CGD Data: Recurrent Events Analysis
Survival analysis using `bayessurvreg1`

Arnošt Komárek

July 5, 2005

In this document we describe the analysis of CGD data presented originally in 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. On request of the referees the CGD data analysis was removed from the paper so the Section dealing with CGD data is given in this document.

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

Correlated survival times are encountered in many medical problems, e.g. when there are recurrent events on an individual or when the observations are clustered (multicenter studies, multivariate survival times).

In an AFT model the covariates are assumed to speed up or slow down the expected time to failure. An extension of the AFT model to incorporate correlated survival data could consist in including random effects in the regression expression as in a classical linear mixed model (Laird and Ware, 1982), i.e.

$$\log(T_{i,l}) = Y_{i,l} = \boldsymbol{\beta}^T \boldsymbol{x}_{i,l} + \boldsymbol{b}_i^T \boldsymbol{z}_{i,l} + \varepsilon_{i,l}, \qquad i = 1, \ldots, N, \quad l = 1, \ldots, n_i, \tag{1}$$

where $T_{i,l}$ is the event time of the lth observation of the ith cluster or the time of the lth recurrent event on the ith patient, $Y_{i,l}$ its logarithmic (or any other monotone) transformation, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ is the unknown regression coefficient vector, $\boldsymbol{x}_{i,l}$ the covariate vector for fixed effects, $\boldsymbol{b}_i = (b_{i,1}, \ldots, b_{i,q})^T$ is the random effect vector causing the possible correlation for the components of $\boldsymbol{Y}_i = (Y_{i,1}, \ldots, Y_{i,n_i})^T$, $\boldsymbol{z}_{i,l}$ is the covariate vector for random effects and $\varepsilon_{i,l}$ are independent and identically distributed random variables. Along the lines of Gelman et al. (2004, Chapter 15) we use the terms 'fixed' and 'random' effects throughout the paper even in a Bayesian context where all unknown parameters are treated as random quantities.

For recurrent events, usually $\boldsymbol{z}_{i,l} = 1$ for all i and l and $\boldsymbol{b}_i = b_{i,1}$ expresses an individual-specific deviation from an overall mean log-event time which is not explained by fixed effects covariates. For clustered data, the vector $\boldsymbol{z}_{i,l}$ may define further sub-clusters allowing for closer dependence of observations within sub-clusters given by common values of appropriate components of the vector $\boldsymbol{b}_i$ while keeping the dependence also across the sub-clusters through the correlation between the components of $\boldsymbol{b}_i$.

2 CGD data: recurrent events analysis

This section gives the description of the CGD data analysis as presented in the original manuscript of Komárek and Lesaffre (2005).

The data example uses the data set from a multicenter placebo-controlled randomized trial of gamma inferon in patients with chronic granulotomous disease (CGD). The data set can be found in Appendix D.2 of Fleming and Harrington (1991). There were 128 patients randomized to either gamma inferon ($n = 63$) or placebo ($n = 65$). For each patient the times from study entry to initial and any recurrent serious infections are available. There is a minimum of one and a maximum of eight (recurrent) infection times per patient, with a total of 203 records.

The problem of recurrent events in this data set was discussed by several authors. Among others, Therneau and Hamilton (1997) used the CGD data to illustrate several approaches for recurrent event analysis based on the Cox's proportional hazards (PH) model. Vaida and Xu (2000) used this dataset to illustrate the PH model with random effects. They specify the hazard function for the $(i, l)th$ event as $\hbar_{i,l}(t) = \hbar_0(t)\ \exp(\boldsymbol{\beta}^T \boldsymbol{x}_{i,l} + \boldsymbol{b}_i \boldsymbol{z}_{i,l})$ and use a normal distribution for $\boldsymbol{b}_i$.

In this section, we present AFT model (1) with response the time from entry or previous infection to the next infection in days. Each patient represents a cluster, i.e. $i = 1, \ldots, 203$, $l = 1, \ldots, n_i$, $n_i \leq 8$. Dependencies between the times of recurrent events of one patient are introduced by a univariate random effect b_i with $z_{i,l} = 1$ for all i and l. As fixed effects covariates, we used the same covariates as Vaida and Xu (2000), see Table 1 for their list.

The initial maximum-likelihood AFT model with a normal error distribution and without random effects gave an estimate of the intercept equal to 3.66 and a scale equal to 1.69. Along the suggestions made in Komárek and Lesaffre (2005) we used the following values of hyperparameters: $\xi = 3.66$, $\kappa = 25 \approx (3 \cdot 1.69)^2$, $\zeta = 2$, $g = 0.2$, $h = 0.1$, $\delta = 1$. For the number of mixture components, k, a truncated Poisson prior with $\lambda = 5$ reflecting our prior belief that the error distribution is skewed and $k_{max} = 30$ was used. Prior means of all regression parameters were equal to 0 and their prior variances to 1000.

For the variance d of the random effect we tried either an inverse-gamma$(0.001, 0.001)$ prior ($\tau = 0.002$, $s = 0.002$ in the terms of the inverse-Wishart distribution used in the DAG (see Figure in Komárek and Lesaffre (2005)) or a uniform Unif$(0, \sqrt{s})$ prior on $\sqrt{d}$ Gelman et al. (2004, pp. 136, 390) with s equal to 100^2, 50^2 and 10^2. Different priors for this parameter had only negligible effect on the

Table 1: CGD Data. Posterior means, 95% equal-tail credibility intervals and Bayesian p-values for regression parameters $\boldsymbol{\beta}$: *trtmt* = treatment (yes), *inher* = pattern of inheritance (autosomal recessive), *age* = age in years, *cortico* = use of corticosteroids (yes), *prophy* = use of prophylactic antibiotics (yes), *gender* = female, *hosp*1 = hosp. category US – other, *hosp*2 = hosp. category Europe – Amsterdam, *hosp*3 = hosp. category Europe – other. Posterior summary statistics for intercept = mean of the error distribution, scale = standard deviation of the error distribution and standard deviation of the random effect.

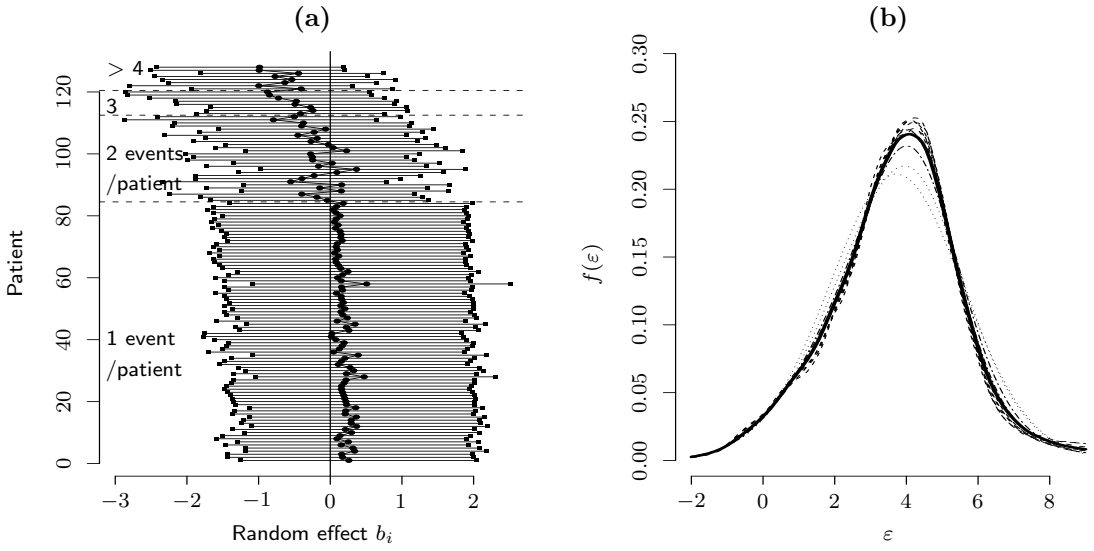
trtmt	*inher*	*age*	*cortic*
1.303	−0.885	0.047	−2.533
(0.496, 2.214)	(−1.812, 0.035)	(0.005, 0.093)	(−5.311, −0.106)
$p = 0.001$	$p = 0.059$	$p = 0.027$	$p = 0.04$
prophy	*gender*	*hosp*1	*hosp*2
1.111	1.369	0.466	1.589
(0.069, 2.265)	(0.03, 2.821)	(−0.464, 1.473)	(0.143, 3.265)
$p = 0.036$	$p = 0.045$	$p = 0.333$	$p = 0.031$
*hosp*3	intercept	scale	std. dev. of $b_{i,1}$
1.213	3.852	1.871	0.826
(−0.071, 2.625)	(2.213, 5.465)	(1.259, 3.321)	(0.197, 1.473)
$p = 0.063$			

posterior distributions of all remaining parameters. However, the posterior distribution of d was strongly driven by the inverse-gamma prior (showing two modes, one of them located close to zero). This was not the case when the uniform prior was used. Additionally, all uniform priors led to essentially identical posterior distributions. All results presented below are then based on Unif(0, 100) prior on $\sqrt{d}$.

Posterior summary statistics of the model can be found in Table 1. It is seen that the treatment significantly increases the time to the infection. Further, the posterior mean of $\exp\{\beta(trtmt)\}$ is equal to 4.01 with 95% CI = (1.60, 9.18) which means that on average, the treatment increases the time to the next event 4.01 times.

Further, the first panel of Figure 1 shows posterior means and 95% posterior credibility intervals of random effects b_i for all patients, sorted according to number of infections they underwent. It is clearly seen that the random effects of patients with higher numbers of total infections on average decrease (consequently the same is true for the time to the next event).

Figure 1: CGD Data – Recurrent Events Analysis. (a) posterior means and 95% PCI for random effects b_i; (b) predictive error densities; solid line: unconditionally, dotted line: $k = 1,\ 2$, dotted-dashed line: $k = 3,\ 4$, dashed line: $k = 5 - 10$.



2.1 Predictive error densities

Averaging the error density across the MCMC run, conditionally on fixed values of k, gives a Bayesian predictive error density estimate of the mixture with k components, i.e. an estimate of

$$E\big\{f(\cdot\,|k, \boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2\,)\big|\ k, \text{data}\big\}.$$

Averaging further across values of k gives an estimate of

$$E\big\{f(\cdot\,|k, \boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2\,)\big|\ \text{data}\big\},$$

the overall Bayesian predictive density estimate of the error distribution. In our sample, the number of mixture components k ranged from 1 to 18 while mixtures with $k \in \{4, 5, 6, 7\}$ occupied each more than 10% of the sample, with the highest frequency for $k = 6$ (13.0%). Mixtures with $k \geq 11$ took each less than 3% of the sample. Apparently, the model did not suffer from the technical restriction given by $k_{max} = 30$. Predictive error density estimates are shown in the second panel of Figure 1. Note that only $k \in \{1, 2\}$ (14.8% of the sample) gives an appreciably different estimate from the unconditional estimate and conditional estimates for $3 \leq k \leq 10$ (79.3% of the sample).

2.2 Predictive survivor curves

Further, we present estimates of predictive survivor curves for a specific value of covariates, say $\boldsymbol{x}_{new}$ and $\boldsymbol{z}_{new}$. Denoting all unknown quantities in the model by $\boldsymbol{\theta}$ and omitting $\boldsymbol{x}_{new}$ and $\boldsymbol{z}_{new}$ in the notation, the predictive survivor function is given by

$$S(t \mid \text{data}) = \int S(t \mid \boldsymbol{\theta}, \text{data})\ p(\boldsymbol{\theta} \mid \text{data})\ d\boldsymbol{\theta}$$

for any $t > 0$. Further

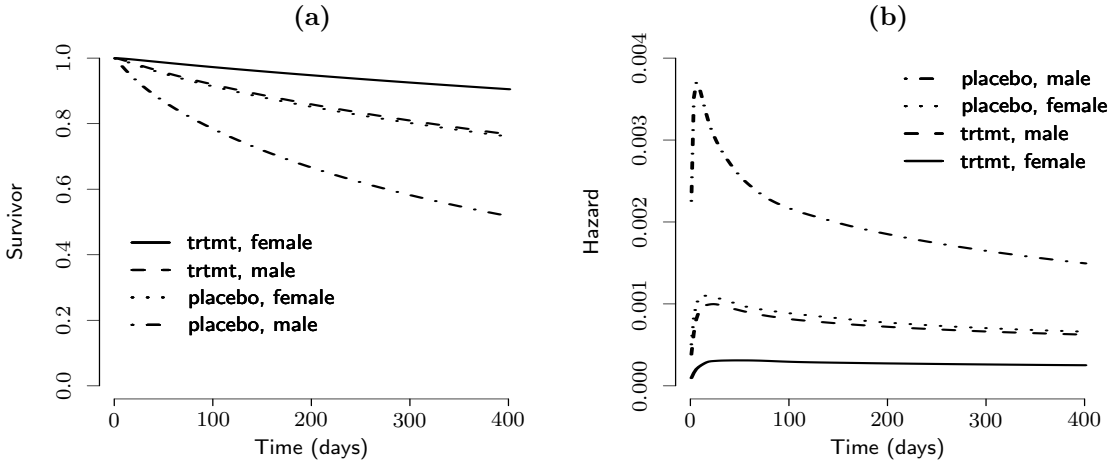
$$S(t \mid \boldsymbol{\theta}, \text{data}) = S(t \mid \boldsymbol{\theta}) = \sum_{j=1}^{k} w_j \Big[1 - \Phi\big\{\log(t) - \boldsymbol{\beta}^T \boldsymbol{x}_{new} - \boldsymbol{b}^T \boldsymbol{z}_{new} \ \big|\ \mu_j, \sigma_j^2\big\}\Big],$$

where $\Phi(\cdot \mid \mu_j, \sigma_j^2)$ is a cumulative distribution function of $\mathrm{N}(\mu_j, \sigma_j^2)$. The MCMC estimate of the predictive survivor function is then given by

$$\hat{S}(t \mid \text{data}) = M^{-1} \sum_{m=1}^{M} \sum_{j=1}^{k^{(m)}} w_j^{(m)} \Big[1 - \Phi\big\{\log(t) - \boldsymbol{\beta}^{(m)T} \boldsymbol{x}_{new} - \boldsymbol{b}^{(m)T} \boldsymbol{z}_{new} \ \big|\ \mu_j^{(m)}, \sigma_j^{(m)2}\big\}\Big],$$

where M denotes number of MCMC iterations. All quantities are available, except $\boldsymbol{b}^{(m)}$. This must be additionally sampled from $\mathrm{N}_q(\boldsymbol{\gamma}^{(m)}, d^{(m)})$. Predictive survivor curves for males and females taking treatment or placebo while controlling for remaining covariates are shown in the left part of Figure 2.

