Signal Tandmobiel® Data
Time to emergence analysis using `bayessurvreg1`

Arnošt Komárek

July 5, 2005

In this document we describe the analysis of the Signal Tandmobiel® data presented in Section 6 of the paper

Komárek, A. and Lesaffre, E.
Bayesian accelerated failure time model for correlated interval-censored data with a normal mixture as an error distribution.

This article will be refered as Komárek and Lesaffre (2005) and can be found in the `doc` directory of the package `bayesSurv` as `KomarekLesaffre2005.pdf`. For the theory I refer therein.

All R commands presented in this document are available in the same directory as `tandmobMixture.R`.

This document should primarily serves as the source of the examples of usage of the functions

- `bayessurvreg1`
- `predictive`
- `bayesDensity`

Please, take also a look at the extensive help pages of these functions!

1 Model

The model we will consider is the following:

$$\begin{aligned}\log(T_{i,l} - 5) = {} & \beta_1\, girl_{i,l} + \beta_2\, dmf_{i,l} + \beta_3\, girl_{i,l} dmf_{i,l} + \\ & + \beta_4\, dmf_{i,l} lower4_{i,l} + \beta_5\, dmf_{i,l} upper5_{i,l} + \beta_6\, dmf_{i,l} lower5_{i,l} + \\ & + \beta_7\, girl_{i,l} lower4_{i,l} + \beta_8\, girl_{i,l} upper5_{i,l} + \beta_9\, girl_{i,l} lower5_{i,l} + \\ & + b_{i,1} + b_{i,2} lower4_{i,l} + b_{i,3} upper5_{i,l} + b_{i,4} lower5_{i,l} + \varepsilon_{i,l},\end{aligned}$$

where $l = 1, \ldots, 8$ indexes 8 permanent teeth (14, 24, 15, 25, 34, 44, 35, 45) in the mouth. Further, $girl_{i,l}$ is equal to 1 if the ith child is girl, $dmf_{i,l}$ is equal to 1 if the primary predecessor of the lth permanent tooth of the ith child has a positive dmf score. Variables $lower4_{i,l}$, $upper5_{i,l}$ and $lower5_{i,l}$, respectively are dummies for lower first premolars (teeth 34, 44), upper second premolars (teeth 15, 25) and lower second premolars (teeth 35, 45), respectively.

Time variable $T_{i,l}$ gives the emergence time of the lth permanent tooth of the ith child in years.

Finally, for the random effect vector $\boldsymbol{b}_i$ it is assumed: $E(b_{i,1}) = 0$, $E(b_{i,m}) = \gamma_m$ $(m = 2, 3, 4)$ and $\mathrm{var}(\boldsymbol{b}_i) = \mathbb{D}$.

Dental therminology remark: upper jaw = maxilla, lower jaw = mandible. Teeth $10 + s$ are upper right teeth, teeth $20 + s$ are upper left, teeth $30 + s$ are lower left and teeth $40 + s$ are lower right with $s = 1, \ldots, 8$. Dental index of a primary predecessor is obtained as $permanent + 40$ where $permanent$ is the index of the permanent tooth (e.g. primary tooth 54 is predecessor of the permanent tooth 14).

2 Initial operations

• Set the directories.

```
> anadir <- "/home/komari/win/work/papers/bayesaft/Tandmob/tana1/"
> docdir <- "/home/komari/win/work/papers/bayesaft/RforCRAN/tandmob/"
> dirsim <- character()
> dirsim[1] <- paste(anadir, "chain1small", sep = "")
> dirsim[2] <- paste(anadir, "chain2small", sep = "")
```

Load the `bayesSurv` package and the data

```
> library(bayesSurv)
> data(tandmob2)
```

Specify teeth we want to analyze (`dteeth` are corresponding deciduous teeth):

```
> pteeth <- c(14, 24, 34, 44, 15, 25, 35, 45)
> dteeth <- pteeth + 40
> nteeth <- length(pteeth)
```

Dummies for horizontally symmetric teeth (`upper4` = teeth 14 and 24, `lower4` = teeth 34 and 44, `upper5` = teeth 15 and 25, `lower5` = teeth 35 and 45) further used in the analysis:

```
> upper4 <- c(1, 1, 0, 0, 0, 0, 0, 0)
> lower4 <- c(0, 0, 1, 1, 0, 0, 0, 0)
> upper5 <- c(0, 0, 0, 0, 1, 1, 0, 0)
> lower5 <- c(0, 0, 0, 0, 0, 0, 1, 1)
```

Variables from the whole dataset that are needed or otherwise interesting (`timevars` are variables that determine the interval where the emergence was recorded, `dmfvars` give the caries status of corresponding deciduous teeth):

```
> childvars <- c("IDNR", "GENDER", "GENDERNum", "DOB", "PROVINCE",
+     "EDUC")
> timevars <- paste(c("EBEG.", "EEND."), rep(pteeth, rep(2, nteeth)),
+     sep = "")
> dmfvars <- paste("T", dteeth, ".DMF", sep = "")
> vars <- c(childvars, timevars, dmfvars)
> print(vars)
```

```
 [1] "IDNR"      "GENDER"    "GENDERNum" "DOB"       "PROVINCE"  "EDUC"
 [7] "EBEG.14"   "EEND.14"   "EBEG.24"   "EEND.24"   "EBEG.34"   "EEND.34"
[13] "EBEG.44"   "EEND.44"   "EBEG.15"   "EEND.15"   "EBEG.25"   "EEND.25"
[19] "EBEG.35"   "EEND.35"   "EBEG.45"   "EEND.45"   "T54.DMF"   "T64.DMF"
[25] "T74.DMF"   "T84.DMF"   "T55.DMF"   "T65.DMF"   "T75.DMF"   "T85.DMF"
```

Take only interesting variables from the data and change some names. To see explanation of the variables take a look at the help page of `tandmob2`.

```
> tandmob2 <- tandmob2[, vars]
```

Change the name of `GENDERNum` variable:

```
> names(tandmob2)[names(tandmob2) == "GENDERNum"] <- "GIRL"
> childvars[childvars == "GENDERNum"] <- "GIRL"
> vars[vars == "GENDERNum"] <- "GIRL"
```

Take only children for who deciduous dmf score is available for all studied teeth:

```
> is.dmf <- !is.na(tandmob2$T54.DMF) & !is.na(tandmob2$T64.DMF) &
+     !is.na(tandmob2$T74.DMF) & !is.na(tandmob2$T84.DMF) & !is.na(tandmob2$T55.DMF) &
+     !is.na(tandmob2$T65.DMF) & !is.na(tandmob2$T75.DMF) & !is.na(tandmob2$T85.DMF)
> tandmob2 <- tandmob2[is.dmf, ]
```

Subtract 5.0 from all time variables (i.e. time 0 in the analyzes = 5 years of age):

```
> startage <- 5
> tandmob2[, timevars] <- tandmob2[, timevars] - startage
```

For testing purposes I take only a subsample of 50 boys and 50 girls (their id's are given in the file `IDsampled50.dat`). In Komárek and Lesaffre (2005) an analysis of the subsample of 500 boys and 500 girls is presented (id's from `IDsampled500.dat`).

Now we take only subsample of 50 boys and 50 girls (their id's are given in the file `IDsampled50.dat`). In Komárek and Lesaffre (2004) an analysis of the subsample of 500 boys and 500 girls is presented (id's from `IDsampled500.dat`).

```
> nFromGender <- 50
> ids <- read.table(paste(docdir, "IDsampled", nFromGender, ".dat",
+     sep = ""), header = TRUE)[, 1]
> wanna <- match(tandmob2$IDNR, ids, nomatch = 0)
> wanna <- as.logical(wanna)
> wanna[wanna > 1] <- TRUE
> tandmob2 <- tandmob2[wanna, ]
```

Look at first few rows of the data:

```
> print(tandmob2[1:10, ])
```

```
    IDNR GENDER GIRL     DOB PROVINCE EDUC EBEG.14 EEND.14 EBEG.24 EEND.24
57    59    boy    0 04OCT89        4    0     2.4     5.7     2.4     5.7
126  130    boy    0 16OCT89        4    0     5.9      NA     5.9      NA
178  183   girl    1 26APR89        1    0     5.7      NA     5.7      NA
218  223    boy    0 31OCT89        0    0     5.0     5.9     5.9      NA
265  270   girl    1 16SEP89        1    0     5.6     6.6     5.6     6.6
268  273   girl    1 04JAN89        2    0     4.8     5.7     4.8     5.7
283  288    boy    0 07AUG89        2    0     4.5     6.1     4.5     6.1
290  295   girl    1 20JUL89        1    0     5.4     6.3     5.4     6.3
330  335    boy    0 21MAY89        3    2     4.8     5.8     4.8     5.8
370  375    boy    0 27NOV89        2    1     4.0      NA     4.0      NA
    EBEG.34 EEND.34 EBEG.44 EEND.44 EBEG.15 EEND.15 EBEG.25 EEND.25 EBEG.35
57      5.7      NA     2.4     5.7     2.4     5.7     5.7      NA     5.7
126     5.9      NA     5.9      NA     5.9      NA     5.9      NA     5.9
178     5.7      NA     5.7      NA     5.7      NA     5.7      NA     5.7
218     5.0     5.9     5.0     5.9     5.9      NA     5.9      NA     5.9
265     5.6     6.6     4.7     5.6     5.6     6.6     5.6     6.6     5.6
268     3.9     4.8     4.8     5.7     5.7     6.7     5.7     6.7     4.8
283     4.5     6.1     4.5     6.1     4.5     6.1     4.5     6.1     4.5
290     5.4     6.3     5.4     6.3     5.4     6.3     6.3      NA     6.3
330     4.0     4.8     4.8     5.8     4.8     5.8     4.8     5.8     4.0
370     3.2     4.0     3.2     4.0     4.0      NA     4.0      NA     4.0
    EEND.35 EBEG.45 EEND.45 T54.DMF T64.DMF T74.DMF T84.DMF T55.DMF T65.DMF
57       NA     5.7      NA       0       0       1       0       0       0
126      NA     5.9      NA       0       0       0       0       0       0
178      NA     5.7      NA       1       0       1       1       1       1
218      NA     5.9      NA       1       0       0       0       0       0
```

```
265     6.6     5.6     6.6       0       1       0       0       0       0
268     5.7     5.7     6.7       1       1       1       1       1       0
283     6.1     4.5     6.1       0       0       0       0       0       0
290      NA     6.3      NA       1       1       1       1       1       1
330     4.8     4.8     5.8       1       0       1       0       1       1
370      NA     4.0      NA       1       1       1       1       1       1
    T75.DMF T85.DMF
57        0       0
126       0       0
178       1       1
218       0       0
265       0       0
268       1       0
283       0       0
290       1       1
330       1       1
370       1       1
```

