOW (CHAIN 3).
##
## Gradient evaluation took 5.7e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.57 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 16.9903 seconds (Warm-up)
##                11.5619 seconds (Sampling)
##                28.5522 seconds (Total)
##
##
## SAMPLING FOR MODEL 'gaussian brms-model' NOW (CHAIN 4).
##
## Gradient evaluation took 5e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.5 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 15.0097 seconds (Warm-up)
##                12.1243 seconds (Sampling)
```

```
##                 27.134 seconds (Total)
```

```
stanplot(cytof_brm_log, type="hist",pars=c("^b"))
```

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



We then want to view the exponent of the posterior distributions, this time interested in the intercept:

```
## [1] "95% HDI, log(Gad) > log(Mag) :"
```

```
##    lower    upper
## 1.026838 1.231149
## attr(,"credMass")
## [1] 0.95
```

```
## [1] "95% HDI, log(Mag) > log(Dot) :"
```

```
##    lower    upper
## 1.075661 1.290451
## attr(,"credMass")
## [1] 0.95
```

In the log case the relationship is multiplicative, so if 95% of the probability mass is greater than 1, there is a 95% confident effect that the contrast agent increases the signal.

Predictive modelling

[While the goal of the analysis was to find the impact of the contrast agent independent of cell type and experiment, it leaves the result somewhat abstract. We can use the model to predict, for example, what impact a change of contrast agent will have on each cell types. For example we can ask the question, if we

use Gadovist instead of Magnevist, what is the expected increase in signal for each unit of concentration, as estimated at concentration 1:

[TO DO]

Latent error model

While many applications of CyTOF simply use the central tendency statistic, CyTOF machines typically produce a range of summary statistics including quantiles. Exploiting these additional statistics should create a more robust model in a way that is clinically easy to apply and interpret, and therefore of benefit to the CyTOF community.

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

mean_se = means
mean_se$value <- log(mean_se$value)
# test
#mean_se$value <- (mean_se$value)
mean_se$qlow <- log(qlow$value)
# test
#mean_se$qlow <- (qlow$value)
mean_se$qdiff_lo <- mean_se$value - mean_se$qlow

mean_se$qhigh <- log(qhigh$value)
# test
#mean_se$qhigh <- (qhigh$value)
mean_se$qdiff_hi <- mean_se$qhigh - mean_se$value

mean_se$SE<-(mean_se$qdiff_hi) / 1.64
mean_se$Concentration <- log(mean_se$Concentration)
mean_se <- mean_se[c("ContrastAgent", "Concentration", "CellType", "GvsM", "MvsD", "qlow", "qdiff_lo",
```

We can see that the current approach to the SE of the model – subtracting log(hi) from log(mean) – does not appear to be a good choice. The difference between log(lo) and log(mean), and between log(hi) and log(mean), are often very different:

```
head(mean_se[,6:ncol(mean_se)])
```

```
##         qlow qdiff_lo      value  qdiff_hi     qhigh        SE
## 1 -1.5141277 1.290984 -0.2231436 0.8649974 0.6418539 0.5274375
## 2 -1.3470736 1.810808  0.4637340 1.0246656 1.4883996 0.6247961
## 3 -0.5978370 2.176816  1.5789787 1.0012381 2.5802168 0.6105111
## 4  1.0543120 1.767067  2.8213789 1.0180734 3.8394523 0.6207765
## 5 -1.2039728 2.036882  0.8329091 1.0280654 1.8609745 0.6268692
## 6 -0.4620355 2.065455  1.6034198 0.9615295 2.5649494 0.5862985
```

Nonetheless, we can now run the same fully-specified brms log-log model including the standard error term:

```
priors<-c(set_prior("cauchy(0,10)", class = "b"),
          set_prior("cauchy(0,5)", class = "b",coef="Concentration"),
          set_prior("normal(0,1)", class = "sd"),
          set_prior("lkj(2)", class = "cor"))
```

```
cytof_brm_se <-brm(formula = value | se(SE) ~
            Concentration +
            GvsM+
            MvsD+
            Concentration:GvsM +
            Concentration:MvsD +
                (1 +
                Concentration +
                GvsM+
                MvsD+
                Concentration:GvsM +
                Concentration:MvsD
                | CellType),
            data = subset(mean_se,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
```