Figure 2: CGD Data – Recurrent Events Analysis. (a) predictive survivor and (b) hazard curves for males and females taking either treatment or placebo Remaining covariates were fixed to either mean value ($age = 14.6$) or to most common value (X-linked pattern of inheritance, no use of corticosteroids, use of prophylactic antibiotics and a hospital category US-other).



2.3 Predictive hazard functions

Also predictive hazard functions can be computed. For any $t > 0$

$$\hbar(t \mid \boldsymbol{\theta}, \text{data}) = \hbar(t \mid \boldsymbol{\theta}) = \frac{p(t \mid \boldsymbol{\theta})}{S(t \mid \boldsymbol{\theta})},$$

where $p(t \mid \boldsymbol{\theta}) = t^{-1} \sum_{j=1}^{k} w_j \varphi\big\{\log(t) - \boldsymbol{\beta}^T \boldsymbol{x}_{new} - \boldsymbol{b}^T \boldsymbol{z}_{new} \mid \mu_j, \sigma_j^2\big\}$. The MCMC estimate of the predictive hazard function is then given by

$$\hat{\hbar}(t \mid \text{data}) = M^{-1} \sum_{m=1}^{M} \frac{t^{-1} \sum_{j=1}^{k^{(m)}} w_j^{(m)} \varphi\big\{\log(t) - \boldsymbol{\beta}^{(m)T} \boldsymbol{x}_{new} - \boldsymbol{b}^{(m)T} \boldsymbol{z}_{new} \mid \mu_j^{(m)}, \sigma_j^{(m)2}\big\}}{\sum_{j=1}^{k^{(m)}} w_j^{(m)} \Big[1 - \Phi\big\{\log(t) - \boldsymbol{\beta}^{(m)T} \boldsymbol{x}_{new} - \boldsymbol{b}^{(m)T} \boldsymbol{z}_{new} \mid \mu_j^{(m)}, \sigma_j^{(m)2}\big\}\Big]}.$$

Predictive hazard curves for same combination of covariates as before are shown in the right part of Figure 2.

3 Summary of the model

We consider the following model

$$\begin{aligned}\log(T_{i,l}) = &\beta_1\, trtmt_i + \beta_2\, inherit_i + \beta_3\, age_{i,l} + \beta_4\, cortico_i + \beta_5\, prophy_i + \beta_6\, (gender_i = female) + \\ &+ \beta_7\, (hospital_i = USother) + \beta_8\, (hospit_i = EUAmsterdam) + \beta_9\, (hospit_i = EUother) + \\ &+ b_i + \varepsilon_{i,l},\end{aligned}$$

where $i = 1, \ldots, 128$ indexes patients and l recurrent events on patients.

4 Initial operations

• Set the directories.

```
> anadir <- "/home/komari/win/work/papers/bayesaft/CGDdata/"
> dirsim1 <- paste(anadir, "anapaper1b/chain1", sep = "")
> dirsim2 <- paste(anadir, "anapaper1b/chain2", sep = "")
```

Firstly we load the package `bayesSurv` and the data and do some arrangements.

```
> library(bayesSurv)
```

```
Loading required package: survival
Loading required package: splines
Loading required package: coda
Loading required package: smoothSurv
```

```
> data(cgd)
> print(cgd[1:6, ])
```

```
  hospit     ID    RDT    IDT trtmt inherit age height weight cortico prophy
1    174 174054 120688 092589     1       2  38 152.20   66.7       2      1
2    174 174077 011389 092589     2       1  14 144.00   32.8       2      1
3    174 174109 022489 092589     2       1  26  81.25   55.0       2      1
4    174 174111 030689 092589     2       1  26 178.50   69.3       2      1
5    204 204001 082888 040489     1       2  12 147.00   62.0       2      2
6    204 204001 040589 090589     1       2  12 147.00   62.0       2      2
  gender hcat  T1  T2 event sequence
1      2    2 293   0     2        1
2      1    2 255   0     2        1
3      1    2 213   0     2        1
4      1    2 203   0     2        1
5      2    2 219   0     1        1
6      2    2 373 220     1        2
```

For our analysis we change all 1-2 variables into 1-0 or 0-1 ones. Such that

Variable	0	1
`trtmt`	placebo	treatment
`gender`	male	female
`inherit`	X-linked	autosomal recessive
`cortico`	no	yes
`prophy`	no	yes
`event`	censored	obsered

```
> cgd$trtmt <- -(cgd$trtmt - 2)
> cgd$gender <- cgd$gender - 1
> cgd$inherit <- cgd$inherit - 1
> cgd$cortico <- -(cgd$cortico - 2)
> cgd$prophy <- -(cgd$prophy - 2)
> cgd$gender <- factor(cgd$gender, labels = c("male", "female"))
> cgd$inherit <- factor(cgd$inherit, labels = c("X-l", "AuRec"))
> cgd$hcat <- factor(cgd$hcat, labels = c("US-NIH", "US-other",
+     "EU-Am", "EU-other"))
> cgd$event <- -(cgd$event - 2)
```

Further we compute times between two consecutive infections and define some additional variables.

```
> cgd$time <- cgd$T1 - cgd$T2
> npatient <- length(unique(cgd$ID))
> nobs <- dim(cgd)[1]
> print(cgd[1:6, ])
```

```
  hospit     ID    RDT    IDT trtmt inherit age height weight cortico prophy
1    174 174054 120688 092589     1   AuRec  38 152.20   66.7       0      1
2    174 174077 011389 092589     0     X-l  14 144.00   32.8       0      1
3    174 174109 022489 092589     0     X-l  26  81.25   55.0       0      1
4    174 174111 030689 092589     0     X-l  26 178.50   69.3       0      1
5    204 204001 082888 040489     1   AuRec  12 147.00   62.0       0      0
6    204 204001 040589 090589     1   AuRec  12 147.00   62.0       0      0
  gender     hcat  T1  T2 event sequence time
1 female US-other 293   0     0        1  293
2   male US-other 255   0     0        1  255
3   male US-other 213   0     0        1  213
4   male US-other 203   0     0        1  203
5 female US-other 219   0     1        1  219
6 female US-other 373 220     1        2  153
```

5 Finding reasonable values for prior hyperparameters

To find reasonable values for prior hyperparameters we fit the log-normal AFT model with and without random intercept using maximum likelihood:

```
> ifit <- survreg(Surv(time, event) ~ trtmt + inherit + age + cortico +
+     prophy + gender + hcat + frailty(ID, dist = "gaussian"),
+     dist = "lognormal", data = cgd)
> resid <- ifit$y[, 1] - ifit$linear.predictors
> R <- max(resid) - min(resid)
> ifit2 <- survreg(Surv(time, event) ~ trtmt + inherit + age +
+     cortico + prophy + gender + hcat, dist = "lognormal", data = cgd)
```

Summary for the model with the random intercept and the range of residuals:

```
> summary(ifit)

Call:
survreg(formula = Surv(time, event) ~ trtmt + inherit + age +
    cortico + prophy + gender + hcat + frailty(ID, dist = "gaussian"),
    data = cgd, dist = "lognormal")
               Value Std. Error     z        p
(Intercept)   3.9152     0.6611  5.92 3.18e-09
trtmt         1.1040     0.3037  3.63 2.78e-04
inheritAuRec -0.6560     0.3793 -1.73 8.38e-02
age           0.0366     0.0176  2.08 3.73e-02
cortico      -1.7607     0.9307 -1.89 5.85e-02
prophy        0.9390     0.4516  2.08 3.76e-02
genderfemale  1.0256     0.5137  2.00 4.59e-02
hcatUS-other  0.3695     0.3807  0.97 3.32e-01
hcatEU-Am     1.2154     0.5881  2.07 3.88e-02
hcatEU-other  0.8248     0.5191  1.59 1.12e-01
Log(scale)    0.1776     0.0907  1.96 5.03e-02

Scale= 1.19

Log Normal distribution
Loglik(model)= -491.7   Loglik(intercept only)= -548.5
        Chisq= 113.44 on 36.3 degrees of freedom, p= 7e-10
Number of Newton-Raphson Iterations:  6 25
n= 203

> print(R)

[1] 6.40232
```

Summary for the model without the random intercept:

```
> summary(ifit2)

Call:
survreg(formula = Surv(time, event) ~ trtmt + inherit + age +
    cortico + prophy + gender + hcat, data = cgd, dist = "lognormal")
               Value Std. Error      z        p
(Intercept)   3.6570     0.6658  5.493 3.96e-08
trtmt         1.3531     0.3226  4.195 2.73e-05
inheritAuRec -0.9582     0.3646 -2.628 8.58e-03
```

```
age           0.0451     0.0185  2.429 1.51e-02
cortico      -2.3894     0.9599 -2.489 1.28e-02
prophy        1.1071     0.4540  2.439 1.47e-02
genderfemale  1.4679     0.5300  2.770 5.61e-03
hcatUS-other  0.2463     0.4030  0.611 5.41e-01
hcatEU-Am     1.4157     0.6451  2.194 2.82e-02
hcatEU-other  0.9850     0.5673  1.736 8.25e-02
Log(scale)    0.5223     0.0864  6.046 1.48e-09

Scale= 1.69

Log Normal distribution
Loglik(model)= -526.3   Loglik(intercept only)= -548.5
        Chisq= 44.31 on 9 degrees of freedom, p= 1.2e-06
Number of Newton-Raphson Iterations: 4
n= 203
```

6 Specification of priors

To specify correctly the prior hyperparameters for β 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(time, event) ~ trtmt + inherit + age +
+     cortico + prophy + gender + hcat + cluster(ID), random = ~1,
+     data = cgd, onlyX = TRUE)
> nregres <- dim(X)[2]
> X[1:3, ]
```

```
  trtmt inheritAuRec age cortico prophy genderfemale hcatUS-other hcatEU-Am
1     1            1  38       0      1            1            1         0
2     0            0  14       0      1            0            1         0
3     0            0  26       0      1            0            1         0
  hcatEU-other
1            0
2            0
3            0
```

We see that $\beta_1 = trtmt$, $\beta_2 = inherit \ldots \beta_7 = hcat(US - other)$, $\beta_8 = hcat(EU - Am)$, $\beta_9 = hcat(EU - other)$.

Now, we can start to specify the prior choices. These will be stored in `lists`. For illustration purposes, we show also some other prior choices than these used in Komárek and Lesaffre (2005).

6.1 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 N(ξ, κ) with $\xi = 3.66$ (taken from `survreg(dist = "lognormal")` fit (approx intercept)) and $\kappa = 5^2 \approx (3 \times 1.69)^2$ (1.69 was estimated scale parameter by `survreg`).

```
> prior$mean.mu <- 3.66
> prior$var.mu <- 5^2
```

Prior for mixture inverse-variances $\sigma_1^{-2}, \ldots, \sigma_k^{-2}$ will be Gamma(ζ, η) and prior for η will be 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] 3.66

$var.mu
[1] 25

$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
```

6.2 Priors for regression parameters β

For illustration purposes, we define several `list`s with the same prior specification (all $\boldsymbol{\beta}$ parameters are assigned $\mathrm{N}(0, 1000)$ prior) however with different possibilities how to update the $\boldsymbol{\beta}$ parameters in the MCMC simulation.

6.2.1 All β parameters updated using the Gibbs step

With the first specification, all $\boldsymbol{\beta}$ will be updated in one block using the Gibbs move. This is usually a recommended choice and was also used to get results presented in Komárek and Lesaffre (2004).

```
> prior.beta.gibbs <- list()
> prior.beta.gibbs$mean.prior <- rep(0, nregres)
> prior.beta.gibbs$var.prior <- rep(1000, nregres)
```

Look how it looks like:

```
> print(prior.beta.gibbs)

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

$var.prior
[1] 1000 1000 1000 1000 1000 1000 1000 1000 1000
```

6.2.2 All β parameters updated in one block using random walk Metropolis-Hastings step

With the second specification, all $\boldsymbol{\beta}$ parameters would be updated in one block using a random walk Metropolis-Hastings step with a proposal covariance matrix `covm`.

```
> prior.beta.mh1 <- list()
> prior.beta.mh1$mean.prior <- rep(0, nregres)
> prior.beta.mh1$var.prior <- rep(1000, nregres)
```

Definition of blocks in which beta parameters will be updated and the way in which they will be updated:

```
> prior.beta.mh1$blocks <- list()
> prior.beta.mh1$blocks$ind.block <- list()
```

There is only one block that contains beta[1:9]:

```
> prior.beta.mh1$blocks$ind.block[[1]] <- 1:9
> nblock <- length(prior.beta.mh1$blocks$ind.block)
```

Further we define a proposal covariance matrix.

