User's guide for R package `simMSM`

Holger Reulen,
Chair of Statistics,
University of Goettingen

June 5, 2013

The Package `simMSM` provides R functions to simulate event histories. The underlying multi state model is parameterized by possibly nonlinear baseline hazard rate functions as well as possibly nonlinear covariate effect functions. Sampling of event histories is then performed using inversion sampling on all-cause hazard rate functions.

This guide gives some background theory in section 1, prints the implemented source code in section 2 and shows the results of 10 simulation replications of an event history dataset in section 3. The simulation is based on a 4 state multi state model with 6 possible transitions and 2 covariates. The results are verified using software `BayesX` and R package `etm` in subsections 3.3 and 3.4.

Contents

1 The theory behind `simMSM` in a nutshell 2

2 Implementation of `simMSM`... 3

3 ... and how we can use it. 7

3.1 Parameterization 7
3.2 Simulation 9
3.3 Check the results using BayesX 10
3.3.1 Baseline hazard rates 11
3.3.2 Covariate effects 12
3.4 Check the results using etm 13

1 The theory behind `simMSM` in a nutshell

`simMSM` simulates event histories using inversion sampling which was outlined before for general multi state models in section 7 of Kneib and Hennerfeind (2008)[1] and described more explicitly for a single competing risks scenario in section 3.2 of Beyersmann et al. (2012)[2].

In brief inversion sampling is performed in the following way:

1. The multi state model and its stochastic behaviour is fully determined by the transition specific hazard rate functions $\lambda_{kl}(t)$. So for a specific initial state k, the integral from 0 to t over the all-cause hazard rate function $\lambda_k(t) = \sum_{l \neq k} \lambda_{kl}(t)$, the cumulative all-cause hazard rate function $\Lambda_k(t)$, specifies the distribution of sojourn times T_k in state k. We therefore perform simulation of sojourn times T_k based on the cumulative all-cause hazard rate functions.

2. For given T_k we can draw the final state j for our current transition using a one-trial multinomial distribution with probabilities p_{kj} calculated as

$$p_{kj} = \frac{\lambda_{kj}(t)}{\sum_{l \neq k} \lambda_{kl}(t)}, j = 1, \ldots, q \text{ and } j \neq k.$$

Step 1 includes the inversion sampling step which maybe needs more explananation. As the hazard rate function $\lambda_{kl}(t)$ are positive functions, the integral from 0 to t over the sums of these functions are strictly increasing and invertible. As already stated above, the cumulative all-cause hazard rate function $\Lambda_k(t)$ specifies the probability distribution function of sojourn times T_k by

$$\mathrm{F}_k(t) := \mathrm{P}(T_k \leq t) = 1 - \exp(-\Lambda_k(t))$$

Consequently the probability distribution function F_k is invertible as well.

Since I don't want to reinvent the wheel here and I could impossibly write it down any better, next up a (slightly adapted) quotation from Beyersmann et al. (2012), p. 47, that describes the remainder of the inversion sampling procedure:

"We write F_k^{-1} for the inverse of F_k and $\Lambda_k^{-1}(t)$ for the inverse of $\Lambda_k(t)$. Consider the transformed sojourn time $\mathrm{F}_k(T_k)$. The key of the inversion method is that $\mathrm{F}_k(T_k)$ is uniformly distributed on $[0, 1]$:

$$\mathrm{P}(\mathrm{F}(T) \leq u) = \mathrm{P}\left(T \leq \mathrm{F}^{-1}(u)\right) = \mathrm{F}\left(\mathrm{F}^{-1}(u)\right) = u, \ u \in [0, 1].$$

Hence, if U is a random variable with uniform distribution on $[0, 1]$, then $\mathrm{F}_k(U)$ has the same distribution as T. The inversion method works as follows:

[1] Kneib, T., Hennerfeind, A. (2008): *Bayesian semi parametric multi-state models*, Statistical Modelling, vol. 8, no. 2, p. 169-198.

[2] Beyersmann, J., Allignol, A., Schumacher, M. (2012): *Competing Risks and Multistate Models with R*, Series: Use R!, Springer, Berlin.

1. Compute $\mathrm{F}_k^{-1}(u) = \Lambda_k^{-1}(-\log(1-U))$, $U \in [0,1]$.
2. Choose a realization u that is uniformly distributed on $[0,1]$.
3. $\mathrm{F}_k^{-1}(u)$ is the desired replicate of T_k.

Sometimes, the $\lambda_{kl}(t)$ are chosen in such a manner that we do not find an explicit expression for Λ_k^{-1}. We may then use numerical inversion."

We will use a Cox-type parameterization of the transition-specific hazard rate functions in the following:

$$\lambda_{kl}(t_i) := \lambda_{kl}^{(0)}(t_i) \cdot \exp\left(\eta_i^{kl}\right),$$

with transition-specific baseline hazard rates $\lambda_{kl}^{(0)}(t_i)$ and transition- as well as individual-specific linear predictors $\eta_i^{kl} = \mathrm{f}_{x_1}^{kl}(x_{1i}) + \mathrm{f}_{x_2}^{kl}(x_{2i}) + \ldots + \mathrm{f}_{x_p}^{kl}(x_{pi})$.

f_x^{kl} may be a linear function $\mathrm{f}_x^{kl}(x_i) = \beta_x^{kl} \cdot x_i$ as well as any other function, e.g. piecewise constant, piecewise linear, trigonometrical, polynomial or any mixture of these function classes.