Now we have one row per child in the `data.frame` `tandmob2`. We reformat the data such that there is one row per tooth and create factors to determine the tooth:

```
> nchild <- dim(tandmob2)[1]
> longdata <- list()
> for (i in 1:length(childvars)) {
+     longdata[[i]] <- rep(tandmob2[[childvars[i]]], nteeth)
+ }
> names(longdata) <- childvars
> longdata <- as.data.frame(longdata)
> longdata$TOOTH <- as.factor(rep(pteeth, rep(nchild, nteeth)))
> longdata$UPPER4 <- rep(upper4, rep(nchild, nteeth))
> longdata$LOWER4 <- rep(lower4, rep(nchild, nteeth))
> longdata$UPPER5 <- rep(upper5, rep(nchild, nteeth))
> longdata$LOWER5 <- rep(lower5, rep(nchild, nteeth))
> dmf <- numeric()
> for (i in 1:nteeth) dmf <- c(dmf, tandmob2[[dmfvars[i]]])
> ebeg <- numeric()
> for (i in 1:nteeth) ebeg <- c(ebeg, tandmob2[[paste("EBEG.",
+     pteeth[i], sep = "")]])
> eend <- numeric()
> for (i in 1:nteeth) eend <- c(eend, tandmob2[[paste("EEND.",
+     pteeth[i], sep = "")]])
> longdata$DMF <- dmf
> longdata$EBEG <- ebeg
> longdata$EEND <- eend
> rm(list = c("dmf", "ebeg", "eend"))
> longdata <- longdata[order(longdata$IDNR), ]
```

Look at first few rows (2 children) of reformatted data:

```
> print(longdata[1:16, ])
```

```
    IDNR GENDER GIRL     DOB PROVINCE EDUC TOOTH UPPER4 LOWER4 UPPER5 LOWER5
1     59    boy    0 04OCT89        4    0    14      1      0      0      0
101   59    boy    0 04OCT89        4    0    24      1      0      0      0
201   59    boy    0 04OCT89        4    0    34      0      1      0      0
301   59    boy    0 04OCT89        4    0    44      0      1      0      0
401   59    boy    0 04OCT89        4    0    15      0      0      1      0
501   59    boy    0 04OCT89        4    0    25      0      0      1      0
```

```
601   59    boy    0 04OCT89       4    0    35      0      0      0      1
701   59    boy    0 04OCT89       4    0    45      0      0      0      1
2    130    boy    0 16OCT89       4    0    14      1      0      0      0
102  130    boy    0 16OCT89       4    0    24      1      0      0      0
202  130    boy    0 16OCT89       4    0    34      0      1      0      0
302  130    boy    0 16OCT89       4    0    44      0      1      0      0
402  130    boy    0 16OCT89       4    0    15      0      0      1      0
502  130    boy    0 16OCT89       4    0    25      0      0      1      0
602  130    boy    0 16OCT89       4    0    35      0      0      0      1
702  130    boy    0 16OCT89       4    0    45      0      0      0      1
    DMF EBEG EEND
1     0  2.4  5.7
101   0  2.4  5.7
201   1  5.7   NA
301   0  2.4  5.7
401   0  2.4  5.7
501   0  5.7   NA
601   0  5.7   NA
701   0  5.7   NA
2     0  5.9   NA
102   0  5.9   NA
202   0  5.9   NA
302   0  5.9   NA
402   0  5.9   NA
502   0  5.9   NA
602   0  5.9   NA
702   0  5.9   NA
```

3 Finding reasonable values for hyperparameters and initial values

We fit the log-normal AFT model without any random effects to get initial values for model parameters and reasonable values for hyperparameters:

```
> ifit <- survreg(Surv(EBEG, EEND, type = "interval2") ~ GIRL *
+     DMF + (DMF + GIRL) * (LOWER4 + UPPER5 + LOWER5), dist = "lognormal",
+     data = longdata)
> summary(ifit)

Call:
survreg(formula = Surv(EBEG, EEND, type = "interval2") ~ GIRL *
    DMF + (DMF + GIRL) * (LOWER4 + UPPER5 + LOWER5), data = longdata,
    dist = "lognormal")
              Value Std. Error       z         p
(Intercept)  1.7410     0.0395  44.046  0.00e+00
GIRL        -0.0816     0.0464  -1.757  7.89e-02
DMF         -0.1154     0.0479  -2.410  1.59e-02
LOWER4       0.0388     0.0542   0.717  4.74e-01
UPPER5       0.1734     0.0556   3.116  1.83e-03
LOWER5       0.2214     0.0565   3.918  8.92e-05
GIRL:DMF     0.1113     0.0432   2.578  9.94e-03
DMF:LOWER4  -0.0417     0.0586  -0.711  4.77e-01
DMF:UPPER5  -0.0623     0.0605  -1.030  3.03e-01
DMF:LOWER5  -0.0632     0.0609  -1.037  3.00e-01
GIRL:LOWER4 -0.0614     0.0589  -1.043  2.97e-01
GIRL:UPPER5  0.0264     0.0609   0.434  6.64e-01
GIRL:LOWER5 -0.0459     0.0615  -0.747  4.55e-01
Log(scale)  -1.3850     0.0378 -36.624 1.21e-293

Scale= 0.250

Log Normal distribution
Loglik(model)= -910.1   Loglik(intercept only)= -955.7
        Chisq= 91.2 on 12 degrees of freedom, p= 2.9e-14
Number of Newton-Raphson Iterations: 5
n= 800
```

4 Specification of priors

To specify correctly the prior hyperparameters for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ parameters we have to know how the covariates are sorted in the design matrix. Normally, the same order should be used as in the `formula` specification. However, one never knows...

The following command returns the design matrix and we look at first few rows to see how are the covariates sorted in the columns. We also define the variable `nregres` (number of covariates). The same model `formula` is used as in the future function call.

```
> X <- bayessurvreg1(Surv(EBEG, EEND, type = "interval2") ~ GIRL *
+     DMF + (DMF + GIRL) * (LOWER4 + UPPER5 + LOWER5) + cluster(IDNR),
+     random = ~LOWER4 + UPPER5 + LOWER5, data = longdata[1:80,
+         ], onlyX = TRUE)
```

Print first few lines of X to see how `beta`s are sorted. Remember that both $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ parameters from Komárek and Lesaffre (2005) are put into one long vector `beta` in the R functions.

```
> print(X[1:5, ])
```

```
    GIRL DMF LOWER4 UPPER5 LOWER5 GIRL:DMF DMF:LOWER4 DMF:UPPER5 DMF:LOWER5
1      0   0      0      0      0        0          0          0          0
101    0   0      0      0      0        0          0          0          0
201    0   1      1      0      0        0          1          0          0
301    0   0      1      0      0        0          0          0          0
401    0   0      0      1      0        0          0          0          0
    GIRL:LOWER4 GIRL:UPPER5 GIRL:LOWER5
1             0           0           0
101           0           0           0
201           0           0           0
301           0           0           0
401           0           0           0
```

```
> nregres <- dim(X)[2]
> nobs <- dim(longdata)[1]
```

So we see that `beta[1:12]` $= (\beta(girl), \beta(dmf), \gamma(lower4), \gamma(upper5), \gamma(lower5), \beta(girl:dmf), \beta(dmf:lower4), \beta(dmf:upper5), \beta(dmf:lower5), \beta(girl:lower4), \beta(girl:upper5), \beta(girl:lower5))'$.

Notice that the random effect vector is now 4-dimensional, i.e. $\boldsymbol{b}_i = (b_{i,1}, b_{i,2}, b_{i,3}, b_{i,4})'$ and $E(b_{i,1}) = 0$, $E(b_{i,2}) = \gamma(lower4)$, $E(b_{i,3}) = \gamma(upper5)$, $E(b_{i,4}) = \gamma(lower5)$.

4.1 Priors for the mixture

```
> prior <- list()
```

Prior for the number of mixture components k will be truncated Poisson(λ, k_{max}) with $k_{max} = 30$ and $\lambda = 5$. Alternative prior distribution would be uniform specified by

```
prior$k.prior = ''uniform''
```

```
> prior$kmax <- 30
> prior$k.prior <- "poisson"
> prior$poisson.k <- 5
```

Prior for mixture weights $w_1, \ldots, w_k$ will be Dirichlet($\delta, \ldots, \delta$) with $\delta = 1$.

```
> prior$dirichlet.w <- 1
```

Prior for mixture means $\mu_1, \ldots, \mu_k$ will be $\mathrm{N}(\xi, \kappa)$ with $\xi = 1.8$ (taken from `survreg(dist = "lognormal")` fit (approx intercept)) and $\kappa = 0.75^2 \approx (3 \times 0.25)^2$ (0.25 was approximately estimated scale parameter by `survreg`).

```
> prior$mean.mu <- 1.8
> prior$var.mu <- 0.75^2
```

Prior for mixture inverse-variances $\sigma_1^{-2}, \ldots, \sigma_k^{-2}$ will be $\mathrm{Gamma}(\zeta, \eta)$ and prior for η will be $\mathrm{Gamma}(g, h)$, with $\zeta = 2.0$, $g = 0.2$ and $h = 0.1$.

```
> prior$shape.invsig2 <- 2
> prior$shape.hyper.invsig2 <- 0.2
> prior$rate.hyper.invsig2 <- 0.1
```

Probabilities of the split move (given current value of k) will be always 0.5 except when $k = 1$ or $k = k_{max}$.

```
> prior$pi.split <- c(1, rep(0.5, prior$kmax - 2), 0)
```

Probabilities of the birth move (given current value of k) will be always 0.5 except when $k = 1$ or $k = k_{max}$.

```
> prior$pi.birth <- c(1, rep(0.5, prior$kmax - 2), 0)
```

The last component of the `list prior` should be always set to `FALSE`. Its value equal to `TRUE` served only for some exploratory purposes of the author.

```
> prior$Eb0.depend.mix <- FALSE
```

Look how it looks like:

```
> print(prior)

$kmax
[1] 30

$k.prior
[1] "poisson"

$poisson.k
[1] 5

$dirichlet.w
[1] 1

$mean.mu
[1] 1.8

$var.mu
[1] 0.5625

$shape.invsig2
[1] 2

$shape.hyper.invsig2
```

```
[1] 0.2

$rate.hyper.invsig2
[1] 0.1

$pi.split
 [1] 1.0 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
[20] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.0

$pi.birth
 [1] 1.0 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
[20] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.0

$Eb0.depend.mix
[1] FALSE
```

4.2 Priors for regression parameters $\boldsymbol{\beta}$ and means of random effects $\boldsymbol{\gamma}$

In this section we specify the prior hyperparameters for `beta[1:12]` $= (\beta(girl),\ \beta(dmf),\ \gamma(lower4),\ \gamma(upper5),\ \gamma(lower5),\ \beta(girl:dmf),\ \beta(dmf:lower4),\ \beta(dmf:upper5),\ \beta(dmf:lower5),\ \beta(girl:lower4),\ \beta(girl:upper5),\ \beta(girl:lower5))'$ and the way these parameters will be updated. In other words, we define a `list` with all the information needed.

```
> prior.beta <- list()
```

Here we specify prior means (all will be zero) and prior variances (all will be 100) for the `beta` parameters.

```
> prior.beta$mean.prior <- rep(0, nregres)
> prior.beta$var.prior <- rep(100, nregres)
```

Further, we do not have to do anything, i.e. default way to update the `beta` parameters will be used. That is by the Gibbs move in two blocks (all fixed effects $\boldsymbol{\beta}$ in one block and means $\boldsymbol{\gamma}$ of all random effects in the second block).

