## Contents

# Assessing the effect of priors on BLUPs

```
load("data/analyses_data/pca.RData")
library(tidyr)
library(dplyr)
library(MCMCglmm)
library(pander)
set.alignment('right', row.names = 'left')

doc_data <- pca_data %>% filter(!is.na(docil))
agg_data <- pca_data %>% filter(!is.na(misPC1))
act_data <- pca_data %>% filter(!is.na(ofPC1))
```

## Priors

From a post to r-sig-me by Ned Dochterman

1. Parameter expanded
2. Another parameter expanded just to see if results vary across runs
3. Parameter expanded variance = docility variance
4. Parameter expanded really high variance
5. Inverse Wishart
6. Inverse Gamma
7. Flat, uniform, prior for just a variance
8. Flat improper prior, equivalent to REML fitting.

```
priors <- list(
  list(
    G=list(G1=list(V=1, nu=1, alpha.mu = 0, alpha.V = 10000)),
    R=list(V=1, nu=1)
  ),
  list(
    G=list(G1=list(V=1, nu=1, alpha.mu = 0, alpha.V = 10000)),
    R=list(V=1, nu=1)
  ),
  list(
    G=list(
      G1=list(
        V=var(doc_data$docil, na.rm = TRUE), nu=1, alpha.mu = 0,
          alpha.V = 10000
      )
    ),
    R=list(V=var(doc_data$docil, na.rm = TRUE), nu=1)
  ),
  list(
    G=list(G1=list(V=1000, nu=1, alpha.mu = 0, alpha.V = 1000)),
    R=list(V=1000, nu=1)
  ),
  list(G=list(G1=list(V=1, nu=1)), R=list(V=1, nu=1)),
  list(G=list(G1=list(V=1, nu=0.002)), R=list(V=1, nu=0.002)),
  list(G=list(G1=list(V=1e-16, nu=-2)), R=list(V=1e-16, nu=-2))  ,
  list(G=list(G1=list(V=1,nu=0)),R = list(V =1, nu = 0))
)
```

## Run models

```
library(foreach)

## foreach: simple, scalable parallel programming from Revolution Analytics
## Use Revolution R for scalability, fault tolerance and more.
## http://www.revolutionanalytics.com

library(doMC)

## Loading required package: iterators
## Loading required package: parallel

registerDoMC(cores = 8)

thin <- 100
burnin <- thin * 100
nitt <- burnin + thin * 1000

time_start <- Sys.time()
m_priors <- foreach(i = 1:length(priors)) %dopar% {
  MCMCglmm(docil ~ julian + Obs + handlevent_year + I(handlevent_year^2),
                                    random = ~ ID,
                                    prior = priors[[i]],
                                    pr = TRUE,
                                    data = doc_data,
                                    thin = thin,
                                    burnin = burnin,
                                    nitt = nitt,
                                     verbose = FALSE
                                     )
}
print(paste("Approx. models run time: ", format(Sys.time() - time_start)))

## [1] "Approx. models run time:  8.042 mins"

save(m_priors, file = "data/analyses_data/m_priors.RData")
```