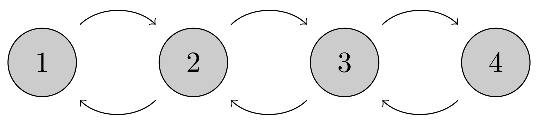
The same applies for baseline hazard rate functions, with the additional requirement that they have to be positive.

2 Implementation of `simMSM`...

```
> #setwd("P:/home/research/06_project_sim_package")
> #install.packages("simMSM_1.0.tar.gz", type="source")
> library("simMSM")
> ####################################################################
> ## function for cox-type proportional hazards parameterization ##
> ####################################################################
> hr

function (bhr, t, eta.ij)
{
    return(bhr(t) * exp(eta.ij))
}
<environment: namespace:simMSM>

> ##############################################################
> ## function calculating (possibly all-cause) hazard rate ##
> ##############################################################
> allcausehr

function (t, all.bhr, eta.ij)
{
    res <- 0
    p <- length(eta.ij)
    for (hi in 1:p) {
        res <- res + hr(all.bhr[[hi]], t, eta.ij[hi])
    }
    return(res)
}
<environment: namespace:simMSM>

> ######################################################################
> ## function calculating (possibly all cause) cumulative hazard rate ##
> ######################################################################
> cumallcausehr

function (entry, exit, all.bhr, eta.ij = eta.ij)
{
    return(integrate(allcausehr, lower = entry, upper = exit,
        all.bhr, eta.ij = eta.ij, subdivisions = 10000)$value)
}
<environment: namespace:simMSM>

> ###############################################################
> ## simulate new exit time using numerical inversion sampling ##
> ###############################################################
> f.for.uniroot

function (exit, u, entry, all.bhr, eta.ij = eta.ij)
{
    return(cumallcausehr(entry, exit, all.bhr, eta.ij = eta.ij) +
        log(1 - u))
}
<environment: namespace:simMSM>

> ##############################
> ## simulate new final state ##
> ##############################
> sim.to

function (entry.ij, from.ij, all.to.all.bhr.all.beta, eta.ij = eta.ij)
{
    exit.ij <- sim.exit(entry.ij, all.to.all.bhr.all.beta[[from.ij]]$all.bhr,
        eta.ij = eta.ij)$new.exit
    hr.at.exit.ij <- rep(NA, length(eta.ij))
    for (hi in all.to.all.bhr.all.beta[[from.ij]]$all.to) {
        hr.at.exit.ij[hi] <- hr(all.to.all.bhr.all.beta[[from.ij]]$all.bhr[[hi]],
            exit.ij, eta.ij[hi])
    }
    hr.at.exit.ij <- hr.at.exit.ij[!is.na(hr.at.exit.ij)]
    if (length(hr.at.exit.ij) > 1.5) {
        probs <- hr.at.exit.ij/sum(hr.at.exit.ij)
        to.ij <- sample(all.to.all.bhr.all.beta[[from.ij]]$all.to,
            size = 1, prob = probs)
    }
    else {
        to.ij <- as.numeric(all.to.all.bhr.all.beta[[from.ij]]$all.to)
    }
    return(list(entry.ij = entry.ij, exit.ij = exit.ij, from.ij = from.ij,
        to.ij = to.ij))
}
<environment: namespace:simMSM>

> ############################################
> ## simulate one individual event history ##
> ############################################
> sim.single.history

function (first.entry = 0, first.from, max.time, all.to.all.bhr.all.beta,
    x.i)
{
    current.exit <- current.entry <- first.entry
    current.to <- current.from <- first.from
    history.i <- NULL
    p <- length(x.i)
    while (current.entry < max.time) {
        f.x <- all.to.all.bhr.all.beta[[current.from]]$all.beta
        eta.ij <- rep(0, length(f.x))
        for (oi in 1:length(f.x)) {
            for (ii in 1:length(f.x[[oi]])) {
                f.now <- f.x[[oi]][[ii]]
                eta.ij[oi] <- eta.ij[oi] + f.now(x.i[ii])
            }
        }
        current.sim <- sim.to(current.entry, current.from, all.to.all.bhr.all.beta,
            eta.ij = eta.ij)
        history.i <- rbind(history.i, c(current.sim$entry.ij,
            current.sim$exit.ij, current.sim$from.ij, current.sim$to.ij))
        current.entry <- current.sim$exit.ij
        current.from <- current.sim$to.ij
    }
    return(history.i)
}
<environment: namespace:simMSM>

> #################################
> ## simulate n event histories  ##
> #################################
> sim.event.histories

function (n, all.to.all.bhr.all.beta, max.time = 10)
{
    hf <- function(x, k) {
        return(x[[k]])
    }
    all.possible.from.states <- as.numeric(do.call(c, lapply(all.to.all.bhr.all.beta,
        FUN = hf, k = 1)))
    all.first.from <- sample(all.possible.from.states, size = n,
        replace = T)
    p <- length(all.to.all.bhr.all.beta[[1]]$all.beta[[1]])
    all.x <- runif(n = n * p, min = -1, max = 1)
    all.x <- matrix(nrow = n, ncol = p, data = all.x)
    histories <- NULL
    for (i in 1:n) {
        history.i <- sim.single.history(first.entry = 0, first.from = all.first.from[i],
            max.time, all.to.all.bhr.all.beta, x.i = all.x[i,
                ])
        history.i <- cbind(history.i, rep(i, nrow(history.i)))
        for (x.index in 1:p) {
            history.i <- cbind(history.i, rep(all.x[i, x.index],
                nrow(history.i)))
        }
        histories <- rbind(histories, history.i)
        rm(history.i)
    }
    histories.as.list <- list(id = histories[, 5], entry = histories[,
        1], exit = histories[, 2], from = histories[, 3], to = histories[,
        4])
    for (x.index in 1:p) {
        histories.as.list[[5 + x.index]] <- histories[, 5 + x.index]
        names(histories.as.list)[5 + x.index] <- paste("x", x.index,
            sep = "")
    }
    histories <- data.frame(histories.as.list)
    rm(histories.as.list)
    return(histories)
}
<environment: namespace:simMSM>
```