Look how the list looks like:

```
> print(prior.beta)

$mean.prior
 [1] 0 0 0 0 0 0 0 0 0 0 0 0

$var.prior
 [1] 100 100 100 100 100 100 100 100 100 100 100 100
```

4.3 Prior specification for the random effects $\boldsymbol{b}_i$ related parameters

Now, we have to specify the prior for the variance matrix $\mathbb{D}$ of the random effect vector $\boldsymbol{b}_i$. Again, we define a `list` with all the information:

```
> prior.b <- list()
```

We use an inverse-Wishart prior for $\mathbb{D}$ with $\tau = 4$ degrees of freedom and a scale matrix $\mathbb{S}$ equal to

$$\mathbb{S} = 0.002 \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Observe that only the lower triangle of $\mathbb{S}$ is given to `prior$scale.D`.

```
> prior.b$df.D <- 4
> prior.b$scale.D <- 0.002 * c(1, 0, 0, 0, 1, 0, 0, 1, 0, 1)
> prior.b$prior.D <- "inv.wishart"
```

Finally, we specify how the individual random effects will be updated (using the Gibbs move):

```
> prior.b$type.upd <- "gibbs"
```

Look how the list looks like:

```
> print(prior.b)

$df.D
[1] 4

$scale.D
 [1] 0.002 0.000 0.000 0.000 0.002 0.000 0.000 0.002 0.000 0.002

$prior.D
[1] "inv.wishart"

$type.upd
[1] "gibbs"
```

4.4 Parameters to perform reversible jumps

```
> prop.revjump <- list()
```

Type of the algorithm:

```
> prop.revjump$algorithm <- "correlated.av"
```

Parameters of a moody ring (ϵ, *delta*, see paper Brooks et al. (2003) for details). Remember, ϵ = time dependence, δ = component dependence.

```
> prop.revjump$moody.ring <- c(0.1, 0.05)
```

Transformation of a canonical seed for split-combine move:

```
> prop.revjump$transform.split.combine <- "brooks"
> prop.revjump$transform.split.combine.parms <- c(2, 2, 2, 2, 1,
+     1)
```

Transformation of a canonical seed for birth-death move:

```
> prop.revjump$transform.birth.death <- "richardson.green"
```

Look how it looks like:

```
> print(prop.revjump)

$algorithm
[1] "correlated.av"

$moody.ring
[1] 0.10 0.05
```

```
$transform.split.combine
[1] "brooks"

$transform.split.combine.parms
[1] 2 2 2 2 1 1

$transform.birth.death
[1] "richardson.green"
```

5 Specification of initial values for the MCMC

We give two sets of initial values to run two chains. Undefined initials are sampled automatically by the program.

5.1 Initials for chain 1

```
> init1 <- list()
```

Iteration number of the nulth iteration:

```
> init1$iter <- 0
```

Initial mixture (from `survreg(dist = "lognormal")`). It will have one component with $w_1 = 1$, $\mu_1 = 1.8$ and $\sigma_1^2 = 0.25^2$.

```
> init1$mixture <- c(1, 1, rep(0, prior$kmax - 1), 1.8, rep(0,
+     prior$kmax - 1), 0.25^2, rep(0, prior$kmax - 1))
```

Initial regression parameters $\boldsymbol{\beta}$ and means of random effects $\boldsymbol{\gamma}$ (from `survreg(dist = "lognormal")`):

```
> init1$beta <- c(-0.09, -0.11, -0.01, 0.16, 0.17, 0.05, 0.02,
+     0.01, 0.03, -0.02, 0.01, 0)
```

Initial covariance matrix $\mathbb{D}$ of the random effects will be identity matrix (only its lower triangle)

```
> init1$D <- c(1, 0, 0, 0, 1, 0, 0, 1, 0, 1)
```

Initial values of individual random effects for each child:

```
> b0 <- rep(0, nchild)
> b1 <- rnorm(nchild, 0, 0.3)
> b2 <- rnorm(nchild, 0.16, 0.3)
> b3 <- rnorm(nchild, 0.16, 0.3)
> init1$b <- as.numeric(rbind(b0, b1, b2, b3))
```

Initial (augmented) log(event) times – let the program sample them:

```
> init1$y <- NULL
```

Initial component pertinence of the observations to the mixture (all observations belong to the first component):

```
> init1$r <- rep(1, nobs)
```

Initial value of a hyperparameter η (sample it from a prior distribution):

```
> init1$otherp <- rgamma(1, shape = prior$shape.hyper.invsig2,
+     rate = prior$rate.hyper.invsig2)
```

Initial values of canonical variables for reversible move (sample it from a uniform distribution):

```
> init1$u <- c(runif(1), 0, 0, runif(3 * (prior$kmax - 1)))
```

5.2 Initials for chain 2

```
> init2 <- list()
```

Iteration number of the nulth iteration:

```
> init2$iter <- 0
```

Initial mixture. It will have one component with $w_1 = 1$, $\mu_1 = 1.7$ and $\sigma_1^2 = 0.2^2$.

```
> init2$mixture <- c(1, 1, rep(0, prior$kmax - 1), 1.7, rep(0,
+     prior$kmax - 1), 0.2^2, rep(0, prior$kmax - 1))
```

Initial regression parameters $\boldsymbol{\beta}$ and means of random effects $\boldsymbol{\gamma}$ (all zeros):

```
> init2$beta <- rep(0, nregres)
```

Initial covariance matrix $\mathbb{D}$ of the random effects. Now we take such matrix $\mathbb{D}$ that $\mathrm{var}(b_{i,m}) = 0.01$ $(m = 1, 2, 3, 4)$ and $\mathrm{corr}(b_{i,m}, b_{i,s}) = 0.2$ $(m \neq s = 1, 2, 3, 4)$.

```
> varb <- c(0.01, 0.01, 0.01, 0.01)
> corb <- c(0.2, 0.2, 0.2, 0.2, 0.2, 0.2)
> Corb <- diag(4)
> Corb[lower.tri(Corb, diag = FALSE)] <- corb
> Corb[upper.tri(Corb, diag = FALSE)] <- t(Corb)[upper.tri(t(Corb),
+     diag = FALSE)]
> Covb <- diag(sqrt(varb)) %*% Corb %*% diag(sqrt(varb))
> init2$D <- Covb[lower.tri(Covb, diag = TRUE)]
```

Initial values of individual random effects for each child:

```
> b0 <- rnorm(nchild, 0, sqrt(init2$D[1]))
> b1 <- rnorm(nchild, 0, sqrt(init2$D[5]))
> b2 <- rnorm(nchild, 0.16, sqrt(init2$D[8]))
> b3 <- rnorm(nchild, 0.16, sqrt(init2$D[10]))
> init2$b <- as.numeric(rbind(b0, b1, b2, b3))
```

Initial (augmented) log(event) times – let the program sample them:

```
> init2$y <- NULL
```

Initial component pertinence of the observations to the mixture (all observations belong to the first component):

```
> init2$r <- rep(1, nobs)
```

Initial value of a hyperparameter η (sample it from a prior distribution):

```
> init2$otherp <- rgamma(1, shape = prior$shape.hyper.invsig2,
+     rate = prior$rate.hyper.invsig2)
```

Initial values of canonical variables for reversible move (sample it from a uniform distribution):

```
> init2$u <- c(runif(1), 0, 0, runif(3 * (prior$kmax - 1)))
```

6 Running the MCMC simulation

Now we are ready to run the MCMC to sample from the posterior distribution.

Here we define which quantities that are not neccessarily needed for the inference will be stored.

```
> store <- list(y = FALSE, r = FALSE, u = FALSE, b = FALSE, MHb = FALSE,
+     regresres = FALSE)
```

How long simulation we want to run? For illustration purposes, only limited simulation is specified here.

```
> nsimul <- list(niter = 1000, nthin = 3, nburn = 500, nwrite = 500)
```

For the analysis presented in Komárek and Lesaffre (2005) we used

```
> nsimul <- list(niter = 15000, nthin = 3, nburn = 1500, nnoadapt = 0, nwrite = 1000)
```

which performed $3 \times 1\,500$ iterations of burn-in and additionally $3 \times 13\,500$ iterations from which each 3rd value was stored. Further, after cumulating 1 000 sampled values, these were stored on a disk.

Define directories where first and second chain will be stored (create them first):

```
> dir.create("tandchain1test")
> dir.create("tandchain2test")
> dirsimtest <- character()
> dirsimtest[1] <- paste(getwd(), "/tandchain1test", sep = "")
> dirsimtest[2] <- paste(getwd(), "/tandchain2test", sep = "")
```

Run the simulation for the first and the second chain:

```
> sim1 <- bayessurvreg1(Surv(EBEG, EEND, type = "interval2") ~
+     GIRL * DMF + (DMF + GIRL) * (LOWER4 + UPPER5 + LOWER5) +
+         cluster(IDNR), random = ~LOWER4 + UPPER5 + LOWER5, data = longdata,
+     dir = dirsimtest[1], nsimul = nsimul, prior = prior, init = init1,
+     prop.revjump = prop.revjump, prior.beta = prior.beta, prior.b = prior.b,
+     store = store)

Simulation started on                       Tue Jul  5 15:57:50 2005
Iteration 500
Simulation without adaptation finished on   Tue Jul  5 15:57:55 2005   (iteration 500)
Iteration 1000
Simulation finished on                      Tue Jul  5 15:58:01 2005   (iteration 1000)

> sim2 <- bayessurvreg1(Surv(EBEG, EEND, type = "interval2") ~
+     GIRL * DMF + (DMF + GIRL) * (LOWER4 + UPPER5 + LOWER5) +
+         cluster(IDNR), random = ~LOWER4 + UPPER5 + LOWER5, data = longdata,
+     dir = dirsimtest[2], nsimul = nsimul, prior = prior, init = init2,
+     prop.revjump = prop.revjump, prior.beta = prior.beta, prior.b = prior.b,
+     store = store)

Simulation started on                       Tue Jul  5 15:58:01 2005
Iteration 500
Simulation without adaptation finished on   Tue Jul  5 15:58:06 2005   (iteration 500)
Iteration 1000
Simulation finished on                      Tue Jul  5 15:58:11 2005   (iteration 1000)
```