### Model Diagnostics

```
load("data/analyses_data/m_priors.RData")

ad <- list()
gd <- list()
hd <- list()

for(i in 1:length(priors)){
  ad[[i]] <- autocorr.diag(m_priors[[i]]$VCV)
  gd[[i]] <- geweke.diag(m_priors[[i]]$VCV)
  hd[[i]] <- heidel.diag(m_priors[[i]]$VCV)
}
ad

## [[1]]
##                ID    units
## Lag 0     1.00000  1.00000
## Lag 100  -0.02922 -0.03013
## Lag 500   0.01560  0.02187
## Lag 1000 -0.05264 -0.03655
## Lag 5000  0.01480  0.01472
##
## [[2]]
##                 ID     units
## Lag 0     1.000000  1.000000
## Lag 100  -0.038620 -0.005688
## Lag 500  -0.038033  0.062510
## Lag 1000  0.041033  0.021035
## Lag 5000  0.008467 -0.030301
##
## [[3]]
##                 ID     units
## Lag 0     1.000000  1.000000
## Lag 100  -0.047543  0.048670
## Lag 500   0.003440 -0.002852
## Lag 1000 -0.005587  0.011396
## Lag 5000  0.003182  0.007828
##
## [[4]]
##                  ID    units
## Lag 0     1.0000000  1.00000
## Lag 100   0.0004055 -0.02710
## Lag 500   0.0480595 -0.03198
## Lag 1000  0.0189893  0.02511
## Lag 5000 -0.0272179  0.01032
##
## [[5]]
##                ID    units
## Lag 0     1.00000  1.00000
## Lag 100   0.06009 -0.03227
## Lag 500   0.06123  0.02975
## Lag 1000  0.01682 -0.01543
## Lag 5000 -0.05175  0.03395
##
## [[6]]
##                  ID    units
## Lag 0     1.0000000  1.00000
## Lag 100   0.0002831 -0.03890
## Lag 500   0.0032093 -0.02185
## Lag 1000 -0.0009037  0.05010
## Lag 5000 -0.0424693 -0.02452
##
## [[7]]
##                 ID    units
## Lag 0     1.000000  1.00000
## Lag 100   0.025706  0.03615
## Lag 500  -0.018627  0.02782
## Lag 1000  0.007562  0.05121
## Lag 5000 -0.035466 -0.02292
##
## [[8]]
##                ID     units
## Lag 0     1.00000  1.000000
## Lag 100  -0.02928 -0.006564
## Lag 500  -0.05204  0.022471
## Lag 1000 -0.01659  0.063554
## Lag 5000 -0.01517 -0.023190
```

```
gd
```

```
## [[1]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     ID  units
## 0.5828 0.5301
##
##
## [[2]]
##
```

```
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##       ID    units
## -0.02447 -1.64978
##
##
## [[3]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      ID   units
## -0.4845  2.2787
##
##
## [[4]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    ID units
## 2.160 1.382
##
##
## [[5]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      ID   units
## -0.4190 -0.1041
##
##
## [[6]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      ID   units
##  0.1051 -1.3288
##
##
## [[7]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##       ID    units
## -0.07847 -2.51430
##
```

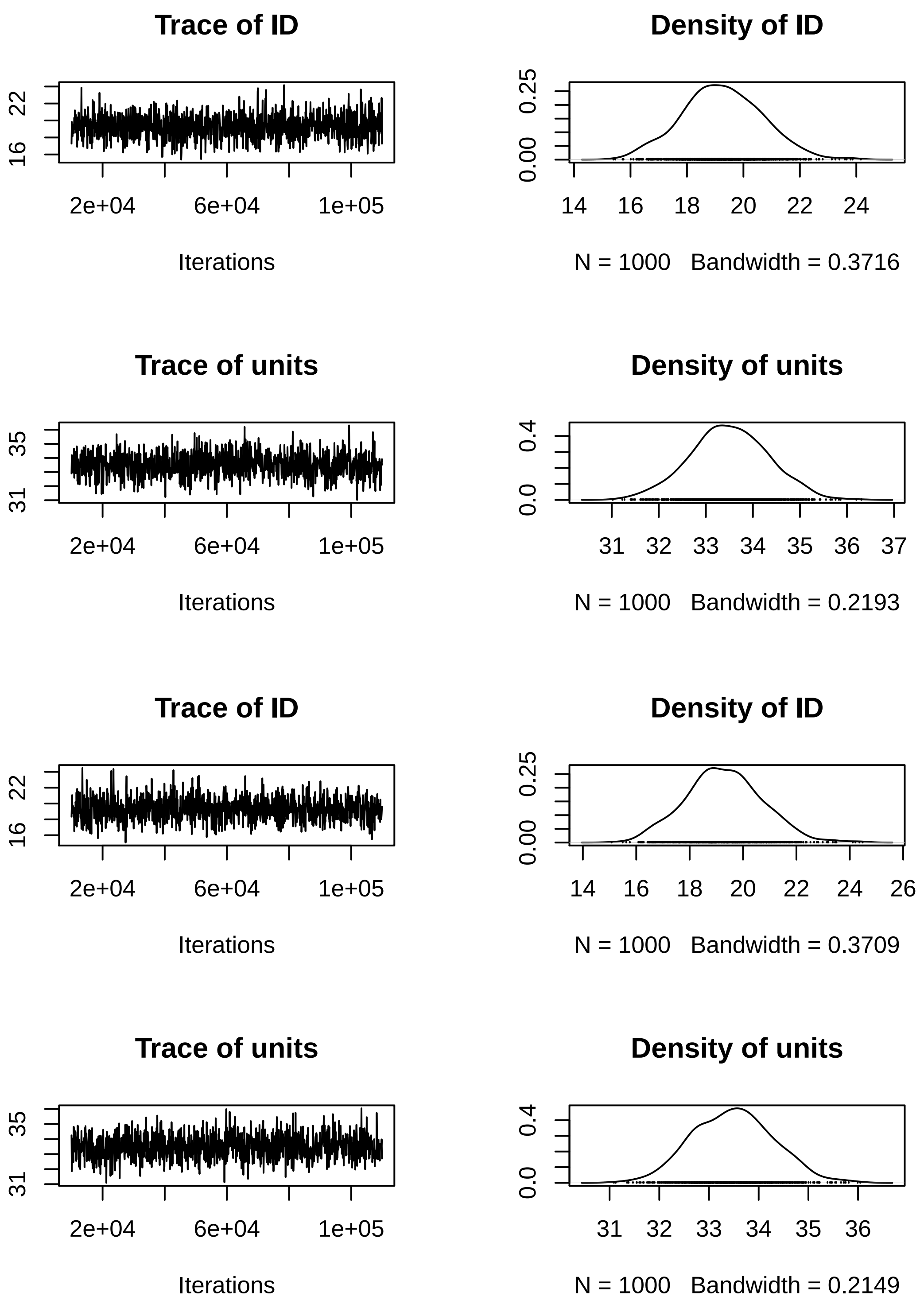
```
##
## [[8]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      ID   units
## -0.3845 -0.1387
```