3 ... and how we can use it.

We will simulate data from 4 state multi state model in the following. Only transitions between adjacent states, as illustrated in the following figure, are possible.



We assume two covariates affecting each of the transition-specific hazard rates as follows:

$$\lambda_{12}(t) := \lambda_0^{12}(t) \cdot \exp\left(\beta_{x_1}^{12} \cdot x_{1i} + \beta_{x_2}^{12} \cdot x_{2i}\right) = 0.5 \cdot \exp(0.5 \cdot x_{1i} + 1 \cdot x_{2i}),$$

$$\lambda_{21}(t) := \lambda_0^{21}(t) \cdot \exp\left(\beta_{x_1}^{21} \cdot x_{1i} + \beta_{x_2}^{21} \cdot x_{2i}\right) = 0.5 \cdot (1 + \sin(t)) \cdot \exp(-0.5 \cdot x_{1i} - 1 \cdot x_{2i}),$$

$$\lambda_{23}(t) := \lambda_0^{23}(t) \cdot \exp\left(\beta_{x_1}^{23} \cdot x_{1i} + \beta_{x_2}^{23} \cdot x_{2i}\right) = 0.5 \cdot (1 + \sin(t)) \cdot \exp(0.5 \cdot x_{1i} + 1 \cdot x_{2i}),$$

$$\lambda_{32}(t) := \lambda_0^{32}(t) \cdot \exp\left(\beta_{x_1}^{32} \cdot x_{1i} + \beta_{x_2}^{32} \cdot x_{2i}\right) = 0.5 \cdot (1 + \sin(t)) \cdot \exp(-0.5 \cdot x_{1i} - 1 \cdot x_{2i}),$$

$$\lambda_{34}(t) := \lambda_0^{34}(t) \cdot \exp\left(\beta_{x_1}^{34} \cdot x_{1i} + \beta_{x_2}^{34} \cdot x_{2i}\right) = 0.5 \cdot (1 + \sin(t)) \cdot \exp(0.5 \cdot x_{1i} + 1 \cdot x_{2i}),$$

$$\lambda_{43}(t) := \lambda_0^{43}(t) \cdot \exp\left(\beta_{x_1}^{43} \cdot x_{1i} + \beta_{x_2}^{43} \cdot x_{2i}\right) = 0.5 \cdot \exp(-0.5 \cdot x_{1i} - 1 \cdot x_{2i}).$$

In the current implementation of `simMSM`, covariates are drawn from uniform distributions on $[-1; 1]$.