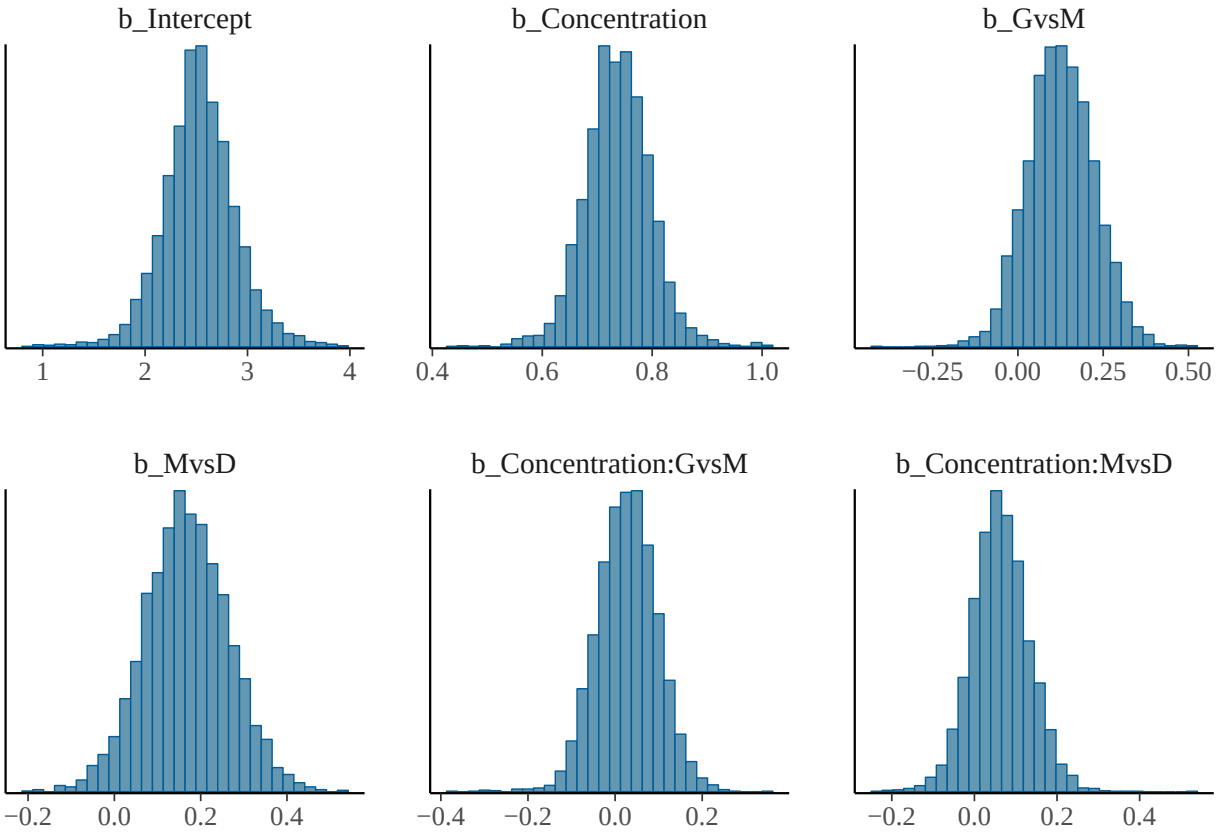
```
##
## SAMPLING FOR MODEL 'gaussian brms-model' NOW (CHAIN 1).
##
## Gradient evaluation took 9.9e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.99 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 8.01611 seconds (Warm-up)
##                6.47925 seconds (Sampling)
##                14.4954 seconds (Total)
##
##
## SAMPLING FOR MODEL 'gaussian brms-model' NOW (CHAIN 2).
##
## Gradient evaluation took 5.8e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.58 seconds.
```

```
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 9.93261 seconds (Warm-up)
##                6.58026 seconds (Sampling)
##                16.5129 seconds (Total)
##
##
## SAMPLING FOR MODEL 'gaussian brms-model' NOW (CHAIN 3).
##
## Gradient evaluation took 8.7e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.87 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 8.14403 seconds (Warm-up)
##                3.34356 seconds (Sampling)
##                11.4876 seconds (Total)
##
##
## SAMPLING FOR MODEL 'gaussian brms-model' NOW (CHAIN 4).
##
## Gradient evaluation took 5.7e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.57 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 7.11085 seconds (Warm-up)
##                4.20043 seconds (Sampling)
##                11.3113 seconds (Total)
```

```
stanplot(cytof_brm_se, type="hist",pars=c("^b"))
```

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



Let's see if there is any difference in the confidence intervals:

```
## [1] "95% HDI, log(Gad) > log(Mag) :"
```

```
##     lower     upper
## 0.9354731 1.3637477
## attr(,"credMass")
## [1] 0.95
```

```
## [1] "95% HDI, log(Mag) > log(Dot) :"
```

```
##     lower     upper
## 0.9653766 1.4188107
## attr(,"credMass")
## [1] 0.95
```