`vars`	=	proposal variances for each beta parameter
`cors`	=	lower triangle of the proposal correlation matrix
`corsm`	=	proposal correlation matrix itself
`covm`	=	proposal covariance matrix

```
> vars <- c(0.15, 0.2, 3e-04, 1.3, 0.08, 0.25, 0.1, 0.35, 0.35)
> cors <- c(1, 0.1, 0, 0.1, 0.15, 0, 0.4, 0.1, 0.2, 1, -0.15, 0.15,
+     -0.2, -0.3, 0.2, -0.1, 0, 1, -0.2, 0.15, 0.2, 0.3, 0.2, 0.1,
+     1, 0.2, -0.5, 0.2, -0.4, 0.4, 1, 0.15, 0.5, 0.3, 0.4, 1,
```

```
+     0.15, 0.15, 0, 1, 0.35, 0.65, 1, 0.2, 1)
> corsm <- diag(9)
> corsm[lower.tri(corsm, diag = TRUE)] <- cors
> corsm[upper.tri(corsm, diag = FALSE)] <- t(corsm)[upper.tri(t(corsm),
+     diag = FALSE)]
> covm <- diag(sqrt(vars)) %*% corsm %*% diag(sqrt(vars))
```

Here is the proposal correlation matrix:

```
> print(corsm)
```

```
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6] [,7]  [,8] [,9]
 [1,] 1.00  0.10  0.00  0.10  0.15  0.00 0.40  0.10 0.20
 [2,] 0.10  1.00 -0.15  0.15 -0.20 -0.30 0.20 -0.10 0.00
 [3,] 0.00 -0.15  1.00 -0.20  0.15  0.20 0.30  0.20 0.10
 [4,] 0.10  0.15 -0.20  1.00  0.20 -0.50 0.20 -0.40 0.40
 [5,] 0.15 -0.20  0.15  0.20  1.00  0.15 0.50  0.30 0.40
 [6,] 0.00 -0.30  0.20 -0.50  0.15  1.00 0.15  0.15 0.00
 [7,] 0.40  0.20  0.30  0.20  0.50  0.15 1.00  0.35 0.65
 [8,] 0.10 -0.10  0.20 -0.40  0.30  0.15 0.35  1.00 0.20
 [9,] 0.20  0.00  0.10  0.40  0.40  0.00 0.65  0.20 1.00
```

Here is the proposal covariance matrix:

```
> print(round(covm, digits = 3))
```

```
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]  [,7]   [,8]  [,9]
 [1,] 0.150  0.017  0.000  0.044  0.016  0.000 0.049  0.023 0.046
 [2,] 0.017  0.200 -0.001  0.076 -0.025 -0.067 0.028 -0.026 0.000
 [3,] 0.000 -0.001  0.000 -0.004  0.001  0.002 0.002  0.002 0.001
 [4,] 0.044  0.076 -0.004  1.300  0.064 -0.285 0.072 -0.270 0.270
 [5,] 0.016 -0.025  0.001  0.064  0.080  0.021 0.045  0.050 0.067
 [6,] 0.000 -0.067  0.002 -0.285  0.021  0.250 0.024  0.044 0.000
 [7,] 0.049  0.028  0.002  0.072  0.045  0.024 0.100  0.065 0.122
 [8,] 0.023 -0.026  0.002 -0.270  0.050  0.044 0.065  0.350 0.070
 [9,] 0.046  0.000  0.001  0.270  0.067  0.000 0.122  0.070 0.350
```

Now we put a lower triangle of the proposal covariance matrix to the resulting list. `cov.prop` component of the resulting list is again a list, now with only one component since there is only one block of regression parameters. Observe that only lower traingle of each proposal covariance matrix must be supplied.

```
> prior.beta.mh1$blocks$cov.prop <- list()
> prior.beta.mh1$blocks$cov.prop[[1]] <- covm[lower.tri(covm, diag = TRUE)]
```

Further, we have to say that all blocks (one here) will be updated using a random-walk Metropolis algorithm (default would be Gibbs).

```
> prior.beta.mh1$type.upd <- rep("random.walk.metropolis", nblock)
```

Subsequently, we have to say how a normal proposal will be mixed with a uniform proposal when updating each block of parameters. We specify weights of a uniform component (here 0.05 for our one block). You can set each weight to zero if you do not want to mix normal and uniform proposals

```
> prior.beta.mh1$weight.unif <- rep(0.05, nblock)
```

Finally, we have to specify half of a range of a uniform component proposal for each regression parameter, i.e. we have to supply a vector of length 9.

```
> prior.beta.mh1$half.range.unif <- c(0.25, 0.25, 0.01, 1, 0.15,
+     0.25, 0.3, 1, 1)
```

Look how it looks like:

```
> print(prior.beta.mh1)

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

$var.prior
[1] 1000 1000 1000 1000 1000 1000 1000 1000 1000

$blocks
$blocks$ind.block
$blocks$ind.block[[1]]
[1] 1 2 3 4 5 6 7 8 9


$blocks$cov.prop
$blocks$cov.prop[[1]]
 [1]  0.1500000000  0.0173205081  0.0000000000  0.0441588043  0.0164316767
 [6]  0.0000000000  0.0489897949  0.0229128785  0.0458257569  0.2000000000
[11] -0.0011618950  0.0764852927 -0.0252982213 -0.0670820393  0.0282842712
[16] -0.0264575131  0.0000000000  0.0003000000 -0.0039496835  0.0007348469
[21]  0.0017320508  0.0016431677  0.0020493902  0.0010246951  1.3000000000
[26]  0.0644980620 -0.2850438563  0.0721110255 -0.2698147513  0.2698147513
[31]  0.0800000000  0.0212132034  0.0447213595  0.0501996016  0.0669328021
[36]  0.2500000000  0.0237170825  0.0443705984  0.0000000000  0.1000000000
[41]  0.0654790043  0.1216038651  0.3500000000  0.0700000000  0.3500000000



$type.upd
[1] "random.walk.metropolis"

$weight.unif
[1] 0.05

$half.range.unif
[1] 0.25 0.25 0.01 1.00 0.15 0.25 0.30 1.00 1.00
```

6.2.3 $\boldsymbol{\beta}$ updated in two blocks using random walk Metropolis-Hastings step

Finally, we show how to specify the prior `list` for the situation we wish to update $\boldsymbol{\beta}$ parameters in two blocks, first of them updated using a Gibbs step, the second one using a random walk Metropolis-Hastings step.

```
> prior.beta.mh2 <- list()
> prior.beta.mh2$mean.prior <- rep(0, nregres)
> prior.beta.mh2$var.prior <- rep(1000, nregres)
```

Definition of blocks in which beta parameters will be updated (two blocks – `beta[1:6]` and `beta[7:9]`) and the way in which they will be updated.

```
> prior.beta.mh2$blocks <- list()
> prior.beta.mh2$blocks$ind.block <- list()
> prior.beta.mh2$blocks$ind.block[[1]] <- 1:6
```

```
> prior.beta.mh2$blocks$ind.block[[2]] <- 7:9
> nblock <- length(prior.beta.mh2$blocks$ind.block)
```

Further we define a proposal covariance matrix for the second block. Note that the proposal covariance matrix for the first block does not have to be defined since the first block is updated using a Gibbs move.

`vars`	=	proposal variances for each beta[7:9] parameter
`cors`	=	lower triangle of the proposal correlation matrix
`corsm`	=	proposal correlation matrix itself
`covm`	=	proposal covariance matrix

```
> vars <- c(0.1, 0.35, 0.35)
> cors <- c(1, 0.9, 0.9, 1, 0.9, 1)
> corsm <- diag(3)
> corsm[lower.tri(corsm, diag = TRUE)] <- cors
> corsm[upper.tri(corsm, diag = FALSE)] <- t(corsm)[upper.tri(t(corsm),
+     diag = FALSE)]
> covm <- diag(sqrt(vars)) %*% corsm %*% diag(sqrt(vars))
```

Here is the proposal correlation matrix for the second block of `beta` parameters:

```
> print(corsm)
```

```
     [,1] [,2] [,3]
[1,]  1.0  0.9  0.9
[2,]  0.9  1.0  0.9
[3,]  0.9  0.9  1.0
```

Here is the proposal covariance matrix for the second block of `beta` parameters:

```
> print(covm)
```

```
          [,1]      [,2]      [,3]
[1,] 0.1000000 0.1683746 0.1683746
[2,] 0.1683746 0.3500000 0.3150000
[3,] 0.1683746 0.3150000 0.3500000
```

Now we put a lower triangle of the proposal covariance matrix to the resulting list. `cov.prop` component of the resulting list is again a list, now with two components (we have 2 blocks). Note that the first component of `cov.prop` may be set to `NULL` since we intend to use Gibbs step for the first block and no proposal covariance matrix is thus needed. Further, only lower traingle of each proposal covariance matrix must be supplied.

```
> prior.beta.mh2$blocks$cov.prop <- list()
> prior.beta.mh2$blocks$cov.prop[[1]] <- NULL
> prior.beta.mh2$blocks$cov.prop[[2]] <- covm[lower.tri(covm, diag = TRUE)]
```

Further, we have to say that the first block will be updated using the Gibbs move and the second block using random-walk Metropolis.

```
> prior.beta.mh2$type.upd <- c("gibbs", "random.walk.metropolis")
```

Subsequently, we have to say how a normal proposal will be mixed with a uniform proposal when updating each block of parameters. So we specify weights of a uniform component (here 0.05 for our second block). Note that the first component of this vector will be ignored since the first block is updated using Gibbs move.

```
> prior.beta.mh2$weight.unif <- c(0.05, 0.05)
```

Finally, we have to specify half of a range of a uniform component proposal for each regression parameter, i.e. we have to supply a vector of length 9. Again, first 6 components of this vector will be ignored since the first block is updated using the Gibbs move.

```
> prior.beta.mh2$half.range.unif <- c(0.25, 0.25, 0.01, 1, 0.15,
+     0.25, 0.3, 1, 1)
```

Look how it looks like:

```
> print(prior.beta.mh2)
```

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

$var.prior
[1] 1000 1000 1000 1000 1000 1000 1000 1000 1000

$blocks
$blocks$ind.block
$blocks$ind.block[[1]]
[1] 1 2 3 4 5 6

$blocks$ind.block[[2]]
[1] 7 8 9


$blocks$cov.prop
$blocks$cov.prop[[1]]
NULL

$blocks$cov.prop[[2]]
[1] 0.1000000 0.1683746 0.1683746 0.3500000 0.3150000 0.3500000



$type.upd
[1] "gibbs"                  "random.walk.metropolis"

$weight.unif
[1] 0.05 0.05

$half.range.unif
[1] 0.25 0.25 0.01 1.00 0.15 0.25 0.30 1.00 1.00
```

7 Prior specification for the random intercept b_i related parameters

The following `list` has only to specify two prior hyperparameters for the covariance matrix $\mathbb{D}$ (which is a scalar here, let say d) and to say how the individual random effects will be updated.

7.1 Inverse-gamma prior distribution for d

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

Hyperparameters for d are degrees of freedom τ and scale parameter $\mathbb{S} = s$. Here, $\tau = 0.002$ and $s = 0.002$ which results in inverse-gamma$(0.001, 0.001)$ prior for d. Remember that inverse-Wishart(τ, invscale $= 1/s$) = inverse-gamma($\tau/2$, scale $= s/2$) and Wishart(τ, scale $= s$) = gamma($\tau/2$, scale $= 2/s$) = gamma($\tau/2$, rate $= s/2$).

```
> prior.b.gamma$prior.D <- "inv.wishart"
> prior.b.gamma$df.D <- 0.002
> prior.b.gamma$scale.D <- 0.002
```

Type of the update of the random intercept will be Gibbs move (this could be omitted since it is a default choice).

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

Look how it looks like:

```
> print(prior.b.gamma)
```

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

$df.D
[1] 0.002

$scale.D
[1] 0.002

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

7.2 Uniform distribution for $\sqrt{d}$

This prior choice gives much better results than the previous one. A uniform prior (here Unif$(0, 100)$) is used for the standard deviation ($\sqrt{d}$) of the random intercept.

```
> prior.b.unif <- list()
> prior.b.unif$prior.D <- "sduniform"
```

Upper limit for the prior uniform distribution of $\sqrt{d}$:

```
> prior.b.unif$scale.D <- 100
```

Type of the update of individual random effects:

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

Look how it looks like:

```
> print(prior.b.unif)
```

```
$prior.D
[1] "sduniform"

$scale.D
[1] 100

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

7.3 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"
```

8 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.

8.1 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 $w_1 = 1$, $\mu_1 = 3.9$ and $\sigma_1^2 = 1.2$.

```
> init1$mixture <- c(1, 1, rep(0, prior$kmax - 1), 3.9, rep(0,
+     prior$kmax - 1), 1.2, rep(0, prior$kmax - 1))
```

Initial regression parameters $\boldsymbol{\beta}$ (from `survreg(dist = "lognormal")`):

```
> init1$beta <- c(1.1, -0.66, 0.04, -1.76, 0.94, 1.03, 0.37, 1.22,
+     0.82)
```

Initial variance d of the random intercept b_i:

```
> init1$D <- 0.16
```

Initial values of a random intercept for each of 128 patients. Here, use zero for all patients.

```
> init1$b <- rep(0, npatient)
```

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)))
```

Look how it look like:

```
> print(init1)
```

```
$iter
[1] 0

$mixture
 [1] 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
[20] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 3.9 0.0 0.0 0.0 0.0 0.0 0.0
[39] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
[58] 0.0 0.0 0.0 0.0 1.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
[77] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

$beta
[1]  1.10 -0.66  0.04 -1.76  0.94  1.03  0.37  1.22  0.82

$D
[1] 0.16

$b
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$r
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

$otherp
[1] 0.03266158

$u
 [1] 0.22426882 0.00000000 0.00000000 0.11736502 0.84354036 0.73159026
 [7] 0.33733652 0.08552319 0.95218129 0.52158337 0.05604173 0.96731274
[13] 0.95872595 0.76761140 0.17639368 0.58206190 0.96675663 0.86472658
[19] 0.01904184 0.05200478 0.97777492 0.83969431 0.08803140 0.12204775
[25] 0.35035234 0.32072050 0.44019452 0.35488914 0.91538853 0.48966645
[31] 0.20657999 0.32986081 0.72475904 0.27539691 0.13332493 0.74030394
[37] 0.14061356 0.06395621 0.10341568 0.65381864 0.13792961 0.17784764
[43] 0.79489556 0.85788704 0.33063705 0.86709928 0.93414418 0.45764323
[49] 0.59183639 0.51002633 0.59447323 0.63029168 0.60384156 0.65028995
[55] 0.86805536 0.19457940 0.81197966 0.88512073 0.47976447 0.57846512
[61] 0.33958715 0.53828747 0.41378806 0.40666027 0.33934426 0.43957083
[67] 0.15014822 0.82429020 0.30523397 0.48535860 0.48590121 0.72644045
[73] 0.66448038 0.61570053 0.51967025 0.29200648 0.66420767 0.76446373
[79] 0.59758915 0.41897332 0.26278643 0.54739382 0.72253283 0.83034265
[85] 0.82955691 0.96244185 0.70275641 0.18289551 0.26501347 0.72150569
```

8.2 Initials for chain 2

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

Iteration number of the nulth iteration:

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

Initial mixture, now with two components and $w_1 = w_2 = 0.5$, $\mu_1 = 2.5$, $\mu_2 = 5.5$, $\sigma_1^2 = \sigma_2^2 = 1$:

```
> init2$mixture <- c(2, 0.5, 0.5, rep(0, prior$kmax - 2), 2.5,
+     5.5, rep(0, prior$kmax - 2), 1, 1, rep(0, prior$kmax - 2))
```

Initial regression parameters $\boldsymbol{\beta}$ (all zeros here):

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

Initial variance d of the random intercept b_i:

```
> init2$D <- 0.05
```

Initial values of a random intercept for each of 128 patients (sample it from a normal distribution):

```
> init2$b <- rnorm(npatient, 0, sqrt(init2$D))
```

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

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

Initial component pertinence of the observations to the mixture (half observations to the first component, half to the second component):

```
> init2$r <- c(rep(1, 102), rep(2, 101))
```

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)))
```

9 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. With this specification, we store only sampled values of individual values of random effects for each patient.

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

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

```
> nsimul <- list(niter = 1000, nthin = 3, nburn = 500, nnoadapt = 0,
+     nwrite = 500)
```

For the analysis and the results presented here we used

```
> nsimul <- list(niter = 60000, nthin = 6, nburn = 30000, nnoadapt = 0, nwrite = 1000)
```

which performed 6×30 000 iterations of burn-in and additionally 6×30 000 iterations from which each 6th value was stored. Further, after cumulating 1 000 sampled values, these were stored on a disk. This would last about 15 minutes on 2GHz machine.

Define directories where first and second chain will be stored.