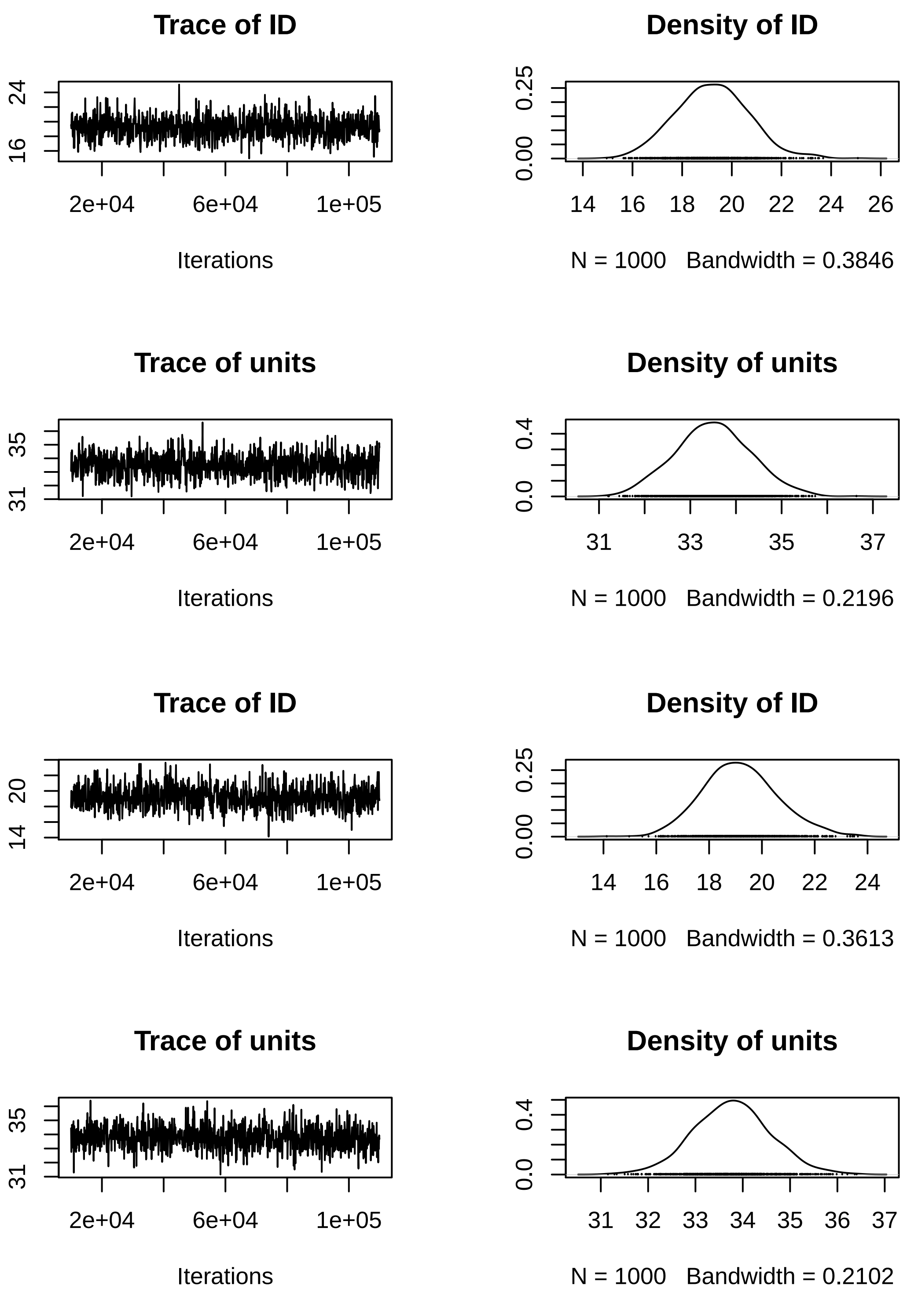
```
hd
```

```
## [[1]]
##
##       Stationarity start     p-value
##       test         iteration
## ID    passed       1         0.794
## units passed       1         0.321
##
##       Halfwidth Mean Halfwidth
##       test
## ID    passed    19.3 0.0880
## units passed    33.5 0.0514
##
## [[2]]
##
##       Stationarity start     p-value
##       test         iteration
## ID    passed       1         0.2942
## units passed       1         0.0967
##
##       Halfwidth Mean Halfwidth
##       test
## ID    passed    19.3 0.0896
## units passed    33.5 0.0500
##
## [[3]]
##
##       Stationarity start     p-value
##       test         iteration
## ID    passed       1         0.800
## units passed       1         0.387
##
##       Halfwidth Mean Halfwidth
##       test
## ID    passed    19.2 0.0906
## units passed    33.5 0.0537
##
## [[4]]
##
##       Stationarity start     p-value
##       test         iteration
## ID    passed       101       0.0706
## units passed       301       0.1075
##
##       Halfwidth Mean Halfwidth
##       test
## ID    passed    19.2 0.0928
## units passed    33.7 0.0623
##
## [[5]]
##
##       Stationarity start     p-value
##       test         iteration
## ID    passed       1         0.572
## units passed       1         0.723
##
##       Halfwidth Mean Halfwidth
##       test
## ID    passed    19.2 0.0943
## units passed    33.5 0.0501
##
## [[6]]
##
##       Stationarity start     p-value
##       test         iteration
## ID    passed       1         0.278
## units passed       1         0.102
##
##       Halfwidth Mean Halfwidth
##       test
## ID    passed    19.2 0.0878
## units passed    33.5 0.0495
##
## [[7]]
##
##       Stationarity start     p-value
##       test         iteration
## ID    passed       1         0.784
## units passed       1         0.487
##
##       Halfwidth Mean Halfwidth
##       test
## ID    passed    19.4 0.0895
## units passed    33.5 0.0513
##
## [[8]]
##
##       Stationarity start     p-value
##       test         iteration
## ID    passed       1         0.535
## units passed       1         0.958
##
```