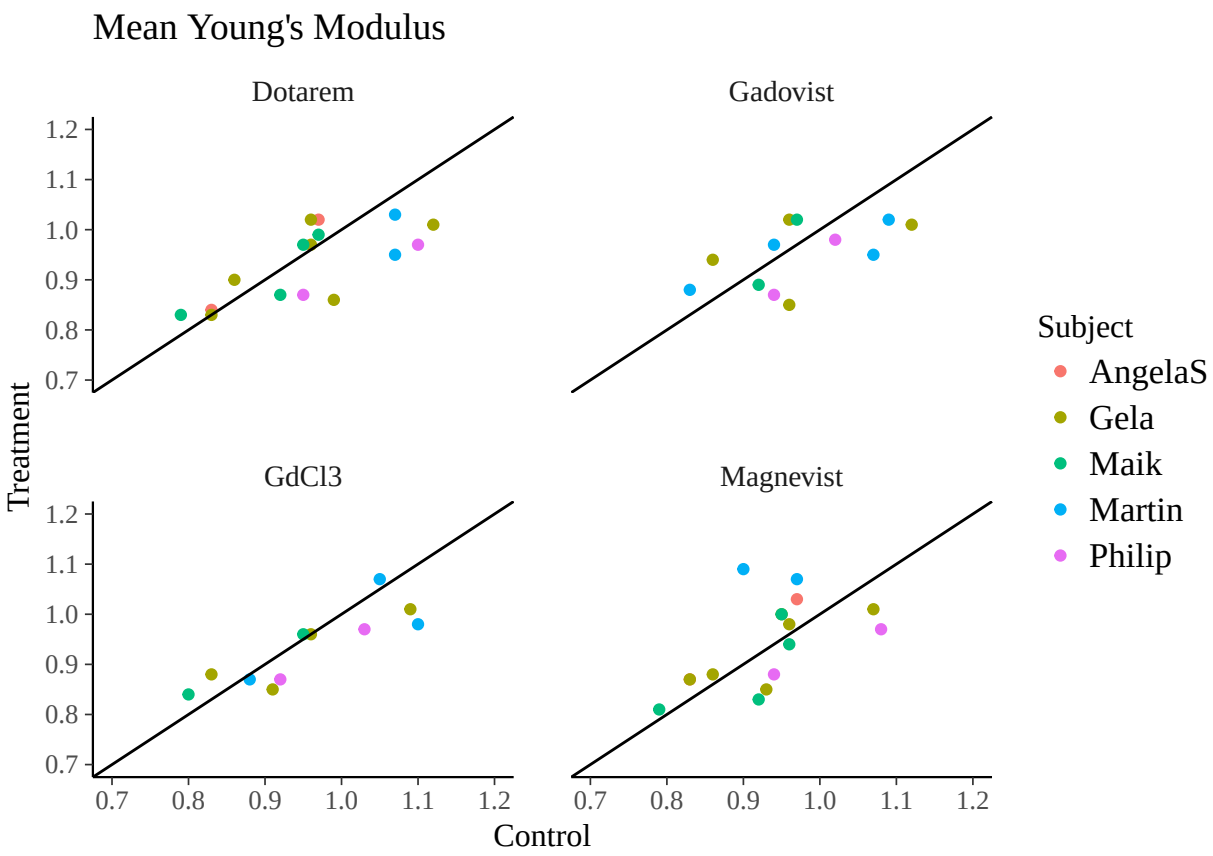
This approach as presently coded simply widens the variance of the estimated parameters. Central tendency is still similar (means of 1.1301526 and 1.1827583 for the log-log model without SE; and means of 1.140262 and 1.1887865 for the log-log model with SE. Including the error creates less confidence in the estimate and takes us outside a 95% confidence interval for the effects. However, I do understand that small studies tend to be underpowered so if this is a superior model, this is what we should report.

RT-DC

RT-DC is a method of measuring cellular elastic deformations developed by TU-Dresden. We use this technology to evaluate whether cells containing common contrast agents show differences in the Young's modulus, which would indicate that their physiological properties are being altered by the contrast agent.

The RT-DC method is currently experimental, and control tests for an individual subject produce a wide range of values. Consequently each measurement with a contrast agent was paired with its own control from the same session. The Figure below shows pairwise control-treatment plots colored by subject:

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



While the range of control values is quite wide (0.008131), the control-treatment pairs deviate an average of 0.0049137 from the unit line, suggesting the control-treatment pairings are a stabler value set.