7 Running additional MCMC simulation to compute predictive quantities

First we have to define covariate values for which we want to do a prediction. These will be all combinations of boy/girl × dmf>0/dmf=0 × upper4/lower4/upper5/lower5:

```
> newIDNR = c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4)
> newGIRL = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1)
> newUPPER4 = c(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0)
> newLOWER4 = c(0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0)
> newUPPER5 = c(0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0)
> newLOWER5 = c(0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1)
> newDMF = c(0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1)
> newEBEG = rep(1, 16)
> newEEND = rep(NA, 16)
> preddata <- data.frame(IDNR = newIDNR, GIRL = newGIRL, UPPER4 = newUPPER4,
+     LOWER4 = newLOWER4, UPPER5 = newUPPER5, LOWER5 = newLOWER5,
+     DMF = newDMF, EBEG = newEBEG, EEND = newEEND)
> print(preddata)

   IDNR GIRL UPPER4 LOWER4 UPPER5 LOWER5 DMF EBEG EEND
1     1    0      1      0      0      0   0    1   NA
2     1    0      0      1      0      0   0    1   NA
3     1    0      0      0      1      0   0    1   NA
4     1    0      0      0      0      1   0    1   NA
5     2    0      1      0      0      0   1    1   NA
6     2    0      0      1      0      0   1    1   NA
7     2    0      0      0      1      0   1    1   NA
8     2    0      0      0      0      1   1    1   NA
9     3    1      1      0      0      0   0    1   NA
10    3    1      0      1      0      0   0    1   NA
11    3    1      0      0      1      0   0    1   NA
12    3    1      0      0      0      1   0    1   NA
13    4    1      1      0      0      0   1    1   NA
14    4    1      0      1      0      0   1    1   NA
15    4    1      0      0      1      0   1    1   NA
16    4    1      0      0      0      1   1    1   NA
```

Observe that variables `EBEG` and `EEND` may contain whatever (under the condition that the values are acceptable for `Surv`).

Further, we specify what we want to predict (with this, survivor function and hazard function). Also, specify whether sampled quantities should be stored, otherwise, only quantiles and predictive means are computed (which usually suffice).

```
> predict <- list(Et = TRUE, t = FALSE, Surv = TRUE, hazard = TRUE,
+     cum.hazard = FALSE)
> store <- list(Et = FALSE, t = FALSE, Surv = FALSE, hazard = FALSE,
+     cum.hazard = FALSE)
```

Grid of values in which predictive survivor and hazard curves should be computed. Observe that this grid takes into account the fact we shifted the response by 5 years in the model. Here we return back to the original time scale, i.e. we wish to see predictive curves between 6.5 and 13 years of age.

```
> grid <- seq(6.5, 13, by = 0.1)
```

Run MCMC simulation to sample from the predictive distribution (only chain 1 will be used here). Observe how we used parameter `time0` in the function call to specify that the response was shifted in the model by `startage` = 5.

```
> simulp1 <- predictive(Surv(EBEG, EEND, type = "interval2") ~
+     GIRL * DMF + (DMF + GIRL) * (LOWER4 + UPPER5 + LOWER5) +
+         cluster(IDNR), random = ~LOWER4 + UPPER5 + LOWER5, time0 = startage,
+     data = preddata, dir = dirsimtest[1], skip = 0, by = 1, predict = predict,
+     store = store, grid = grid, type = "mixture")

Simulation started on                       Tue Jul  5 15:58:11 2005
Reading mixture files.
Reading /home/komari/win/work/papers/bayesaft/RforCRAN/tandmob/tandchain1test/beta.sim
Reading /home/komari/win/work/papers/bayesaft/RforCRAN/tandmob/tandchain1test/D.sim
Iteration 500
Computing quantiles.
 observ. 0  Done.
 observ. 1  Done.
 observ. 2  Done.
 observ. 3  Done.
 observ. 4  Done.
 observ. 5  Done.
 observ. 6  Done.
 observ. 7  Done.
 observ. 8  Done.
 observ. 9  Done.
 observ. 10  Done.
 observ. 11  Done.
 observ. 12  Done.
 observ. 13  Done.
 observ. 14  Done.
 observ. 15  Done.

 observ. 0  Done.
 observ. 1  Done.
 observ. 2  Done.
 observ. 3  Done.
 observ. 4  Done.
 observ. 5  Done.
 observ. 6  Done.
 observ. 7  Done.
 observ. 8  Done.
 observ. 9  Done.
 observ. 10  Done.
 observ. 11  Done.
 observ. 12  Done.
 observ. 13  Done.
 observ. 14  Done.
 observ. 15  Done.

Storing quantiles.
Simulation finished on                      Tue Jul  5 15:58:13 2005
```

In a directory `./tandchain1test` few new files should appear:

- `quantS1.sim` – `quantS8.sim`;
- `quanthazard1.sim` – `quanthazard8.sim`;
- `quantET.sim`.

Files `quantS*.sim` and `quanthazard*.sim` contain pointwise (evaluated at the grid specified above) posterior predictive quantiles and means of the survivor and hazard function for each combination of

covariates specified in `preddata`. File `quantET.sim` contains posterior predictive quantiles and mean for expected survivor time of each combination of covariates. Note that

1. There is one file per survivor/hazard function and per covariate combination. Indeces of these files (1, ..., 8) correspond to rows of `preddata`. Structure of these files is following

1st row	=	grid values
2nd row	=	post. predictive 0% quantile (minimum)
3rd row	=	post. predictive 2.5% quantile
4th row	=	post. predictive 50% quantile (median)
5th row	=	post. predictive 97.5% quantile
6th row	=	post. predictive 100% quantile (maximum)
last row	=	post. predictive mean

2. There is only one file for posterior predictive expected survivor times and all combinations of covariates (`quantET.sim`). Structure of this file is following:

1st row	=	character labels ET1 - ET8 indicating that each column corresponds to one covariate combination
remaining rows	=	same as for `quantS*.sim` or `quanthazard*.sim`

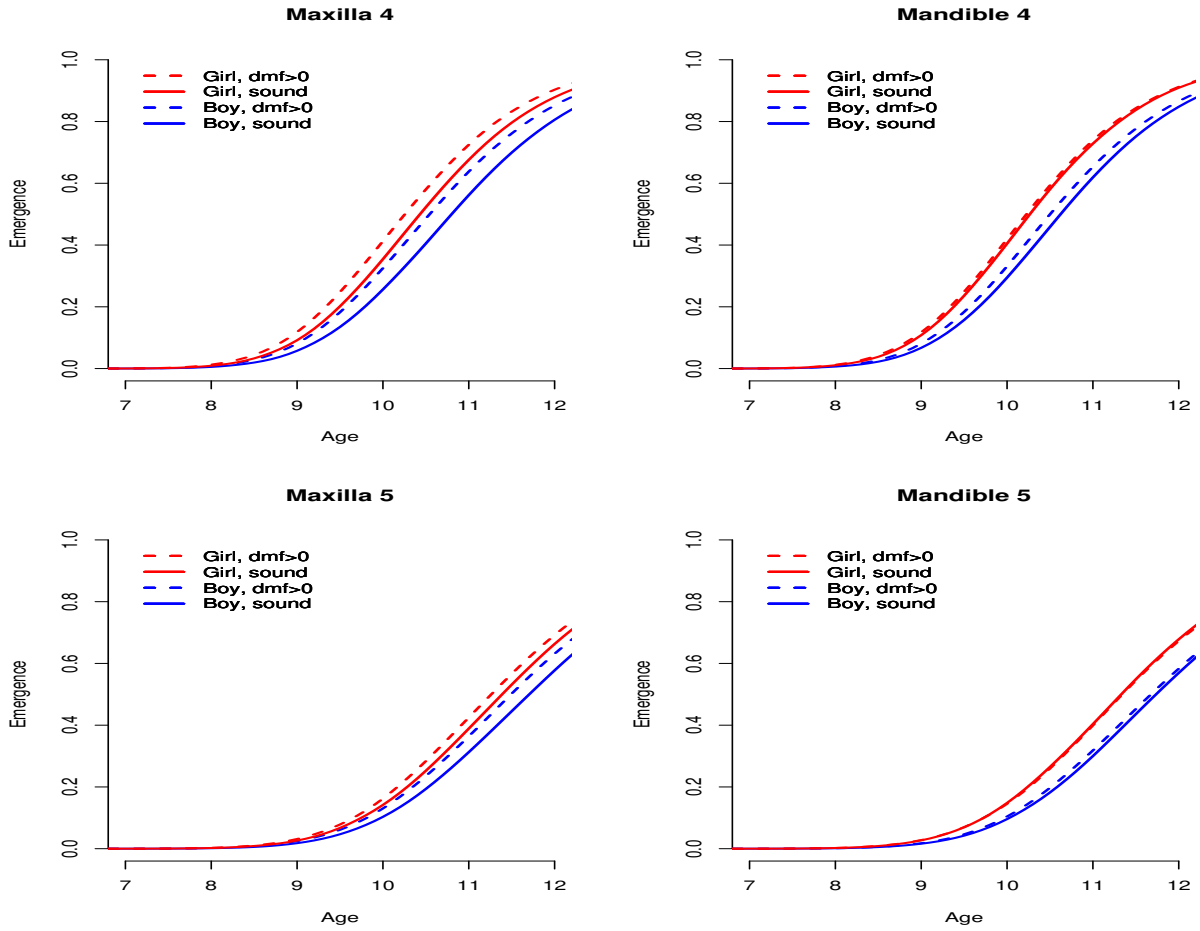
You might specify also other quantiles (parameter `quantile` in function `predictive`) to be computed. Posterior predictive mean is always computed and stored on the last row.

8 Drawing of posterior predictive emergence curves

Here we draw predictive emergence curves (cumulative distribution functions), separately for each pair of horizontally symmetric teeth (14+24, 34+44, 15+25, 35+45), for boys and girls and finally depending on the *dmf* score of the primary predecessor. We use simulated predictive curves based on the first chain. See Figure 1 for the result.

```
> ch <- 1
> line <- numeric(13)
> line[1] <- 1
> line[5] <- 2
> line[9] <- 1
> line[13] <- 2
> col <- character(13)
> col[1] <- "blue"
> col[5] <- "blue"
> col[9] <- "red"
> col[13] <- "red"
> title <- c("Maxilla 4", "Mandible 4", "Maxilla 5", "Mandible 5")
> lwd <- 2
> xlim <- c(7, 12)
> par(mfrow = c(2, 2))
> for (i in 1:4) {
+     for (j in c(1, 5, 9, 13)) {
+         gridS <- scan(paste(dirsim[ch], "/quantS", (i - 1) +
+             j, ".sim", sep = ""), nlines = 1)
+         Sfun <- read.table(paste(dirsim[ch], "/quantS", (i -
+             1) + j, ".sim", sep = ""), header = TRUE)
+         rownames(Sfun) <- c("0%", "2.5%", "50%", "97.5%", "100%",
+             "mean")
+         if (j == 1)
+             plot(gridS, 1 - Sfun["mean", ], type = "l", lty = line[1],
+                 ylim = c(0, 1), xlab = "Age", ylab = "Emergence",
+                 bty = "n", lwd = lwd, col = col[1], xlim = xlim)
+         else lines(gridS, 1 - Sfun["mean", ], lty = line[j],
+             lwd = lwd, col = col[j])
+         legend(7, 1, legend = c("Girl, dmf>0", "Girl, sound",
+             "Boy, dmf>0", "Boy, sound"), lty = line[c(13, 9,
+             5, 1)], col = col[c(13, 9, 5, 1)], lwd = lwd, bty = "n")
+     }
+     title(main = title[i])
+ }
```

Figure 1: Predictive emergence curves.