3.1 Parameterization

```
> bhr.11 <- function(t){return(0*t)}
> bhr.12 <- function(t){return(0.5)}
> bhr.13 <- function(t){return(0*t)}
> bhr.14 <- function(t){return(0*t)}
> bhr.21 <- function(t){return(0.5*sin(t)+0.5)}
> bhr.22 <- function(t){return(0*t)}
> bhr.23 <- function(t){return(0.5*sin(t)+0.5)}
> bhr.24 <- function(t){return(0*t)}
> bhr.31 <- function(t){return(0*t)}
> bhr.32 <- function(t){return(0.5*sin(t)+0.5)}
> bhr.33 <- function(t){return(0*t)}
> bhr.34 <- function(t){return(0.5*sin(t)+0.5)}
> bhr.41 <- function(t){return(0*t)}
> bhr.42 <- function(t){return(0*t)}
> bhr.43 <- function(t){return(0.5)}
> bhr.44 <- function(t){return(0*t)}
> all.beta = list("to.1" = list("x.1" = NULL, "x.2" = NULL),
+                 "to.2" = list("x.1" = NULL, "x.2" = NULL),
+                 "to.3" = list("x.1" = NULL, "x.2" = NULL),
+                 "to.4" = list("x.1" = NULL, "x.2" = NULL))
> all.to.all.bhr.all.beta <- list(from.1=list(from = 1, all.to = c(2),
+                                             all.bhr = list(bhr.11, bhr.12,
+                                                            bhr.13, bhr.14),
+                                             all.beta = all.beta),
+                                 from.2=list(from = 2, all.to = c(1, 3),
+                                             all.bhr = list(bhr.21, bhr.22,
+                                                            bhr.23, bhr.24),
+                                             all.beta = all.beta),
+                                 from.3=list(from = 3, all.to = c(2, 4),
+                                             all.bhr = list(bhr.31, bhr.32,
+                                                            bhr.33, bhr.34),
+                                             all.beta = all.beta),
+                                 from.4=list(from = 4, all.to = c(3),
+                                             all.bhr = list(bhr.41, bhr.42,
+                                                            bhr.43, bhr.44),
+                                             all.beta = all.beta))
> all.to.all.bhr.all.beta$from.1$all.beta$to.1$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.1$all.beta$to.1$x.2 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.1$all.beta$to.2$x.1 <- function(x){return(0.5*x)}
> all.to.all.bhr.all.beta$from.1$all.beta$to.2$x.2 <- function(x){return(1*x)}
> all.to.all.bhr.all.beta$from.1$all.beta$to.3$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.1$all.beta$to.3$x.2 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.1$all.beta$to.4$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.1$all.beta$to.4$x.2 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.2$all.beta$to.1$x.1 <- function(x){return(-0.5*x)}
> all.to.all.bhr.all.beta$from.2$all.beta$to.1$x.2 <- function(x){return(-1*x)}
> all.to.all.bhr.all.beta$from.2$all.beta$to.2$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.2$all.beta$to.2$x.2 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.2$all.beta$to.3$x.1 <- function(x){return(0.5*x)}
> all.to.all.bhr.all.beta$from.2$all.beta$to.3$x.2 <- function(x){return(1*x)}
> all.to.all.bhr.all.beta$from.2$all.beta$to.4$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.2$all.beta$to.4$x.2 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.3$all.beta$to.1$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.3$all.beta$to.1$x.2 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.3$all.beta$to.2$x.1 <- function(x){return(-0.5*x)}
> all.to.all.bhr.all.beta$from.3$all.beta$to.2$x.2 <- function(x){return(-1*x)}
> all.to.all.bhr.all.beta$from.3$all.beta$to.3$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.3$all.beta$to.3$x.2 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.3$all.beta$to.4$x.1 <- function(x){return(0.5*x)}
> all.to.all.bhr.all.beta$from.3$all.beta$to.4$x.2 <- function(x){return(1*x)}
> all.to.all.bhr.all.beta$from.4$all.beta$to.1$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.4$all.beta$to.1$x.2 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.4$all.beta$to.2$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.4$all.beta$to.2$x.2 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.4$all.beta$to.3$x.1 <- function(x){return(0.5*x)}
> all.to.all.bhr.all.beta$from.4$all.beta$to.3$x.2 <- function(x){return(1*x)}
> all.to.all.bhr.all.beta$from.4$all.beta$to.4$x.1 <- function(x){return(0*x)}
> all.to.all.bhr.all.beta$from.4$all.beta$to.4$x.2 <- function(x){return(0*x)}
```

3.2 Simulation

```
> max.time <- 10
> N <- 100
> set.seed(13)
> d <- sim.event.histories(n=N, all.to.all.bhr.all.beta, max.time=max.time)
> head(d, n=30)

   id       entry         exit from to        x1         x2
1   1 0.000000000  0.002185974    3  2 0.2152461 -0.2880010
2   1 0.002185974  1.833642846    2  3 0.2152461 -0.2880010
3   1 1.833642846  1.891975644    3  2 0.2152461 -0.2880010
4   1 1.891975644  2.136696669    2  1 0.2152461 -0.2880010
5   1 2.136696669  3.667099687    1  2 0.2152461 -0.2880010
6   1 3.667099687  3.726154920    2  1 0.2152461 -0.2880010
7   1 3.726154920  5.360039839    1  2 0.2152461 -0.2880010
8   1 5.360039839  6.601822157    2  1 0.2152461 -0.2880010
9   1 6.601822157  6.613853327    1  2 0.2152461 -0.2880010
10  1 6.613853327  7.047962219    2  1 0.2152461 -0.2880010
11  1 7.047962219 12.730551231    1  2 0.2152461 -0.2880010
12  2 0.000000000  3.224473254    1  2 0.2073390  0.9572130
13  2 3.224473254  3.326072233    2  3 0.2073390  0.9572130
14  2 3.326072233  5.707245122    3  4 0.2073390  0.9572130
15  2 5.707245122  6.795029415    4  3 0.2073390  0.9572130
16  2 6.795029415  6.957340396    3  4 0.2073390  0.9572130
17  2 6.957340396  7.899518311    4  3 0.2073390  0.9572130
18  2 7.899518311  8.131079152    3  4 0.2073390  0.9572130
19  2 8.131079152  8.549863420    4  3 0.2073390  0.9572130
20  2 8.549863420  8.612540824    3  4 0.2073390  0.9572130
21  2 8.612540824  9.267444034    4  3 0.2073390  0.9572130
22  2 9.267444034 11.957285490    3  4 0.2073390  0.9572130
23  3 0.000000000  0.082162733    2  3 0.5500660  0.4591297
24  3 0.082162733  0.398982767    3  4 0.5500660  0.4591297
25  3 0.398982767  3.307642783    4  3 0.5500660  0.4591297
```