Monocytes

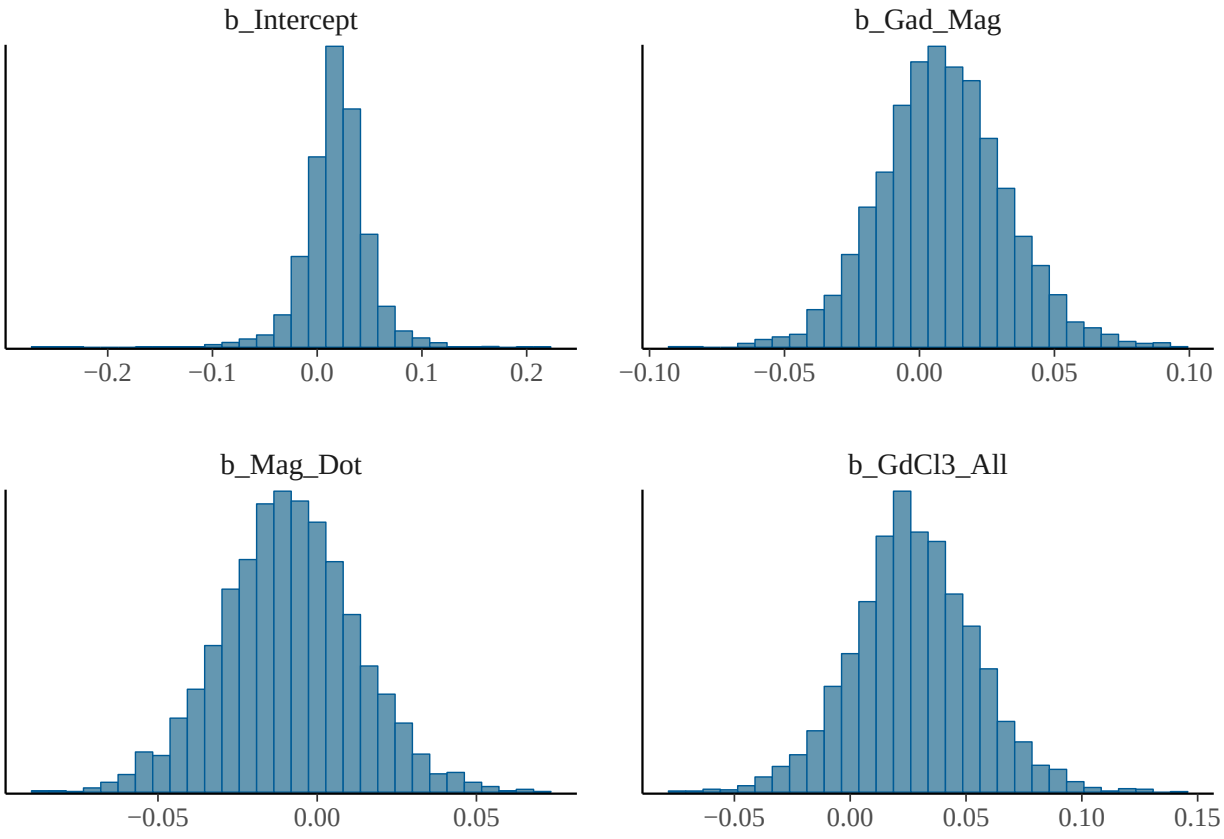
```
priors_cauchy_diff2 <- c(set_prior("cauchy(0, 10)", class = "Intercept"),
            set_prior("cauchy(0, 10)", class = "b"),
            set_prior("cauchy(0, 10)", class = "sd"),
            set_prior("cauchy(0, 10)", class = "sigma")
            )


mdiffMonoContrAgt_E <- brm(formula = diff ~ 1+Gad_Mag+Mag_Dot+GdCl3_All+
                (1| Subject),
              data = diffMono, family = gaussian(), prior = priors_cauchy_diff2,
              iter = 2000, chains = 4, control = list(adapt_delta = 0.999))
```

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

```
## Start sampling
```

```
stanplot(mdiffMonoContrAgt_E, type="hist",pars=c("^b"))
```

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




```
mMonoContrAgtdiffEpost<-posterior_samples(mdiffMonoContrAgt_E, "^b")
```

For Monocytes, there is weak evidence that GdCl3 causes a decrease in Young's Modulus against the other three contrast agents. The comparison of the individual contrast agents is not determinate at this time:

```
## [1] "Prob Gad > Mag"
```

```
## [1] 0.648
```

```
## [1] "Prob Mag > Dot"
```

```
## [1] 0.33
```

```
## [1] "Prob GdCl3 > All"
```

```
## [1] 0.86475
```