9 Summary statistics

Check how many chains we have

```
> nchains <- length(dirsim)
> print(nchains)

[1] 2
```

Read sampled $\mathbb{D}$ matrix and perform some transformation on it. Data frames in `covb` will store chains for standard deviations of $b_{i,1}, b_{i,2}, b_{i,3}, b_{i,4}$ and correlations among b's. Further, to see an inter-teeth relationship we define $d_{i,1} = b_{i,1}$, $d_{i,2} = b_{i,1} + b_{i,2}$, $d_{i,3} = b_{i,1} + b_{i,3}$ and $d_{i,4} = b_{i,1} + b_{i,4}$, i.e. each d is actually the child-tooth-specific (apart the horizontal symmetry) shift of log-emergence time ($d_{i,1}$ for teeth 14, 24, $d_{i,2}$ for teeth 34, 44, $d_{i,3}$ for teeth 15, 25 and $d_{i,4}$ for teeth 35, 45). Date frames in `covt` then store chains for standard deviations of $d_{i,1}, d_{i,2}, d_{i,3}, d_{i,4}$ and correlations among d's.

```
> covb <- list()
> covt <- list()
> for (ch in 1:nchains) {
+     dd <- read.table(paste(dirsim[ch], "/D.sim", sep = ""), header = TRUE)
+     varb <- cbind(dd$D.1.1, dd$D.2.2, dd$D.3.3, dd$D.4.4)
+     sdb <- sqrt(varb)
+     corb12 <- dd$D.2.1/sqrt(dd$D.1.1 * dd$D.2.2)
+     corb13 <- dd$D.3.1/sqrt(dd$D.1.1 * dd$D.3.3)
+     corb14 <- dd$D.4.1/sqrt(dd$D.1.1 * dd$D.4.4)
+     corb23 <- dd$D.3.2/sqrt(dd$D.2.2 * dd$D.3.3)
+     corb24 <- dd$D.4.2/sqrt(dd$D.2.2 * dd$D.4.4)
+     corb34 <- dd$D.4.3/sqrt(dd$D.3.3 * dd$D.4.4)
+     covb[[ch]] <- data.frame(sdb, corb12, corb13, corb14, corb23,
+         corb24, corb34)
+     colnames(covb[[ch]]) <- c(paste("sd", 1:4, sep = ""), paste("cor",
+         c(12, 13, 14, 23, 24, 34), sep = ""))
+     sdt1 <- sqrt(dd$D.1.1)
+     sdt2 <- sqrt(dd$D.1.1 + dd$D.2.2 + 2 * dd$D.2.1)
+     sdt3 <- sqrt(dd$D.1.1 + dd$D.3.3 + 2 * dd$D.3.1)
+     sdt4 <- sqrt(dd$D.1.1 + dd$D.4.4 + 2 * dd$D.4.1)
+     cort12 <- (dd$D.1.1 + dd$D.2.1)/(sdt1 * sdt2)
+     cort13 <- (dd$D.1.1 + dd$D.3.1)/(sdt1 * sdt3)
+     cort14 <- (dd$D.1.1 + dd$D.4.1)/(sdt1 * sdt4)
+     cort23 <- (dd$D.1.1 + dd$D.2.1 + dd$D.3.1 + dd$D.3.2)/(sdt2 *
+         sdt3)
+     cort24 <- (dd$D.1.1 + dd$D.2.1 + dd$D.4.1 + dd$D.4.2)/(sdt2 *
+         sdt4)
+     cort34 <- (dd$D.1.1 + dd$D.3.1 + dd$D.4.1 + dd$D.4.3)/(sdt3 *
+         sdt4)
+     covt[[ch]] <- data.frame(sdt1, sdt2, sdt3, sdt4, cort12,
+         cort13, cort14, cort23, cort24, cort34)
+ }
```

Read sampled $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ parameters:

```
> beta <- list()
> for (ch in 1:nchains) {
+     beta[[ch]] <- read.table(paste(dirsim[ch], "/beta.sim", sep = ""),
+         header = TRUE)
+ }
```

Read sampled numbers of mixture components k and mixture overall mean (intercept) and mixture overall standard deviation (scale):

```
> mixture <- list()
> for (ch in 1:nchains) {
+     mixture[[ch]] <- read.table(paste(dirsim[ch], "/mixmoment.sim",
+         sep = ""), header = TRUE)
+ }
```

Create `CODA` `mcmc` objects from sampled chains:

```
> library(coda)
> mcovb <- list()
> mcovt <- list()
> mbeta <- list()
> mmixture <- list()
> for (ch in 1:nchains) {
+     mcovb[[ch]] <- files2coda(data.frames = "covb", thin = 1,
+         chain = ch)
+     mcovt[[ch]] <- files2coda(data.frames = "covt", thin = 1,
+         chain = ch)
+     mbeta[[ch]] <- files2coda(data.frames = "beta", thin = 1,
+         chain = ch)
+     mmixture[[ch]] <- files2coda(data.frames = "mixture", thin = 1,
+         chain = ch)
+ }
> mcovb <- mcmc.list(mcovb[[1]], mcovb[[2]])
> mcovt <- mcmc.list(mcovt[[1]], mcovt[[2]])
> mbeta <- mcmc.list(mbeta[[1]], mbeta[[2]])
> mmixture <- mcmc.list(mmixture[[1]], mmixture[[2]])
```

Compute some summary statistics:

```
> quant <- c(0, 0.025, 0.5, 0.75, 0.975, 1)
> qnames <- paste(quant * 100, "\\%", sep = "")
> meancovb <- apply(rbind(covb[[1]], covb[[2]]), 2, mean)
> meancovt <- apply(rbind(covt[[1]], covt[[2]]), 2, mean)
> meanbeta <- apply(rbind(beta[[1]], beta[[2]]), 2, mean)
> meanmixture <- apply(rbind(mixture[[1]], mixture[[2]]), 2, mean)
> quantcovb <- apply(rbind(covb[[1]], covb[[2]]), 2, quantile,
+     probs = quant)
> quantcovt <- apply(rbind(covt[[1]], covt[[2]]), 2, quantile,
+     probs = quant)
> quantbeta <- apply(rbind(beta[[1]], beta[[2]]), 2, quantile,
+     probs = quant)
> quantmixture <- apply(rbind(mixture[[1]], mixture[[2]]), 2, quantile,
+     probs = quant)
> sumcovb <- rbind(meancovb, quantcovb[c("50%", "2.5%", "97.5%"),
+     ])
> rownames(sumcovb)[1] <- "Mean"
> sumcovt <- rbind(meancovt, quantcovt[c("50%", "2.5%", "97.5%"),
+     ])
> rownames(sumcovt)[1] <- "Mean"
> sumbeta <- rbind(meanbeta, quantbeta[c("50%", "2.5%", "97.5%"),
+     ])
> rownames(sumbeta)[1] <- "Mean"
> summixture <- rbind(meanmixture, quantmixture[c("50%", "2.5%",
+     "97.5%"), ])
> rownames(summixture)[1] <- "Mean"
> sumwant <- cbind(sumcovt, sumbeta, summixture)
> print(sumwant)
```

```
          sdt1      sdt2      sdt3      sdt4    cort12    cort13    cort14
Mean  0.2042222 0.1983714 0.2048045 0.2023484 0.8864512 0.9139874 0.8410793
50%   0.2040386 0.1982670 0.2046224 0.2021866 0.8870772 0.9144700 0.8418553
2.5%  0.1916935 0.1862445 0.1898085 0.1874332 0.8556079 0.8867081 0.8038447
97.5% 0.2175396 0.2111030 0.2209952 0.2179746 0.9138670 0.9379874 0.8742514
        cort23    cort24    cort34        GIRL         DMF      LOWER4
Mean  0.7922175 0.8947478 0.8467664 -0.06836461 -0.04570800 -0.03254852
50%   0.7929055 0.8952042 0.8474862 -0.06803672 -0.04574820 -0.03258596
2.5%  0.7487231 0.8644521 0.8103506 -0.10033914 -0.06311243 -0.04947324
97.5% 0.8316609 0.9227729 0.8797049 -0.03675370 -0.02842491 -0.01590043
        UPPER5    LOWER5     GIRL.DMF  DMF.LOWER4   DMF.UPPER5 DMF.LOWER5
Mean  0.1432326 0.1490098  0.010493725 0.025446662  0.013957467 0.03665927
50%   0.1429807 0.1488353  0.010514335 0.025279185  0.013890850 0.03659677
2.5%  0.1245450 0.1281856 -0.007339762 0.007152071 -0.005376605 0.01586346
97.5% 0.1632935 0.1709299  0.027908040 0.044379541  0.033329682 0.05797958
       GIRL.LOWER4  GIRL.UPPER5  GIRL.LOWER5         k Intercept     Scale
Mean   0.001477880  0.018245946  0.003253059  8.881216  1.756950 0.1049977
50%    0.001631572  0.018354980  0.003202558  8.000000  1.756604 0.1045545
2.5%  -0.018682192 -0.002988832 -0.020633988  3.000000  1.733759 0.0991566
97.5%  0.021706223  0.039228161  0.026901881 18.000000  1.782165 0.1139403
```

In the previous table, `sdt1` $= \sqrt{\mathrm{var}(d_{i,1})}$, `sdt2` $= \sqrt{\mathrm{var}(d_{i,2})}$, `sdt3` $= \sqrt{\mathrm{var}(d_{i,3})}$, `sdt4` $= \sqrt{\mathrm{var}(d_{i,4})}$, `cort12` $= \mathrm{corr}(d_{i,1}, d_{i,2})$, ...,`cort34` $= \mathrm{corr}(d_{i,3}, d_{i,4})$.