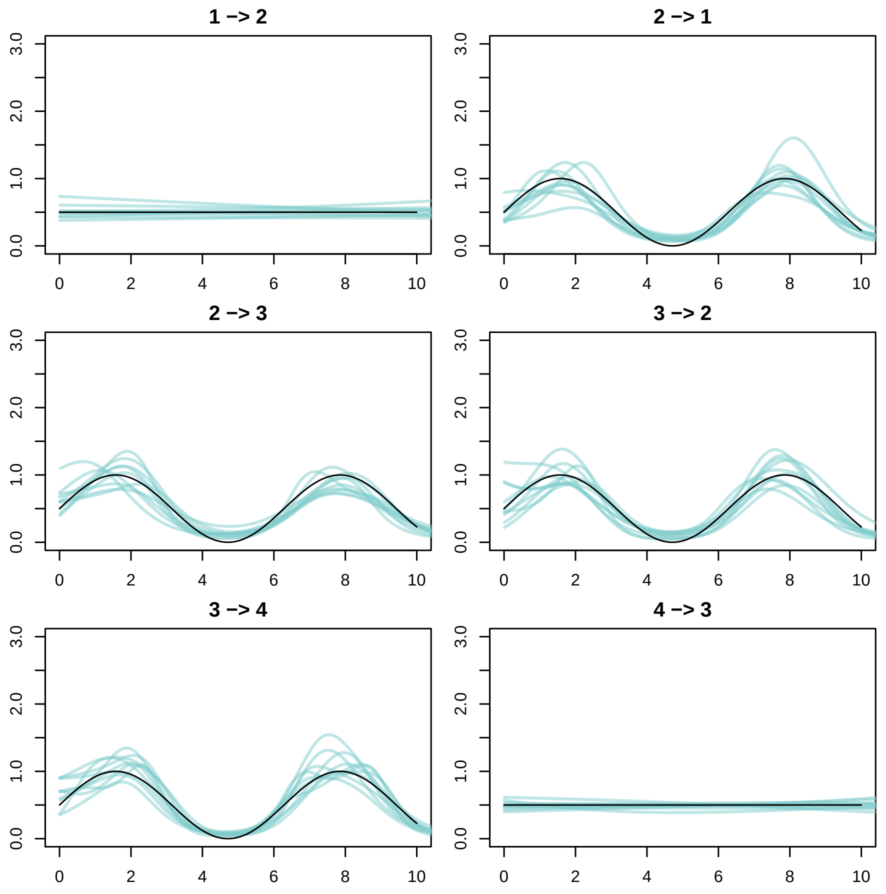
```
26   3 3.307642783  8.378749765    3  2 0.5500660  0.4591297
27   3 8.378749765  8.574201714    2  3 0.5500660  0.4591297
28   3 8.574201714  9.721422824    3  4 0.5500660  0.4591297
29   3 9.721422824 15.494700419    4  3 0.5500660  0.4591297
30   4 0.000000000  0.115538953    1  2 0.2447418  0.1778753
```