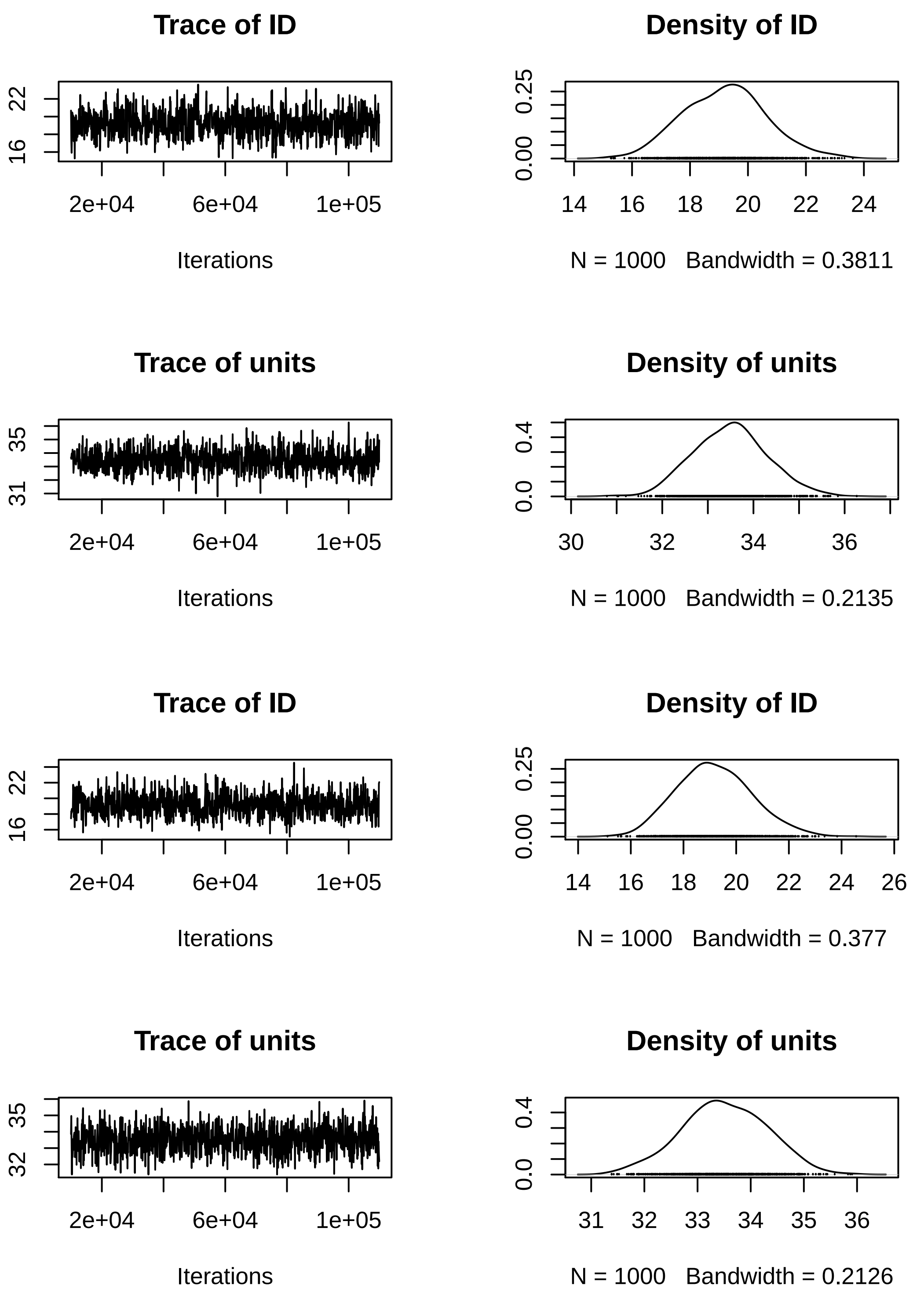
```
##       Halfwidth Mean Halfwidth
##       test
## ID    passed    19.2 0.0883
## units passed    33.5 0.0515
```

```
for(i in 1:length(priors)){
  plot(m_priors[[i]]$VCV)
}
```