Bayesian p-values for regression parameters:

```
> regres <- list()
> pval <- numeric(9)
> for (i in 1:12) {
+     regres[[i]] <- c(beta[[1]][, i], beta[[2]][, i])
+     M <- length(regres[[i]])
+     p1 <- sum(regres[[i]] < 0)/M
+     p2 <- sum(regres[[i]] > 0)/M
+     pval[[i]] <- 2 * min(p1, p2)
+     names(pval)[i] <- colnames(beta[[1]])[i]
+ }
> pval <- round(pval, 3)
> rm(list = "regres")
> print(pval)
```

```
      GIRL         DMF      LOWER4      UPPER5      LOWER5    GIRL.DMF
     0.000       0.000       0.000       0.000       0.000       0.237
DMF.LOWER4  DMF.UPPER5  DMF.LOWER5 GIRL.LOWER4 GIRL.UPPER5 GIRL.LOWER5
     0.006       0.157       0.000       0.875       0.095       0.791
```

Bayesian p-values for effect of dmf for different teeth (maxillary 4, mandibular 4, maxillary 5, mandibular 5) and different genders (p-values for proper combinations of β and γ parameters):

```
> betaall <- rbind(beta[[1]], beta[[2]])
> dmfeff <- data.frame(girl.max4 = betaall$DMF + betaall$GIRL.DMF,
+     boy.max4 = betaall$DMF, girl.max5 = betaall$DMF + betaall$GIRL.DMF +
+         betaall$DMF.UPPER5, boy.max5 = betaall$DMF + betaall$DMF.UPPER5,
+     girl.man4 = betaall$DMF + betaall$GIRL.DMF + betaall$DMF.LOWER4,
+     boy.man4 = betaall$DMF + betaall$DMF.LOWER4, girl.man5 = betaall$DMF +
+         betaall$GIRL.DMF + betaall$DMF.LOWER5, boy.man5 = betaall$DMF +
+         betaall$DMF.LOWER5)
> meandmfeff <- apply(dmfeff, 2, mean)
> quantdmfeff <- apply(dmfeff, 2, quantile, probs = quant)
> sumdmfeff <- rbind(meandmfeff, quantdmfeff[c("50%", "2.5%", "97.5%"),
```

```
+     ])
> rownames(sumdmfeff)[1] <- "Mean"
> dmfpval <- numeric()
> for (i in 1:8) {
+     M <- length(dmfeff[[i]])
+     p1 <- sum(dmfeff[[i]] < 0)/M
+     p2 <- sum(dmfeff[[i]] > 0)/M
+     dmfpval[[i]] <- 2 * min(p1, p2)
+     names(dmfpval)[i] <- colnames(dmfeff)[i]
+ }
> print(sumdmfeff)

        girl.max4    boy.max4    girl.max5    boy.max5    girl.man4
Mean  -0.03521428 -0.04570800 -0.021256809 -0.03175053 -0.009767615
50%   -0.03515911 -0.04574820 -0.021231418 -0.03170497 -0.009751722
2.5%  -0.05221067 -0.06311243 -0.038973509 -0.05004472 -0.026684948
97.5% -0.01851130 -0.02842491 -0.003534773 -0.01346895  0.006972091
          boy.man4    girl.man5     boy.man5
Mean  -0.020261340  0.001444998 -0.009048727
50%   -0.020126970  0.001486195 -0.009006635
2.5%  -0.037790925 -0.016229219 -0.028287810
97.5% -0.003177792  0.019252297  0.009795347

> print(dmfpval)

   girl.max4     boy.max4    girl.max5     boy.max5    girl.man4     boy.man4
0.0000000000 0.0000000000 0.0189189189 0.0008648649 0.2551351351 0.0208648649
   girl.man5     boy.man5
0.8704864865 0.3529729730
```

10 Selected convergence diagnostics and further summary using the coda package

We show some basic convergence diagnostics and some additional summary based on the first chain.

Posterior densities for regression parameters $\boldsymbol{\beta}$, means of random effects $\boldsymbol{\gamma}$, standard deviations and correlations of transformed random effects $\boldsymbol{d}_i$ and overall mixture mean (intercept), mixture standard deviation (scale) and number of mixture components. See Figures 2, 3, 4 for the results.

```
> ch <- 1
> par(mfrow = c(3, 4))
> for (i in 1:12) {
+     densplot2(mcmc(beta[[ch]][, i]), main = colnames(beta[[ch]])[i])
+ }
```

```
> par(mfrow = c(3, 4))
> for (i in 1:10) {
+     densplot2(mcmc(covt[[ch]][, i]), main = colnames(covt[[ch]])[i])
+ }
```

```
> par(mfrow = c(2, 2))
> for (i in 1:3) {
+     densplot2(mcmc(mixture[[ch]][, i]), main = colnames(mixture[[ch]])[i])
+ }
```

Autocorrelation plots for regression parameters $\boldsymbol{\beta}$, means of random effects $\boldsymbol{\gamma}$, standard deviations and correlations of transformed random effects $\boldsymbol{d}_i$ and overall mixture mean (intercept), mixture standard deviation (scale) and number of mixture components. See Figures 5, 6, 7 for the results.

```
> par(mfrow = c(3, 4))
> autocorr.plot(mcmc(beta[[ch]]), auto.layout = FALSE, ask = FALSE,
+     bty = "n")
```

```
> par(mfrow = c(3, 4))
> autocorr.plot(mcmc(covt[[ch]]), auto.layout = FALSE, ask = FALSE,
+     bty = "n")
```

```
> par(mfrow = c(2, 2))
> autocorr.plot(mcmc(mixture[[ch]]), auto.layout = FALSE, ask = FALSE,
+     bty = "n")
```

Trace-plots for regression parameters $\boldsymbol{\beta}$, means of random effects $\boldsymbol{\gamma}$, standard deviations and correlations of transformed random effects $\boldsymbol{d}_i$ and overall mixture mean (intercept), mixture standard deviation (scale) and number of mixture components can be drawn using the following code.

```
> par(mfrow = c(3, 4))
> traceplot2(mcmc(beta[[ch]]))
```

```
> par(mfrow = c(3, 4))
> traceplot2(mcmc(covt[[ch]]))
```

```
> par(mfrow = c(2, 2))
> traceplot2(mcmc(mixture[[ch]]))
```

Gelman-Rubin convergence diagnostics:

```
> gelm.beta <- gelman.diag(mbeta)
> gelm.covt <- gelman.diag(mcovt)
> gelm.mixture <- gelman.diag(mmixture)
> rownames(gelm.beta$psrf) <- dimnames(mbeta[[1]])[[2]]
> rownames(gelm.covt$psrf) <- dimnames(mcovt[[1]])[[2]]
> rownames(gelm.mixture$psrf) <- dimnames(mmixture[[1]])[[2]]

> print(gelm.beta)

Potential scale reduction factors:

            Point est. 97.5% quantile
GIRL              1.14           1.50
DMF               1.02           1.07
LOWER4            1.05           1.20
UPPER5            1.07           1.26
LOWER5            1.04           1.16
GIRL.DMF          1.00           1.00
DMF.LOWER4        1.03           1.12
DMF.UPPER5        1.02           1.10
DMF.LOWER5        1.01           1.06
GIRL.LOWER4       1.01           1.03
GIRL.UPPER5       1.03           1.13
GIRL.LOWER5       1.01           1.04

Multivariate psrf

1.15+0i

> print(gelm.covt)

Potential scale reduction factors:

       Point est. 97.5% quantile
sdt1         1.01           1.03
sdt2         1.00           1.00
sdt3         1.00           1.01
sdt4         1.00           1.00
cort12       1.00           1.00
cort13       1.00           1.00
cort14       1.00           1.00
cort23       1.00           1.00
cort24       1.01           1.03
cort34       1.00           1.00

Multivariate psrf

1.02+0i

> print(gelm.mixture)

Potential scale reduction factors:

          Point est. 97.5% quantile
k               1.07           1.25
Intercept       1.27           1.87
Scale           1.14           1.38
```

```
Multivariate psrf

1.20+0i
```

Figure 2: Posterior distribution of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ parameters.

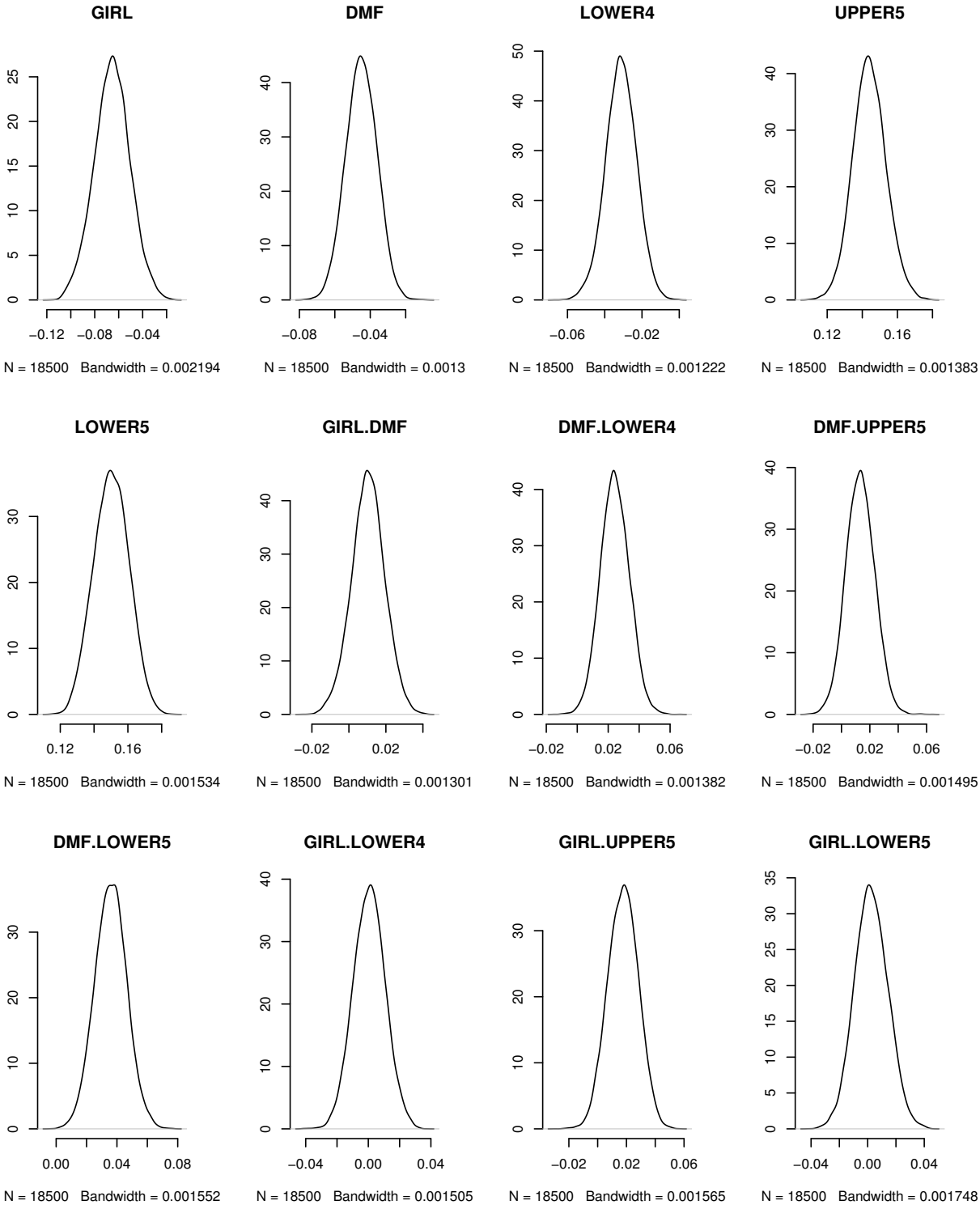


Figure 3: Posterior distribution of standard deviations and correlations of transformed random effects $\boldsymbol{d}_i$.