3.3 Check the results using BayesX

```
> library("BayesX")
> library("R2BayesX")
> ##############################
> ## code for bayesX program ##
> ##############################
> ## teile des bayesx programms:
> p.1 <- '
+ delimiter = ;
+ dataset dat;
+ dat.infile using P:/home/research/06_project_sim_package/shortnote_checking_results/dat
> p.2 <- '.raw;
+ remlreg simmsm;
+ simmsm.mregress t12=exit(baseline,gridchoice=all,nrknots=30,lambdastart=5)+x1+x2:
+ t21=exit(baseline,gridchoice=all,nrknots=30,lambdastart=5)+x1+x2:
+ t23=exit(baseline,gridchoice=all,nrknots=30,lambdastart=5)+x1+x2:
+ t32=exit(baseline,gridchoice=all,nrknots=30,lambdastart=5)+x1+x2:
+ t34=exit(baseline,gridchoice=all,nrknots=30,lambdastart=5)+x1+x2:
+ t43=exit(baseline,gridchoice=all,nrknots=30,lambdastart=5)+x1+x2,
+ family=multistate lefttrunc=entry state=from maxit=1000 using dat;
+ delimiter = return;'

> max.time <- 10
> N <- 100
> replicates <- 10
> set.seed(13)
> log.bhr <- beta <- vector("list", replicates)
> for(rep.index in 1:replicates){
+   d <- sim.event.histories(n=N, all.to.all.bhr.all.beta, max.time=max.time)
+   d$t12 <- as.integer(d$from == 1 & d$to == 2)
+   d$t21 <- as.integer(d$from == 2 & d$to == 1)
+   d$t23 <- as.integer(d$from == 2 & d$to == 3)
+   d$t32 <- as.integer(d$from == 3 & d$to == 2)
+   d$t34 <- as.integer(d$from == 3 & d$to == 4)
+   d$t43 <- as.integer(d$from == 4 & d$to == 3)
+   setwd("P:/home/research/06_project_sim_package/shortnote_checking_results")
+   write.table(d, row.names=FALSE, quote=FALSE,
+               file=paste("dat", rep.index, ".raw", sep=""))
+   ## BayesX:
+   setwd("C:/BayesX/commandline/")
+   writeLines(paste(p.1, "", rep.index, p.2, sep=""), con="prg")
+   system("bayesx prg")
+   setwd("C:/BayesX/commandline/output/")
+   log.bhr[[rep.index]] <- beta[[rep.index]] <- vector("list", 6)
+
+   log.bhr[[rep.index]][[1]] <- read.table("simmsm_f_exit_logbaseline.res",
+                                            header=T)
+   log.bhr[[rep.index]][[2]] <- read.table("simmsm_f_2_exit_logbaseline.res",
+                                            header=T)
+   log.bhr[[rep.index]][[3]] <- read.table("simmsm_f_3_exit_logbaseline.res",
+                                            header=T)
+   log.bhr[[rep.index]][[4]] <- read.table("simmsm_f_4_exit_logbaseline.res",
+                                            header=T)
+   log.bhr[[rep.index]][[5]] <- read.table("simmsm_f_5_exit_logbaseline.res",
+                                            header=T)
+   log.bhr[[rep.index]][[6]] <- read.table("simmsm_f_6_exit_logbaseline.res",
+                                            header=T)
+   beta[[rep.index]][[1]] <- read.table("simmsm_FixedEffects.res", header=T)
+   beta[[rep.index]][[2]] <- read.table("simmsm_FixedEffects_2.res", header=T)
+   beta[[rep.index]][[3]] <- read.table("simmsm_FixedEffects_3.res", header=T)
+   beta[[rep.index]][[4]] <- read.table("simmsm_FixedEffects_4.res", header=T)
+   beta[[rep.index]][[5]] <- read.table("simmsm_FixedEffects_5.res", header=T)
+   beta[[rep.index]][[6]] <- read.table("simmsm_FixedEffects_6.res", header=T)}
```

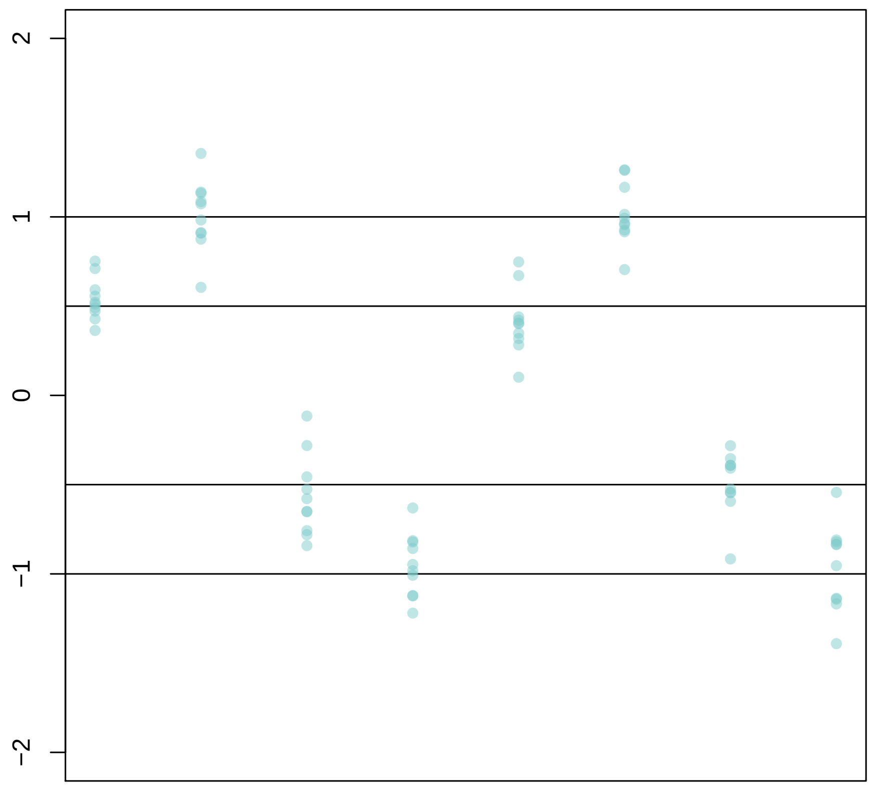
3.3.1 Baseline hazard rates

```
> ## plot baseline hazard rates:
> main.titles <- c("1 -> 2", "2 -> 1", "2 -> 3", "3 -> 2", "3 -> 4", "4 -> 3")
> par(mfrow=c(3, 2),mar=c(2, 2, 2, 1), oma=c(0, 0, 0, 0))
> for(i in 1:6){
+   plot(1, 1, ylim=c(0, 3), xlim=c(0, 10), type="n", ylab="Baseline hazard rate",
+        xlab="Time", main=main.titles[i])
+   for(rep.index in 1:replicates){
+     hilf <- beta[[rep.index]][[i]]
+     intercept <- hilf$pmode[1]
+     hilf <- log.bhr[[rep.index]][[i]]
+     lines(hilf$exit, exp(hilf$pmode+intercept), col=rgb(0.5, 0.8, 0.8, alpha=0.5),
+           lwd=2)}
+   x <- seq(0, 10, length=100)
+   if(i == 1){lines(x, apply(as.matrix(x), MAR=1, FUN=bhr.12))}
+   if(i == 2){lines(x, bhr.21(x))}
+   if(i == 3){lines(x, bhr.23(x))}
+   if(i == 4){lines(x, bhr.32(x))}
+   if(i == 5){lines(x, bhr.34(x))}
+   if(i == 6){lines(x, apply(as.matrix(x), MAR=1, FUN=bhr.43))}}
```