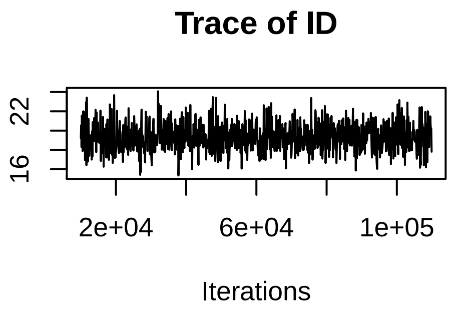

Trace of ID
Density of ID
Iterations
N = 1000 Bandwidth = 0.3716
Trace of units
Density of units
Iterations
N = 1000 Bandwidth = 0.2193
Trace of ID
Density of ID
Iterations
N = 1000 Bandwidth = 0.3709
Trace of units
Density of units
Iterations
N = 1000 Bandwidth = 0.2149

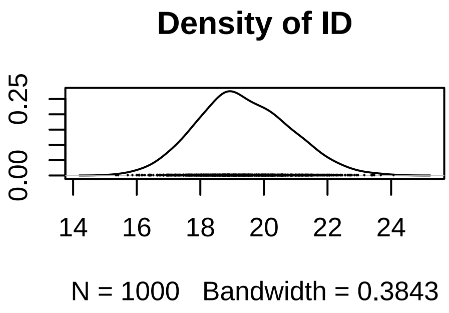


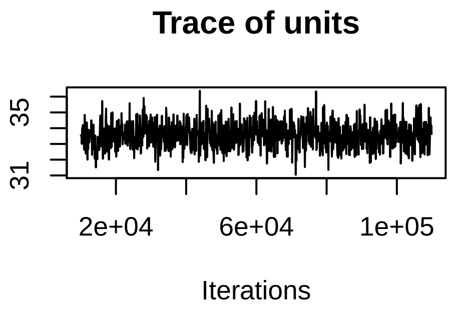
Trace of ID
24
16
2e+04
6e+04
1e+05
Iterations
Density of ID
0.25
0.00
14 16 18 20 22 24 26
N = 1000 Bandwidth = 0.3846
Trace of units
35
31
2e+04
6e+04
1e+05
Iterations
Density of units
0.4
0.0
31 33 35 37
N = 1000 Bandwidth = 0.2196
Trace of ID
20
14
2e+04
6e+04
1e+05
Iterations
Density of ID
0.25
0.00
14 16 18 20 22 24
N = 1000 Bandwidth = 0.3613
Trace of units
35
31
2e+04
6e+04
1e+05
Iterations
Density of units
0.4
0.0
31 32 33 34 35 36 37
N = 1000 Bandwidth = 0.2102