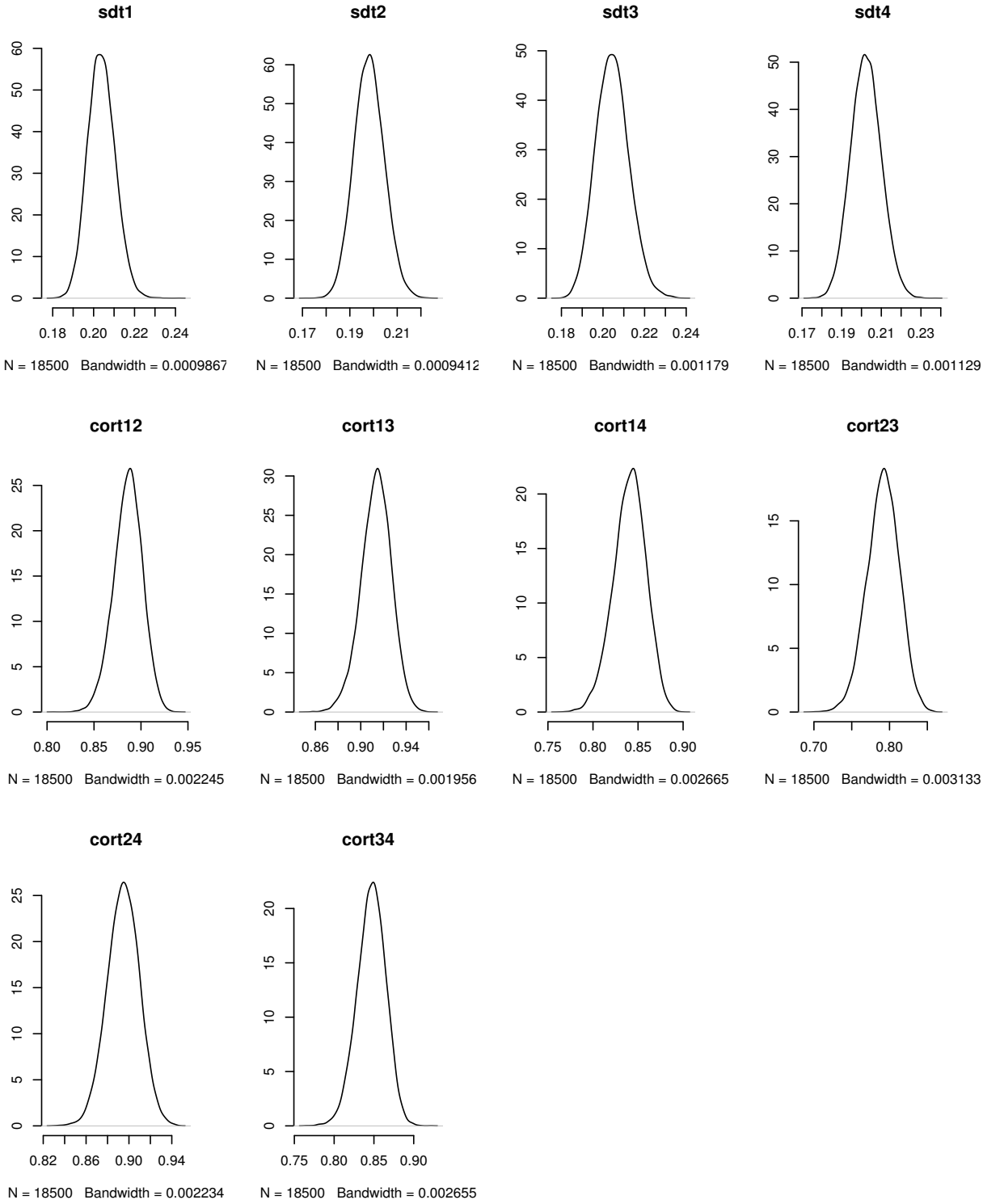


Figure 4: Posterior distribution for mixture related parameters.

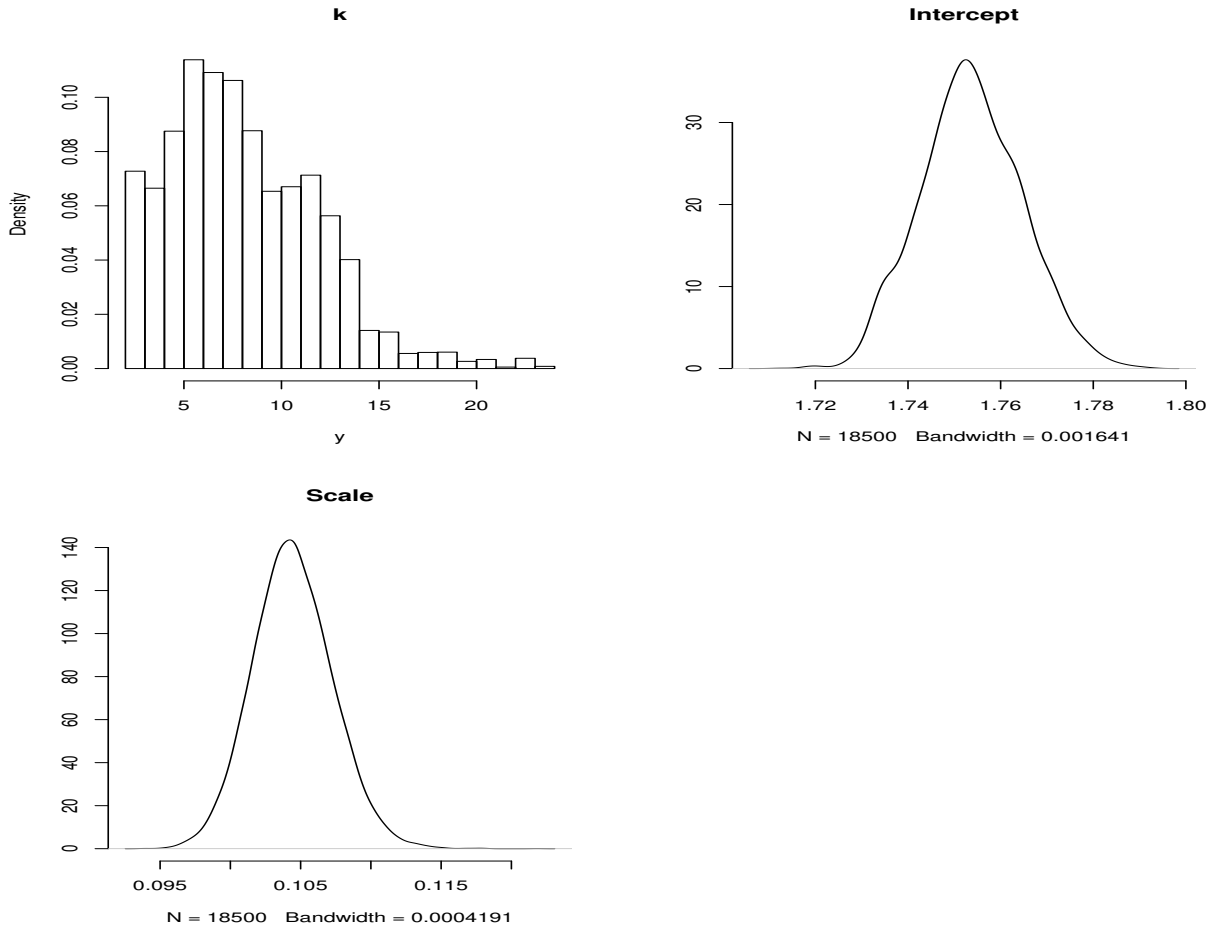


Figure 5: Autocorrelation for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ parameters.