```
> dir.create("cgdchain1test")
> dir.create("cgdchain2test")
> dirsim1test <- paste(getwd(), "/cgdchain1test", sep = "")
> dirsim2test <- paste(getwd(), "/cgdchain2test", sep = "")
```

Run the simulation for the first and the second chain.

```
> simul1 <- bayessurvreg1(Surv(time, event) ~ trtmt + inherit +
+     age + cortico + prophy + gender + hcat + cluster(ID), random = ~1,
+     data = cgd, dir = dirsim1test, nsimul = nsimul, prior = prior,
+     prior.beta = prior.beta.gibbs, prior.b = prior.b.unif, prop.revjump = prop.revjump,
+     init = init1, store = store)

Simulation started on                       Tue Jul  5 13:44:17 2005
Iteration 500
Simulation without adaptation finished on   Tue Jul  5 13:44:18 2005   (iteration 500)
Iteration 1000
Simulation finished on                      Tue Jul  5 13:44:19 2005   (iteration 1000)

> simul2 <- bayessurvreg1(Surv(time, event) ~ trtmt + inherit +
+     age + cortico + prophy + gender + hcat + cluster(ID), random = ~1,
+     data = cgd, dir = dirsim2test, nsimul = nsimul, prior = prior,
+     prior.beta = prior.beta.gibbs, prior.b = prior.b.unif, prop.revjump = prop.revjump,
+     init = init2, store = store)

Simulation started on                       Tue Jul  5 13:44:19 2005
Iteration 500
Simulation without adaptation finished on   Tue Jul  5 13:44:20 2005   (iteration 500)
Iteration 1000
Simulation finished on                      Tue Jul  5 13:44:22 2005   (iteration 1000)
```

10 Running additional MCMC simulation to compute predictive quantities

First we have to define covariate values for which we want to do a prediction. Here, we want 8 predictive distributions, for each combination of treatment/placebo × X-linked/autosomal recessive pattern of inheritance × male/female. Remaining covariates are set to modus/mean values (age = 14.6, no corticosteroids, yes prophylactic antibiotica, hospital category = US-other). Time (response) variable is set to 1 for all 'new' patients (it does not matter what it is set to). Event variable is set to 0 for all 'new' patients (again, it does not matter, provided that `Surv` is subsequently able to create a survival object from such 'new' data).

```
> nnewpat <- 8
> nID <- 1:nnewpat
> ntrtmt <- c(0, 1, 0, 1, 0, 1, 0, 1)
> ninherit <- factor(c(0, 0, 1, 1, 0, 0, 1, 1), levels = 0:1, labels = c("X-l",
+     "AuRec"))
> nage <- rep(14.6, nnewpat)
> ncortico <- rep(0, nnewpat)
> nprophy <- rep(1, nnewpat)
> ngender <- factor(c(0, 0, 0, 0, 1, 1, 1, 1), levels = 0:1, labels = c("male",
+     "female"))
> nhcat <- factor(rep(2, nnewpat), levels = 1:4, labels = c("US-NIH",
+     "US-other", "EU-Am", "EU-other"))
> ntime <- rep(1, nnewpat)
> nevent <- rep(0, nnewpat)
```

Data frame with 'new' data:

```
> preddata <- data.frame(ID = nID, trtmt = ntrtmt, inherit = ninherit,
+     age = nage, cortico = ncortico, prophy = nprophy, gender = ngender,
+     hcat = nhcat, time = ntime, event = nevent)
> print(preddata)

  ID trtmt inherit  age cortico prophy gender     hcat time event
1  1     0     X-l 14.6       0      1   male US-other    1     0
2  2     1     X-l 14.6       0      1   male US-other    1     0
3  3     0   AuRec 14.6       0      1   male US-other    1     0
4  4     1   AuRec 14.6       0      1   male US-other    1     0
5  5     0     X-l 14.6       0      1 female US-other    1     0
6  6     1     X-l 14.6       0      1 female US-other    1     0
7  7     0   AuRec 14.6       0      1 female US-other    1     0
8  8     1   AuRec 14.6       0      1 female US-other    1     0
```

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:

```
> grid <- seq(1, 401, by = 2.5)
```

Run MCMC simulation to sample from the predictive distribution (only chain 1 will be used here):

```
> simulp <- predictive(Surv(time, event) ~ trtmt + inherit + age +
+     cortico + prophy + gender + hcat + cluster(ID), random = ~1,
+     data = preddata, dir = dirsim1test, quantile = c(0, 0.025,
+         0.5, 0.975, 1), skip = 0, by = 1, predict = predict,
+     store = store, grid = grid, Eb0.depend.mix = FALSE, type = "mixture")

Simulation started on                       Tue Jul  5 13:44:22 2005
Reading mixture files.
Reading /home/komari/win/work/papers/bayesaft/RforCRAN/cgdchain1test/beta.sim
Reading /home/komari/win/work/papers/bayesaft/RforCRAN/cgdchain1test/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. 0  Done.
 observ. 1  Done.
 observ. 2  Done.
 observ. 3  Done.
 observ. 4  Done.
 observ. 5  Done.
 observ. 6  Done.
 observ. 7  Done.

Storing quantiles.
Simulation finished on                      Tue Jul  5 13:44:27 2005
```

In a directory `./cgdchain1test` 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.

11 Drawing posterior predictive survivor/hazard curves

In this section we draw posterior predictive survivor and hazard curves for new patients with covariate combinations defined in the previous section.

Posterior predictive survivor curve and its 95% pointwise CI (1 plot per covariate combination). The result is given in Figure 3.

```
> labels <- c("Male, X-l, plcb", "Male, X-l, trt", "Male, AR, plcb",
+     "Male, AR, trt", "Female, X-l, plcb", "Female, X-l, trt",
+     "Female, AR, plcb", "Female, AR, trt")
> par(mfrow = c(4, 2))
> for (i in 1:8) {
+     gridS <- scan(paste(dirsim1, "/quantS", i, ".sim", sep = ""),
+         nlines = 1)
+     Sfun <- read.table(paste(dirsim1, "/quantS", i, ".sim", sep = ""),
+         header = TRUE)
+     rownames(Sfun) <- c("0%", "2.5%", "50%", "97.5%", "100%",
+         "mean")
+     plot(gridS, Sfun["mean", ], type = "l", lty = 1, ylim = c(0,
+         1), xlab = "Time (days)", ylab = "Survivor", bty = "n")
+     lines(gridS, Sfun["2.5%", ], lty = 2)
+     lines(gridS, Sfun["97.5%", ], lty = 2)
+     title(main = labels[i])
+ }
```

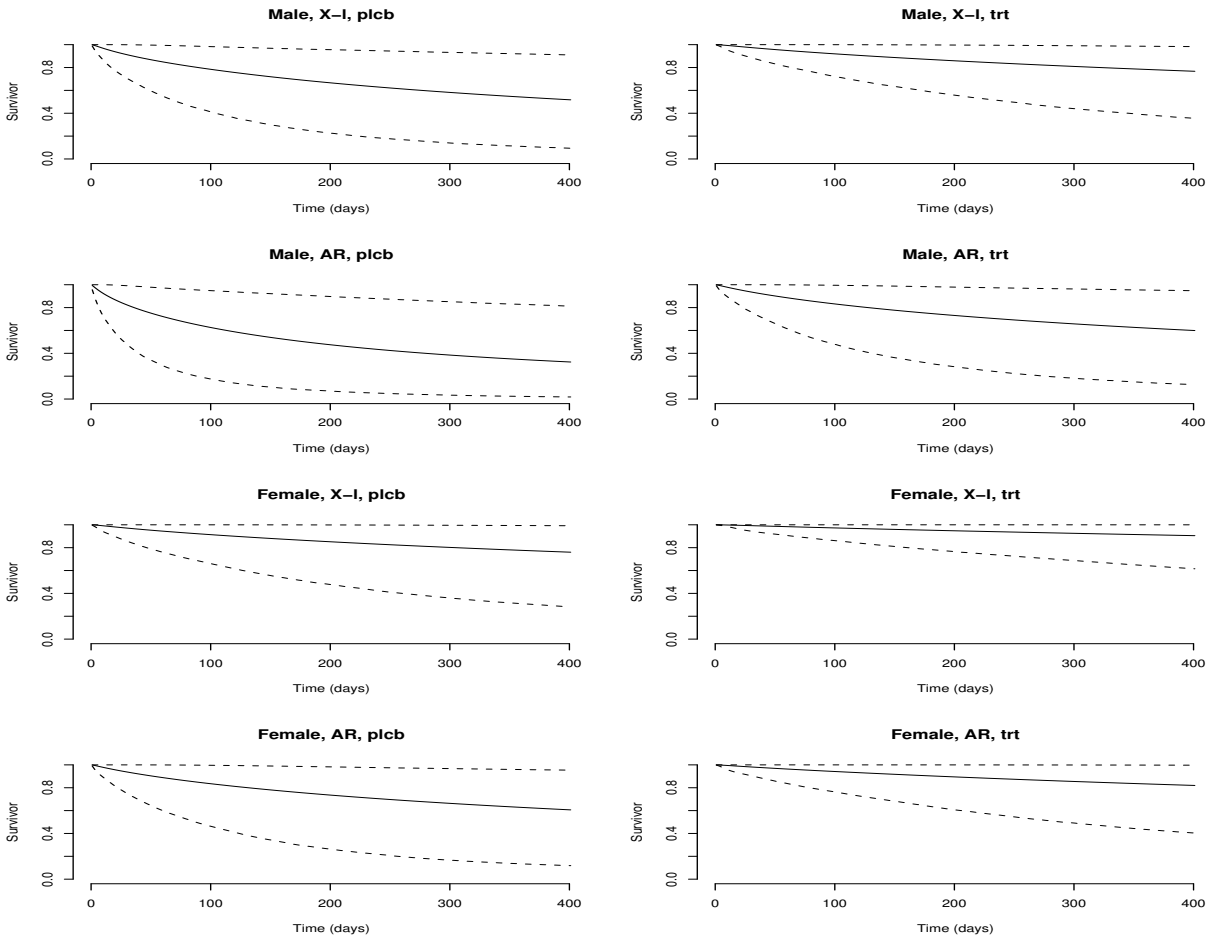
Posterior predictive hazard curve and its 95% pointwise CI (1 plot per covariate combination). The result is given in Figure 4.

```
> labels <- c("Male, X-l, plcb", "Male, X-l, trt", "Male, AR, plcb",
+     "Male, AR, trt", "Female, X-l, plcb", "Female, X-l, trt",
+     "Female, AR, plcb", "Female, AR, trt")
> par(mfrow = c(4, 2))
> for (i in 1:8) {
+     gridhaz <- scan(paste(dirsim1, "/quanthazard", i, ".sim",
+         sep = ""), nlines = 1)
+     hfun <- read.table(paste(dirsim1, "/quanthazard", i, ".sim",
+         sep = ""), header = TRUE)
+     rownames(hfun) <- c("0%", "2.5%", "50%", "97.5%", "100%",
+         "mean")
+     plot(gridhaz, hfun["97.5%", ], type = "l", lty = 2, xlab = "Time (days)",
+         ylab = "Hazard", bty = "n")
+     lines(gridhaz, hfun["mean", ], lty = 1)
+     lines(gridhaz, hfun["2.5%", ], lty = 2)
+     title(main = labels[i])
+ }
```

Posterior predictive survivor curves (1 plot per gender with 4 curves on it). The result is given in Figure 5.

```
> gg <- c("Male", "Female")
> par(mfrow = c(2, 1))
> for (j in 1:2) {
+     for (i in 1:4) {
+         gridS <- scan(paste(dirsim1, "/quantS", (j - 1) * 4 +
+             i, ".sim", sep = ""), nlines = 1)
+         Sfun <- read.table(paste(dirsim1, "/quantS", (j - 1) *
+             4 + i, ".sim", sep = ""), header = TRUE)
+         rownames(Sfun) <- c("0%", "2.5%", "50%", "97.5%", "100%",
```

Figure 3: Posterior predictive survivor curves and their 95% pointwise CI.




```
+             "mean")
+         if (i == 1)
+             plot(gridS, Sfun["mean", ], type = "l", lty = 1,
+                 ylim = c(0, 1), xlab = "Time (days)", ylab = "Survivor",
+                 bty = "n")
+         else lines(gridS, Sfun["mean", ], lty = i)
+         legend(0, 0.6, legend = c("trtmt, X-l", "trtmt, AuRec",
+             "placebo, X-l", "placebo, AuRec"), lty = c(2, 4,
+             1, 3), bty = "n")
+     }
+     title(main = gg[j])
+ }
```

Posterior predictive hazard curves (1 plot per gender with 4 curves on it). The result is given in Figure 6.