Neutrophils

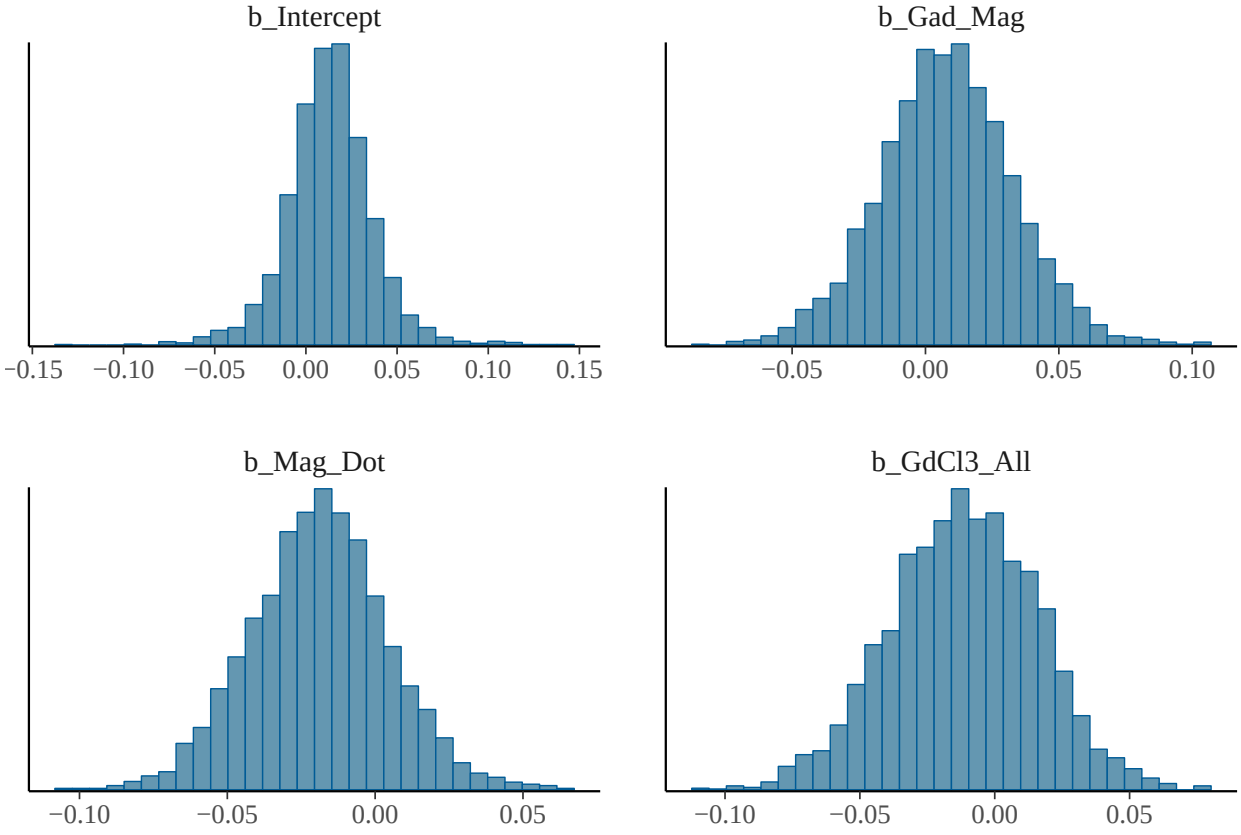
```
mdiffNeutroContrAgt_E <- brm(formula = diff ~ 1+Gad_Mag+Mag_Dot+GdCl3_All+
                (1| Subject),
              data = diffNeutro, family = gaussian(), prior = priors_cauchy_diff2,
              iter = 2000, chains = 4, control = list(adapt_delta = 0.999))
```

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

```
## Start sampling
```

```
stanplot(mdiffNeutroContrAgt_E, type="hist",pars=c("^b"))
```

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




```
mNeutroContrAgtdiffEpost<-posterior_samples(mdiffNeutroContrAgt_E, "^b")
```

For Neutrophils, there isn't much evidence for differences by Contrast Agent, though there is again maybe some weak evidence (80% likelihood) that Magnevist has higher Young's modulus than Dotarem:

```
## [1] "Prob Gad > Mag"
```

```
## [1] 0.63
```

```
## [1] "Prob Mag > Dot"
## [1] 0.19625
## [1] "Prob GdCl3 > All"
## [1] 0.3415
```

Latent Error Model

TODO: Use hacker stats to create distributions off the mean and quantiles, then subtract the two distributions to get a distribution of the differences??