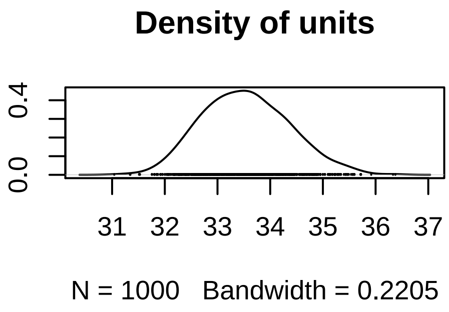


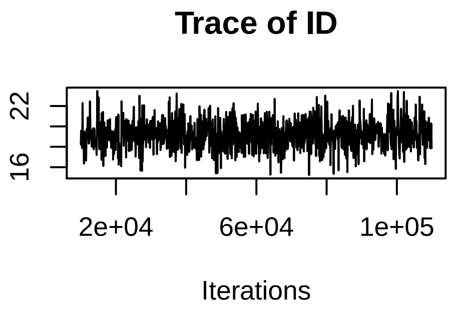
Trace of ID
Iterations
Density of ID
N = 1000 Bandwidth = 0.3811
Trace of units
Iterations
Density of units
N = 1000 Bandwidth = 0.2135
Trace of ID
Iterations
Density of ID
N = 1000 Bandwidth = 0.377
Trace of units
Iterations
Density of units
N = 1000 Bandwidth = 0.2126

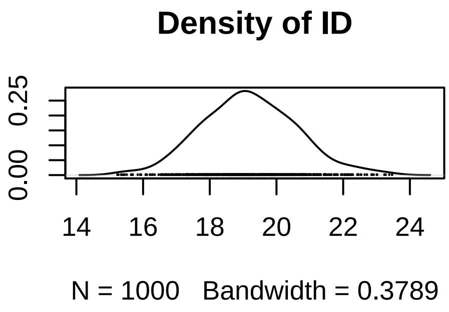

Trace of ID
22
16
2e+04
6e+04
1e+05
Iterations

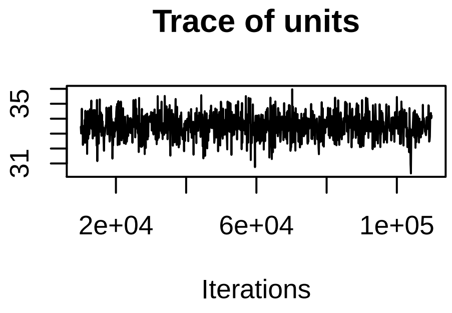

Density of ID
0.25
0.00
14
16
18
20
22
24
N = 1000   Bandwidth = 0.3843

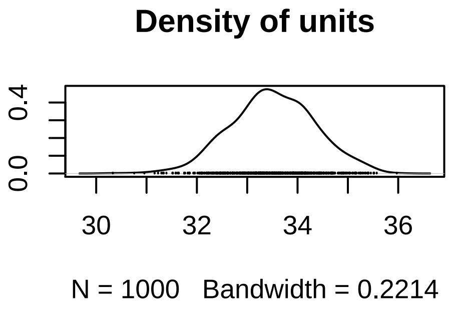

Trace of units
35
31
2e+04
6e+04
1e+05
Iterations


Density of units
0.4
0.0
31
32
33
34
35
36
37
N = 1000   Bandwidth = 0.2205


Trace of ID
22
16
2e+04
6e+04
1e+05
Iterations


Density of ID
0.25
0.00
14
16
18
20
22
24
N = 1000   Bandwidth = 0.3789


Trace of units
35
31
2e+04
6e+04
1e+05
Iterations


Density of units
0.4
0.0
30
32
34
36
N = 1000   Bandwidth = 0.2214

## Extract raneffs

```
extractMCMCglmmBLUPs <- function(x, value, ptype = "1"){
  p_modes <- posterior.mode(x$Sol) ## Get posterior_modes of the BLUPs
  p_modes <- p_modes[grep("ID", names(p_modes))] ## Get all the ID rows
  p_modes <- stack(p_modes)
  names(p_modes) <- c(value, "ID")
  p_modes$type <- paste("mcmc.mode", ptype, sep = '.')
  p_modes$ID <- gsub("ID\\.", "", p_modes$ID)
  p_modes$itt <- NA
  sols <- data.frame(x$Sol) ## Get BLUPs
  sols <- sols[ ,grep("ID", names(sols))] ## Get all the ID columns
  sols <- stack(sols)
  names(sols) <- c(value, "ID")
  sols$itt <- 1:1000 ## Just an index for each MCMC sample
  sols$type = paste("mcmc", ptype, sep = '.')
  sols$ID <- gsub("ID\\.", "", sols$ID)
  rbind(sols, p_modes)
}

doc_mcmc <- list()
for(i in 1:length(priors)){
  doc_mcmc[[i]] <- extractMCMCglmmBLUPs(m_priors[[i]],
    value = "docility", ptype = i)
}

mcmc_priors <- do.call("rbind", doc_mcmc)
```

## Compare MCMC priors

Comparing the effect of priors on the posterior distributions.