```
> gg <- c("Male", "Female")
> par(mfrow = c(2, 1))
> leg <- c(0.008, 0.004)
> ylim <- c(0.01, 0.004)
> for (j in 1:2) {
+     for (i in 1:4) {
+         gridhaz <- scan(paste(dirsim1, "/quanthazard", (j - 1) *
+             4 + i, ".sim", sep = ""), nlines = 1)
```

Figure 4: Posterior predictive hazard curves and their 95% pointwise CI

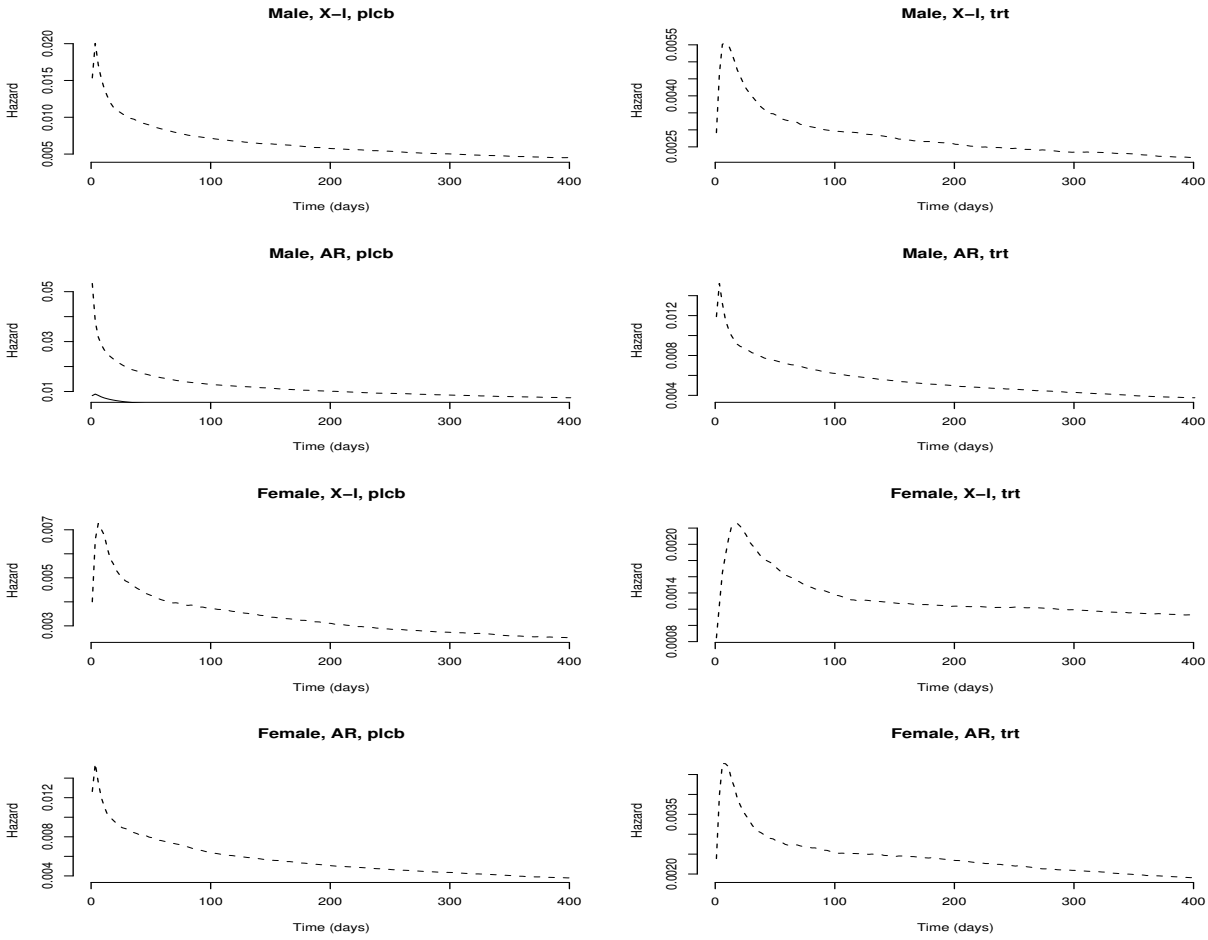


Figure 5: Posterior predictive survivor curves.

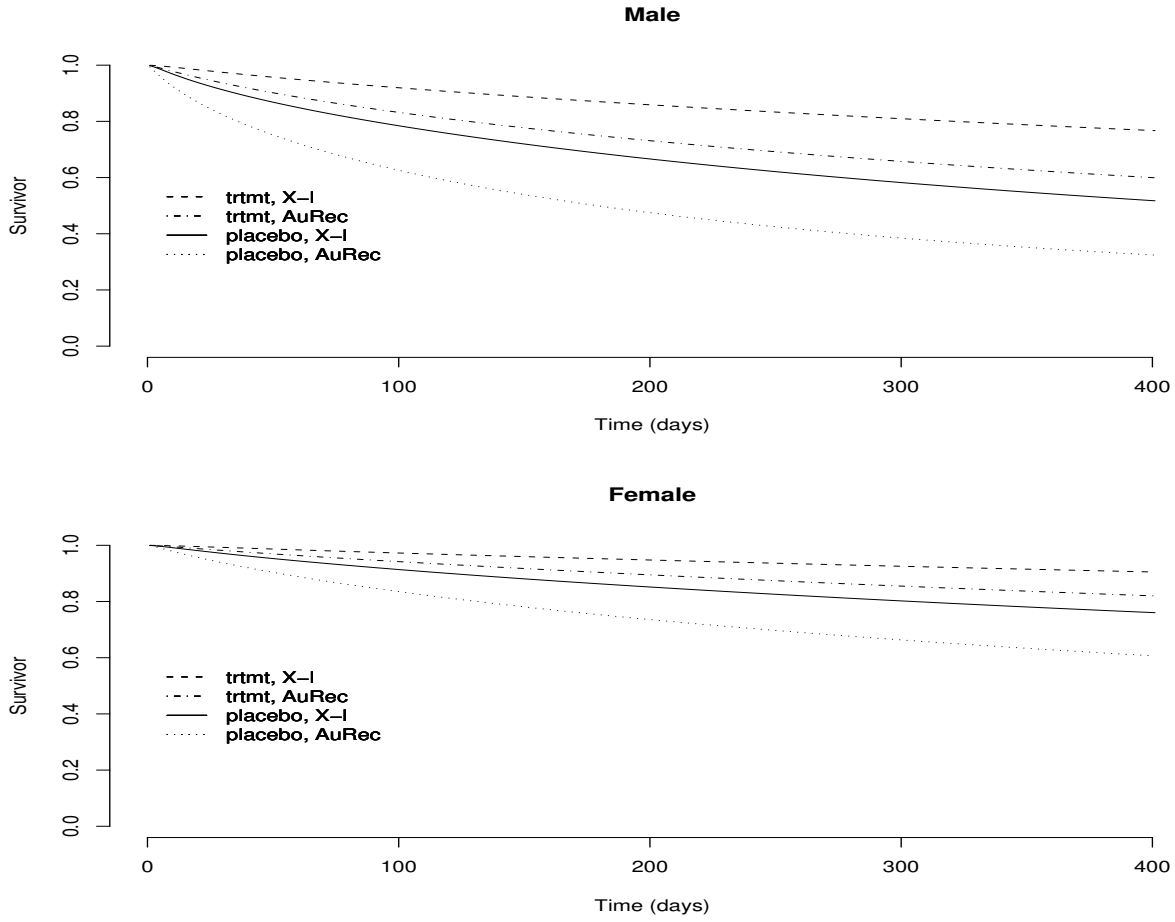
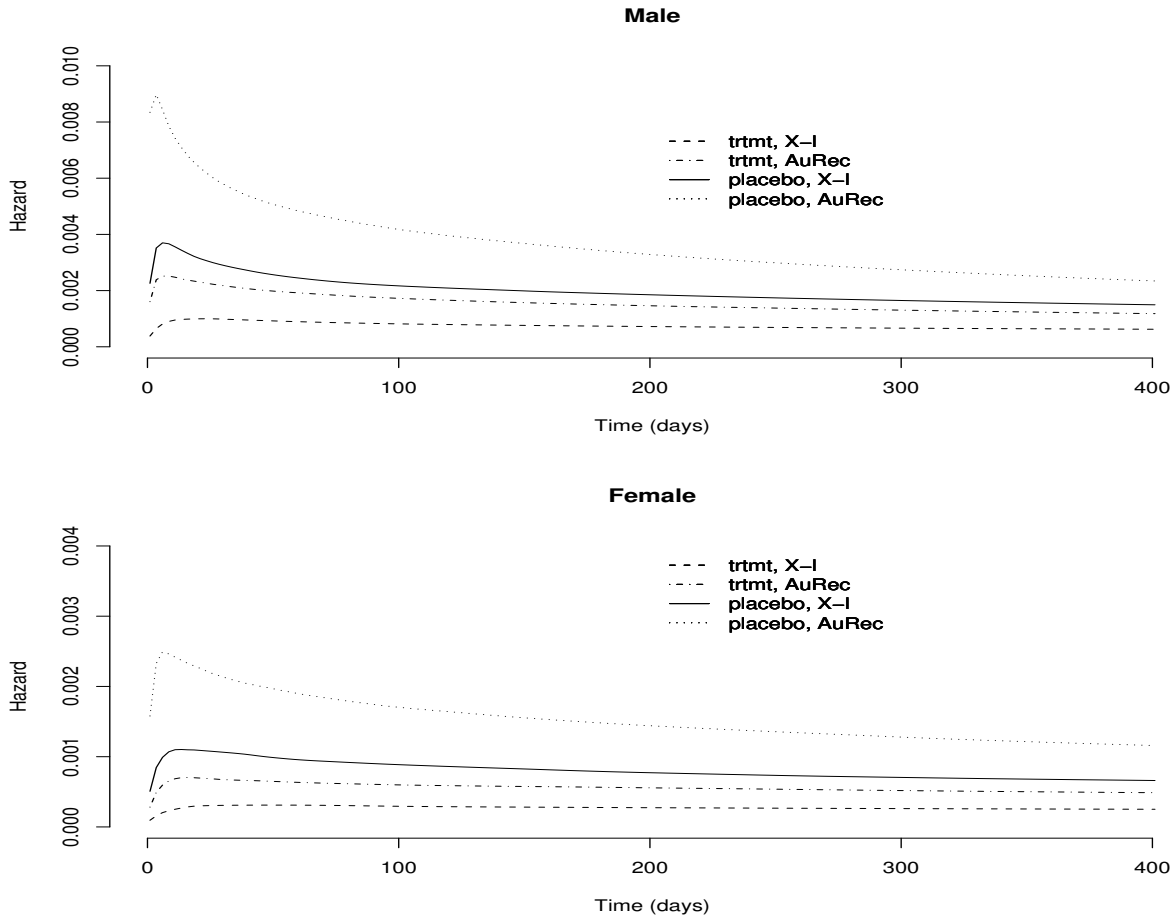


Figure 6: Posterior predictive hazard curves.




```
+         hfun <- read.table(paste(dirsim1, "/quanthazard", (j -
+             1) * 4 + i, ".sim", sep = ""), header = TRUE)
+         rownames(hfun) <- c("0%", "2.5%", "50%", "97.5%", "100%",
+             "mean")
+         if (i == 1)
+             plot(gridhaz, hfun["mean", ], type = "l", lty = 1,
+                 ylim = c(0, ylim[j]), xlab = "Time (days)", ylab = "Hazard",
+                 bty = "n")
+         else lines(gridhaz, hfun["mean", ], lty = i)
+         legend(200, leg[j], legend = c("trtmt, X-l", "trtmt, AuRec",
+             "placebo, X-l", "placebo, AuRec"), lty = c(2, 4,
+             1, 3), bty = "n")
+     }
+     title(main = gg[j])
+ }
```

12 Computing and drawing predictive error density

Here, we compute posterior standardized (zero mean, unit variance) and unstandardized predictive error densities, separately for each chain. Vector `dgrid` is a grid of values where the unstandardized density is to be evaluated, vector `dgrids` is a grid of values where the standardized density is to be evaluated.

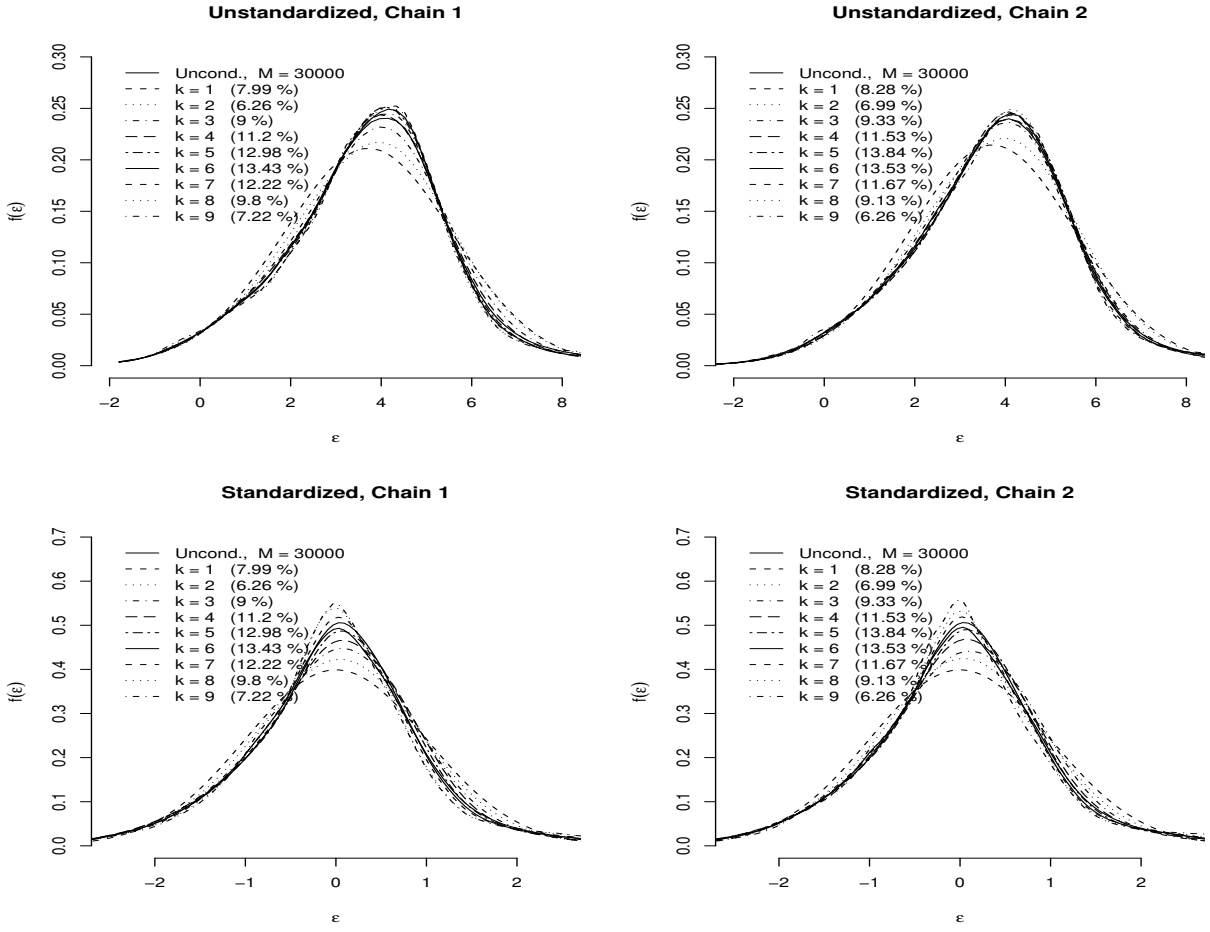
```
> dgrid <- seq(-2, 9, length = 100)
> dgrids <- seq(-3, 3, length = 100)
> dens <- list()
> for (ch in 1:2) {
+     dens[[ch]] <- bayesDensity(dir = get(paste("dirsim", ch,
+         sep = "")), grid = dgrid, stgrid = dgrids)
+ }

Reading mixture files.
Computing predictive densities.
Reading mixture files.
Computing predictive densities.
```

Now, we plot first the unstandardized predictive error densities for each chain and then the standardized ones (conditional densities given k are plotted only for $k = 1, \ldots, 9$). The result is seen in Figure 7.