3.3.2 Covariate effects

```
> betas <- c(beta[[1]][[1]]$pmode[2:3], beta[[1]][[2]]$pmode[2:3],
+           beta[[1]][[3]]$pmode[2:3], beta[[1]][[4]]$pmode[2:3])
> par(mfrow=c(1, 1), mar=c(2, 2, 2, 1), oma=c(0, 0, 0, 0))
> plot(1, 1, type="n", xlim=c(1, 8), ylim=c(-2, 2), xaxt="n")
> abline(h=c(1, 0.5, -0.5, -1))
> for(i in 1:10){
+   betas <- c(beta[[i]][[1]]$pmode[2:3], beta[[i]][[2]]$pmode[2:3],
+             beta[[i]][[3]]$pmode[2:3], beta[[i]][[4]]$pmode[2:3])
+   points(1:8, betas, pch=16, col=rgb(0.5, 0.8, 0.8, alpha=0.5))}
```

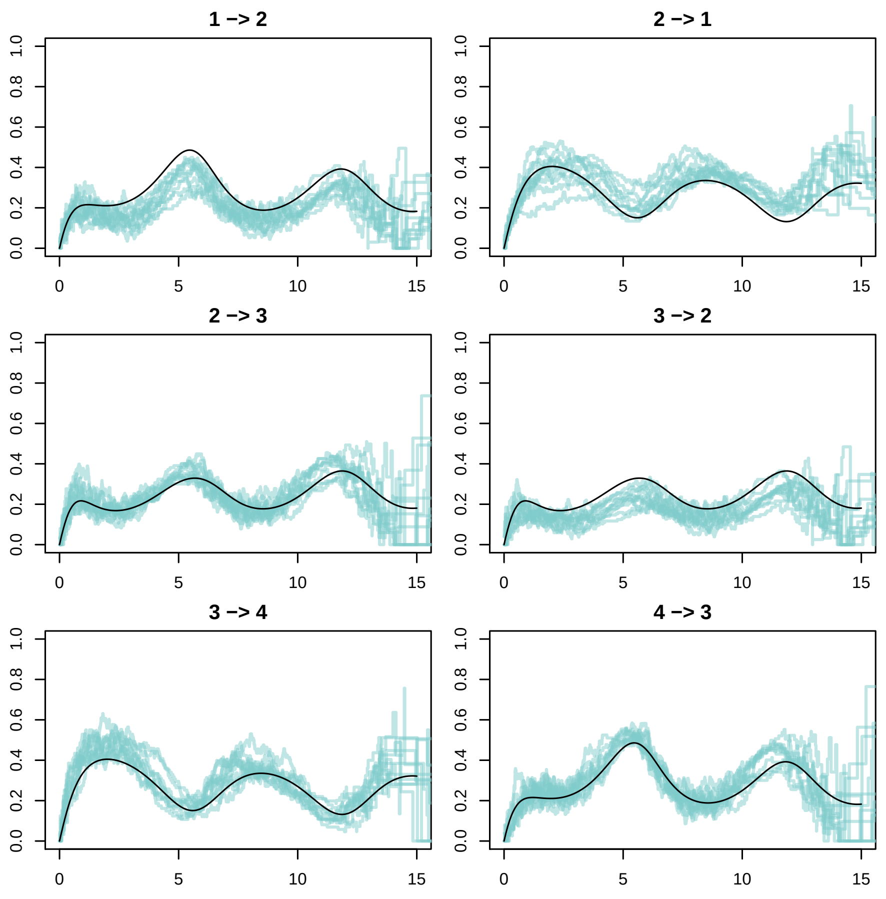


3.4 Check the results using etm

```
> library("etm")
> tra <- matrix(nrow=4, ncol=4, F)
> tra[1, 2] <- tra[2, 3] <- T
> tra[2, 1] <- tra[3, 2] <- T
> tra[3, 4] <- tra[4, 3] <- T

> etm.res <- vector("list", 10)
> d <- read.table("dat1.raw", header=T)
> etm.res[[1]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> d <- read.table("dat2.raw", header=T)
> etm.res[[2]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> d <- read.table("dat3.raw", header=T)
> etm.res[[3]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> d <- read.table("dat4.raw", header=T)
> etm.res[[4]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> d <- read.table("dat5.raw", header=T)
> etm.res[[5]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> d <- read.table("dat6.raw", header=T)
> etm.res[[6]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> d <- read.table("dat7.raw", header=T)
> etm.res[[7]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> d <- read.table("dat8.raw", header=T)
> etm.res[[8]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> d <- read.table("dat9.raw", header=T)
> etm.res[[9]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> d <- read.table("dat10.raw", header=T)
> etm.res[[10]] <- etm(d, tra=tra, state.names=1:4, cens.name=NULL, s=0)
> ## theoretical transition matrix:
> P <- matrix(nrow=4, ncol=4, data=0)
> gridlength <- 200
> times <- seq(0, 15, length=gridlength)
> A <- P.t <- vector("list", gridlength)
> for(i in 1:gridlength){
+   A[[i]] <- P
+   A[[i]][1, 1] <- -0.5*times[i]
+   A[[i]][1, 2] <- 0.5*times[i]
+   A[[i]][2, 1] <- 0.5*times[i]-0.5*cos(times[i])
+   A[[i]][2, 3] <- 0.5*times[i]-0.5*cos(times[i])