### Posterior modes

```
mcmc_modes <- mcmc_priors[grep("mode", mcmc_priors$type), ]
mcmc_modes$itt <- NULL
mcmc_modes <- spread(mcmc_modes, type, docility)

cov_modes <- cov(mcmc_modes[ ,2:ncol(mcmc_modes)])
cor_modes <- cor(mcmc_modes[ ,2:ncol(mcmc_modes)])

cov_modes[upper.tri(cov_modes)] <- cor_modes[upper.tri(cor_modes)]

pandoc.table(cov_modes)
```

| | mcmc.mode.1 | mcmc.mode.2 | mcmc.mode.3 |
|---|---:|---:|---:|
| **mcmc.mode.1** | 12.03 | 0.9508 | 0.9484 |
| **mcmc.mode.2** | 11.5 | 12.16 | 0.9524 |

|  | mcmc.mode.1 | mcmc.mode.2 | mcmc.mode.3 |
|---|---|---|---|
| **mcmc.mode.3** | 11.55 | 11.66 | 12.33 |
| **mcmc.mode.4** | 11.57 | 11.65 | 11.71 |
| **mcmc.mode.5** | 11.55 | 11.62 | 11.65 |
| **mcmc.mode.6** | 11.51 | 11.63 | 11.61 |
| **mcmc.mode.7** | 11.58 | 11.72 | 11.72 |
| **mcmc.mode.8** | 11.47 | 11.56 | 11.52 |