```
> par(bty = "n", mfrow = c(2, 2))
> for (ch in 1:2) {
+     xlim <- c(-2, 8)
+     xleg <- -2
+     yleg <- 0.3
+     ylim <- c(0, 0.3)
+     plot(dens[[ch]], k.cond = 0:9, standard = FALSE, dim.plot = FALSE,
+         xlim = xlim, ylim = ylim, xleg = xleg, yleg = yleg, main = "")
+     title(main = paste("Unstandardized, Chain ", ch, sep = ""))
+ }
> for (ch in 1:2) {
+     xlim <- c(-2.5, 2.5)
+     xleg <- -2.5
+     yleg <- 0.7
+     ylim <- c(0, 0.7)
+     plot(dens[[ch]], k.cond = 0:9, standard = TRUE, dim.plot = FALSE,
+         xlim = xlim, ylim = ylim, xleg = xleg, yleg = yleg, main = "")
+     title(main = paste("Standardized, Chain ", ch, sep = ""))
+ }
```

Figure 7: Predictive error densities.

.



13 Predictive values of individual random effects b_i

Sampled individual random effects (from both chains):

```
> ids <- unique(cgd$ID)
> bb <- list()
> for (ch in 1:2) {
+     bb[[ch]] <- matrix(scan(paste(get(paste("dirsim", ch, sep = "")),
+         "/b.sim", sep = ""), skip = 1), ncol = 128, byrow = TRUE)
+     colnames(bb[[ch]]) <- ids
+ }
> bbs <- rbind(bb[[1]], bb[[2]])
```

Compute the posterior mean and some quantiles for each individual random effect:

```
> b.mean <- apply(bbs, 2, mean)
> b.median <- apply(bbs, 2, quantile, 0.5)
> b.low <- apply(bbs, 2, quantile, 0.025)
> b.up <- apply(bbs, 2, quantile, 0.975)
```

Sort patients according to number of events:

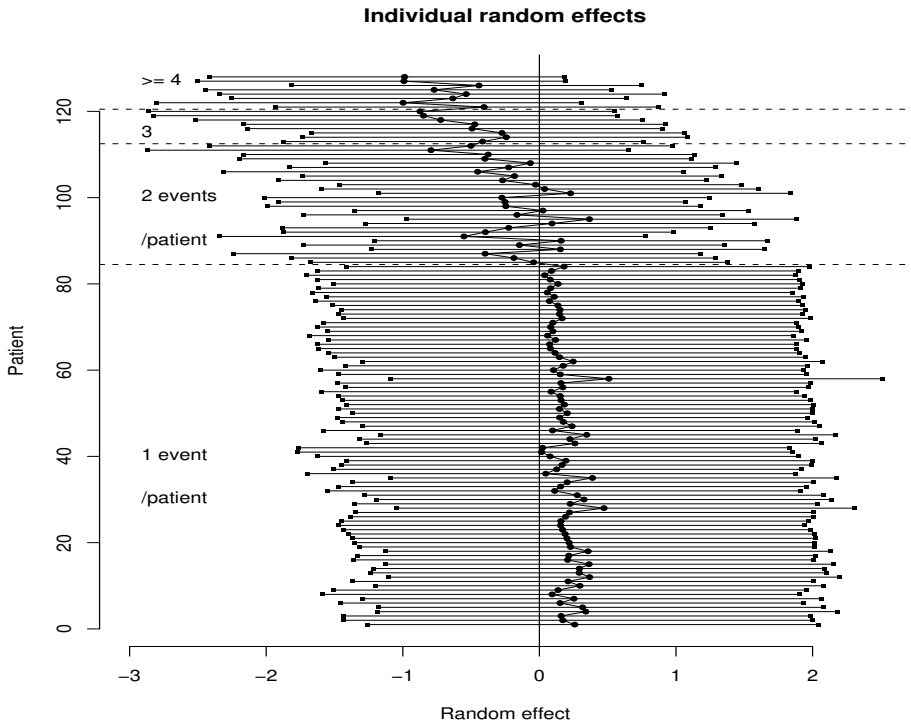
```
> n <- dim(cgd)[1]
> id1 <- cgd$ID[1:(n - 1)]
> id2 <- cgd$ID[2:n]
> difid <- c(1, id2 - id1)
> first <- difid > 0
> frevent <- table(cgd$ID)
> freqv <- as.numeric(frevent)
> frval <- data.frame(ID = cgd$ID[first], trtmt = cgd$trtmt[first],
+     freq = as.numeric(frevent), b.mean, b.median, b.low, b.up,
+     nevent = freqv)
> frval <- frval[order(frval$trtmt), ]
> frval <- frval[order(frval$freq), ]
```

Plot means and 95% CI for each individual random effect b_i (see Figure 8 for the result).

```
> par(bty = "n")
> plot(frval$b.mean, 1:128, type = "p", pch = 20, xlim = c(-3,
+     2.5), bty = "n", ylab = "Patient", xlab = "Random effect")
> lines(frval$b.mean, 1:128)
> points(frval$b.low, 1:128, pch = 15, cex = 0.5)
> points(frval$b.up, 1:128, pch = 15, cex = 0.5)
> for (pat in 1:n) {
+     lines(c(frval$b.low[pat], frval$b.up[pat]), c(pat, pat),
+         lty = 1)
+ }
> title(main = "Individual random effects")
> abline(h = 84.5, lty = 2)
> abline(h = 112.5, lty = 2)
> abline(h = 120.5, lty = 2)
> abline(v = 0, lty = 1)
> text(-3, 40, "1 event", pos = 4)
> text(-3, 30, "/patient", pos = 4)
> text(-3, 100, "2 events", pos = 4)
> text(-3, 90, "/patient", pos = 4)
> text(-3, 115, "3", pos = 4)
> text(-3, 127, ">= 4", pos = 4)
```

Figure 8: Individual random effects b_i.

.



14 Summary statistics and convergence diagnostics

Now we compute some summary statistics and perform some convergence diagnostics using the R package `coda`.

First, we load the `coda` package and say how many chains we have:

```
> library(coda)
> nchains <- 2
```

Here we compute separately chains with the standard deviation $\sqrt{d}$ of the random intercept b_i (only the variance d is stored in the file `D.sim`). Further, we compute the log-scale of the mixture (only the variance of the whole mixture is stored in the file `mixmoment.sim`).

```
> sdb <- list()
> logscale <- list()
> for (ch in 1:nchains) {
+     sdb[[ch]] <- read.table(paste(get(paste("dirsim", ch, sep = "")),
+         "/D.sim", sep = ""), header = TRUE)
+     sdb[[ch]] <- data.frame(sdb = sqrt(sdb[[ch]][, 2]))
+     logscale[[ch]] <- read.table(paste(get(paste("dirsim", ch,
+         sep = "")), "/mixmoment.sim", sep = ""), header = TRUE)
+     logscale[[ch]] <- data.frame(logscale = sqrt(logscale[[ch]][,
+         2]))
+ }
```

Using the function `files2coda` we create the CODA `mcmc` objects for each parameter and each chain:

```
> pars <- list()
> for (ch in 1:nchains) {
+     pars[[ch]] <- files2coda(files = c("beta.sim", "mixmoment.sim"),
+         data.frames = c("sdb", "logscale"), thin = 1, dir = paste(get(paste("dirsim",
+             ch, sep = ""))), chain = ch)
+ }
```

We combine both chains into a CODA `mcmc.list`:

```
> parsls <- mcmc.list(pars[[1]], pars[[2]])
> rm(list = c("pars", "sdb", "logscale"))
```

Look what are the model parameters stored in this object:

```
> dimnames(parsls[[1]])[[2]]
```

```
 [1] "trtmt"        "inheritAuRec" "age"          "cortico"      "prophy"
 [6] "genderfemale" "hcatUS.other" "hcatEU.Am"    "hcatEU.other" "k"
[11] "Intercept"    "Scale"        "sdb"          "logscale"
```

14.1 Summary statistics

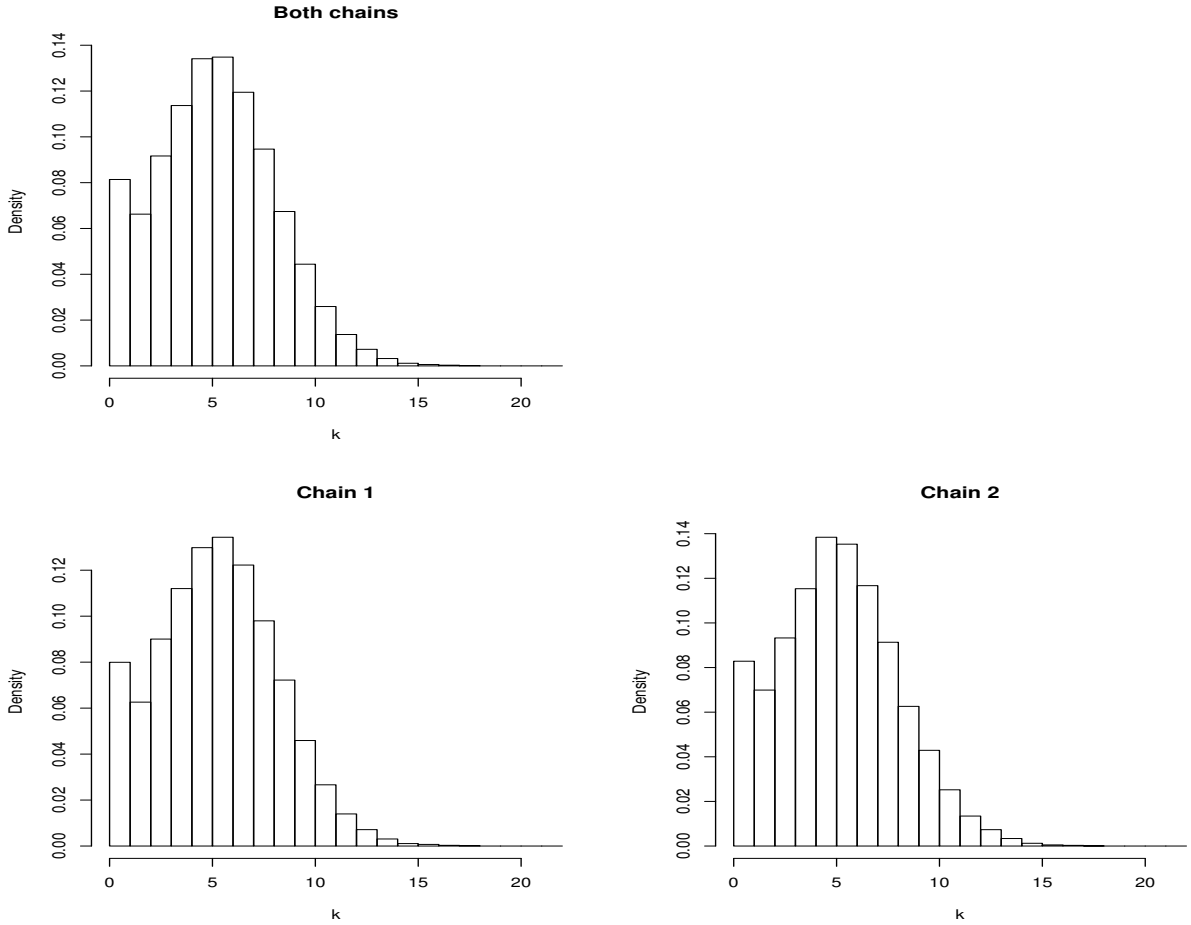
Summary statistics (separately for each chain):

```
> quant <- c(0, 0.025, 0.5, 0.75, 0.975, 1)
> means <- list()
> quantiles <- list()
> summ <- list()
> for (ch in 1:nchains) {
```

```
+     means[[ch]] <- apply(parsls[[ch]], 2, mean)
+     quantiles[[ch]] <- apply(parsls[[ch]], 2, quantile, quant)
+     summ[[ch]] <- rbind(means[[ch]], quantiles[[ch]])
+     rownames(summ[[ch]])[1] <- "mean"
+ }
> names(summ) <- paste("Chain ", 1:nchains, sep = "")
> print(summ)

$"Chain 1"
          trtmt inheritAuRec          age    cortico      prophy  genderfemale
mean   1.2952393  -0.88889973  0.047711099 -2.5305725  1.12710456  1.3426808385
0%    -0.4113581  -3.25038300 -0.044453680 -8.5209030 -1.22550700 -1.4097830000
2.5%   0.4871340  -1.81794835  0.005455738 -5.2309858  0.08478048 -0.0007541623
50%    1.2719995  -0.88578360  0.046848355 -2.4731380  1.10761450  1.3182455000
75%    1.5811632  -0.57989282  0.062110178 -1.6773263  1.47657200  1.8018465000
97.5%  2.2136023   0.03692818  0.093768835 -0.1364918  2.29062897  2.8135636250
100%   3.3180850   1.02502200  0.167860400  2.6627620  3.88840800  4.8311500000
      hcatUS.other  hcatEU.Am hcatEU.other         k Intercept     Scale
mean     0.4708446  1.5933397   1.22442635  5.757333 3.8333191 1.8571086
0%      -1.3670960 -1.6433490  -1.61149600  1.000000 0.5132017 0.9629224
2.5%    -0.4520949  0.1705688  -0.07181092  1.000000 2.1747884 1.2601308
50%      0.4598775  1.5470980   1.20281100  6.000000 3.8586200 1.7278890
75%      0.7767884  2.0873605   1.66435525  8.000000 4.3754333 2.0496230
97.5%    1.4965285  3.2851955   2.62552402 12.000000 5.4051494 3.1729197
100%     2.7890120  5.9596500   4.74614800 18.000000 6.9536050 5.8307780
             sdb logscale
mean  0.833728846 1.946205
0%    0.004319551 0.716381
2.5%  0.188781233 1.474716
50%   0.833260313 1.964337
75%   1.043242062 2.091754
97.5% 1.479499071 2.324898
100%  2.804735816 2.636969