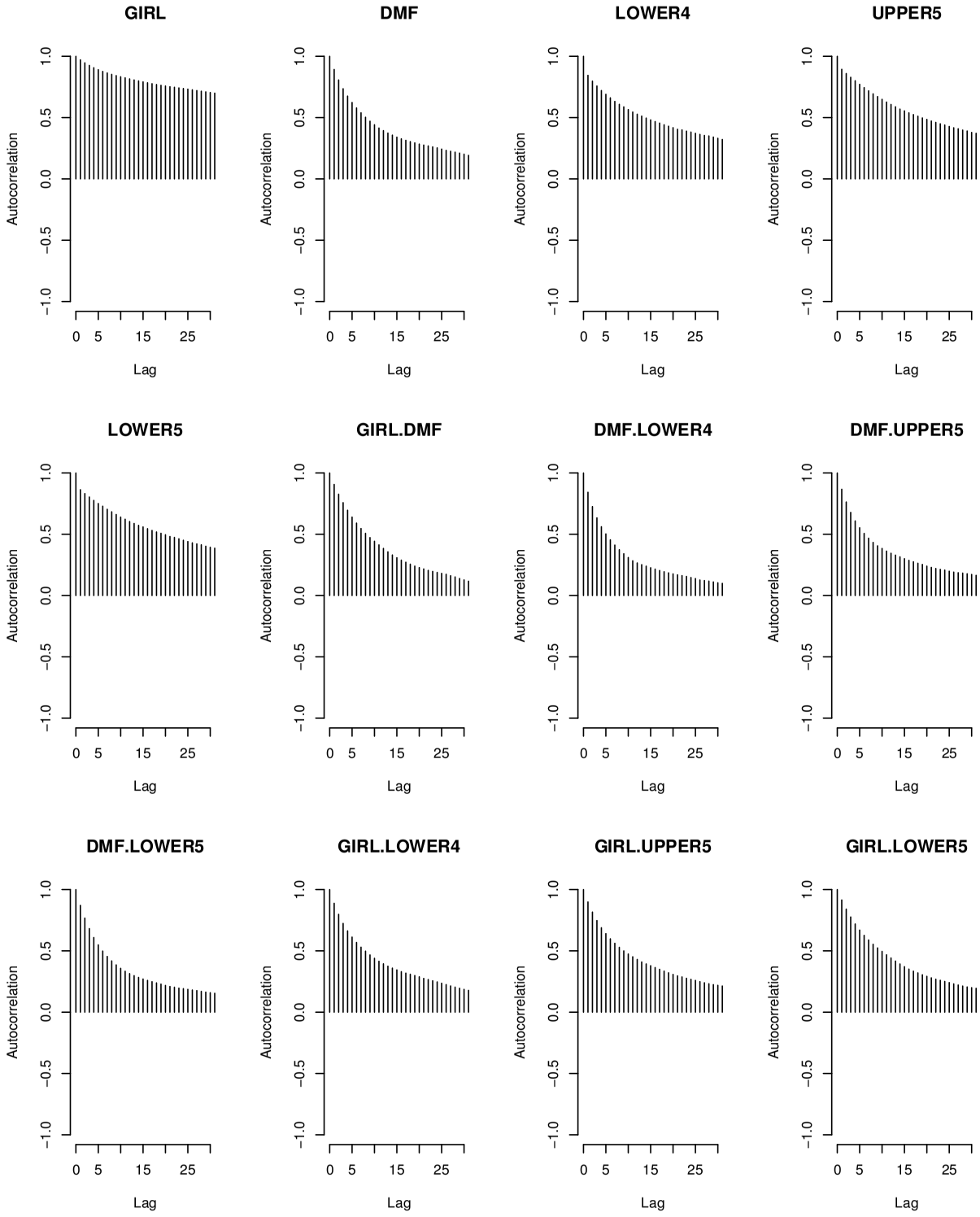


Figure 6: Autocorrelation for standard deviations and correlations of transformed random effects $\boldsymbol{d}_i$.

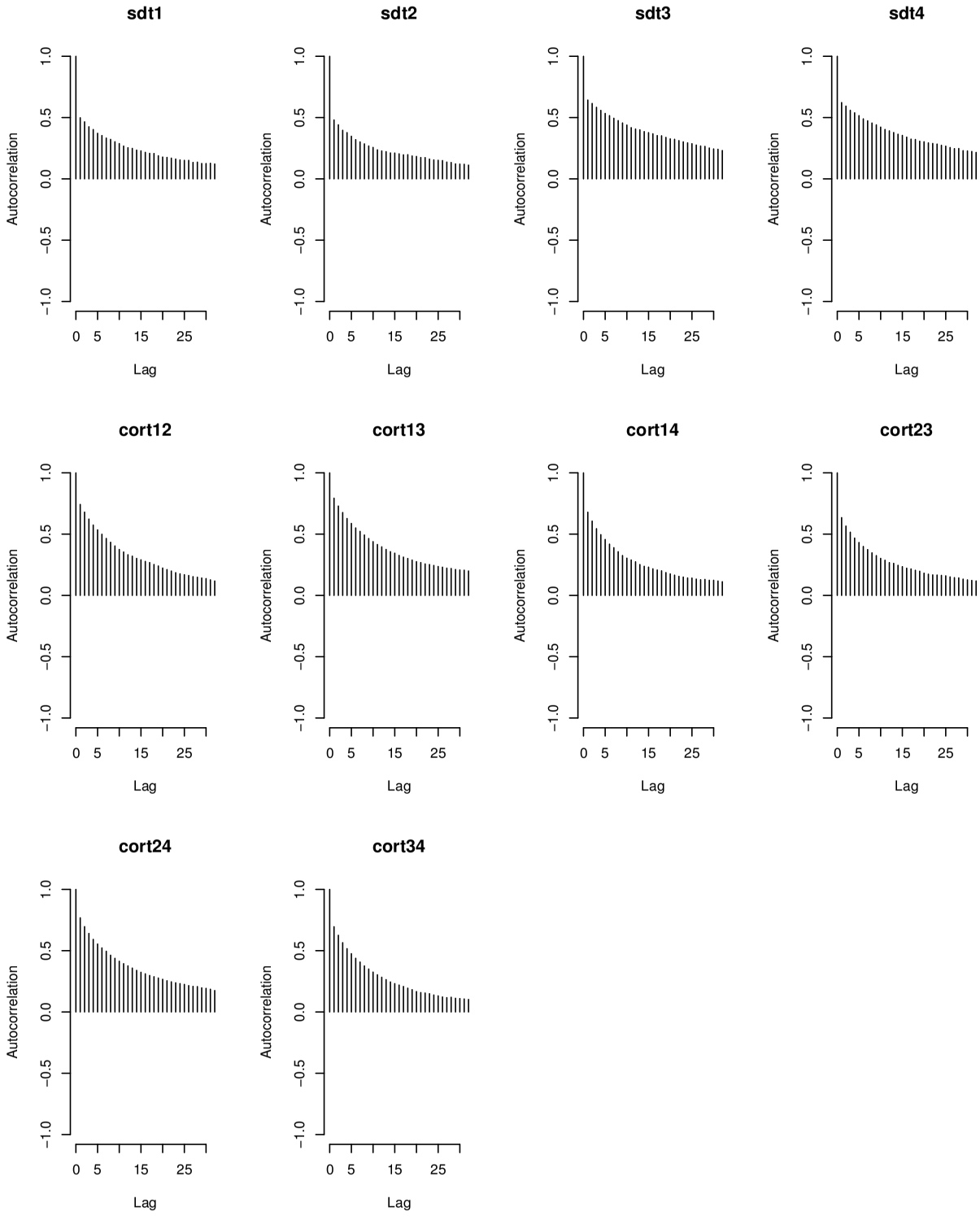
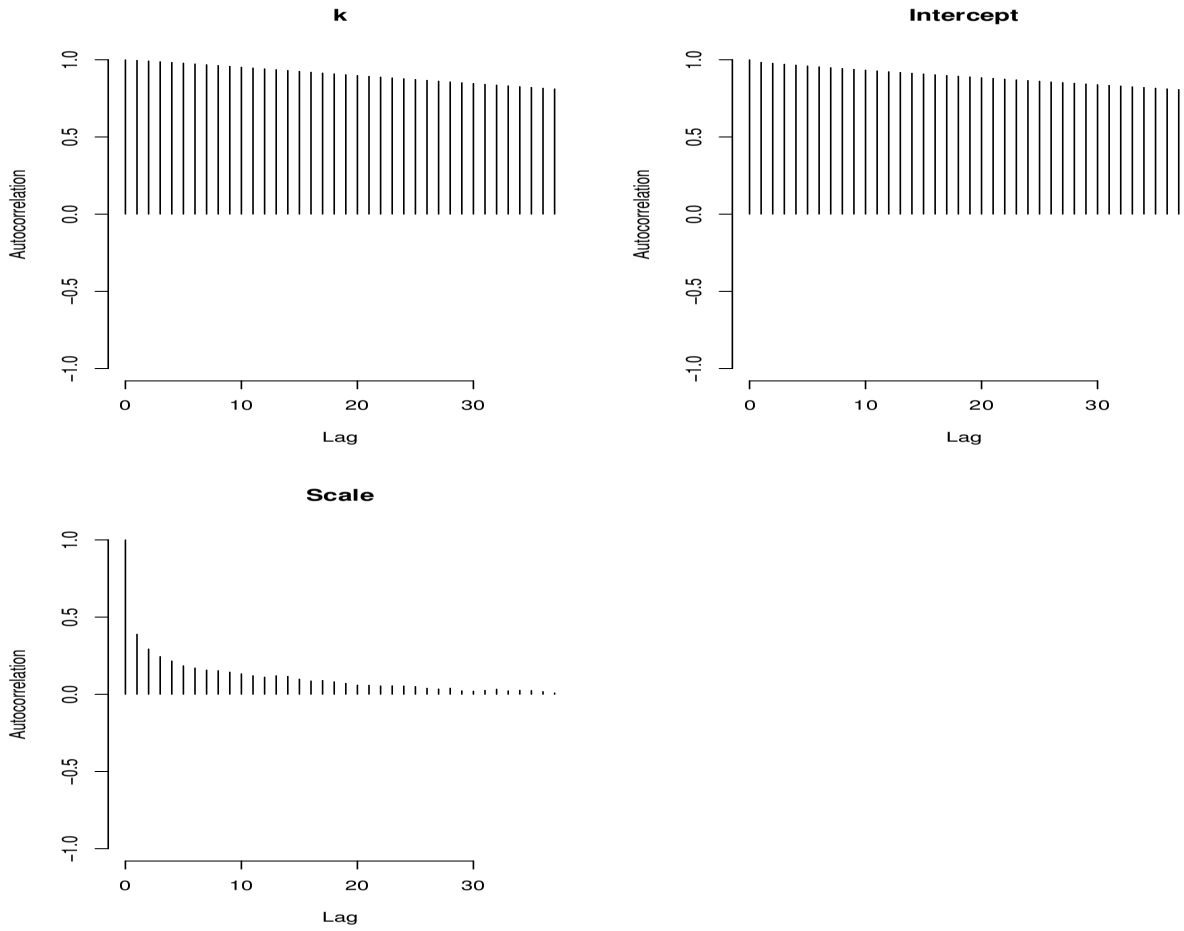


Figure 7: Autocorrelation for mixture related parameters.



11 Posterior predictive error distribution

Based separately on both chains (see Figure 8 for the result):

```
> dens <- list()
> dens[[1]] <- bayesDensity(dirsim[[1]], grid = seq(1.4, 2, by = 0.005),
+     stgrid = seq(-3, 2, by = 0.05))

Reading mixture files.
Computing predictive densities.

> dens[[2]] <- bayesDensity(dirsim[[2]], grid = seq(1.4, 2, by = 0.005),
+     stgrid = seq(-3, 2, by = 0.05))

Reading mixture files.
Computing predictive densities.
```

Now, we can draw them (see Figure 8 for the result). Observe that conditional on k we draw only densities for $k = 1, \ldots, 10$.

```
> par(mfrow = c(2, 2), bty = "n")
> plot(dens[[1]], standard = TRUE, dim.plot = FALSE, main = "Standardized, chain 1",
+     k.cond = 0:10)
> plot(dens[[1]], standard = FALSE, dim.plot = FALSE, main = "Unstandardized, chain 1",
+     k.cond = 0:10)
> plot(dens[[2]], standard = TRUE, dim.plot = FALSE, main = "Standardized, chain 2",
+     k.cond = 0:10)
> plot(dens[[2]], standard = FALSE, dim.plot = FALSE, main = "Unstandardized, chain 2",
+     k.cond = 0:10)
```

Finally, we perform cleaning of generated files:

```
> files1 <- dir("./tandchain1test")
> files2 <- dir("./tandchain2test")
> file.remove(paste("./tandchain1test/", files1, sep = ""))
> file.remove(paste("./tandchain2test/", files2, sep = ""))
> file.remove("tandchain1test")
> file.remove("tandchain2test")
```

Figure 8: Posterior predictive error densities.