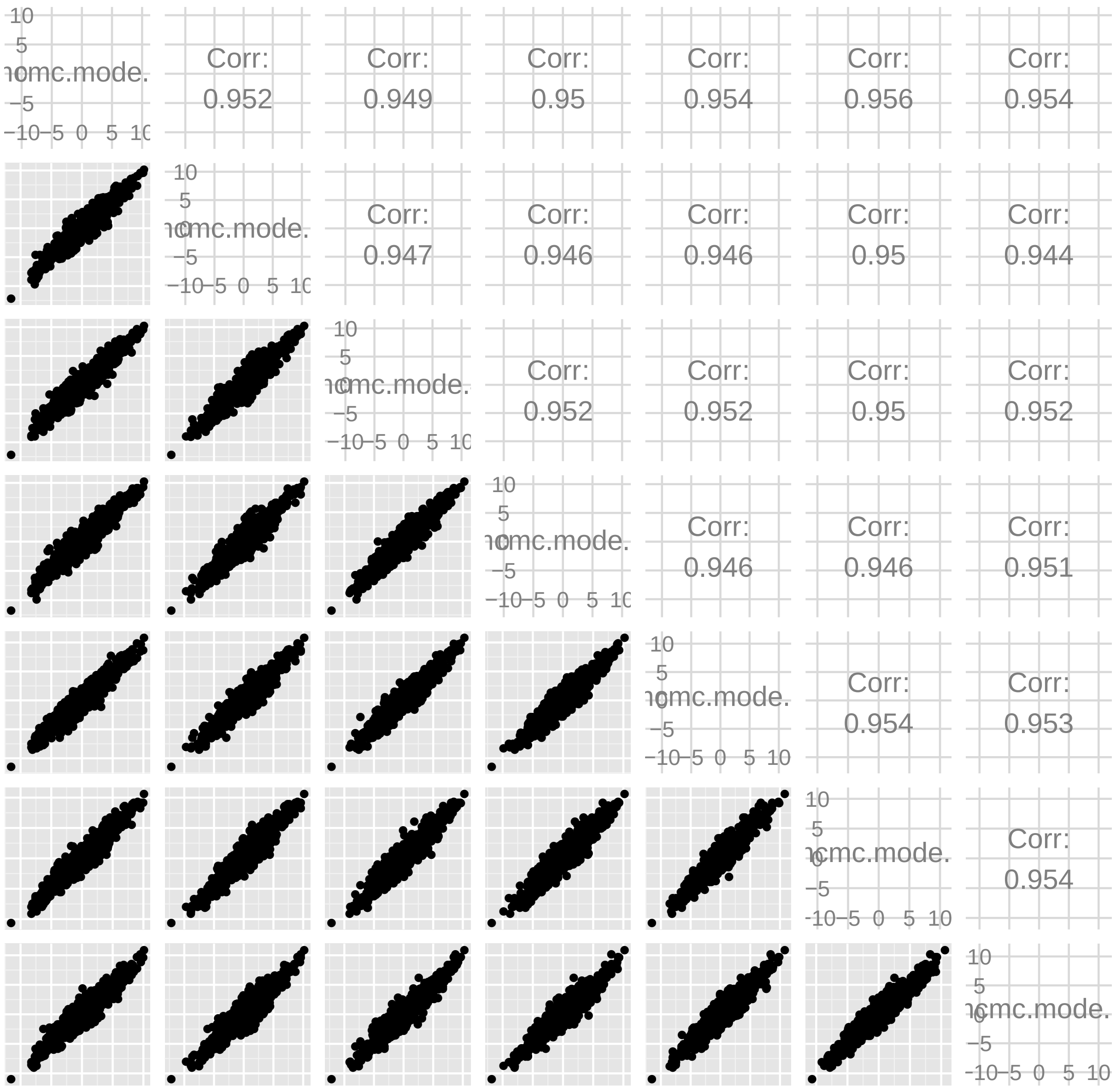
Table 1: Table continues below

|  | mcmc.mode.4 | mcmc.mode.5 | mcmc.mode.6 |
|---|---|---|---|
| **mcmc.mode.1** | 0.9475 | 0.9488 | 0.9495 |
| **mcmc.mode.2** | 0.9488 | 0.9498 | 0.9542 |
| **mcmc.mode.3** | 0.9473 | 0.9458 | 0.9464 |
| **mcmc.mode.4** | 12.4 | 0.9516 | 0.9522 |
| **mcmc.mode.5** | 11.76 | 12.31 | 0.9463 |
| **mcmc.mode.6** | 11.72 | 11.6 | 12.21 |
| **mcmc.mode.7** | 11.76 | 11.67 | 11.72 |
| **mcmc.mode.8** | 11.66 | 11.6 | 11.58 |

Table 2: Table continues below

|  | mcmc.mode.7 | mcmc.mode.8 |
|---|---|---|
| **mcmc.mode.1** | 0.9496 | 0.9516 |
| **mcmc.mode.2** | 0.9562 | 0.9536 |
| **mcmc.mode.3** | 0.9497 | 0.9439 |
| **mcmc.mode.4** | 0.95 | 0.9524 |
| **mcmc.mode.5** | 0.9463 | 0.9511 |
| **mcmc.mode.6** | 0.9541 | 0.953 |
| **mcmc.mode.7** | 12.36 | 0.9543 |
| **mcmc.mode.8** | 11.66 | 12.08 |

```r
library(ggplot2)
library(GGally)
ggpairs(mcmc_modes, columns = 3:ncol(mcmc_modes))
```



mcmc.mode.
Corr: 0.952
Corr: 0.949
Corr: 0.95
Corr: 0.954
Corr: 0.956
Corr: 0.954
Corr: 0.947
Corr: 0.946
Corr: 0.946
Corr: 0.95
Corr: 0.944
Corr: 0.952
Corr: 0.952
Corr: 0.95
Corr: 0.952
Corr: 0.946
Corr: 0.946
Corr: 0.951
Corr: 0.954
Corr: 0.953
Corr: 0.954
−10 −5 0 5 10

Ok, the models are all converging on the same point estimates. Why 0.95 correlation???

### Variance of blups

```
mcmc_itts <- mcmc_priors[!is.na(mcmc_priors$itt), ]
tapply(mcmc_itts$docility, mcmc_itts$type, var)
```

```
## mcmc.1 mcmc.2 mcmc.3 mcmc.4 mcmc.5 mcmc.6 mcmc.7 mcmc.8
##  19.20  19.30  19.16  19.12  19.20  19.13  19.26  19.14
```

```
tapply(mcmc_itts$docility, mcmc_itts$type, range)
```

```
## $mcmc.1
## [1] -18.35  18.04
##
## $mcmc.2
## [1] -20.60  19.28
##
## $mcmc.3
## [1] -20.06  18.31
##
## $mcmc.4
## [1] -20.17  18.76
##
## $mcmc.5
## [1] -19.19  17.83
##
## $mcmc.6
## [1] -19.51  18.44
##
## $mcmc.7
## [1] -18.99  18.26
##
## $mcmc.8
## [1] -22.01  17.66
```

No variation in variances either…