$"Chain 2"
          trtmt inheritAuRec          age    cortico     prophy genderfemale
mean   1.3111115   -0.8809694  0.047097248 -2.5351073  1.0952944   1.39451035
0%    -0.3263018   -3.1168670 -0.051531330 -8.8917600 -1.3768960  -1.53255700
2.5%   0.5035612   -1.8043968  0.005417265 -5.3730490  0.0553078   0.06416812
50%    1.2923485   -0.8778754  0.046289855 -2.4714840  1.0748635   1.37927250
75%    1.5922275   -0.5704011  0.061379605 -1.6621527  1.4523812   1.86064125
97.5%  2.2135942    0.0338839  0.092566307 -0.0752535  2.2398131   2.82542137
100%   3.4475450    1.2762030  0.159643500  2.6708220  3.5902680   4.29455200
      hcatUS.other  hcatEU.Am hcatEU.other       k  Intercept     Scale
mean     0.4608834  1.5854185    1.2012839  5.6212 3.87034905 1.8842063
0%      -1.6901830 -1.8143200   -1.8439460  1.0000 0.01712473 0.9913276
2.5%    -0.4737526  0.1215553   -0.0693819  1.0000 2.26385775 1.2584596
50%      0.4551632  1.5531525    1.1781505  6.0000 3.86100400 1.7270735
75%      0.7772544  2.0898462    1.6224070  7.0000 4.40330250 2.0574698
97.5%    1.4478403  3.2455566    2.6241531 12.0000 5.52184587 3.4634859
100%     3.2358270  5.3361250    6.9541700 18.0000 7.49345500 6.0880480
             sdb  logscale
mean  0.81753077 1.9553976
0%    0.02315312 0.1308615
2.5%  0.20300459 1.5046122
50%   0.81327108 1.9649438
75%   1.02240110 2.0984048
97.5% 1.46854112 2.3498608
100%  2.19889268 2.7374176
```

Figure 9: Histogram of sampled k.

.



14.2 Posterior densities

Histogram of sampled k (number of mixture components). The result is shown in Figure 9.

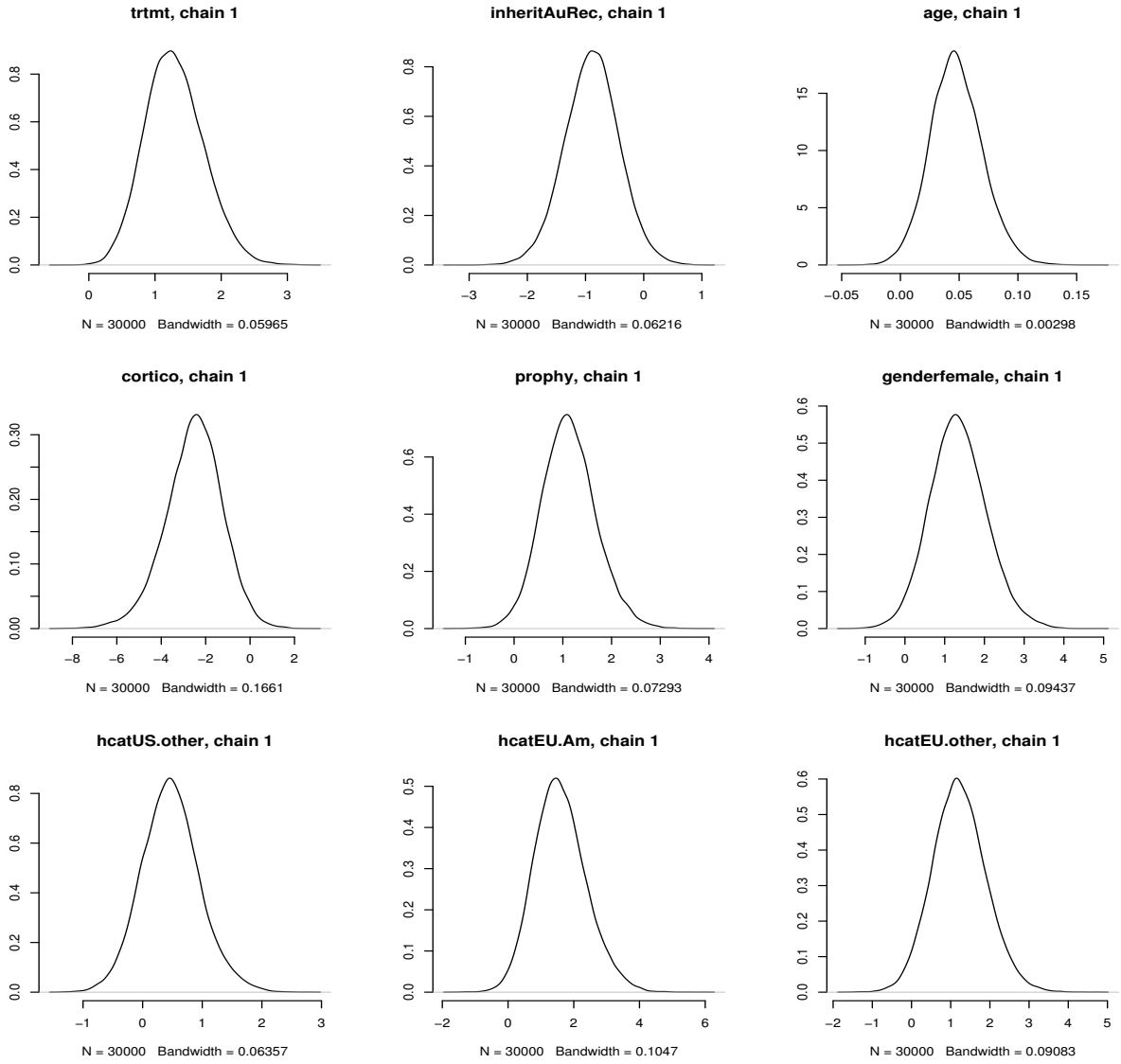
```
> par(bty = "n")
> par(mfrow = c(2, 2))
> kall <- c(parsls[[1]][, "k"], parsls[[2]][, "k"])
> hist(kall, xlab = "k", prob = TRUE, main = "Both chains", breaks = 0:22)
> plot.new()
> hist(parsls[[1]][, "k"], xlab = "k", prob = TRUE, main = "Chain 1",
+     breaks = 0:22)
> hist(parsls[[2]][, "k"], xlab = "k", prob = TRUE, main = "Chain 2",
+     breaks = 0:22)
```

Posterior densities of $\boldsymbol{\beta}$ parameters (based on the first chain only). The result is shown in Figure 10.

```
> ch <- 1
> par(bty = "n")
> par(mfrow = c(3, 3))
> for (i in 1:9) {
+     densplot(parsls[[ch]][, i], show.obs = FALSE, bty = "n")
+     title(main = paste(attr(parsls[[ch]], "dimnames")[[2]][i],
```

Figure 10: Posterior densities of $\boldsymbol{\beta}$ parameters.

.



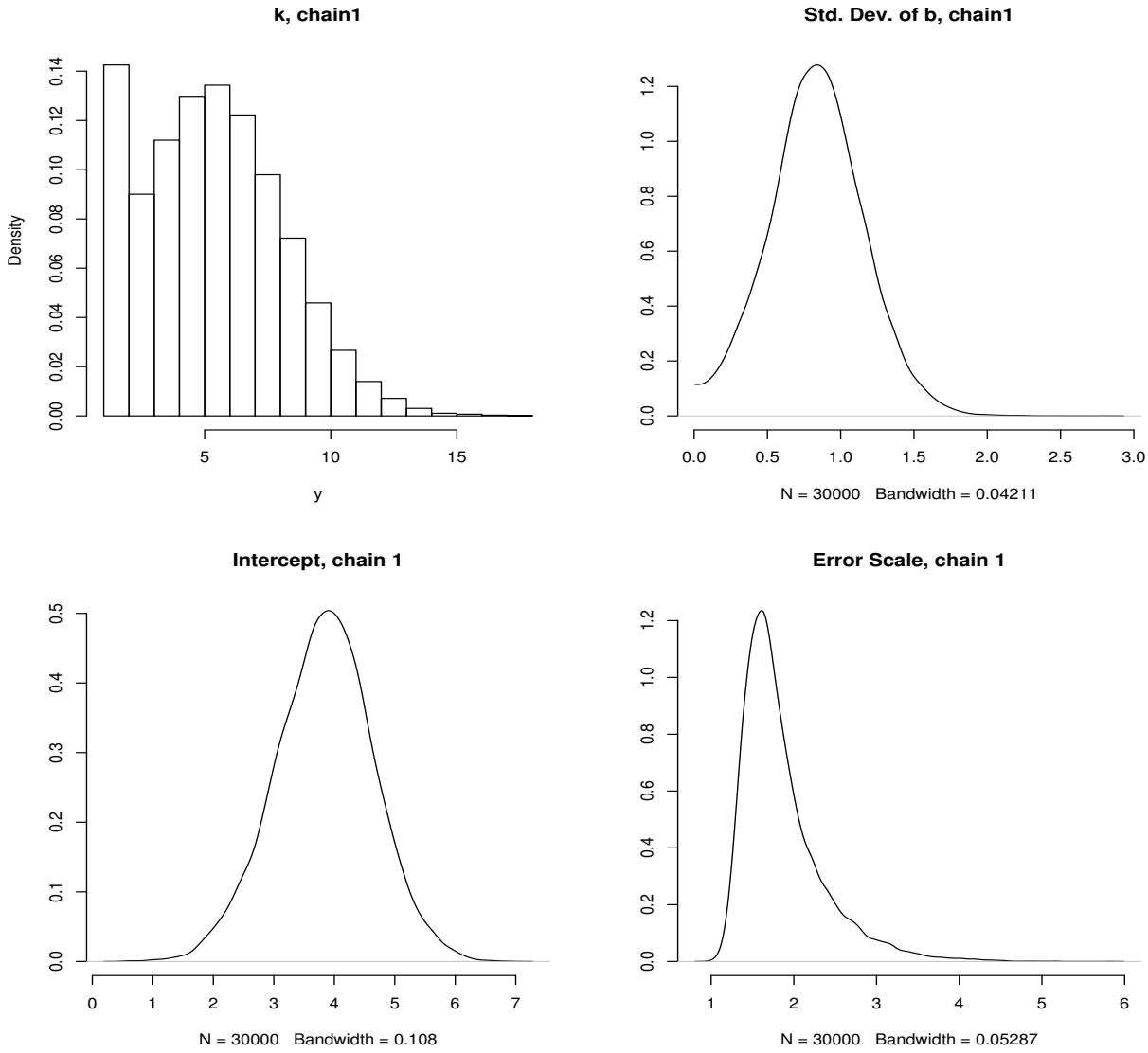

```
+         ", chain ", ch, sep = ""))
+ }
```

Posterior densities of k, standard deviation $\sqrt{d}$ of the random intercept b_i, mixture overall mean (intercept) and mixture overall standard deviation (error scale) (based on the first chain). The result is shown in Figure 11.

```
> ch <- 1
> par(mfrow = c(2, 2))
> densplot(parsls[[ch]][, "k"], show.obs = FALSE, bty = "n")
> title(main = paste("k, chain", ch, sep = ""))
> densplot(parsls[[ch]][, "sdb"], show.obs = FALSE, bty = "n")
> title(main = paste("Std. Dev. of b, chain", ch, sep = ""))
> densplot(parsls[[ch]][, "Intercept"], show.obs = FALSE, bty = "n")
> title(main = paste("Intercept, chain ", ch, sep = ""))
> densplot(parsls[[ch]][, "Scale"], show.obs = FALSE, bty = "n")
```

Figure 11: Posterior densities of some other parameters.

.




```
> title(main = paste("Error Scale, chain ", ch, sep = ""))
```

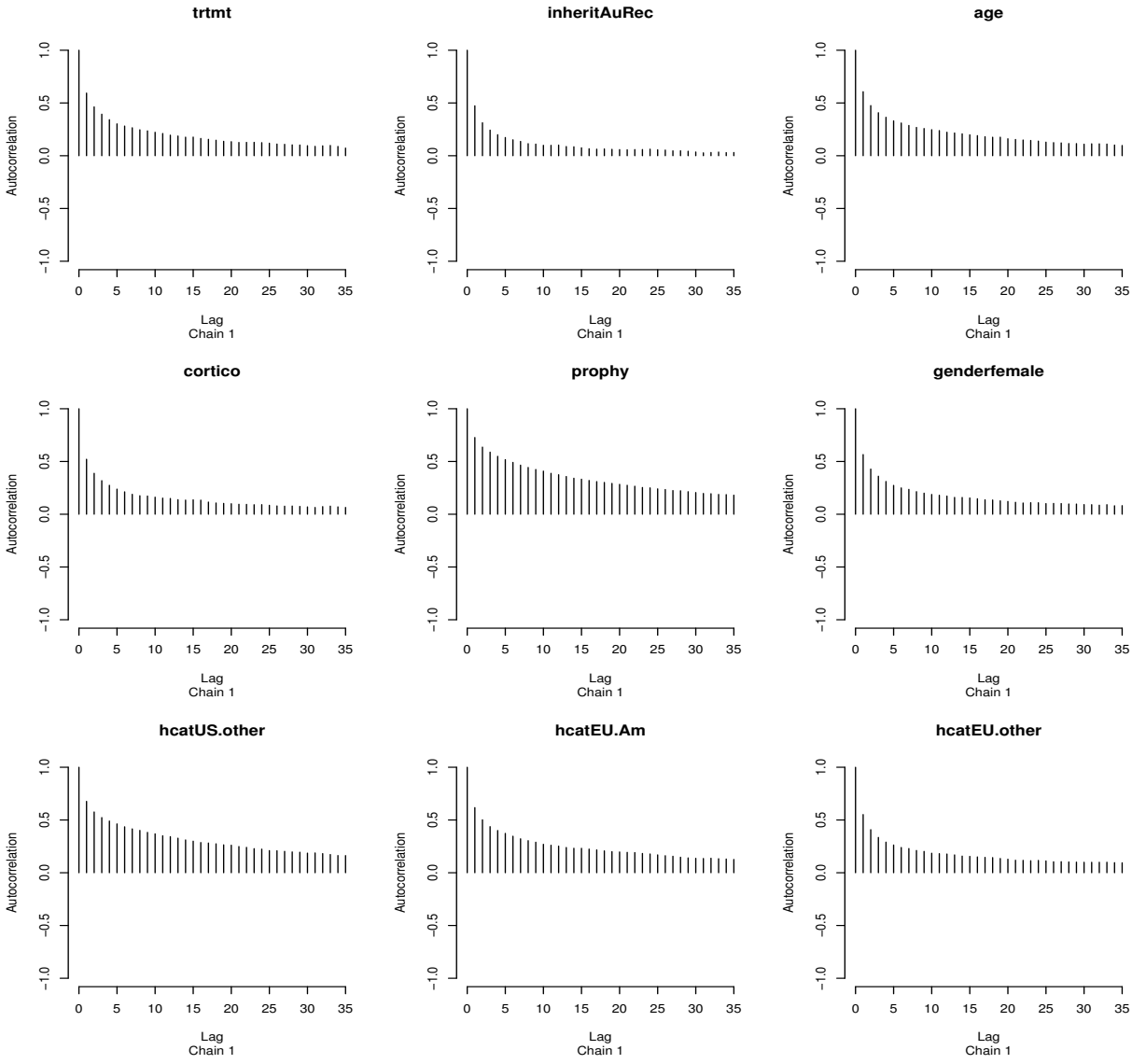
14.3 Autocorrelations

Autocorrelation plots for some parameters in the first chain (see Figure 12 and 13 for the results).