+   A[[i]][2, 2] <- (A[[i]][2, 1]+A[[i]][2, 3])*(-1)
+   A[[i]][3, 2] <- 0.5*times[i]-0.5*cos(times[i])
+   A[[i]][3, 4] <- 0.5*times[i]-0.5*cos(times[i])
+   A[[i]][3, 3] <- (A[[i]][3, 2]+A[[i]][3, 4])*(-1)
+   A[[i]][4, 3] <- 0.5*times[i]
+   A[[i]][4, 4] <- -0.5*times[i]}
> P.t[[1]] <- diag(4)
> for(i in 2:gridlength){
+   P.t[[i]] <- P.t[[i-1]]%*%(diag(4)+(A[[i]]-A[[i-1]]))}
> hf <- function(mat, k, l){return(mat[k, l])}
> P.12 <- do.call(c, lapply(P.t, FUN=hf, k=1, l=2))
> P.21 <- do.call(c, lapply(P.t, FUN=hf, k=2, l=1))
> P.23 <- do.call(c, lapply(P.t, FUN=hf, k=2, l=3))
> P.32 <- do.call(c, lapply(P.t, FUN=hf, k=3, l=2))
> P.34 <- do.call(c, lapply(P.t, FUN=hf, k=3, l=4))
> P.43 <- do.call(c, lapply(P.t, FUN=hf, k=4, l=3))
```

Plot empirical transition probabilities:

```
> par(mfrow=c(3, 2),mar=c(2, 2, 2, 1), oma=c(0, 0, 0, 0))
> plot(1, 1, type="n", xlim=c(0, 15), ylim=c(0, 1), main="1 -> 2")
> for(i in c(1:10)){lines(etm.res[[i]], tr.choice="1 2",
+                           rgb(0.5, 0.8, 0.8, alpha=0.5), lwd=2)}
> lines(times, P.12)
> plot(1, 1, type="n", xlim=c(0, 15), ylim=c(0, 1), main="2 -> 1")
> for(i in c(1:10)){lines(etm.res[[i]], tr.choice="2 1",
+                         rgb(0.5, 0.8, 0.8, alpha=0.5), lwd=2)}
> lines(times, P.21)
> plot(1, 1, type="n", xlim=c(0, 15), ylim=c(0, 1), main="2 -> 3")
> for(i in c(1:10)){lines(etm.res[[i]], tr.choice="2 3",
+                         rgb(0.5, 0.8, 0.8, alpha=0.5), lwd=2)}
> lines(times, P.23)
> plot(1, 1, type="n", xlim=c(0, 15), ylim=c(0, 1), main="3 -> 2")
> for(i in c(1:10)){lines(etm.res[[i]], tr.choice="3 2",
+                         rgb(0.5, 0.8, 0.8, alpha=0.5), lwd=2)}
> lines(times, P.32)
> plot(1, 1, type="n", xlim=c(0, 15), ylim=c(0, 1), main="3 -> 4")
> for(i in c(1:10)){lines(etm.res[[i]], tr.choice="3 4",
+                         rgb(0.5, 0.8, 0.8, alpha=0.5), lwd=2)}
> lines(times, P.34)
> plot(1, 1, type="n", xlim=c(0, 15), ylim=c(0, 1), main="4 -> 3")
> for(i in c(1:10)){lines(etm.res[[i]], tr.choice="4 3",
+                         rgb(0.5, 0.8, 0.8, alpha=0.5), lwd=2)}
> lines(times, P.43)
```


1 –> 2
2 –> 1
2 –> 3
3 –> 2
3 –> 4
4 –> 3
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15