```
> ch <- 1
> par(bty = "n")
> par(mfrow = c(3, 3))
> autocorr.plot(parsls[[ch]][, 1:9], ask = FALSE, sub = paste("Chain ",
+     ch, sep = ""))

> par(bty = "n")
> par(mfrow = c(2, 2))
> plot.new()
```

Figure 12: Autocorrelation plots.

.




```
> autocorr.plot(parsls[[ch]][, 13], auto.layout = FALSE, ask = FALSE,
+     sub = paste("Chain ", ch, sep = ""), main = "sdb")
> autocorr.plot(parsls[[ch]][, 11:12], auto.layout = FALSE, ask = FALSE,
+     sub = paste("Chain ", ch, sep = ""))
```

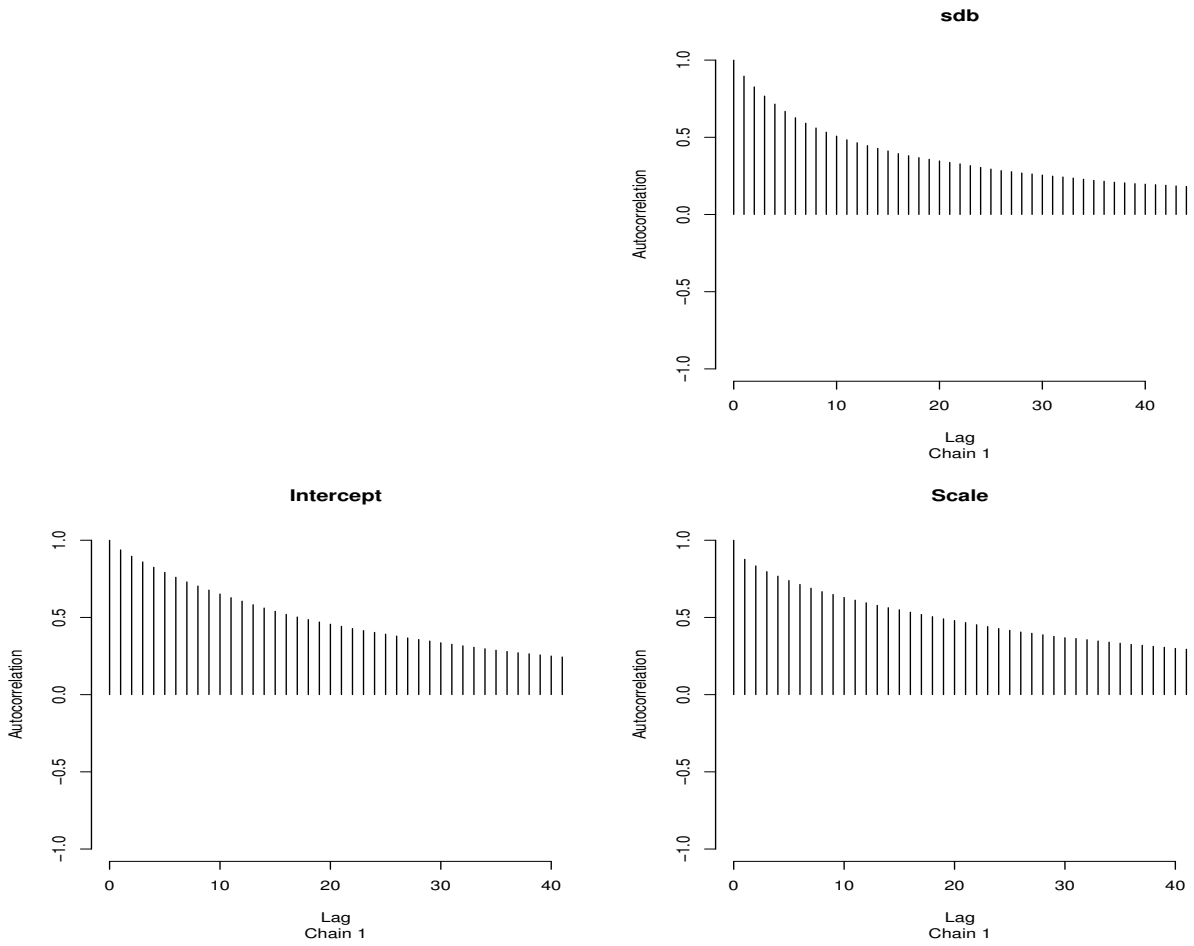
14.4 Crosscorrelations

Crosscorrelations (separately for each chain)

```
> croscor <- lapply(parsls, crosscorr)
> croscor <- lapply(croscor, round, digits = 2)
> names(croscor) <- paste("Chain ", 1:nchains, sep = "")
> print(croscor)
```

Figure 13: Autocorrelation plots.

.




```
$"Chain 1"
             trtmt inheritAuRec   age cortico prophy genderfemale hcatUS.other
trtmt         1.00         0.06  0.04    0.05   0.03         0.09         0.17
inheritAuRec  0.06         1.00 -0.14    0.31  -0.21        -0.49        -0.01
age           0.04        -0.14  1.00   -0.15   0.16         0.00         0.29
cortico       0.05         0.31 -0.15    1.00  -0.10        -0.48        -0.08
prophy        0.03        -0.21  0.16   -0.10   1.00         0.21         0.23
genderfemale  0.09        -0.49  0.00   -0.48   0.21         1.00         0.06
hcatUS.other  0.17        -0.01  0.29   -0.08   0.23         0.06         1.00
hcatEU.Am     0.06        -0.23  0.08   -0.43   0.41         0.23         0.48
hcatEU.other  0.07        -0.09  0.11   -0.06   0.10         0.05         0.45
k            -0.08         0.00 -0.01   -0.10  -0.05        -0.01         0.02
Intercept    -0.26         0.03 -0.47    0.07  -0.70        -0.15        -0.63
Scale         0.07        -0.04  0.12   -0.19   0.08         0.13         0.07
sdb           0.07        -0.01  0.09    0.01   0.11        -0.02         0.19
logscale     -0.26         0.03 -0.48    0.08  -0.70        -0.15        -0.63
             hcatEU.Am hcatEU.other     k Intercept Scale   sdb logscale
trtmt             0.06         0.07 -0.08     -0.26  0.07  0.07    -0.26
inheritAuRec     -0.23        -0.09  0.00      0.03 -0.04 -0.01     0.03
age               0.08         0.11 -0.01     -0.47  0.12  0.09    -0.48
cortico          -0.43        -0.06 -0.10      0.07 -0.19  0.01     0.08
prophy            0.41         0.10 -0.05     -0.70  0.08  0.11    -0.70
```

```
genderfemale      0.23         0.05 -0.01     -0.15  0.13 -0.02    -0.15
hcatUS.other      0.48         0.45  0.02     -0.63  0.07  0.19    -0.63
hcatEU.Am         1.00         0.30  0.04     -0.47  0.17  0.06    -0.47
hcatEU.other      0.30         1.00  0.05     -0.33  0.04  0.12    -0.33
k                 0.04         0.05  1.00      0.09  0.25 -0.03     0.09
Intercept        -0.47        -0.33  0.09      1.00  0.13  0.00     1.00
Scale             0.17         0.04  0.25      0.13  1.00 -0.09     0.12
sdb               0.06         0.12 -0.03      0.00 -0.09  1.00    -0.01
logscale         -0.47        -0.33  0.09      1.00  0.12 -0.01     1.00

$"Chain 2"
             trtmt inheritAuRec   age cortico prophy genderfemale hcatUS.other
trtmt         1.00         0.05  0.04    0.10   0.05         0.04         0.16
inheritAuRec  0.05         1.00 -0.13    0.28  -0.22        -0.45        -0.01
age           0.04        -0.13  1.00   -0.15   0.18        -0.01         0.28
cortico       0.10         0.28 -0.15    1.00  -0.08        -0.47        -0.04
prophy        0.05        -0.22  0.18   -0.08   1.00         0.20         0.24
genderfemale  0.04        -0.45 -0.01   -0.47   0.20         1.00         0.05
hcatUS.other  0.16        -0.01  0.28   -0.04   0.24         0.05         1.00
hcatEU.Am     0.05        -0.23  0.08   -0.42   0.39         0.22         0.48
hcatEU.other  0.09        -0.06  0.11   -0.01   0.12         0.03         0.47
k            -0.01         0.03 -0.01   -0.09  -0.06         0.05         0.00
Intercept    -0.27         0.05 -0.48    0.03  -0.70        -0.13        -0.62
Scale         0.02         0.00  0.11   -0.24   0.05         0.14         0.04
sdb           0.07         0.02  0.06    0.03   0.10        -0.03         0.21
logscale     -0.27         0.05 -0.48    0.03  -0.70        -0.13        -0.62
             hcatEU.Am hcatEU.other     k Intercept Scale   sdb logscale
trtmt             0.05         0.09 -0.01     -0.27  0.02  0.07    -0.27
inheritAuRec     -0.23        -0.06  0.03      0.05  0.00  0.02     0.05
age               0.08         0.11 -0.01     -0.48  0.11  0.06    -0.48
cortico          -0.42        -0.01 -0.09      0.03 -0.24  0.03     0.03
prophy            0.39         0.12 -0.06     -0.70  0.05  0.10    -0.70
genderfemale      0.22         0.03  0.05     -0.13  0.14 -0.03    -0.13
hcatUS.other      0.48         0.47  0.00     -0.62  0.04  0.21    -0.62
hcatEU.Am         1.00         0.32  0.04     -0.45  0.15  0.05    -0.46
hcatEU.other      0.32         1.00  0.02     -0.35  0.04  0.13    -0.35
k                 0.04         0.02  1.00      0.09  0.28 -0.04     0.08
Intercept        -0.45        -0.35  0.09      1.00  0.18  0.00     0.99
Scale             0.15         0.04  0.28      0.18  1.00 -0.11     0.17
sdb               0.05         0.13 -0.04      0.00 -0.11  1.00    -0.01
logscale         -0.46        -0.35  0.08      0.99  0.17 -0.01     1.00
```

14.5 Gelman-Rubin convergence diagnostics

Gelman-Rubin convergence diagnostics:

```
> gelm <- gelman.diag(parsls)
> rownames(gelm$psrf) <- dimnames(parsls[[1]])[[2]]
> print(gelm)

Potential scale reduction factors:

             Point est. 97.5% quantile
trtmt              1.00          1.00
inheritAuRec       1.00          1.00
age                1.00          1.00
cortico            1.00          1.00
prophy             1.00          1.01
```

```
genderfemale       1.00           1.01
hcatUS.other       1.00           1.00
hcatEU.Am          1.00           1.00
hcatEU.other       1.00           1.00
k                  1.00           1.00
Intercept          1.00           1.00
Scale              1.01           1.01
sdb                1.00           1.00
logscale           1.00           1.00

Multivariate psrf

1.01+0i
```

14.6 Traceplots

Traceplots for the first chain can be drawn using the following commands. As they take quite lots of memory we do not include them in this report. Observe that the function `densplot2` of the package `bayesSurv` is used.

```
> ch <- 1
> par(bty = "n")
> par(mfrow = c(2, 2))
> traceplot2(parsls[[ch]], chains = 1:3, sub = paste("Chain ",
+     ch, sep = ""))

> par(bty = "n")
> par(mfrow = c(2, 2))
> traceplot2(parsls[[ch]], chains = 4:6, sub = paste("Chain ",
+     ch, sep = ""))

> par(bty = "n")
> par(mfrow = c(2, 2))
> traceplot2(parsls[[ch]], chains = 7:9, sub = paste("Chain ",
+     ch, sep = ""))

> par(bty = "n")
> par(mfrow = c(2, 2))
> traceplot2(parsls[[ch]], chains = 10, sub = paste("Chain ", ch,
+     sep = ""))
> traceplot2(parsls[[ch]], chains = 13, sub = paste("Chain ", ch,
+     sep = ""))
> traceplot2(parsls[[ch]], chains = 11:12, sub = paste("Chain ",
+     ch, sep = ""))
```

14.7 Performance of reversible jumps

Check the performance of reversible jumps (acceptance probabilities in split-combine move and in birth-death move):

```
> mh <- list()
> averMH <- list()
> for (ch in 1:nchains) {
+     mh[[ch]] <- files2coda(files = c("MHinfo.sim"), start = 1,
+         thin = 1, dir = get(paste("dirsim", ch, sep = "")))
+     averMH[[ch]] <- apply(mh[[ch]][, c(1, 3)], 2, mean)
```

```
+ }
> for (ch in 1:nchains) {
+     cat("Chain ", ch, ":\n", sep = "")
+     print(averMH[[ch]])
+ }

Chain 1:
   accept.spl.comb accept.birth.death
        0.19944146         0.07615784
Chain 2:
   accept.spl.comb accept.birth.death
        0.19883586         0.07468886
```

Finally, we perform cleaning of generated files:

```
> files1 <- dir("./cgdchain1test")
> files2 <- dir("./cgdchain2test")
> file.remove(paste("./cgdchain1test/", files1, sep = ""))
> file.remove(paste("./cgdchain2test/", files2, sep = ""))
> file.remove("cgdchain1test")
> file.remove("cgdchain2test")
```